Wissenschaftliche Methodik – Übung Quantitative Datenanalyse mit R

Diverse

Stand SoSe 2017

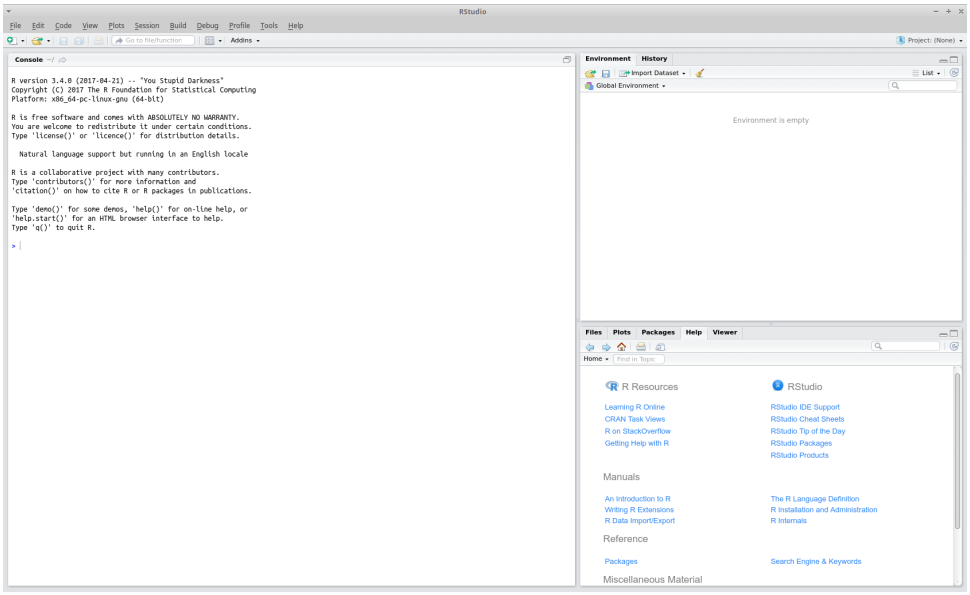
Inhaltsverzeichnis

Kapitel 0: Erste Schritte in R

Hinweise

R ist der Name eines Programms für Statistik und Datenanalyse, RStudio ist eine komfortable Entwicklungsumgebung für R.

Nach dem Start von RStudio erscheint folgender Bildschirm, wobei die Version neuer sein kann.



Links, in der *Console* werden die Befehle eingegeben. Rechts oben können Sie z. B. die Daten, aber auch andere Objekte, mit denen Sie arbeiten, betrachten, auch die Historie der Befehle wird dort angezeigt. Rechts unten können Sie u. a. Dateien und Abbildungen auswählen, aber auch Hilfeseiten und Tipps betrachten.

Wir werden zunächst in der Konsole arbeiten.

Ein paar Anmerkungen vorweg:

- R unterscheidet zwischen Groß- und Kleinbuchstaben, d.h. `Oma` und `oma` sind zwei verschiedene Dinge für R!
- R verwendet den Punkt `.` als Dezimaltrennzeichen
- Fehlende Werte werden in R durch `NA` kodiert
- Kommentare werden mit dem Rautezeichen `#` eingeleitet; der Rest der Zeile von von R dann ignoriert.
- R wendet Befehle direkt an
- R ist objektorientiert, d. h. dieselbe Funktion hat evtl. je nach Funktionsargument unterschiedliche Rückgabewerte
- Hilfe zu einem Befehl erhält man über ein vorgestelltes Fragezeichen `?`
- Zusätzliche Funktionalität kann über Zusatzpakete hinzugeladen werden. Diese müssen ggf. zunächst installiert werden
- Mit der Pfeiltaste nach oben können Sie einen vorherigen Befehl wieder aufrufen

- Sofern Sie das Skriptfenster verwenden: einzelne Befehle aus dem Skriptfenster in R Studio können Sie auch mit `Str` und `Enter` an die Console schicken

R als Taschenrechner

Auch wenn Statistik nicht Mathe ist, so kann man mit R auch rechnen. Geben Sie zum Üben die Befehle in der R Konsole hinter der Eingabeaufforderung > ein und beenden Sie die Eingabe mit `Return` bzw. `Enter`.

```
4+2
```

```
## [1] 6
```

Das Ergebnis wird direkt angezeigt. Bei

```
x <- 4+2
```

erscheint zunächst kein Ergebnis. Über `<-` wird der Variable `x` der Wert `4+2` zugewiesen. Wenn Sie jetzt

```
x
```

eingeben, wird das Ergebnis

```
## [1] 6
```

angezeigt. Sie können jetzt auch mit `x` weiterrechnen.

```
x/4
```

```
## [1] 1.5
```

Vielleicht fragen Sie sich was die `[1]` vor dem Ergebnis bedeutet. R arbeitet vektororientiert, und die `[1]` zeigt an, dass es sich um das erste (und hier auch letzte) Element des Vektors handelt.

R zur Datenanalyse

Wir wollen R aber als Tool zur Datenanalyse verwenden. Daher müssen wir zunächst Daten einlesen.

Zunächst laden wir die Daten als `csv` Datei herunter

```
download.file("https://goo.gl/whKjnl", destfile = "tips.csv")
```

Der Inhalt der Datei ist jetzt als Tabelle `tips` in R verfügbar.

Hier können Sie mehr über die Daten erfahren.

Das Einlesen von `csv` Dateien aus dem Arbeitsverzeichnis in R kann erfolgen über

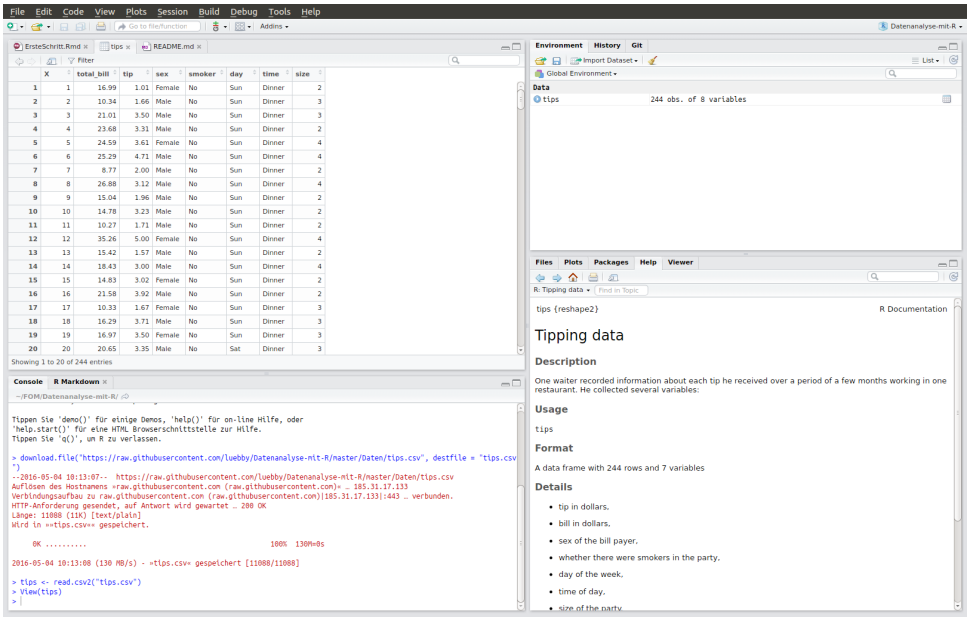
```
tips <- read.csv2("tips.csv")
```

Wo das lokale Verzeichnis (“working directory”) ist, können Sie über

```
getwd()
```

erfahren.

Der Datensatz `tips` taucht jetzt im `Enviroment` Fenster rechts oben in RStudio auf. Durch Klicken auf den Namen können Sie diese betrachten.



Alternativ können Sie Daten in RStudio komfortabel mit dem Button `Import Dataset` (im Fenster `Environment` oder über das Menü `File`) öffnen.

Erste Analyse des tips Datensatzes

Dieser Datensatz aus

Bryant, P. G. and Smith, M (1995) Practical Data Analysis: Case Studies in Business Statistics. Homewood, IL: Richard D. Irwin Publishing

enthält Trinkgelddaten. Diese sind in tabellarischer Form dargestellt, d. h. üblicherweise, dass die Beobachtungen zeilenweise untereinander stehen, die einzelnen Variablen spaltenweise nebeneinander. In R heißen solche Daten *data frame*. Um einen ersten Überblick über die verschiedenen Variablen zu erhalten geben wir den Befehl `str()` ein:

```
str(tips)
```

```
## 'data.frame':    244 obs. of  7 variables:
##  $ total_bill: num  17 10.3 21 23.7 24.6 ...
##  $ tip       : num  1.01 1.66 3.5 3.31 3.61 4.71 2 3.12 1.96 3.23 ...
##  $ sex       : Factor w/ 2 levels "Female","Male": 1 2 2 2 1 2 2 2 2 2 ...
##  $ smoker    : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ day       : Factor w/ 4 levels "Fri","Sat","Sun",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ time      : Factor w/ 2 levels "Dinner","Lunch": 1 1 1 1 1 1 1 1 1 1 ...
```

```
##  $ size       : int  2 3 3 2 4 4 2 4 2 2 ...
```

Dieser enthält also 244 Zeilen (Beobachtungen) und 7 Spalten (Variablen). Alternativ kann man diese Information auch über

```
dim(tips)
```

```
## [1] 244   7
```

erhalten.

Numerische (metrische) Variablen sind in R in der Regel vom Typ `numeric` (stetig) oder `int` (Ganze Zahlen), kategorielle (nominale, ordinale) Variablen vom Typ `factor` (bei ordinal: Option `ordered = TRUE`) oder `character`. `str()` und `dim()` sind erste Befehle, d. h., Funktionen in R, denen in der Klammer das jeweilige Funktionsargument übergeben wird.

```
head(tips) # Obere Zeilen
tail(tips) # Untere Zeilen
```

Ermöglichen ebenfalls einen Einblick über die Daten. Der Befehl

```
names(tips)
```

gibt die Variablennamen zurück. Mit Hilfe des `$` Operators kann auf einzelne Variablen eines Dataframes zugegriffen werden:

```
tips$sex
```

erhalten Sie bspw. das Geschlecht des Rechnungszahlers.

Übung: Lassen Sie sich die Variable Rechnungshöhe (`total_bill`) anzeigen.

---

mosaic

`mosaic` ist ein Zusatzpaket, welches die Analyse mit R erleichtert. Sofern noch nicht geschehen, muss es *einmalig* über

```
install.packages("mosaic")
```

installiert werden.

Um es verwenden zu können, muss es - wie jedes Paket - für *jede* neue R-Sitzung über `library(mosaic)` geladen werden. Die angegebenen Hinweise sind keine Fehlermeldung!

```
library(mosaic)
```

Sollte beim Laden eines Paketes eine Meldung wie

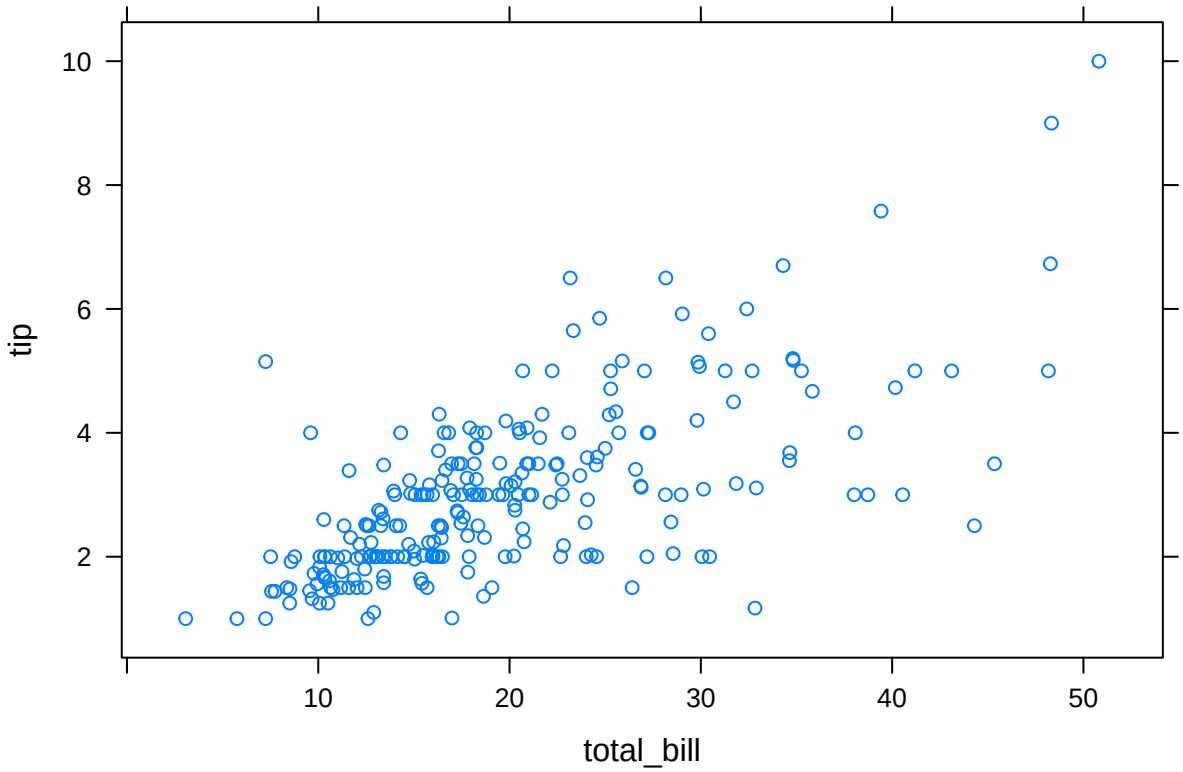
```
library(xyz)
```

```
## Error in library(xyz): there is no package called 'xyz'
```

erscheinen, muss das Paket `xyz` zunächst installiert werden (siehe oben).

Der Grundgedanke von `mosaic` ist *Modellierung*. In R und insbesondere in mosaic wir dafür die Tilde `~` verwendet. `y~x` kann dabei gelesen werden wie "y ist eine Funktion von x". Beispielsweise um eine Abbildung (Scatterplot) des Trinkgeldes `tip` (auf der Y-Achse) und Rechnungshöhe `total_bill` (auf der X-Achse) zu erhalten, kann man in R folgenden Befehl eingeben:

```
xyplot(tip ~ total_bill, data=tips)
```



Das Argument `data=tips` stellt klar, aus welchen Datensatz die Variablen kommen. Die Abbildung ist im RStudio jetzt rechts unten im Reiter *Plots* zu sehen.

Übung: Wie würden Sie den Trend beschreiben?

Wie oben erwähnt können wir R auch gut als Taschenrechner benutzen, sollten aber bedenken, dass R vektorweise arbeitet. D. h.

```
tips$tip/tips$total_bill
```

gibt für *jede Beobachtung* die relative Trinkgeldhöhe bezogen auf die Rechnungshöhe an. Über

```
(tips$tip/tips$total_bill)<0.10
```

erhalten wir einen Vektor vom Typ `logical`. Dieser nimmt nur zwei Werte an, nämlich `TRUE` und `FALSE`, je nach dem ob der jeweilige Wert kleiner als 0.10 ist oder nicht. Neben < und > bzw. <= und >= gibt es ja auch noch die Prüfung auf Gleichheit. Hierfür werden in R gleich *zwei* Gleichheitszeichen verwendet, also ==.

Übung: Was gibt folgender der Befehl zurück?

```
tips$sex=="Female"
```

Logische Vektoren können z.B. mit "und" `&` oder "oder" `|` verknüpft werden:

```
tips$sex=="Female" & tips$smoker=="Yes"
```

gibt die Tischgesellschaften als `TRUE` wieder, in denen die Rechnung von Frauen beglichen wurde *und* geraucht wurde,

```
tips$sex=="Female" | tips$smoker=="Yes"
```

gibt die Tischgesellschaften als `TRUE` wieder, in denen die Rechnung von Frauen beglichen wurde *oder* geraucht wurde.

Intern wird `TRUE` in R mit der Zahl 1 hinterlegt, `FALSE` mit 0. Mit dem Befehl `sum()` kann man daher die Elemente eines Vektor aufsummieren, also erfahren wir über

```
sum(tips$sex=="Female" & tips$smoker=="Yes")
```

dass bei 33 Tischgesellschaften bei denen geraucht wurde, eine Frau die Rechnung bezahlte. Im Verhältnis zu allen Tischgesellschaften, bei denen eine Frau zahlte, liegt der Raucheranteil also bei 0.3793103:

```
sum(tips$sex=="Female" & tips$smoker=="Yes") / sum(tips$sex=="Female")
```

Übung: Wurde bei den Tischgesellschaften, bei denen ein Mann zahlte, relativ häufiger geraucht als bei den Frauen?

Übung: Teaching Rating

Dieser Datensatz analysiert u. a. den Zusammenhang zwischen Schönheit und Evaluierungsergebnis:

Hamermesh, D.S., and Parker, A. (2005). Beauty in the Classroom: Instructors' Pulchritude and Putative Pedagogical Productivity. Economics of Education Review, 24, 369–376.

Sie können ihn von https://goo.gl/6Y3KoK herunterladen. Hier gibt es eine Beschreibung.

1. Lesen Sie den Datensatz in R ein.
2. Wie viele Zeilen, wie viele Spalten liegen vor?
3. Wie heißen die Variablen?
4. Betrachten Sie visuell den Zusammenhang von dem Evaluierungsergebnis `eval` und Schönheit `beauty`. Was können Sie erkennen?
5. Sind relativ mehr Frauen oder mehr Männer (`gender`) in einem unbefristeten Arbeitsverhältnis (*Tenure Track*, `tenure`)?

Daten importieren

Der Datenimport in R ist in vielen unterschiedlichen Dateiformaten möglich. Das `csv` Format eignet sich besonders zum Übertragen von Datendateien. Im deutschsprachigen Raum wird dabei als *Dezimaltrennzeichen* das Komma `,` und als *Datentrennzeichen* das Semikolon `;` verwendet. In der ersten Zeile sollten die Variablennamen stehen. Das Einlesen in einen R Data-Frame (hier `meineDaten`) kann dann über

```
meineDaten <- read.csv2(file.choose()) # Datei auswählen
```

erfolgen.

Der Befehl `file.choose()` öffnet dabei den Dateiordner. Bei "internationalen" `csv` Dateien ist das Datentrennzeichen i. d. R. ein Komma `,`, das Dezimaltrennzeichen ein Punkt `.`. Hier funktioniert der Import in R dann über den Befehl `read.csv`.

In R Studio gibt es im Reiter `Environment` im Fenster Rechts oben einen Menüpunkt `Import Dataset` der mehr Einstellmöglichkeiten bietet.

Excel Dateien können unter anderem mit Hilfe des Zusatzpaketes `readxl` können eingelesen werden:

```
library(readxl) # Paket laden
meineDaten <- read_excel(file.choose())  # Datei auswählen und in R einlesen
```

Hier finden Sie eine Linksammlung zu verschiedenen Datenquellen.

Literatur

- Nicholas J. Horton, Randall Pruim, Daniel T. Kaplan (2015): Project MOSAIC Little Books *A Student's Guide to R* https://github.com/ProjectMOSAIC/LittleBooks/raw/master/StudentGuide/MOSAIC-StudentGuide.pdf, Kapitel 1, 2, 13
- Chester Ismay (2016): Getting used to R, RStudio, and R Markdown https://ismayc.github.io/rbasics-book/
- Maike Luhmann (2015): *R für Einsteiger*, Kapitel 1-8
- Daniel Wollschläger (2014): *Grundlagen der Datenanalyse mit R*, Kapitel 1-4

Lizenz

Versionshinweise:

- Datum erstellt: 2017-06-30
- R Version: 3.4.0
- `mosaic` Version: 0.14.4

Kapitel 1: Einführung in Daten

Datensatz

Wir werden jetzt den *tips* Datensatz aus *Bryant, P. G. and Smith, M (1995) Practical Data Analysis: Case Studies in Business Statistics. Homewood, IL: Richard D. Irwin Publishing* näher analysieren.

Sofern noch nicht geschehen, können Sie ihn z. B. hier als `csv`-Datei direkt nach R herunterladen:

```
download.file("https://goo.gl/whKjnl", destfile = "tips.csv")
```

Wenn sich die Daten auf Ihrem Computer gespeichert sind, können Sie sie laden:

```
tips <- read.csv2("tips.csv")
```

Achtung: `read.csv` geht vom amerikanischen Format aus. Wenn es sich um eine "deutsche CSV-Datei" handelt, verwenden Sie `read.csv2`.

Tipp: Wenn Sie nicht mehr wissen, wo die Daten liegen: statt `tips.csv` den Befehl `file.choose()` als Argument für die Funktion `read.csv2` verwenden.

Inwieweit das Einlesen wie gewünscht geklappt hat, kann über

```
str(tips)
```

überprüft werden: Der Datensatz hat also 244 Zeilen (= Beobachtungen) und 7 Spalten (= Merkmale/Variablen).

Zur folgenden Analyse muss zunächst das Paket `mosaic` geladen werden:

```
library(mosaic)
```

Grafische Verfahren der Datenanalyse

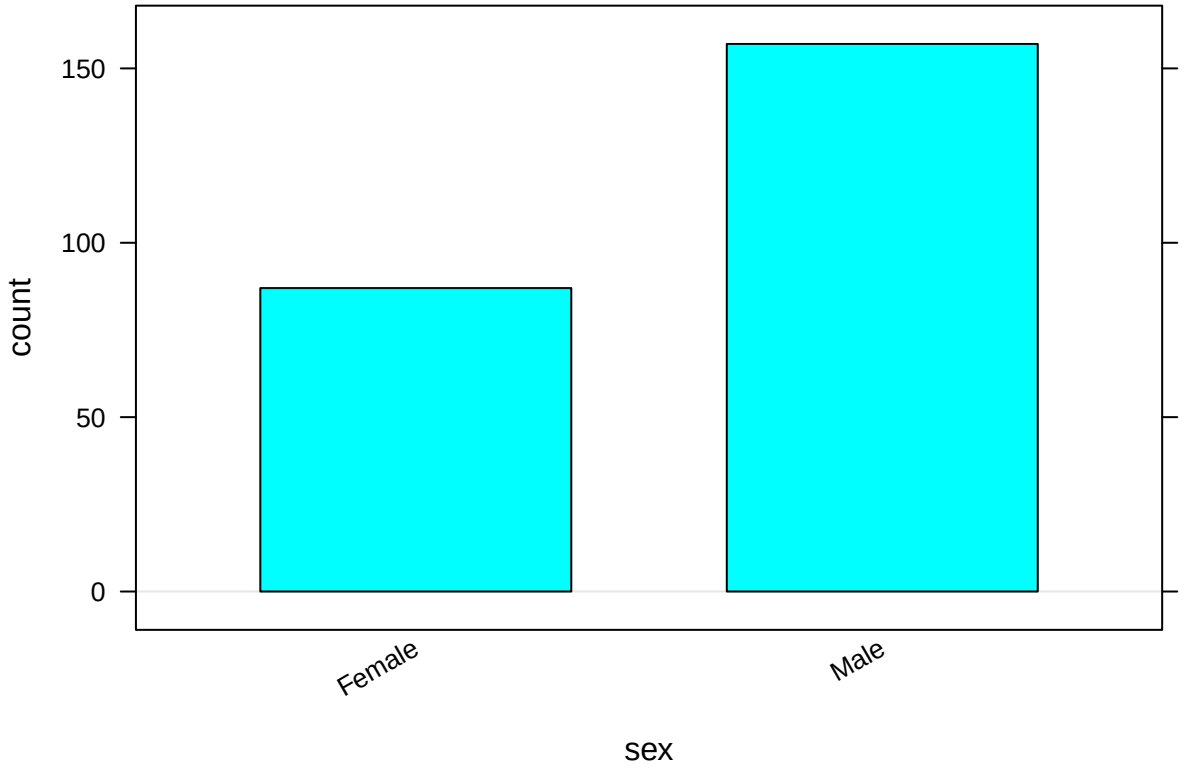
Bevor evtl. wichtige Information in zusammenfassenden Kennzahlen verloren geht, versuchen wir einen Gesamtüberblick zu erhalten.

Balkendiagramm

Balkendiagramme eignen sich am besten um Häufigkeiten darzustellen, also für kategorielle Variablen (`factor`) oder für metrische Variablen (`numeric`) mit wenigen Merkmalsausprägungen.

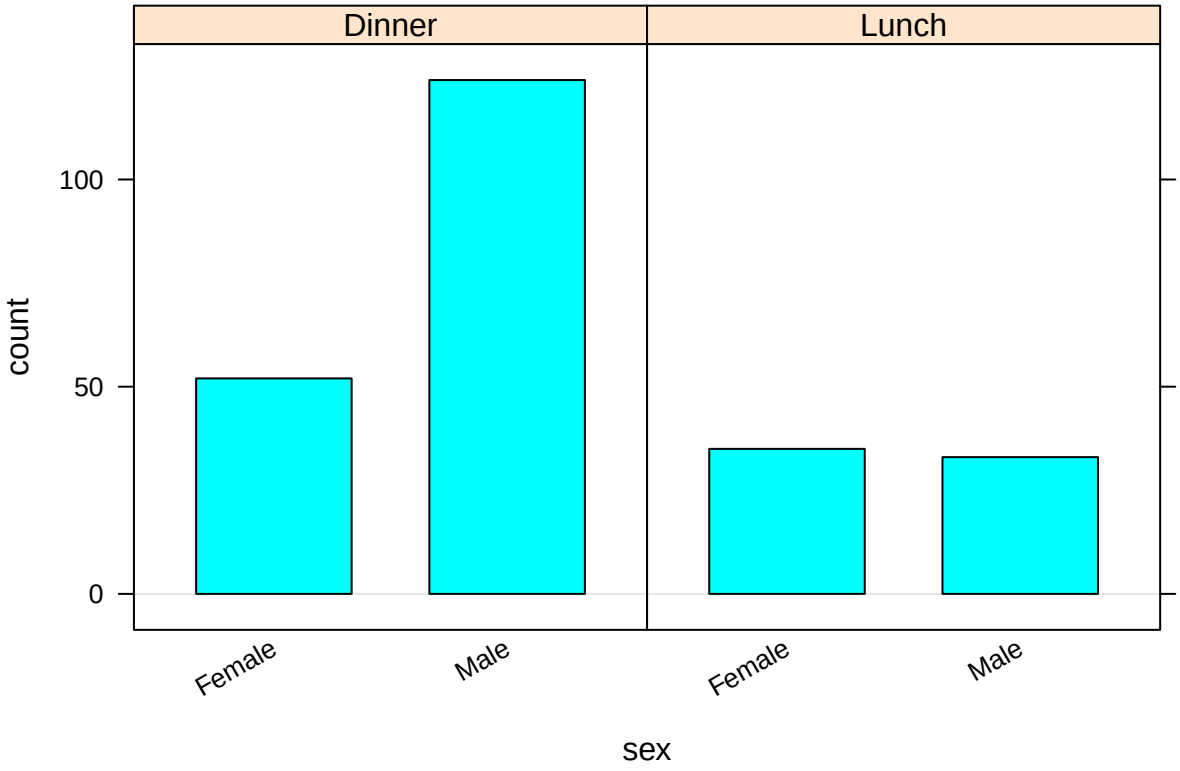
Um einen Überblick über die Geschlechterverteilung `sex` zu bekommen kann die Funktion `bargraph` aus dem Paket `mosaic` verwendet werden:

```
bargraph(~ sex, data=tips)
```



In mosaic wird (fast) immer die Formeldarstellung `y ~ x | z` verwendet: `y` in Abhängigkeit von `x` (`y` wird modelliert durch `x`), gruppiert nach den Werten von `z`, wobei einzelne Teile fehlen können, so wie im Beispiel `y` und `z`. Aber um z. B. die Verteilung des Geschlechts des Zahlenden je Tageszeit `time` darzustellen muss hier eingegeben werden:

```
bargraph(~ sex | time, data=tips)
```



Übung:

1. Zeichnen Sie ein Balkendiagramm des Rauchverhaltens `smoker` je Wochentag `day` und interpretieren Sie das Ergebnis.

Histogramm

Histogramme werden für metrische Daten verwendet, der Befehl lautet `histogram`.

Übung:

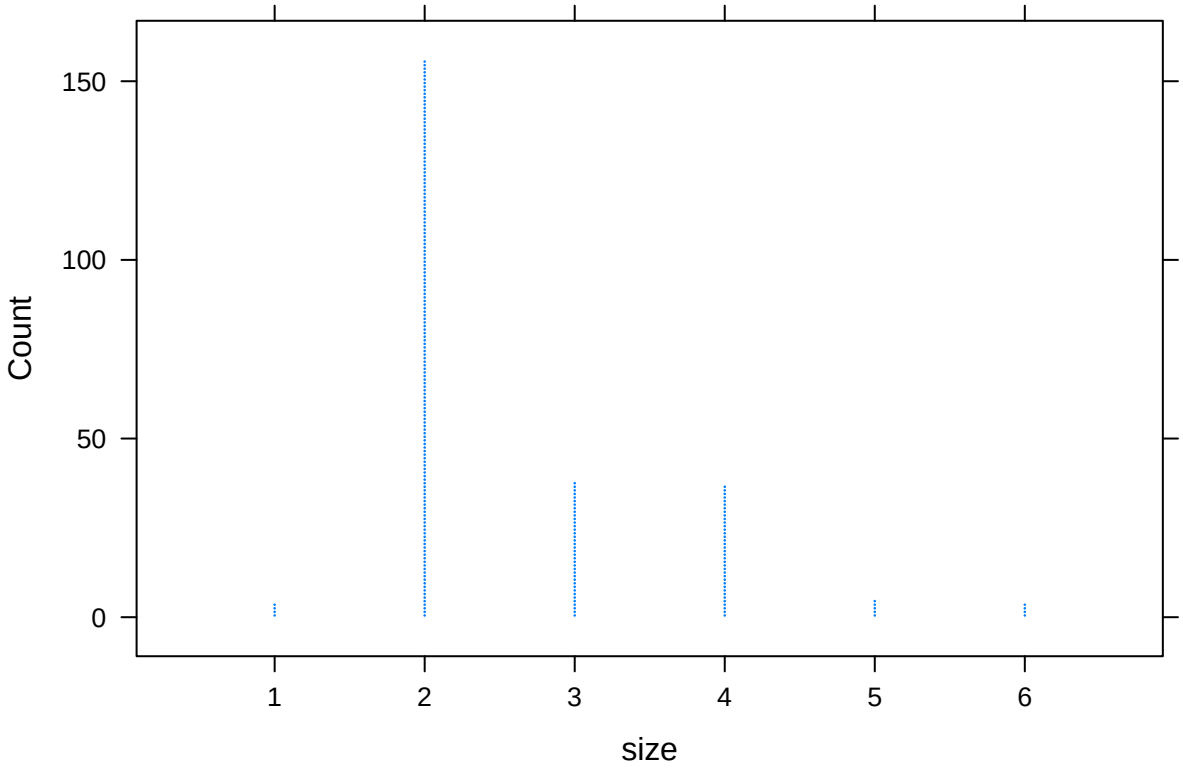
2. Welche Abbildung wird über

```
histogram(~ total_bill | sex, data=tips)
```

erzeugt?

Punktdiagramme sind eine Variante von Histogrammen, die besonders für metrische Variablen mit wenigen Merkmalsausprägungen geeignet sind.

```
dotPlot(~ size, nint=6, data=tips)
```

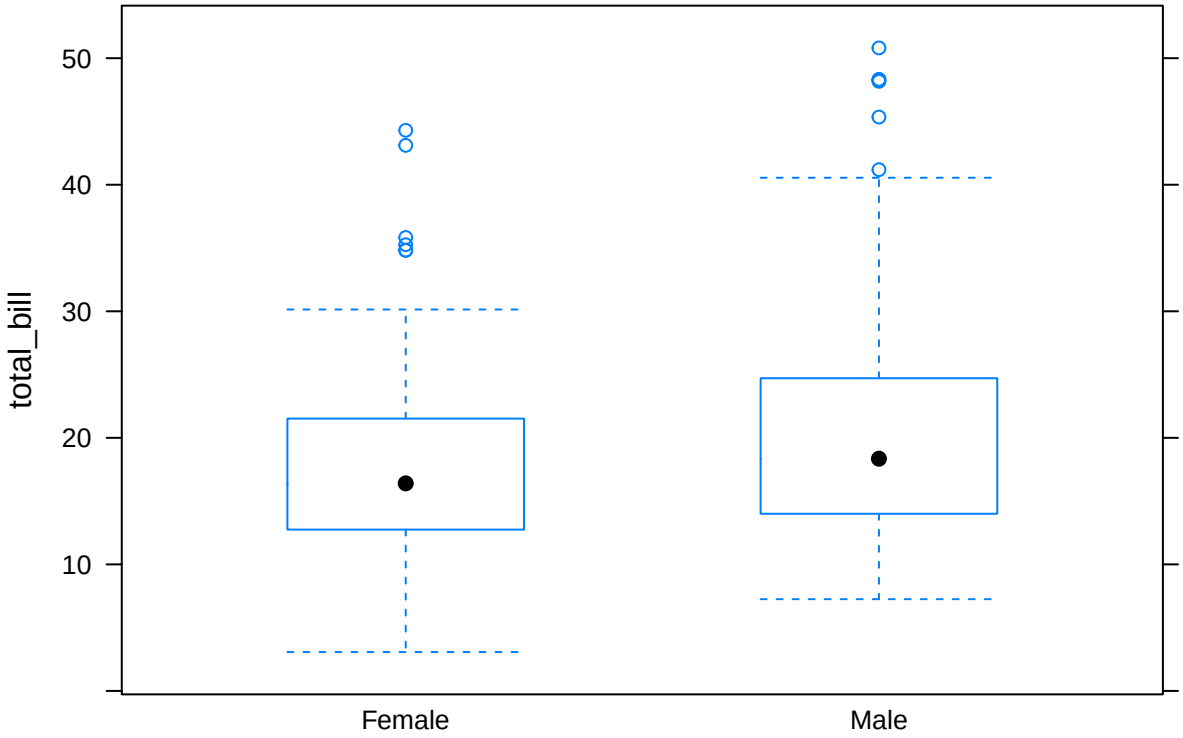


Hier wurde ein zusätzliche Parameter der Funktion `dotPlot` übergeben: `nint=6`. Dieser Parameter wurde wurde verwendet, um die Abbildung schöner zu machen. Welche Optionen es gibt und was diese bedeuten, kann man in R häufig einfach über die Hilfe, hier also `?dotPlot`, erfahren.

Boxplots

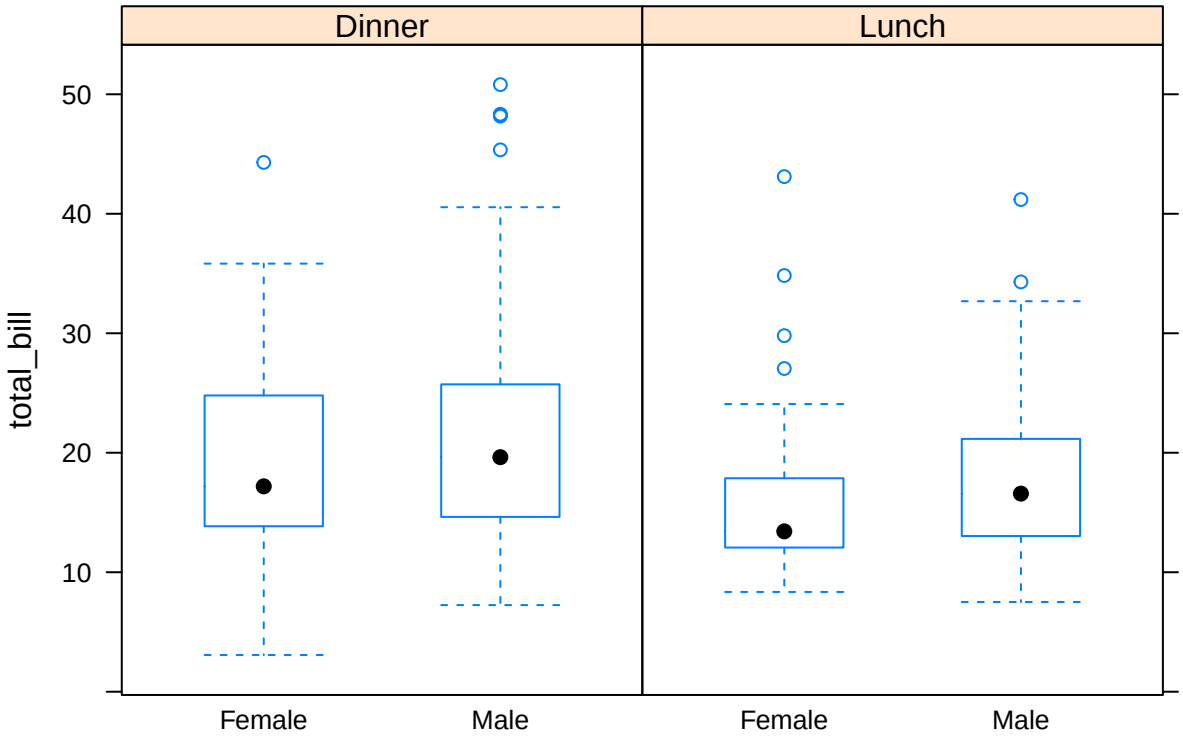
Boxplots zeigen nicht nur den Median (50%-Quantil) sowie das obere (75%) und untere (25%) Quartil - und damit den Interquartilsabstand -, sondern geben auch Hinweise auf potentielle Ausreißer:

```
bwplot(total_bill ~ sex, data=tips)
```



und gruppiert nach Tageszeit:

```
bwplot(total_bill ~ sex | time, data=tips)
```



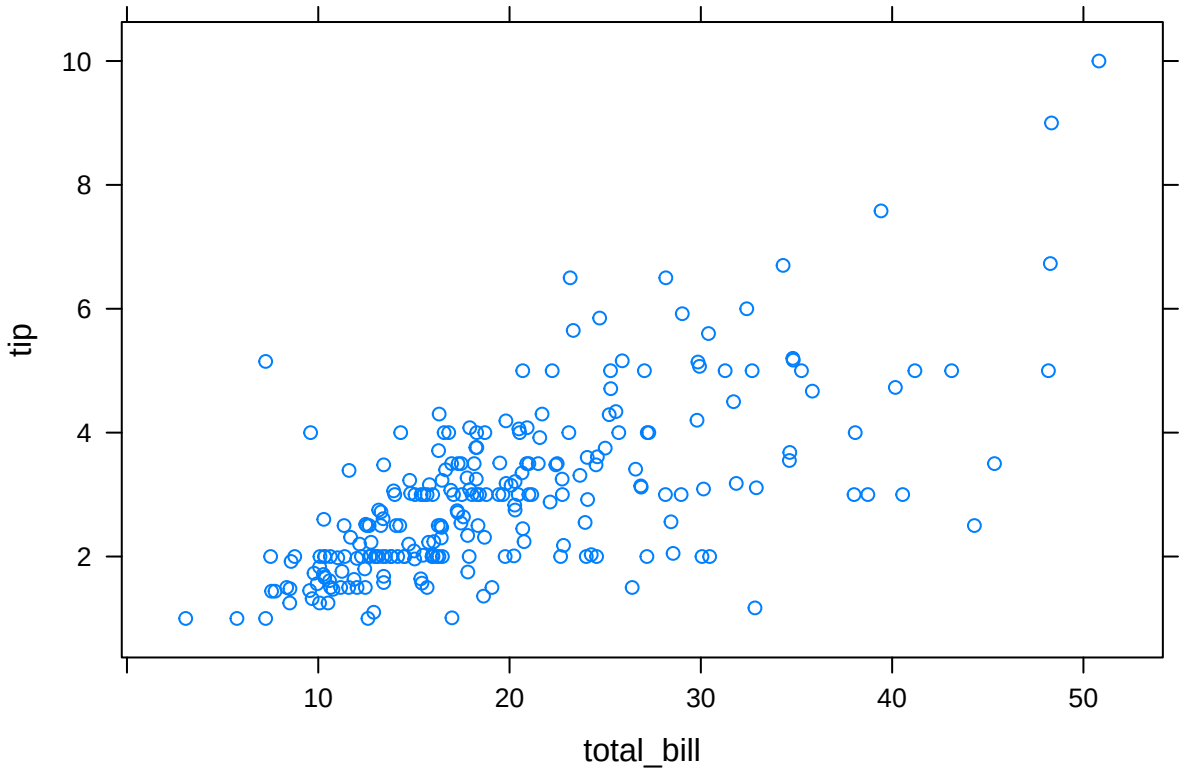
Übung:

3. Zeichen Sie einen Boxplot für die Trinkgeldhöhe `tip` in Abhängigkeit davon, ob geraucht wurde (`smoker`). Gibt es Unterschiede in der Trinkgeldhöhe und, wenn ja, in welchem Bereich?

Scatterplot (Streudiagramme)

Streudiagramme sind besonders gut geeignet, um einen Überblick auf den Zusammenhang zweier metrischer Merkmale zu erhalten; beispielsweise um den Zusammenhang von `tip` und `total_bill` zu analysieren.

```
xyplot(tip ~ total_bill, data=tips)
```



Wenig überraschend steigt die Trinkgeldhöhe mit der Rechnung. Wie sieht es relativ aus? Dazu müssen wir zunächst ein neues Merkmal im Datensatz erzeugen, z. B.:

```
tips$tip_relativ <- tips$tip / tips$total_bill
```

Im Datensatz `tips` wird der (neuen) Variable `tip_relativ` der Quotient aus Trinkgeld und Rechnungshöhe zugewiesen.

Übung:

4. Erstellen Sie eine Abbildung, mit der Sie visuell gucken können, wie der Zusammenhang zwischen der relativen Trinkgeldhöhe (abhängige Variable) und der Rechnungshöhe (uanbhängige Variable) aussieht, und ob sich dieser je nach Geschlecht des Rechnungszahlers unterscheidet.

Mosaicplot

Mosaicplots eignen sich, um den Zusammenhang zwischen kategoriellen Variablen darzustellen. Zunächst müssen wir dazu eine Kreuztabelle erstellen. Das geht in `mosaic` über den Befehl `tally`. Dieser Befehl ist recht mächtig – dazu später mehr. Wir erzeugen eine solche Kreuztabelle zwischen Tageszeit und Rauchen über

```
tab_smoke_time <- tally(smoker ~ time, data=tips)
```

Dem (neuen) R Objekt `tab_smoke_time` wird also das Ergebnis des `tally` Befehls zugewiesen. Wie das Ergebnis aussieht, und welchen Typ es hat erfahren wir über

```
print(tab_smoke_time)
```

```
##        time
## smoker Dinner Lunch
##    No     106    45
##    Yes     70    23
```

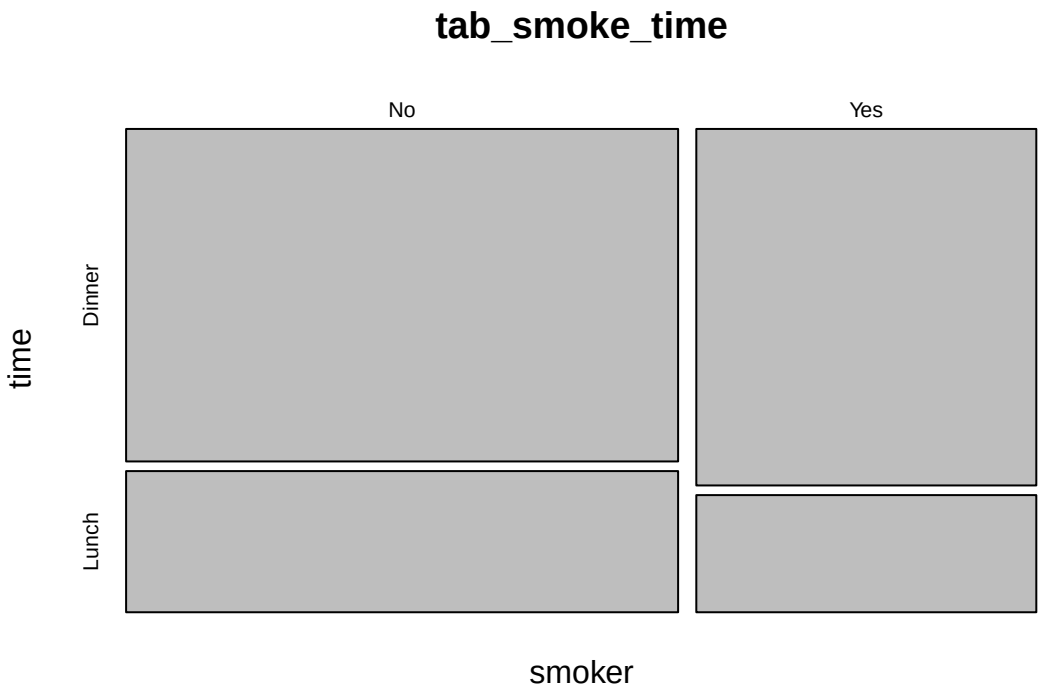
```
str(tab_smoke_time)
```

```
##  'table' int [1:2, 1:2] 106 70 45 23
##  - attr(*, "dimnames")=List of 2
##   ..$ smoker: chr [1:2] "No" "Yes"
##   ..$ time  : chr [1:2] "Dinner" "Lunch"
```

Es handelt sich also um eine Tabelle (`table`) der Dimension 2, 2, also 2 Zeilen, 2 Spalten.

Der Befehl für einen Mosaicplot lautet `mosaicplot`:

```
mosaicplot(tab_smoke_time)
```



Korrelationsplot

Mit Hilfe des Zusatzpakets `corrplot` lassen sich Korrelationen besonders einfach visualisieren. Das Paket muss wie jedes Paket *einmalig* über

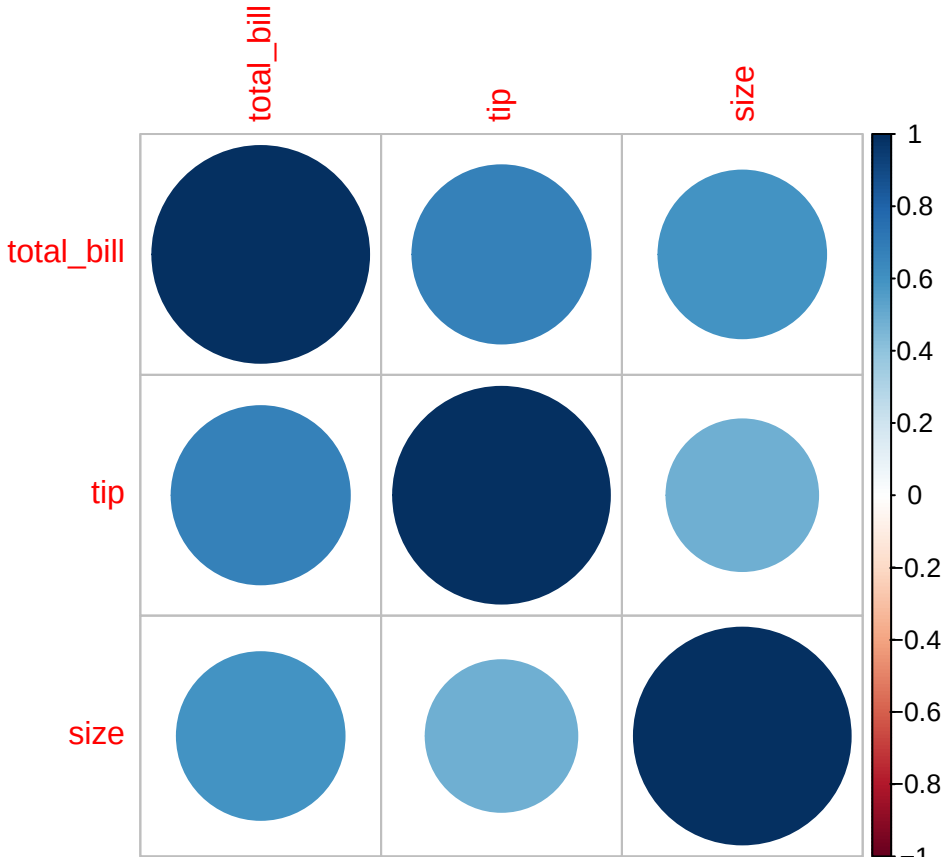
```
install.packages("corrplot")
```

installiert werden – wiederum werden evt. weitere benötigte Pakete mit-installiert. Nach dem Laden des Pakets über

```
library(corrplot)
```

kann dies über

```
corrplot(cor(tips[,c("total_bill", "tip", "size")]))
```



gezeichnet werden. Je intensiver die Farbe, desto höher die Korrelation. Hier gibt es unzählige Einstellmöglichkeiten, siehe `?corrplot` bzw. für Beispiele:

```
vignette("corrplot-intro")
```

Kennzahlen der Datenanalyse

Nachdem wir einen ersten visuellen Eindruck gewonnen haben, wollen wir uns jetzt Kennzahlen widmen.

Lagemaße

Das Minimum und Maximum von metrischen Daten kann einfach durch `min` bzw. `max` bestimmt werden, in `mosaic` auch "modelliert":

```
min(~ total_bill | smoker, data=tips)
```

```
##    No  Yes
## 7.25 3.07
```

gibt also das Minimum der Rechnungshöhe, getrennt nach Raucher und Nichtrauchern an, d. h. das Minimum bei den Rauchern lag bei 3.07$.

Übung:

5. Bestimmen Sie das Maximum der Trinkgeldhöhe je Wochentag (`day`)

Lagemaße sollen die zentrale Tendenz der Daten beschreiben. Gebräuchlich sind in der Regel der arithmetische Mittelwert `mean`

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{x_1 + x_2 + x_3 + \cdots + x_n}{n}$$

```
mean(~ total_bill, data=tips)
```

```
## [1] 19.78594
```

sowie der Median (Zentralwert) `median`:

```
median(~ total_bill, data=tips)
```

```
## [1] 17.795
```

Den jeweiligen Rang der Beonbachtungen erhalten Sie über

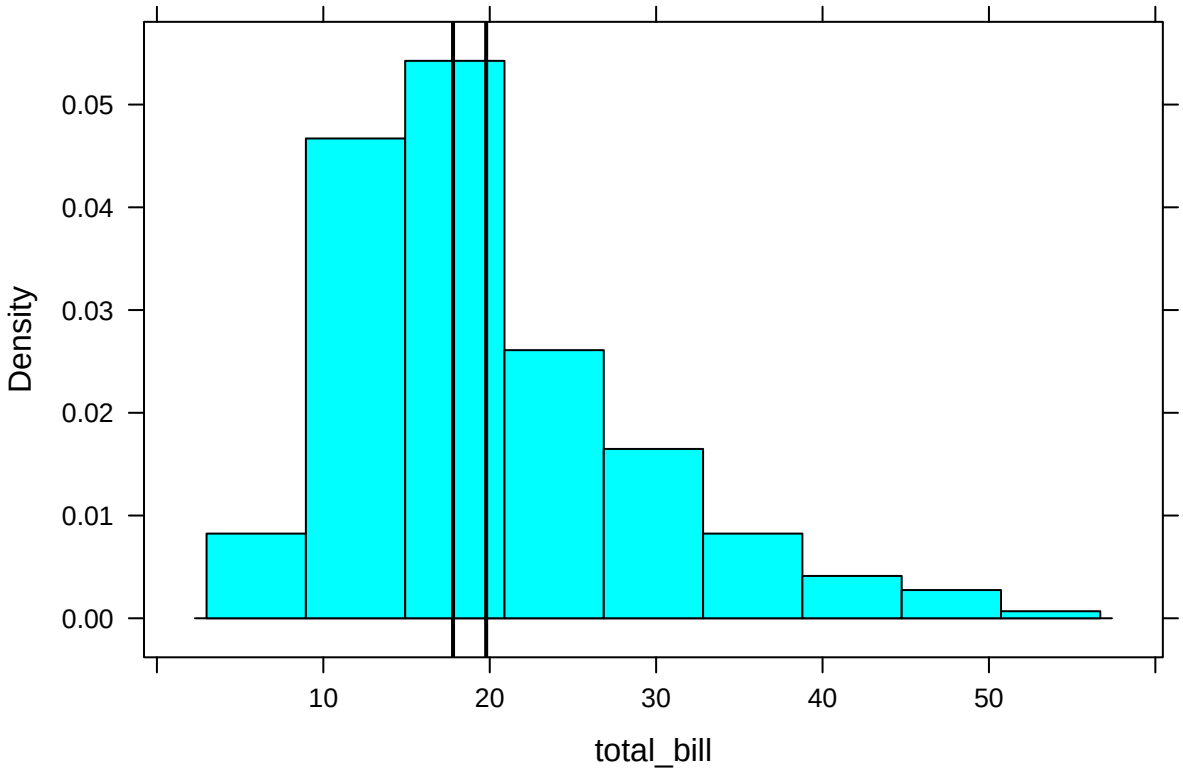
```
rank(tips$total_bill)
```

```
##   [1] 113.0  25.5 162.5 179.0 187.0 192.0  12.0 199.0  82.0  79.0  21.0
##  [12] 229.0  86.0 136.0  80.0 166.0  23.5 100.0 112.0 156.0 126.5 151.5
##  [23]  91.0 234.0 147.0 123.0  62.0  51.0 167.0 144.0  13.0 135.0  83.0
##  [34] 157.5 122.0 182.0 101.0 111.0 138.0 217.0  97.0 118.0  70.0  15.0
##  [45] 215.0 133.5 169.0 220.0 207.0 128.0  48.0  22.0 227.0  17.0 193.0
##  [56] 143.0 231.0 196.0  34.0 242.0 151.5  68.5  32.0 133.5 121.0 148.0
##  [67] 105.0   1.0 149.0  81.0  41.0 114.0 198.0 191.0  78.0  27.0 126.5
##  [78] 202.0 174.0 116.0 142.0 109.0  18.5 221.0  94.5 228.0  57.0 132.0
##  [89] 188.0 164.0 208.0 171.0   2.0 102.0 173.0 235.0 203.0  42.0 162.5
## [100]  46.0  35.0  85.0 239.0 170.0 161.0  84.0 154.0 190.0 130.0  75.0
```

```
##  [111]  71.0   3.5 232.0 180.0 194.0 117.0 212.0  30.0  45.0 183.0  39.0
##  [122]  65.0  74.0  93.0  47.0 210.0  10.0  77.0  36.0 175.0 141.0 150.0
##  [133]  33.0  44.0 131.0   9.0  23.5  73.0  96.0  59.0 119.0 224.0 237.0
##  [144] 200.0 104.0   8.0 137.0  40.0  16.0   5.0  72.0  58.0 115.0 186.0
##  [155] 145.0 211.0 241.0 189.0  63.0 107.0 165.0  50.0  98.0  68.5 120.0
##  [166] 185.0 159.0 218.0  28.0  29.0 244.0  92.0   3.5 219.0 110.0 223.0
##  [177] 125.0  76.0  14.0 225.0 226.0 178.0 240.0 177.0 236.0 157.5 160.0
##  [188] 216.0 129.0 176.0  89.5 146.0 206.0  87.0 108.0   6.0  25.5 238.0
##  [199]  55.5  67.0 139.0  52.0  55.5 103.0 155.0 106.0 197.0 233.0 184.0
##  [210]  53.0 213.0 195.0 243.0  60.0 205.0  54.0 204.0  37.0   7.0 214.0
##  [221]  43.0  65.0  11.0  94.5  65.0  99.0  20.0 153.0  61.0 168.0 181.0
##  [232]  89.5  38.0  31.0  88.0  18.5  49.0 222.0 230.0 209.0 201.0 172.0
##  [243] 124.0 140.0
```

Diese unterscheiden sich:

```
meantb <- mean(~ total_bill, data=tips) # Mittelwert
mediantb <- median(~ total_bill, data=tips) # Median
histogram(~ total_bill, v=c(meantb, mediantb), data=tips)
```



Über die Option `v=` werden vertikale Linien an den entsprechenden Stellen gezeichnet. Mit `h=` können horizontale Linien gezeichnet werden.

Übung:

6. Begründen Sie anhand des Histogramms, warum hier der Median kleiner als der arithmetische Mittelwert ist.

Auch Lagemaße zu berechnen in Abhängigkeit der Gruppenzugehörigkeit ist einfach. So können Sie den arithmetischen Mittelwert in Abhängigkeit von Geschlecht und Tageszeit berechnen:

```
mean(total_bill ~ sex + time, data=tips)
```

```
## Female.Dinner   Male.Dinner  Female.Lunch    Male.Lunch
##      19.21308      21.46145      16.33914      18.04848
```

Übung:

7. Bestimmen Sie den Median der Trinkgeldhöhe anhand der Anzahl Personen in der Tischgesellschaft.

Für kategorielle Variablen können eigentlich zunächst nur die Häufigkeiten bestimmt werden:

```
tally(~day, data=tips)
```

```
## day
##  Fri  Sat  Sun Thur
##   19   87   76   62
```

Relative Häufigkeiten werden bei `mosaic` mit der zusätzlichen Option `format='proportion'` angefordert:

```
tally(~day, format="proportion", data=tips)
```

```
## day
##        Fri        Sat        Sun       Thur
## 0.07786885 0.35655738 0.31147541 0.25409836
```

Streuungsmaße

Die Variation der Daten, die wir grafisch und auch in den (bedingten) Lagemaßen gesehen haben ist eines der zentralen Themen der Statistik: Können wir die Variation vielleicht erklären? Variiert die Rechnungshöhe vielleicht mit der Anzahl Personen?

Zur Bestimmung der Streuung werden in der Regel der Interquartilsabstand `IQR` sowie Varianz `var` bzw. Standardabweichung `sd`

$$s = sd = \sqrt{x^2} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

herangezogen:

```
IQR(~total_bill, data=tips)
```

```
## [1] 10.78
```

```
var(~total_bill, data=tips)
```

```
## [1] 79.25294
```

```
sd(~total_bill, data=tips)
```

```
## [1] 8.902412
```

Um die Standardabweichung in Abhängigkeit der Gruppengröße zu berechnen genügt der Befehl:

```
sd(~total_bill | size, data=tips)
```

```
##        1        2        3        4        5        6
## 3.010729 6.043729 9.407065 8.608603 7.340396 9.382000
```

Bei 4 Personen lag die Standardabweichung als bei 8.61$.

Um jetzt z. B. den Variationskoeffizienten zu berechnen wird

```
sd(~total_bill | size, data=tips) / mean(~total_bill | size, data=tips)
```

```
##         1         2         3         4         5         6
## 0.4157031 0.3674443 0.4041247 0.3008579 0.2441265 0.2693655
```

gebildet.

Übung:

8. Zu welcher Tageszeit ist die Standardabweichung des Trinkgelds geringer? Zum Lunch oder zum Dinner?

Die *üblichen* deskriptiven Kennzahlen sind in `mosaic` übrigens in einer Funktion zusammengefasst: `favstats`.

```
favstats(tip~day, data=tips)
```

```
##    day  min     Q1 median     Q3   max     mean       sd  n missing
## 1  Fri 1.00 1.9600  3.000 3.3650  4.73 2.734737 1.019577 19       0
## 2  Sat 1.00 2.0000  2.750 3.3700 10.00 2.993103 1.631014 87       0
## 3  Sun 1.01 2.0375  3.150 4.0000  6.50 3.255132 1.234880 76       0
## 4 Thur 1.25 2.0000  2.305 3.3625  6.70 2.771452 1.240223 62       0
```

Zusammenhangsmaße

Kennzahlen für den linearen Zusammenhang von metrischen Variablen sind Kovarianz `cov`

$$s_{xy} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

und der Korrelationskoeffizient `cor`

$$r = \frac{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2}} = \frac{s_{xy}}{s_x s_y}$$

```
cov(tip ~ total_bill, data=tips)
```

```
## [1] 8.323502
```

```
cor(tip ~ total_bill, data=tips)
```

```
## [1] 0.6757341
```

Für kategorielle Variablen wird in diesen Abschnitt zunächst nur die Kreuztabelle verwendet:

```
tally(smoker~sex, format="proportion", data=tips)
```

```
##        sex
## smoker    Female      Male
##    No  0.6206897 0.6178344
##    Yes 0.3793103 0.3821656
```

Übung:

9. Zu welcher Tageszeit wurde relativ häufiger von einer Frau die Rechnung bezahlt?

Übung: Teaching Rating

Dieser Datensatz analysiert u. a. den Zusammenhang zwischen Schönheit und Evaluierungsergebnis von Dozenten:

Hamermesh, D.S., and Parker, A. (2005). Beauty in the Classroom: Instructors' Pulchritude and Putative Pedagogical Productivity. Economics of Education Review, 24, 369–376.

Sie können ihn von https://goo.gl/6Y3KoK herunterladen.

1. Erstellen Sie ein Balkendiagramm der Variable `native` gruppiert nach der Variable `minority`.
2. Erstellen Sie ein Histogramm der Variable `beauty` gruppiert nach der Variable `gender`.

3. Vergleichen Sie das Evaluationsergebnis `eval` in Abhängigkeit ob es sich um einen Single-Credit Kurs `credits` handelt mit Hilfe eines Boxplots.
4. Zeichnen Sie ein Scatterplot der Variable `eval` in Abhängigkeit der zu definierenden Variable "Evaluierungsquote": `students/allstudents`.
5. Berechnen Sie deskriptive Kennzahlen der Variable `eval` in Abhängigkeit ob es sich um einen Single-Credit Kurs `credits` handelt.

Literatur

- David M. Diez, Christopher D. Barr, Mine Çetinkaya-Rundel (2014): *Introductory Statistics with Randomization and Simulation*, https://www.openintro.org/stat/textbook.php?stat_book=isrs, Kapitel 1
- Nicholas J. Horton, Randall Pruim, Daniel T. Kaplan (2015): *Project MOSAIC Little Books – A Student's Guide to R*, https://github.com/ProjectMOSAIC/LittleBooks/raw/master/StudentGuide/MOSAIC-StudentGuide.pdf, Kapitel 3.1, 3.2, 4.1, 5.1, 5.2, 6.1
- Chester Ismay, Albert Y. Kim (2017): *ModernDive – An Introduction to Statistical and Data Sciences via R*, https://ismayc.github.io/moderndiver-book/
- Maike Luhmann (2015): *R für Einsteiger*, Kapitel 9-11
- Andreas Quatember (2010): *Statistik ohne Angst vor Formeln*, Kapitel 1
- Daniel Wollschläger (2014): *Grundlagen der Datenanalyse mit R*, Kapitel 14

Lizenz

Versionshinweise:

- Datum erstellt: 2017-06-30
- R Version: 3.4.0
- `mosaic` Version: 0.14.4

Kapitel 2: Einführung Wahrscheinlichkeit und Inferenz

Zufall und Wahrscheinlichkeit

In dieser Übung werden wir ein wenig programmieren, daher bietet es sich an, die Befehle in einem Skript zu speichern. Gehen Sie dazu in RStudio in das Menü `File` und dort auf `New File` und wählen `R Script` aus. Dies können Sie dann am Ende über `File` und `Save` bzw. `Safe as` speichern – und über

`Open File` später auch wieder öffnen. Um die Befehle an die Konsole zu übergeben klicken Sie entweder auf `Run` (nur ausgewählte Zeile, Tastenkürzel `Strg+Enter`) oder `Source` (ganze Programm).

Zunächst laden wir wieder das Zusatzpaket mosaic, falls noch nicht geschehen:

```
library(mosaic)
```

Um den Zufall zu bändigen, setzen wir den Zufallszahlengenerator, z. B. auf `1896`

```
set.seed(1896)
```

Dieser Befehl sorgt dafür, dass wir immer denselben "Zufall" haben.

Beim Roulette gibt es 37 Zahlen und 3 Farben: 0-37, wobei 18 Zahlen Schwarz, 18 Zahlen Rot und die 0 Grün ist – auf diese können Sie auch nicht setzen.

Angenommen Sie setzen auf Farbe. Dann beträgt Ihre Gewinnwahrscheinlichkeit $\frac{18}{37}$, da 18 von 37 Fächern "ihre" Farbe hat, die Wahrscheinlichkeit eines Verlustes liegt bei $\frac{19}{37} = 1 - \frac{18}{37}$.

Definieren wir in R einen `factor`-Vektor mit zwei Elementen für Gewinn und Verlust:

```
roulette <- factor(c("Gewinn", "Verlust"))
```

Mit diesem Vektor können wir jetzt virtuell und ganz ohne Risiko über den Befehl `resample` Roulette spielen

```
resample(roulette, size=1, prob=c(18/37, 19/37))
```

```
## [1] Gewinn
## Levels: Gewinn Verlust
```

```
resample(roulette, size=10, prob=c(18/37, 19/37))
```

```
##  [1] Gewinn  Gewinn  Gewinn  Gewinn  Gewinn  Gewinn  Verlust Gewinn
##  [9] Verlust Gewinn
## Levels: Gewinn Verlust
```

Mit dem Argument `size` wird also eingestellt, wie oft Roulette gespielt wird, `prob` ist der Vektor der Wahrscheinlichkeiten für die einzelnen Elemente im Ereignisvektor, hier `roulette`.

Über

```
spiele <- resample(roulette, size=100, prob=c(18/37, 19/37))
```

wird dem Vektor `spiele` das Ergebnis von `100` Roulettespielen zugewiesen. Die Häufigkeitsverteilung erhalten wir wie gewohnt über den Befehl `tally`:

```
tally(~spiele, format="proportion")
```

```
## spiele
##  Gewinn Verlust
##    0.51    0.49
```

Das **Gesetz der großen Zahl** sagt aus, dass sich auf *lange* Sicht die beobachtete relative Häufigkeit der theoretischen Wahrscheinlichkeit annähert:

```
tally(~resample(roulette, size=10, prob=c(18/37, 19/37)), format="proportion")
```

```
## resample(roulette, size = 10, prob = c(18/37, 19/37))
##  Gewinn Verlust
##      0.7     0.3
```

```
tally(~resample(roulette, size=100, prob=c(18/37, 19/37)), format="proportion")
```

```
## resample(roulette, size = 100, prob = c(18/37, 19/37))
##  Gewinn Verlust
##    0.44    0.56
```

```
tally(~resample(roulette, size=1000, prob=c(18/37, 19/37)), format="proportion")
```

```
## resample(roulette, size = 1000, prob = c(18/37, 19/37))
##  Gewinn Verlust
##   0.488   0.512
```

```
tally(~resample(roulette, size=1000000, prob=c(18/37, 19/37)), format="proportion")
```

```
## resample(roulette, size = 1e+06, prob = c(18/37, 19/37))
##   Gewinn  Verlust
## 0.486433 0.513567
```

Die theoretische Wahrscheinlichkeit eines Gewinns liegt bei $\frac{18}{37} \approx 0.4865$: Achtung, dass Gesetz der großen Zahl gilt für den Durchschnitt und auf lange Sicht, evtl. Ungleichgewichte, z. B. 5 Gewinne in Folge werden im Laufe der Zeit abgeschwächt, und nicht durch anschließende 5 Verluste ausgeglichen.

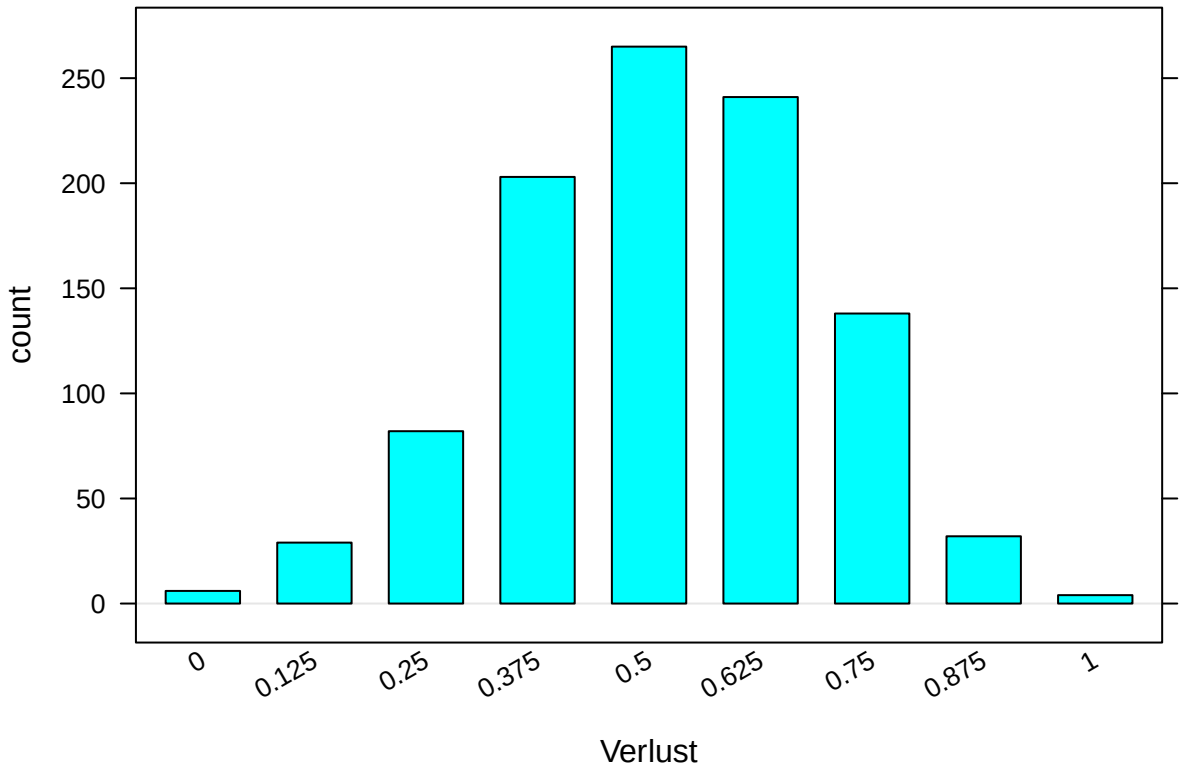
Bei bestimmten Spielstrategien, z. B. bei der sogenannten Martingale oder Verdoppelungsstrategie, ist man daran interessiert, wie wahrscheinlich es ist, z. B. 8-mal in Folge zu verlieren. Natürlich kann das mit Hilfe der *Binomialverteilung* ausgerechnet werden, wir können es aber auch simulieren: `do()` ist eine einfache Schleifenfunktion in mosaic. Um z. B. 1000-mal jeweils 8 Runden Roulette zu spielen - und das Ergebnis zu speichern - genügt:

```
farbspiele <- do(1000)*tally(~resample(roulette, size=8, prob=c(18/37, 19/37)),
                             format="proportion")
```

`farbspiele` ist jetzt ein Datensatz (`data.frame`) mit 1000 Zeilen (=Simulationen) und den relativen Häufigkeiten für Gewinn und Verlust in den 8 Spielen in den Spalten.

Das Balkendiagramm der relativen Verlusthäufigkeit zeigt, dass es zwar selten, aber doch vorkommt, alle 8 Spiele zu verlieren.

```
bargraph(~Verlust, data=farbspiele)
```



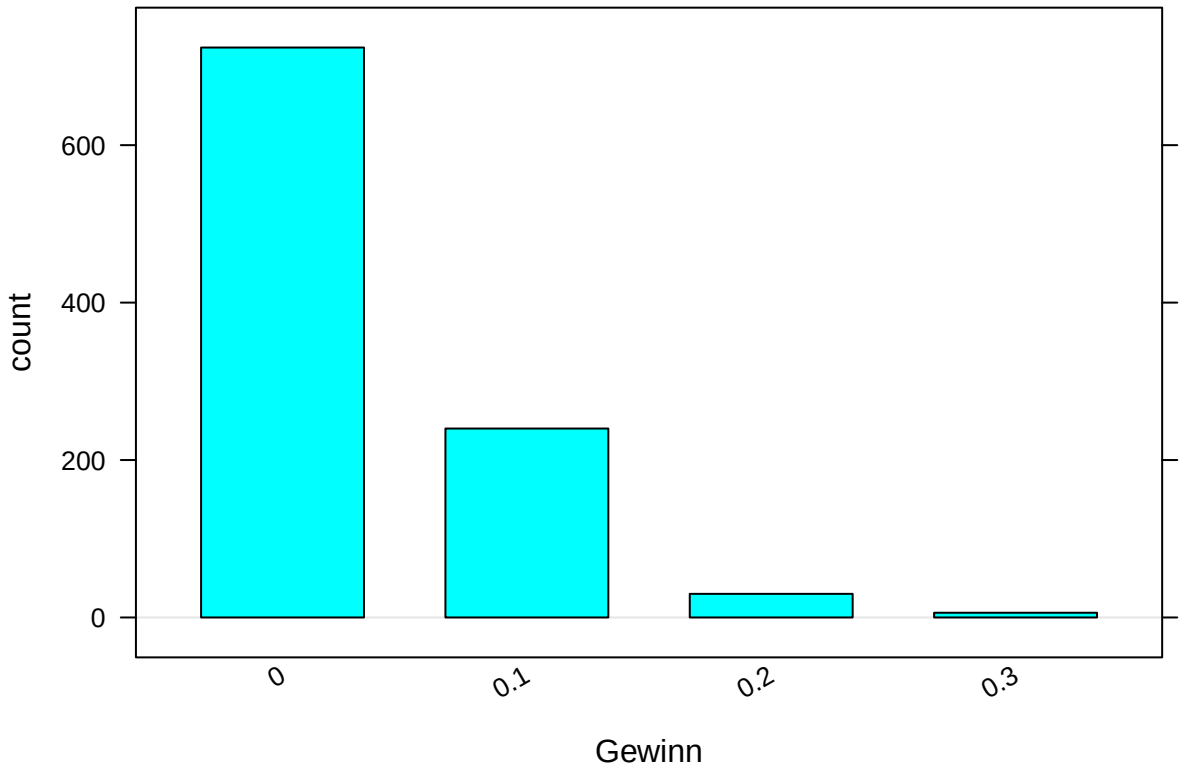
Wir haben in 4 von 1000 Wiederholungen nur verloren, d. h. 8 von 8 Spielen.

Übung:

1. Wenn Sie statt auf Farbe auf eine Zahl setzen, beträgt Ihre Gewinnwahrscheinlichkeit $\frac{1}{37}$. Simulieren Sie 1000-mal 10 Spiele. Wie oft haben Sie mindestens 1-mal gewonnen?

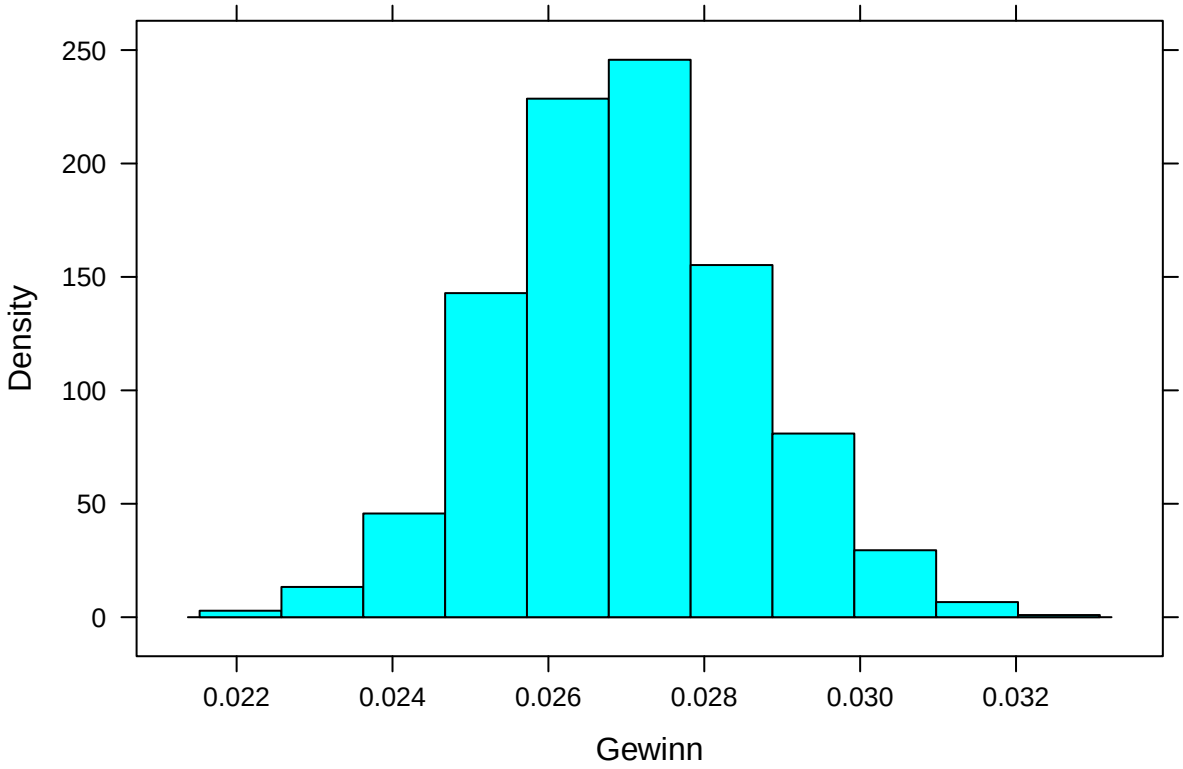
Wenn wir uns die Verteilung der Daten der Übung angucken

```
zahlspiele <- do(1000)*tally(~resample(roulette, size=10,
                                       prob=c(1/37, 36/37)), format="proportion")
bargraph(~Gewinn, data=zahlspiele)
```



stellen wir fest, dass diese Daten (leider) extrem rechtsschief sind, d. h., i. d. R. gewinnen wir in keiner der 10 Runden, Gewinn=0. Wenn wir `size=10` durch `size=10000` ersetzen (d. h., bei jeden der 1000 Simulationen 10000 Runden spielen), passiert folgendes (Darstellung jetzt als Histogramm, da es zu sehr viele mögliche Ausprägungen für die Anzahl Gewinne gibt):

```
zahlspiele2 <- do(1000)*tally(~resample(roulette, size=10000,
                                        prob=c(1/37, 36/37)), format="proportion")
histogram(~Gewinn, data=zahlspiele2)
```



Die Daten werden *normaler*, symmetrischer, d. h., die Verteilung des Gewinnanteilswertes nähert sich einer Normalverteilung an. Diese Phänomen ist der Hintergrund des **Zentralen Grenzwertsatzes**.

Übung:

2. Zurück zum Farbspiel (`farbspiele`): Wie hoch schätzen Sie die Wahrscheinlichkeit anhand der Simulation, dass Sie mindestens die Hälfte Ihrer 8 Spiele gewinnen?

Richtig: 0.585, das ist also anscheinend recht wahrscheinlich, während der relative Anteil der Spiele, in denen Sie maximal 1 der 8 Spiele gewinnen, recht klein ist:

```
anteil <- prop(farbspiele$Gewinn <= 1/8)
anteil
```

Das kommt also eher selten vor. Pech. Vielleicht würden Ihnen aber auch Zweifel kommen, ob der Tisch fair ist. In der Simulation liegt also die Wahrscheinlichkeit, bei einem fairen Tisch bei 8 Spielen höchstens einmal zu gewinnen bei 3.6%.

Hypothesentest, p-Wert und Konfidenzintervall

Im Paper *Hose, C., Lübke, K., Nolte, T., und Obermeier, T. (2012): Ratingprozess und Ratingergebnis: Ein Experiment bei qualitativen Ratingkriterien, Kredit & Rating Praxis (6), 12-14* wird folgendes Experiment untersucht: Unterscheidet sich die Einschätzung (Rating) eines Unternehmens, je nach dem, ob die Person alleiniger Entscheider (Typ A) oder derjenige ist, der die Entscheidungsvorlage vorbereitet (Typ B). Im

Rahmen des Experiments wurden die Studierenden zufällig den verschiedenen Typen A und B zugeordnet. Von 151 alleinigen Entscheidern (Typ A) beurteilten 79 das Beispielunternehmen überwiegend positiv (++, +), von 143 Entscheidungsvorlagenerstellern (Typ B) entschieden ebenfalls 79 überwiegend positiv.

Zeigt das unterschiedliche Verhältnis: Typ A: $\frac{79}{151} = 52.32\%$ zu Typ B: $\frac{79}{143} = 55.24\%$, dass alleinige Entscheider die Bonität kritischer einstufen, oder könnte das Ergebnis Zufall sein?

Das Chancenverhältnis, das **Odds Ratio** liegt bei $\frac{\frac{79}{151-79}}{\frac{79}{143-79}} = 0.89$, dass ein alleiniger Entscheider positiv einstuft – im Vergleich zum vorläufigen Entscheider.

Zunächst erzeugen wir einen Vektor der Länge 2 mit den Entscheidungstypen, aus dem wir simulieren können:

```
typ <- factor(c("A", "B"))
entscheider <- rep(typ, c(151,143))
tally(~entscheider)
```

```
## entscheider
##   A   B
## 151 143
```

sowie einen Vektor der Entscheidungen:

```
rating <- factor(c("Positiv", "Nicht Positiv"))
entscheidungen <- rep(rating, c(79+79, (151+143)-(79+79)))
tally(~entscheidungen)
```

```
## entscheidungen
## Nicht Positiv       Positiv
##           136           158
```

Aus diesem Vektor ziehen wir eine zufällige Stichprobe von 151 Entscheidungen von Typ A.

```
simentscheidung <- sample(entscheidungen, size=151)
tally(~simentscheidung)
```

```
## simentscheidung
## Nicht Positiv       Positiv
##            71            80
```

Hier wären also zufällig 80 der 151 Entscheidungen des Typ A Positiv gewesen – wenn es keinen Zusammanhang zwischen Entscheidugnstyp und Ratingentscheidung gibt.

Wir oft kommt also zufällig heraus, dass höchstens 79 der 151 Entscheidungen des Typs A (alleinige Entscheider) positiv zugeordnet werden? Simulieren wir das z. B. 1000-mal:

```
entsim <- do(1000)*tally(~sample(entscheidungen, size=151))
prop(entsim$Positiv<=79)
```

```
##  TRUE
```

```
## 0.371
```

Unter der **Nullhyothese**, dass das Ergebnis zufällig war (d. h. es gibt keinen Zusammenhang zwischen Typ und Rating), wurden in der Simulation in 37.1% der Fälle höchstens 79 positive dem Typ A zufällig zugeordnet. Dieser **p-Wert** spricht also nicht wirklich gegen das Zufallsmodell. *Hinweis:* Wir werden in späteren Kapiteln bessere Methoden kennenlernen, insbesondere auch solche die alle Informationen aus den Daten enthalten und sich nicht nur auf einen Anteilswert beziehen.

Über

```
typA <- rep(rating, c(79, 151-79))
```

erzeugen wir uns einen Vektor, der die 79 positiven und $151 - 79$ nicht positiven Urteile von Typ A (alleinige Entscheidung) enthält.

```
tally(~ typA)
```

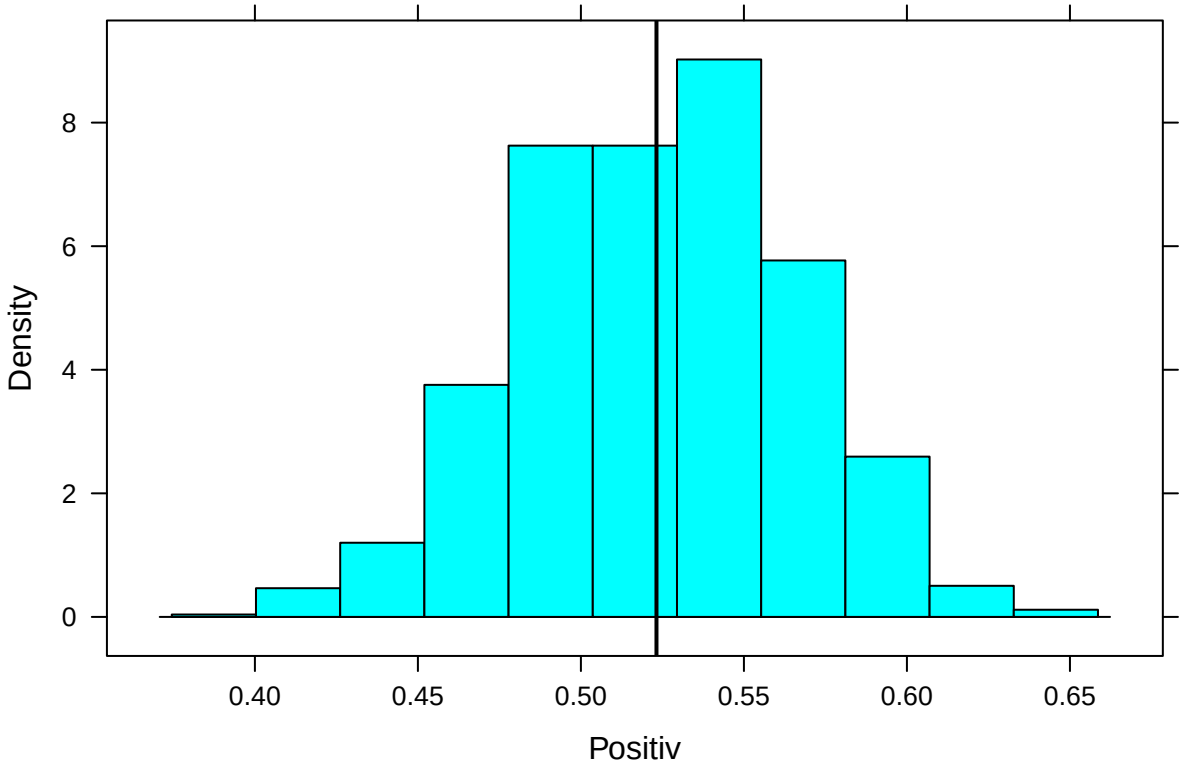
```
## typA
## Nicht Positiv       Positiv
##            72            79
```

Wenn wir jetzt diesen Vektor z. B. 1000-mal resampeln:

```
typAboot <- do(1000)*tally(~resample(typA), format="proportion")
```

erhalten wir 1000 (resampelte) Stichproben, die jeweils einen zufälligen Stichprobenanteil haben:

```
histogram(~Positiv, data=typAboot, v=79/151) # 79/151: Anteil der Originalstichprobe
```

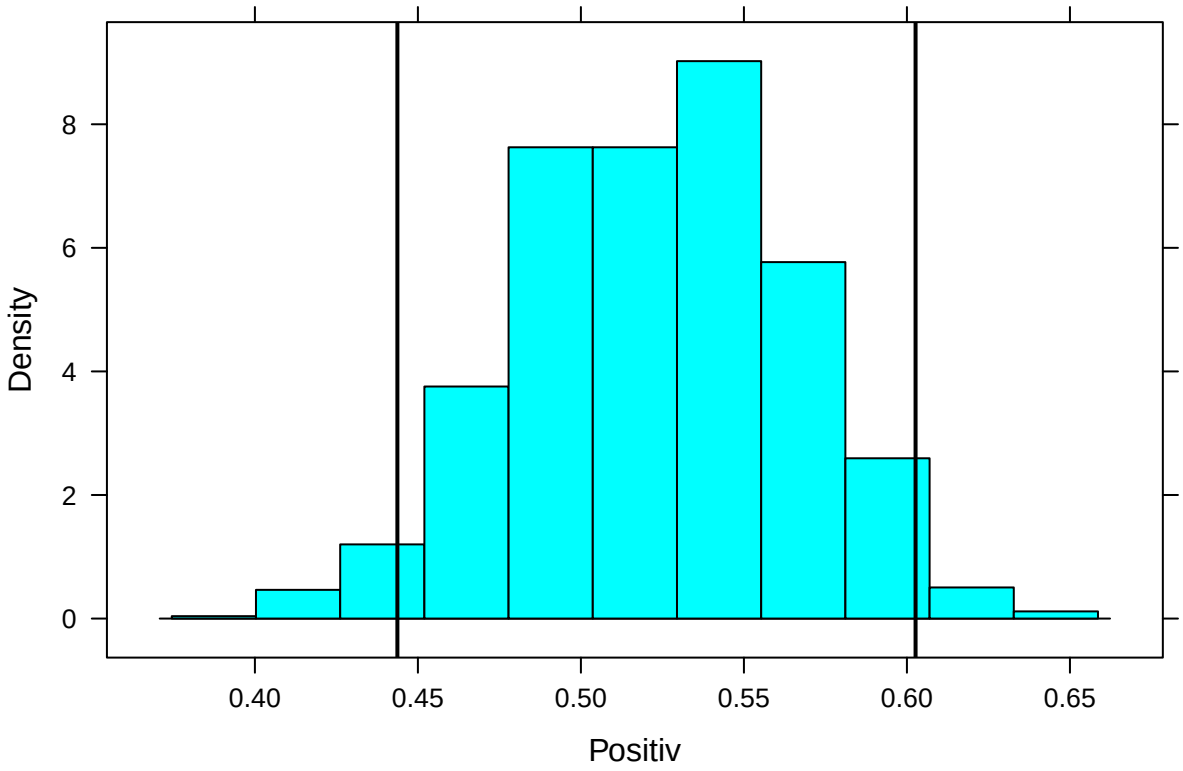


In 95% der Fälle liegt dieser zufällige Stichprobenanteil hier zwischen

```
ki <- quantile(~Positiv, data=typAboot, probs = c(0.025, 0.975))
ki
```

```
##      2.5%     97.5%
## 0.4437086 0.6026490
```

```
histogram(~Positiv, data=typAboot, v=ki)
```



Dies ist das **Nicht-parametrische Bootstrap-Konfidenzintervall**.

Übung:

3. Bestimmen Sie das 90% nicht-parametrische Bootstrap-Konfidenzintervall für eine nicht-Positive Einschätzung im Fall Entscheidungsvorlage (Typ B). Würde damit eine Nullyhpothese $p = 0.5$ zum Signifikanzniveau 10% verworfen?

Rechnen mit der Normalverteilung

Random Walk

Beim Glücksspiel ist es offensichtlich, aber auch an vielen, vielen anderen Stellen im Leben begegnen wir dem *Zufall*. Daten, Beobachtungen sind häufig Realisationen von sogenannten Zufallsvariablen. Das sind Variablen, deren Werte vom Zufall (und damit auch seinen Modellen und Gesetzen) abhängen. So

werden Aktienkurse und -renditen häufig als Random Walk aufgefasst und modelliert - häufig unter der *Annahme* einer Normalverteilung.[1]

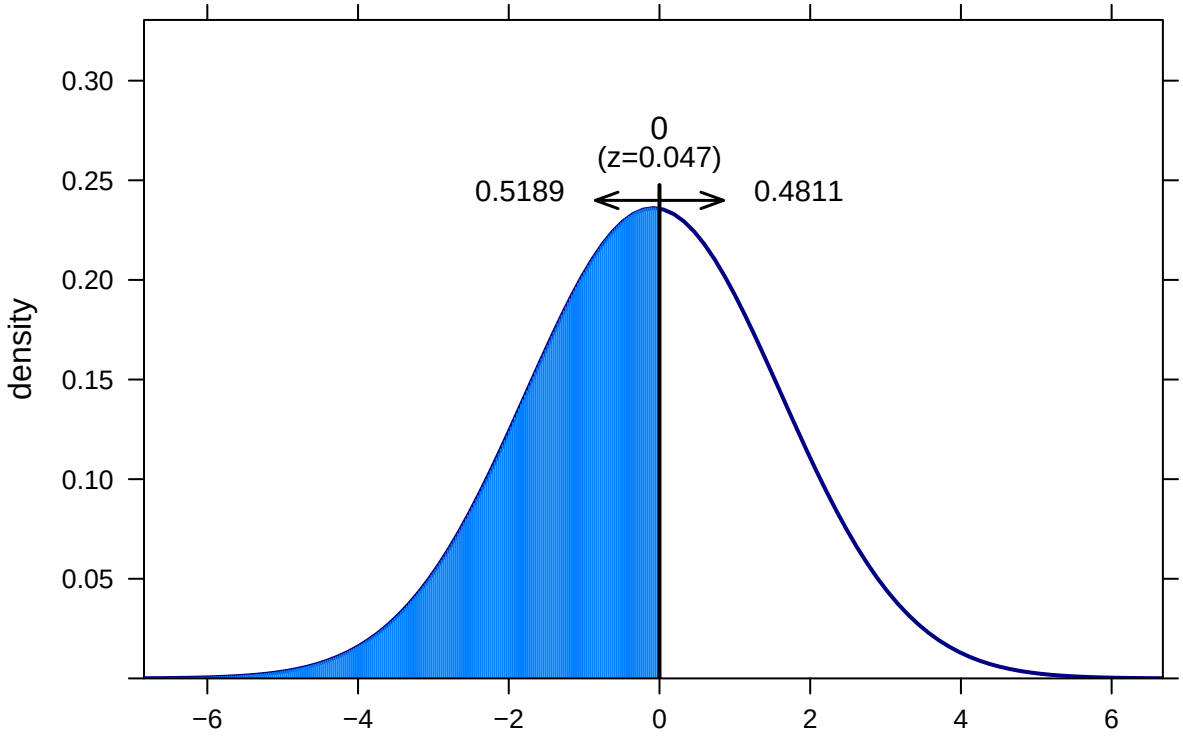
Hier drei Kennzahlen der logarithmierten Tagesrenditen von Aktienunternehmen in 2015 in %.

Anlage	AAPL	FB	GOOGL
Mittelwert	-0.08	0.11	0.15
Standardabweichung	1.69	1.62	1.77

Unter der Annahme der unabhängig, identischen Normalverteilung der logarithmierten Renditen können wir jetzt die Wahrscheinlichkeit eines Tagesverlustes der Apple Aktie (AAPL) berechnen über

```
xpnorm(0, mean=-0.08, sd=1.69 )
```

```
##
## If X ~ N(-0.08, 1.69), then
##
##  P(X <= 0) = P(Z <= 0.04734) = 0.5189
##  P(X >  0) = P(Z >  0.04734) = 0.4811
```




```
## [1] 0.5188778
```

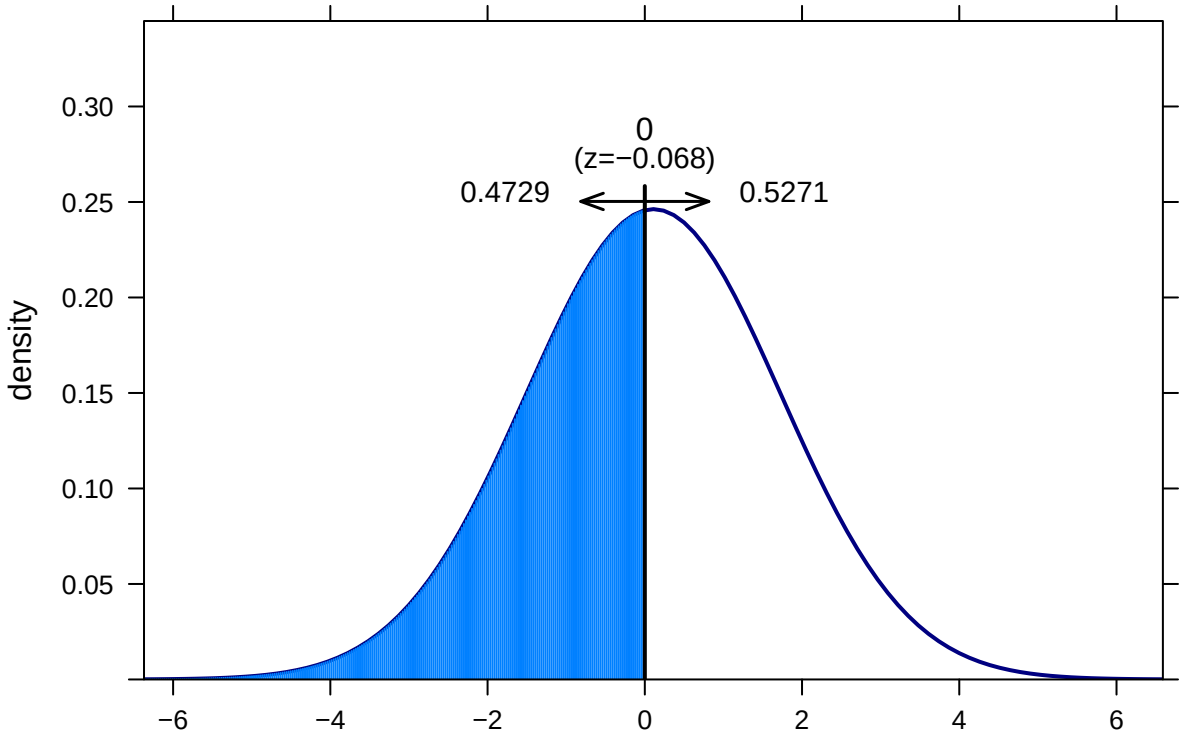
Die mosaic Funktion `xpnorm` ist eine Erweiterung der normalen R Funktion `pnorm`, die den Wert der Verteilungsfunktion an einer gegebenen Stelle zurückgibt – für jede Verteilung wird hierfür der vorgestellte Buchstabe `p` verwendet.

[1] Sowohl die Annahme einer Normalverteilung, als auch die Annahme, dass die Renditen unabhängig voneinander sind (d. h., dass keine *Autokorrelation* vorliegt) und einer identischen Verteilung folgen (hier gleiche Varianz) sind in der Praxis kritisch zu hinterfragen.

Für Facebook (FB) lag die Wahrscheinlichkeit eines Gewinns demnach bei

```
xpnorm(0, mean=0.11, sd=1.62, lower.tail = FALSE)
```

```
##
## If X ~ N(0.11, 1.62), then
##
##  P(X <= 0) = P(Z <= -0.0679) = 0.4729
##  P(X >  0) = P(Z >  -0.0679) = 0.5271
```




```
## [1] 0.5270679
```

Übung:

4. Welche der drei Aktien hat die höchste Wahrscheinlichkeit eine Tagesrendite über 2.5% zu erzielen?

Dabei wird hier immer auch die Z-Transformation, die Standardisierung, mit angegeben. Am 26.05.2015 ($r = -2.23$) betrug der z-Wert der Apple Aktie demnach bei

```
(-2.23 - (-0.08)) / 1.69
```

```
## [1] -1.272189
```

Die Tagesrendite von Apple war also 1.2721893 Standardabweichungen *unter* dem Mittelwert. Für Facebook lag die Tagesrendite bei -1.51, der z-Wert demnach bei:

```
(-1.51 - (0.11)) / 1.62
```

```
## [1] -1
```

Der 26. Mai 2015 war also auch für Facebook-Anlegerinnen kein guter Tag, aber immer noch besser als bei Apple.

Übung:

5. Die Rendite von Google am 26.05.2015 betrug -1.33. Wie hoch ist der z-Wert? Interpretieren Sie die Aussage des Ergebnisses.

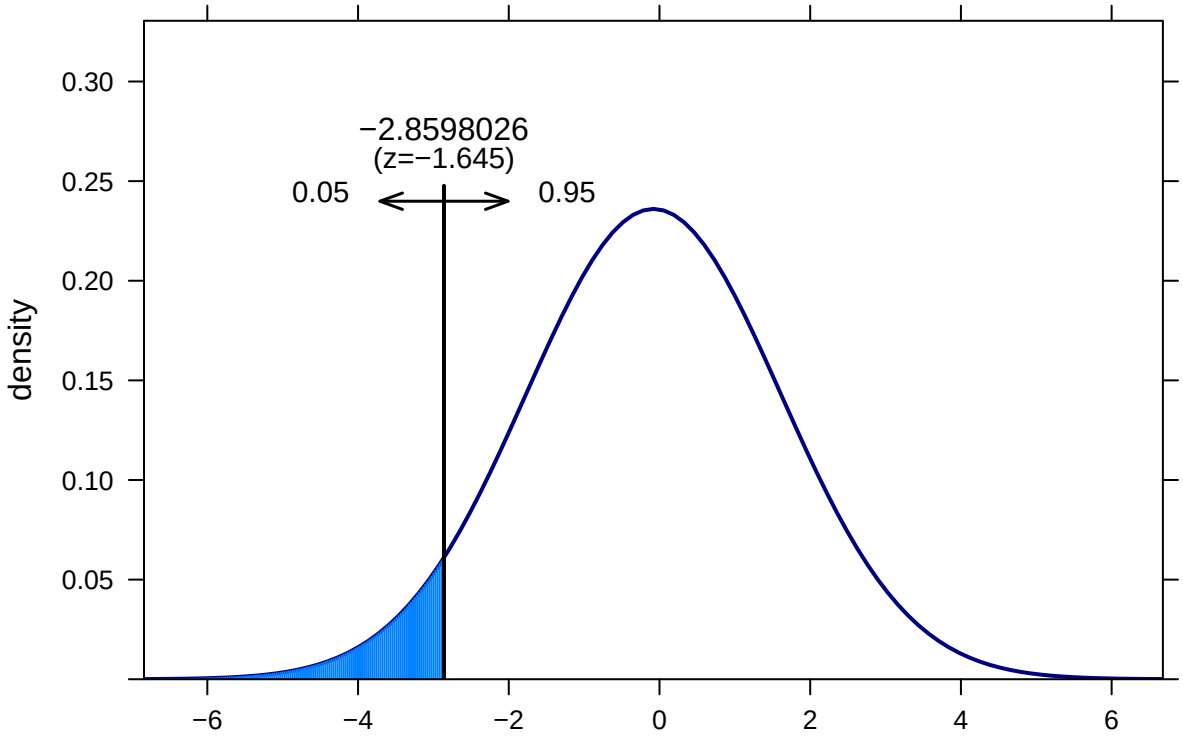
Wenn wir zu einen gegebenen Wert der Rendite den Wert der Verteilungsfunktion, d. h. den prozentualen Anteil kleiner oder gleich großer Werte suchen ($P(X \leq x)$) verwenden wir `pnorm` bzw. `xpnorm`. Wenn die Überschreitungswahrscheinlichkeit ($P(X > x)$) gesucht ist, kann zusätzlich die Option `lower.tail = TRUE` gesetzt werden, oder `1-pnorm()` gerechnet werden.

Um zu einem gegebenen Anteil (Prozentwert) den zugehörigen Wert der Rendite zu finden, wir also das Quantil suchen, dann wird `p` durch `q` ersetzt, also `qnorm` bzw. `xqnorm`.

Z. B. für die 5% schlechtesten Tage der Appleaktie

```
xqnorm(0.05, mean=-0.08, sd=1.69 )
```

```
##
## If X ~ N(-0.08, 1.69), then
##
##  P(X <= -2.859803) = 0.05
##  P(X >  -2.859803) = 0.95
```




```
## [1] -2.859803
```

Die Wahrscheinlichkeit beträgt also 5%, dass die Tagesrendite unter -2.86 liegt.

Für die Facebook Aktie gilt, dass Sie nur mit einer Wahrscheinlichkeit von 1% über 3.8786836 lag:

```
xqnorm(0.01, mean=0.11, sd=1.62, lower.tail = FALSE)
```

Übung:

6. Sie wollen Ihre Google-Aktien absichern. Wie groß ist bei einer maximalen Eintretenswahrscheinlichkeit von 1% der Tagesverlust mindestens?

Übung: Achtsamkeit

In einem Test zur Achtsamkeit *Sauer S, Lemke J, Wittmann M, Kohls N, Mochty U, and Walach H. (2012) How long is now for mindfulness meditators? Personality and Individual Differences 52(6), 750–754* konnten 34 von 38 Studienteilnehmern der Kontrollgruppe nach einer Instruktion die Dauer der Fixierung des Necker Würfels (Link/Bild) steigern.

1. Kann diese Verbesserung bei fast 90% der Personen zufällig sein? Bestimmen Sie die Wahrscheinlichkeit, dass zufällig mindestens 34 von 38 Personen eine Verbesserung erzielen mit Hilfe einer Simulation.
2. Bestimmen Sie ein nicht-paramatrisches Bootstrap-Konfidenzintervall, dass den Anteilswert der Verbesserung in 95% der Fälle überdeckt.

Übung: Intelligenzquotient

Der IQ hat nach Konstruktion einen arithmetischen Mittelwert von 100 bei einer Standardabweichung von 15.

1. Wie hoch ist der Anteil der Personen mit einem IQ von 130 oder größer?
2. Welchen IQ sollte eine Person mindestens haben, wenn Sie zu den 1% Personen mit dem höchsten IQ gehören will?

Literatur

- David M. Diez, Christopher D. Barr, Mine Çetinkaya-Rundel (2014): *Introductory Statistics with Randomization and Simulation*, https://www.openintro.org/stat/textbook.php?stat_book=isrs, Kapitel 2
- Nicholas J. Horton, Randall Pruim, Daniel T. Kaplan (2015): Project MOSAIC Little Books *A Student's Guide to R*, https://github.com/ProjectMOSAIC/LittleBooks/raw/master/StudentGuide/MOSAIC-StudentGuide.pdf, Kapitel 3.5, 3.6
- Chester Ismay, Albert Y. Kim (2017): ModernDive – An Introduction to Statistical and Data Sciences via R, https://ismayc.github.io/moderndiver-book/
- Maike Luhmann (2015): *R für Einsteiger*, Kapitel 12
- Andreas Quatember (2010): *Statistik ohne Angst vor Formeln*, Kapitel 2, 3.1-3.3, 3.13
- Daniel Wollschläger (2014): *Grundlagen der Datenanalyse mit R*, Kapitel 5, 11

Lizenz

Versionshinweise:

- Datum erstellt: 2017-06-30
- R Version: 3.4.0
- `mosaic` Version: 0.14.4

Kapitel 3: Einführung Inferenz kategoriale Werte

Globaler Index der Religiösität und Atheismus

Im Jahre 2012 führte das WIN/Gallup International Institut in 57 Ländern eine Untersuchung zur Religiosität durch. Die Pressemitteilung zur Studie finden Sie hier.

Dabei wurde die Frage gestellt: *Unabhängig davon, ob Sie an Gottesdiensten teilnehmen oder nicht ("attend place of worship"), würden Sie sagen, dass Sie ein religiöser Mensch sind, eine nicht religiöse Person oder ein überzeugter Atheist?*

Die Befragten hatten dabei drei Antwortmöglichkeiten: Religiöser Mensch ("A religious person"), Nicht religiöser Mensch ("Not a religious person"), und Atheist ("A convinced atheist"). Die Befragten klassifizierten sich, es wurden als kategorielle (nominale) Daten (`factor`) erzeugt.

Übung:

1. Handelt es sich bei den im Bericht angegeben Kennzahlen um *Stichprobenstatistiken* oder um *Populationsparameter*?
2. Um die Ergebnisse der Studie zu verallgemeinern, also auf die Gesamtbevölkerung zu schließen, welche Annahmen müssen dafür erfüllt sein und klingen diese hier plausibel erfüllt?

Ein Teil der Daten kann direkt von OpenIntro als R Datensatz heruntergeladen werden, und anschließend in R eingelesen:

```
meine_url <- "http://www.openintro.org/stat/data/atheism.RData"
load(url(meine_url)) # Einlesen
```

Einen Überblick erhält man wie immer über:

```
str(atheism) # Datenstruktur
```

```
## 'data.frame':    88032 obs. of  3 variables:
##  $ nationality: Factor w/ 57 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ response   : Factor w/ 2 levels "atheist","non-atheist": 2 2 2 2 2 2 2 2 2 2 ...
##  $ year       : int  2012 2012 2012 2012 2012 2012 2012 2012 2012 2012 ...
```

```
head(atheism) # Erste Beobachtungen
```

```
##   nationality    response year
## 1 Afghanistan non-atheist 2012
## 2 Afghanistan non-atheist 2012
## 3 Afghanistan non-atheist 2012
## 4 Afghanistan non-atheist 2012
## 5 Afghanistan non-atheist 2012
## 6 Afghanistan non-atheist 2012
```

```
tail(atheism) # letzte Beobachtungen
```

```
##       nationality    response year
## 88027     Vietnam non-atheist 2005
## 88028     Vietnam non-atheist 2005
```

```
## 88029     Vietnam non-atheist 2005
## 88030     Vietnam non-atheist 2005
## 88031     Vietnam non-atheist 2005
## 88032     Vietnam non-atheist 2005
```

Zur Analyse wird wieder das Paket mosaic verwendet:

```
library(mosaic)
```

Inferenz eines Anteilswerts

In Tabelle 6 der Pressemitteilung wird der Anteil der Atheisten für Deutschland mit 15% angegeben. Dies ist eine *Statistik* der Stichprobe, nicht der Parameter der *Population*. Es wird also die Frage beantwortet "Wie hoch ist der Anteil der Atheisten in der Stichprobe?". Um die Frage "Wie hoch ist der Anteil der Atheisten in der Population?" zu beantworten, muss von der Stichprobe auf die Population geschlossen werden, d. h., es wird z. B. der Anteilswert *geschätzt*.

Der folgende Befehl filtert den Datensatz auf das Ergebnis für Deutschland im Jahr 2012, d. h., es werden nur die gewünschten Zeilen im Datensatz belassen:

```
de12 <- filter(atheism, nationality == "Germany", year == "2012")
```

Die *Punktschätzung* des Anteilswertes der Atheisten für Deutschland im Jahr 2012 liegt dann bei

```
pdach <- tally(~response, data=de12, format='proportion')["atheist"]
pdach
```

```
##   atheist
## 0.1494024
```

also bei 15%.

Um jetzt ein 95% Konfidenzintervall für den Populationsparameter zu konstruieren (*Bereichsschätzung*) muss der Standardfehler *se* bestimmt werden, hier:

```
n <- nrow(de12) # Anzahl Beobachtungen
se <- sqrt( pdach * (1-pdach) / n)
se
```

```
##    atheist
## 0.01591069
```

Der Standardfehler, d. h., die Standardabweichung des Anteilswertes liegt hier also bei 1.59%. Zusammen mit dem 2,5% und 97,5% Quantil der Standardnormalverteilung ergibt sich folgendes Konfidenzintervall:

```
pdach + qnorm(c(0.025, 0.975)) * se
```

```
## [1] 0.1182180 0.1805868
```

Da der Populationsparameter unbekannt aber nicht zufällig ist, werden die 95% auch als *Überdeckungswahrscheinlichkeit* bezeichnet.

In `mosaic` besteht die Möglichkeit, Punkt- und Bereichsschätzungen mit dem Befehl `prop.test` durchzuführen. Sofern kein Wert für `p` angegeben wird lautet die Nullyhpothese $H_0 : p = 0.5$.

```
prop.test(~response, data=de12)
```

```
##
##  1-sample proportions test with continuity correction
##
## data:  de12$response  [with success = atheist]
## X-squared = 245.42, df = 1, p-value < 2.2e-16
## alternative hypothesis: true p is not equal to 0.5
## 95 percent confidence interval:
##  0.1199815 0.1843172
## sample estimates:
##         p
## 0.1494024
```

Beachte, dass das Konfidenzintervall, anders als bei der *Approximation* über die Normalverteilung hier nicht symmetrisch um den Punktschätzer ist.

Übung:

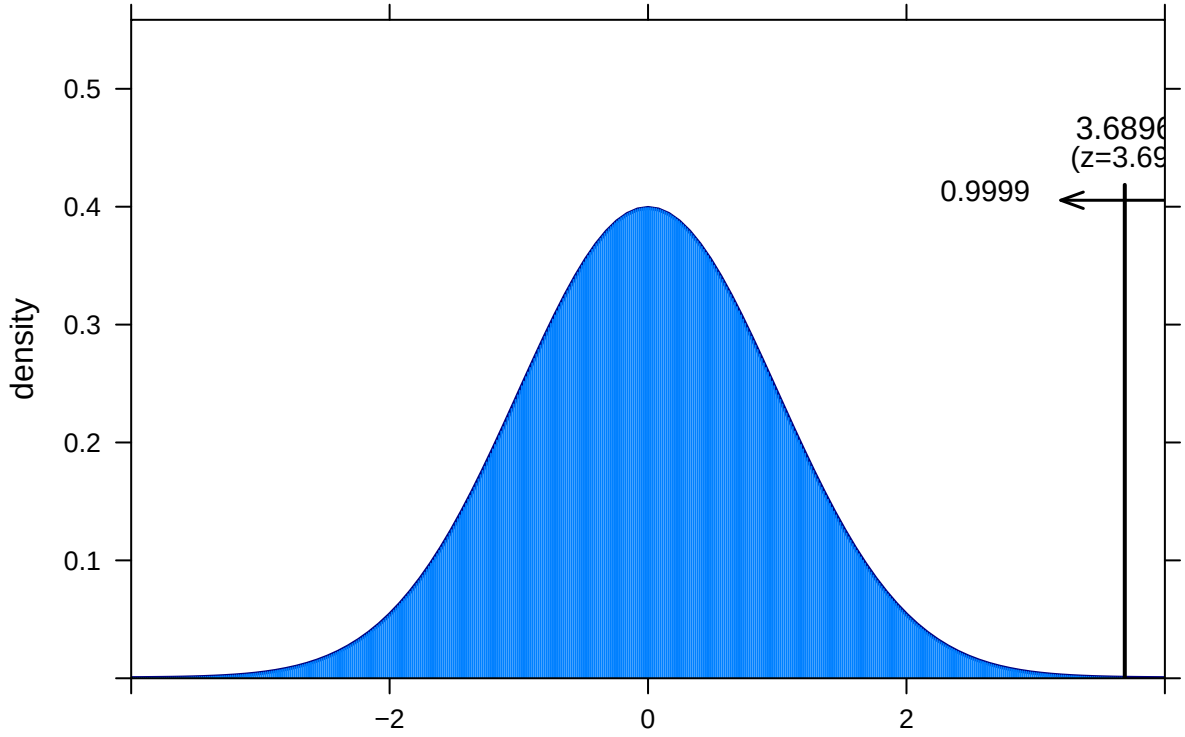
3. Bei annähernd gleicher Stichprobengröße liegt der Anteil der Atheisten in Saudi Arabien bei 5%. Wie verändert sich der Standardfehler und damit die Breite des Konfidenzintervalls?
4. Der Anteil der Atheisten in Südkorea liegt in etwa ebenfalls bei 15%, allerdings liegen die Daten von 1523 Befragten vor. Wie verändert sich der Standardfehler und damit die Breite des Konfidenzintervalls?

Um für Deutschland die Nullhypothese "Der Anteil der Atheisten liegt nicht über 10%" gegen die Alternativhypothese (Forschungshypothese) "Der Anteil der Atheisten liegt über 10%" können entweder wieder Simulations- und Resamplingtechniken verwendet werden oder die Approximation durch die Normalverteilung :

```
se0 <- sqrt( (0.1 * (1-0.1)) /n)
z <- (pdach - 0.10) / se0
xpnorm(z, lower.tail = FALSE)
```

```
##
## If X ~ N(0, 1), then
##
```

```
##  P(X <= 3.69) = P(Z <= 3.69) = 0.9999
##  P(X >  3.69) = P(Z >  3.69) = 0.0001123
```




```
##      atheist
## 0.0001123062
```

Der *p-Wert* liegt also bei 0.0112%, die Nullhypothese wird also zum Signifikanzniveau von 5% verworfen.

Auch hier direkt über `prop.test`:

```
prop.test(~response, p=0.1, alternative="greater", data=de12)
```

```
##
##  1-sample proportions test with continuity correction
##
## data:  de12$response  [with success = atheist]
## X-squared = 13.07, df = 1, p-value = 0.0001501
## alternative hypothesis: true p is greater than 0.1
## 95 percent confidence interval:
##  0.1241944 1.0000000
## sample estimates:
##         p
## 0.1494024
```

Je nach Alternativhypothese ergeben sich unterschiedliche Tests:

- `alternative='two.sided'`: ungerichtet, d. h. zweiseitig: $H_0 : p = 0.1$ gegen $H_A : p \neq 0.1$
- `alternative='less'`: gerichtet, d. h. einseitig: $H_0 : p \geq 0.1$ gegen $H_A : p < 0.1$
- `alternative='greater'`: gerichtet, d. h. einseitig: $H_0 : p \leq 0.1$ gegen $H_A : p > 0.1$

Differenz zweier Anteilswerte

In den Daten liegen außerdem die Ergebnisse aus 2005 vor:

```
de05 <- filter(atheism, nationality == "Germany" & year == "2005")
```

Im Jahre 2005 lag der Anteil der Atheisten in Deutschland bei

```
tally(~response, data=de05, format="proportion")
```

```
## response
##     atheist non-atheist
##  0.09960159  0.90039841
```

Der Aneil lag also bei unter 10% – in der *Stichprobe*! Können wir daraus auf eine Erhöhung des Anteils von 2005 zu 2012 in der *Population* schließen?

```
# 2012
a12 <- tally(~response, data=de12)["atheist"] # Anzahl Atheisten 2012
n12 <- nrow(de12) # Anzahl Studienteilnehmer 2012
p12 <- a12/n12 # Anteil Atheisten 2012
# 2005
a05 <- tally(~response, data=de05)["atheist"] # Anzahl Atheisten 2005
n05 <- nrow(de05) # Anzahl Studienteilnehmer 2005
p05 <- a05/n05 # Anteil Atheisten 2005
# Punktschätzer Differenz Population
pdiff <- p12-p05
pdiff
```

```
##   atheist
## 0.0498008
```

```
# Pooling zur Berechnen Standardfehler unter H_0
ppool <- (a12 + a05)/(n12+n05)
ppool
```

```
##  atheist
## 0.124502
```

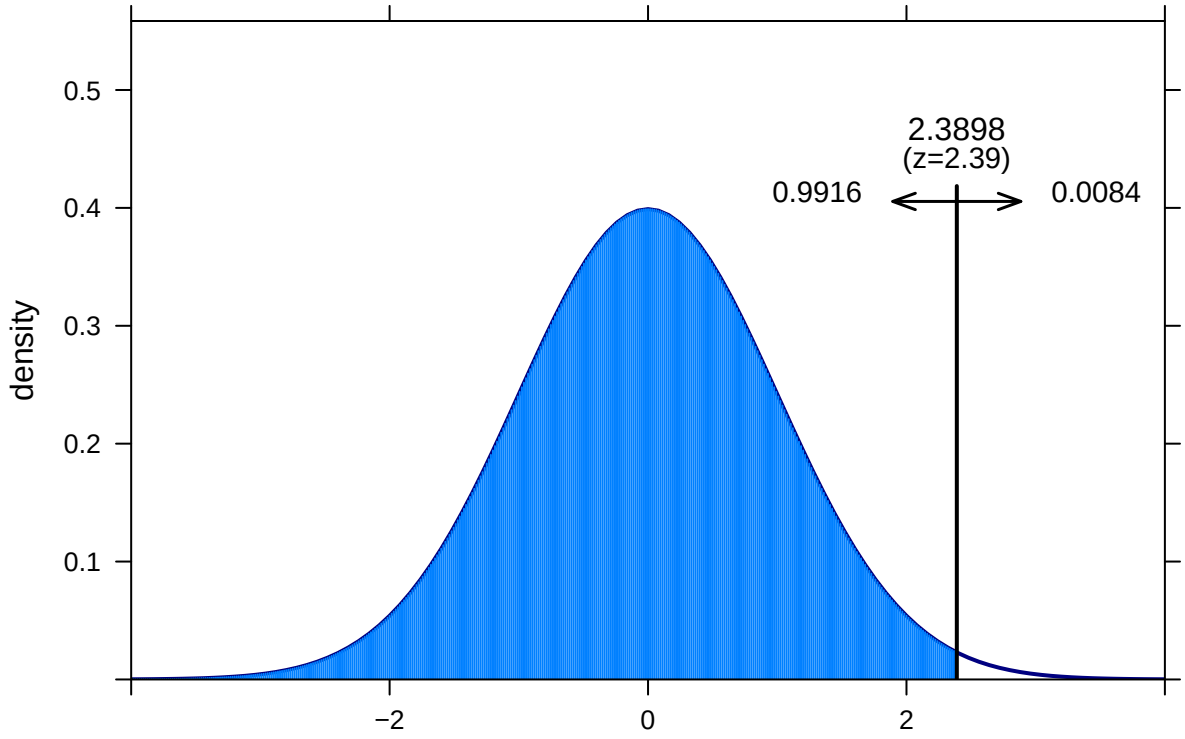
```
# Standardfehler se
se0 <- sqrt( (ppool * (1-ppool) / n12) + (ppool * (1-ppool) / n05) )
se0
```

```
##   atheist
## 0.0208391
```

```
# z-Wert z
z <- (pdiff - 0)/se0
```

```
# p-Wert
xpnorm(z, lower.tail = FALSE)
```

```
##
## If X ~ N(0, 1), then
##
##   P(X <= 2.39) = P(Z <= 2.39) = 0.9916
##   P(X >  2.39) = P(Z >  2.39) = 0.008429
```




```
##     atheist
## 0.008429297
```

Der p-Wert ist klein und das Ergebnis damit *statistisch signifikant*. Die Wahrscheinlichkeit *zufällig* eine solche Erhöhung der Anteilswerte zu beobachten ist also gering – wenn die H_0 gilt! d. h. es wird auf eine Veränderung des Anteilswertes in der *Population* geschlossen.

Auch dies kann direkt durch den Befehl `prop.test` gestestet werden. Dazu wird zunächst ein gemeinsamer Datensatz erzeugt:

```
de <- filter(atheism, nationality == "Germany")
```

und anschließend geteset:

```
prop.test(response~year, alternative="less", data=de)
```

```
##
##  2-sample test for equality of proportions with continuity
##  correction
##
```

```
## data:  tally(response ~ year)
## X-squared = 5.2633, df = 1, p-value = 0.01089
## alternative hypothesis: less
## 95 percent confidence interval:
##  -1.00000000 -0.01362913
## sample estimates:
##     prop 1     prop 2
## 0.09960159 0.14940239
```

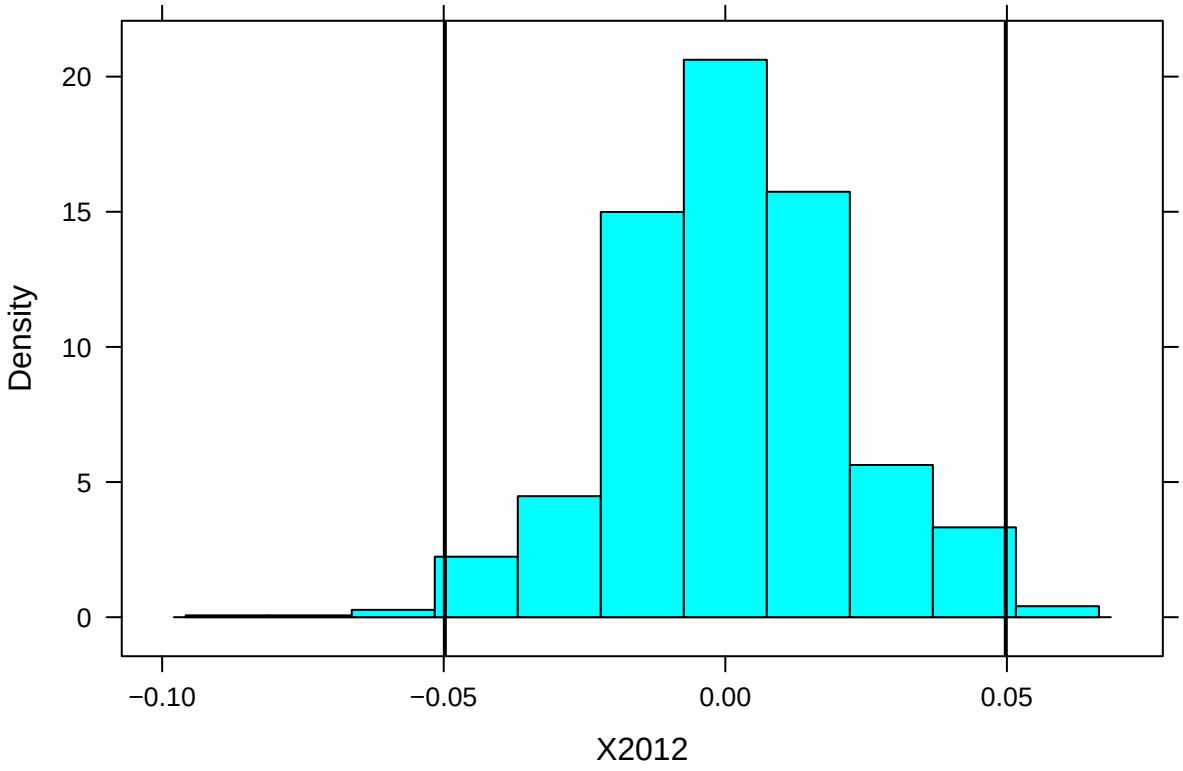
Hier wird die Alternativhypothese `less` verwendet: $H_0 : p_1 \geq p_2$, gegen $H_A : p_1 < p_2$.

Exkurs: Mit dem Paket mosaic können Sie das auch einfach über Permutationen testen, indem das Erhebungsjahr zufällig gesampelt wird, wobei hier ungerichtet, d. h., zweiseitig getestet wird. Mit anderen Worten ist sowohl eine Erhöhung als auch ein Verringerung des Anteils in der Alternativhypothese möglich, daher werden die Absolutwerte der Differenz (`abs()`) verwendet:

```
# Beobachtete Differenz
pdiff <- diff(tally( ~ response | year, data=de, format="proportion")["atheist",])
pdiff
```

```
##      2012
## 0.0498008
```

```
# Zufallszahlengenerator setzen (Reproduzierbarkeit!)
set.seed(1896)
# 1000-mal das Jahr permutieren: Nullhypothese kein Unterschied
pdiff.null <- do(1000) * diff(tally( ~ response | shuffle(year), data=de,
                                    format="proportion")["atheist",])
# Histogramm
histogram(~ X2012, data=pdiff.null, v=c(pdiff, -pdiff))
```




```
# p-Wert
prop(abs(pdiff.null$X2012) >= abs(pdiff))
```

```
## TRUE
## 0.02
```

Übung:

5. Überprüfen Sie für das Jahr 2012, ob es eine zum Niveau 5% signifikante Differenz zwischen den Anteil der Atheisten in Deutschland und den Niederlanden (`Netherlands`) in der Population gibt.
6. Überprüfen Sie für das Jahr 2012, ob es eine zum Niveau 5% signifikante Differenz zwischen den Anteil der Atheisten in Deutschland und Polen (`Poland`) in der Population gibt.

Chi-Quadrat Unabhängigkeitstest

Soll allgemein der Zusammenhang zwischen zwei kategoriellen (nominalen) Variablen untersucht werden, wird der Chi^2-Unabhängigkeitstest verwendet. Diese testet die Nullhypothese der Unabhängigkeit gegen die Alternativhypothese des Zusammenhangs. Im vorliegenden Datensatz können wir z. B. testen, ob die Verteilung (Anteil) der Atheisten in den teilnehmenden G7 Ländern gleich ist:

```
G7 <- c("United States", "Canada", "Germany", "France", "Italy", "Japan")
G7.12 <- filter(atheism, nationality %in% G7 & year == 2012)
G7.12 <- droplevels(G7.12)
```

```
G7atheist <- tally(response ~ nationality, data = G7.12)
G7atheist
```

```
##              nationality
## response      Canada France Germany Italy Japan United States
##   atheist         90    485      75    79   372            50
##   non-atheist    912   1203     427   908   840           952
```

(Der Befehl `droplevels` sorgt dafür, dass die nicht mehr benötigten Ausprägungen der kategoriellen Variablen (`factor`) gelöscht werden.)

Der Test selber erfolgt in mosaic über `xchisq.test`, d. h.:

```
xchisq.test(G7atheist)
```

```
##
##  Pearson's Chi-squared test
##
## data:  x
## X-squared = 504.04, df = 5, p-value < 2.2e-16
##
##     90      485       75       79      372       50
## ( 180.40) ( 303.91) (  90.38) ( 177.70) ( 218.21) ( 180.40)
## [ 45.30] [107.91] [  2.62] [ 54.82] [108.39] [ 94.26]
## <-6.73>  <10.39>  <-1.62>  <-7.40>  <10.41>  <-9.71>
##
##    912     1203      427      908      840      952
## ( 821.60) (1384.09) ( 411.62) ( 809.30) ( 993.79) ( 821.60)
## [  9.95] [ 23.69] [  0.57] [ 12.04] [ 23.80] [ 20.70]
## < 3.15>  <-4.87>  < 0.76>  < 3.47>  <-4.88>  < 4.55>
##
## key:
##  observed
##  (expected)
##  [contribution to X-squared]
##  <Pearson residual>
```

Der Wert der Teststatistik Chi² liegt bei 504.04, die Anzahl der Freiheitsgrade ("degrees of freedom", df) bei 5, der p-Wert ist sehr klein, die Nullhypothese der Unabhängigkeit von Nationalität und Verteilung Atheismus wird für die *Population* verworfen.

Die Formel zur Berechnung der χ^2 Teststatistik lautet:

$$\chi^2 = \sum_{i=1}^{k} \sum_{j=1}^{m} \frac{(h_{ij} - e_{ij})^2}{e_{ij}}$$

mit $e_{ij} = \frac{h_{i\cdot} h_{\cdot j}}{n}$ wobei für die Zeilensumme $h_{i\cdot} = \sum_{j=1}^{m} h_{ij}$ und für die Spaltensumme $h_{\cdot j} = \sum_{i=1}^{k} h_{ij}$ gilt.

Mit diesen Daten kann auch über den Befehl `pchisq` der p-Wert berechnet werden:

```
pchisq(504, 5, lower.tail = FALSE)
```

```
## [1] 1.093548e-106
```

Übung:

7. Gibt es einen Zusammenhang zwischen der Verteilung des Atheismus und der Nationalität im Jahr 2012 innerhalb der afrikanischen Länder `c('Nigeria','Kenya', 'Tunisia', 'Ghana', 'Cameroon', 'South Sudan')`?

Übung: Trinkgelddaten

Wir werden jetzt den *tips* Datensatz aus *Bryant, P. G. and Smith, M (1995) Practical Data Analysis: Case Studies in Business Statistics. Homewood, IL: Richard D. Irwin Publishing* weiter analysieren.

Sofern noch nicht geschehen, können Sie diesen als `csv` Datei herunterladen:

```
download.file("https://goo.gl/whKjnl", destfile = "tips.csv")
```

Das Einlesen erfolgt, sofern die Daten im aktuellen Verzeichnis liegen, über:

```
tips <- read.csv2("tips.csv")
```

Tipp: Wenn Sie nicht mehr wissen wo die Daten liegen: statt `tips.csv` den Befehl `file.choose()` als Argument für die Funktion `read.csv2` verwenden.

1. Bestimmen Sie ein 90% Konfidenzintervall für den Anteil der Raucher (`smoker`) in der Population.
2. Testen Sie zum Niveau 5%, ob sich der Anteil der Raucher in der Population beim Lunch von dem beim Dinner (`time`) unterscheidet.
3. Gibt es einen Zusammenhang zwischen Rauchen und Wochentag (`day`)?

Literatur

- David M. Diez, Christopher D. Barr, Mine Çetinkaya-Rundel (2014): *Introductory Statistics with Randomization and Simulation*, https://www.openintro.org/stat/textbook.php?stat_book=isrs, Kapitel 3
- Nicholas J. Horton, Randall Pruim, Daniel T. Kaplan (2015): Project MOSAIC Little Books *A Student's Guide to R*, https://github.com/ProjectMOSAIC/LittleBooks/raw/master/StudentGuide/MOSAIC-StudentGuide.pdf, Kapitel 4.2, 4.3, 6.3
- Maike Luhmann (2015): *R für Einsteiger*, Kapitel 18.1
- Andreas Quatember (2010): *Statistik ohne Angst vor Formeln*, Kapitel 3.4, 3.6, 3.8

- Daniel Wollschläger (2014): *Grundlagen der Datenanalyse mit R*, Kapitel 10.2

Lizenz

Versionshinweise:

- Datum erstellt: 2017-06-30
- R Version: 3.4.0
- `mosaic` Version: 0.14.4

Kapitel 4: Einführung Inferenz metrische Werte

t-Test für eine Stichprobe

Der B3 Datensatz *Heilemann, U. and Münch, H.J. (1996): West German Business Cycles 1963-1994: A Multivariate Discriminant Analysis. CIRET–Conference in Singapore, CIRET–Studien 50.* enthält Quartalsweise Konjunkturdaten aus (West-)Deutschland von 1955-1994.

Er kann von https://goo.gl/0YCEHf heruntergeladen werden:

```
download.file("https://goo.gl/0YCEHf", destfile = "B3.csv")
```

Anschließend können die Daten in R eingelesen werden:

```
B3 <- read.csv2("B3.csv")
str(B3) # Datenstruktur
```

```
## 'data.frame':    157 obs. of  14 variables:
##  $ PHASEN  : int  2 2 3 3 3 3 3 3 3 ...
##  $ BSP91JW : num  10.53 10.6 9.21 5.17 4.93 ...
##  $ CP91JW  : num  9.31 12.66 6.55 7.87 8.6 ...
##  $ DEFRATE : num  0.05 0.06 0.05 0.05 0.04 0.04 0.04 0.03 0.03 0 ...
##  $ EWAJW   : num  5.7 5.2 4.8 3.3 2.1 3.2 2.5 2.7 3 0.3 ...
##  $ EXIMRATE: num  3.08 1.96 2.82 3.74 4.16 2.9 3.65 4.57 4.37 2.89 ...
##  $ GM1JW   : num  11.15 11.03 10.04 8.33 7.69 ...
##  $ IAU91JW : num  23.56 12.72 11.52 0.85 -2.08 ...
##  $ IB91JW  : num  14.69 24.95 14.9 7.55 3.23 ...
##  $ LSTKJW  : num  3 2.36 3.39 5.3 6.91 1.03 3.73 6.2 4.12 7.94 ...
##  $ PBSPJW  : num  2.89 2.59 3.01 3.03 3.46 1.95 3.18 3.98 3.29 5.63 ...
```

```
##  $ PCPJW   : num  1.91 2.2 3.09 2.08 1.48 1.65 1.47 3.29 3.59 4.19 ...
##  $ ZINSK   : num  6.27 4.6 6.19 6.71 7.1 4.96 5.21 4.83 4.5 3.83 ...
##  $ ZINSLR  : num  3.21 3.54 3.22 3.37 3.14 4.95 3.82 3.09 3.91 1.47 ...
```

```
head(B3); tail(B3)
```

```
##   PHASEN BSP91JW CP91JW DEFRATE EWAJW EXIMRATE GM1JW IAU91JW IB91JW LSTKJW
## 1      2   10.53   9.31    0.05   5.7     3.08 11.15   23.56  14.69   3.00
## 2      2   10.60  12.66    0.06   5.2     1.96 11.03   12.72  24.95   2.36
## 3      3    9.21   6.55    0.05   4.8     2.82 10.04   11.52  14.90   3.39
## 4      3    5.17   7.87    0.05   3.3     3.74  8.33    0.85   7.55   5.30
## 5      3    4.93   8.60    0.04   2.1     4.16  7.69   -2.08   3.23   6.91
## 6      3    8.39   5.62    0.04   3.2     2.90  6.62   -3.76  14.58   1.03
##   PBSPJW PCPJW ZINSK ZINSLR
## 1   2.89  1.91  6.27   3.21
## 2   2.59  2.20  4.60   3.54
## 3   3.01  3.09  6.19   3.22
## 4   3.03  2.08  6.71   3.37
## 5   3.46  1.48  7.10   3.14
## 6   1.95  1.65  4.96   4.95
```

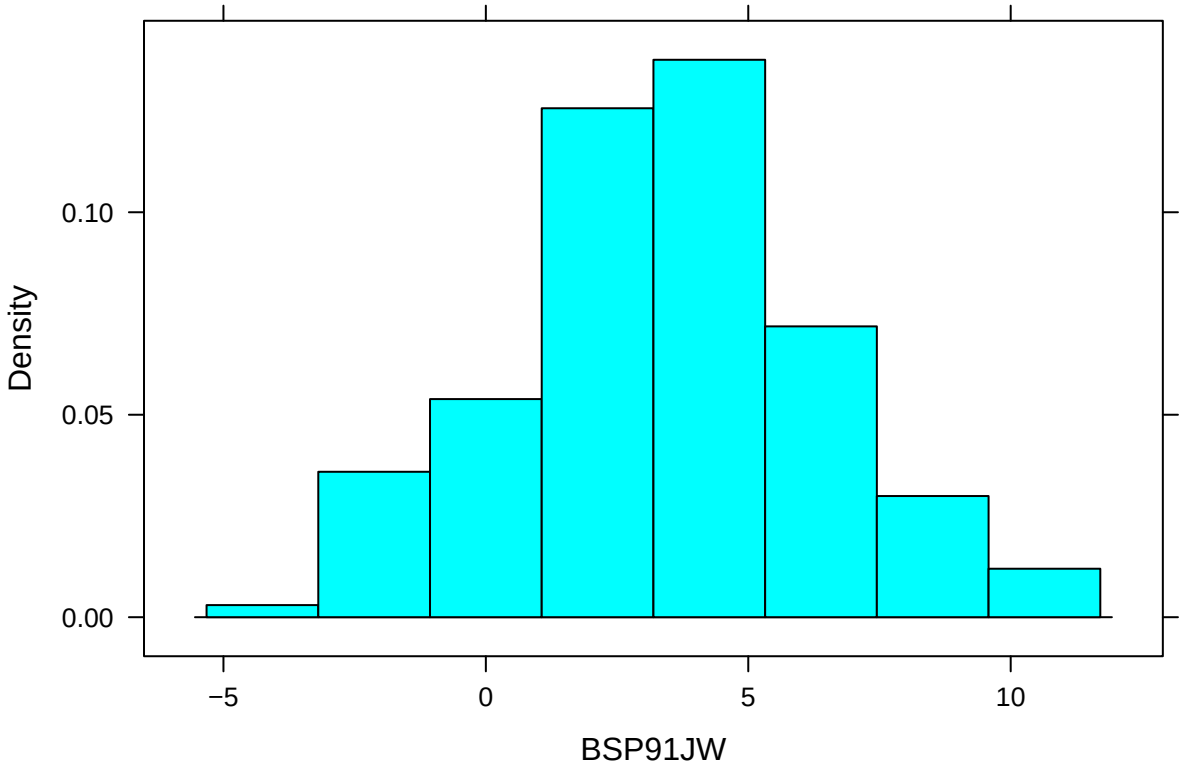
```
##     PHASEN BSP91JW CP91JW DEFRATE EWAJW EXIMRATE GM1JW IAU91JW IB91JW
## 152      3   -1.27   1.29   -4.87 -1.97     6.03  9.79  -18.29   1.73
## 153      3   -2.13  -0.57   -2.98 -2.05     7.59  0.72  -15.82  -3.23
## 154      3    1.39   2.33   -2.86 -1.84     7.49 11.33  -10.59   4.62
## 155      4    1.63   0.64    1.20 -1.58     7.75 11.38   -4.90   3.62
## 156      1    1.40   0.57   -3.56 -1.34     5.58  9.53   -0.76   2.19
## 157      1    1.83  -0.08   -2.22 -0.93     7.50 15.20    2.75   6.12
##     LSTKJW PBSPJW PCPJW ZINSK ZINSLR
## 152   1.08   2.73  2.98  6.83   3.55
## 153   1.67   2.67  3.31  6.35   3.05
## 154  -0.12   2.66  2.94  5.88   3.17
## 155  -1.81   1.77  2.58  5.29   4.82
## 156  -1.54   1.85  2.60  5.01   5.27
## 157  -0.92   1.79  2.49  5.28   5.62
```

Zur Analyse wird wieder das Paket mosaic verwendet:

```
library(mosaic)
```

Wie sah die (jährliche, quartalsweise) Entwicklung des Bruttosozialproduktes (`BSP91JW`) in dem Zeitraum (1955-1994) aus?

```
histogram( ~ BSP91JW, data = B3)
```



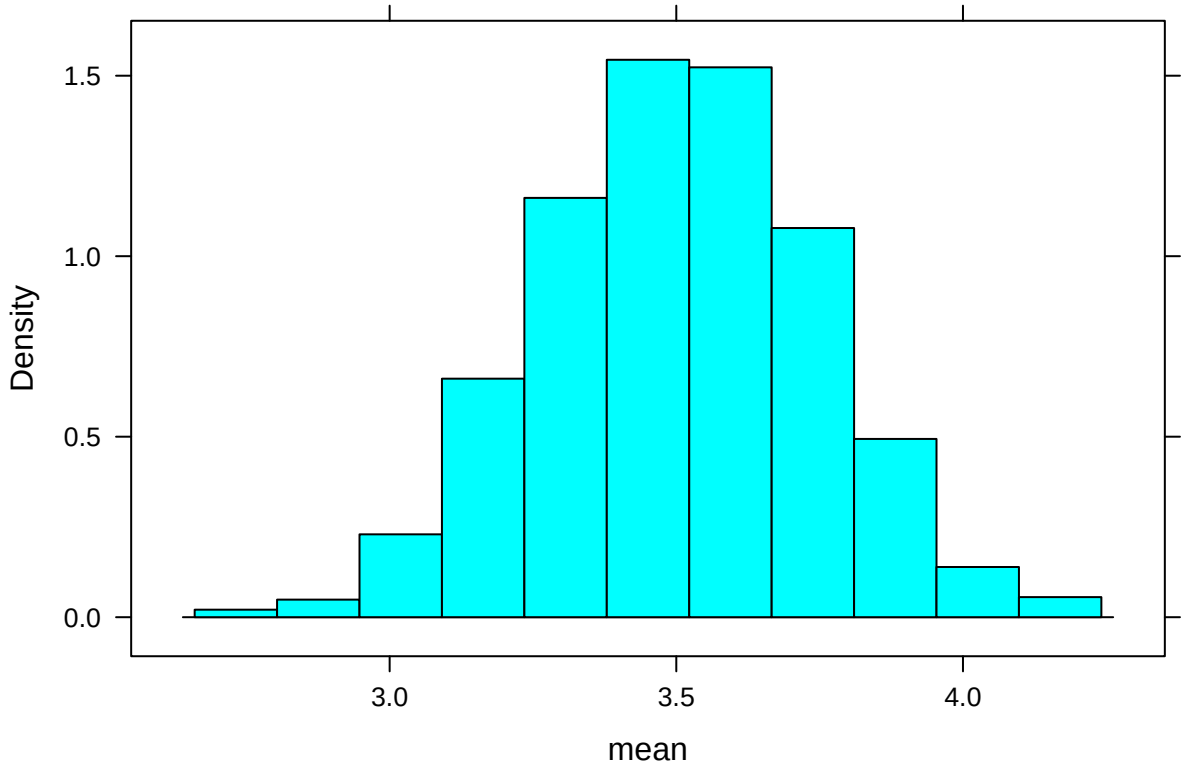

```
favstats( ~ BSP91JW, data = B3)
```

```
##     min    Q1 median    Q3   max     mean       sd   n missing
##  -3.78 1.58   3.59 5.17 11.12 3.498662 2.974273 157       0
```

Liefern die Daten Belege für die Forschungsthese, dass der Mittelwert nicht zufällig $> 0\%$ ist?

Zunöcht ein Konfidenzintervall für den unbekannten Wert μ:

```
BootBSP91JW <- do(1000) * mean( ~ BSP91JW, data = resample(B3))
histogram( ~ mean, data = BootBSP91JW)
```



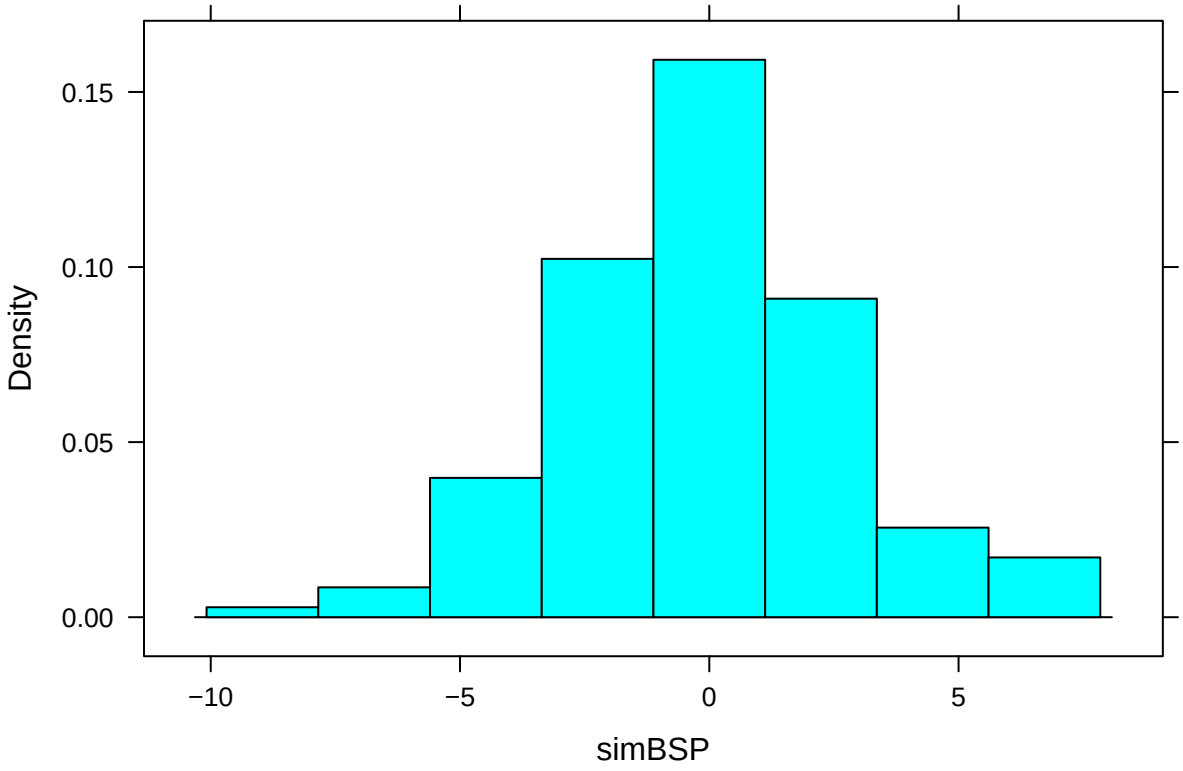

```
quantile( ~ mean, data = BootBSP91JW, probs=c(0.025, 0.975))
```

```
##     2.5%    97.5%
## 3.035919 3.963293
```

Hier ist die 0 schon einmal nicht enthalten.

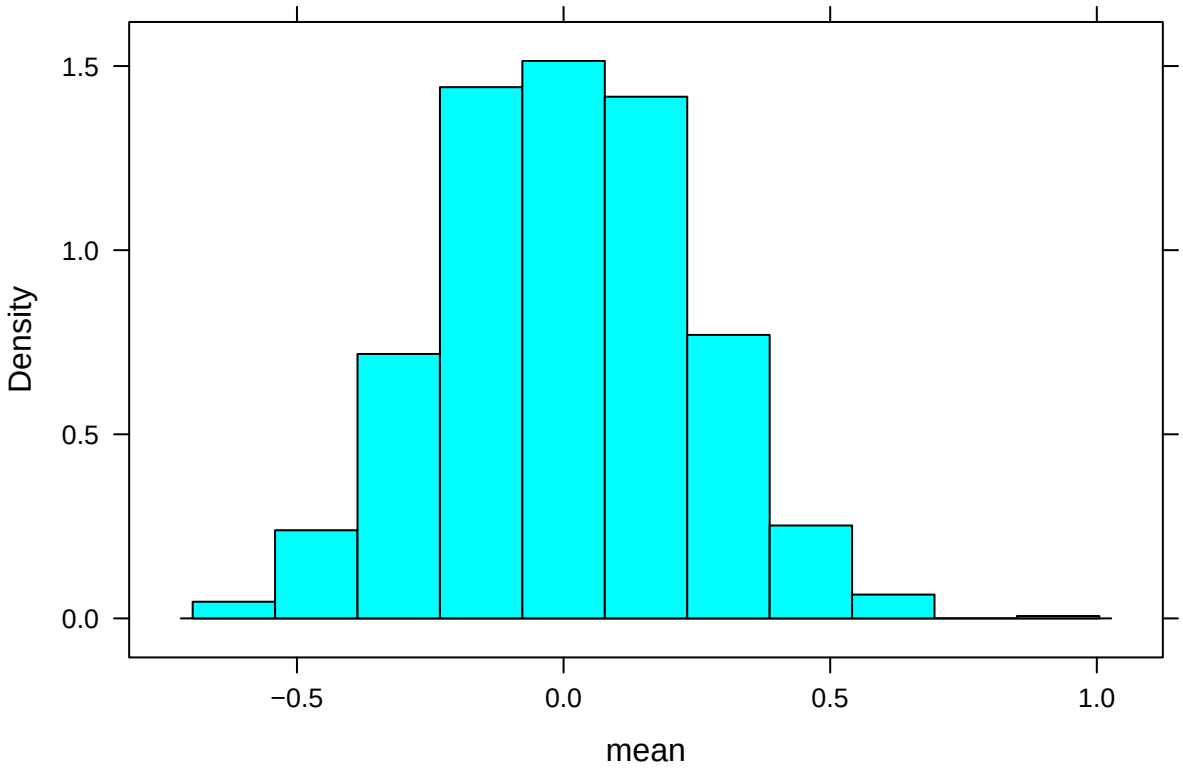
Unter der Annahme einer Normalverteilung kann (mit) der geschätzten Standardabweichung eine Stichprobe der gegebenen Länge unter $H_0 : \mu = 0$ erzeugt werden:

```
# Mittelwert
meandBSP <- mean( ~ BSP91JW, data = B3)
# Standardabweichung
sdBSP <- sd( ~ BSP91JW, data = B3)
# Anzahl Beobachtungen
n <-  length( B3$BSP91JW)
# Zufallszahlengenerator setzen
set.seed(1896)

simBSP <- rnorm(n=n, mean=0, sd=sdBSP)
histogram( ~ simBSP)
```



Für die Simulation der Verteilung unter H_0 wird dies jetzt z. B. 1000-mal wiederholt – und der Mittelwert berechnet:

```
SimBSP91JW <- do(1000) * mean( ~ rnorm(n=n, mean=0, sd=sdBSP))
histogram( ~ mean, SimBSP91JW)
```



Der Anteil der unter H_0 simulierten Daten, die einen größeren Mittelwert als den beobachteten aufweisen

ist sehr klein:

```
prop( mean( ~mean, data=SimBSP91JW)>= meandBSP)
```

```
## TRUE
##    0
```

Der beobachtete Wert der Teststatistik ist also unter $H_0 : \mu \leq 0$ sehr unwahrscheinlich, H_0 würde also verworfen.

Dieses Ergebnis liefert auch der t-Test:

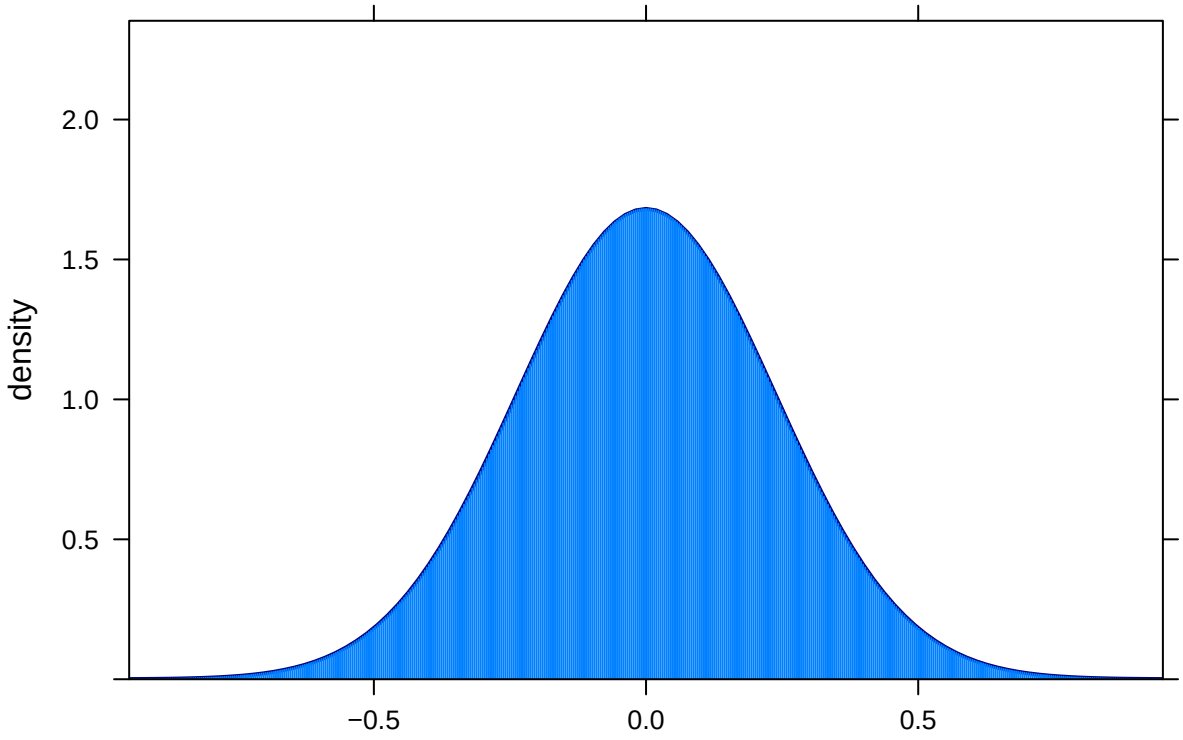
```
t.test( ~ BSP91JW, data = B3, alternative="greater")
```

```
##
##  One Sample t-test
##
## data:  B3$BSP91JW
## t = 14.739, df = 156, p-value < 2.2e-16
## alternative hypothesis: true mean is greater than 0
## 95 percent confidence interval:
##  3.105886      Inf
## sample estimates:
## mean of x
##  3.498662
```

Oder eine Berechnung "per Hand":

```
# Standardfehler
se <- sdBSP/sqrt(n)
# p-Wert
xpnorm( meandBSP, mean=0, sd=se, lower.tail = FALSE)
```

```
##
## If X ~ N(0, 0.2374), then
##
##  P(X <= 3.499) = P(Z <= 14.74) = 1
##  P(X >  3.499) = P(Z >  14.74) = 0
```




```
## [1] 1.807738e-49
```

t-Test für eine abhängige/gepaarte Stichprobe

Hier interessieren besonders die (Veränderung) der Investitionen in Ausrüstungsgüter (`IAUJW91`) und in Bauten (`IB91JW`). Die deskriptiven Kennzahlen zeigen,

```
favstats( ~ IAU91JW, data=B3)
```

```
##     min    Q1 median  Q3   max     mean       sd   n missing
##  -19.95 -1.25    5.3 9.1 27.25 3.992675 8.864805 157       0
```

```
favstats( ~ IB91JW, data=B3)
```

```
##     min    Q1 median   Q3   max     mean       sd   n missing
##  -21.59 -1.16    2.6 5.55 40.25 2.565096 7.481063 157       0
```

dass im betrachteten Zeitraum die Investitionen in Ausrüstungsgüter mit im arithmetischen Mittelwert von 3.99 im Mittel stärker gestiegen sind als die in Bauten mit 2.57. Da die Investitionen sicherlich in Zusammenhang mit der gesamten konjunkturellen Entwicklung stehen, ist davon auszugehen, dass es sich hier um vom jeweiligen Zeitpunkt abhängige Beobachtungen handelt. Daher wird hier die Differenz der Werte betrachtet: `IB91JW - IAU91JW`. Der R Befehl für einen t-Test lautet `t.test`:

```
t.test (~ (IB91JW - IAU91JW), data=B3)
```

```
##
##  One Sample t-test
##
```

```
## data:  B3$(IB91JW - IAU91JW)
## t = -1.9612, df = 156, p-value = 0.05164
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -2.86544149  0.01028226
## sample estimates:
## mean of x
##  -1.42758
```

Der (umfangreichen) Ausgabe können Sie neben dem z- bzw. t-Wert (-1.96) mit unter der Nullhypothese der Gleichheit des Lageparameters

$$H_0 : \mu_{\text{IB91JW}-\text{IAU91JW}} = 0$$

insbesondere den p-Wert (0.0516) und das Konfidenzintervall (-2.87, 0.01) entnehmen. Zum Signifikanznvieau von 5% wird die Nullhypothese also gerade so *nicht* abgelehnt, da der p-Wert über 5% liegt sowie der Wert der Nullhypothese, $\mu = 0$, im Konfidenzintervall.

Übung:

1. Testen Sie, ob es einen nicht zufälligen mittleren Lageunterschied zwischen der Veränderung des Preisindex des Bruttosozialproduktes `PBSPJW` und dem des privaten Verbrauchs `PCPJW` gibt.

---

t-Test für zwei unabhängige Stichprobe

Untersuchen wir, ob sich makroökonomische Kennzahlen im Auf- und Abschwung unterscheiden. Zunächst stellen wir fest, dass die eigentlich kategorielle Variable `PHASEN` hier numerisch kodiert wurde, was aber schnell verwirren würde.

```
typeof(B3$PHASEN)
```

```
## [1] "integer"
```

Typänderung zu `factor` geht einfach:

```
B3$PHASEN <- as.factor(B3$PHASEN)
```

Wenn wir die einzelnen `levels` des Faktors als numerische Werte verwenden wollen, würden wir den Befehl `as.numeric()` verwenden. Aber sicherheitshalber vorher über `levels()` schauen, ob die Reihenfolge auch stimmt.

Um die Interpretation zu erleichtern können wir hier einfach die Faktorstufe umbenennen.

```
levels(B3$PHASEN) <- c("Aufschwung", "Oberer Wendepunkt",
                       "Abschwung", "Unterer Wendepunkt")
```

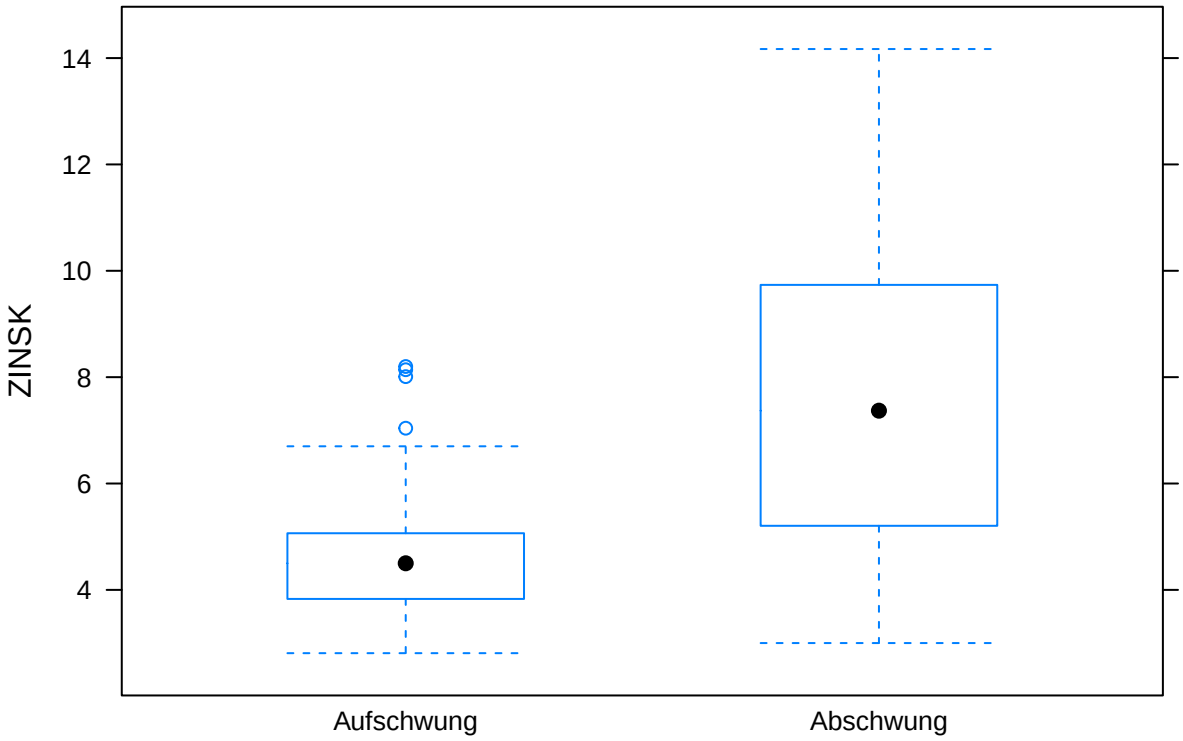
Jetzt ist keine Verwechselung von kategoriellen und metrischen Variablen mehr möglich.

Zunächst wird der Datensatz, der auch die konjunkturellen Wendepunkte enthält, nur auf Auf- und Abschwung eingeschränkt.

```
B3AufAb <- filter(B3, PHASEN %in% c("Aufschwung", "Abschwung"))
B3AufAb <- droplevels(B3AufAb)
```

In der politischen Diskussion werden immer niedrige Zinsen gefordert. Schauen wir mal, wie die Zinsen in den Konjunkturphasen in der Vergangenheit (1955-1994) verteilt waren:

```
bwplot(ZINSK ~ PHASEN, data=B3AufAb)
```



Anscheinend waren die Zinsen in Zeiten des Aufschwungs niedriger.

Was sagen die deskriptiven Kennzahlen:

```
favstats(ZINSK ~ PHASEN, data=B3AufAb)
```

```
##       PHASEN  min    Q1 median    Q3   max     mean       sd  n missing
## 1 Aufschwung 2.81 3.830   4.50 5.065  8.20 4.715085 1.209989 59       0
## 2  Abschwung 3.00 5.205   7.37 9.735 14.17 7.682553 3.020254 47       0
```

Alle Lagemaße für die Zinskosten sind in der Aufschwungphase niedriger.

Der t-Test der Zinskosten für

$$H_0 : \mu_{\text{Aufschwung}} = \mu_{\text{Abschwung}} \Leftrightarrow \mu_{\text{Aufschwung}} - \mu_{\text{Abschwung}} = 0$$

mit der Teststatistik

$$T = \frac{\bar{x}_A - \bar{x}_B}{\sqrt{\frac{sd_A^2}{n_A} + \frac{sd_B^2}{n_B}}}$$

hat dann den gleichen Aufbau des Syntax wie `bwplot` oder `favstats`: Teste die Zinskosten in Abhängigkeit der Konjunkturphase.

Die Berechnung der Teststatistik

```
t.test(ZINSK ~ PHASEN, data=B3AufAb)
```

```
##
##  Welch Two Sample t-test
##
## data:  ZINSK by PHASEN
## t = -6.3426, df = 57.766, p-value = 3.743e-08
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -3.904085 -2.030852
## sample estimates:
## mean in group Aufschwung  mean in group Abschwung
##                 4.715085                 7.682553
```

Der kleine p-Wert von 3.7430775×10^{-8} zeigt, dass die Nullhypothese der Gleichheit der Lageparameter verworfen werden kann. Wir können der Funktion auch eine spezielle Alternativhypothese übergeben:

```
t.test(ZINSK ~ PHASEN, data=B3AufAb, alternative = "less")
```

```
##
##  Welch Two Sample t-test
##
## data:  ZINSK by PHASEN
## t = -6.3426, df = 57.766, p-value = 1.872e-08
## alternative hypothesis: true difference in means is less than 0
## 95 percent confidence interval:
##       -Inf -2.185354
## sample estimates:
## mean in group Aufschwung  mean in group Abschwung
##                 4.715085                 7.682553
```

Jetzt haben wir die Nullhypothese "Das Lagemaß für die Zinskosten ist im Aufschwung *nicht* kleiner als im Abschwung" gegen die Alternativhypothese (Forschungshypothese) "Das Lagemaß für die Zinskosten ist im Aufschwung kleiner als im Abschwung" getestet:

$$H_0 : \mu_{\text{Aufschwung}} \geq \mu_{\text{Abschwung}} \quad vs. \quad H_A : \mu_{\text{Aufschwung}} < \mu_{\text{Abschwung}}$$

bzw.

$$H_0 : \mu_{\text{Aufschwung}} - \mu_{\text{Abschwung}} \geq 0 \quad vs. \quad H_A : \mu_{\text{Aufschwung}} - \mu_{\text{Abschwung}} < 0$$

Übung:

2. Untersuchen Sie, ob sich die mittlere Entwicklung des privaten Verbrauchs `CP91JW` zwischen den Konjunkturphasen unterscheidet.

Auch hier können wir, ohne eine Verteilungsannahme zu verwenden, permutieren.

```
mdiff <- diff(mean(ZINSK ~ PHASEN, data=B3AufAb))
mdiff
```

```
## Abschwung
##  2.967468
```

```
mdiff.null <- do(1000) * diff(mean(ZINSK ~ shuffle(PHASEN), data=B3AufAb))
```

Unter der Nullhypothese der Gleichheit der Lagemaße kommt eine gleich große oder größere Differenz also

```
prop(mdiff.null$Abschwung >= mdiff)
```

```
## TRUE
##    0
```

mal vor!

Da die statistische *Signifikanz* vom Standardfehler abhängt, welcher wiederum vom Stichprobenumfang abhängt, wurde von Cohen ein Maß für die *Effektstärke*, **Cohen's d** vorgeschlagen:

$$d = \frac{\bar{x}_A - \bar{x}_B}{sd_{\text{pool}}}$$

mit

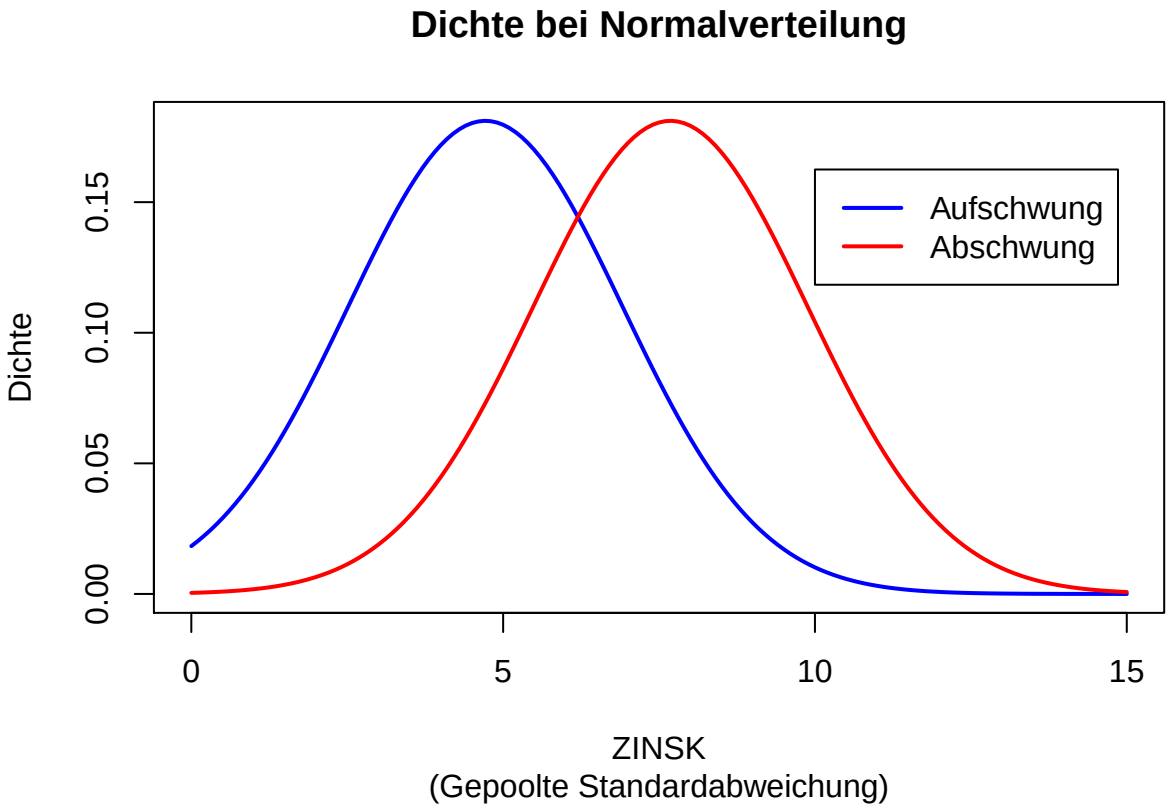
$$sd_{\text{pool}} = \sqrt{\frac{1}{n_A + n_B - 2}\Big((n_1 - 1)sd_A^2 + (n_2 - 1)sd_B^2\Big)}$$

```
# Kennzahlen 1. Stichprobe
m1 <- mean(B3$ZINSK[B3$PHASEN=="Aufschwung"])
sd1 <- sd(B3$ZINSK[B3$PHASEN=="Aufschwung"])
n1 <- length(B3$ZINSK[B3$PHASEN=="Aufschwung"])
# Kennzahlen 2. Stichprobe
m2 <- mean(B3$ZINSK[B3$PHASEN=="Abschwung"])
sd2 <- sd(B3$ZINSK[B3$PHASEN=="Abschwung"])
n2 <- length(B3$ZINSK[B3$PHASEN=="Abschwung"])
# Gepoolte Standardabweichung
sdpool <- sqrt( ((n1-1)*sd1^2 + (n2-1)*sd2^2) / (n1+n2-2))
```

```
# Cohen's d
d <- (m1-m2)/sdpool
d
```

```
## [1] -1.347291
```

Cohen's d ist ein Maß der Überlappung der Verteilungen:



Häufig werden Werte

- $|d| \leq 0.2$ als kleine
- $|d| \leq 0.5$ als mittlere
- $|d| \geq 0.8$ als große Effekte

bezeichnet.

Eine direkte Berechnung geht über das Paket `lsr`:

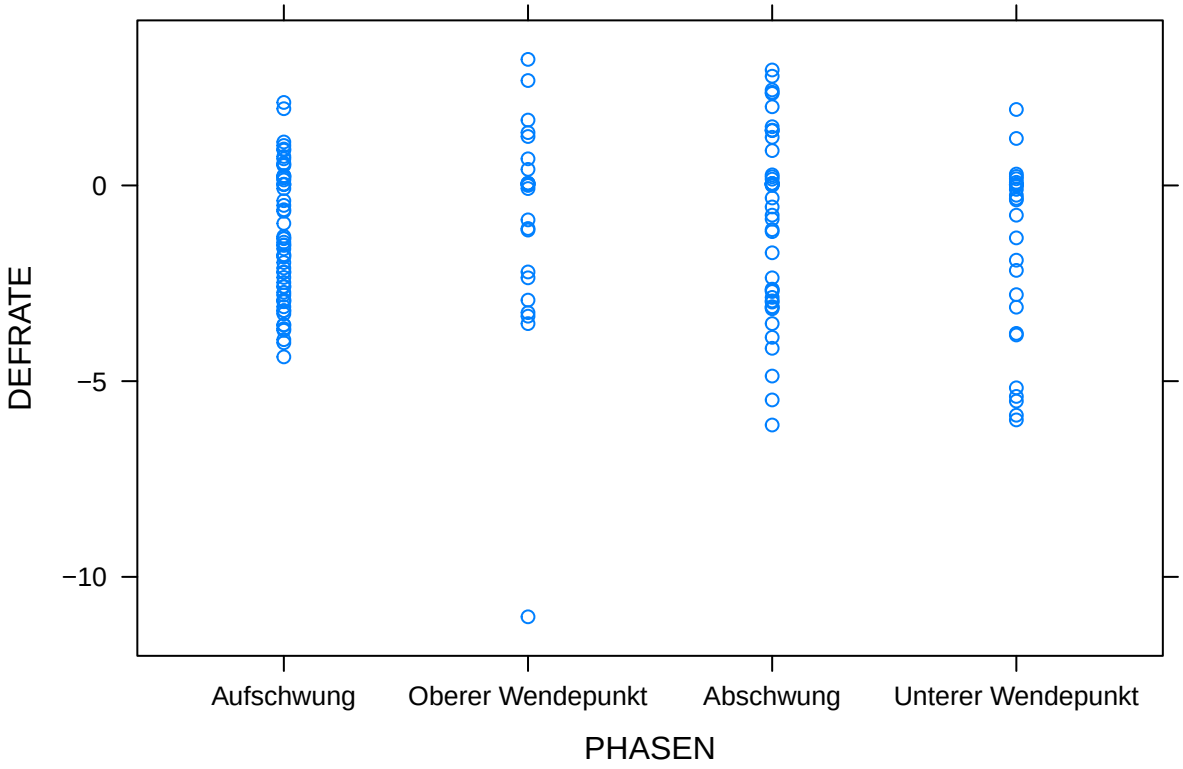
```
# Einmalig installieren:
# install.packages("lsr")
library(lsr)
cohensD(ZINSK ~ PHASEN, data=B3AufAb)
```

```
## [1] 1.347291
```

Varianzanalyse (ANOVA)

Bei mehr als zwei Gruppen funktionieren die Techniken des t-Tests nicht mehr. Um Lagemaßunterschiede zu testen, wird anstelle der Mittelwerte die Streuung verglichen: Ist die Streuung zwischen den Gruppen groß im Vergleich zur Streuung innerhalb der Gruppen?

```
xyplot(DEFRATE ~ PHASEN, data=B3)
```

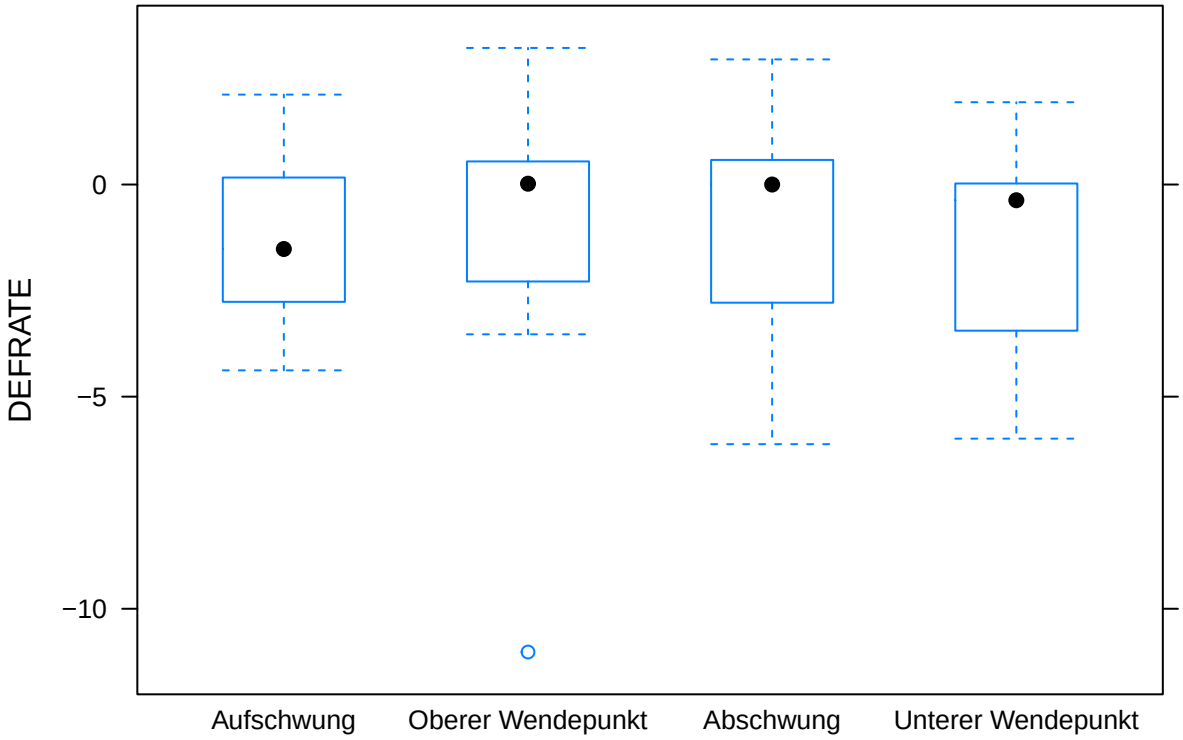


Es gilt, dass sich die Gesamtstreung (SST) zusammensetzt aus der Streuung zwischen den Gruppen (SSG) und innerhalb der Gruppen (SSE), d. h.,

$$SST = SSG + SSE.$$

Unterscheidet sich der mittlere Anteil des Staatsdefizits `DEFRATE` nicht zufällig zwischen den Konjunkturphasen?

```
bwplot(DEFRATE ~ PHASEN, data=B3)
```




```
favstats(DEFRATE ~ PHASEN, data=B3)
```

```
##                PHASEN    min      Q1 median     Q3  max       mean       sd
## 1         Aufschwung  -4.38 -2.7650  -1.52 0.1650 2.12 -1.3394915 1.680638
## 2  Oberer Wendepunkt -11.02 -2.2475   0.02 0.4775 3.22 -0.8479167 2.836558
## 3          Abschwung  -6.12 -2.7850   0.00 0.5800 2.95 -0.8380851 2.287536
## 4 Unterer Wendepunkt  -5.99 -3.4450  -0.37 0.0250 1.94 -1.6548148 2.364026
##    n missing
## 1 59       0
## 2 24       0
## 3 47       0
## 4 27       0
```

Vielleicht, vielleicht nicht.

Um eine Varianzanalyse (*Analysis of Variance, ANOVA*) mit

$$H_0 : \mu_1 = \mu_2 = \ldots = \mu_k$$

gegen

$$H_A : \text{Mindestens ein } \mu \text{ ist verschieden.}$$

durchzuführen kann in R u. a. der Befehl `aov` verwendet werden:

```
DEFaov <- aov(DEFRATE ~ PHASEN, data=B3)
summary(DEFaov)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
```

```
## PHASEN        3   15.7   5.236    1.09  0.355
## Residuals   153  734.9   4.803
```

Der p-Wert des F-Tests der Nullhypothese

$$H_0 : \mu_{\text{Aufschwung}} = \mu_{\text{Oberer Wendepunt}} = \mu_{\text{Abschwung}} = \mu_{\text{Unterer Wendepunkt}}$$

der Gleichheit der Lage ist mit 0.3552 größer als 0.05, die Nullhypothese kann also für das Staatsdefizit nicht verworfen werden.

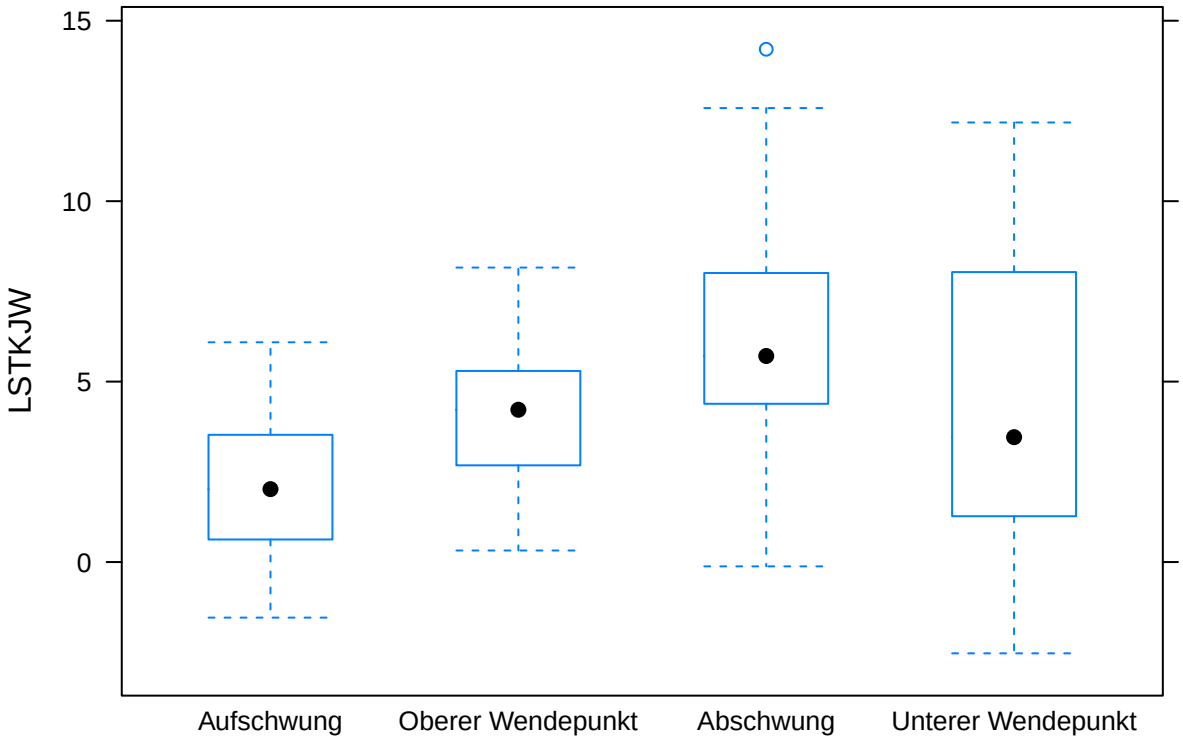
Bei k Gruppen ist die mittlere quadratische Varianz $MSG = \frac{1}{k-1}\sum_{i=1}^{k} n_i(\bar{x}_i - \bar{x})^2 = \underbrace{\frac{SSG}{k-1}}_{df_G}$ (n_i ist die Anzahl Beobachtungen in Gruppe i, $\bar{x}_i$ der Gruppenmittelwert und $\bar{x}$ der Gesamtmittelwert.) Hier also $MSG = 5.2361$. Der mittlere Quadratische Fehler, MSE, ist dann $MSE = \frac{1}{n-k}\sum_{i=1}^{k}(n_i - 1)sd_i^2 = \underbrace{\frac{SSE}{n-k}}_{df_E}$, mit sd_i der Standardabweichung in Gruppe i. Hier: $MSE = 4.8032$.

Der Wert der Teststatistik F ist dann

$$F = \frac{MSG}{MSE} = \frac{5.2361}{4.8032} = 1.0901.$$

Unterscheidet sich das Lagemaß der Veränderung der Lohnstückkosten `LSTKJW` nicht zufällig?

```
bwplot(LSTKJW ~ PHASEN, data=B3)
```




```
favstats(LSTKJW ~ PHASEN, data=B3)
```

```
##                PHASEN   min    Q1 median     Q3   max     mean       sd  n
## 1         Aufschwung -1.54 0.625   2.02 3.5250  6.09 2.111017 1.837423 59
```

```
## 2  Oberer Wendepunkt  0.32 2.840   4.22 5.2225  8.16 4.195833 2.074516 24
## 3           Abschwung -0.12 4.385   5.71 8.0100 14.21 6.291064 3.122604 47
## 4 Unterer Wendepunkt -2.53 1.270   3.46 8.0350 12.18 4.249630 4.449861 27
##   missing
## 1       0
## 2       0
## 3       0
## 4       0
```

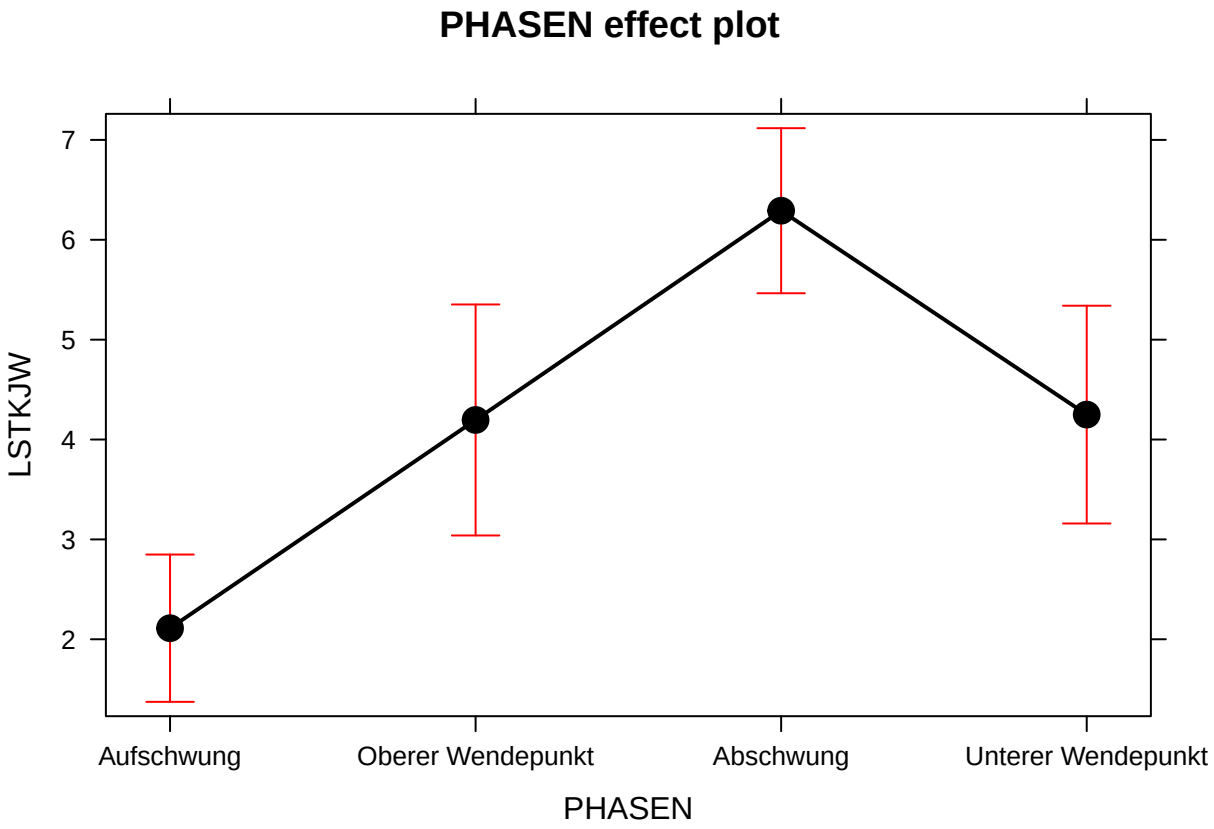
```
LSTKaov <- aov(LSTKJW ~ PHASEN, data=B3)
summary(LSTKaov)
```

```
##              Df Sum Sq Mean Sq F value   Pr(>F)
## PHASEN        3  459.5  153.15   18.62 2.37e-10 ***
## Residuals   153 1258.2    8.22
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Die Nullhypothese der Gleichheit wird hier also verworfen. Interessanterweise unterscheiden sich insbesondere die Lagemaße von Auf- und Abschwung, die beiden Wendepunkte liegen dazwischen.

Im Paket `effects` gibt es übrigens eine schöne Plotfunktion für die Effekte:

```
# Einmalig installieren:
# install.packages("effects")
library(effects)
plot(allEffects(LSTKaov))
```



Neben dem arithmetischen Mittelwert (Punktschätzer) wird in der Standardeinstellung das 95% Konfidenzintervall eingezeichnet.

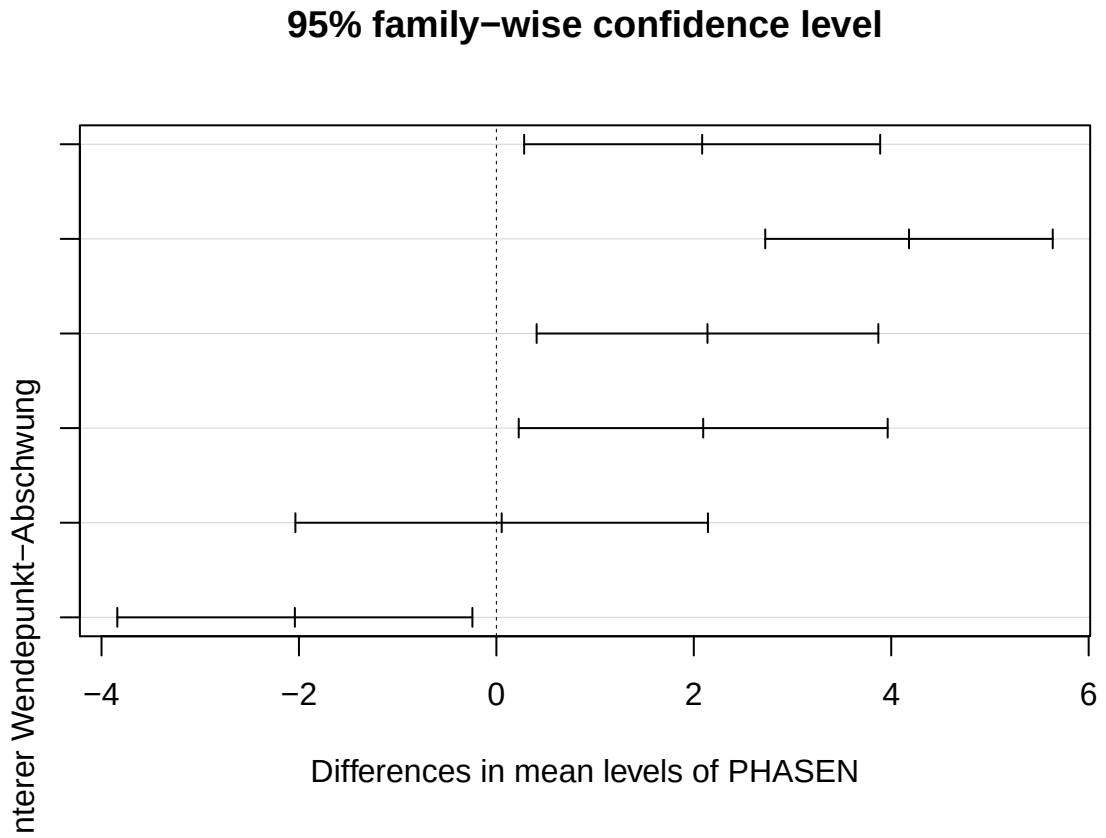
Um nach einer signifikanten Varianzanalyse sogenannte Post-Hoc Analysen durchzuführen (z. B. welche der paarweisen Vergleiche sind signifikant?) kann die Funktion `TukeyHSD()` (Tukey's "Honest Significant Difference") verwendet werden. Aufgrund des multiplen Testproblems (Kumulierung Fehler 1. Art) müssen die p-Werte angepasst werden.

```
LSTKposthoc <- TukeyHSD(LSTKaov)
LSTKposthoc
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = LSTKJW ~ PHASEN, data = B3)
##
## $PHASEN
##                                          diff        lwr       upr
## Oberer Wendepunkt-Aufschwung          2.0848164  0.2814507  3.888182
## Abschwung-Aufschwung                  4.1800469  2.7237350  5.636359
## Unterer Wendepunkt-Aufschwung         2.1386127  0.4079289  3.869296
## Abschwung-Oberer Wendepunkt           2.0952305  0.2264813  3.963980
## Unterer Wendepunkt-Oberer Wendepunkt  0.0537963 -2.0358558  2.143448
## Unterer Wendepunkt-Abschwung         -2.0414342 -3.8401454 -0.242723
```

```
##                                             p adj
## Oberer Wendepunkt-Aufschwung          0.0163388
## Abschwung-Aufschwung                  0.0000000
## Unterer Wendepunkt-Aufschwung         0.0087085
## Abschwung-Oberer Wendepunkt           0.0212622
## Unterer Wendepunkt-Oberer Wendepunkt 0.9998923
## Unterer Wendepunkt-Abschwung          0.0191825
```

```
plot(LSTKposthoc)
```



Hinweis: Da die einzelnen Faktorstufen hier unbalanciert sind (d .h., unterschiedliche Stichprobenumfänge haben) ist das Ergebnis hier nicht exakt.

Übung:

3. Gibt es nicht zufällige Lageunterschiede bei der Änderung der Erwerbstätigen `EWAJW` zwischen den Konjunkturphasen?

Erweiterung: Mehrfaktorielle Varianzanalyse mit Wechselwirkung

Betrachten wir noch einmal den *tips* Datensatz aus *Bryant, P. G. and Smith, M (1995) Practical Data Analysis: Case Studies in Business Statistics. Homewood, IL: Richard D. Irwin Publishing.*

Sofern noch nicht geschehen, können Sie in hier als `csv`-Datei herunterladen:

```
download.file("https://goo.gl/whKjnl", destfile = "tips.csv")
```

Das Einlesen der Daten in R erfolgt, sofern die Daten im Arbeitsverzeichnis liegen, über:

```
tips <- read.csv2("tips.csv")
```

Um zu schauen, inwieweit das Trinkgeld vom Geschlecht *und* dem Rauchverhalten abhängt, kann folgende Analyse durchgeführt werden:

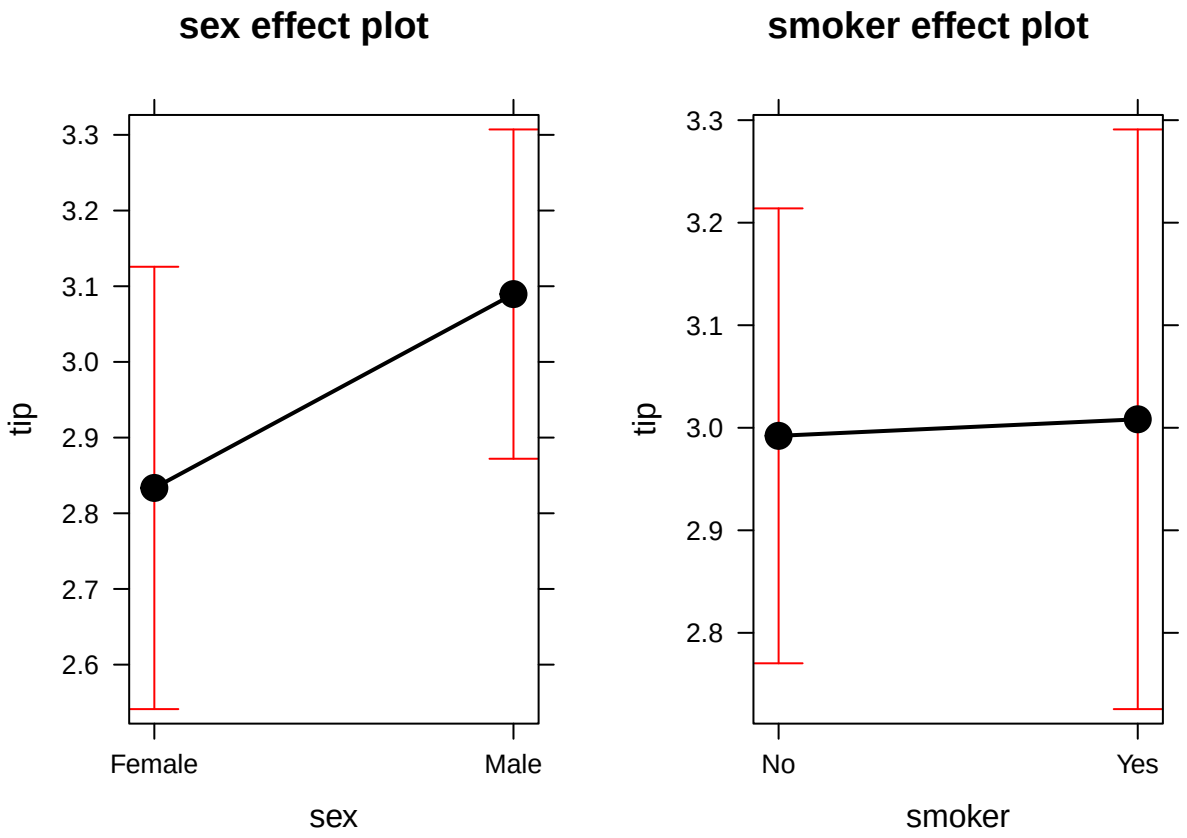
```
favstats(tip ~ sex + smoker, data=tips)
```

```
##   sex.smoker  min Q1 median     Q3  max     mean       sd  n missing
## 1  Female.No 1.00  2   2.68 3.4375  5.2 2.773519 1.128425 54       0
## 2    Male.No 1.25  2   2.74 3.7100  9.0 3.113402 1.489559 97       0
## 3 Female.Yes 1.00  2   2.88 3.5000  6.5 2.931515 1.219916 33       0
## 4   Male.Yes 1.00  2   3.00 3.8200 10.0 3.051167 1.500120 60       0
```

```
tipaov <- aov(tip ~ sex + smoker, data=tips)
summary(tipaov)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## sex           1    3.7   3.674   1.918  0.167
## smoker        1    0.0   0.015   0.008  0.930
## Residuals   241  461.5   1.915
```

```
plot(allEffects(tipaov))
```



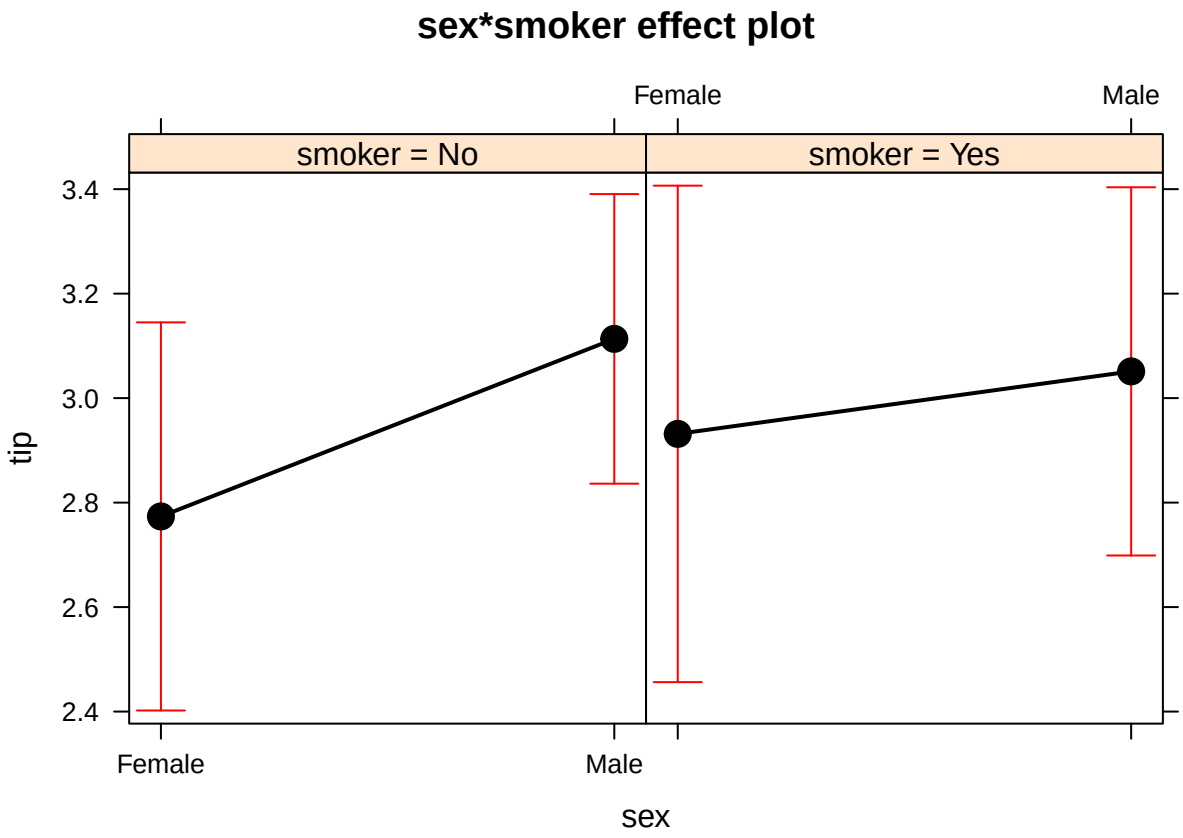
Beide Faktoren sind zum Signifikanzniveau 5% *nicht* signifikant, d. h., H_0, dass sich die Mittelwerte in der Population nicht unterscheiden, wird nicht verworfen.

Allerdings beobachten wir etwas anderes: Während der Mittelwert des Trinkgeldes bei den Frauen bei den Rauchern größer ist, ist es bei den Männern umgekehrt. Hier könnte also eine Wechselwirkung, eine Interaktion vorliegen. Diese wird in R über ein `:` in der Formel eingefügt:

```
tipaovww <- aov(tip ~ sex + smoker + sex:smoker, data=tips)
summary(tipaovww)
```

```
##                Df Sum Sq Mean Sq F value Pr(>F)
## sex             1    3.7   3.674   1.913  0.168
## smoker          1    0.0   0.015   0.008  0.930
## sex:smoker      1    0.6   0.640   0.333  0.564
## Residuals     240  460.9   1.920
```

```
plot(allEffects(tipaovww ))
```



Auch hier gilt: Mit einem p-Wert von 0.564 wird die Nullhypothese, dass in der Population keine Wechselwirkung von Geschlecht und Rauchverhalten für den Mittelwert vorliegt, nicht verworfen.

Übung: Teaching Rating

Dieser Datensatz analysiert u. a. den Zusammenhang zwischen Schönheit und Evaluierungsergebnis von Dozenten:

Hamermesh, D.S., and Parker, A. (2005). Beauty in the Classroom: Instructors' Pulchritude and Putative Pedagogical Productivity. Economics of Education Review, 24, 369–376.

Sie können ihn, sofern noch nicht geschehen, von https://goo.gl/6Y3KoK als `csv` herunterladen.

1. Ist das arithmetische Mittel der Evaluierung `eval` nicht zufällig größer als befriedigend (3)?
2. Gibt es einen nicht zufälligen Unterschied im Lagemaß der Evaluation `eval` zwischen männlichen und weiblichen Dozent/innen (`gender`)?

Literatur

- David M. Diez, Christopher D. Barr, Mine Çetinkaya-Rundel (2014): *Introductory Statistics with Randomization and Simulation*, https://www.openintro.org/stat/textbook.php?stat_book=isrs, Kapitel 4
- Nicholas J. Horton, Randall Pruim, Daniel T. Kaplan (2015): Project MOSAIC Little Books *A Student's Guide to R*, https://github.com/ProjectMOSAIC/LittleBooks/raw/master/StudentGuide/

MOSAIC-StudentGuide.pdf, Kapitel 7, 10.1

- Maike Luhmann (2015): *R für Einsteiger*, Kapitel 13, 14
- Andreas Quatember (2010): *Statistik ohne Angst vor Formeln*, Kapitel 3.5, 3.7, 3.12
- Daniel Wollschläger (2014): *Grundlagen der Datenanalyse mit R*, Kapitel 7.2, 7.3, 7.5

Lizenz

Versionshinweise:

- Datum erstellt: 2017-06-30
- R Version: 3.4.0
- `mosaic` Version: 0.14.4

Kapitel 5: Einführung Lineare Regression

Einfache Regression

Wir werden weiter den *tips* Datensatz aus *Bryant, P. G. and Smith, M (1995) Practical Data Analysis: Case Studies in Business Statistics. Homewood, IL: Richard D. Irwin Publishing* analysieren.

Sofern noch nicht geschehen, können Sie in hier als `csv`-Datei herunterladen:

```
download.file("https://goo.gl/whKjnl", destfile = "tips.csv")
```

Das Einlesen erfolgt, sofern die Daten im Arbeitsverzeichnis liegen, über:

```
tips <- read.csv2("tips.csv")
```

Zur Unterstützung der Analyse wird (wieder) `mosaic` verwendet; außerdem laden wir `ggplot2` für `qplot`:

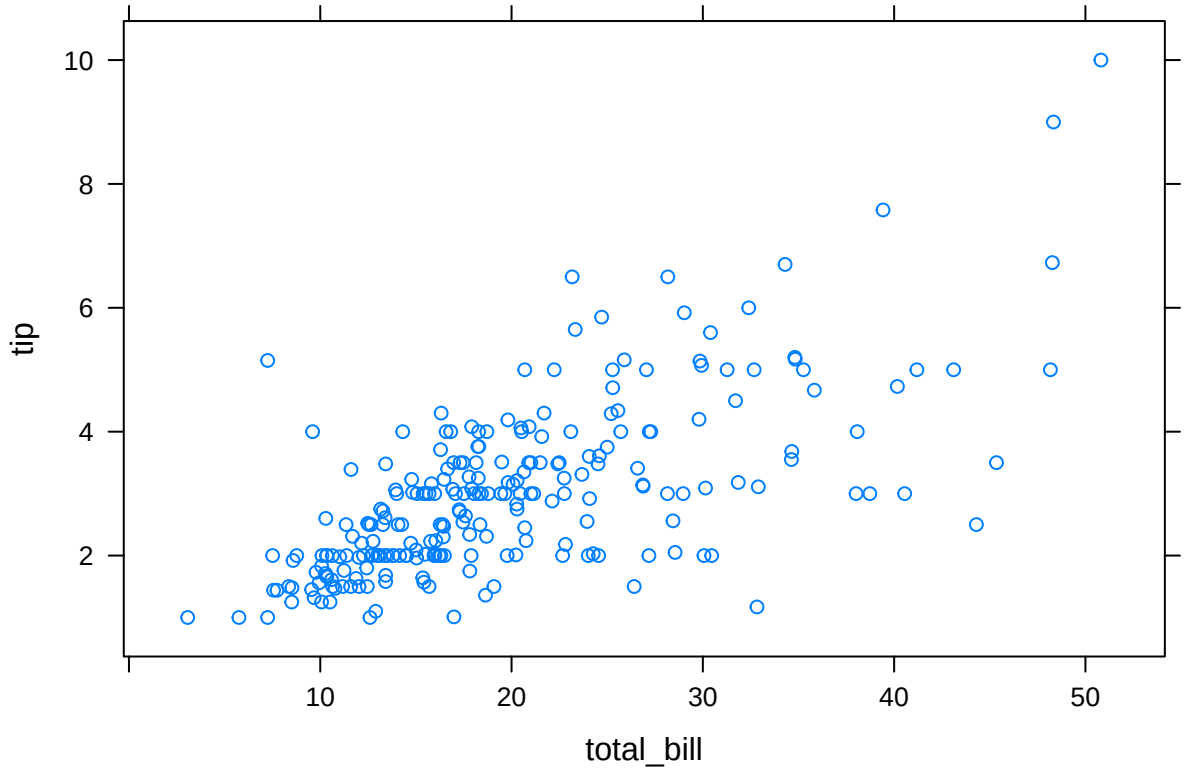
```
library(mosaic)
```

Wie hängen Trinkgeldhöhe `tip` und Rechnungshöhe `total_bill` zusammen? Kann die Höhe des Trinkgeldes als *lineare* Funktion der Rechnungshöhe linear modelliert werden?

$$tip_i = \beta_0 + \beta_1 \cdot total\_bill_i + \epsilon_i$$

Zunächst eine visuelle Analyse mi Hilfe eines Scatterplots.

```
xyplot(tip ~ total_bill, data=tips)
```



Es scheint einen positiven Zusammenhang zu geben. Modellieren wir die **abhängige** Variable `tip` (inhaltliche Entscheidung!) als lineare Funktion der **unabhängigen** Variable `total_bill`:

```
LinMod.1 <- lm(tip ~ total_bill, data=tips)
summary(LinMod.1)
```

```
##
## Call:
## lm(formula = tip ~ total_bill, data = tips)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1982 -0.5652 -0.0974  0.4863  3.7434
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.920270   0.159735   5.761 2.53e-08 ***
## total_bill  0.105025   0.007365  14.260  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.022 on 242 degrees of freedom
## Multiple R-squared:  0.4566, Adjusted R-squared:  0.4544
```

```
## F-statistic: 203.4 on 1 and 242 DF,  p-value: < 2.2e-16
```

Der Achsenabschnitt (`intercept`) wird mit 0.92 geschätzt, die Steigung in Richtung `total_bill` mit 0.11: steigt `total_bill` um einen Dollar, steigt im *Durchschnitt* `tip` um 0.11. Die (Punkt-)Prognose für `tip` lautet also

`tip` = 0.92 + 0.11 * `total_bill`

Die Koeffzienten werden dabei so geschätzt, dass $\sum \epsilon_i^2$ minimiert wird. Dies wird auch als *Kleinste Quadrate* (*Ordinary Least Squares*, *OLS*) Kriterium bezeichnet. Eine robuste Regression ist z. B. mit der Funktion `rlm()` aus dem Paket `MASS` möglich.

In mosaic kann ein solches Modell einfach als neue Funktion definiert werden:

```
LinMod.1Fun <- makeFun(LinMod.1)
```

Die (Punkt-)Prognose für die Trinkgeldhöhe, bspw. für eine Rechnung von 30$ kann dann berechnet werden

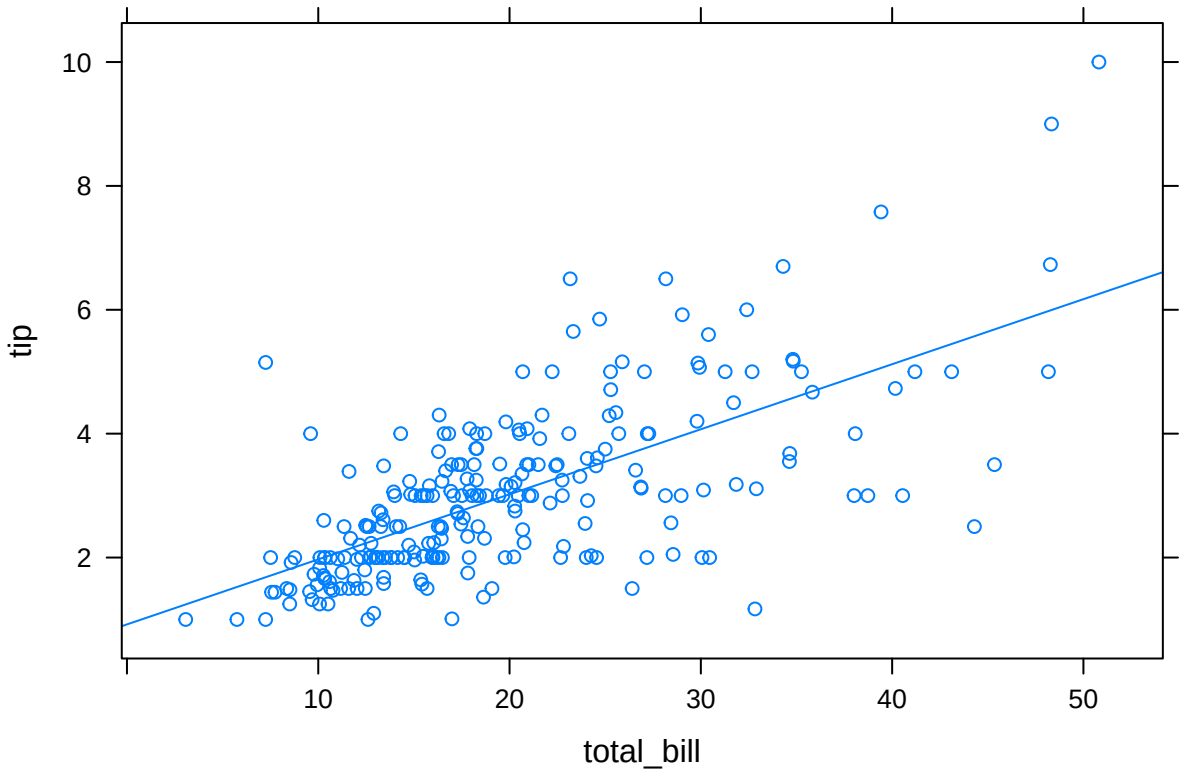
```
LinMod.1Fun(total_bill=30)
```

```
##        1
## 4.071005
```

also 4.07$.

In mosaic kann die Modellgerade über

```
plotModel(LinMod.1)
```



betrachtet werden. Das **Bestimmtheitsmaß**, d. h. der Anteil der im Modell erklärten Varianz,

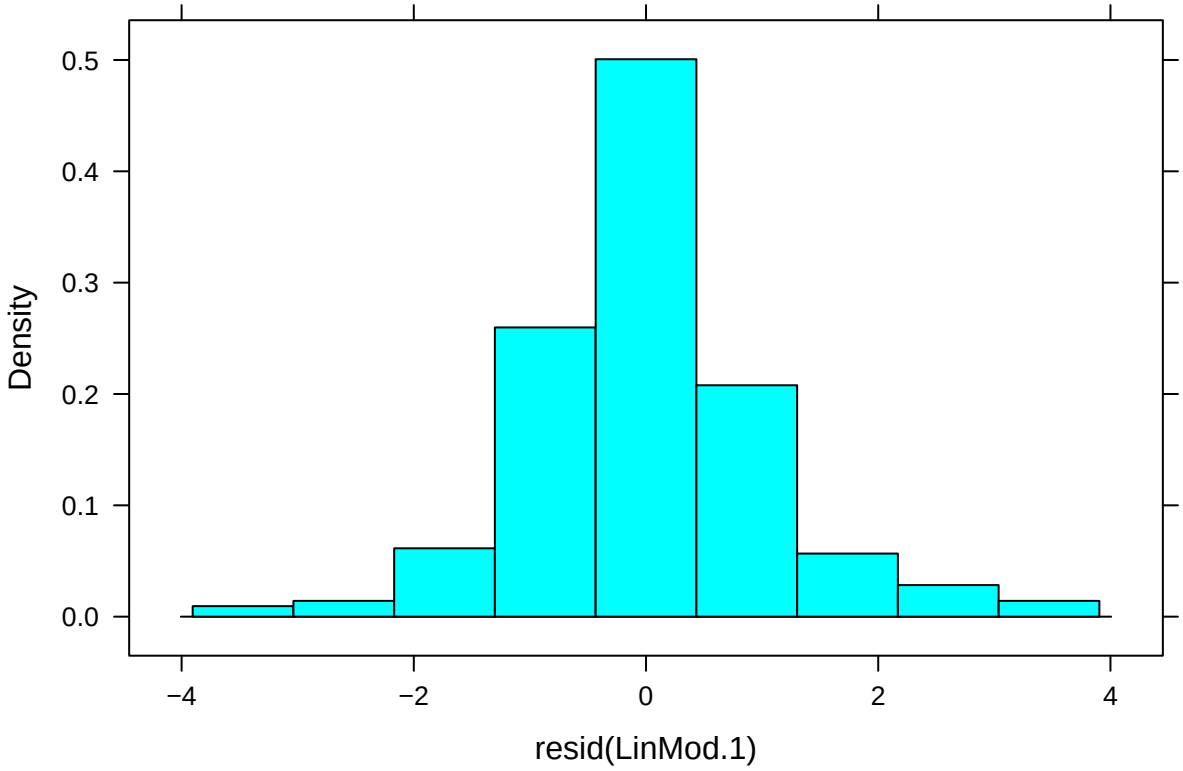
$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

ist mit 0.46 "ok": 46-% der Variation des Trinkgeldes wird im Modell erklärt.

Aber wie sieht es mit den Annahmen aus?

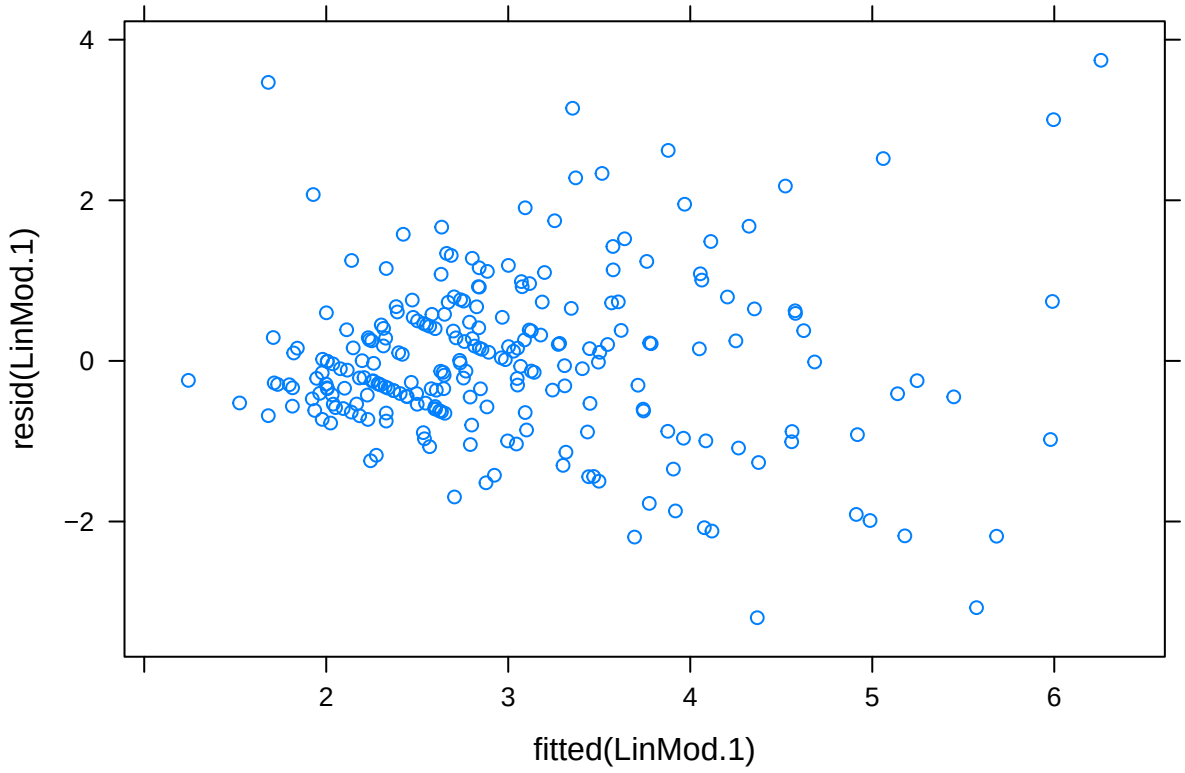
- Die Linearität des Zusammenhangs haben wir zu Beginn mit Hilfe des Scatterplots "überprüft".
- Zur Überprüfung der Normalverteilung der Residuen zeichnen wir ein Histogramm. Die Residuen können über den Befehl `resid()` aus einem Linearen Modell extrahiert werden. Hier scheint es zu passen:

```
histogram( ~ resid(LinMod.1))
```



- Konstante Varianz: Dies kann z. B. mit einem Scatterplot der Residuen auf der y-Achse und den angepassten Werten auf der x-Achse überprüft werden. Die angepassten Werte werden über den Befehl `fitted()` extrahiert. Diese Annahme scheint verletzt zu sein (siehe unten): je größer die Prognose des Trinkgeldes, desto größer wirkt die Streuung der Residuen. Dieses Phänomen ließ sich schon aus dem ursprünglichen Scatterplot `xyplot(tip ~ total_bill, data=tips)` erahnen. Das ist auch inhaltlich plausibel: je höher die Rechnung, desto höher die Varianz beim Trinkgeld. Die Verletzung dieser Annahme beeinflusst *nicht* die Schätzung der Steigung, sondern die Schätzung des Standardfehlers, also des p-Wertes des Hypothesentests, d. h., $H_0 : \beta_1 = 0$.

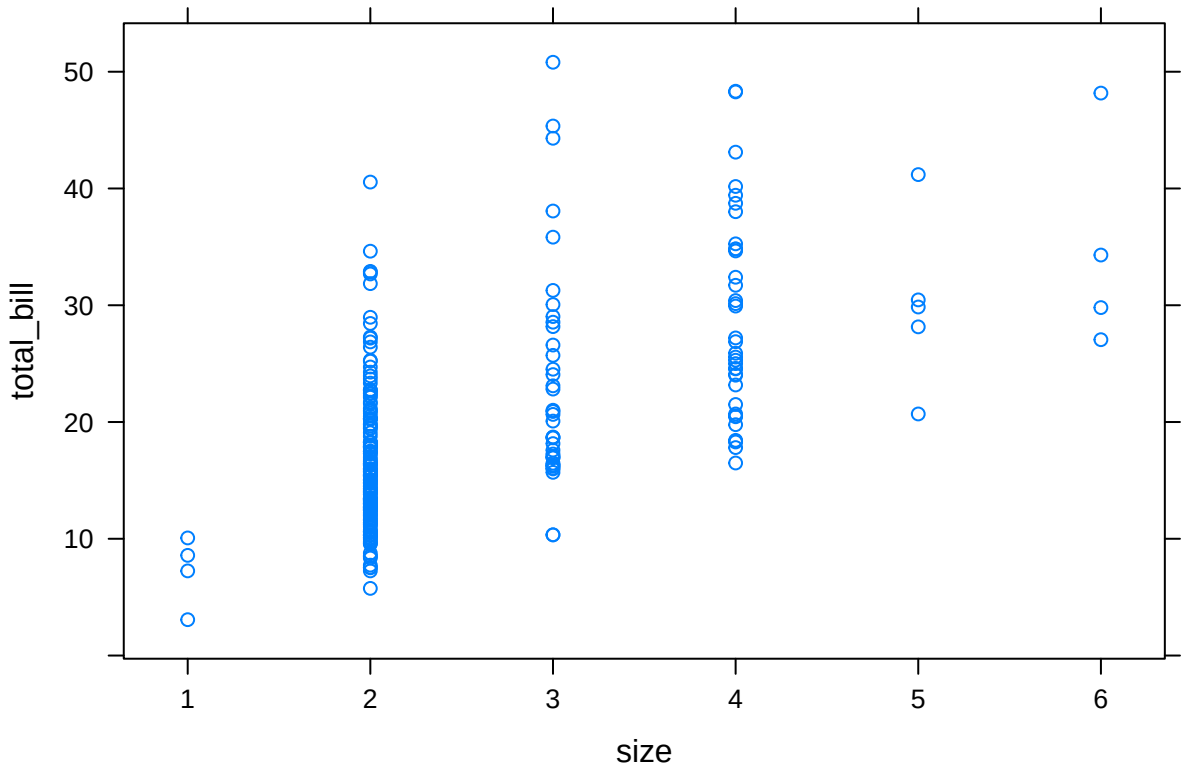
```
xyplot(resid(LinMod.1) ~ fitted(LinMod.1))
```



- Extreme Ausreißer: Wie am Plot der Linearen Regression `plotModel(LinMod.1)` erkennbar, gibt es vereinzelt Ausreißer nach oben, allerdings ohne einen extremen Hebel.

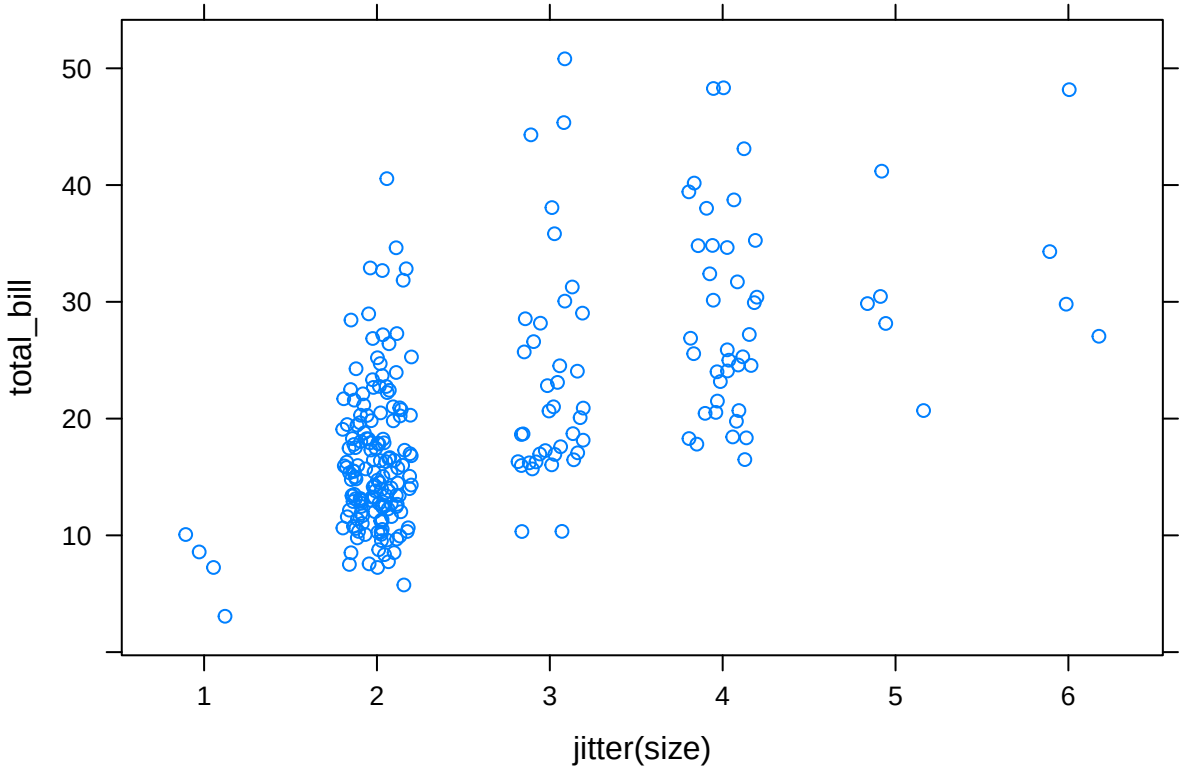
Hängt die Rechnungshöhe von der Anzahl der Personen ab? Bestimmt, aber wie?

```
xyplot(total_bill ~ size, data=tips)
```



Da bei diskreten metrischen Variablen (hier `size`) Punkte übereinander liegen können, sollte man "jittern", d. h., eine (kleine) Zufallszahl addieren:

```
set.seed(1896) # Zufallszahlengenerater setzen
xyplot(total_bill ~ jitter(size), data=tips)
```



Übung:

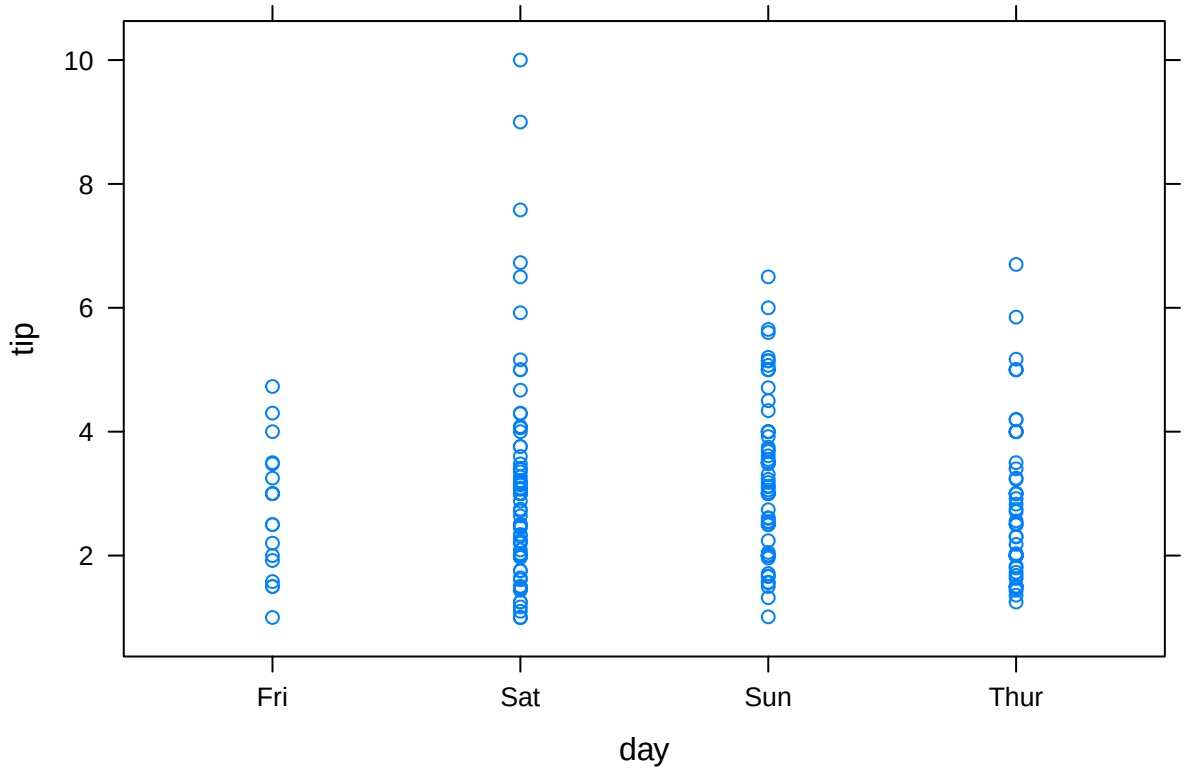
1. Um wie viel Dollar steigt im Durchschnitt das Trinkgeld, wenn eine Person mehr am Tisch sitzt?
2. Für wie aussagekräftig halten Sie Ihr Ergebnis aus 1.?

Regression mit kategorialen Werten

Der Wochentag `day` ist eine kategoriale Variable. Wie sieht eine Regression des Trinkgeldes darauf aus?

Zunächst grafisch:

```
xyplot(tip ~ day, data=tips)
```



Und als Lineares Modell:

```
LinMod.2 <- lm(tip ~ day, data=tips)
summary(LinMod.2)
```

```
##
## Call:
## lm(formula = tip ~ day, data = tips)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.2451 -0.9931 -0.2347  0.5382  7.0069
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.73474    0.31612   8.651 7.46e-16 ***
## daySat       0.25837    0.34893   0.740    0.460
## daySun       0.52039    0.35343   1.472    0.142
## dayThur      0.03671    0.36132   0.102    0.919
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.378 on 240 degrees of freedom
```

```
## Multiple R-squared:  0.02048,    Adjusted R-squared:  0.008232
## F-statistic: 1.672 on 3 and 240 DF,  p-value: 0.1736
```

Die im Modell angegebenen Schätzwerte sind die Änderung der Trinkgeldprognose, wenn z. B. der Tag ein Samstag (`daySat`) im Vergleich zu einer Referenzkategorie. Dies ist in R das erste Element des Vektors der Faktorlevel. Welcher dies ist ist über den Befehl `levels()` zu erfahren

```
levels(tips$day)
```

```
## [1] "Fri"  "Sat"  "Sun"  "Thur"
```

hier also Fri (aufgrund der standardmäßig aufsteigenden alphanumerischen Sortierung). Dies kann über `relevel()` geändert werden. Soll z. B. die Referenz der Donnerstag, `Thur` sein:

```
tips$day <- relevel(tips$day, ref = "Thur")
levels(tips$day)
```

```
## [1] "Thur" "Fri"  "Sat"  "Sun"
```

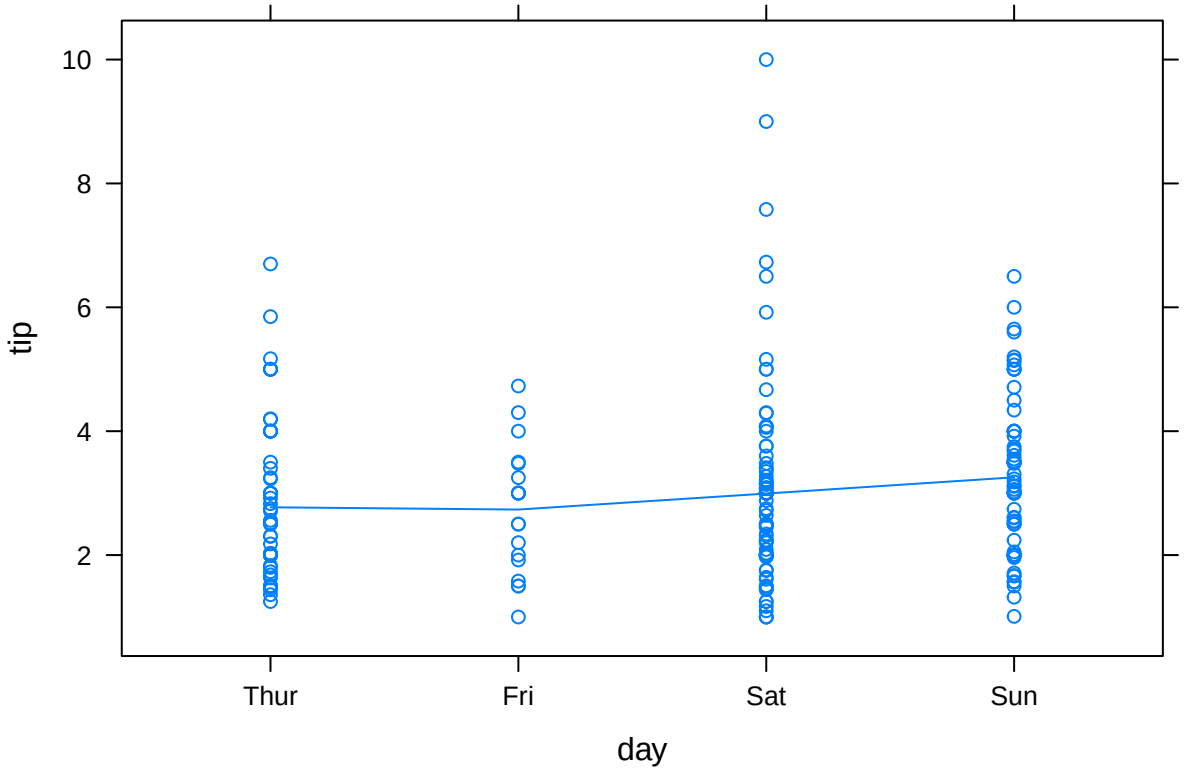
Das Modell ändert sich entsprechend:

```
LinMod.3 <- lm(tip ~ day, data=tips)
summary(LinMod.3)
```

```
##
## Call:
## lm(formula = tip ~ day, data = tips)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.2451 -0.9931 -0.2347  0.5382  7.0069
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.77145    0.17500  15.837   <2e-16 ***
## dayFri      -0.03671    0.36132  -0.102   0.9191
## daySat       0.22165    0.22902   0.968   0.3341
## daySun       0.48368    0.23581   2.051   0.0413 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.378 on 240 degrees of freedom
## Multiple R-squared:  0.02048,    Adjusted R-squared:  0.008232
## F-statistic: 1.672 on 3 and 240 DF,  p-value: 0.1736
```

sowie als Plot:

```
plotModel(LinMod.3)
```



Eine Alternative zu `relevel()` zur Bestimmung der Referenzkategorie ist es, innerhalb von `factor()` die Option `levels=` direkt in der gewünschten Sortierung zu setzen.

```
day <- factor(tips$day, levels=c("Thur", "Fri", "Sat",  "Sun"))
```

Die (Punkt-)Prognose für die Trinkgeldhöhe, bspw. an einen Freitag kann dann berechnet werden

```
LinMod.3Fun <- makeFun(LinMod.3)
LinMod.3Fun(day="Fri")
```

```
##        1
## 2.734737
```

Übung:

3. Wie verändert sich die Rechnungshöhe im Durchschnitt, wenn die Essenszeit Dinner statt Lunch ist?
4. Wie viel % der Variation der Rechnungshöhe können Sie durch die Essenszeit modellieren?

Multivariate Regression

Aber wie wirken sich die Einflussgrößen *zusammen* auf das Trinkgeld aus?

```
LinMod.4 <- lm(tip ~ total_bill + size + sex  + smoker + day + time, data=tips)
summary(LinMod.4)
```

```
##
## Call:
## lm(formula = tip ~ total_bill + size + sex + smoker + day + time,
##     data = tips)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8475 -0.5729 -0.1026  0.4756  4.1076
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.641558   0.497586   1.289   0.1985
## total_bill   0.094487   0.009601   9.841   <2e-16 ***
## size         0.175992   0.089528   1.966   0.0505 .
## sexMale     -0.032441   0.141612  -0.229   0.8190
## smokerYes   -0.086408   0.146587  -0.589   0.5561
## dayFri       0.162259   0.393405   0.412   0.6804
## daySat       0.040801   0.470604   0.087   0.9310
## daySun       0.136779   0.471696   0.290   0.7721
## timeLunch    0.068129   0.444617   0.153   0.8783
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.024 on 235 degrees of freedom
## Multiple R-squared:  0.4701, Adjusted R-squared:  0.452
## F-statistic: 26.06 on 8 and 235 DF,  p-value: < 2.2e-16
```

Interessant sind die negativen Vorzeichen vor den Schätzwerten für `sexMale` und `smokerYes` – anscheinend geben Männer und Raucher weniger Trinkgeld, wenn alle anderen Faktoren konstant bleiben. Bei einer rein univariaten Betrachtung wäre etwas anderes herausgekommen.

```
summary(lm(tip ~ sex, data=tips))
```

```
##
## Call:
## lm(formula = tip ~ sex, data = tips)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.0896 -1.0896 -0.0896  0.6666  6.9104
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.8334     0.1481  19.137   <2e-16 ***
## sexMale       0.2562     0.1846   1.388    0.166
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.381 on 242 degrees of freedom
## Multiple R-squared:  0.007896,   Adjusted R-squared:  0.003797
## F-statistic: 1.926 on 1 and 242 DF,  p-value: 0.1665
```

```
summary(lm(tip ~ smoker, data=tips))
```

```
##
## Call:
## lm(formula = tip ~ smoker, data = tips)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.0087 -0.9936 -0.1003  0.5580  6.9913
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.99185    0.11283  26.517   <2e-16 ***
## smokerYes    0.01686    0.18276   0.092    0.927
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.386 on 242 degrees of freedom
## Multiple R-squared:  3.515e-05,  Adjusted R-squared:  -0.004097
## F-statistic: 0.008506 on 1 and 242 DF,  p-value: 0.9266
```

Diese *Umkehrung* des modellierten Effektes liegt daran, dass es auch einen positiven Zusammenhang zur Rechnungshöhe gibt:

```
summary(lm(total_bill ~ sex, data=tips))
```

```
##
## Call:
## lm(formula = total_bill ~ sex, data = tips)
##
## Residuals:
```

```
##    Min     1Q Median     3Q    Max
## -14.99  -6.02  -1.94   3.99  30.07
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  18.0569     0.9463  19.081   <2e-16 ***
## sexMale       2.6872     1.1797   2.278   0.0236 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.827 on 242 degrees of freedom
## Multiple R-squared:  0.02099,    Adjusted R-squared:  0.01694
## F-statistic: 5.188 on 1 and 242 DF,  p-value: 0.02361
```

```
summary(lm(total_bill ~ smoker, data=tips))
```

```
##
## Call:
## lm(formula = total_bill ~ smoker, data = tips)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.686  -6.459  -1.888   4.583  30.054
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  19.1883     0.7233  26.529   <2e-16 ***
## smokerYes     1.5681     1.1716   1.338    0.182
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.888 on 242 degrees of freedom
## Multiple R-squared:  0.007348,   Adjusted R-squared:  0.003246
## F-statistic: 1.791 on 1 and 242 DF,  p-value: 0.182
```

Im vollem Modell `LinMod.4` sind alle unabhängigen Variablen berücksichtigt, die Koeffizienten beziehen sich dann immer auf: gegeben, die anderen Variablen bleiben konstant, d. h. ceteris paribus.

Vergleichen wir mal zwei Modelle:

```
LinMod.5a <- lm(tip ~  sex, data=tips)
coef(LinMod.5a) # Koeffizienten extrahieren
```

```
## (Intercept)     sexMale
```

```
##   2.8334483   0.2561696
```

```
LinMod.5b <- lm(tip ~  sex + total_bill, data=tips)
coef(LinMod.5b) # Koeffizienten extrahieren
```

```
## (Intercept)     sexMale  total_bill
##  0.93327849 -0.02660871  0.10523236
```

Ohne die Berücksichtigung der **Kovariable/Störvariable** Rechnungshöhe geben `Male` ein um im Durchschnitt 0.26$ *höheres* Trinkgeld, bei Kontrolle, d. h. gleicher Rechnungshöhe ein um 0.03$ *niedrigeres* Trinkgeld als die Referenzklasse `Female` (`levels(tips$sex)[1]`).

Inferenz in der linearen Regression

Kehren wir noch einmal zur multivariaten Regression (`LinMod.4`) zurück.

```
summary(LinMod.4)
```

```
##
## Call:
## lm(formula = tip ~ total_bill + size + sex + smoker + day + time,
##     data = tips)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8475 -0.5729 -0.1026  0.4756  4.1076
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.641558   0.497586   1.289   0.1985
## total_bill   0.094487   0.009601   9.841   <2e-16 ***
## size         0.175992   0.089528   1.966   0.0505 .
## sexMale     -0.032441   0.141612  -0.229   0.8190
## smokerYes   -0.086408   0.146587  -0.589   0.5561
## dayFri       0.162259   0.393405   0.412   0.6804
## daySat       0.040801   0.470604   0.087   0.9310
## daySun       0.136779   0.471696   0.290   0.7721
## timeLunch    0.068129   0.444617   0.153   0.8783
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.024 on 235 degrees of freedom
## Multiple R-squared:  0.4701, Adjusted R-squared:  0.452
## F-statistic: 26.06 on 8 and 235 DF,  p-value: < 2.2e-16
```

In der 4. Spalte der, mit Zeilennamen versehenen Tabelle `Coefficients` stehen die p-Werte der Nullhypothese, die unabhängige Variable hat, gegeben alle anderen Variablen im Modell, keinen linearen Einfluss auf die abhängige Variable: $H_0 : \beta_i = 0$. Zur Bestimmung des p-Wertes wird der Schätzer (`Estimate`) durch den Standardfehler (`Std. Error`) dividiert. Der resultierende t-Wert (`t value`) wird dann, zusammen mit der Anzahl an Freiheitsgraden zur Berechnung des p-Wertes (`Pr(>|t|)`) verwendet. Ein einfacher t-Test!

Zur schnelleren Übersicht finden sich dahinter "Sternchen" und "Punkte", die die entsprechenden Signifikanzniveaus symbolisieren: $***$ bedeutet eine Irrtumswahrscheinlichkeit, Wahrscheinlichkeit für Fehler 1. Art, von unter 0.001, d. h. unter 0,1%. $**$ entsprechend 1%, $*$ 5% und . 10%.

Zum Signifikanzniveau von 10% sind hier also zwei Faktoren und der Achsenabschnitt (`(Intercept)`) signifikant – nicht notwendigerweise relevant: Rechnungshöhe `total_bill` sowie Anzahl Personen `size`. Beides wirkt sich linear positiv auf die Trinkgeldhöhe aus: Mit jedem Dollar Rechnungshöhe steigt im Mittelwert die Trinkgeldhöhe um 0.09 Dollar, mit jeder Person um 0.18 Dollar – gegeben alle anderen Faktoren bleiben konstant. Das Bestimmtheitsmaß R^2 (`Multiple R-squared:`) liegt bei 0.47, also 47% der Variation des Trinkgeldes wird im Modell erklärt.

Außerdem wird getestet, ob alle Koeffizienten der unabhängigen Variablen gleich Null sind:

$$H_0 : \beta_1 = \beta_2 = \cdots = \beta_k = 0$$

Das Ergebnis des zugrundeliegenden F-Tests (vgl. Varianzanalyse) wird in der letzten Zeile angegeben (`F-Statistic`). Hier wird H_0 also verworfen.

Erweiterungen

Modellwahl

Das Modell mit allen Variablen des Datensatzes, d. h., mit 6 unabhängigen (`LinMod.4`) erklärt 47.01% der Variation, das Modell *nur* mit der Rechnungshöhe als erklärende Variable (`LinMod.1`) schon 45.66%, der Erklärungszuwachs liegt also gerade einmal bei 1.35 Prozentpunkten. In der Statistik ist die Wahl des *richtigen* Modells eine der größten Herausforderungen, auch deshalb, weil das wahre Modell in der Regel nicht bekannt ist und es schwer ist, die richtige Balance zwischen Einfachheit und Komplexität zu finden. Aufgrund des Zufalls kann es immer passieren, dass das Modell sich zu sehr an die *zufälligen* Daten anpasst (Stichwort: Overfitting). Es gibt unzählige Modellwahlmethoden, und leider garantiert keine, dass immer das beste Modell gefunden wird. Eine Möglichkeit ist die sogenannte Schrittweise-Rückwärtsselektion auf Basis des Akaike-Informationskriteriums (AIC)[2]. Diese ist nicht nur recht weit verbreitet - und liefert unter bestimmten Annahmen das "richtige" Modell - sondern in R durch den Befehl `step()` einfach umsetzbar:

[2]siehe z. B. Rob J Hyndman & George Athanasopoulos, Forecasting: principles and practice, Kapitel 5.3: Selecting predictors, https://www.otexts.org/fpp/5/3

```r
step(LinMod.4)
```

```
## Start:  AIC=20.51
## tip ~ total_bill + size + sex + smoker + day + time
##
##              Df Sum of Sq    RSS     AIC
## - day         3     0.609 247.14  15.116
## - time        1     0.025 246.55  18.538
## - sex         1     0.055 246.58  18.568
## - smoker      1     0.365 246.89  18.874
## <none>                    246.53  20.513
## - size        1     4.054 250.58  22.493
## - total_bill  1   101.595 348.12 102.713
##
## Step:  AIC=15.12
## tip ~ total_bill + size + sex + smoker + time
##
##              Df Sum of Sq    RSS    AIC
## - time        1     0.001 247.14 13.117
## - sex         1     0.042 247.18 13.157
## - smoker      1     0.380 247.52 13.490
## <none>                    247.14 15.116
## - size        1     4.341 251.48 17.365
## - total_bill  1   101.726 348.86 97.232
##
## Step:  AIC=13.12
## tip ~ total_bill + size + sex + smoker
##
##              Df Sum of Sq    RSS    AIC
## - sex         1     0.041 247.18 11.157
## - smoker      1     0.379 247.52 11.491
## <none>                    247.14 13.117
## - size        1     4.342 251.48 15.366
## - total_bill  1   103.327 350.46 96.350
##
## Step:  AIC=11.16
## tip ~ total_bill + size + smoker
##
##              Df Sum of Sq    RSS    AIC
## - smoker      1     0.376 247.55  9.528
## <none>                    247.18 11.157
```

```
## - size        1     4.344 251.52 13.408
## - total_bill  1   104.263 351.44 95.029
##
## Step:  AIC=9.53
## tip ~ total_bill + size
##
##              Df Sum of Sq    RSS    AIC
## <none>                     247.55  9.528
## - size        1     5.235 252.79 12.634
## - total_bill  1   106.281 353.83 94.685
```

```
##
## Call:
## lm(formula = tip ~ total_bill + size, data = tips)
##
## Coefficients:
## (Intercept)   total_bill         size
##     0.66894      0.09271      0.19260
```

In den letzten Zeilen der Ausgabe steht das beste Modell, das diese Methode (schrittweise, rückwärts) mit diesem Kriterium (AIC) bei diesen Daten findet (Punktprognose, d. h. ohne Residuum):

```
tip =  0.66894 +  0.09271 * total_bill + 0.19260 * size
```

Der Ausgabe können Sie auch entnehmen, welche Variablen in welcher Reihenfolge *entfernt* wurden: Zunächst `day`, dann `time`, danach `sex` und schließlich `smoker`. Hier sind also dieselben Variablen noch im Modell, die auch in `LinMod.4` signifikant zum Niveau 10% waren, eine Auswahl der dort signifikanten Variablen hätte also dasselbe Modell ergeben. Das ist häufig so, aber nicht immer!

Interaktionen

Wir haben gesehen, dass es einen Zusammenhang zwischen der Trinkgeldhöhe und der Rechnungshöhe gibt. Vielleicht unterscheidet sich der Zusammenhang je nachdem, ob geraucht wurde, d. h., vielleicht gibt es eine Interaktion (Wechselwirkung). Die kann in `lm` einfach durch ein `*` zwischen den unabhängigen Variablen modelliert werden (`a*b` entspricht in R Formeln `a+b+a:b`):

```
LinMod.6 <- lm(tip ~ smoker*total_bill, data = tips)
summary(LinMod.6)
```

```
##
## Call:
## lm(formula = tip ~ smoker * total_bill, data = tips)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -2.6789 -0.5238 -0.1205  0.4749  4.8999
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)           0.360069   0.202058   1.782 0.076012 .
## smokerYes             1.204203   0.312263   3.856 0.000148 ***
## total_bill            0.137156   0.009678  14.172  < 2e-16 ***
## smokerYes:total_bill -0.067566   0.014189  -4.762 3.32e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9785 on 240 degrees of freedom
## Multiple R-squared:  0.506,  Adjusted R-squared:  0.4998
## F-statistic: 81.95 on 3 and 240 DF,  p-value: < 2.2e-16
```

Der Schätzwert für die Interaktion steht bei :. Hier also: Wenn geraucht wurde, ist die Steigung im Durchschnitt um 6,8 Cent geringer. Aber wenn geraucht wurde, ist die Rechnung im Achsenabschnitt erstmal um 1,20$ höher (Effekt, ceteris paribus). Wer will, kann ausrechnen, ab welcher Rechnungshöhe Rauchertische im Mittelwert lukrativer sind...

Das gleiche Bild (höhere Achsenabschnitt, geringere Steigung) ergibt sich übrigens bei getrennten Regressionen:

```
lm(tip~total_bill, data=tips, subset = smoker=="Yes")
```

```
##
## Call:
## lm(formula = tip ~ total_bill, data = tips, subset = smoker ==
##     "Yes")
##
## Coefficients:
## (Intercept)   total_bill
##     1.56427      0.06959
```

```
lm(tip~total_bill, data=tips, subset = smoker=="No")
```

```
##
## Call:
## lm(formula = tip ~ total_bill, data = tips, subset = smoker ==
##     "No")
##
## Coefficients:
## (Intercept)   total_bill
```

```
##      0.3601       0.1372
```

Weitere Modellierungsmöglichkeiten

Über das Formelinterface `y~x` können auch direkt z. B. Polynome modelliert werden. Hier eine quadratische Funktion:

```
summary(lm(tip~I(total_bill^2)+total_bill, data=tips))
```

```
##
## Call:
## lm(formula = tip ~ I(total_bill^2) + total_bill, data = tips)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2005 -0.5586 -0.0979  0.4838  3.7762
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      0.8911170  0.3467554   2.570 0.010776 *
## I(total_bill^2) -0.0000571  0.0006025  -0.095 0.924573
## total_bill       0.1078555  0.0307696   3.505 0.000544 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.024 on 241 degrees of freedom
## Multiple R-squared:  0.4566, Adjusted R-squared:  0.4521
## F-statistic: 101.3 on 2 and 241 DF,  p-value: < 2.2e-16
```

D. h., die geschätzte Funktion ist eine "umgedrehte Parabel" (negatives Vorzeichen bei `I(total_bill^2)`), bzw. die Funktion ist konkav, die Steigung nimmt ab. Allerdings ist der Effekt nicht signifikant. **Hinweis:** Um zu "rechnen" und nicht beispielsweise Interaktion zu modellieren, geben Sie die Variablen in der Formel in der Funktion `I()` (*As Is*) ein.

Prognoseintervalle

Insgesamt haben wir viel "Unsicherheit" u. a. aufgrund von Variabilität in den Beobachtungen und in den Schätzungen. Wie wirken sich diese auf die Prognose aus?

Dazu können wir über die Funktion `predict.lm` Prognoseintervalle berechnen – hier für das einfache Modell `LinMod.1`:

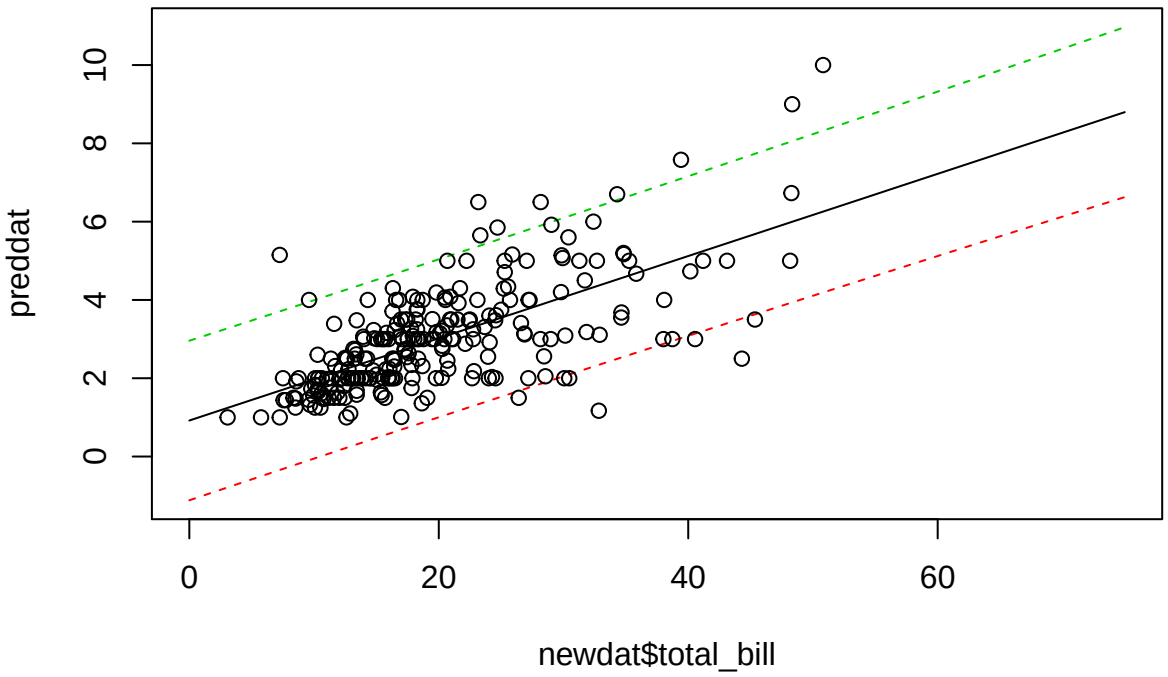
```
newdat <- data.frame(total_bill = seq(0, 75))
preddat <- predict(LinMod.1, newdata = newdat, interval = "prediction")
head(preddat)
```

```
##          fit        lwr      upr
## 1 0.9202696 -1.1174151 2.957954
## 2 1.0252941 -1.0103977 3.060986
## 3 1.1303186 -0.9034818 3.164119
## 4 1.2353432 -0.7966677 3.267354
## 5 1.3403677 -0.6899557 3.370691
## 6 1.4453922 -0.5833461 3.474130
```

```
tail(preddat)
```

```
##          fit      lwr      upr
## 71 8.271986 6.127124 10.41685
## 72 8.377010 6.227178 10.52684
## 73 8.482035 6.327145 10.63692
## 74 8.587059 6.427028 10.74709
## 75 8.692084 6.526825 10.85734
## 76 8.797108 6.626538 10.96768
```

```
matplot(newdat$total_bill, preddat, lty = c(1,2,2), type="l" )
points(x=tips$total_bill, y=tips$tip)
```



Sie sehen, dass 95% Prognoseintervall ist recht breit: über den gewählten Rechnungsbereich von $0 - 75$\$ im Mittelwert bei 4.11\$.

```
favstats((preddat[,3]-preddat[,2]))
```

```
##       min       Q1   median       Q3      max     mean         sd  n
##  4.034738 4.044428 4.072224 4.171158 4.341141 4.115859 0.09042352 76
##  missing
##        0
```

Zu den Rändern hin wird es breiter. Am schmalsten ist es übrigens beim Mittelwert der unabhängigen Beobachtungen, hier also bei 19.79$.

Kreuzvalidierung

Je komplexer und flexibler ein Modell ist, desto besser kann es sich an die *vorhandenen*, sogenannte Traininingsdaten anpassen, **aber** für *neue*, sogenannte Testdaten wird es nicht immer besser – siehe z. B. Kapitel 2.2 aus James et. al (2013).

Allgemein können Modelle mit Hilfe des Mean Squared Error verglichen werden:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2$$

Vorhandene Daten können aber genutzt werden um die Prognose für neue Daten (x_0, y_0) zu simulieren: z. B. über Kreuzvalidierung. Im einfachen Fall einer Leave-One-Out Kreuzvalidierung werden alle Beobachtungen bis auf die i-te, $\quad i = 1, 2, \ldots n$ zum Schätzen oder Lernen des Modells verwendet, das Testen des Modells erfolgt dann anhand der Prognosegüte für die i-te, $\quad i = 1, 2, \ldots n$ Beobachtung. In R kann dies einfach über *Schleifen* durchgeführt werden:

```
n <- nrow(tips) # Anzahl Beobachtungen
y <- tips$tip # "Wahre" Werte
yprog <- numeric(n) # Vektor in dem die Prognosen geschrieben werden

### Modellanpassung
mod <- lm(tip ~ ., data=tips) # Modell mit allen Beobachtungen schätzen
yfit <- mod$fitted.values # "Vorhersagen" für Trainingsdaten

### Leave-One-Out Kreuzvalidierung
for (i in 1:n) # i nehme nacheinander die Werte von 1 bis n an
{
  modloo <- lm(tip ~ ., data=tips[-i,]) # Modell schätzen ohne i-te Beobachtung
  yprog[i] <- predict(modloo, newdata = tips[i,]) # Vorhsage von y_i anhand des Modells
}

### Vergleich:
MSEfit <- mean((y-yfit)^2)
```

```r
MSEprog <- mean((y-yprog)^2)


cat("MSE Modellanpassung: ", MSEfit, "\n")
```

```
## MSE Modellanpassung:  1.010354
```

```r
cat("MSE Kreuzvalidierung: ", MSEprog, "\n")
```

```
## MSE Kreuzvalidierung:  1.100501
```

Der Mean Squared Error ist also bei der Leave-One-Out Kreuzvalidierung um 9% schlechter als bei der Modellanpassung.

Übung: Teaching Rating

Dieser Datensatz analysiert u. a. den Zusammenhang zwischen Schönheit und Evaluierungsergebnis von Dozenten:

Hamermesh, D.S., and Parker, A. (2005). Beauty in the Classroom: Instructors' Pulchritude and Putative Pedagogical Productivity. Economics of Education Review, 24, 369–376.

Sie können ihn, sofern noch nicht geschehen, von https://goo.gl/6Y3KoK als `csv` herunterladen.

Versuchen Sie, das Evaluierungsergebnis als abhängige Variable anhand geeigneter Variablen des Datensatzes zu erklären. Wie groß ist der Einfluss der Schönheit? Sind die Modellannahmen erfüllt und wie beurteilen Sie die Modellgüte?

Literatur

- David M. Diez, Christopher D. Barr, Mine Çetinkaya-Rundel (2014): *Introductory Statistics with Randomization and Simulation*, https://www.openintro.org/stat/textbook.php?stat_book=isrs, Kapitel 5, 6.1-6.3
- Nicholas J. Horton, Randall Pruim, Daniel T. Kaplan (2015): Project MOSAIC Little Books *A Student's Guide to R*, https://github.com/ProjectMOSAIC/LittleBooks/raw/master/StudentGuide/MOSAIC-StudentGuide.pdf, Kapitel 5.4, 10.2
- Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani (2013): *An Introduction to Statistical Learning – with Applications in R*, http://www-bcf.usc.edu/~gareth/ISL/, Kapitel 3
- Maike Luhmann (2015): *R für Einsteiger*, Kapitel 16, 17.1-17.3
- Andreas Quatember (2010): *Statistik ohne Angst vor Formeln*, Kapitel 3.11
- Daniel Wollschläger (2014): *Grundlagen der Datenanalyse mit R*, Kapitel 6

Versionshinweise:

- Datum erstellt: 2017-06-30
- R Version: 3.4.0
- `mosaic` Version: 0.14.4

Kapitel 6: Einführung Logistische Regression

Vorbereitung

Hier werden wir den Datensatz *Aktienkauf* der Universität Zürich (Universität Zürich, Methodenberatung) analysieren. Es handelt es sich hierbei um eine Befragung einer Bank im Zusammenhang mit den Fakten, die mit der Wahrscheinlichkeit, dass jemand Aktien erwirbt, zusammenhängen. Es wurden 700 Personen befragt. Folgende Daten wurden erhoben: Aktienkauf (0 = nein, 1 = ja), Jahreseinkommen (in Tausend CHF), Risikobereitschaft (Skala von 0 bis 25) und Interesse an der aktuellen Marktlage (Skala von 0 bis 45).

Den Datensatz können Sie in von hier als `csv`-Datei herunterladen:

```
download.file("https://goo.gl/AiUSSI", destfile = "Aktienkauf.csv")
```

Das Einlesen erfolgt, sofern die Daten im Arbeitsverzeichnis liegen, über:

```
Aktien <- read.csv2("Aktienkauf.csv")
```

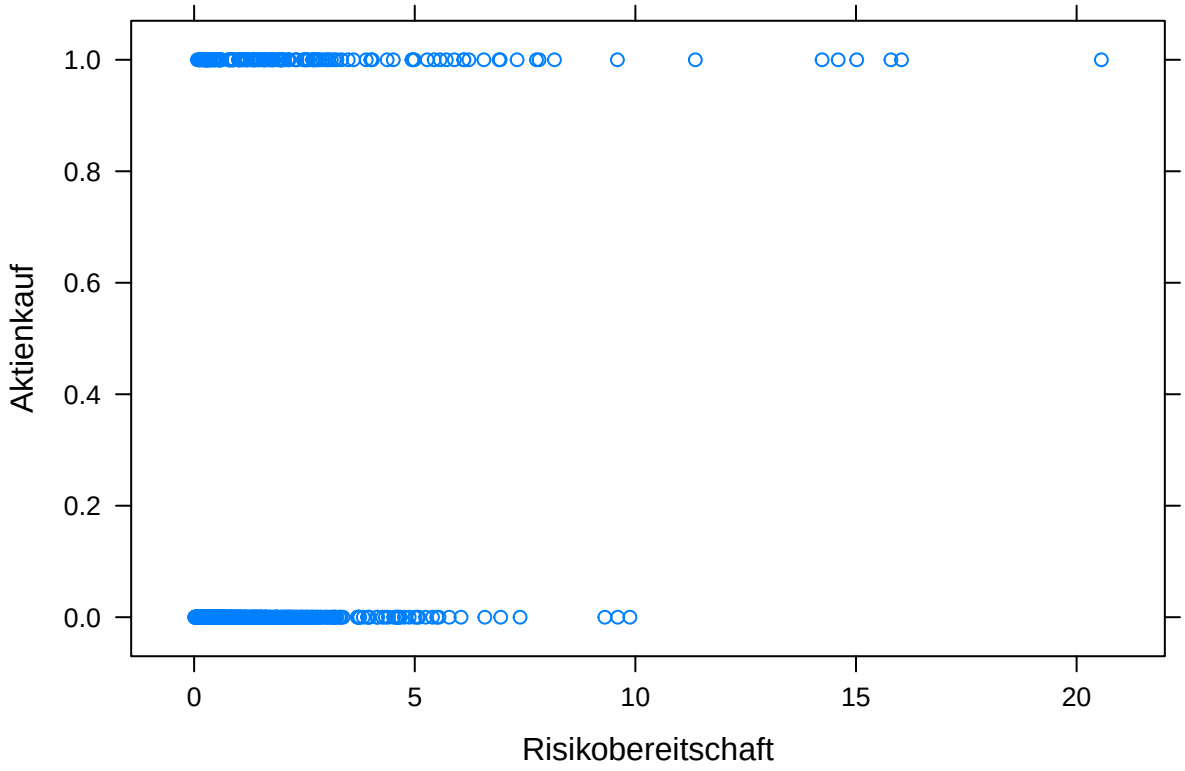
Zur Unterstützung der Analyse wird (wieder) `mosaic` verwendet.

```
library(mosaic)
```

Problemstellung

Können wir anhand der Risikobereitschaft abschätzen, ob die Wahrscheinlichkeit für einen Aktienkauf steigt? Schauen wir uns zunächst ein Streudiagramm an:

```
xyplot(Aktienkauf ~ Risikobereitschaft, data = Aktien)
```

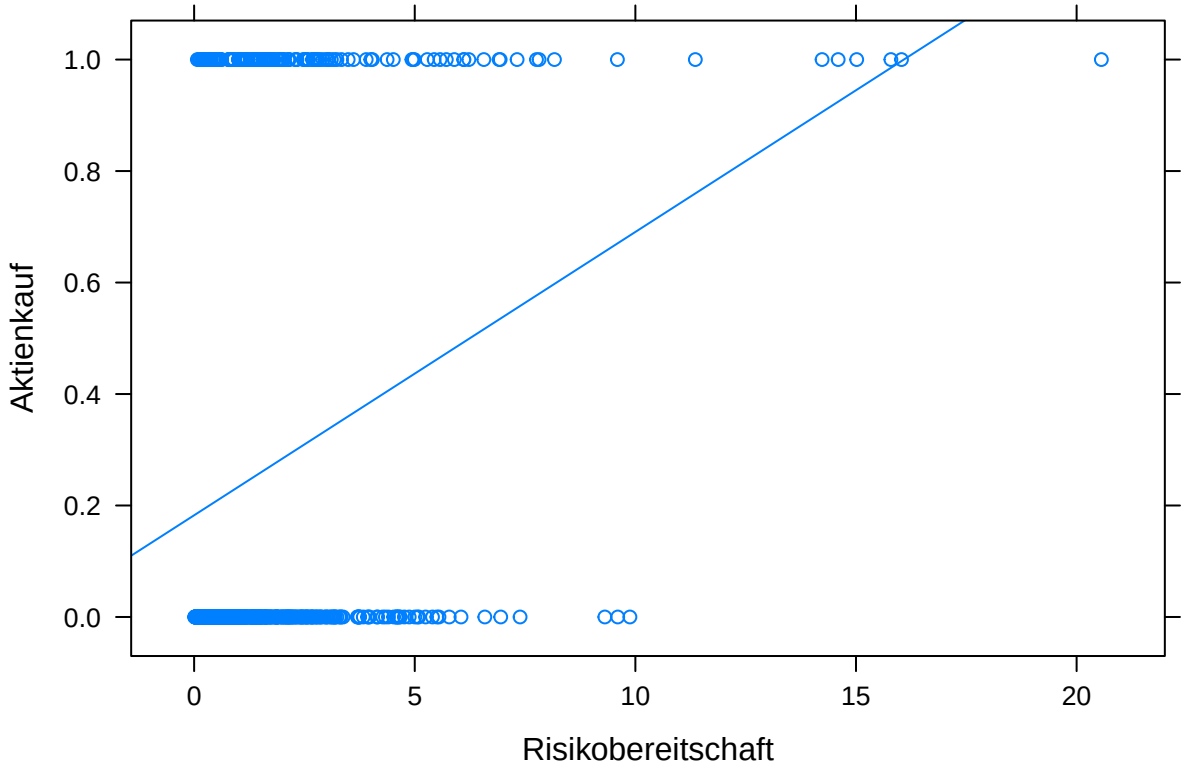


Der Zusammenhang scheint nicht sehr ausgeprägt zu sein. Lassen Sie uns dennoch ein lineare Regression durchführen und das Ergebnis auswerten und graphisch darstellen.

```
lm1 <- lm(Aktienkauf ~ Risikobereitschaft, data = Aktien)
summary(lm1)
```

```
##
## Call:
## lm(formula = Aktienkauf ~ Risikobereitschaft, data = Aktien)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.6845 -0.2434 -0.2044  0.3480  0.8138
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.182460   0.020007   9.120  < 2e-16 ***
## Risikobereitschaft 0.050831   0.007622   6.669 5.25e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4267 on 698 degrees of freedom
## Multiple R-squared:  0.0599, Adjusted R-squared:  0.05855
## F-statistic: 44.47 on 1 and 698 DF,  p-value: 5.249e-11
```

```
plotModel(lm1)
```



Der Schätzer für die Steigung für `Risikobereitschaft` ist signifikant. Das Bestimmtheitsmaß R^2 ist allerdings sehr niedrig, aber wir haben bisher ja auch nur eine unabhängige Variable für die Erklärung der abhängigen Variable herangezogen.

Doch was bedeutet es, dass die Wahrscheinlichkeit ab einer Risikobereitsschaft von ca. 16 über 1 liegt?

Wahrscheinlichkeiten müssen zwischen 0 und 1 liegen. Daher brauchen wir eine Funktion, die das Ergebnis einer linearen Regression in einen Bereich von 0 bis 1 bringt, die sogenannte *Linkfunktion*. Eine häufig dafür verwendete Funktion ist die logistische Funktion:

$$p(y = 1) = \frac{e^{\eta}}{1 + e^{\eta}} = \frac{1}{1 + e^{-\eta}}$$

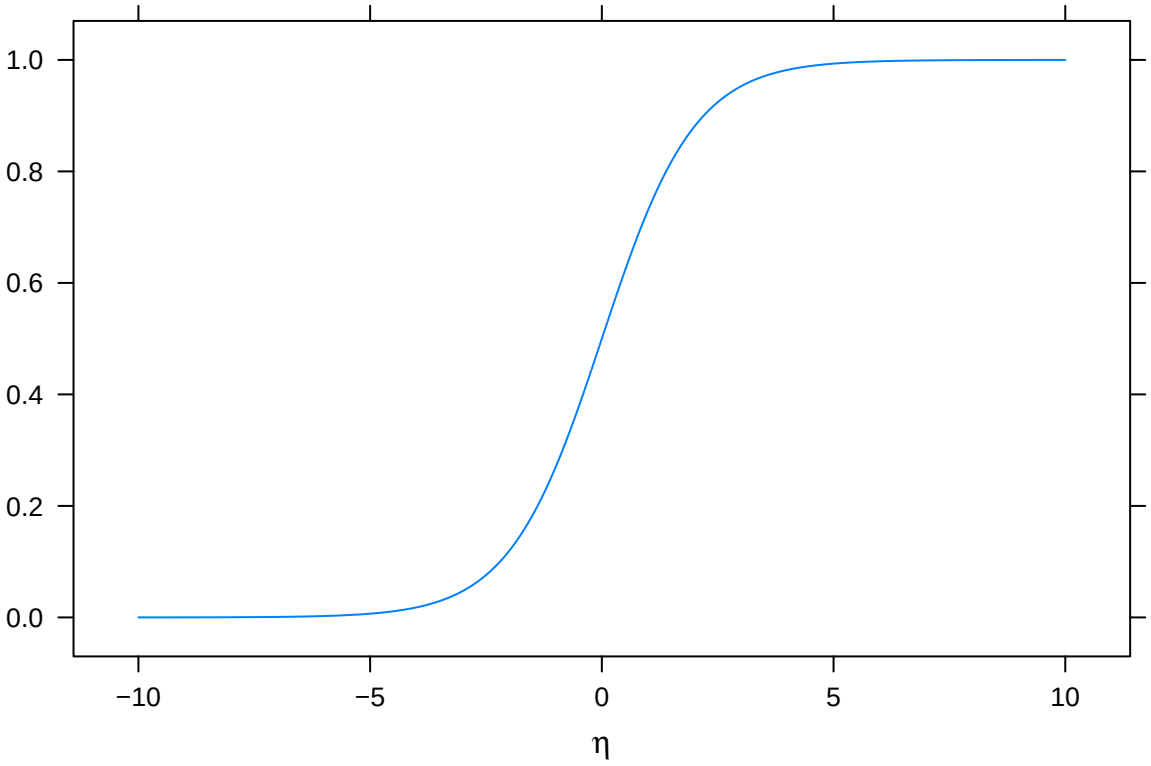
η, das sogenannte *Logit*, ist darin die Linearkombination der Einflussgrößen:

$$\eta = \beta_0 + \beta_1 \cdot x_1 + \ldots$$

Exemplarisch können wir die logistische Funktion für einen Bereich von $\eta = -10$ bis $+10$ darstellen:

```
# eta-Werte von -10 bis +10 erzeugen
eta <- seq(-10,10,by = 0.1)
# y-Werte mit logistischer Funktion berechnen
y <- 1/(1+exp(-eta))          # exp() ist die e-Funktion
# Graphik ausgeben mit fogenden Plotparametern:
# für das Label der x-Achse wird ein mathematisches Symbol genutzt
```

```
# Label der y-Achse wird nicht angezeigt
# statt Punkten wird eine Liniengraphik ausgegeben
xyplot(y ~ eta, xlab = expression(eta), ylab = "", type = "l")
```



Logistische Regression

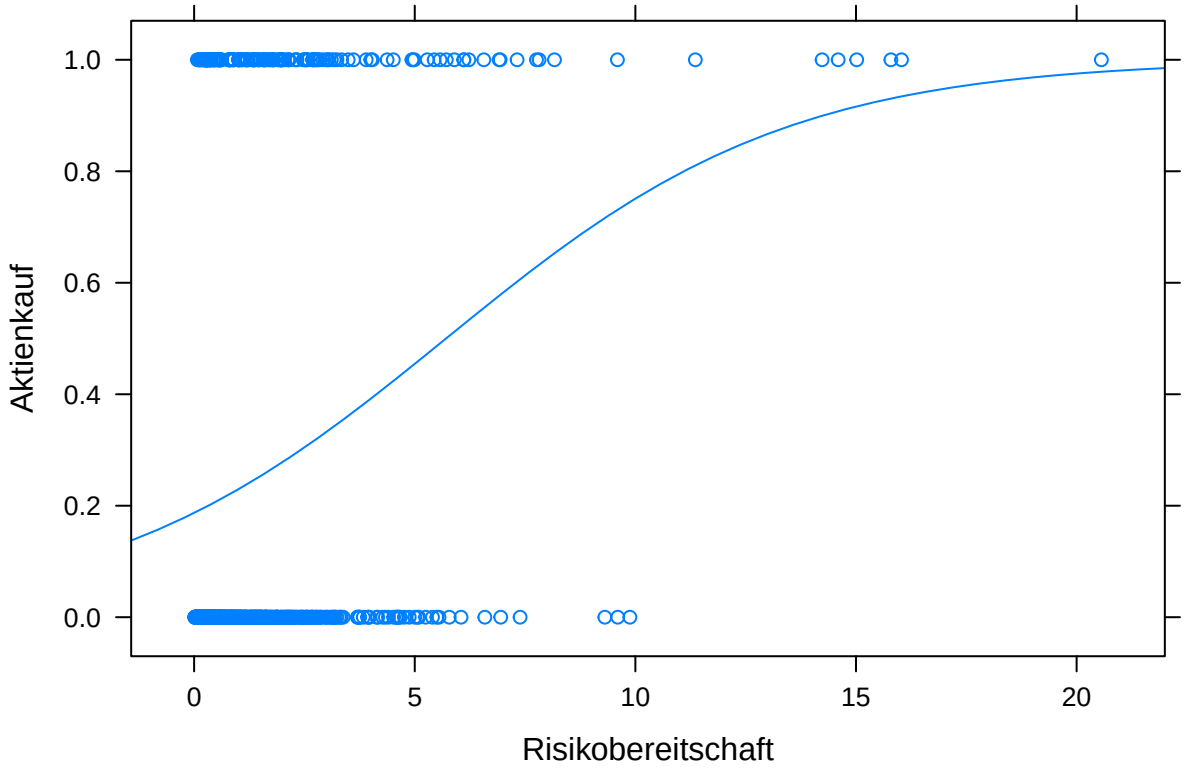
Die logistische Regression ist eine Anwendung des allgemeinen linearen Modells (*general linear model, GLM*). Die Modellgleichung lautet:

$$p(y_i = 1) = L\big(\beta_0 + \beta_1 \cdot x_{i1} + \cdots + \beta_K \cdot x_{ik}\big) + \epsilon_i$$

L ist die Linkfunktion, in unserer Anwendung die logistische Funktion.
x_{ik} sind die beobachten Werte der unabhängigen Variablen X_k.
k sind die unabhängigen Variablen 1 bis K.

Die Funktion `glm` führt die logistische Regression durch. Wir schauen uns im Anschluss zunächst den Plot an.

```
glm1 <- glm(Aktienkauf ~ Risikobereitschaft, family = binomial("logit"),
            data = Aktien)
plotModel(glm1)
```



Es werden ein Streudiagramm der beobachten Werte sowie die *Regressionslinie* ausgegeben. Wir können so z. B. ablesen, dass ab einer Risikobereitschaft von etwa 7 die Wahrscheinlichkeit für einen Aktienkauf nach unserem Modell bei mehr als 50 % liegt.

Die Zusammenfassung des Modells zeigt folgendes:

```
summary(glm1)
```

```
##
## Call:
## glm(formula = Aktienkauf ~ Risikobereitschaft, family = binomial("logit"),
##     data = Aktien)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6531  -0.7384  -0.6766   0.8247   1.8226
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -1.46895    0.11842 -12.405  < 2e-16 ***
## Risikobereitschaft  0.25726    0.04676   5.501 3.77e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 804.36  on 699  degrees of freedom
## Residual deviance: 765.86  on 698  degrees of freedom
## AIC: 769.86
##
## Number of Fisher Scoring iterations: 4
```

Der Achsenabschnitt (`intercept`) des logits η wird mit -1.47 geschätzt, die Steigung in Richtung `Risikobereitschaft` mit 0.26. Die (Punkt-)Prognose für die Wahrscheinlichkeit eines Aktienkaufs $p(y = 1)$ benötigt anders als in der linearen Regression noch die Linkfunktion und ergibt sich somit zu:

$$p(\texttt{Aktienkauf} = 1) = \frac{1}{1 + e^{-(-1.47 + 0.26 \cdot \texttt{Risikobereitschaft})}}$$

Die p-Werte der Koeffizienten können in der Spalte `Pr(>|z|)` abgelesen werden. Hier wird ein *Wald*-Test durchgeführt, nicht wie bei der linearen Regression ein t-Test, ebenfalls mit der $H_0 : \beta_i = 0$. Die Teststastistik (`z value`) wird wie in der linearen Regression durch Divisions des Schätzers (`Estimate`) durch den Standardfehler (`Std. Error`) ermittelt. Im *Wald*-Test ist die Teststatistik allerdings χ^2-verteilt mit einem Freiheitsgrad.

Welche Unterschiede zur linearen Regression gibt es in der Ausgabe?

Es gibt kein R^2 im Sinne einer erklärten Streuung der y-Werte, da die beobachteten y-Werte nur 0 oder 1 annehmen können. Das Gütemaß bei der logistischen Regression ist das *Akaike Information Criterion* (*AIC*). Hier gilt allerdings: je **kleiner**, desto **besser**. (Anmerkung: es kann ein Pseudo-R^2 berechnet werden – kommt später.)

Es gibt keine F-Statistik (oder ANOVA) mit der Frage, ob das Modell als Ganzes signifikant ist. (Anmerkung: es kann aber ein vergleichbarer Test durchgeführt werden – kommt später.)

Interpretation der Koeffizienten

y-Achsenabschnitt (`Intercept`) β_0

Für $\beta_0 > 0$ gilt, dass selbst wenn alle anderen unabhängigen Variablen 0 sind, es eine Wahrscheinlichkeit von mehr als 50% gibt, dass das modellierte Ereignis eintritt. Für $\beta_0 < 0$ gilt entsprechend das Umgekehrte.

Steigung β_i **mit** $i = 1, 2, ..., K$

Für $\beta_i > 0$ gilt, dass mit zunehmenden x_i die Wahrscheinlichkeit für das modellierte Ereignis steigt. Bei $\beta_i < 0$ nimmt die Wahrscheinlichkeit entsprechend ab.

Eine Abschätzung der Änderung der Wahrscheinlichkeit (*relatives Risiko*, *relative risk RR*) kann über das

Chancenverhältnis (*Odds Ratio OR*) gemacht werden.[3] Es ergibt sich vereinfacht e^{β_i}. Die Wahrscheinlichkeit ändert sich näherungsweise um diesen Faktor, wenn sich x_i um eine Einheit erhöht. **Hinweis:** $RR \approx OR$ gilt nur, wenn der Anteil des modellierten Ereignisses in den beobachteten Daten sehr klein ($< 5\%$) oder sehr groß ist ($> 95\%$).

Übung: Berechnen Sie das relative Risiko für unser Beispielmodell, wenn sich die `Risikobereitschaft` um 1 erhöht (Funktion `exp()`). Vergleichen Sie das Ergebnis mit der Punktprognose für `Risikobereitschaft=` 7 im Vergleich zu `Risikobereitschaft=` 8. Zur Erinnerung: Sie können `makeFun(model)` verwenden.

```
# aus Koeffizient abgeschätzt
exp(coef(glm1)[2])
```

```
## Risikobereitschaft
##           1.293379
```

```
# mit dem vollständigen Modell berechnet
fun1 <- makeFun(glm1)
fun1(Risikobereitschaft = 7)
```

```
##         1
## 0.582213
```

```
fun1(Risikobereitschaft = 8)
```

```
##         1
## 0.6431639
```

```
# als Faktor ausgeben
fun1(Risikobereitschaft = 8)/fun1(Risikobereitschaft = 7)
```

```
##         1
## 1.104688
```

Sie sehen also, die ungefähr abgeschätzte Änderung der Wahrscheinlichkeit weicht hier doch deutlich von der genau berechneten Änderung ab. Der Anteil der Datensätze mit `Risikobereitschaft=` 1 liegt allerdings auch bei 0.26.

Kategoriale Variablen

Wie schon in der linearen Regression können auch in der logistschen Regression kategoriale Variablen als unabhängige Variablen genutzt werden. Als Beispiel nehmen wir den Datensatz `tips` und versuchen abzuschätzen, ob sich die Wahrscheinlichkeit dafür, dass ein Raucher bezahlt hat (`smoker = yes`), in Abhängigkeit vom Wochentag ändert.

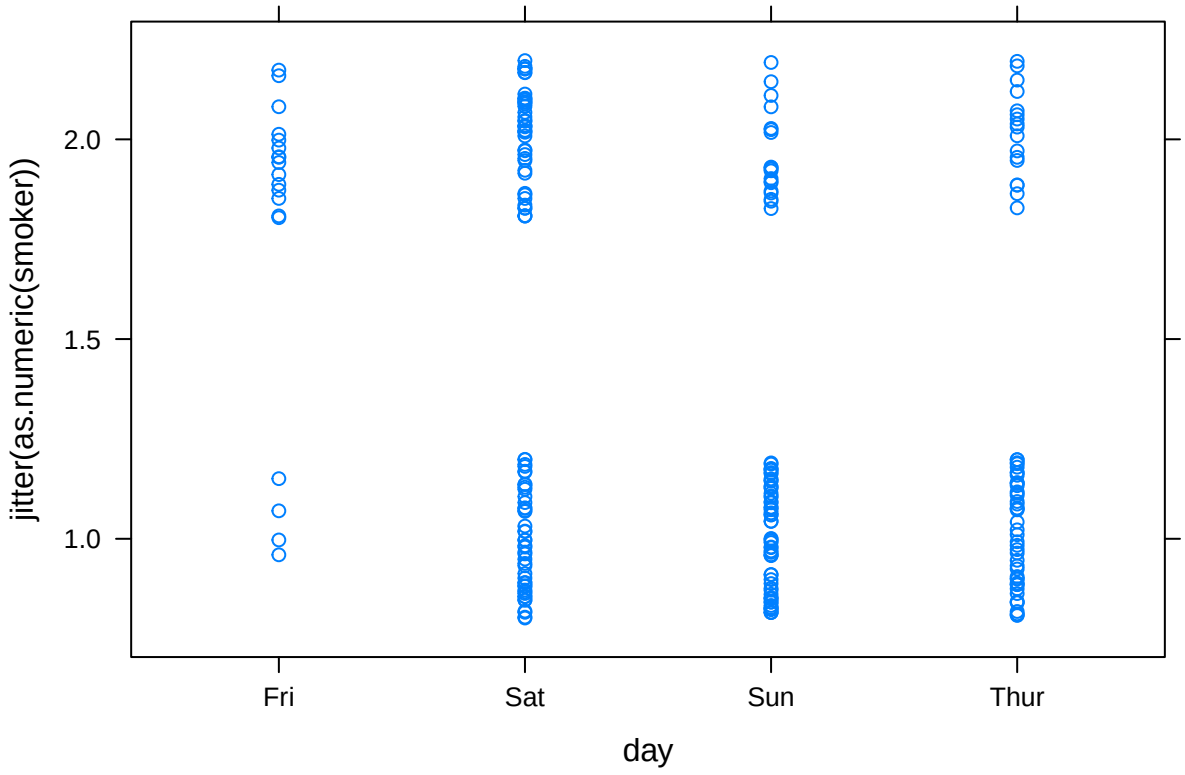
Sofern noch nicht geschehen, können Sie in hier als `csv`-Datei herunterladen:

[3] Wahrscheinlichkeit vs. Chance: Die Wahrscheinlichkeit bei einem fairen Würfel, eine 6 zu würfeln, ist 1/6. Die Chance (*Odd*), eine 6 zu würfeln, ist die Wahrscheinlichkeit dividiert durch die Gegenwahrscheinlichkeit, also $\frac{1/6}{5/6} = 1/5$.

```
download.file("https://goo.gl/whKjnl", destfile = "tips.csv")
```

Zunächst ein Plot:

```
tips <- read.csv2("tips.csv")
xyplot(jitter(as.numeric(smoker)) ~ day, data = tips)
```



Hinweis: Um zu sehen, ob es an manchen Tagen mehr Raucher gibt, sollten Sie zumindest eine Variable "verrauschen" ("*jittern*"). Da die Variable `smoker` eine nominale Variable ist und die Funktion `jitter()` nur mit numerischen Variablen arbeitet, muss sie mit `as.numeric()` in eine numerische Variable umgewandelt werden.

Die relativen Häufigkeiten zeigt folgende Tabelle:

```
(tab_smoke <- tally(smoker ~ day, data = tips, format = "proportion"))
```

```
##        day
## smoker       Fri       Sat       Sun      Thur
##    No  0.2105263 0.5172414 0.7500000 0.7258065
##    Yes 0.7894737 0.4827586 0.2500000 0.2741935
```

Hinweis: Durch die Klammerung wird das Objekt `tab_smoke` direkt ausgegeben.

Probieren wir die logistische Regression aus:

```
glmtips <- glm(smoker ~ day, family = binomial("logit"),data = tips)
summary(glmtips)
```

```
##
```

```
## Call:
## glm(formula = smoker ~ day, family = binomial("logit"), data = tips)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7653  -0.8006  -0.7585   1.2068   1.6651
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.3218     0.5627   2.349 0.018833 *
## daySat       -1.3907     0.6022  -2.309 0.020928 *
## daySun       -2.4204     0.6220  -3.891 9.96e-05 ***
## dayThur      -2.2952     0.6306  -3.639 0.000273 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 324.34  on 243  degrees of freedom
## Residual deviance: 298.37  on 240  degrees of freedom
## AIC: 306.37
##
## Number of Fisher Scoring iterations: 4
```

Auch hier können wir die Koeffizienten in Relation zur Referenzkategorie (hier: Freitag) interpretieren. Die Wahrscheinlichkeit ist an einem Samstag niedriger, der Wert für `daySat` ist negativ. Eine Abschätzung erhalten wir wieder mit e^{β_i}:

```
exp(coef(glmtips)[2])
```

```
##    daySat
## 0.2488889
```

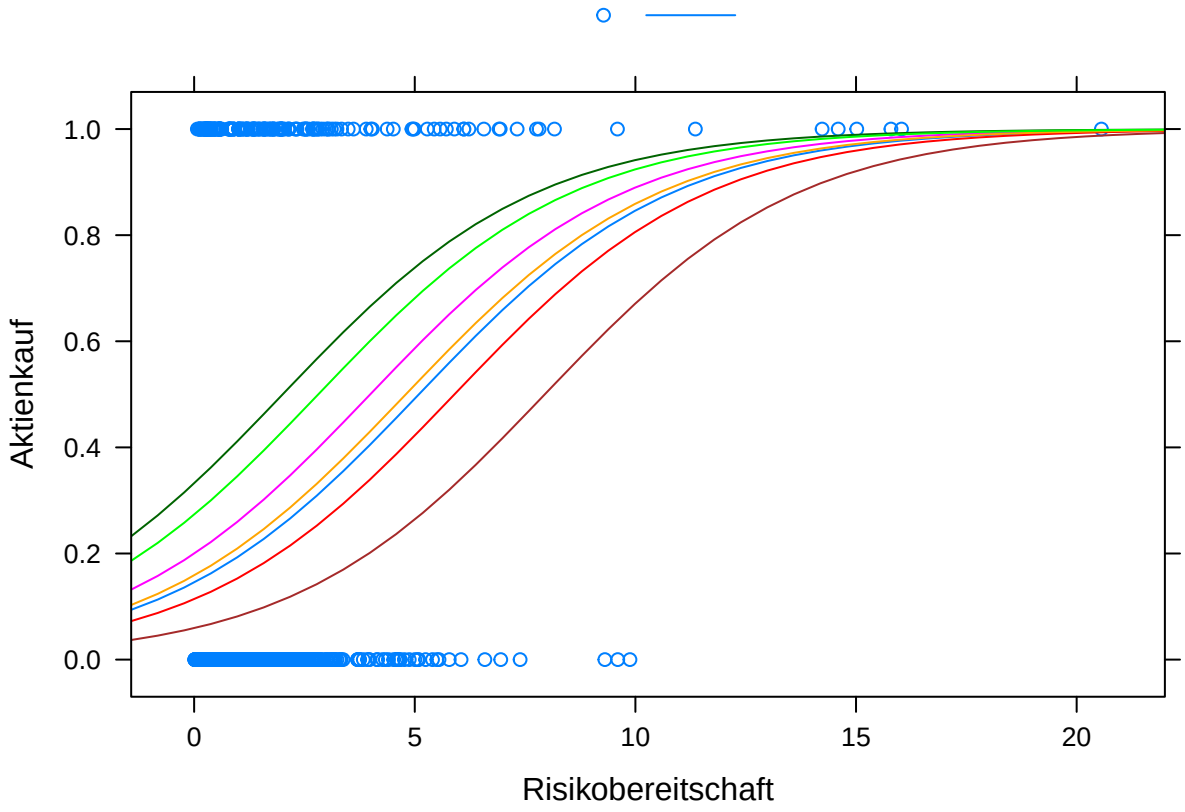
Daher ist das Chancenverhältnis (*Odds Ratio*), dass am Samstag ein Raucher am Tisch sitzt, näherungsweise um den Faktor 0.25 niedriger als am Freitag:

$$OR = \frac{\frac{P(Raucher|Samstag)}{1-P(Raucher|Samstag)}}{\frac{P(Raucher|Freitag)}{1-P(Raucher|Freitag)}} = \frac{\frac{0.4828}{0.5172}}{\frac{0.7895}{0.2105}} \approx 0.2489$$

Multiple Regression

Wir kehren wieder zurück zu dem Datensatz *Aktienkauf*. Können wir unser Model `glm1` mit nur einer erklärenden Variable verbessern, indem weitere unabhängige Variablen hinzugefügt werden?

```
glm2 <- glm(Aktienkauf ~ Risikobereitschaft + Einkommen + Interesse,
            family = binomial("logit"),data = Aktien)
plotModel(glm2)
```




```
summary(glm2)
```

```
##
## Call:
## glm(formula = Aktienkauf ~ Risikobereitschaft + Einkommen + Interesse,
##     family = binomial("logit"), data = Aktien)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1303  -0.7150  -0.5390   0.5178   3.2143
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -1.667913   0.279029  -5.978 2.27e-09 ***
## Risikobereitschaft   0.347805   0.088224   3.942 8.07e-05 ***
## Einkommen           -0.021573   0.005636  -3.828 0.000129 ***
## Interesse            0.085200   0.017751   4.800 1.59e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 804.36  on 699  degrees of freedom
## Residual deviance: 679.01  on 696  degrees of freedom
## AIC: 687.01
##
## Number of Fisher Scoring iterations: 5
```

Alle Schätzer sind signifkant zum 0.1 %-Niveau (⋆⋆⋆ in der Ausgabe). Zunehmende Risikobereitschaft (der Einfluss ist im Vergleich zum einfachen Modell stärker geworden) und zunehmendes Interesse erhöhen die Wahrscheinlichkeit für einen Aktienkauf. Steigendes Einkommen hingegen senkt die Wahrscheinlichkeit.

Ist das Modell besser als das einfache? Ja, da der AIC-Wert von 769.86 auf 687.01 gesunken ist.

Die Graphik zeigt die Verläufe in Abhängigkeit von den verschiedenen Variablen und den Kombinationen der Variablen.

Erweiterungen

Klassifikationsgüte

Logistische Regressionsmodelle werden häufig zur Klassifikation verwendet, z. B. ob der Kredit für einen Neukunden ein "guter" Kredit ist oder nicht. Daher sind die Klassifikationseigenschaften bei logistischen Modellen wichtige Kriterien.

Hierzu werden die aus dem Modell ermittelten Wahrscheinlichkeiten ab einem Schwellenwert (*cutpoint*), häufig 0.5, einer geschätzten 1 zugeordnet, unterhalb des Schwellenwertes einer 0. Diese aus dem Modell ermittelten Häufigkeiten werden dann in einer sogenannten Konfusionsmatrix (*confusion matrix*) mit den beobachteten Häufigkeiten verglichen.

Daher sind wichtige Kriterien eines Modells, wie gut diese Zuordnung erfolgt. Dazu werden die Sensitivität (*True Positive Rate, TPR*), also der Anteil der mit 1 geschätzten an allen mit 1 beobachten Werten, und die Spezifität (*True Negative Rate*) berechnet. Ziel ist es, dass beide Werte möglichst hoch sind.

Sie können die Konfusionsmatrix "zu Fuß" berechnen, in dem Sie eine neue Variable einfügen, die ab dem cutpoint 1 und sonst 0 ist und mit dem Befehl `tally()` ausgeben. Alternativ können Sie das Paket `SDMTools` verwenden mit der Funktion `confusion.matrix()`. Ein Parameter ist `cutpoint`, der standardmäßig auf 0.5 steht.

```
# Konfusionsmatrix "zu Fuß" berechnen
# cutpoint = 0.5 setzen
# neue Variable predicted anlegen mit 1, wenn modellierte Wahrscheinlichkeit > 1 ist
cutpoint = 0.5
Aktien$predicted <- ((glm1$fitted.values) > cutpoint)*1
```

```
# Kreuztabelle berechnen
(cm <- tally(~predicted+Aktienkauf, data = Aktien))
```

```
##           Aktienkauf
## predicted   0   1
##         0 509 163
##         1   8  20
```

```
# Sensitivität (TPR)
cm[2,2]/sum(cm[,2])
```

```
## [1] 0.1092896
```

```
# Spezifität (TNR)
cm[1,1]/sum(cm[,1])
```

```
## [1] 0.9845261
```

```
# mit Hilfe des Pakets SDMTools
# ggf. install.packages("SDMTools")
library(SDMTools)
# optional noch Parameter cutpoint = 0.5  angeben
(cm <- confusion.matrix(Aktien$Aktienkauf, glm1$fitted.values))
```

```
##      obs
## pred   0   1
##    0 509 163
##    1   8  20
## attr(,"class")
## [1] "confusion.matrix"
```

```
sensitivity(cm)
```

```
## [1] 0.1092896
```

```
specificity(cm)
```

```
## [1] 0.9845261
```

Wenn die Anteile der 1 in den beobachteten Daten sehr gering sind (z. B. bei einem medizinischem Test auf eine seltene Krankheit, Klicks auf einen Werbebanner oder Kreditausfall), kommt eine Schwäche der logistischen Regression zum Tragen: Das Modell wird so optimiert, dass die Wahrscheinlichkeiten $p(y = 1)$ alle unter 0.5 liegen. Das würde zu einer Sensitität von 0 und einer Spezifiät von 1 führen. Daher kann es sinnvoll sein, den Cutpoint zu varieren. Daraus ergibt sich ein verallgemeinertes Gütemaß, die *ROC*-Kurve (*Return Operating Characteristic*) und den daraus abgeleiteten *AUC*-Wert (*Area Under Curve*).

Hierzu wird der Cutpoint zwischen 0 und 1 variiert und die Sensitivität gegen 1−Spezifität (welche

Werte sind als 1 modelliert worden unter den beobachten 0, *False Positive Rate, FPR*). Um diese Werte auszugeben, benötigen Sie das Paket `ROCR` und die Funktion `performance()`.

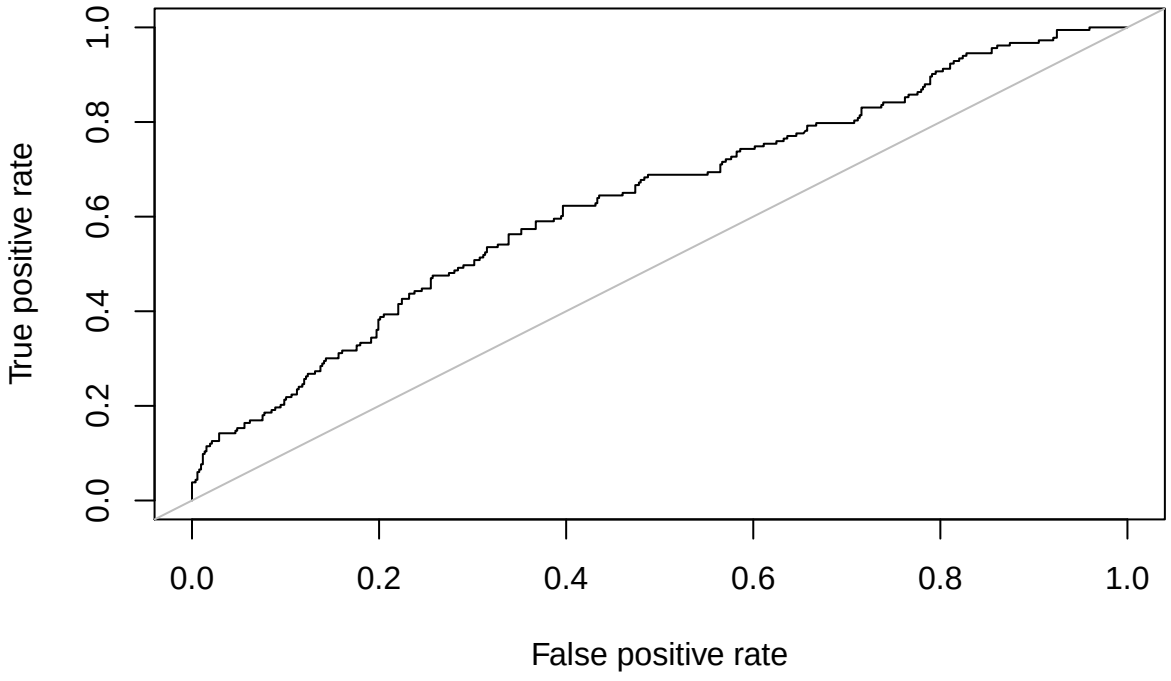
```
# ggf. install.packages("ROCR")
library(ROCR)
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
# Ein für die Auswertung notwendiges prediction Objekt anlegen
pred <- prediction(glm1$fitted.values, Aktien$Aktienkauf)
# ROC Kurve
perf <- performance(pred,"tpr","fpr")
plot(perf)
abline(0,1, col = "grey")
```




```
# Area under curve (ROC-Wert)
performance(pred,"auc")@y.values
```

```
## [[1]]
## [1] 0.6361892
```

AUC liegt zwischen 0.5, wenn das Modell gar nichts erklärt (im Plot die graue Linie) und 1. Hier ist der

Wert also recht gering. Akzeptable Werte liegen bei 0.7 und größer, gute Werte sind es ab 0.8.[4]

Modellschätzung

Das Modell wird nicht wie bei der lineare Regression über die Methode der kleinsten Quadrate (OLS) geschätzt, sondern über die *Maximum Likelihood* Methode. Die Koeffizienten werden so gewählt, dass die beobachteten Daten am wahrscheinlichsten (*Maximum Likelihood*) werden.

Das ist ein iteratives Verfahren (OLS erfolgt rein analytisch), daher wird in der letzten Zeile der Ausgabe auch die Anzahl der Iterationen (`Fisher Scoring Iterations`) ausgegeben.

Die Devianz des Modells (`Residual deviance`) ist -2 mal die logarithmierte Likelihood. Die Nulldevianz (`Null deviance`) ist die Devianz eines Nullmodells, d. h., alle β außer der Konstanten sind 0.

Likelihood Quotienten Test

Der Likelihood Quotienten Test (*Likelihood Ratio Test, LR-Test*) vergleicht die Likelihood L_0 des Nullmodels mit der Likelihood L_β des geschätzten Modells. Die Prüfgröße des LR-Tests ergibt sich aus:

$$T = -2 \cdot ln\left(\frac{L_0}{L_\beta}\right)$$

T ist näherungsweise χ^2-verteilt mit k Freiheitsgraden.

In R können Sie den Test mit `lrtest()` aufrufen. Sie benötigen dazu das Paket `lmtest`.

```
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
lrtest(glm2)
```

```
## Likelihood ratio test
##
## Model 1: Aktienkauf ~ Risikobereitschaft + Einkommen + Interesse
## Model 2: Aktienkauf ~ 1
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   4 -339.50
## 2   1 -402.18 -3 125.36  < 2.2e-16 ***
```

[4] Hosmer/Lemeshow, Applied Logistic Regression, 3rd Ed. (2013), S. 164

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

Das Modell `glm2` ist als Ganzes signifikant, der p-Wert ist sehr klein.

Den Likelihood Quotienten Test können Sie auch verwenden, um zwei Modelle miteinander zu vergleichen, z. B., wenn Sie eine weitere Variable hinzugenommen haben und wissen wollen, ob die Verbesserung auch signifikant war.

```
lrtest(glm1, glm2)
```

```
## Likelihood ratio test
##
## Model 1: Aktienkauf ~ Risikobereitschaft
## Model 2: Aktienkauf ~ Risikobereitschaft + Einkommen + Interesse
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   2 -382.93
## 2   4 -339.50  2 86.856  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Ja, die Modelle `glm1` (mit einer erklärenden Variable) und `glm2` unterscheiden sich signifikant voneinander.

Pseudo-R^2

Verschiedene Statistiker haben versucht, aus der Likelihood eine Größe abzuleiten, die dem R^2 der linearen Regression entspricht. Exemplarisch sei hier McFaddens R^2 gezeigt:

$$R^2 = 1 - \frac{ln(L_\beta)}{ln(L_0)}$$

Wie bei bei dem R^2 der linearen Regression liegt der Wertebereich zwischen 0 und 1. Ab einem Wert von 0,4 kann die Modellanpassung als gut eingestuft werden. Wo liegen R^2 der beiden Modelle `glm1` und `glm2`? Sie können es direkt berechnen oder das Paket`BaylorEdPsych` verwenden.

```
# direkte Berechnung
1 - glm1$deviance/glm1$null.deviance
```

```
## [1] 0.04786616
```

```
1 - glm2$deviance/glm2$null.deviance
```

```
## [1] 0.1558465
```

```
# ggf. install.packages("BaylorEdPsych")
library(BaylorEdPsych)
PseudoR2(glm1)
```

```
##         McFadden     Adj.McFadden        Cox.Snell       Nagelkerke
##       0.04786616       0.04040685       0.05351732       0.07834758
## McKelvey.Zavoina           Effron            Count        Adj.Count
##       0.08260722       0.05840899       0.75571429       0.06557377
##              AIC    Corrected.AIC
##     769.86238269     769.87959934
```

```
PseudoR2(glm2)
```

```
##         McFadden     Adj.McFadden        Cox.Snell       Nagelkerke
##       0.15584653       0.14341435       0.16396262       0.24003587
## McKelvey.Zavoina           Effron            Count        Adj.Count
##       0.28278782       0.18453835       0.76142857       0.08743169
##              AIC    Corrected.AIC
##     687.00683459     687.06438855
```

Insgesamt ist die Modellanpassung, auch mit allen Variablen, als schlecht zu bezeichnen. **Hinweis:** Die Funktion `PseudoR2(model)` zeigt verschiedene Pseudo-R^2 Statistiken, die jeweils unter bestimmten Bedingungen vorteilhaft einzusetzen sind. Für weitere Erläuterungen sei auf die Literatur verwiesen.

Übung: Rot- oder Weißwein?

Der Datensatz untersucht den Zusammenhang zwischen der Qualität und physiochemischen Eigenschaften von portugisieschen Rot- und Weißweinen.

P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.

Sie können ihn hier herunterladen. Die Originaldaten finden Sie im UCI Machine Learning Repository.

Versuchen Sie anhand geeigneter Variablen, Rot- und Weißweine zu klassifizieren.[5]

Zusatzaufgabe: Die Originaldaten bestehen aus einem Datensatz für Weißweine und einem für Rotweine. Laden Sie diese, beachten Sie die Fehlermeldung und beheben die damit verbundenen Fehler und fassen beide Datensätze zu einem gemeinsamen Datensatz zusammen, in dem eine zusätzliche Variable `color` aufgenommen wird (Rot = 0, Weiß = 1).

Literatur

- David M. Diez, Christopher D. Barr, Mine Çetinkaya-Rundel (2014): *Introductory Statistics with Randomization and Simulation*, https://www.openintro.org/stat/textbook.php?stat_book=isrs, Kapitel 6.4

[5] Anregungen zu dieser Übung stammen von INTW Statistics

- Nicholas J. Horton, Randall Pruim, Daniel T. Kaplan (2015): Project MOSAIC Little Books *A Student's Guide to R*, https://github.com/ProjectMOSAIC/LittleBooks/raw/master/StudentGuide/MOSAIC-StudentGuide.pdf, Kapitel 8
- Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani (2013): *An Introduction to Statistical Learning – with Applications in R*, http://www-bcf.usc.edu/~gareth/ISL/, Kapitel 4.1-4.3
- Maike Luhmann (2015): *R für Einsteiger*, Kapitel 17.5
- Daniel Wollschläger (2014): *Grundlagen der Datenanalyse mit R*, Kapitel 8.1

Lizenz

Versionshinweise:

- Datum erstellt: 2017-06-30
- R Version: 3.4.0
- `mosaic` Version: 0.14.4

Kapitel 7: Dimensionsreduktion mit PCA und EFA

Dimensionsreduktion

Datensätze in den Sozialwissenschaften haben oft viele Variablen - oder auch Dimensionen - und es ist vorteilhaft, diese auf eine kleinere Anzahl von Variablen (oder Dimensionen) zu reduzieren. Zusammenhänge zwischen Konstrukten können so klarer identifiziert werden.

In dieser Übung betrachten wir zwei gängige Methoden, um die Komplexität von multivarianten, metrischen Daten zu reduzieren, indem wir die Anzahl der Dimensionen in den Daten reduzieren.

- Die *Hauptkomponentenanalyse (PCA)* versucht, unkorrelierte Linearkombinationen zu finden, die die maximale Varianz in den Daten erfassen. Die Blickrichtung ist von den Daten zu den Komponenten.
- Die *Exploratorische Faktorenanalyse (EFA)* versucht, die Varianz auf Basis einer kleinen Anzahl von Dimensionen zu modellieren, während sie gleichzeitig versucht, die Dimensionen in Bezug auf die ursprünglichen Variablen interpretierbar zu machen. Es wird davon ausgegangen, dass die Daten einem Faktoren Modell entsprechen. Die Blickrichtung ist von den Faktoren zu den Daten.

Gründe für die Notwendigkeit der Datenreduktion

- Im technischen Sinne der Dimensionsreduktion können wir statt Variablen-Sets die Faktor-/ Komponentenwerte verwenden (z. B. für Mittelwertvergleiche, Regressionsanalyse und Clusteranalyse).

- Wir können Unsicherheit verringern. Wenn wir glauben, dass ein Konstrukt nicht eindeutig messbar ist, dann kann mit einem Variablen-Set die unsicherheit reduziert werden.
- Wir können den Aufwand bei der Datenerfassung vereinfachen, indem wir uns auf Variablen konzentrieren, von denen bekannt ist, dass sie einen hohen Beitrag zum interessierenden Faktor/ Komponente leisten. Wenn wir feststellen, dass einige Variablen für einen Faktor nicht wichtig sind, können wir sie aus dem Datensatz eliminieren.

Benötigte Pakete

Pakete, die für diese Datenanalyse benötigt werden, müssen vorher einmalig in R installiert werden.

```
# install.packages("corrplot")
# install.packages("gplots")
# install.packages("nFactors")
# install.packages("scatterplot3d")
```

Daten

Wir untersuchen die Dimensionalität mittels eines simulierten Datensatzes der typisch für die Wahrnehmung von Umfragen ist. Die Daten spiegeln Verbraucherbewertungen von Marken in Bezug auf Adjektive wieder, die in Umfragen in folgender Form abgefragt werden:

Auf einer Skala von 1 bis 10 (wobei 1 am wenigsten und 10 am meisten zutrifft)

wie... [ADJECTIV]... ist ... [Marke A]...?

Wir verwenden einen *simulierten* Datensatz aus *Chapman & Feit (2015): R for Marketing Research and Analytics* (http://r-marketing.r-forge.r-project.org). Die Daten umfassen simulierte Bewertungen von 10 Marken ("a" bis "j") mit 9 Adjektiven ("performance", "leader", "latest", "fun" usw.) für n = 100 simulierte Befragte.

Das Einlesen der Daten erfolgt direkt über das Internet.

```
brand.ratings <- read.csv("http://goo.gl/IQl8nc")
```

Wir überprüfen zuerst die Struktur des Datensatzes, die ersten 6 Zeilen und die Zusammenfassung

```
str(brand.ratings)
```

```
## 'data.frame':    1000 obs. of  10 variables:
##  $ perform: int  2 1 2 1 1 2 1 2 2 3 ...
##  $ leader : int  4 1 3 6 1 8 1 1 1 1 ...
##  $ latest : int  8 4 5 10 5 9 5 7 8 9 ...
##  $ fun    : int  8 7 9 8 8 5 7 5 10 8 ...
##  $ serious: int  2 1 2 3 1 3 1 2 1 1 ...
##  $ bargain: int  9 1 9 4 9 8 5 8 7 3 ...
```

```
##  $ value  : int  7 1 5 5 9 7 1 7 7 3 ...
##  $ trendy : int  4 2 1 2 1 1 1 7 5 4 ...
##  $ rebuy  : int  6 2 6 1 1 2 1 1 1 1 ...
##  $ brand  : Factor w/ 10 levels "a","b","c","d",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
head(brand.ratings)
```

```
##   perform leader latest fun serious bargain value trendy rebuy brand
## 1       2      4      8   8       2       9     7      4     6     a
## 2       1      1      4   7       1       1     1      2     2     a
## 3       2      3      5   9       2       9     5      1     6     a
## 4       1      6     10   8       3       4     5      2     1     a
## 5       1      1      5   8       1       9     9      1     1     a
## 6       2      8      9   5       3       8     7      1     2     a
```

```
summary(brand.ratings)
```

```
##     perform          leader          latest           fun       
##  Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000  
##  1st Qu.: 1.000   1st Qu.: 2.000   1st Qu.: 4.000   1st Qu.: 4.000  
##  Median : 4.000   Median : 4.000   Median : 7.000   Median : 6.000  
##  Mean   : 4.488   Mean   : 4.417   Mean   : 6.195   Mean   : 6.068  
##  3rd Qu.: 7.000   3rd Qu.: 6.000   3rd Qu.: 9.000   3rd Qu.: 8.000  
##  Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.000  
##
##     serious          bargain          value           trendy      
##  Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.00  
##  1st Qu.: 2.000   1st Qu.: 2.000   1st Qu.: 2.000   1st Qu.: 3.00  
##  Median : 4.000   Median : 4.000   Median : 4.000   Median : 5.00  
##  Mean   : 4.323   Mean   : 4.259   Mean   : 4.337   Mean   : 5.22  
##  3rd Qu.: 6.000   3rd Qu.: 6.000   3rd Qu.: 6.000   3rd Qu.: 7.00  
##  Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.00  
##
##      rebuy           brand    
##  Min.   : 1.000   a      :100  
##  1st Qu.: 1.000   b      :100  
##  Median : 3.000   c      :100  
##  Mean   : 3.727   d      :100  
##  3rd Qu.: 5.000   e      :100  
##  Max.   :10.000   f      :100  
##                   (Other):400
```

Jeder der 100 simulierten Befragten beurteilt 10 Marken, das ergibt insgesamt 1000 Beobachtungen

(Zeilen) im Datensatz.

Wir sehen in der `summary ()`, dass die Bereiche der Bewertungen für jedes Adjektiv 1-10 sind. In `str()` sehen wir, dass die Bewertungen als numerisch eingelesen wurden, während die Markennamen als Faktoren eingelesen wurden. Die Daten sind somit richtig formatiert.

Neuskalierung der Daten

In vielen Fällen ist es sinnvoll, Rohdaten neu zu skalieren. Dies wird üblicherweise als **Standardisierung**, **Normierung**, oder **Z Scoring/Transformation** bezeichnet. Als Ergebnis ist der Mittelwert aller Variablen über alle Beobachtungen dann 0. Da wir hier gleiche Skalenstufen haben, ist ein Skalieren nicht unbedingt notwendig, wir führen es aber trotzdem durch.

Ein einfacher Weg, alle Variablen im Datensatz auf einmal zu skalieren ist der Befehl `scale()`. Da wir die Rohdaten nie ändern wollen, weisen wir die Rohwerte zuerst einem neuen Dataframe `brand.sc` zu und scalieren anschließend die Daten. Wir skalieren in unserem Datensatz nur die ersten 9 Variablen, weil die 10. Variable der Faktor für die Markenamen ist.

```
brand.sc <- brand.ratings
brand.sc[, 1:9] <- scale(brand.ratings[, 1:9])
summary(brand.sc)
```

```
##     perform            leader            latest             fun          
##  Min.   :-1.0888   Min.   :-1.3100   Min.   :-1.6878   Min.   :-1.84677  
##  1st Qu.:-1.0888   1st Qu.:-0.9266   1st Qu.:-0.7131   1st Qu.:-0.75358  
##  Median :-0.1523   Median :-0.1599   Median : 0.2615   Median :-0.02478  
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000  
##  3rd Qu.: 0.7842   3rd Qu.: 0.6069   3rd Qu.: 0.9113   3rd Qu.: 0.70402  
##  Max.   : 1.7206   Max.   : 2.1404   Max.   : 1.2362   Max.   : 1.43281  
##                                                                          
##     serious           bargain             value             trendy        
##  Min.   :-1.1961   Min.   :-1.22196   Min.   :-1.3912   Min.   :-1.53897  
##  1st Qu.:-0.8362   1st Qu.:-0.84701   1st Qu.:-0.9743   1st Qu.:-0.80960  
##  Median :-0.1163   Median :-0.09711   Median :-0.1405   Median :-0.08023  
##  Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.00000  
##  3rd Qu.: 0.6036   3rd Qu.: 0.65279   3rd Qu.: 0.6933   3rd Qu.: 0.64914  
##  Max.   : 2.0434   Max.   : 2.15258   Max.   : 2.3610   Max.   : 1.74319  
##                                                                           
##      rebuy             brand    
##  Min.   :-1.0717   a      :100  
##  1st Qu.:-1.0717   b      :100  
##  Median :-0.2857   c      :100  
##  Mean   : 0.0000   d      :100
```

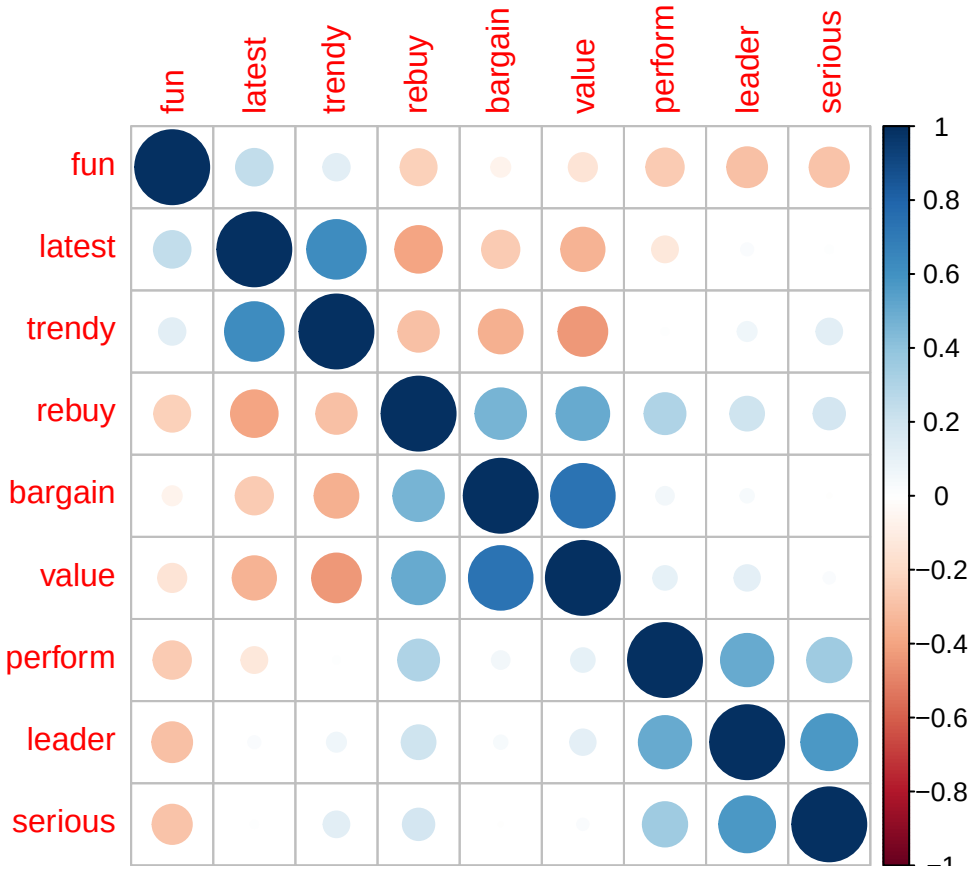
```
##   3rd Qu.: 0.5003   e       :100
##   Max.   : 2.4652   f       :100
##                     (Other):400
```

Die Daten wurden richtig skaliert, da der Mittelwert aller Variablen über alle Beobachtungen 0 ist.

Zusammenhänge in den Daten

Wir verwenden den Befehl `corrplot()` für die Erstinspektion von bivariaten Beziehungen zwischen den Variablen. Das Argument `order = 'hclust'` ordnet die Zeilen und Spalten entsprechend der Ähnlichkeit der Variablen in einer hierarchischen Cluster-Lösung der Variablen (mehr dazu im Teil *Clusteranalyse*) neu an.

```
library(corrplot)
corrplot(cor(brand.sc[, 1:9]), order="hclust")
```



Die Visualisierung der Korrelation der Adjektive scheint drei allgemeine Cluster zu zeigen:

- fun/latest/trendy
- rebuy/bargain/value
- perform/leader/serious

Daten mit fehlende Werten

Wenn in den Daten leere Zellen, also fehlende Werte, vorhanden sind, dann kann es bei bestimmten Rechenoperationen zu Fehlermeldungen kommen. Dies betrifft zum Beispiel Korrelationen, PCA und EFA. Der Ansatz besteht besteht deshalb darin, NA-Werte explizit zu entfernen. Dies kann am einfachsten mit dem Befehl `na.omit()` geschehen:

Beispiel:

```
corrplot(cor(na.omit((brand.sc[, 1:9]), order="hclust"))
```

Da wir in unserem Datensatz simulierte Daten verwenden, gibt es auch keine Leerzellen.

Hinweis: In vielen Funktionen gibt es auch die Option `na.rm = TRUE`, die fehlende Werte entfernt, z. B.:

```
var(brand.sc[, 1:9], na.rm = TRUE)
```

Aggregation der durchschnittlichen Bewertungen nach Marke

Um die Frage "Was ist die durchschnittliche (mittlere) Bewertung der Marke auf jedem Adjektiv?" zu beantworten, können wir den Befehl `aggregate()` verwenden. Dieser berechnet den Mittelwert jeder Variable nach Marke.

```
brand.mean <- aggregate(.~ brand, data=brand.sc, mean)
brand.mean
```

```
##    brand     perform     leader     latest        fun     serious
## 1      a -0.88591874 -0.5279035  0.4109732  0.6566458 -0.91894067
## 2      b  0.93087022  1.0707584  0.7261069 -0.9722147  1.18314061
## 3      c  0.64992347  1.1627677 -0.1023372 -0.8446753  1.22273461
## 4      d -0.67989112 -0.5930767  0.3524948  0.1865719 -0.69217505
## 5      e -0.56439079  0.1928362  0.4564564  0.2958914  0.04211361
## 6      f -0.05868665  0.2695106 -1.2621589 -0.2179102  0.58923066
## 7      g  0.91838369 -0.1675336 -1.2849005 -0.5167168 -0.53379906
## 8      h -0.01498383 -0.2978802  0.5019396  0.7149495 -0.14145855
## 9      i  0.33463879 -0.3208825  0.3557436  0.4124989 -0.14865746
## 10     j -0.62994504 -0.7885965 -0.1543180  0.2849595 -0.60218870
##        bargain       value      trendy       rebuy
## 1   0.21409609  0.18469264 -0.52514473 -0.59616642
## 2   0.04161938  0.15133957  0.74030819  0.23697320
## 3  -0.60704302 -0.44067747  0.02552787 -0.13243776
## 4  -0.88075605 -0.93263529  0.73666135 -0.49398892
## 5   0.55155051  0.41816415  0.13857986  0.03654811
## 6   0.87400696  1.02268859 -0.81324496  1.35699580
```

```
## 7   0.89650392  1.25616009 -1.27639344  1.36092571
## 8  -0.73827529 -0.78254646  0.86430070 -0.60402622
## 9  -0.25459062 -0.80339213  0.59078782 -0.20317603
## 10 -0.09711188 -0.07379367 -0.48138267 -0.96164748
```

Zusätzlich setzten wir die Markennamen als Fallbezeichnung in der Datenmatrix ein.

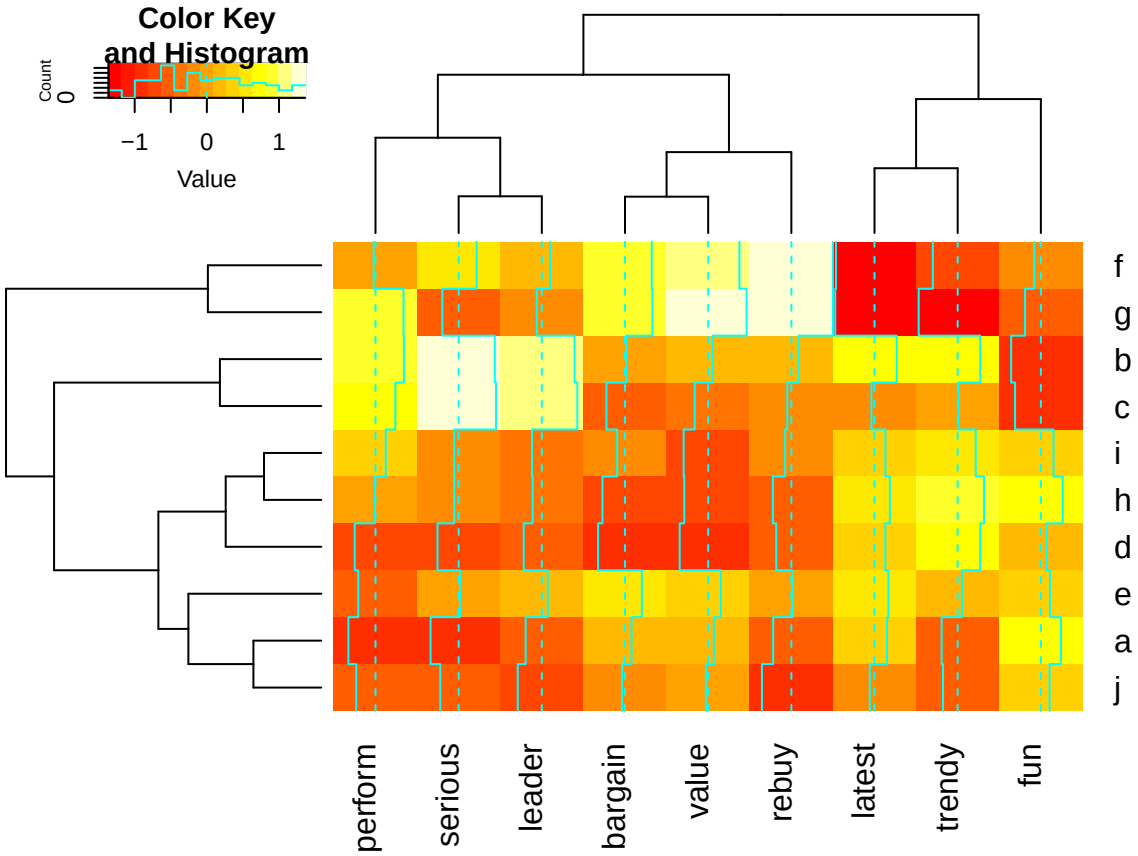
```
rownames(brand.mean) <- brand.mean[, 1] # Markenname als Fallbezeichnung setzen
brand.mean <- brand.mean[, -1]          # Variablenname brand entfernen
brand.mean
```

```
##       perform     leader     latest        fun     serious     bargain
## a -0.88591874 -0.5279035  0.4109732  0.6566458 -0.91894067  0.21409609
## b  0.93087022  1.0707584  0.7261069 -0.9722147  1.18314061  0.04161938
## c  0.64992347  1.1627677 -0.1023372 -0.8446753  1.22273461 -0.60704302
## d -0.67989112 -0.5930767  0.3524948  0.1865719 -0.69217505 -0.88075605
## e -0.56439079  0.1928362  0.4564564  0.2958914  0.04211361  0.55155051
## f -0.05868665  0.2695106 -1.2621589 -0.2179102  0.58923066  0.87400696
## g  0.91838369 -0.1675336 -1.2849005 -0.5167168 -0.53379906  0.89650392
## h -0.01498383 -0.2978802  0.5019396  0.7149495 -0.14145855 -0.73827529
## i  0.33463879 -0.3208825  0.3557436  0.4124989 -0.14865746 -0.25459062
## j -0.62994504 -0.7885965 -0.1543180  0.2849595 -0.60218870 -0.09711188
##         value      trendy       rebuy
## a  0.18469264 -0.52514473 -0.59616642
## b  0.15133957  0.74030819  0.23697320
## c -0.44067747  0.02552787 -0.13243776
## d -0.93263529  0.73666135 -0.49398892
## e  0.41816415  0.13857986  0.03654811
## f  1.02268859 -0.81324496  1.35699580
## g  1.25616009 -1.27639344  1.36092571
## h -0.78254646  0.86430070 -0.60402622
## i -0.80339213  0.59078782 -0.20317603
## j -0.07379367 -0.48138267 -0.96164748
```

Visualisierung der aggregierten Markenbewertungen

Eine **Heatmap** ist eine nützliche Darstellungsmöglichkeit, um solche Ergebnisse zu visualisieren und analysieren, da sie Datenpunkte durch die Intensitäten ihrer Werte färbt. Hierzu laden wir das Paket `gplots`.

```
library(gplots)
heatmap.2(as.matrix(brand.mean))
```



`heatmap.2()` sortiert die Spalten und Zeilen, um Ähnlichkeiten und Muster in den Daten hervorzuheben. Eine zusätzliche Analysehilfe ist das Spalten- und Zeilendendrogramm. Hier werden Beobachtungen die nahe beineinanderliegen in einem Baum abgebildet (näheres hierzu im Abschnitt *Clusteranalyse*.)

Auch hier sehen wir wieder die gleiche Zuordnung der Adjektive nach

- fun/latest/trendy
- rebuy/bargain/value
- perform/leader/serious

Zusätzlich können die Marken nach Ähnlichkeit bezüglich bestimmer Adjektive zugeordnet werden:

- f und g
- b und c
- i, h und d
- a und j

Hauptkomponentenanalyse (PCA)

Die PCA berechnet ein Variablenset (Komponenten) in Form von linearen Gleichungen, die die linearen Beziehungen in den Daten erfassen. Die erste Komponente erfasst so viel Streuung (Varianz) wie möglich von allen Variablen als eine einzige lineare Funktion. Die zweite Komponente erfasst unkorreliert zur ersten Komponente so viel Streuung wie möglich, die nach der ersten Komponente verbleibt. Das geht so lange weiter, bis es so viele Komponenten gibt wie Variablen.

Bestimmung der Anzahl der Hauptkomponenten

Betrachten wir in einem ersten Schritt die wichtigsten Komponenten für die Brand-Rating-Daten. Wir finden die Komponenten mit prcomp(), wobei wir wieder nur die Bewertungsspalten 1-9 auswählen:

```
brand.pc <- prcomp(brand.sc[, 1:9])
summary(brand.pc)
```

```
## Importance of components%s:
##                           PC1    PC2    PC3    PC4     PC5     PC6     PC7
## Standard deviation     1.726 1.4479 1.0389 0.8528 0.79846 0.73133 0.62458
## Proportion of Variance 0.331 0.2329 0.1199 0.0808 0.07084 0.05943 0.04334
## Cumulative Proportion  0.331 0.5640 0.6839 0.7647 0.83554 0.89497 0.93831
##                           PC8     PC9
## Standard deviation     0.55861 0.49310
## Proportion of Variance 0.03467 0.02702
## Cumulative Proportion  0.97298 1.00000
```

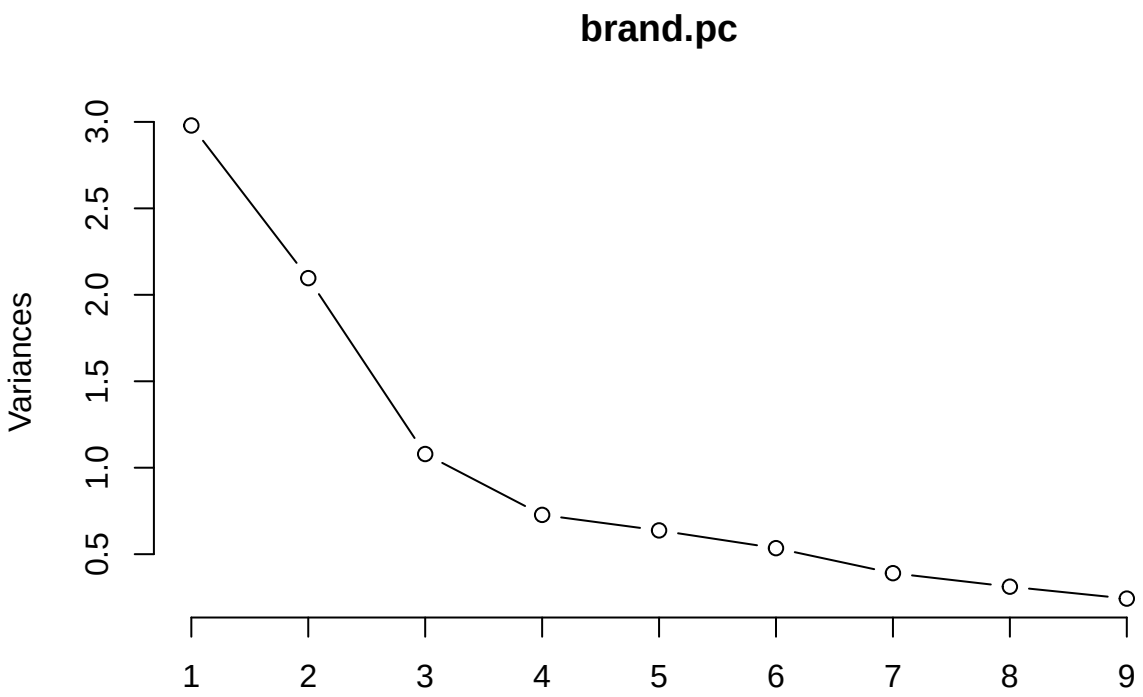
```
# Berechnung der Gesamtvarianz
Gesamtvarianz <- sum(brand.pc$sdev^2)

# Bei sum(brand.pc$sdev^2) wird berechnet:
# 1.726^2 + 1.4479^2 + 1.0389^2 + 0.8528^2 + 0.79846^2 + 0.73133^2 + 0.62458^2 + 0.5586

# Varianzanteil der ersten Hauptkomponente
brand.pc$sdev[1]^2/Gesamtvarianz
```

```
## [1] 0.3310328
```

Scree-Plot

Der Standard-Plot `plot()` für die PCA ist ein **Scree-Plot**, Dieser zeigt uns in Reihenfolge der Hauptkomponenten jeweils die durch diese Hauptkomponente erfasste Streuung (Varianz). Wir plotten ein Liniendiagramm mit dem Argument `type = 'l'` (`l` für Linie):

```
plot(brand.pc, type="l")
```



Wir sehen anhand des Scree-Plots, dass bei den Brand-Rating-Daten der Anteil der Streuung nach der dritten Komponente nicht mehr wesentlich abnimmt. Es soll die Stelle gefunden werden, ab der die Varianzen der Hauptkomponenten deutlich kleiner sind. Je kleiner die Varianzen, desto weniger Streuung erklärt diese Hauptkomponente.

Elbow-Kriterium

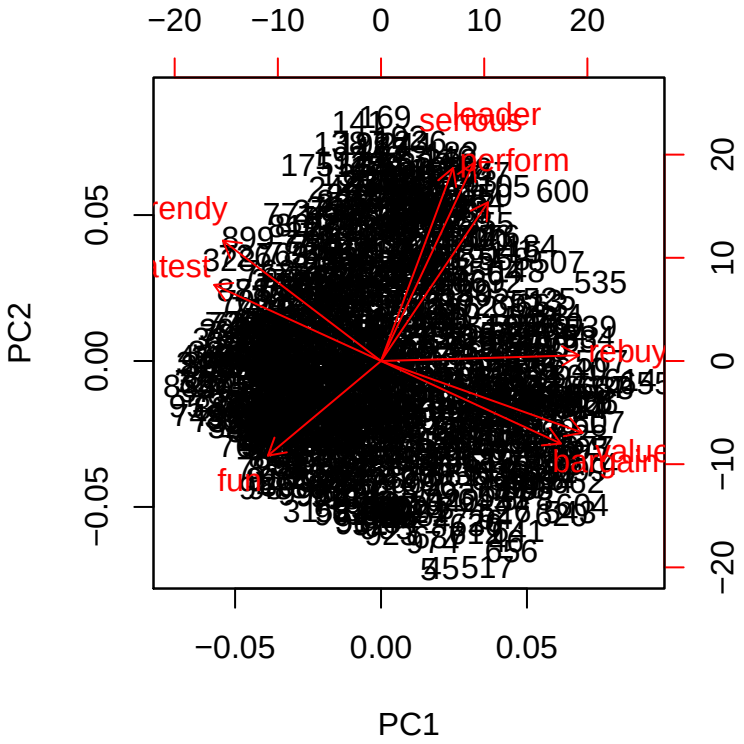
Nach diesem Kriterium werden alle Hauptkomponenten berücksichtigt, die links von der Knickstelle im Scree-Plot liegen. Gibt es mehrere Knicks, dann werden jene Hauptkomponenten ausgewählt, die links vom rechtesten Knick liegen. Gibt es keinen Knick, dann hilft der Scree-Plot nicht weiter. Bei den Brand-Rating-Daten tritt der Ellbogen, je nach Betrachtungsweise, entweder bei drei oder vier Komponenten auf. Dies deutet darauf hin, dass die ersten zwei oder drei Komponenten die meiste Streuung in den Brand-Rating-Daten erklären.

Biplot

Eine gute Möglichkeit die Ergebnisse der PCA zu analysieren, besteht darin, die ersten Komponenten zuzuordnen, die es uns ermöglichen, die Daten in einem niedrigdimensionalen Raum zu visualisieren. Eine gemeinsame Visualisierung ist ein Biplot. Dies ist ein zweidimensionales Diagramm von Datenpunkten in Bezug auf die ersten beiden Hauptkomponenten, die mit einer Projektion der Variablen auf die Komponenten überlagert wird.

Dazu verwenden wir `biplot()`:

```
biplot(brand.pc)
```



Die Adjektiv-Gruppierungen auf den Variablen sind als rote Ladungspfeile sichbar. ZUsätzlich erhalten wir einen Einblick in die Bewertungscluster (als dichte Bereiche von Beobachtungspunkten). Der Biplot ist hier durch die große Anzahl an Beobachtung recht unübersichtlich.

Deshalb führen wir die PCA mit den aggregierten Daten durch:

```
brand.mean
```

```
##       perform     leader     latest        fun     serious     bargain
## a -0.88591874 -0.5279035  0.4109732  0.6566458 -0.91894067  0.21409609
## b  0.93087022  1.0707584  0.7261069 -0.9722147  1.18314061  0.04161938
## c  0.64992347  1.1627677 -0.1023372 -0.8446753  1.22273461 -0.60704302
## d -0.67989112 -0.5930767  0.3524948  0.1865719 -0.69217505 -0.88075605
## e -0.56439079  0.1928362  0.4564564  0.2958914  0.04211361  0.55155051
## f -0.05868665  0.2695106 -1.2621589 -0.2179102  0.58923066  0.87400696
## g  0.91838369 -0.1675336 -1.2849005 -0.5167168 -0.53379906  0.89650392
## h -0.01498383 -0.2978802  0.5019396  0.7149495 -0.14145855 -0.73827529
## i  0.33463879 -0.3208825  0.3557436  0.4124989 -0.14865746 -0.25459062
## j -0.62994504 -0.7885965 -0.1543180  0.2849595 -0.60218870 -0.09711188
##         value      trendy       rebuy
## a  0.18469264 -0.52514473 -0.59616642
## b  0.15133957  0.74030819  0.23697320
## c -0.44067747  0.02552787 -0.13243776
## d -0.93263529  0.73666135 -0.49398892
## e  0.41816415  0.13857986  0.03654811
## f  1.02268859 -0.81324496  1.35699580
```

```
## g  1.25616009 -1.27639344  1.36092571
## h -0.78254646  0.86430070 -0.60402622
## i -0.80339213  0.59078782 -0.20317603
## j -0.07379367 -0.48138267 -0.96164748
```

```
brand.mu.pc<- prcomp(brand.mean, scale=TRUE)
summary(brand.mu.pc)
```

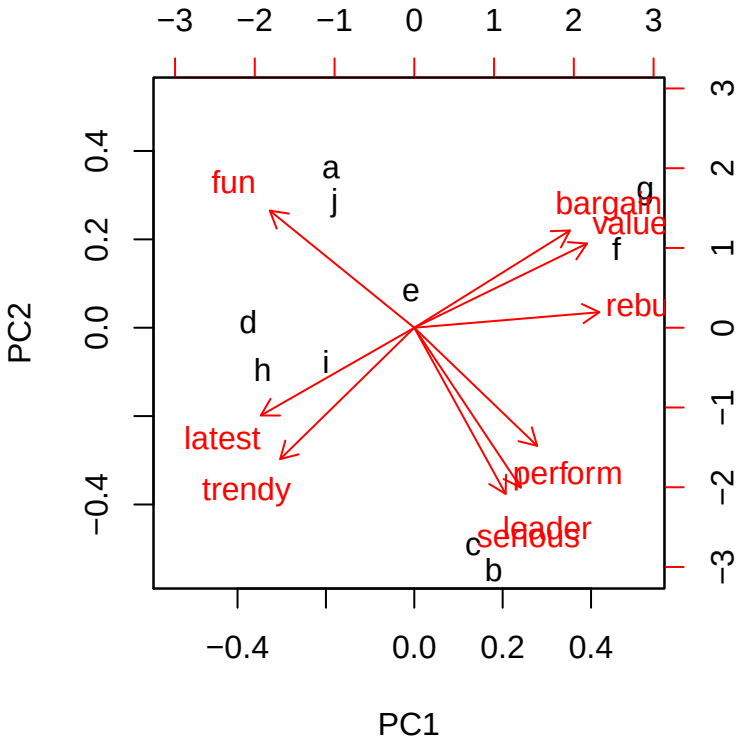
```
## Importance of components%s:
##                           PC1    PC2    PC3     PC4     PC5     PC6
## Standard deviation     2.1345 1.7349 0.7690 0.61498 0.50983 0.36662
## Proportion of Variance 0.5062 0.3345 0.0657 0.04202 0.02888 0.01493
## Cumulative Proportion  0.5062 0.8407 0.9064 0.94842 0.97730 0.99223
##                            PC7     PC8     PC9
## Standard deviation     0.21506 0.14588 0.04867
## Proportion of Variance 0.00514 0.00236 0.00026
## Cumulative Proportion  0.99737 0.99974 1.00000
```

Dem Befehl `prcomp()` wurde `Skalierung = TRUE` hinzugefügt, um die Daten neu zu skalieren. Obwohl die Rohdaten bereits skaliert waren, haben die aggregierten Daten eine etwas andere Skala als die standardisierten Rohdaten. Die Ergebnisse zeigen, dass die ersten beiden Komponenten für 84% der erklärbaren Streuung bei den aggregierten Daten verantwortlich sind.

Wahrnehmungsraum

Wenn ein Biplot Marken in einem zweidimensionalen Raum abbildet, dann wird dieser Raum **zweidimensionaler Wahrnehmungsraum** bezeichnet.

```
biplot(brand.mu.pc)
```



Der Biplot der PCA-Lösung für die Mittelwerte gibt einen interpretierbaren Wahrnehmungsraum, der zeigt, wo die Marken in Bezug auf die ersten beiden Hauptkomponenten liegen. Die Variablen auf den beiden Komponenten sind mit der PCA auf den gesamten Datensatz konsistent. Wir sehen vier Bereiche (Positionen) mit gut differenzierten Adjektiven und Marken.

Exploratorische Faktorenanalyse (EFA)

EFA ist eine Methode, um die Beziehung von Konstrukten (Konzepten), d. h. Faktoren zu Variablen zu beurteilen. Dabei werden die Faktoren als **latente Variablen** betrachtet, die nicht direkt beobachtet werden können. Stattdessen werden sie empirisch durch mehrere Variablen beobachtet, von denen jede ein Indikator der zugrundeliegenden Faktoren ist. Diese beobachteten Werte werden als **manifeste Variablen** bezeichnet und umfassen Indikatoren. Die EFA versucht den Grad zu bestimmen, in dem Faktoren die beobachtete Streuung der manifesten Variablen berücksichtigen.

Das Ergebnis der EFA ist ähnlich zur PCA: eine Matrix von Faktoren (ähnlich zu den PCA-Komponenten) und ihre Beziehung zu den ursprünglichen Variablen (Ladung der Faktoren auf die Variablen). Im Gegensatz zur PCA versucht die EFA, Lösungen zu finden, die in den **manifesten variablen maximal interpretierbar** sind. Im allgemeinen versucht sie, Lösungen zu finden, bei denen eine kleine Anzahl von Ladungen für jeden Faktor sehr hoch ist, während andere Ladungen für diesen Faktor gering sind. Wenn dies möglich ist, kann dieser Faktor mit diesem Variablen-Set interpretiert werden. Innerhalb einer PCA kann die Interpretierbarkeit über eine **Rotation** (z. B. `varimax()`) erhöht werden.

Finden einer EFA Lösung

Als erstes muss die Anzahl der zu schätzenden Faktoren bestimmt werden. Hierzu verwenden wir zwei gebräuchliche Methoden:

1. Das Elbow-Kriterium

Den Scree-plot haben wir bereits bei der PCA durchgeführt. Ein Knick konnten wir bei der dritte oder vierten Hauptkomponente feststellen. Somit zeigt der Skreeplot eine 2 oder 3 Faktorenlösung an.

Durch das Paket `nFactors` bekommen wir eine formalisierte Berechnung der Scree-Plot Lösung mit dem Befehl `nScree()`

```
library(nFactors)
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
## Loading required package: psych
```

```
##
## Attaching package: 'psych'
```

```
## The following objects are masked from 'package:mosaic':
##
##     logit, read.file, rescale
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
## Loading required package: boot
```

```
##
## Attaching package: 'boot'
```

```
## The following object is masked from 'package:psych':
##
##     logit
```

```
## The following object is masked from 'package:mosaic':
##
##     logit
```

```
## The following object is masked from 'package:lattice':
```

```
##
##     melanoma
##
## Attaching package: 'nFactors'
## The following object is masked from 'package:lattice':
##
##     parallel
```

```
nScree(brand.sc[, 1:9])
```

```
##   noc naf nparallel nkaiser
## 1   3   2         3       3
```

`nScree` gibt vier methodische Schätzungen für die Anzahl an Faktoren durch den Scree-Plot aus. Wir sehen, dass drei von vier Methoden drei Faktoren vorschlagen.

2. Das Eigenwert-Kriterium

Der Eigenwert ist eine Metrik für den Anteil der erklärten Varianz. Die Anzahl Eigenwerte können wir über den Befehl `eigen()` ausgeben.

```
eigen(cor(brand.sc[, 1:9]))
```

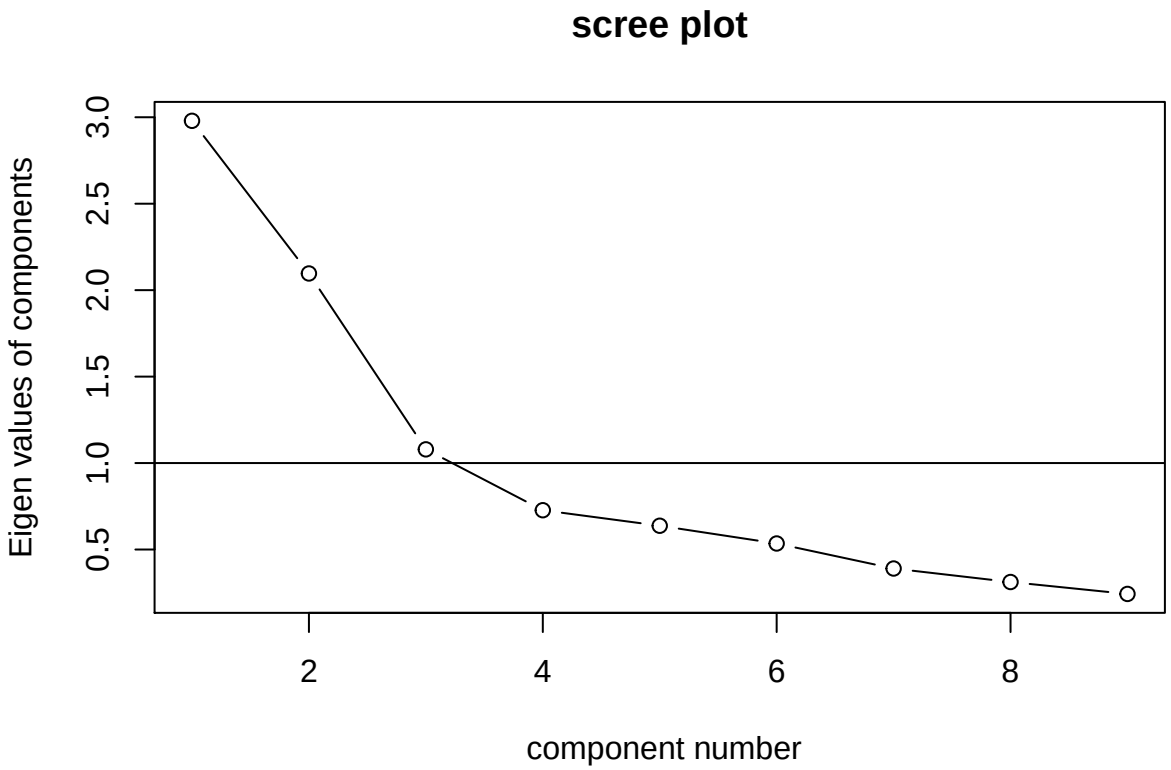
```
## eigen() decomposition
## $values
## [1] 2.9792956 2.0965517 1.0792549 0.7272110 0.6375459 0.5348432 0.3901044
## [8] 0.3120464 0.2431469
##
## $vectors
##             [,1]        [,2]        [,3]        [,4]        [,5]
##  [1,] -0.2374679 -0.41991179  0.03854006  0.52630873  0.46793435
##  [2,] -0.2058257 -0.52381901 -0.09512739  0.08923461 -0.29452974
##  [3,]  0.3703806 -0.20145317 -0.53273054 -0.21410754  0.10586676
##  [4,]  0.2510601  0.25037973 -0.41781346  0.75063952 -0.33149429
##  [5,] -0.1597402 -0.51047254 -0.04067075 -0.09893394 -0.55515540
##  [6,] -0.3991731  0.21849698 -0.48989756 -0.16734345 -0.01257429
##  [7,] -0.4474562  0.18980822 -0.36924507 -0.15118500 -0.06327757
##  [8,]  0.3510292 -0.31849032 -0.37090530 -0.16764432  0.36649697
##  [9,] -0.4390184 -0.01509832 -0.12461593  0.13031231  0.35568769
##             [,6]         [,7]        [,8]        [,9]
##  [1,]  0.3370676  0.364179109 -0.14444718 -0.05223384
##  [2,]  0.2968860 -0.613674301  0.28766118  0.17889453
##  [3,]  0.1742059 -0.185480310 -0.64290436 -0.05750244
##  [4,] -0.1405367 -0.007114761  0.07461259 -0.03153306
```

```
##  [5,] -0.3924874  0.445302862 -0.18354764 -0.09072231
##  [6,]  0.1393966  0.288264900  0.05789194  0.64720849
##  [7,]  0.2195327  0.017163011  0.14829295 -0.72806108
##  [8,] -0.2658186  0.153572108  0.61450289 -0.05907022
##  [9,] -0.6751400 -0.388656160 -0.20210688  0.01720236
```

Der Eigenwert eines Faktors sagt aus, wie viel Varianz dieser Faktor an der Gesamtvarianz aufklärt. Laut dem Eigenwert-Kriterium sollen nur Faktoren mit einem Eigenwert größer 1 extrahiert werden. Dies sind bei den Brand-Rating Daten drei Faktoren, da drei Eigenwerte größer 1 sind.

Dies kann auch grafisch mit dem `VSS.Scree` geplotet werden.

```
VSS.scree(brand.sc[, 1:9])
```



Schätzung der EFA

Eine EFA wird geschätzt mit dem Befehl `factanal(x,factors=k)`, wobei `k` die Anzahl Faktoren angibt.

```
brand.fa<-factanal(brand.sc[, 1:9], factors=3)
brand.fa
```

```
##
## Call:
## factanal(x = brand.sc[, 1:9], factors = 3)
##
## Uniquenesses:
## perform  leader  latest     fun serious bargain   value  trendy   rebuy
```

```
##   0.624   0.327   0.005   0.794   0.530   0.302   0.202   0.524   0.575
##
## Loadings:
##         Factor1 Factor2 Factor3
## perform          0.607
## leader           0.810   0.106
## latest  -0.163           0.981
## fun             -0.398   0.205
## serious          0.682
## bargain  0.826          -0.122
## value    0.867          -0.198
## trendy  -0.356           0.586
## rebuy    0.499   0.296  -0.298
##
##                Factor1 Factor2 Factor3
## SS loadings      1.853   1.752   1.510
## Proportion Var   0.206   0.195   0.168
## Cumulative Var   0.206   0.401   0.568
##
## Test of the hypothesis that 3 factors are sufficient.
## The chi square statistic is 64.57 on 12 degrees of freedom.
## The p-value is 3.28e-09
```

Eine übersichtlichere Ausgabe bekommen wir mit dem `print` Befehl, in dem wir zusätzlich noch die Dezimalstellen kürzen mit `digits=2`, alle Ladungen kleiner als 0,5 ausblenden mit `cutoff=.5` und die Ladungen mit `sort=TRUE` so sortieren, dass die Ladungen, die auf einen Faktor laden, untereinander stehen.

```
print(brand.fa, digits=2, cutoff=.5, sort=TRUE)
```

```
##
## Call:
## factanal(x = brand.sc[, 1:9], factors = 3)
##
## Uniquenesses:
## perform  leader  latest     fun serious bargain   value  trendy   rebuy
##    0.62    0.33    0.00    0.79    0.53    0.30    0.20    0.52    0.58
##
## Loadings:
##         Factor1 Factor2 Factor3
## bargain  0.83
## value    0.87
```

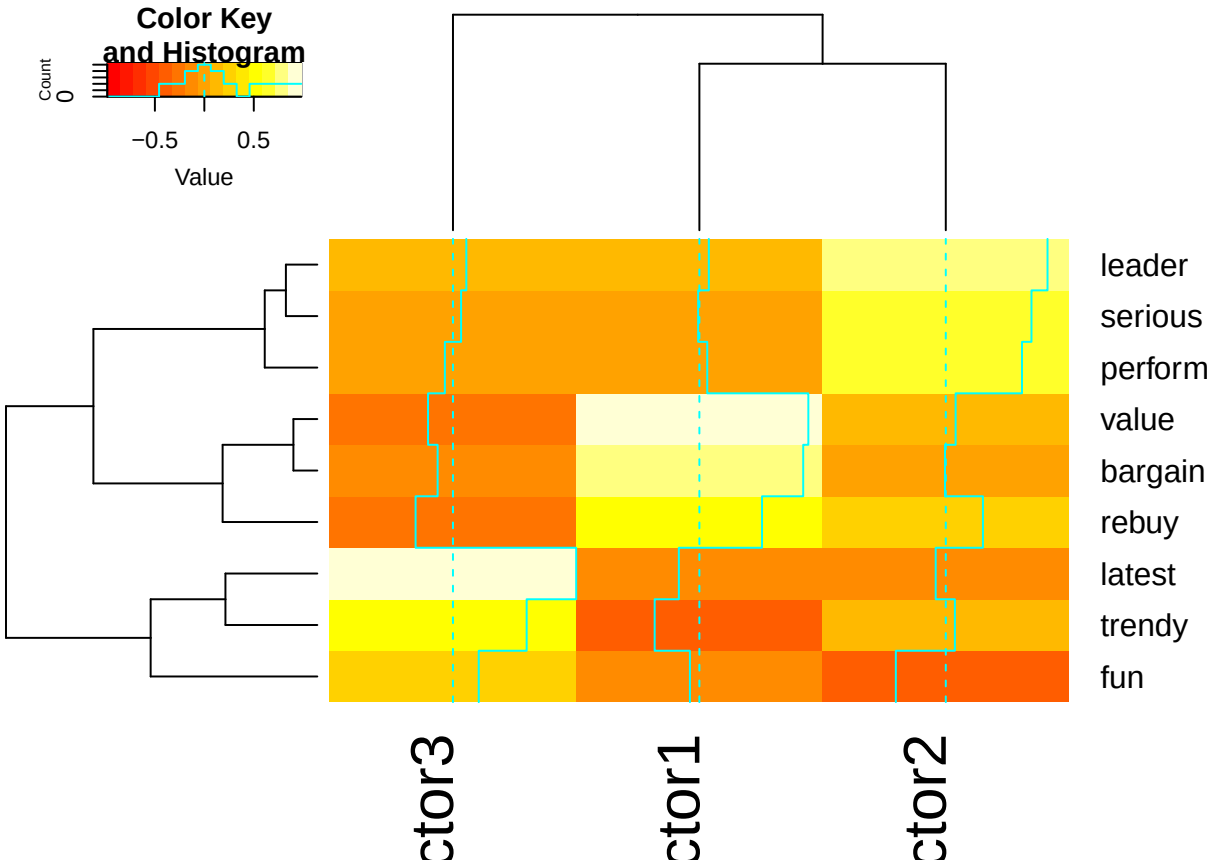
```
## perform          0.61
## leader           0.81
## serious          0.68
## latest                   0.98
## trendy                   0.59
## fun
## rebuy
##
##                Factor1 Factor2 Factor3
## SS loadings       1.85    1.75    1.51
## Proportion Var    0.21    0.19    0.17
## Cumulative Var    0.21    0.40    0.57
##
## Test of the hypothesis that 3 factors are sufficient.
## The chi square statistic is 64.57 on 12 degrees of freedom.
## The p-value is 3.28e-09
```

Standardmäßig wird bei `factanal()` eine Varimax-Rotation durchgeführt (das Koordinatensystem der Faktoren wird so rotiert, das eine optimale Zuordnung zu den Variablen erfolgt). Bei Varimax gibt es keine Korrelationen zwischen den Faktoren. Sollen Korrelationen zwischen den Faktoren zugelassen werden, empfielt sich die Oblimin-Rotation mit dem Argument `rotation='oblimin'` aus dem Paket `GPArotation`.

Heatmap mit Ladungen

In der obigen Ausgabe werden die Item-to-Faktor-Ladungen angezeigt. Im zurückgegebenen Objekt `brand.fa` sind diese als `$loadings` vorhanden. Wir können die Item-Faktor-Beziehungen mit einer Heatmap von `$loadings` visualisieren:

```
heatmap.2(brand.fa$loadings)
```



Das Ergebnis aus der Heatmap zeigt eine deutliche Trennung der Items in 3 Faktoren, die grob interpretierbar sind als **value**, **leader** und **latest**.

Berechnung der Faktor-Scores

Zusätzlich zur Schätzung der Faktorstruktur kann die EFA auch die latenten Faktorwerte für jede Beobachtung schätzen. Die gängige Extraktionsmethode ist die Bartlett-Methode.

```
brand.fa.ob <- factanal(brand.sc[, 1:9], factors=3, scores="Bartlett")
brand.scores <- data.frame(brand.fa.ob$scores)
head(brand.scores)
```

```
##      Factor1    Factor2    Factor3
## 1  1.9097351 -0.8668153  0.8448345
## 2 -1.5787358 -1.5067438 -1.1191067
## 3  1.1724921 -1.1298348 -0.2958264
## 4  0.4960526 -0.4295001  1.3020740
## 5  2.1137739 -2.0471144 -0.2104525
## 6  1.6906002  0.1545123  1.2186914
```

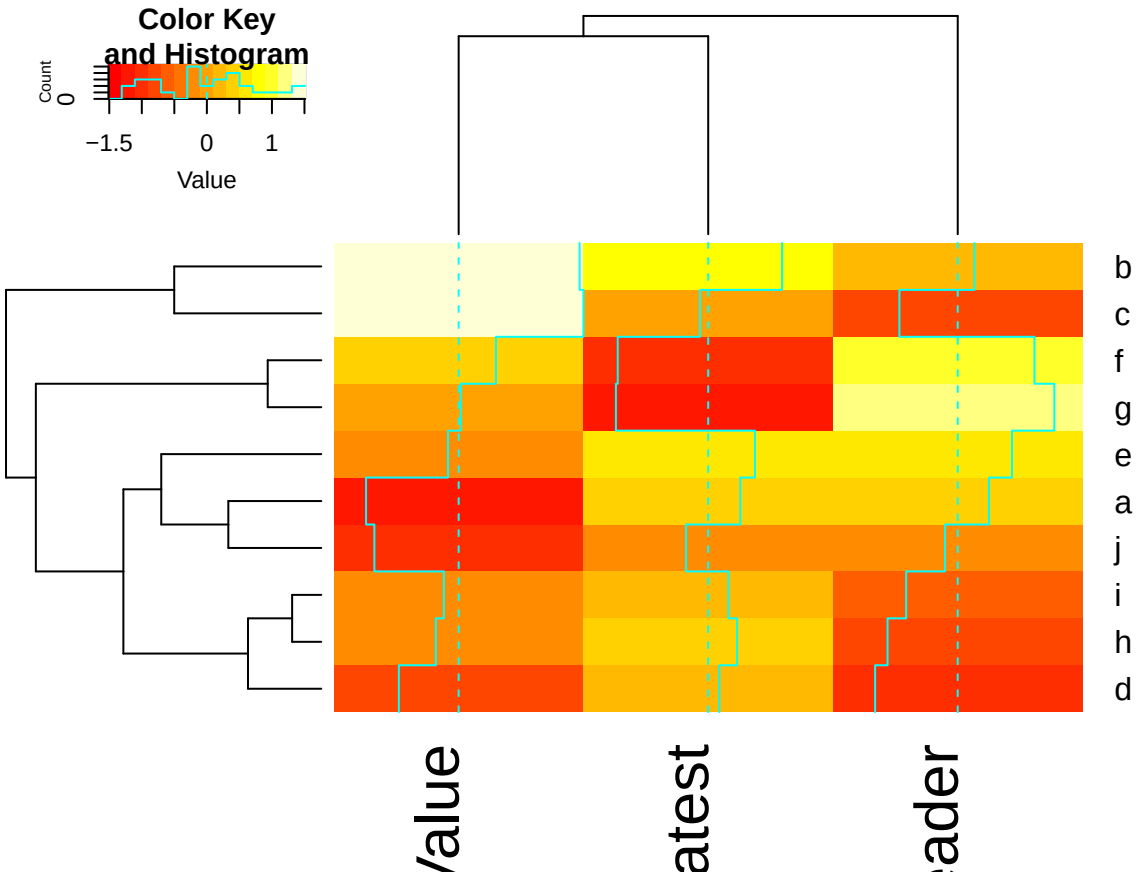
Wir können damit die Faktor-Scores verwenden, um die Positionen der Marken auf den Faktoren zu bestimmen.

```
brand.scores$brand <- brand.sc$brand # Zuweisung der Markennamen zur Scores-Matrix
brand.fa.mean <- aggregate(. ~ brand, data=brand.scores, mean) # Aggregation Marken
rownames(brand.fa.mean) <- brand.fa.mean[, 1] # Fallbezeichnung mit Markennamen setzen
brand.fa.mean <- brand.fa.mean[, -1] # Erste Spalte löschen
names(brand.fa.mean) <- c("Leader", "Value", "Latest") # Spaltennamen neu zuweisen
brand.fa.mean
```

```
##        Leader       Value      Latest
## a  0.3778894 -1.12059561  0.38878416
## b  0.2021913  1.46306266  0.89377982
## c -0.7056202  1.50959052 -0.09947966
## d -0.9997980 -0.72387907  0.13460869
## e  0.6564191 -0.12905899  0.56566118
## f  0.9285971  0.45161454 -1.09464854
## g  1.1688809  0.02452425 -1.11657333
## h -0.8494187 -0.27625354  0.34995876
## i -0.6240513 -0.17881277  0.24499689
## j -0.1550895 -1.02019199 -0.26708797
```

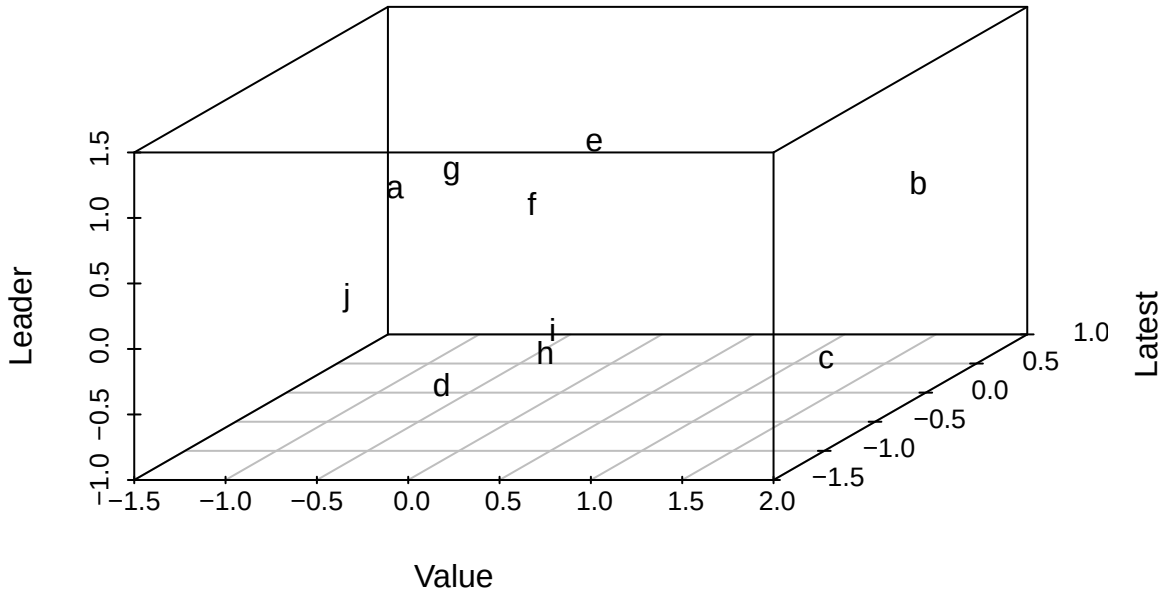
Mittels Heatmap kann dann sehr schnell analysiert werden, welche Marke auf welcher Dimension gute oder schlechte Ausprägungen hat.

```
heatmap.2(as.matrix(brand.fa.mean))
```



Drei Dimensionen lassen sich in einem dreidimensionalen Raum darstellen:

```
library(scatterplot3d)
attach(brand.fa.mean) # Datensatz zum Suchpfad hinzufügen
scatterplot3d(Leader~Value+Latest, pch=row.names(brand.fa.mean))
```




```
detach(brand.fa.mean) # Datensatz vom Suchpfad entfernen
```

Interne Konsistenz der Skalen

Das einfachste Maß für die **interne Konsistenz** ist die **Split-Half-Reliabilität**. Die Items werden in zwei Hälften unterteilt und die resultierenden Scores sollten in ihren Kenngrößen ähnlich sein. Hohe Korrelationen zwischen den Hälften deuten auf eine hohe interne Konsistenz hin. Das Problem ist, dass die Ergebnisse davon abhängen, wie die Items aufgeteilt werden. Ein üblicher Ansatz zur Lösung dieses Problems besteht darin, den Koeffizienten **Alpha (Cronbachs Alpha)** zu verwenden.

Der Koeffizient **Alpha** ist der Mittelwert aller möglichen Split-Half-Koeffizienten, die sich aus verschiedenen Arten der Aufteilung der Items ergeben. Dieser Koeffizient variiert von 0 bis 1. Formal ist es ein korrigierter durschnittlicher Korrelationskoeffizient.

Faustreglen für die Bewertung von Cronbachs Alpha:

Alpha	Bedeutung
größer 0,9	excellent
größer 0,8	gut
größer 0,7	akzeptabel
größer 0,6	fragwürdig
größer 0,5	schlecht

Wir bewerten nun die interne Konsistent der Itmes für die Konstrukte `Leader`, `Value` und `Latest`.

```
alpha(brand.sc[, c("leader","serious","perform")], check.keys=TRUE)
```

```
##
## Reliability analysis
## Call: alpha(x = brand.sc[, c("leader", "serious", "perform")], check.keys = TRUE)
##
##   raw_alpha std.alpha G6(smc) average_r S/N   ase     mean   sd
##       0.73      0.73    0.66      0.48 2.7 0.015 -7.5e-17 0.81
##
##  lower alpha upper     95% confidence boundaries
## 0.7 0.73 0.76
##
##  Reliability if an item is dropped:
##         raw_alpha std.alpha G6(smc) average_r S/N alpha se
## leader       0.53      0.53    0.36      0.36 1.1    0.030
## serious      0.67      0.67    0.50      0.50 2.0    0.021
## perform      0.73      0.73    0.57      0.57 2.7    0.017
##
##  Item statistics
##            n raw.r std.r r.cor r.drop     mean sd
## leader  1000  0.86  0.86  0.76   0.65  7.0e-17  1
## serious 1000  0.80  0.80  0.64   0.54 -1.5e-16  1
## perform 1000  0.77  0.77  0.57   0.48 -1.6e-16  1
```

```
alpha(brand.sc[, c("value", "bargain", "rebuy")], check.keys=TRUE)
```

```
##
## Reliability analysis
## Call: alpha(x = brand.sc[, c("value", "bargain", "rebuy")], check.keys = TRUE)
##
##   raw_alpha std.alpha G6(smc) average_r S/N   ase    mean   sd
##       0.8       0.8    0.75      0.57   4 0.011 9.7e-18 0.84
##
##  lower alpha upper     95% confidence boundaries
## 0.78 0.8 0.82
##
##  Reliability if an item is dropped:
##         raw_alpha std.alpha G6(smc) average_r S/N alpha se
## value        0.64      0.64    0.47      0.47 1.8   0.0230
## bargain      0.67      0.67    0.51      0.51 2.0   0.0207
## rebuy        0.85      0.85    0.74      0.74 5.7   0.0095
```

```
##
##  Item statistics
##            n raw.r std.r r.cor r.drop     mean sd
## value   1000  0.89  0.89  0.83   0.73  1.2e-16  1
## bargain 1000  0.87  0.87  0.80   0.70 -1.2e-16  1
## rebuy   1000  0.78  0.78  0.57   0.52  5.2e-17  1
```

```
alpha(brand.sc[, c("latest","trendy","fun")], check.keys=TRUE)
```

```
##
## Reliability analysis
## Call: alpha(x = brand.sc[, c("latest", "trendy", "fun")], check.keys = TRUE)
##
##   raw_alpha std.alpha G6(smc) average_r S/N   ase    mean   sd
##         0.6       0.6    0.58      0.33 1.5 0.022 4.4e-17 0.75
##
##  lower alpha upper     95% confidence boundaries
## 0.56 0.6 0.64
##
##  Reliability if an item is dropped:
##         raw_alpha std.alpha G6(smc) average_r  S/N alpha se
## latest       0.23      0.23    0.13      0.13 0.29    0.049
## trendy       0.39      0.39    0.25      0.25 0.65    0.038
## fun          0.77      0.77    0.63      0.63 3.37    0.014
##
##  Item statistics
##            n raw.r std.r r.cor r.drop     mean sd
## latest 1000  0.84  0.84  0.76   0.58 -9.7e-17  1
## trendy 1000  0.79  0.79  0.68   0.48  8.8e-17  1
## fun    1000  0.61  0.61  0.26   0.21  1.5e-16  1
```

Bis auf `Latest` sind alle Konstrukte bezüglich ihrer internen Konsistenz akzeptabel. Bei dem Konstrukt `Latest` können wir durch Elimination von `fun` das Cronbachs Alpha von einem fragwürdigen Wert auf einen akteptablen Wert von 0,77 erhöhen.

Das Argument `check.keys=TRUE` gibt uns eine Warung aus, sollte die Ladung eines oder mehrerer Items negativ sein. Dies ist hier nicht der Fall, somit müssen auch keine Items recodiert werden.

Übung: Unskalierte Daten

Führen Sie eine Dimensionsreduktion mit den nichtskalierten original Daten durch. Berechenn Sie zur Interpretaion keine Faktor-Scores, sondern berechnen Sie stattdessen den Mittelwert der Variablen, die

hoch (mindestens 0,5) auf einen Faktor laden. Für die Berechnung verwenden Sie

```
Datensatz$Neue_Variable <- apply(Datensatz[,c("Variable1","Variable2", "etc..")],
                                 1, mean, na.rm=TRUE)
```

Literatur

- Chris Chapman, Elea McDonnell Feit (2015): *R for Marketing Research and Analytics*, Kapitel 8.1-8.3
- Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani (2013): *An Introduction to Statistical Learning – with Applications in R*, http://www-bcf.usc.edu/~gareth/ISL/, Kapitel 10.2, 10.4
- Reinhold Hatzinger, Kurt Hornik, Herbert Nagel (2011): *R – Einführung durch angewandte Statistik.* Kapitel 11
- Maike Luhmann (2015): R für Einsteiger, Kapitel 19

Versionshinweise:

- Datum erstellt: 2017-06-30
- R Version: 3.4.0

Kapitel 8: Einführung Clusteranalyse

Clusteranalyse

Wir werden einen *simulierten* Datensatz aus *Chapman & Feit (2015): R for Marketing Research and Analytics. Springer* analysieren (http://r-marketing.r-forge.r-project.org). Näheres dazu siehe Kapitel 5 dort.

Sie können ihn von hier als `csv`-Datei herunterladen:

```
download.file("https://goo.gl/eUm8PI", destfile = "segment.csv")
```

Das Einlesen erfolgt, sofern die Daten im Arbeitsverzeichnis liegen, wieder über:

```
segment <- read.csv2("segment.csv")
```

Ein Überblick über die Daten:

```
str(segment)
```

```
## 'data.frame':    300 obs. of  7 variables:
##  $ Alter         : num  50.2 40.7 43 40.3 41.1 ...
##  $ Geschlecht    : Factor w/ 2 levels "Frau","Mann": 2 2 1 2 1 2 1 2 1 1 ...
##  $ Einkommen     : num  51356 64411 71615 42728 71641 ...
##  $ Kinder        : int  0 3 2 1 4 2 5 1 1 0 ...
##  $ Eigenheim     : Factor w/ 2 levels "Ja","Nein": 2 2 1 2 2 1 2 2 2 2 ...
##  $ Mitgliedschaft: Factor w/ 2 levels "Ja","Nein": 2 2 2 2 2 2 1 1 2 2 ...
##  $ Segment       : Factor w/ 4 levels "Aufsteiger","Gemischte Vorstadt",..: 2 2 2 2 2
```

```
head(segment)
```

```
##      Alter Geschlecht Einkommen Kinder Eigenheim Mitgliedschaft
## 1 50.16825       Mann  51355.75      0      Nein           Nein
## 2 40.67330       Mann  64411.10      3      Nein           Nein
## 3 42.98939       Frau  71614.94      2        Ja           Nein
## 4 40.31963       Mann  42727.96      1      Nein           Nein
## 5 41.08368       Frau  71640.62      4      Nein           Nein
## 6 40.17045       Mann  60325.41      2        Ja           Nein
##              Segment
## 1 Gemischte Vorstadt
## 2 Gemischte Vorstadt
## 3 Gemischte Vorstadt
## 4 Gemischte Vorstadt
## 5 Gemischte Vorstadt
## 6 Gemischte Vorstadt
```

Zur Unterstützung der Analyse wird (wieder) `mosaic` verwendet:

```
library(mosaic)
```

Das Ziel einer Clusteranalyse ist es, Gruppen von Beobachtungen (d. h. *Cluster*) zu finden, die innerhalb der Cluster möglichst homogen, zwischen den Clustern möglichst heterogen sind. Um die Ähnlichkeit von Beobachtungen zu bestimmen, können verschiedene Distanzmaße herangezogen werden. Für metrische Merkmale wird z. B. häufig die euklidische Metrik verwendet, d. h., Ähnlichkeit und Distanz werden auf Basis des euklidischen Abstands bestimmt. Aber auch andere Abstände wie Manhatten oder Gower sind möglich. Letztere haben den Vorteil, dass sie nicht nur für metrische Daten sondern auch für gemischte Variablentypen verwendet werden können.

Auf Basis der drei metrischen Merkmale (d. h. `Alter`, `Einkommen` und `Kinder`) ergeben sich für die ersten sechs Beobachtungen folgende Abstände:

```
dist(head(segment))
```

```
## Warning in dist(head(segment)): NAs durch Umwandlung erzeugt
```

```
##             1           2           3           4           5
## 2 19942.38233
## 3 30946.42599 11004.04804
## 4 13179.17559 33121.54360 44125.59103
## 5 30985.65446 11043.27434    39.45317 44164.81792
## 6 13701.39082  6240.99480 17245.04247 26880.54893 17284.26910
```

Sie können erkennen, dass die Beobachtungen `5` und `3` den kleinsten Abstand haben, während `5` und `4` den größten haben. Allerdings zeigen die Rohdaten auch, dass die euklidischen Abstände von der Skalierung der Variablen abhängen (`Einkommen` streut stärker als `Kinder`). Daher kann es evt. sinnvoll sein, die Variablen vor der Analyse zu standardisieren (z. B. über `scale()`). Die Funktion `daisy()` aus dem Paket `cluster` bietet hier nützliche Möglichkeiten.

```
library(cluster)

daisy(head(segment))
```

```
## Dissimilarities :
##           1         2         3         4         5
## 2 0.3073211
## 3 0.5598210 0.3901170
## 4 0.2190697 0.1836184 0.5023068
## 5 0.5157498 0.2201562 0.2416431 0.4037746
## 6 0.4014618 0.2059440 0.2389181 0.2676518 0.4261002
##
## Metric :  mixed ;  Types = I, N, I, I, N, N, N
## Number of objects : 6
```

Hierarchische Clusteranalyse

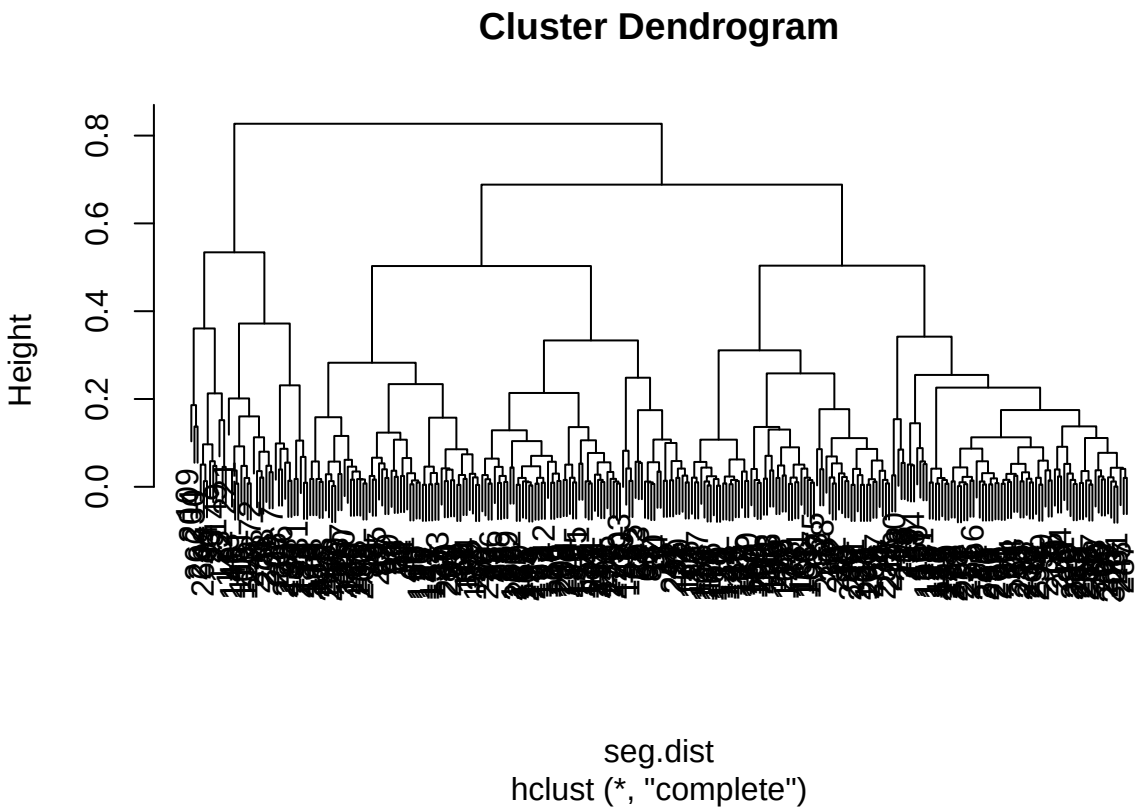
Bei hierarchischen Clusterverfahren werden Beobachtungen sukzessiv zusammengefasst (agglomerativ). Zunächst ist jede Beobachtung ein eigener Cluster, die dann je nach Ähnlichkeitsmaß zusammengefasst werden.

Fassen wir die Beobachtungen *ohne* die Segmentvariable `Segment`, Variable 7, zusammen:

```
seg.dist <- daisy(segment[,-7]) # Abstände
seg.hc <- hclust(seg.dist) # Hierarchische Clusterung
```

Das Ergebnis lässt sich schön im Dendrogramm darstellen:

```
plot(seg.hc)
```

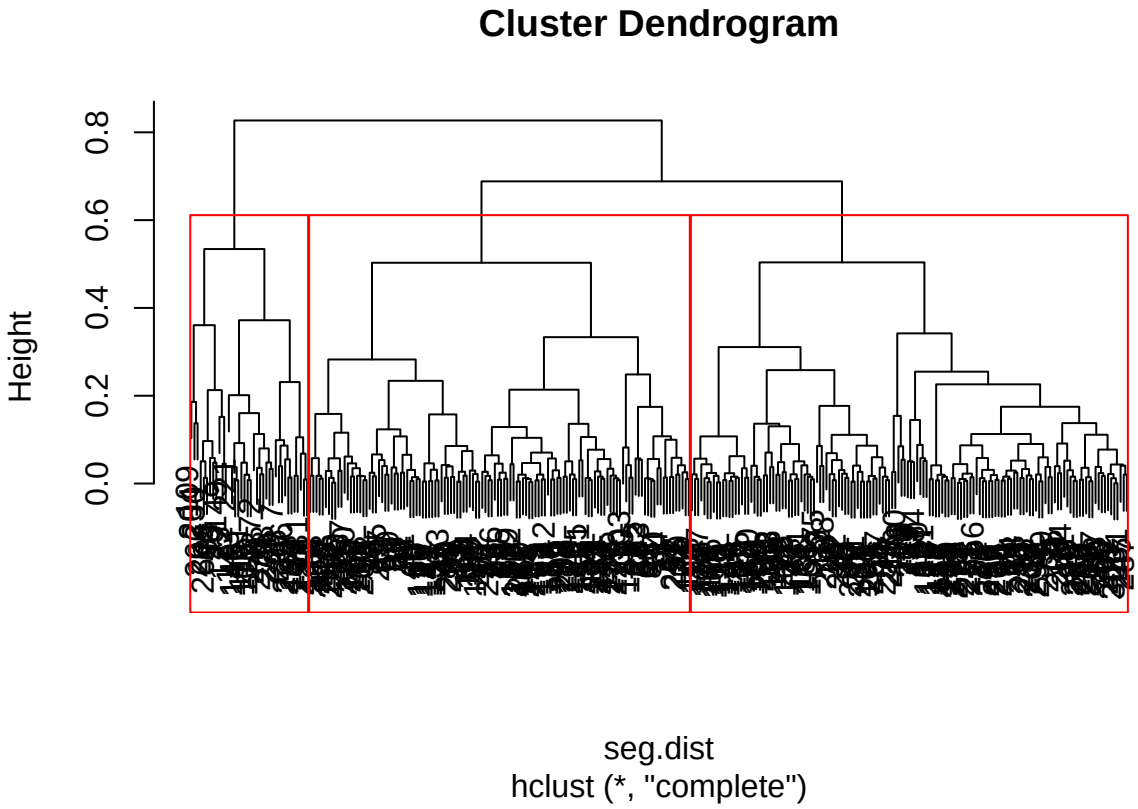


Je höher (`Height`) die Stelle ist, an der zwei Beobachtungen oder Cluster zusammengefasst werden, desto größer ist die Distanz. D. h., Beobachtungen bzw. Cluster, die unten zusammengefasst werden, sind sich ähnlich, die, die oben zusammengefasst werden unähnlich.

Hier wurde übrigens die Standardeinstellung für die Berechnung des Abstands von Clustern verwendet: Complete Linkage bedeutet, dass die Distanz zwischen zwei Clustern auf Basis des maximalen Abstands der Beobachtungen innerhalb des Clusters gebildet wird.

Es ist nicht immer einfach zu entscheiden, wie viele Cluster es gibt. In der Praxis und Literatur finden sich häufig Zahlen zwischen 3 und 10. Evt. gibt es im Dendrogramm eine Stelle, an der der Baum gut geteilt werden kann. In unserem Fall vielleicht bei einer Höhe von 0.6, da sich dann 3 Cluster ergeben:

```
plot(seg.hc)
rect.hclust(seg.hc, h=0.6, border="red")
```



Das Ergebnis, d. h. die Clusterzuordnung, kann durch den Befehl `cutree()` den Beobachtungen zugeordnet werden.

```
segment$hc.clust <- cutree(seg.hc, k=3)
```

Z. B. haben wir folgende Anzahlen für Beobachtungen je Cluster:

```
tally(~hc.clust, data=segment)
```

```
## hc.clust
##   1   2   3
## 140 122  38
```

Cluster 3 ist also mit Abstand der kleinste Cluster (mit 38 Beobachtungen).

Für den Mittelwert des Alters je Cluster gilt:

```
mean(Alter~hc.clust, data=segment)
```

```
##        1        2        3
## 38.53910 46.42365 34.48440
```

D. h., das Durchschnittsalter ist in Cluster 3 ca. 4 Jahre jünger als in 1 – und 12 Jahre jünger als in 2.

Das spiegelt sich auch im Einkommen wieder:

```
mean(Einkommen~hc.clust, data=segment)
```

```
##        1        2        3
## 49452.33 54355.46 44113.01
```

Allerdings sind die Unterschiede in der Geschlechtsverteilung eher gering:

```
tally(Geschlecht~hc.clust, data=segment, format="proportion")
```

```
##           hc.clust
## Geschlecht         1         2         3
##       Frau 0.5428571 0.5491803 0.5263158
##       Mann 0.4571429 0.4508197 0.4736842
```

k-Means Clusteranalyse

Beim k-Means Clusterverfahren handelt es sich im Gegensatz zur hierarchischen Clusteranalyse um ein partitionierendes Verfahren. Die Daten werde in k Cluster aufgeteilt – dabei muss die Anzahl der Cluster im vorhinein feststehen. Ziel ist es, dass die Quadratsumme der Abweichungen der Beobachtungen im Cluster zum Clusterzentrum minimiert wird.

Der Ablauf des Verfahrens ist wie folgt:

1. Zufällige Beobachtungen als Clusterzentrum
2. Zuordnung der Beobachtungen zum nächsten Clusterzentrum (Ähnlichkeit, z. B. über die euklidische Distanz)
3. Neuberechnung der Clusterzentren als Mittelwert der dem Cluster zugeordneten Beobachtungen

Dabei werden die Schritte 2. und 3. solange wiederholt, bis sich keine Änderung der Zuordnung mehr ergibt – oder eine maximale Anzahl an Iterationen erreicht wurde.

Hinweis: Die (robuste) Funktion `pam()` aus dem Paket `cluster` kann auch mit allgemeinen Distanzen umgehen. Außerdem für gemischte Variablentypen gut geeignet: Das Paket `clustMixType`.

Zur Vorbereitung überführen wir die nominalen Merkmale in logische, d. h. binäre Merkmale, und löschen die Segmente sowie das Ergebnis der hierarchischen Clusteranalyse:

```
segment.num <- segment %>%
  mutate(Frau = Geschlecht=="Frau") %>%
  mutate(Eigenheim = Eigenheim=="Ja") %>%
  mutate(Mitgliedschaft = Mitgliedschaft=="Ja") %>%
  select(-Geschlecht, -Segment, -hc.clust)
```

Über die Funktion `mutate()` werden Variablen im Datensatz erzeugt oder verändert. Über `select()` werden einzene Variablen ausgewählt. Die "Pfeife" `%>%` übergeben das Ergebnis der vorherigen Funktion an die folgende.

Aufgrund von (1.) hängt das Ergebnis einer k-Means Clusteranalyse vom Zufall ab. Aus Gründen der Reproduzierbarkeit sollte daher der Zufallszahlengenerator gesetzt werden. Außerdem bietet es sich an verschiedene Startkonfigurationen zu versuchen. in der Funktion `kmeans()` erfolgt dies durch die Option `nstart=`. Hier mit `k=4` Clustern:

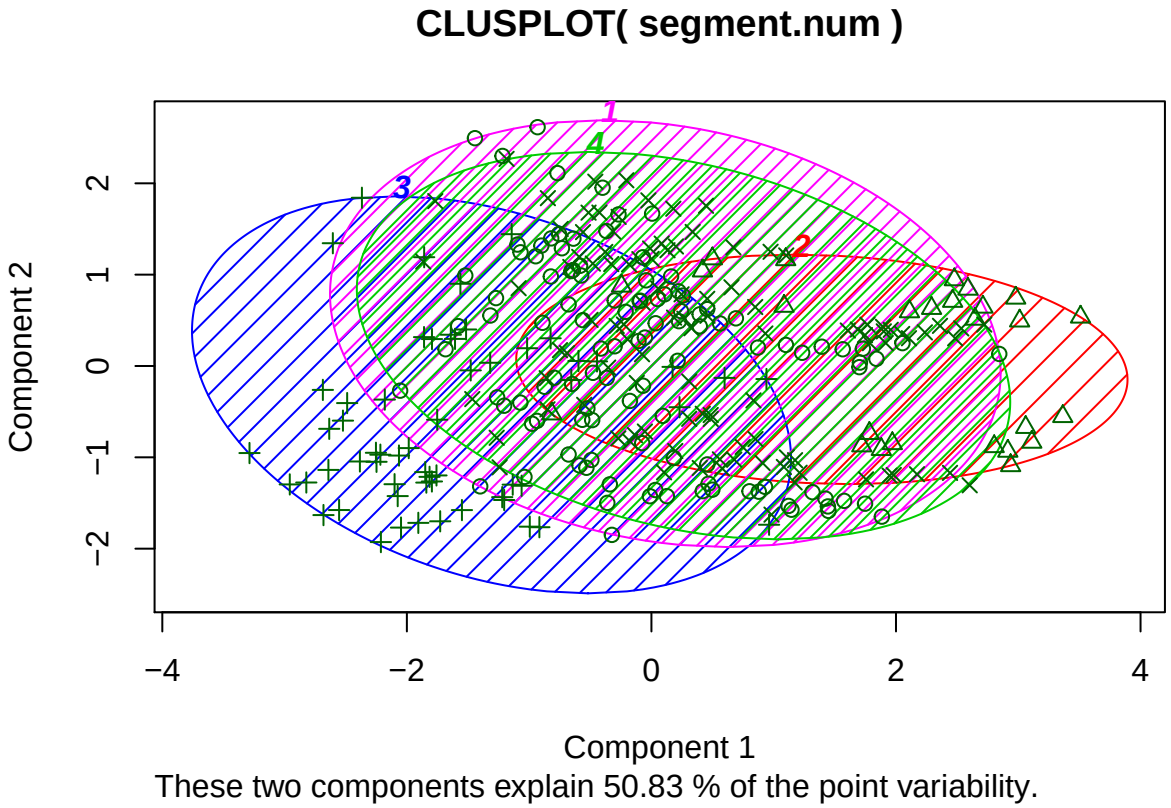
```
set.seed(1896)

seg.k <- kmeans(segment.num, centers = 4, nstart = 10)
seg.k
```

```
## K-means clustering with 4 clusters of sizes 111, 26, 58, 105
##
## Cluster means:
##      Alter Einkommen    Kinder Eigenheim Mitgliedschaft      Frau
## 1 42.91293  46049.08 1.6486486 0.5045045     0.10810811 0.5675676
## 2 56.35833  85972.56 0.3846154 0.5384615     0.03846154 0.5384615
## 3 27.01882  22607.81 1.2241379 0.2758621     0.20689655 0.4137931
## 4 43.56022  62599.99 1.5047619 0.4571429     0.12380952 0.5904762
##
## Clustering vector:
##   [1] 1 4 4 1 4 4 4 1 2 4 1 1 4 4 1 1 1 1 1 4 4 4 1 4 1 1 1 1 4 1 4 4 1 1 2
##  [36] 1 4 1 1 4 4 4 1 4 4 4 4 1 1 1 1 1 2 1 1 4 4 4 4 1 4 1 4 1 1 1 1 4 4 4
##  [71] 4 1 1 4 1 1 4 4 4 4 1 4 1 3 1 4 1 1 1 1 4 4 4 1 1 4 1 4 4 4 3 3 3 3 3
## [106] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [141] 3 3 3 3 3 3 3 3 3 3 1 2 4 2 2 4 1 1 2 2 4 4 1 1 4 2 4 4 1 2 2 3 4 1 2
## [176] 2 4 2 3 4 4 4 1 1 1 1 1 1 4 3 1 4 4 4 4 1 1 1 2 4 4 1 2 4 4 1 4 2 1 2
## [211] 4 3 4 2 2 4 2 1 4 3 1 2 2 4 2 4 4 1 4 4 1 1 1 1 1 3 1 1 4 1 4 3 1 4 1
## [246] 4 1 4 1 4 4 4 4 1 1 1 4 4 1 1 1 1 1 1 4 1 1 1 1 1 2 4 4 1 4 1 1 1 1 2
## [281] 4 4 4 4 1 4 1 4 4 4 1 4 1 4 1 4 1 1 4 1
##
## Within cluster sum of squares by cluster:
## [1] 3178960729 2217438154 1689348448 2811630349
##  (between_SS / total_SS =  90.6 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

Neben der Anzahl Beobachtungen im Cluster (z. B. 26 in Cluster 2) werden auch die Clusterzentren ausgegeben. Diese können dann direkt verglichen werden. Sie sehen z. B., dass das Durchschnittsalter in Cluster 3 mit 27 am geringsten ist. Der Anteil der Eigenheimbesitzer ist mit 54 % in Cluster 2 am höchsten.

Einen Plot der Scores auf den beiden ersten Hauptkomponenten können Sie über die Funktion `clusplot()` aus dem Paket `cluster` erhalten.

```
clusplot(segment.num, seg.k$cluster,
         color = TRUE, shade = TRUE, labels = 4)
```



Wie schon im deskriptiven Ergebnis: Die Cluster `1` und `4` unterscheiden sich (in den ersten beiden Hauptkomponenten) nicht wirklich. Vielleicht sollten dies noch zusammengefasst werden, d. h., mit `centers=3` die Analyse wiederholt werden?[6]

Übung: B3 Datensatz

Der B3 Datensatz *Heilemann, U. and Münch, H.J. (1996): West German Business Cycles 1963-1994: A Multivariate Discriminant Analysis. CIRET–Conference in Singapore, CIRET–Studien 50.* enthält Quartalsweise Konjunkturdaten aus (West-)Deutschland.

Er kann von https://goo.gl/0YCEHf heruntergeladen werden.

1. Wenn die Konjunkturphase `PHASEN` nicht berücksichtigt wird, wie viele Cluster könnte es geben? Ändert sich das Ergebnis, wenn die Variablen standardisiert werden?
2. Führen Sie eine k-Means Clusteranalyse mit 4 Clustern durch. Worin unterscheiden sich die gefundenen Segmente?

[6] Das Paket `NbClust`, siehe Malika Charrad, Nadia Ghazzali, Veronique Boiteau, Azam Niknafs (2014) *NbClust: An R Package for Determining the Relevant Number of Clusters in a Data Set*, Journal of Statistical Software, 61(6), 1-36. http://dx.doi.org/10.18637/jss.v061.i06, bietet viele Möglichkeiten die Anzahl der Cluster optimal zu bestimmen.

Literatur

- Chris Chapman, Elea McDonnell Feit (2015): *R for Marketing Research and Analytics*, Kapitel 11.3
- Reinhold Hatzinger, Kurt Hornik, Herbert Nagel (2011): *R – Einführung durch angewandte Statistik.* Kapitel 12
- Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani (2013): *An Introduction to Statistical Learning – with Applications in R*, http://www-bcf.usc.edu/~gareth/ISL/, Kapitel 10.3, 10.5

Lizenz

Versionshinweise:

- Datum erstellt: 2017-06-30
- R Version: 3.4.0
- `mosaic` Version: 0.14.4
- `cluster` Version: 2.0.6

Anhang 1: R Kurzreferenz

Vorbemerkungen

Eine Übersicht von nützlichen R Funktionen innerhalb der Datenanalyse.

Diese Kurzreferenz beschreibt ein kleinen Teil der R Funktionen, wobei größtenteils auf das Zusatzpaket `mosaic` zurückgegriffen wird. Sie basiert größtenteils auf der Vignette Minimal R von Randall Pruim.

Weitere Hilfe und Beispiele finden Sie, wenn Sie

```
?Befehl
```

eingeben.

- R unterscheidet zwischen Groß- und Kleinbuchstaben
- R verwendet den Punkt `.` als Dezimaltrennzeichen
- Fehlende Werte werden in R durch `NA` kodiert
- Kommentare werden mit dem Rautezeichen `#` eingeleitet; der Rest der Zeile von von R dann ignoriert.
- R wendet Befehle direkt an
- R ist objektorientiert, d. h. dieselbe Funktion hat evtl. je nach Funktionsargument unterschiedliche Rückgabewerte
- Zusätzliche Funktionalität kann über Zusatzpakete hinzugeladen werden. Diese müssen ggf. zunächst installiert werden
- Mit der Pfeiltaste nach oben können Sie einen vorherigen Befehl in der Konsole wieder aufrufen
- Eine Ergebniszuweiseung erfolgt über `<-`

Innerhalb von `mosaic`:

```
analysiere(y ~ x | z , data=Daten)
```

d. h., modelliere `y` in Abhängigkeit von `x` getrennt bzw. bedingt für `z` aus dem Datensatz `Daten`. Dabei können Teile (z. B. `y` und/ oder `z`) fehlen.[7]

Zusatzpakete müssen vor der ersten Benutzung einmalig installiert und geladen werden:

```
install.packages("Paket") # Einmalig installieren
library(Paket) # Laden, einmalig in jeder Sitzung
```

Daten

Daten einlesen und Datenvorverarbeitung sind häufig der (zeitlich) aufwendigste Teil einer Datenanalyse. Da die Daten die Grundlage sind, sollte auch hier sorgfältig gearbeitet und überprüft werden.

[7] Beim Mac ist `~` die Tastenkombination `alt+n`, `|` die Tastenkombination `alt+7`

Daten einlesen

```
read.table() # Allgemeinen Datensatz einlesen. Achtung: Optionen anpassen
read.csv2() # csv Datensatz einlesen (aus deutschsprachigem Excel)
file.choose() # Datei auswählen
meineDaten <- read.csv2(file.choose())
```

U. a. mit Hilfe des Zusatzpaketes `readxl` können Excel Dateien eingelesen werden:

```
meineDaten <- read_excel(file.choose())
```

Daten verarbeiten

```
str() # Datenstruktur
head() # Obere Zeilen
tail() # Untere Zeilen
nrow(); ncol() # Anzahl Zeilen; Spalten
rownames(); colnames() # Zeilennamen, Spaltennamen
```

Daten transformieren

Einzelne Variablen eines Datensatzes können über `$` ausgewählt werden: `Daten$Variable`. Allgemein kann über `Daten[i,j]` die i. Zeile und j. Spalte ausgewählt werden, wobei auch mehrere oder keine Zeile(n) bzw. Spalte(n) ausgewählt werden können. Über `c()` wird ein Vektor erzeugt.

```
as.factor() # Daten als Faktoren definieren
relevel() # Faktorstufen umordnen
droplevels() # Ungenutzte Faktorstufen entfernen
recode() # Umkodierung von Werten, Paket car
as.numeric() # Faktorstufen als numerische Daten verwenden
cut() # Aufteilung numerischer Werte in Intervalle

subset() # Teilmenge der Daten auswählen
na.omit() # Zeilen mit fehlenden Werten entfernen

log() # Logarithmusfunktion
exp() # Exponentialfunktion
sqrt() # Quadratwurzelfunktion
abs() # Betragsfunktion

rowSums() # Zeilensumme
rowMeans() # Zeilenmittelwert
```

Innerhalb des Paketes `dplyr` (wird mit `mosaic` geladen) gibt es u. a. folgende Funktionen:

```
filter() # Filtert Beobachtungen eines Datensatzes
select() # Wählt Variablen eines Datensatzes aus
mutate() # Erzeugt neue Variable bzw. verändert bestehende
rename() # Benennt Variablen um
arrange() # Sortiert Beobachtungen eines Datensatzes
%>% # Übergebe das Ergebnis der vorhergehenden Funktion an die folgende
```

Grafische Verfahren

Vor jeder mathematisch-statistischen Analyse sollte eine explorative, grafische Analyse erfolgen. Die folgenden Befehle sind aus dem Paket `mosaic`.

```
bargraph() # Balkendiagramm
histogram() # Histogramm
bwplot() # Boxplot
xyplot() # Streudiagrmm
```

Nicht aus dem Paket `mosaic` sind:

```
mosaicplot() # Mosaicplot
corrplot() # Korrelationsplot, Paket corrplot
ggpairs() # Matrixplot, Paket GGally
heatmap() # Heatmap
```

Deskriptive Statistik

Eine gute Zusammenfassung liefert der `mosaic` Befehl:

```
favstats()
```

Ansonsten (`mosaic` angepasst):

```
tally() # Tabellierung, Häufigkeiten
prop() # Anteile
mean() # Arithmetischer Mittelwert
median() # Median
quantile() # Quantile
sd() # Standardabweichung
var() # Varianz
IQR() # Interquartilsabstand
cov() # Kovarianz
cor() # Korrelationskoefizient
```

Inferenzstatistik

Randomisierung, Simulationen

Größtenteils `mosaic`:

```
set.seed() # Zufallszahlengenerator setzen
rflip() # Münzwurf
do() # Wiederholung (Schleife)
sample() # Stichprobe ohne Zurücklegen
resample() # Stichprobe mit Zurücklegen
shuffle() # Permutation
rnorm() # Normalverteilte Zufallszahlen
```

Verteilungen

Innerhalb der Funktionen müssen ggf. die Parameter, d. h. `mean=`, `sd=`, bzw. `df=` angepasst werden. (Das vorgestellte `x` steht für in `mosaic` angepasste Versionen.)

```
xpchisq() # Verteilungsfunktion Chi² Verteilung
xqchisq() # Quantilsfunktion Chi² Verteilung
xpnorm() # Verteilungsfunktion Normalverteilung
xqnorm() # Quantilsfunktion Normalverteilung
xpt() # Verteilungsfunktion t-Vverteilung
xqt() # Quantilsfunktion t-Vverteilung
```

Analoger Aufbau für weitere Verteilungen, z. B. `_pniom()`, `_f()`.

Testverfahren

Einige der Testverfahren wurden von `mosaic` angepasst.

```
t.test() # t-Test
prop.test() # Binomialtest (approximativ)
xchisq.test() # Chi²-Test
aov() # Varianzanalyse
```

Der Nicht-parametrische Wilcoxon-Test `wilcox.test()` ist nicht im Paket `mosaic`, hat daher einen leicht anderen Funktionsaufruf. Einen Test auf Normalverteilung führt der Shapiro-Wilk Test durch: `shapiro.test()`.

Multivariate Verfahren

```
lm() # Lineare Regression
glm(, family="binomial") # Logistische Regression
plotModel() # Modell zeichnen
coef() # Koeffizienten extrahieren
residuals() # Residuen einer Regression
fitted() # Angepasste Werte einer Regression
predict() # Vorhersagen
```

In `mosaic` kann das Ergebnis einer solchen Regression über `makeFun()` in eine einfache mathematische Funktion überführt werden. `plotFun()` zeichnet das Ergebnis. `step()` führt eine Variablenselektion durch.

Weitere Verfahren - nicht `mosaic`:

```
prcomp() # Hauptkomponentenanalyse (PCA)
alpha() # Reliabilitätsanalys, Paket psych
dist() # Distanzen
hclust() # Hierachsiche Clusteranalyse
kmeans() # k-Means Clusterverfahren
rpart() # Klassifikations- und Regressionsbäume, Paket rpart
```

Versionshinweise:

- Datum erstellt: 2017-06-30
- R Version: 3.4.0

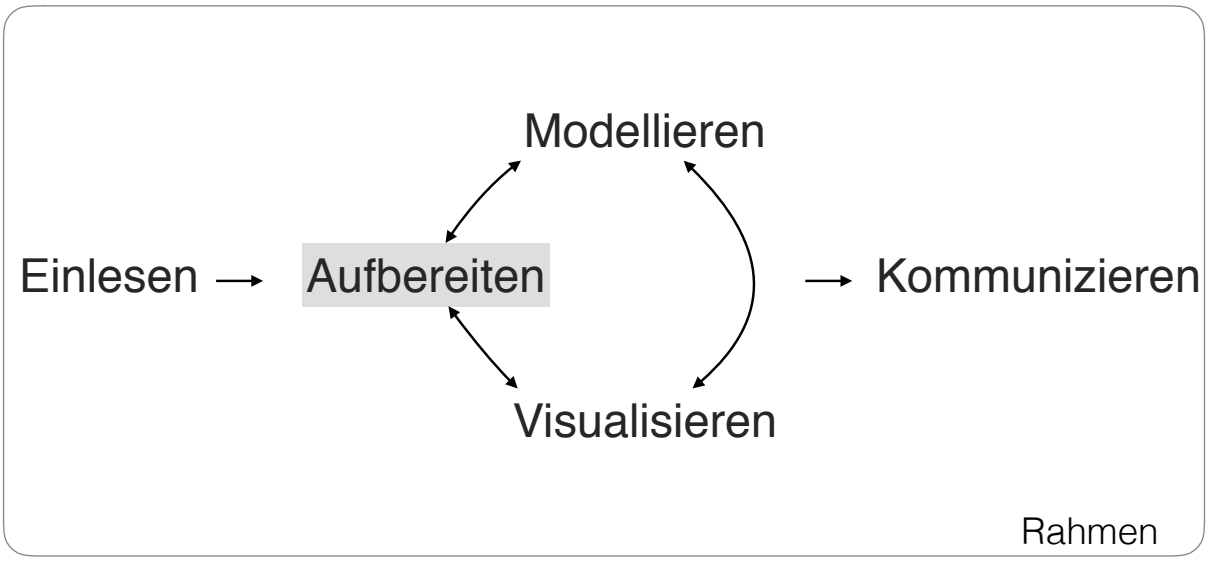


Abbildung 1: Daten aufbereiten

- `mosaic` Version: 0.14.4

Anhang 2: Datenjudo

In diesem Kapitel benötigte Pakete:

```
library(tidyverse)  # Datenjudo
```

```
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
## Loading tidyverse: purrr
```

```
## Conflicts with tidy packages ----------------------------------------------
```

```
## %+%():    ggplot2, psych
## alpha():  ggplot2, psych
## count():  dplyr, mosaic
## do():     dplyr, mosaic
## expand(): tidyr, Matrix
## filter(): dplyr, stats
## lag():    dplyr, stats
## select(): dplyr, MASS
## tally():  dplyr, mosaic
```

```
library(stringr)   # Texte bearbeiten
library(car)  # für 'recode'
```

```
##
```

```
## Attaching package: 'car'

## The following object is masked from 'package:purrr':
##
##     some

## The following object is masked from 'package:boot':
##
##     logit

## The following object is masked from 'package:psych':
##
##     logit

## The following object is masked from 'package:effects':
##
##     Prestige

## The following objects are masked from 'package:mosaic':
##
##     deltaMethod, logit

## The following object is masked from 'package:dplyr':
##
##     recode
```

Mit *Datenjudo* ist gemeint, die Daten für die eigentliche Analyse "aufzubereiten". Unter *Aufbereiten* ist hier das Umformen, Prüfen, Bereinigen, Gruppieren und Zusammenfassen von Daten gemeint. Die deskriptive Statistik fällt unter die Rubrik Aufbereiten. Kurz gesagt: Alles, wan tut, nachdem die Daten "da" sind und bevor man mit anspruchsvoller(er) Modellierung beginnt.

Ist das Aufbereiten von Daten auch nicht statistisch anspruchsvoll, so ist es trotzdem von großer Bedeutung und häufig recht zeitintensiv. Eine Anekdote zur Relevanz der Datenaufbereitung, die (so will es die Geschichte) mir an einer Bar nach einer einschlägigen Konferenz erzählt wurde (daher keine Quellenangebe, Sie verstehen...). Eine Computerwissenschaftlerin aus den USA (deutschen Ursprungs) hatte einen beeindruckenden "Track Record" an Siegen in Wettkämpfen der Datenanalyse. Tatsächlich hatte sie keine besonderen, raffinierten Modellierungstechniken eingesetzt; klassische Regression war ihre Methode der Wahl. Bei einem Wettkampf, bei dem es darum ging, Krebsfälle aus Krankendaten vorherzusagen (z.B. von Röntgenbildern) fand sie nach langem Datenjudo heraus, dass in die "ID-Variablen" Information gesickert war, die dort nicht hingehörte und die sie nutzen konnte für überraschend (aus Sicht der Mitstreiter) gute Vorhersagen zu Krebsfällen. Wie war das möglich? Die Daten stammten aus mehreren Kliniken, jede Klinik verwendete ein anderes System, um IDs für Patienten zu erstellen. Überall waren die IDs stark genug, um die Anonymität der Patienten sicherzustellen, aber gleich wohl konnte man (nach einigem Judo) unterscheiden, welche ID von welcher Klinik stammte. Was das bringt? Einige Kliniken waren reine Screening-Zentren, die die Normalbevölkerung versorgte. Dort sind wenig Krebsfälle zu erwarten. Andere Kliniken jedoch waren Onkologie-Zentren für bereits bekannte Patienten oder für

Patienten mit besonderer Risikolage. Wenig überraschen, dass man dann höhere Krebsraten vorhersagen kann. Eigentlich ganz einfach; besondere Mathe steht hier (zumindest in dieser Geschichte) nicht dahinter. Und, wenn man den Trick kennt, ganz einfach. Aber wie so oft ist es nicht leicht, den Trick zu finden. Sorgfältiges Datenjudo hat hier den Schlüssel zum Erfolg gebracht.

Typische Probleme

Bevor man seine Statistik-Trickkiste so richtig schön aufmachen kann, muss man die Daten häufig erst noch in Form bringen. Das ist nicht schwierig in dem Sinne, dass es um komplizierte Mathe ginge. Allerdings braucht es mitunter recht viel Zeit und ein paar (oder viele) handwerkliche Tricks sind hilfreich. Hier soll das folgende Kapitel helfen.

Typische Probleme, die immer wieder auftreten, sind:

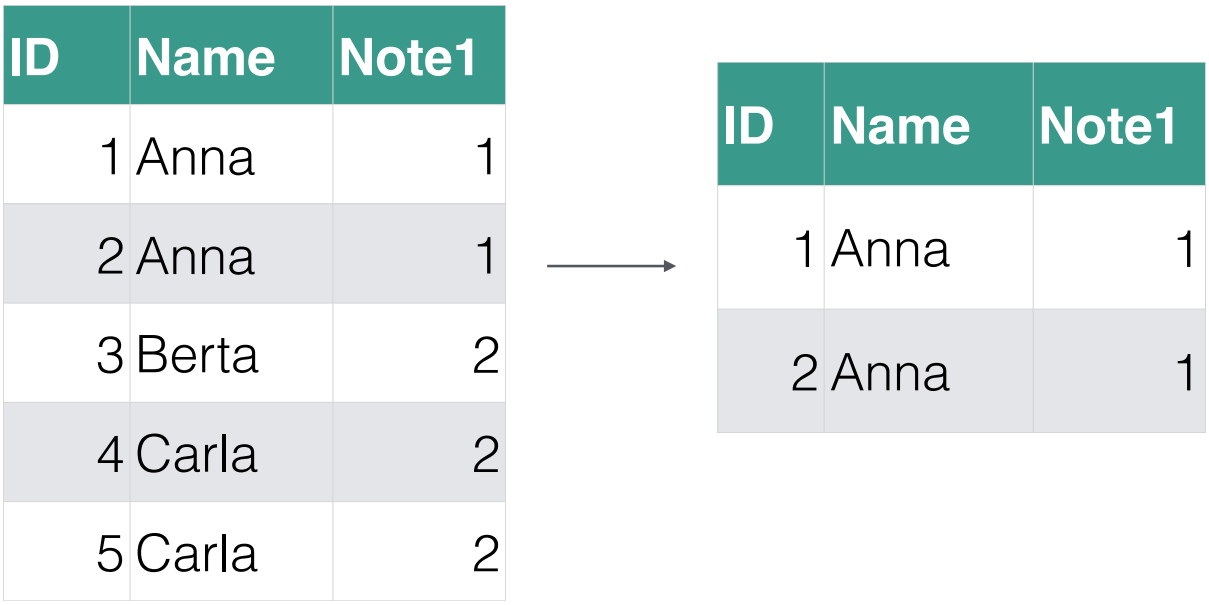
- *Fehlende Werte*: Irgend jemand hat auf eine meiner schönen Fragen in der Umfrage nicht geantwortet!
- *Unerwartete Daten*: Auf die Frage, wie viele Facebook-Freunde er oder sie habe, schrieb die Person "I like you a lot". Was tun???
- *Daten müssen umgeformt werden*: Für jede der beiden Gruppen seiner Studie hat Joachim einen Google-Forms-Fragebogen aufgesetzt. Jetzt hat er zwei Tabellen, die er "verheiraten" möchte. Geht das?
- *Neue Variablen (Spalten) berechnen*: Ein Student fragt nach der Anzahl der richtigen Aufgaben in der Statistik-Probeklausur. Wir wollen helfen und im entsprechenden Datensatz eine Spalte erzeugen, in der pro Person die Anzahl der richtig beantworteten Fragen steht.

Daten aufbereiten mit `dplyr`

Es gibt viele Möglichkeiten, Daten mit R aufzubereiten; `dplyr` ist ein populäres Paket dafür. Eine zentrale Idee von `dplyr` ist, dass es nur ein paar wenige Grundbausteine geben sollte, die sich gut kombinieren lassen. Sprich: Wenige grundlegende Funktionen mit eng umgrenzter Funktionalität. Der Autor, Hadley Wickham, sprach einmal in einem Forum (citation needed), dass diese Befehle wenig können, das Wenige aber gut. Ein Nachteil dieser Konzeption kann sein, dass man recht viele dieser Bausteine kombinieren muss, um zum gewünschten Ergebnis zu kommen. Außerdem muss man die Logik des Baukastens gut verstanden habe - die Lernkurve ist also erstmal steiler. Dafür ist man dann nicht darauf angewiesen, dass es irgendwo "Mrs Right" gibt, die genau das kann, was ich will. Außerdem braucht man sich auch nicht viele Funktionen merken. Es reicht einen kleinen Satz an Funktionen zu kennen (die praktischerweise konsistent in Syntax und Methodik sind).

Willkommen in der Welt von `dyplr`! `dplyr` hat seinen Namen, weil es sich ausschließlich um *D*ataframes bemüht; es erwartet einen Dataframe als Eingabe und gibt einen Dataframe zurück (zumindest bei den meisten Befehlen).

Diese Bausteine sind typische Tätigkeiten im Umgang mit Daten; nichts Überraschendes. Schauen wir uns diese Bausteine näher an.



ID	Name	Note1
1	Anna	1
2	Anna	1
3	Berta	2
4	Carla	2
5	Carla	2

ID	Name	Note1
1	Anna	1
2	Anna	1

Abbildung 2: Zeilen filtern

Zeilen filtern mit `filter`

Häufig will man bestimmte Zeilen aus einer Tabelle filtern. Zum Beispiel man arbeitet für die Zigarettenindustrie und ist nur an den Rauchern interessiert (die im Übrigen unser Gesundheitssystem retten [@kraemer2011wir]), nicht an Nicht-Rauchern; es sollen die nur Umsatzzahlen des letzten Quartals untersucht werden, nicht die vorherigen Quartale; es sollen nur die Daten aus Labor X (nicht Labor Y) ausgewertet werden etc.

Ein Sinnbild:

Merke:

> Die Funktion `filter` filtert Zeilen aus einem Dataframe.

Schauen wir uns einige Beispiel an; zuerst die Daten laden nicht vergessen. Achtung: "Wohnen" die Daten in einem Paket, muss dieses Paket installiert sein, damit man auf die Daten zugreifen kann.

```
data(profiles, package = "okcupiddata")  # Das Paket muss installiert sein
```

```
df_frauen <- filter(profiles, sex == "f")  # nur die Frauen
df_alt <- filter(profiles, age > 70)  # nur die alten
df_alte_frauen <- filter(profiles, age > 70, sex == "f")  # nur die alten Frauen, d.h. U
df_nosmoke_nodrinks <- filter(profiles, smokes == "no" | drinks == "not at all")
# liefert alle Personen, die Nicht-Raucher *oder* Nicht-Trinker sind
```

Gar nicht so schwer, oder? Allgemeiner gesprochen werden diejenigen Zeilen gefiltert (also behalten bzw. zurückgeliefert), für die das Filterkriterium `TRUE` ist.

```
Manche Befehle wie `filter` haben einen Allerweltsnamen; gut möglich, dass ein Befehl mi
```

Aufgaben[8]

```
Richtig oder Falsch!?


1. `filter` filtert Spalten.
1. `filter` ist eine Funktion aus dem Paket `dplyr`.
1. `filter` erwartet als ersten Parameter das Filterkriterium.
1. `filter` lässt nur ein Filterkriterium zu.
1. Möchte man aus dem Datensatz `profiles` (`okcupiddata`) die Frauen filtern, so ist fo
```

Vertiefung: Fortgeschrittene Beispiele für `filter`

Einige fortgeschrittene Beispiele für `filter`:

Man kann alle Elemente (Zeilen) filtern, die zu einer Menge gehören und zwar mit diesem Operator: `%in%`:

```
filter(profiles, body_type %in% c("a little extra", "average"))
```

Besonders Textdaten laden zu einigen Extra-Überlegungen ein; sagen wir, wir wollen alle Personen filtern, die Katzen bei den Haustieren erwähnen. Es soll reichen, wenn `cat` ein Teil des Textes ist; also `likes dogs and likes cats` wäre OK (soll gefiltert werden). Dazu nutzen wir ein Paket zur Bearbeitung von Strings (Textdaten):

```
filter(profiles, str_detect(pets, "cats"))
```

Ein häufiger Fall ist, Zeilen *ohne* fehlende Werte (`NA`s) zu filtern. Das geht einfach:

```
profiles_keine_nas <- na.omit(profiles)
```

Aber was ist, wenn wir nur bei bestimmten Spalten wegen fehlender Werte besorgt sind? Sagen wir bei `income` und bei `sex`:

```
filter(profiles, !is.na(income) | !is.na(sex))
```

Spalten wählen mit `select`

Das Gegenstück zu `filter` ist `select`; dieser Befehl liefert die gewählten Spalten zurück. Das ist häufig praktisch, wenn der Datensatz sehr "breit" ist, also viele Spalten enthält. Dann kann es übersichtlicher sein, sich nur die relevanten auszuwählen. Das Sinnbild für diesen Befehl:

Merke:

> Die Funktion select wählt Spalten aus einem Dataframe aus.

[8] F, R, F, F, R

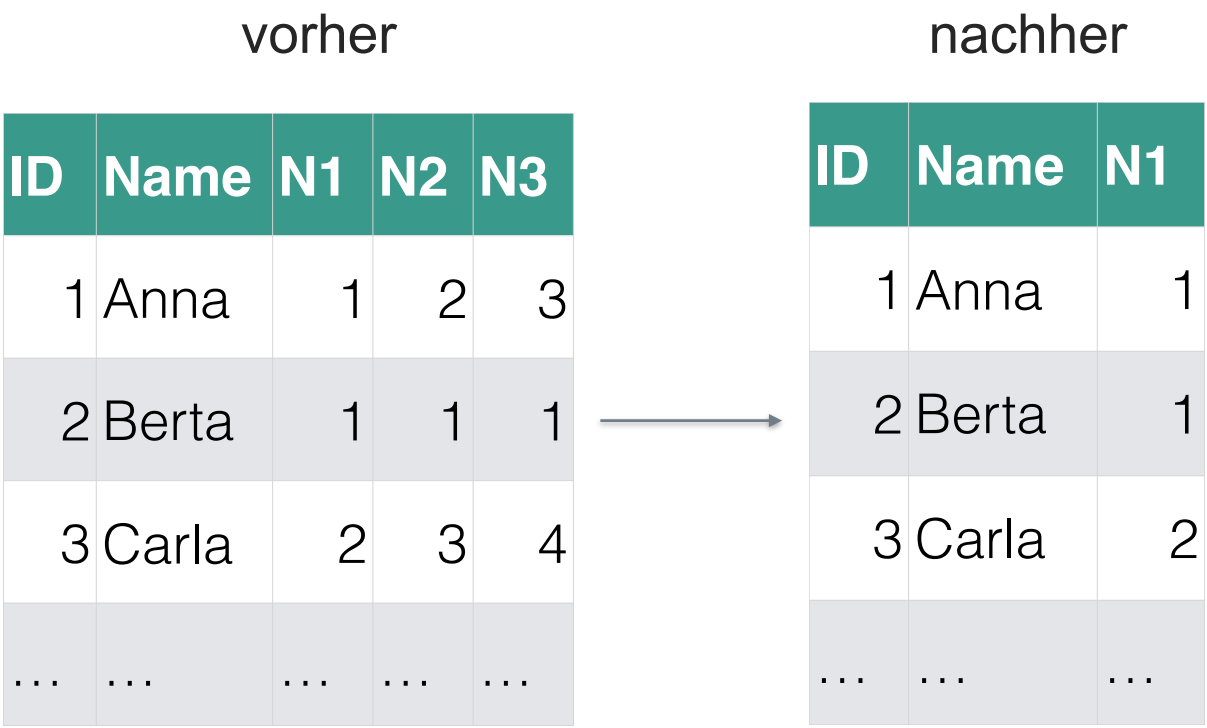


Abbildung 3: Spalten auswählen

Laden wir als ersten einen Datensatz.

```
stats_test <- read.csv("./data/test_inf_short.csv")
```

Dieser Datensatz beinhaltet Daten zu einer Statistikklausur.

Beachten Sie, dass diese Syntax davon ausgeht, dass sich die Daten in einem Unterordner mit dem Namen `data` befinden, welcher sich im Arbeitsverzeichnis befindet[9].

```
select(stats_test, score)  # Spalte `score` auswählen
select(stats_test, score, study_time)  # Splaten `score` und `study_time` auswählen
select(stats_test, score:study_time) # dito
select(stats_test, 5:6) Spalten 5 bis 6 auswählen
```

Tatsächlich ist der Befehl `select` sehr flexibel; es gibt viele Möglichkeiten, Spalten auszuwählen. Im `dplyr`-Cheatsheet findet sich ein guter Überblick dazu.

[Aufgaben]Aufgaben[10]

```
Richtig oder Falsch!?

1. `select` wählt *Zeilen* aus.
1. `select` ist eine Funktion aus dem Paket `knitr`.
1. Möchte man zwei Spalten auswählen, so ist folgende Syntax prinzipiell korrekt: `selec
```

[9] der angegebene Pfad ist also *relativ* zum aktuellen Verzeichnis.
[10] F, F, R, R, F

1. Möchte man Spalten 1 bis 10 auswählen, so ist folgende Syntax prinzipiell korrekt: `s
1. Mit `select` können Spalten nur bei ihrem Namen, aber nicht bei ihrer Nummer aufgeruf

Zeilen sortieren mit `arrange`

Man kann zwei Arten des Umgangs mit R unterscheiden: Zum einen der "interaktive Gebrauch" und zum anderen "richtiges Programmieren". Im interaktiven Gebrauch geht es uns darum, die Fragen zum aktuell vorliegenden Datensatz (schnell) zu beantworten. Es geht nicht darum, eine allgemeine Lösung zu entwickeln, die wir in die Welt verschicken können und die dort ein bestimmtes Problem löst, ohne dass der Entwickler (wir) dabei Hilfestellung geben muss. "Richtige" Software, wie ein R-Paket oder Microsoft Powerpoint, muss diese Erwartung erfüllen; "richtiges Programmieren" ist dazu vonnöten. Natürlich sind in diesem Fall die Ansprüche an die Syntax (der "Code", hört sich cooler an) viel höher. In dem Fall muss man alle Eventualitäten voraussehen und sicherstellen, dass das Programm auch beim merkwürdigsten Nutzer brav seinen Dienst tut. Wir haben hier, beim interaktiven Gebrauch, niedrigere Ansprüche bzw. andere Ziele.

Beim interaktiven Gebrauch von R (oder beliebigen Analyseprogrammen) ist das Sortieren von Zeilen eine recht häufige Tätigkeit. Typisches Beispiel wäre der Lehrer, der eine Tabelle mit Noten hat und wissen will, welche Schüler die schlechtesten oder die besten sind in einem bestimmten Fach. Oder bei der Prüfung der Umsätze nach Filialen möchten wir die umsatzstärksten sowie -schwächsten Niederlassungen kennen.

Ein R-Befehl hierzu ist `arrange`; einige Beispiele zeigen die Funktionsweise am besten:

```
arrange(stats_test, score) # liefert die *schlechtesten* Noten zuerst zurück
arrange(stats_test, -score) # liefert die *besten* Noten zuerst zurück
arrange(stats_test, interest, score)
```

```
##     X                 V_1 study_time self_eval interest score
## 1 234 23.01.2017 18:13:15          3         1        1    17
## 2   4 06.01.2017 09:58:05          2         3        2    18
## 3 131 19.01.2017 18:03:45          2         3        4    18
## 4 142 19.01.2017 19:02:12          3         4        1    18
## 5  35 12.01.2017 19:04:43          1         2        3    19
## 6  71 15.01.2017 15:03:29          3         3        3    20
```

```
##    X                 V_1 study_time self_eval interest score
## 1  3 05.01.2017 23:33:47          5        10        6    40
## 2  7 06.01.2017 14:25:49         NA        NA       NA    40
## 3 29 12.01.2017 09:48:16          4        10        3    40
## 4 41 13.01.2017 12:07:29          4        10        3    40
## 5 58 14.01.2017 15:43:01          3         8        2    40
## 6 83 16.01.2017 10:16:52         NA        NA       NA    40
```

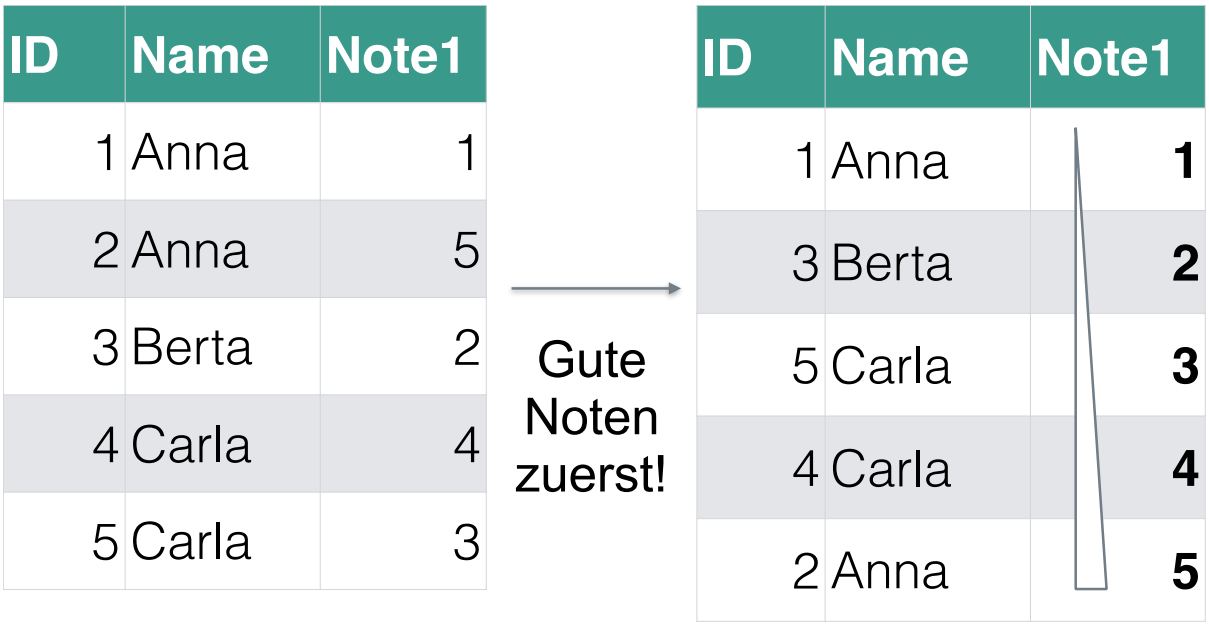


Abbildung 4: Spalten sortieren

```
##     X                 V_1 study_time self_eval interest score
## 1 234 23.01.2017 18:13:15          3         1        1    17
## 2 142 19.01.2017 19:02:12          3         4        1    18
## 3 221 23.01.2017 11:40:30          1         1        1    23
## 4 230 23.01.2017 16:27:49          1         1        1    23
## 5  92 17.01.2017 17:18:55          1         1        1    24
## 6 107 18.01.2017 16:01:36          3         2        1    24
```

Einige Anmerkungen. Die generelle Syntax lautet `arrange(df, Spalte1, ...)`, wobei `df` den Dataframe bezeichnet und `Spalte1` die erste zu sortierende Spalte; die Punkte `...` geben an, dass man weitere Parameter übergeben kann. Man kann sowohl numerische Spalten als auch Textspalten sortieren. Am wichtigsten ist hier, dass man weitere Spalten übergeben kann. Dazu gleich mehr.

Standardmäßig sortiert `arrange` *aufsteigend* (weil kleine Zahlen im Zahlenstrahl vor den großen Zahlen kommen). Möchte man diese Reihenfolge umdrehen (große Werte zuert, d.h. *absteigend*), so kann man ein Minuszeichen vor den Namen der Spalte setzen.

Gibt man *zwei oder mehr* Spalten an, so werden pro Wert von `Spalte1` die Werte von `Spalte2` sortiert etc; man betrachte den Output des Beispiels oben dazu.

Merke:

> Die Funktion arrange sortiert die Zeilen eines Datafames.

Ein Sinnbild zur Verdeutlichung:

Ein ähnliches Ergebnis erhält mit man `top_n()`, welches die `n` *größten Ränge* widergibt:

```
top_n(stats_test, 3)
```

```
## Selecting by score
```

```
##      X                 V_1 study_time self_eval interest score
## 1    3 05.01.2017 23:33:47          5        10        6    40
## 2    7 06.01.2017 14:25:49         NA        NA       NA    40
## 3   29 12.01.2017 09:48:16          4        10        3    40
## 4   41 13.01.2017 12:07:29          4        10        3    40
## 5   58 14.01.2017 15:43:01          3         8        2    40
## 6   83 16.01.2017 10:16:52         NA        NA       NA    40
## 7  116 18.01.2017 23:07:32          4         8        5    40
## 8  119 19.01.2017 09:05:01         NA        NA       NA    40
## 9  132 19.01.2017 18:22:32         NA        NA       NA    40
## 10 175 20.01.2017 23:03:36          5        10        5    40
## 11 179 21.01.2017 07:40:05          5         9        1    40
## 12 185 21.01.2017 15:01:26          4        10        5    40
## 13 196 22.01.2017 13:38:56          4        10        5    40
## 14 197 22.01.2017 14:55:17          4        10        5    40
## 15 248 24.01.2017 16:29:45          2        10        2    40
## 16 249 24.01.2017 17:19:54         NA        NA       NA    40
## 17 257 25.01.2017 10:44:34          2         9        3    40
## 18 306 27.01.2017 11:29:48          4         9        3    40
```

```
top_n(stats_test, 3, interest)
```

```
##     X                 V_1 study_time self_eval interest score
## 1   3 05.01.2017 23:33:47          5        10        6    40
## 2   5 06.01.2017 14:13:08          4         8        6    34
## 3  43 13.01.2017 14:14:16          4         8        6    36
## 4  65 15.01.2017 12:41:27          3         6        6    22
## 5 110 18.01.2017 18:53:02          5         8        6    37
## 6 136 19.01.2017 18:22:57          3         1        6    39
## 7 172 20.01.2017 20:42:46          5        10        6    34
## 8 214 22.01.2017 21:57:36          2         6        6    31
## 9 301 27.01.2017 08:17:59          4         8        6    33
```

Gibt man *keine* Spalte an, so bezieht sich `top_n` auf die letzte Spalte im Datensatz.

Da sich hier mehrere Personen den größten Rang (Wert 40) teilen, bekommen wir *nicht* 3 Zeilen zurückgeliefert, sondern entsprechend mehr.

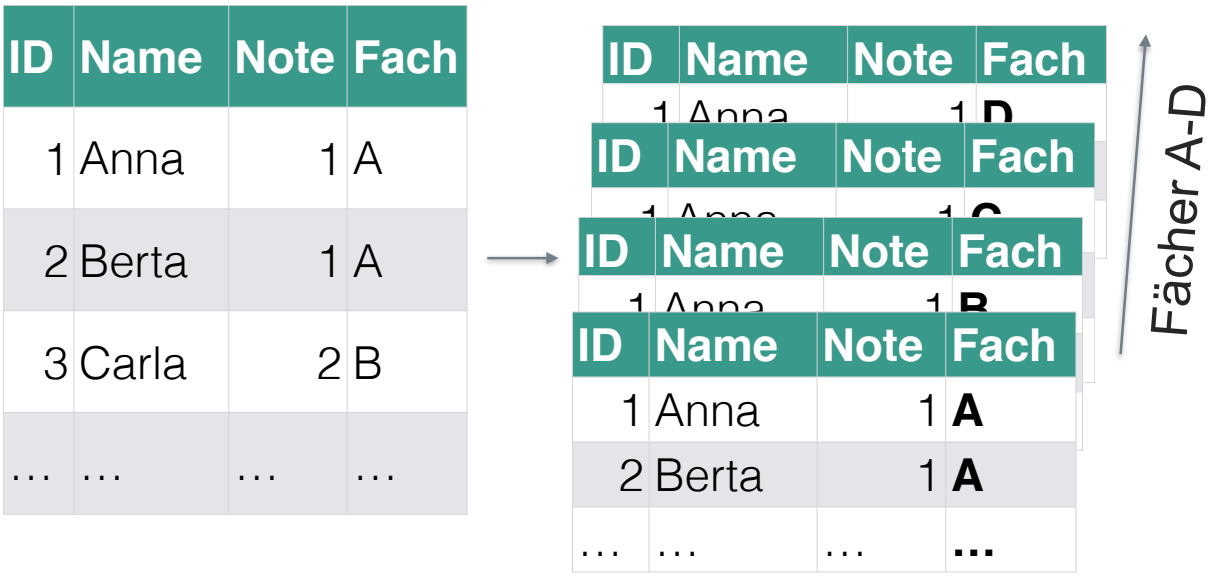


Abbildung 5: Datensätze nach Subgruppen aufteilen

Aufgaben[11]

```
Richtig oder Falsch!?

1. `arrange` arrangiert Spalten.
1. `arrange` sortiert im Standard absteigend.
1. `arrange` lässt nur ein Sortierkriterium zu.
1. `arrange` kann numerische Werte, aber nicht Zeichenketten sortieren.
1. `top_n(5)` liefert die fünf kleinsten Ränge.
```

Datensatz gruppieren mit `group_by`

Einen Datensatz zu gruppieren ist ebenfalls eine häufige Angelegenheit: Was ist der mittlere Umsatz in Region X im Vergleich zu Region Y? Ist die Reaktionszeit in der Experimentalgruppe kleiner als in der Kontrollgruppe? Können Männer schneller ausparken als Frauen? Man sieht, dass das Gruppieren v.a. in Verbindung mit Mittelwerten oder anderen Zusammenfassungen sinnvol ist; dazu im nächsten Abschnitt mehr.

In der Abbildung wurde der Datensatz anhand der Spalte `Fach` in mehrere Gruppen geteilt. Wir könnten uns als nächstes z.B. Mittelwerte pro Fach - d.h. pro Gruppe (pro Ausprägung von `Fach`) - ausgeben lassen; in diesem Fall vier Gruppen (Fach A bis D).

```
test_gruppiert <- group_by(stats_test, interest)
test_gruppiert
```

```
## # A tibble: 306 x 6
## # Groups:   interest [7]
```

[11] F, F, F, F, R

```
##        X                 V_1 study_time self_eval interest score
##    <int>              <fctr>      <int>     <int>    <int> <int>
##  1     1 05.01.2017 13:57:01          5         8        5    29
##  2     2 05.01.2017 21:07:56          3         7        3    29
##  3     3 05.01.2017 23:33:47          5        10        6    40
##  4     4 06.01.2017 09:58:05          2         3        2    18
##  5     5 06.01.2017 14:13:08          4         8        6    34
##  6     6 06.01.2017 14:21:18         NA        NA       NA    39
##  7     7 06.01.2017 14:25:49         NA        NA       NA    40
##  8     8 06.01.2017 17:24:53          2         5        3    24
##  9     9 07.01.2017 10:11:17          2         3        5    25
## 10    10 07.01.2017 18:10:05          4         5        5    33
## # ... with 296 more rows
```

Schaut man sich nun den Datensatz an, sieht man erstmal wenig Effekt der Gruppierung. R teilt uns lediglich mit `Groups: interest [7]`, dass es die Gruppen gibt, aber es gibt keine extra Spalte oder sonstige Anzeichen der Gruppierung. Aber keine Sorge, wenn wir gleich einen Mittelwert ausrechnen, bekommen wir den Mittelwert pro Gruppe!

Ein paar Hinweise: `Source: local data frame [306 x 6]` will sagen, dass die Ausgabe sich auf einen `tibble` bezieht[12], also eine bestimmte Art von Dataframe. `Groups: interest [7]` zeigt, dass der Tibble in 7 Gruppen - entsprechend der Werte von `interest` aufgeteilt ist.

`group_by` an sich ist nicht wirklich nützlich. Nützlich wird es erst, wenn man weitere Funktionen auf den gruppierten Datensatz anwendet - z.B. Mittelwerte ausrechnet (z.B mit `summarise`, s. unten). Die nachfolgenden Funktionen (wenn sie aus `dplyr` kommen), berücksichtigen nämlich die Gruppierung. So kann man einfach Mittelwerte pro Gruppe ausrechnen.

Aufgaben[13]

```
Richtig oder Falsch!?

1. Mit `group_by` gruppiert man einen Datensatz.
1. `group_by` lässt nur ein Gruppierungskriterium zu.
1. Die Gruppierung durch `group_by` wird nur von Funktionen aus `dplyr` erkannt.
1. `group_by` ist sinnvoll mit `summarise` zu kombinieren.
```

Merke:

> Mit group_by teilt man einen Datensatz in Gruppen ein, entsprechend der Werte einer mehrerer Spalten.

[12] http://stackoverflow.com/questions/29084380/what-is-the-meaning-of-the-local-data-frame-message-from-dplyrprint-tbl-df
[13] R, F, R, R

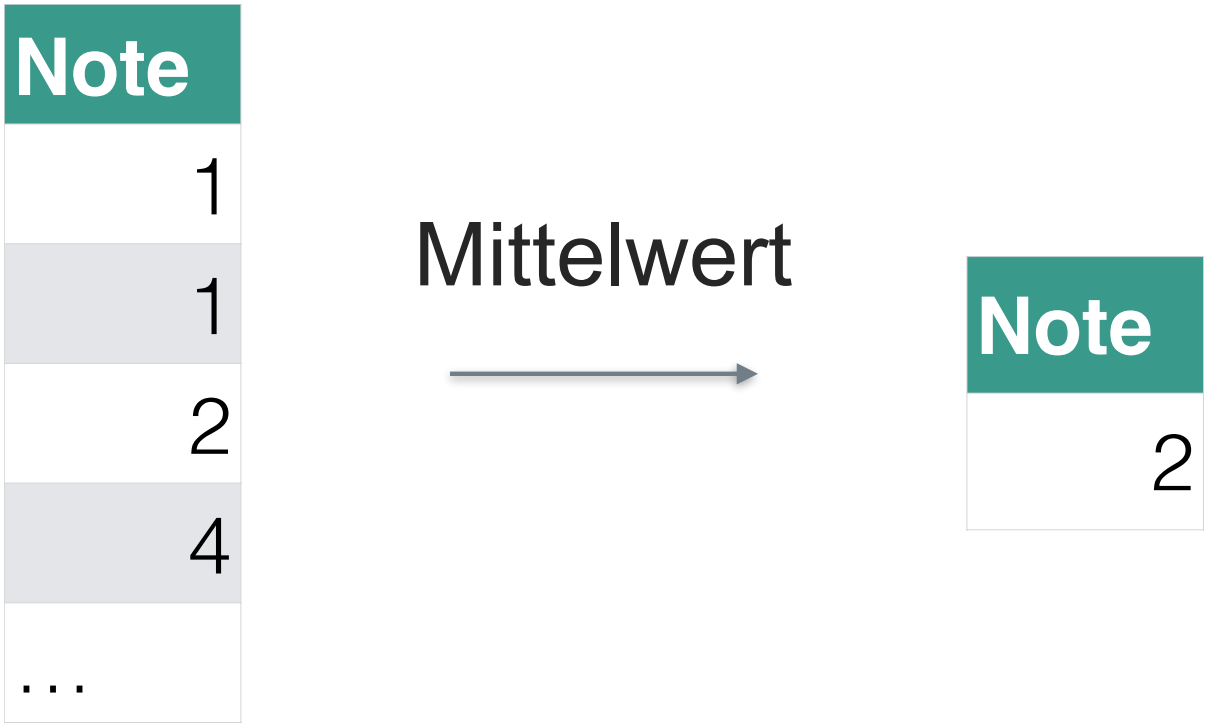


Abbildung 6: Spalten zu einer Zahl zusammenfassen

Eine Spalte zusammenfassen mit `summarise`

Vielleicht die wichtigste oder häufigte Tätigkeit in der Analyse von Daten ist es, eine Spalte zu *einem* Wert zusammenzufassen. Anders gesagt: Einen Mittelwert berechnen, den größten (kleinsten) Wert heraussuchen, die Korrelation berechnen oder eine beliebige andere Statistik ausgeben lassen. Die Gemeinsamkeit dieser Operaitonen ist, dass sie eine Spalte zu einem Wert zusammenfassen, "aus Spalte mach Zahl", sozusagen. Daher ist der Name des Befehls `summarise` ganz passend. Genauer gesagt fasst dieser Befehl eine Spalte zu einer Zahl zusammen *anhand* einer Funktion wie `mean` oder `max`. Hierbei ist jede Funktion erlaubt, die eine Spalte als Input verlangt und eine Zahl zurückgibt; andere Funktionen sind bei `summarise` nicht erlaubt.

```
summarise(stats_test, mean(score))
```

```
##   mean(score)
## 1    31.12092
```

Man könnte diesen Befehl so ins Deutsche übersetzen: `Fasse aus Tabelle stats_test die Spalte score anhand des Mittelwerts zusammen`. Nicht vergessen, wenn die Spalte `score` fehlende Werte hat, wird der Befehl `mean` standardmäßig dies mit `NA` quittieren.

Jetzt können wir auch die Gruppierung nutzen:

```
test_gruppiert <- group_by(stats_test, interest)
summarise(test_gruppiert, mean(score))
```

```
## # A tibble: 7 x 2
```

```
##   interest `mean(score)`
##      <int>         <dbl>
## 1        1      28.26667
## 2        2      29.70213
## 3        3      30.80303
## 4        4      29.92683
## 5        5      32.51111
## 6        6      34.00000
## 7       NA      33.08824
```

Der Befehl `summarise` erkennt also, wenn eine (mit `group_by`) gruppierte Tabelle vorliegt. Jegliche Zusammenfassung, die wir anfordern, wird anhand der Gruppierungsinformation aufgeteilt werden. In dem Beispiel bekommen wir einen Mittelwert für jeden Wert von `interest`. Interessanterweise sehen wir, dass der Mittelwert tendenziell größer wird, je größer `interest` wird.

Alle diese `dplyr`-Befehle geben einen Dataframe zurück, was praktisch ist für weitere Verarbeitung. In diesem Fall heißen die Spalten `interst` und `mean(score)`. Zweiter Name ist nicht so schön, daher ändern wir den wie folgt:

Jetzt können wir auch die Gruppierung nutzen:

```
test_gruppiert <- group_by(stats_test, interest)
summarise(test_gruppiert, mw_pro_gruppe = mean(score, na.rm = TRUE))
```

```
## # A tibble: 7 x 2
##   interest mw_pro_gruppe
##      <int>         <dbl>
## 1        1      28.26667
## 2        2      29.70213
## 3        3      30.80303
## 4        4      29.92683
## 5        5      32.51111
## 6        6      34.00000
## 7       NA      33.08824
```

Nun heißt die zweite Spalte `mw_pro_Gruppe`. `na.rm = TRUE` veranlasst, bei fehlenden Werten trotzdem einen Mittelwert zurückzuliefern (die Zeilen mit fehlenden Werten werden in dem Fall ignoriert).

Grundsätzlich ist die Philosophie der `dplyr`-Befehle: “Mach nur eine Sache, aber die dafür gut”. Entsprechend kann `summarise` nur *Spalten* zusammenfassen, aber keine *Zeilen*.

Merke:

> Mit summarise kann man eine Spalte eines Dataframes zu einem Wert zusammenfassen.

[Aufgaben]Aufgaben[14]

Richtig oder Falsch!?

1. Möchte man aus der Tabelle `stats_test` den Mittelwert für die Spalte `score` berechn
1. `summarise` liefert eine Tabelle, genauer: einen Tibble, zurück.
1. Die Tabelle, die diese Funktion zurückliefert: `summarise(stats_test, mean(score))`,
1. `summarise` lässt zu, dass die zu berechnende Spalte einen Namen vom Nutzer zugewiese
1. `summarise` darf nur verwendet werden, wenn eine Spalte zu einem Wert zusammengefasst

Zeilen zählen mit n und count

Ebenfalls nützlich ist es, Zeilen zu zählen. Im Gegensatz zum Standardbefehl[15] `nrow` versteht der `dyplr`-Befehl nauch Gruppierungen. `n` darf nur innerhalb von `summarise` oder ähnlichen `dplyr`-Befehlen verwendet werden.

```
summarise(stats_test, n())
```

```
##   n()
## 1 306
```

```
summarise(test_gruppiert, n())
```

```
## # A tibble: 7 x 2
##   interest `n()`
##      <int> <int>
## 1        1    30
## 2        2    47
## 3        3    66
## 4        4    41
## 5        5    45
## 6        6     9
## 7       NA    68
```

```
nrow(stats_test)
```

```
## [1] 306
```

Außerhalb von gruppierten Datensätzen ist `nrow` meist praktischer.

Praktischer ist der Befehl `count`, der nichts anderes ist als die Hintereinanderschaltung von `group_by` und `n`. Mit `count` zählen wir die Häufigkeiten nach Gruppen; Gruppen sind hier zumeist die Werte

[14] R, R, R, R, R

[15] Standardbefehl meint, dass die Funktion zum Standardrepertoire von R gehört, also nicht über ein Paket extra geladen werden muss

einer auszuzählenden Variablen (oder mehrerer auszuzählender Variablen). Das macht `count` zu einem wichtigen Helfer bei der Analyse von Häufigkeitsdaten.

```
dplyr::count(stats_test, interest)
```

```
## # A tibble: 7 x 2
##   interest     n
##      <int> <int>
## 1        1    30
## 2        2    47
## 3        3    66
## 4        4    41
## 5        5    45
## 6        6     9
## 7       NA    68
```

```
dplyr::count(stats_test, study_time)
```

```
## # A tibble: 6 x 2
##   study_time     n
##        <int> <int>
## 1          1    31
## 2          2    49
## 3          3    85
## 4          4    56
## 5          5    17
## 6         NA    68
```

```
dplyr::count(stats_test, interest, study_time)
```

```
## # A tibble: 29 x 3
##    interest study_time     n
##       <int>      <int> <int>
##  1        1          1    12
##  2        1          2     7
##  3        1          3     8
##  4        1          4     2
##  5        1          5     1
##  6        2          1     9
##  7        2          2    15
##  8        2          3    16
##  9        2          4     6
## 10        2          5     1
## # ... with 19 more rows
```

Abbildung 7: La trahison des images [Ceci n'est pas une pipe], René Magritte, 1929, © C. Herscovici, Brussels / Artists Rights Society (ARS), New York, http://collections.lacma.org/node/239578

Allgemeiner formuliert lautet die Syntax: `count(df, Spalte1, ...)`, wobei `df` der Dataframe ist und `Spalte1` die erste (es können mehrere sein) auszuzählende Spalte. Gibt man z.B. zwei Spalten an, so wird pro Wert der 1. Spalte die Häufigkeiten der 2. Spalte ausgegeben.

Merke:

> n und count zählen die Anzahl der Zeilen, d.h. die Anzahl der Fälle.

[Aufgaben]Aufgaben[16]

```
Richtig oder Falsch!?

1. Mit `count` kann man Zeilen zählen.
1. `count` ist ähnlich (oder identisch) zu einer Kombination von `group_by` und `n()`.
1. Mit `count` kann man nur nur eine Gruppe beim Zählen berücksichtigen.
1. `count` darf nicht bei nominalskalierten Variablen verwendet werden.
```

Vertiefung

Die Pfeife

Die zweite Idee kann man salopp als "Durchpfeifen" bezeichnen; ikonographisch mit diesem Symbol dargestellt `%>%`. Der Begriff "Durchpfeifen" ist frei vom Englischen "to pipe" übernommen. Das berühmte Bild von René Magritte stand dabei Pate.

Hierbei ist gemeint, einen Datensatz sozusagen auf ein Fließband zu legen und an jedem Arbeitsplatz einen Arbeitsschritt auszuführen. Der springende Punkt ist, dass ein Dataframe als "Rohstoff" eingegeben wird und jeder Arbeitsschritt seinerseits wieder einen Datafram ausgiebt. Damit kann man sehr schön, einen "Flow" an Verarbeitung erreichen, außerdem spart man sich Tipparbeit und die Syntax wird lesbarer. Damit das Durchpfeifen funktioniert, benötigt man Befehle, die als Eingabe einen Dataframe

[16] R, R, F, F

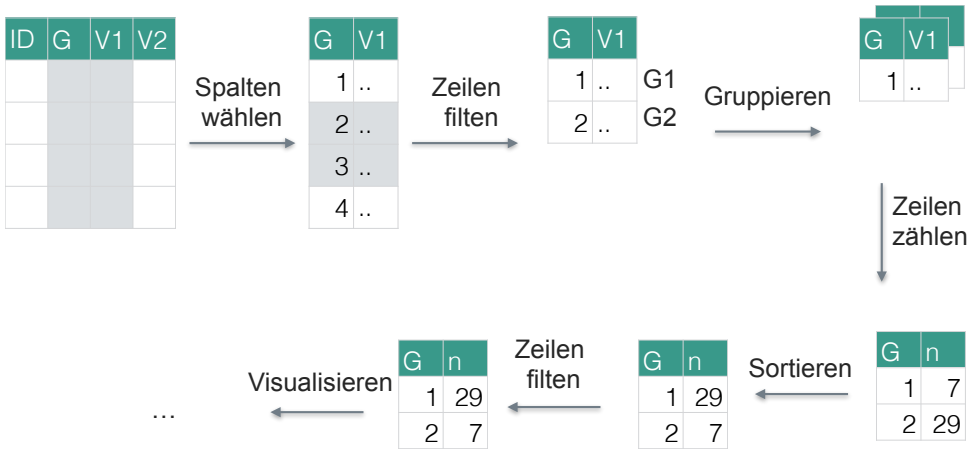


Abbildung 8: Das 'Durchpeifen'

erwarten und wieder einen Dataframe zurückliefern. Das Schaubild verdeutlich beispielhaft eine Abfolge des Durchpfeifens.

Die sog. "Pfeife" (pipe: `%>%`) in Anspielung an das berühmte Bild von René Magritte, verkettet Befehle hintereinander. Das ist praktisch, da es die Syntax vereinfacht. Vergleichen Sie mal diese Syntax

```
filter(summarise(group_by(filter(stats_test, !is.na(score)), interest), mw = mean(score)
```

mit dieser

```
stats_test %>%
  filter(!is.na(score)) %>%
  group_by(interest) %>%
  summarise(mw = mean(score)) %>%
  filter(mw > 30)
```

```
## # A tibble: 4 x 2
##   interest       mw
##      <int>    <dbl>
## 1        3 30.80303
## 2        5 32.51111
## 3        6 34.00000
## 4       NA 33.08824
```

Es ist hilfreich, diese "Pfeifen-Syntax" in deutschen Pseudo-Code zu übersetzen.

```
Nimm die Tabelle "stats_test" UND DANN
filtere alle nicht-fehlenden Werte UND DANN
gruppiere die verbleibenden Werte nach "interest" UND DANN
bilde den Mittelwert (pro Gruppe) für "score" UND DANN
liefere nur die Werte größer als 30 zurück.
```

Die Pfeife zerlegt die "russische Puppe", also ineinander verschachelteten Code, in sequenzielle Schritte

und zwar in der richtigen Reihenfolge (entsprechend der Abarbeitung). Wir müssen den Code nicht mehr von innen nach außen lesen (wie das bei einer mathematischen Formel der Fall ist), sondern können wie bei einem Kochrezept "erstens ..., zweitens .., drittens ..." lesen. Die Pfeife macht die Syntax einfacher. Natürlich hätten wir die verschachtelte Syntax in viele einzelne Befehle zerlegen können und jeweils eine Zwischenergebnis speichern mit dem Zuweisungspfeil `<-` und das Zwischenergebnis dann explizit an den nächsten Befehl weitergeben. Eigentlich macht die Pfeife genau das - nur mit weniger Tipparbeit. Und auch einfacher zu lesen. Flow!

Werte umkodieren und "binnen"

`car::recode`

Manchmal möchte man z.B. negativ gepolte Items umdrehen oder bei kategoriellen Variablen kryptische Bezeichnungen in sprechendere umwandeln (ein Klassiker ist `1` in `maennlich` bzw. `2` in `weiblich` oder umgekehrt, kann sich niemand merken). Hier gibt es eine Reihe praktischer Befehle, z.B. `recode` aus dem Paket `car`. Übrigens: Wenn man explizit angeben möchte, aus welchem Paket ein Befehl stammt (z.B. um Verwechslungen zu vermeiden), gibt man `Paketnamen::Befehlnamen` an. Schauen wir uns ein paar Beispiele zum Umkodieren an.

```
stats_test$score_fac <- car::recode(stats_test$study_time, "5 = 'sehr viel'; 2:4 = 'mitt
stats_test$score_fac <- car::recode(stats_test$study_time, "5 = 'sehr viel'; 2:4 = 'mitt

stats_test$study_time <- car::recode(stats_test$study_time, "5 = 'sehr viel'; 4 = 'wenig

head(stats_test$study_time)
```

```
## [1] sehr viel Hilfe     sehr viel Hilfe     wenig     Hilfe
## Levels: Hilfe sehr viel wenig
```

Der Befehle `recode` ist wirklich sehr prkatisch; mit `:` kann man "von bis" ansprechen (das ginge mit `c()` übrigens auch); `else` für "ansonsten" ist möglich und mit `as.factor.result` kann man entweder einen Faktor oder eine Text-Variable zurückgeliefert bekommen. Der ganze "Wechselterm" steht in Anführungsstrichen (`'`). Einzelne Teile des Wechselterms sind mit einem Strichpunkt (`;`) voneinander getrennt.

Das klassiche Umkodieren von Items aus Fragebögen kann man so anstellen; sagen wir `interest` soll umkodiert werden:

```
stats_test$no_interest <- car::recode(stats_test$interest, "1 = 6; 2 = 5; 3 = 4; 4 = 3;
glimpse(stats_test$no_interest)
```

```
##  num [1:306] 2 4 1 5 1 NA NA 4 2 2 ...
```

Bei dem Wechselterm muss man aufpassen, nichts zu verwechseln; die Zahlen sehen alle ähnlich aus...

Testen kann man den Erfolg des Umpolens mit

```
dplyr::count(stats_test, interest)
```

```
## # A tibble: 7 x 2
##   interest     n
##      <int> <int>
## 1        1    30
## 2        2    47
## 3        3    66
## 4        4    41
## 5        5    45
## 6        6     9
## 7       NA    68
```

```
dplyr::count(stats_test, no_interest)
```

```
## # A tibble: 7 x 2
##   no_interest     n
##         <dbl> <int>
## 1           1     9
## 2           2    45
## 3           3    41
## 4           4    66
## 5           5    47
## 6           6    30
## 7          NA    68
```

Scheint zu passen. Noch praktischer ist, dass man so auch numerische Variablen in Bereiche aufteilen kann ("binnen"):

```
stats_test$Ergebnis <- car::recode(stats_test$score, "1:38 = 'durchgefallen'; else = 'be
```

Natürlich gibt es auch eine Pfeifen komptatible Version, um Variablen umzukodieren bzw. zu binnen: `dplyr::recode`[17]. Die Syntax ist allerdings etwas weniger komfortabel (da strenger), so dass wir an dieser Stelle bei `car::recode` bleiben.

Numerische Werte in Klassen gruppieren mit `cut`

Numerische Werte in Klassen zu gruppieren ("to bin", denglisch: "binnen") kann mit dem Befehl `cut` (and friends) besorgt werden.

Es lassen sich drei typische Anwendungsformen unterscheiden:

Eine numerische Variable ...

1. in k gleich große Klassen grupieren (gleichgroße Intervalle)

[17] https://blog.rstudio.org/2016/06/27/dplyr-0-5-0/

2. so in Klassen gruppieren, dass in jeder Klasse n Beobachtungen sind (gleiche Gruppengrößen)
3. in beliebige Klassen gruppieren

gleichgroße Intervalle

Nehmen wir an, wir möchten die numerische Variable "Körpergröße" in drei Gruppen einteilen: "klein", "mittel" und "groß". Der Range von Körpergröße soll gleichmäßig auf die drei Gruppen aufgeteilt werden, d.h. der Range (Interval) der drei Gruppen soll gleich groß sein. Dazu kann man `cut_interval` aus `ggplot2` nehmen [^d.h. `ggplot2` muss geladen sein; wenn man `tidyverse` lädt, wird `ggplot2` automatisch auch geladen].

```
wo_men <- read_csv("./data/wo_men.csv")
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   X1 = col_integer(),
##   time = col_character(),
##   sex = col_character(),
##   height = col_double(),
##   shoe_size = col_double()
## )
```

```
wo_men %>%
  filter(height > 150, height < 220) -> wo_men2


temp <- cut_interval(x = wo_men2$height, n = 3)


levels(temp)
```

```
## [1] "[155,172]" "(172,189]" "(189,206]"
```

`cut_interval` liefert eine Variabel vom Typ `factor` zurück.

gleiche Gruppengrößen

```
temp <- cut_number(wo_men2$height, n = 2)
str(temp)
```

```
##  Factor w/ 2 levels "[155,169]","(169,206]": 1 2 2 2 2 1 1 2 1 2 ...
```

Mit `cut_number` (aus ggplot2) kann man einen Vektor in `n` Gruppen mit (etwa) gleich viel Observationen einteilen.

> Teilt man einen Vektor in zwei gleich große Gruppen, so entspricht das einer Aufteilung am Median (Median-Split).

In beliebige Klassen gruppieren

```
wo_men$groesse_gruppe <- cut(wo_men$height,
                             breaks = c(-Inf, 100, 150, 170, 200, 230, Inf))

count(wo_men, groesse_gruppe)
```

```
## # A tibble: 1 x 1
##   `1.n`
##   <int>
## 1   101
```

`cut` ist im Standard-R (Paket "base") enthalten. Mit `breaks` gibt man die Intervallgrenzen an. Zu beachten ist, dass man eine Unter- bzw. Obergrenze angeben muss. D.h. der kleinste Wert wird nicht automatisch als unterste Intervallgrenze herangezogen.

Verweise

- Eine schöne Demonstration der Mächtigkeit von `dplyr` findet sich hier[18].

Hinweis

Erstellt von Sebastian Sauer

Anhang 3: Daten visualisieren mit ggplot

In diesem Kapitel werden folgende Pakete benötigt::

```
library(tidyverse)  # Zum Plotten
```

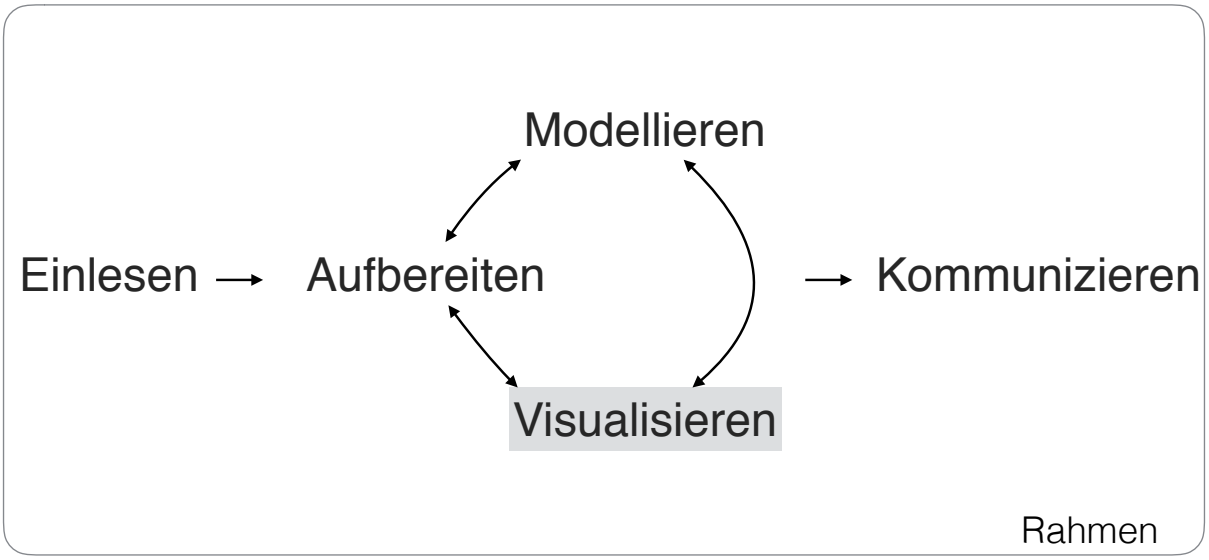
```
## Loading tidyverse: ggplot2
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
## Loading tidyverse: purrr
## Loading tidyverse: dplyr
## Conflicts with tidy packages ----------------------------------------------
## filter(): dplyr, stats
## lag():    dplyr, stats
```

[18]: http://bit.ly/2kX9lvC

```
library(car)   # Umkodieren
```

```
##
## Attaching package: 'car'
## The following object is masked from 'package:dplyr':
##
##     recode
## The following object is masked from 'package:purrr':
##
##     some
```

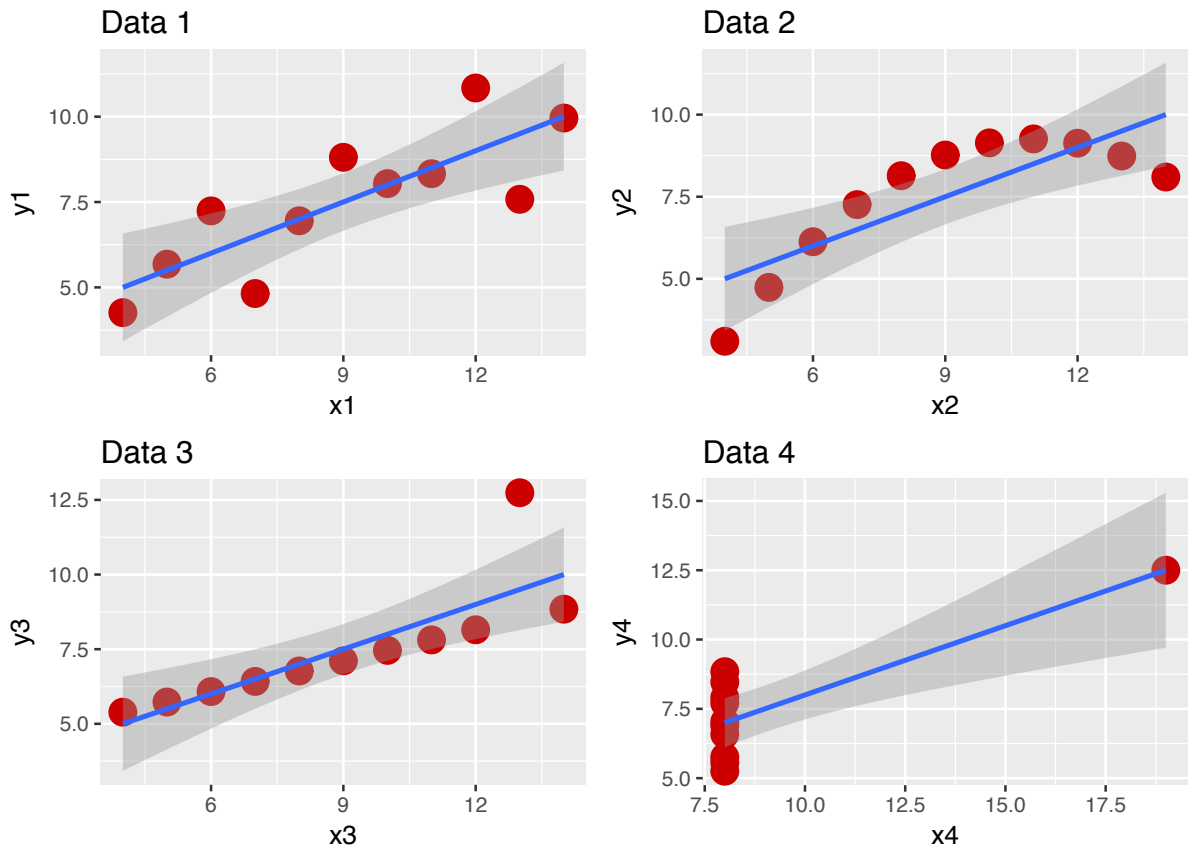
```
library(knitr)   # HTML-Tabellen
```



Ein Bild sagt mehr als 1000 Worte

Ein Bild sagt bekanntlich mehr als 1000 Worte. Schauen wir uns zur Verdeutlichung das berühmte Beispiel von Anscombe[19] an. Es geht hier um vier Datensätze mit zwei Variablen (Spalten; X und Y). Offenbar sind die Datensätze praktisch identisch: Alle X haben den gleichen Mittelwert und die gleiche Varianz; dasselbe gilt für die Y. Die Korrelation zwischen X und Y ist in allen vier Datensätzen gleich. Allerdings erzählt eine Visualisierung der vier Datensätze eine ganz andere Geschichte.

[19] https://de.wikipedia.org/wiki/Anscombe-Quartett



Offenbar "passieren" in den vier Datensätzen gänzlich unterschiedliche Dinge. Dies haben die Statistiken nicht aufgedeckt; erst die Visualisierung erhellte uns... Kurz: Die Visualisierung ist ein unverzichtbares Werkzeug, um zu verstehen, was in einem Datensatz (und damit in der zugrunde liegenden "Natur") passiert.

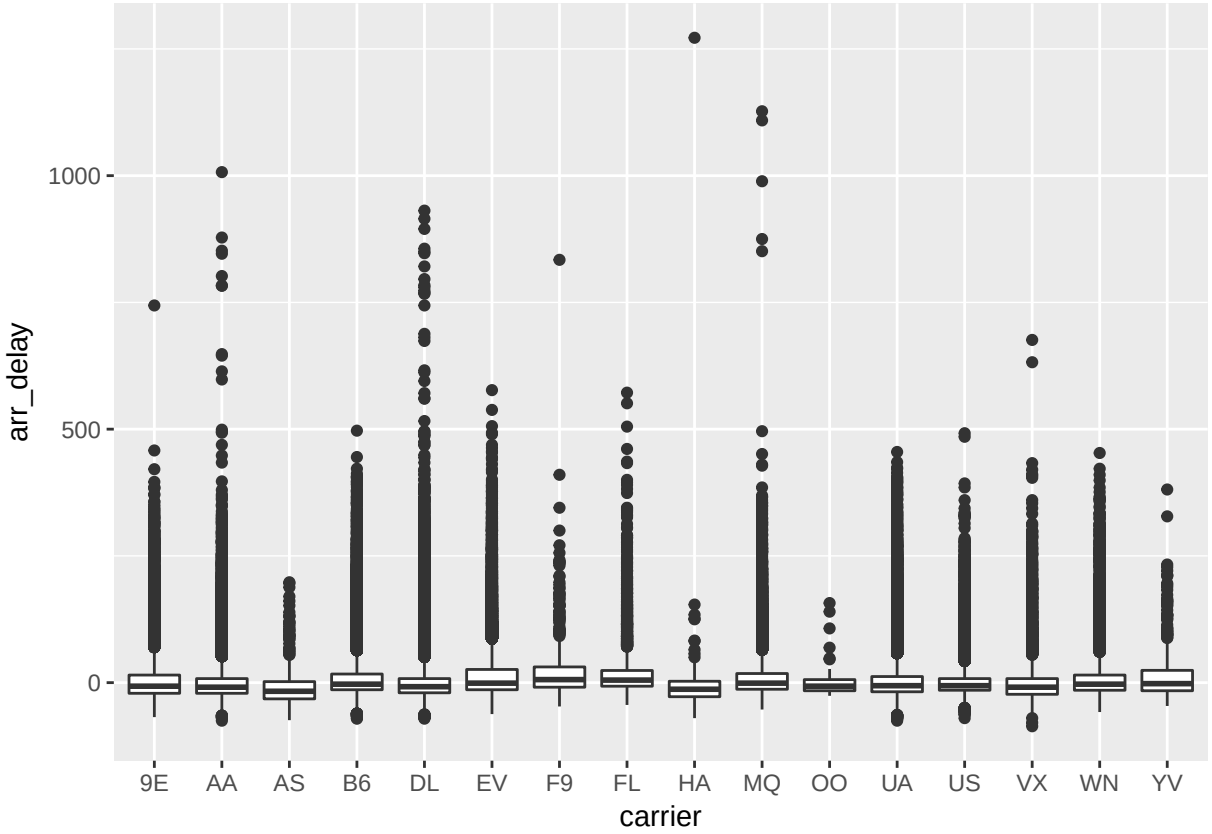
Es gibt viele Möglichkeiten, Daten zu visualisieren (in R). Wir werden uns hier auf einen Weg bzw. ein Paket konzentrieren, der komfortabel, aber mächtig ist und gut zum Prinzip des Durchpfeifens passt: `ggplot2`[^"gg" steht für "grammer of graphics" nach einem Buch von Wilkinson[-@wilkinson2006grammar]; "plot" steht für "to plot", also ein Diagramm erstellen ("plotten"); vgl. https://en.wikipedia.org/wiki/Ggplot2].

Laden wir dazu den Datensatz `nycflights::flights`.

```
data(flights, package = "nycflights13")
```

```
qplot(x = carrier, y = arr_delay, geom = "boxplot", data = flights)
```

```
## Warning: Removed 9430 rows containing non-finite values (stat_boxplot).
```



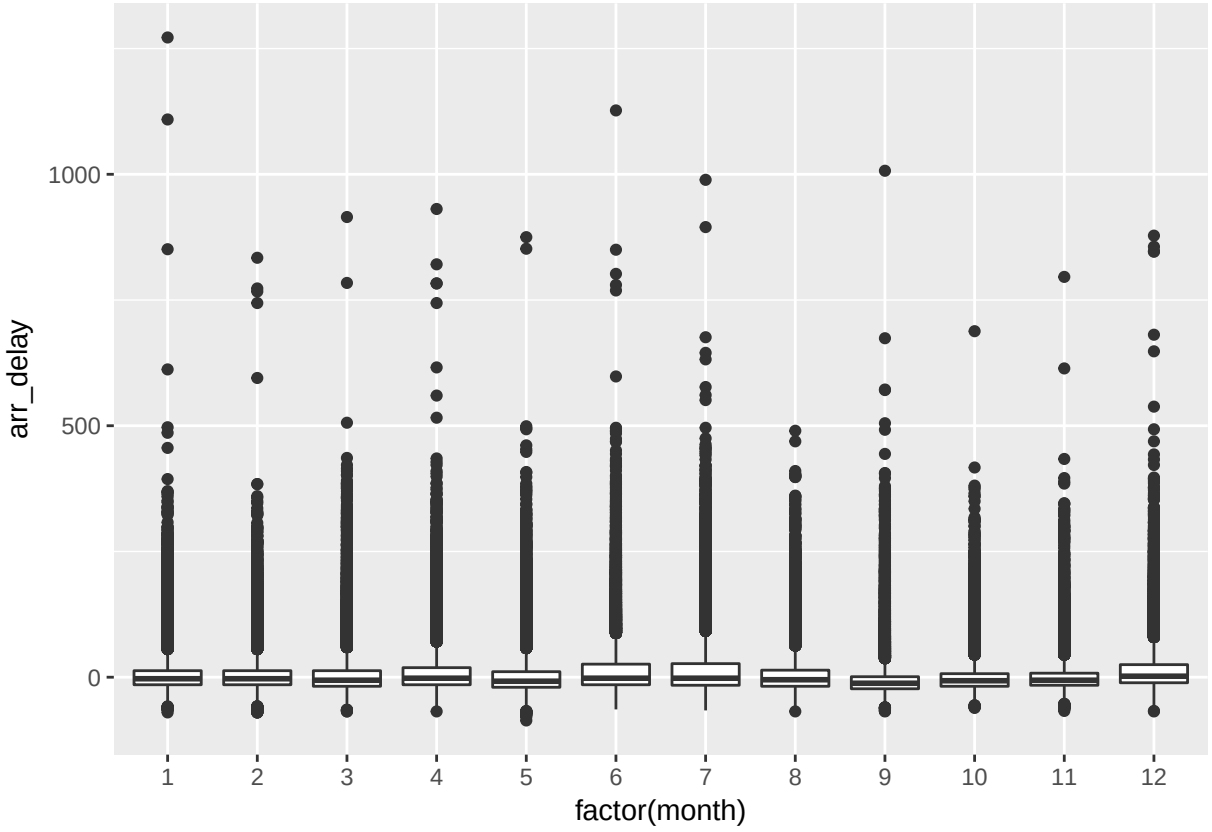
Schauen wir uns den Befehl `qplot` etwas näher an. Wie ist er aufgebaut?

`qplot`: Erstelle schnell (q wie quick in `qplot`) mal einen Plot (engl. "plot": Diagramm).
`x`: Der X-Achse soll die Variable "carrier" zugeordnet werden.
`y`: Der Y-Achse soll die Variable "arr_dely" zugeorndet werden.
`geom`: ("geometriches Objekt") Gemalt werden soll ein Boxplot, nicht etwa Punkte, Linien oder sonstiges.
`data`: Als Datensatz bitte `flights` verwenden.

Offenbar gibt es viele Extremwerte, was die Verspätung betrifft. Das erscheint mir nicht unplausibel (Schneesturm im Winter, Flugzeug verschwunden...). Vor dem Hintergrund der Extremwerte erscheinen die mittleren Verspätungen (Mediane) in den Boxplots als ähnlich. Vielleicht ist der Unterschied zwischen den Monaten ausgeprägter?

```
qplot(x = factor(month), y = arr_delay, geom = "boxplot", data = flights)
```

```
## Warning: Removed 9430 rows containing non-finite values (stat_boxplot).
```



Kaum Unterschied; das spricht gegen die Schneesturm-Idee als Grund für Verspätung. Aber schauen wir uns zuerst die Syntax von `qplot` näher an. "q" in `qplot` steht für "quick". Tatsächlich hat `qplot` einen großen Bruder, `ggplot`[20], der deutlich mehr Funktionen aufweist - und daher auch die umfangreichere (=komplexere) Syntax. Fangen wir mit `qplot` an.

Diese Syntax des letzten Beispiels ist recht einfach, nämlich:

```
qplot (x = X_Achse, y = Y_Achse, data = mein_dataframe, geom = "ein_geom")
```

Wir definieren mit `x`, welche Variable der X-Achse des Diagramms zugewiesen werden soll, z.B. `month`; analog mit Y-Achse. Mit `data` sagen wir, in welchem Dataframe die Spalten "wohnen" und als "geom" ist die Art des statistischen "*geom*etrischen Objects" gemeint, also Punkte, Linien, Boxplots, Balken...

Häufige Arten von Diagrammen

Unter den vielen Arten von Diagrammen und vielen Arten, diese zu klassifizieren greifen wir uns ein paar häufige Diagramme heraus und schauen uns diese der Reihe nach an.

Eine kontinuierliche Variable

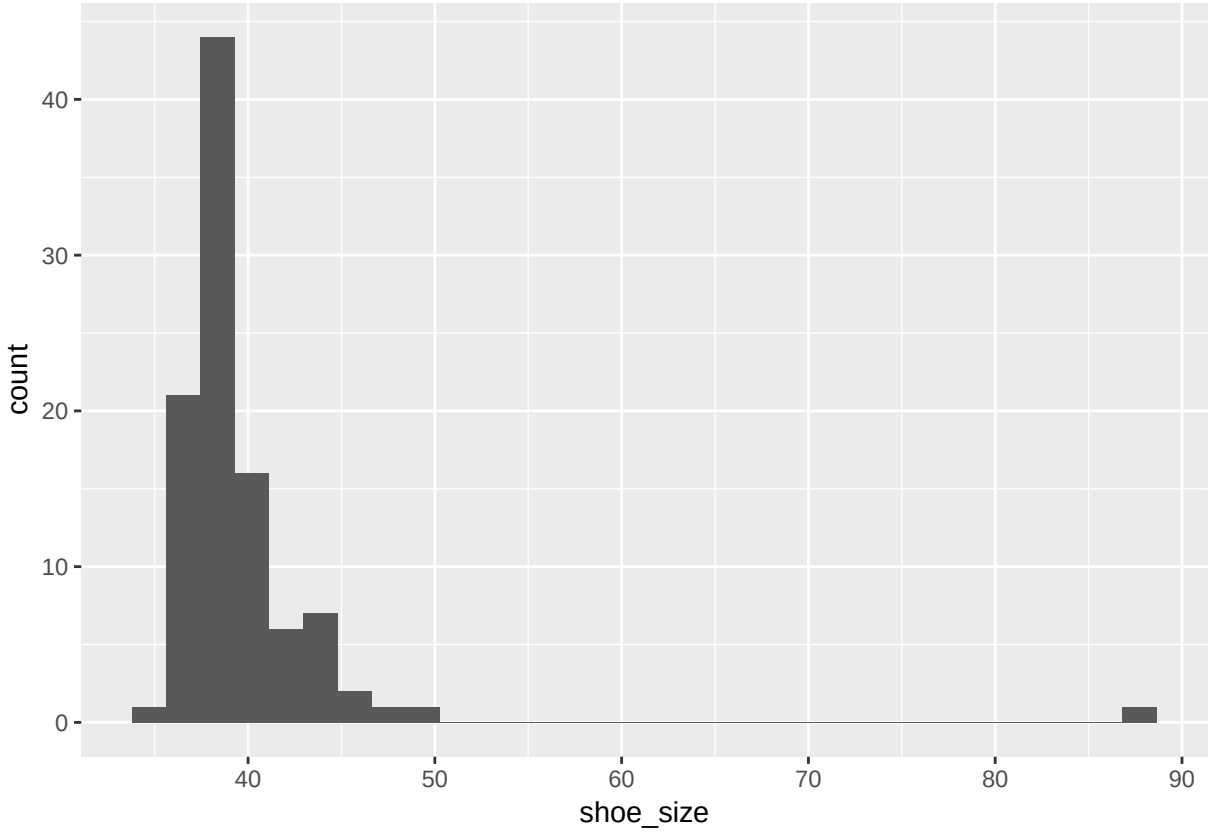
Schauen wir uns die Verteilung der Schuhgrößen von Studierenden an.

[20] Achtung: Nicht `qqplot`, nicht `ggplot2`, nicht `gplot`...

```
wo_men <- read.csv("./data/wo_men.csv")

qplot(x = shoe_size, data = wo_men)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

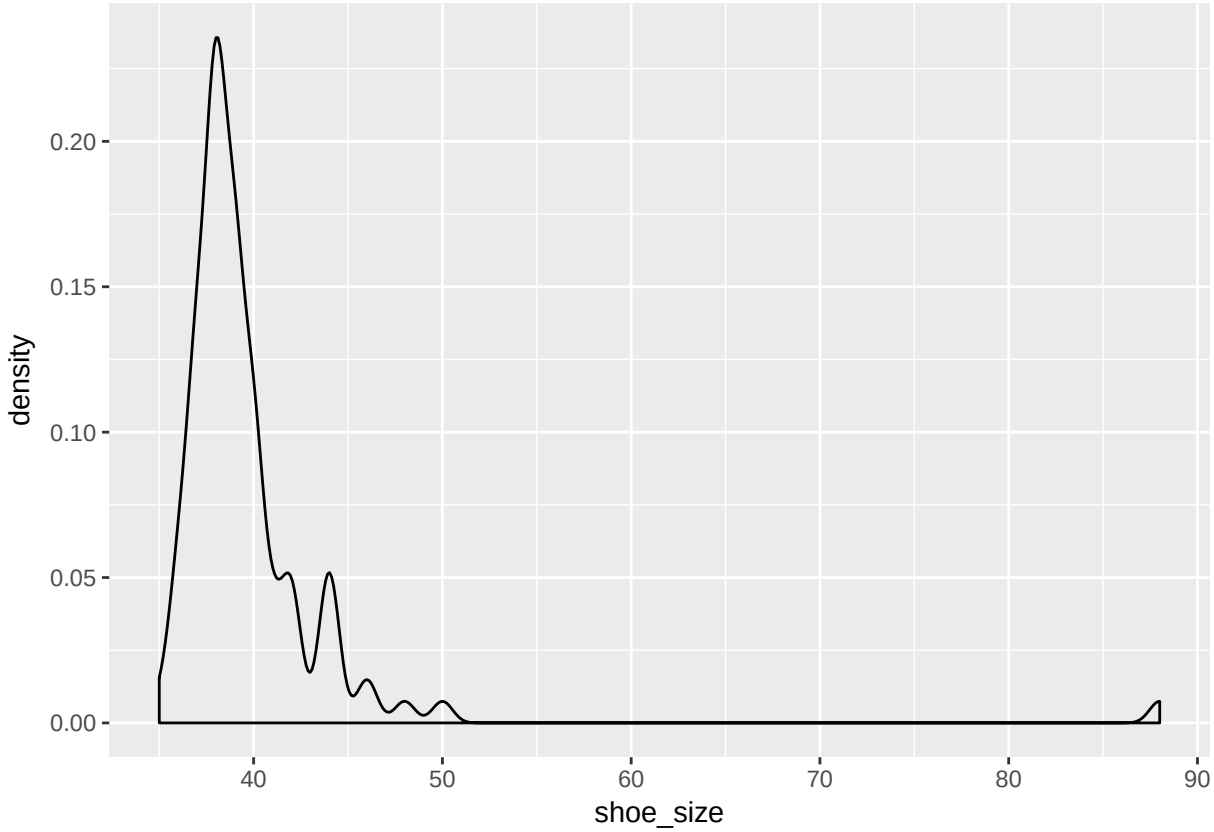
```
## Warning: Removed 1 rows containing non-finite values (stat_bin).
```



Weisen wir nur der X-Achse (aber nicht der Y-Achse) eine kontinuierliche Variable zu, so wählt `ggplot2` automatisch als Geom automatisch ein Histogramm; wir müssen daher nicht explizieren, dass wir ein Histogramm als Geom wünschen (aber wir könnten es hinzufügen). Alternativ wäre ein Dichtediagramm hier von Interesse:

```
# qplot(x = shoe_size, data = wo_men)  wie oben

qplot(x = shoe_size, data = wo_men, geom = "density")
```

```
## Warning: Removed 1 rows containing non-finite values (stat_density).
```



Was man sich merken muss, ist, dass hier nur das Geom mit Anführungsstrichen zu benennen ist, die übrigen Parameter *ohne*.

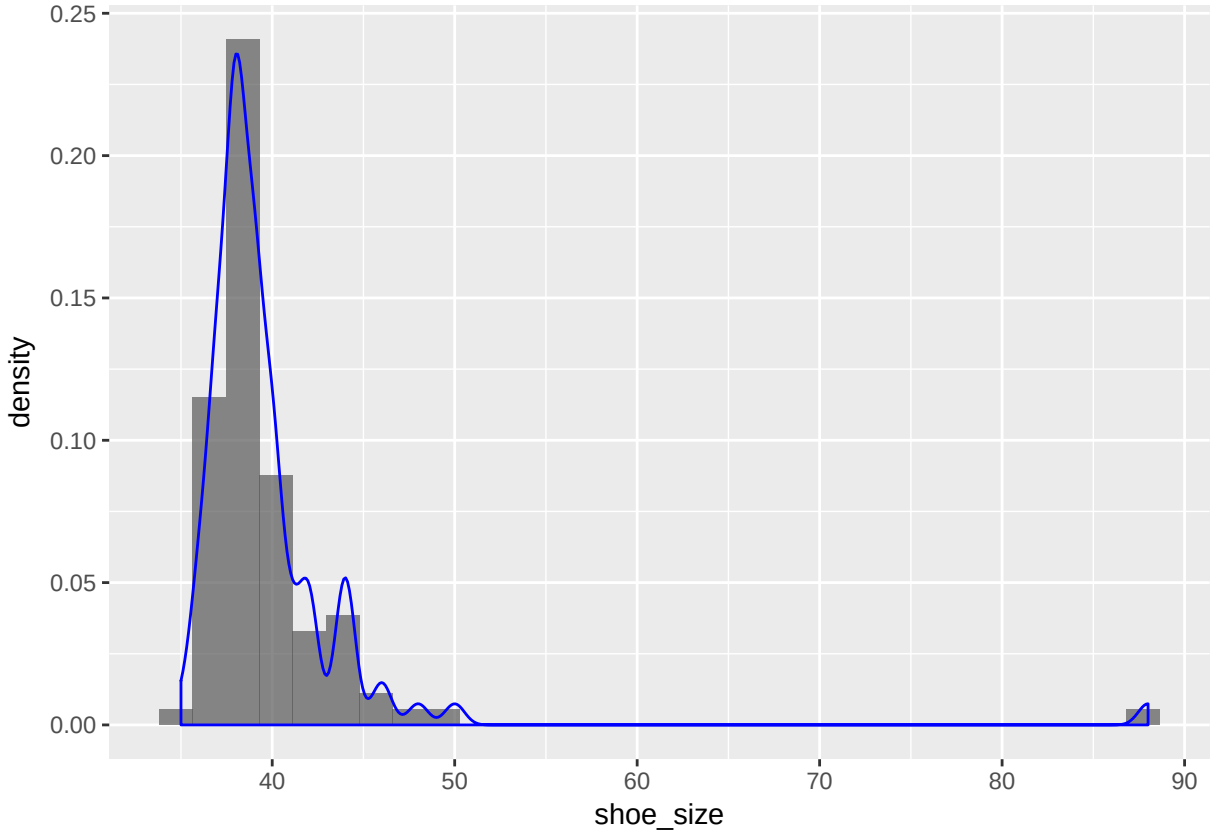
Vielleicht wäre es noch schön, beide Geome zu kombinieren in einem Diagramm. Das ist etwas komplizierter; wir müssen zum großen Bruder `ggplot` umsteigen, da `qplot` nicht diese Funktionen anbietet.

```
ggplot(data = wo_men) +
  aes(x = shoe_size) +
  geom_histogram(aes(y = ..density..), alpha = .7) +
  geom_density(color = "blue")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 1 rows containing non-finite values (stat_bin).
```

```
## Warning: Removed 1 rows containing non-finite values (stat_density).
```

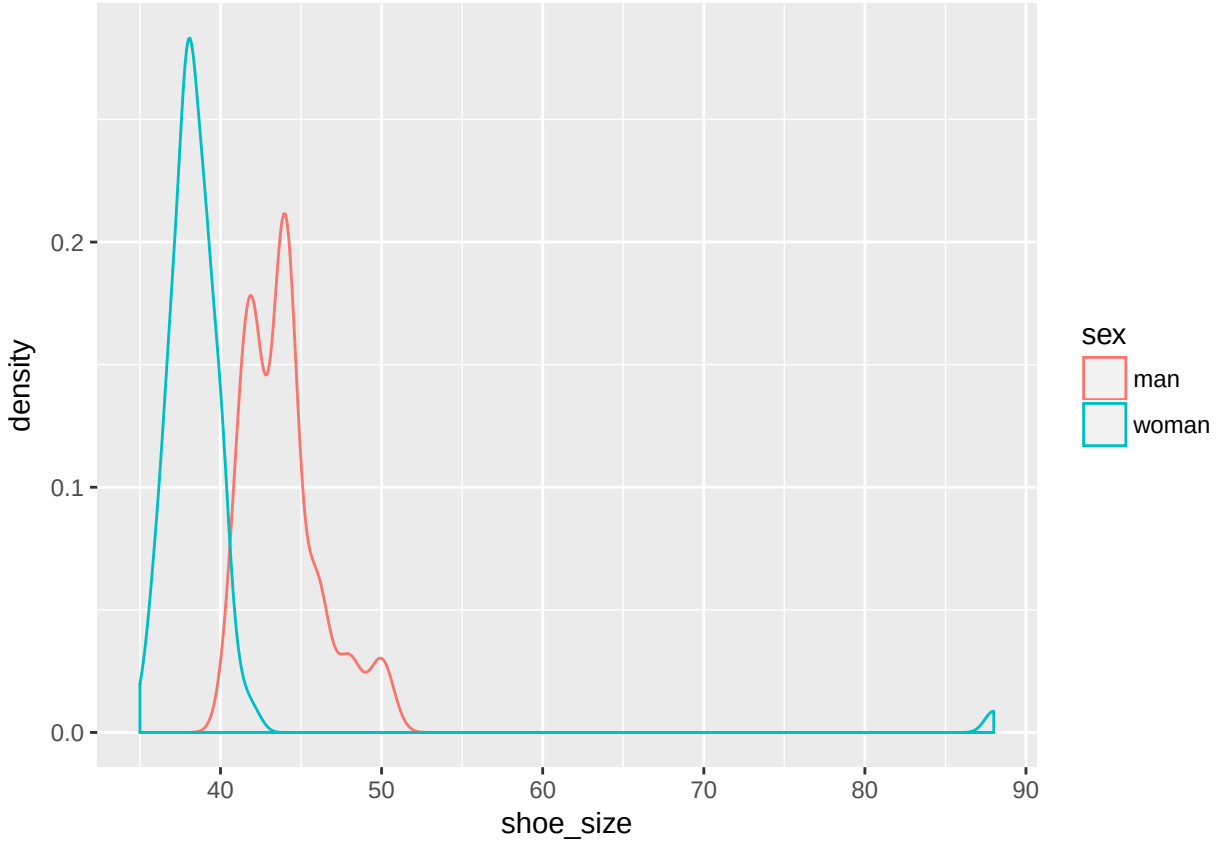


Zuerst haben wir mit dem Parameter `data` den Dataframe benannt. `aes` definiert, welche Variablen welchen Achsen (oder auch z.B. Füllfarben) zugewiesen werden. Hier sagen wir, dass die Schuhgröße auf X-Achse stehen soll. Das +-Zeichen trennt die einzelnen Bestandteile des `ggplot`-Aufrufs voneinander. Als nächstes sagen wir, dass wir gerne ein Histogram hätten: `geom_histogram`. Dabei soll aber nicht wie gewöhnlich auf der X-Achse die Häufigkeit stehen, sondern die Dichte. `ggplot` berechnet selbständig die Dichte und nennt diese Variable `..density..`; die vielen Punkte sollen wohl klar machen, dass es sich nicht um eine "normale" Variable aus dem eigenen Datenframe handelt, sondern um eine "interne" Variable von `ggplot` - die wir aber nichtsdestotrotz verwenden können. `alpha` bestimmt die "Durchsichtigkeit" eines Geoms; spielen Sie mal etwas damit herum. Schließlich malen wir noch ein blaues Dichtediagramm *über* das Histogramm.

Wünsche sind ein Fass ohne Boden... Wäre es nicht schön, ein Diagramm für Männer und eines für Frauen zu haben, um die Verteilungen vergleichen zu können?

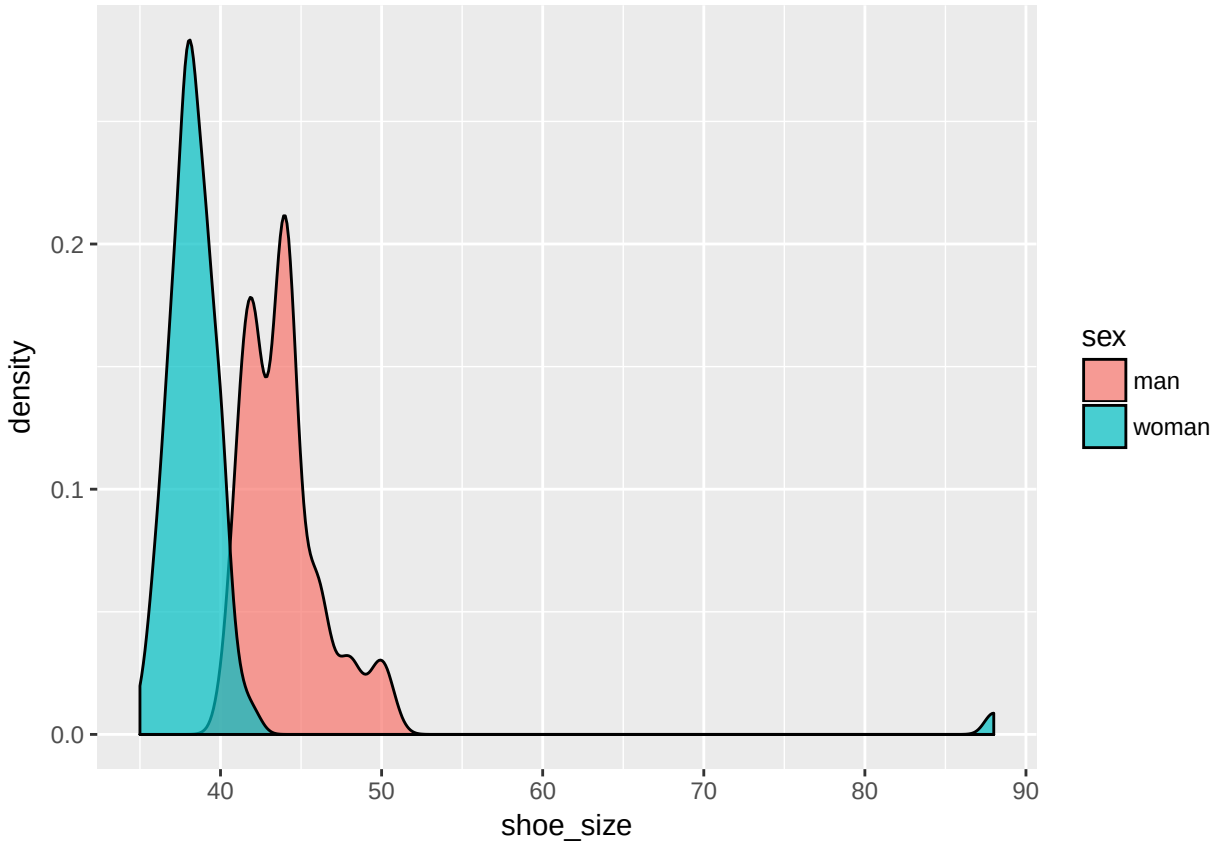
```
qplot(x = shoe_size, data = wo_men, geom = "density", color = sex)
```

```
## Warning: Removed 1 rows containing non-finite values (stat_density).
```



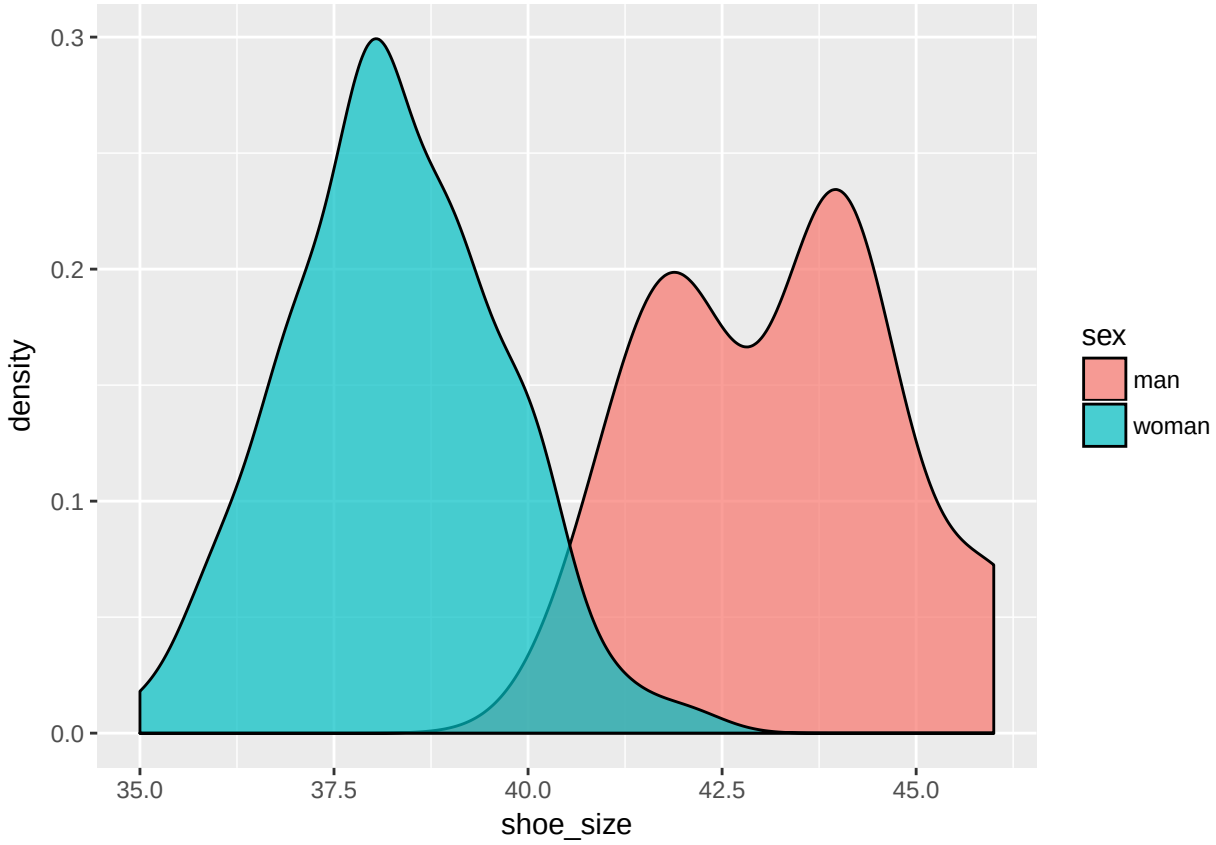

```
qplot(x = shoe_size, data = wo_men, geom = "density", fill = sex, alpha = I(.7))
```

```
## Warning: Removed 1 rows containing non-finite values (stat_density).
```



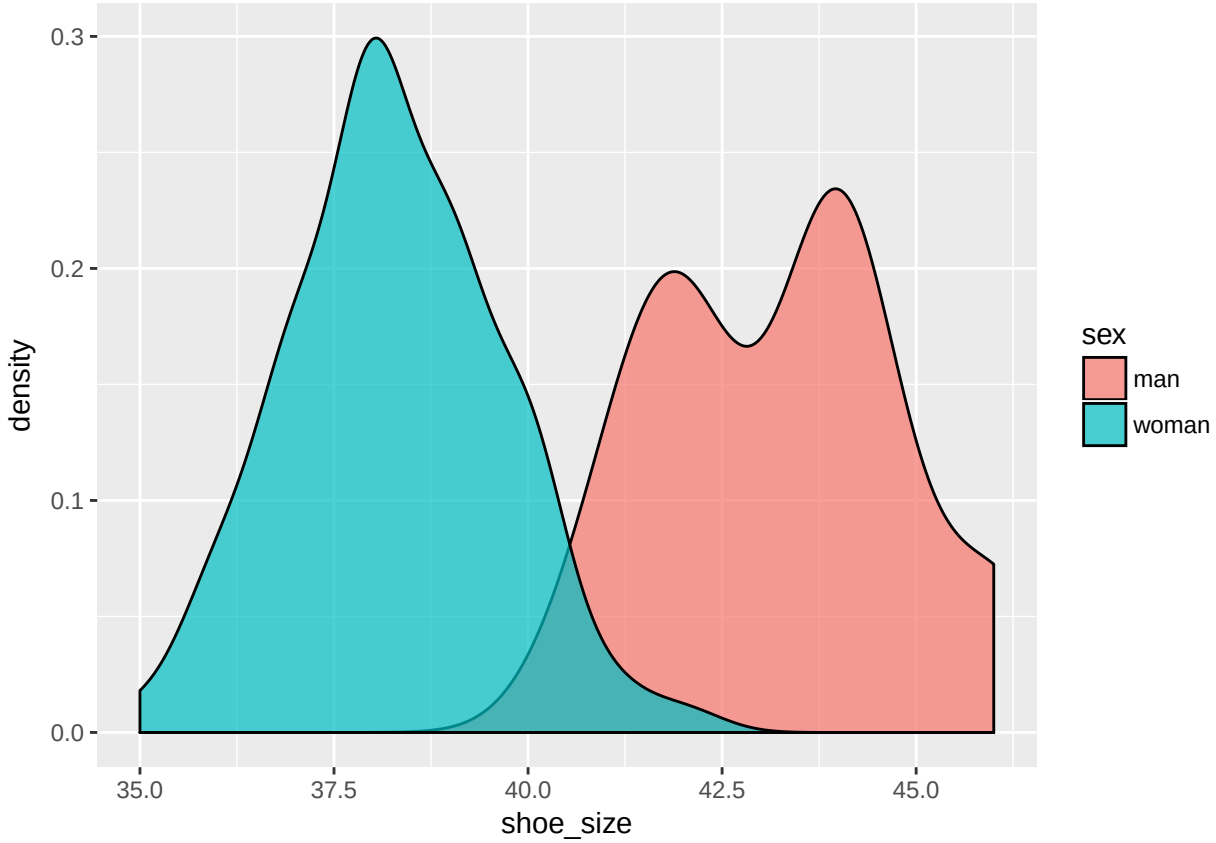
Hier sollten vielleicht noch die Extremwerte entfernt werden, um den Blick auf das Gros der Werte nicht zu verstellen:

```
wo_men %>%
  filter(shoe_size <= 47) -> wo_men2


qplot(x = shoe_size, data = wo_men2, geom = "density", fill = sex, alpha = I(.7))
```



Besser. Man kann das Durchpfeifen auch bis zu `qplot` weiterführen:

```
wo_men %>%
  filter(shoe_size <= 47) %>%
  qplot(x = shoe_size, data = ., geom = "density", fill = sex, alpha = I(.7))
```



Die Pfeife versucht im Standard, das Endprodukt des letzten Arbeitsschritts an den *ersten* Parameter des nächsten Befehls weiterzugeben. Ein kurzer Blick in die Hilfe von `qplot` zeigt, dass der erste Parameter nicht `data` ist, sondern `x`. Daher müssen wir explizit sagen, an welchen Parameter wir das Endprodukt des letzen Arbeitsschritts geben wollen. Netterweise müssen wir dafür nicht viel tippen: Mit einem schlichten Punkt `.` können wir sagen "nimm den Dataframe, so wie er vom letzten Arbeitsschritt ausgegeben wurde".

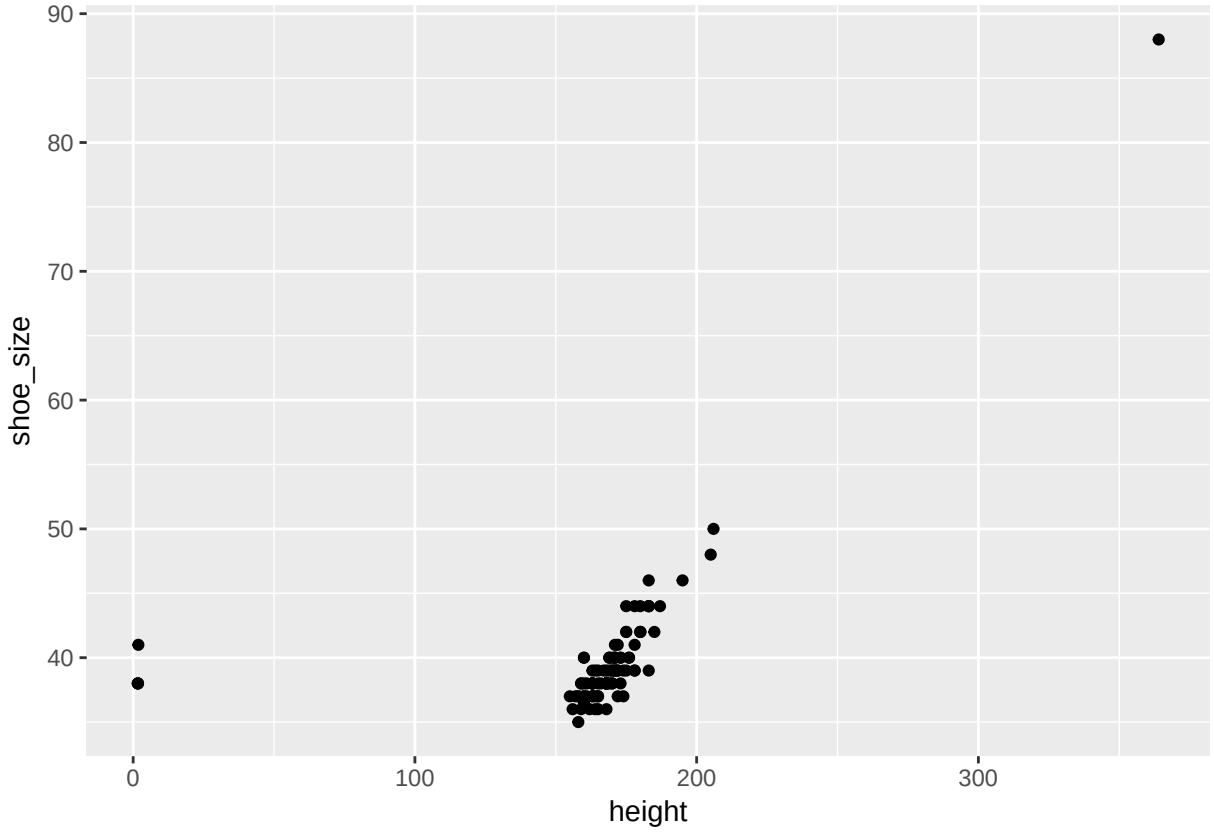
Mit `fill = sex` sagen wir `qplot`, dass er für Männer und Frauen jeweils ein Dichtediagramm erzeugen soll; jedem Dichtediagramm wird dabei eine Farbe zugewiesen (die uns `ggplot2` im Standard voraussucht). Mit anderen Worten: Die Werte von `sex` werden der Füllfarbe der Histogramme zugeordnet. Anstelle der Füllfarbe hätten wir auch die Linienfarbe verwenden können; die Syntax wäre dann: `color = sex`.

Zwei kontinuierliche Variablen

Ein Streudiagramm ist die klassische Art, zwei metrische Variablen darzustellen. Das ist mit `qplot` einfach:

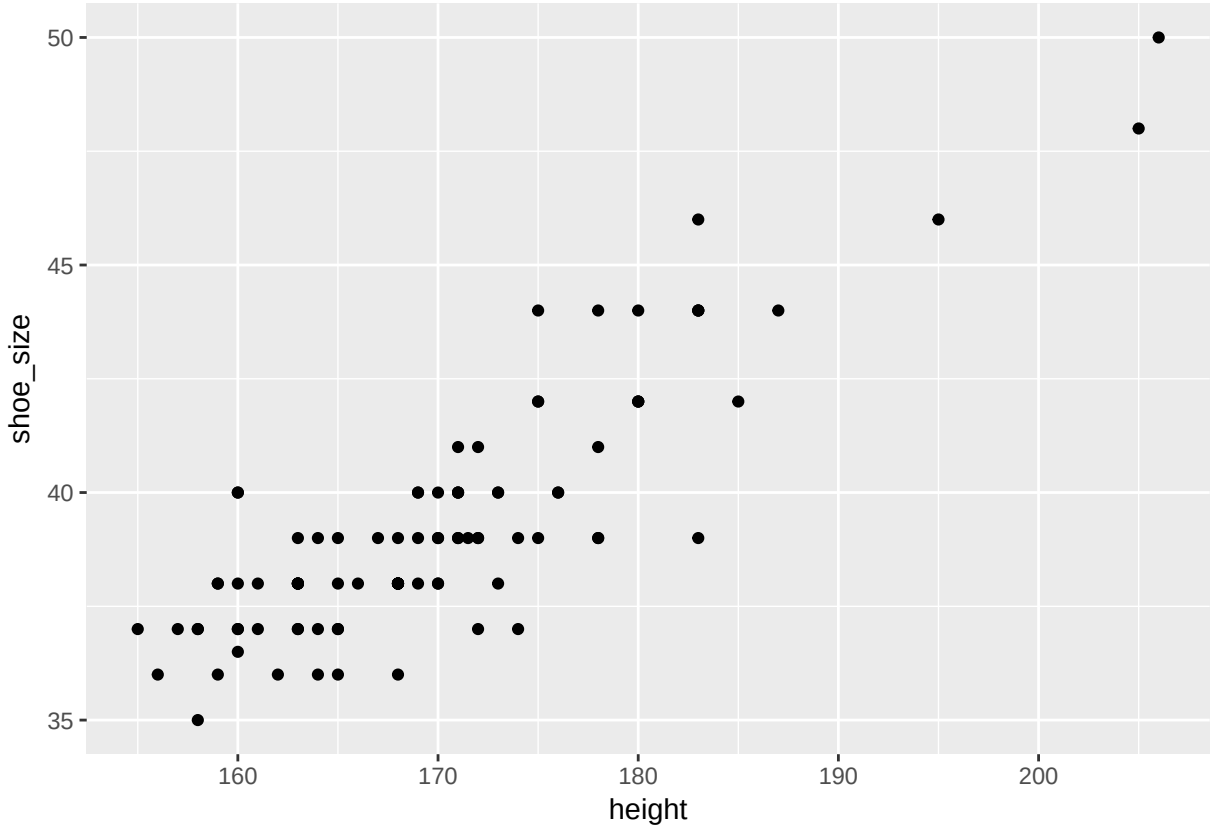
```
qplot(x = height, y = shoe_size, data = wo_men)
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```



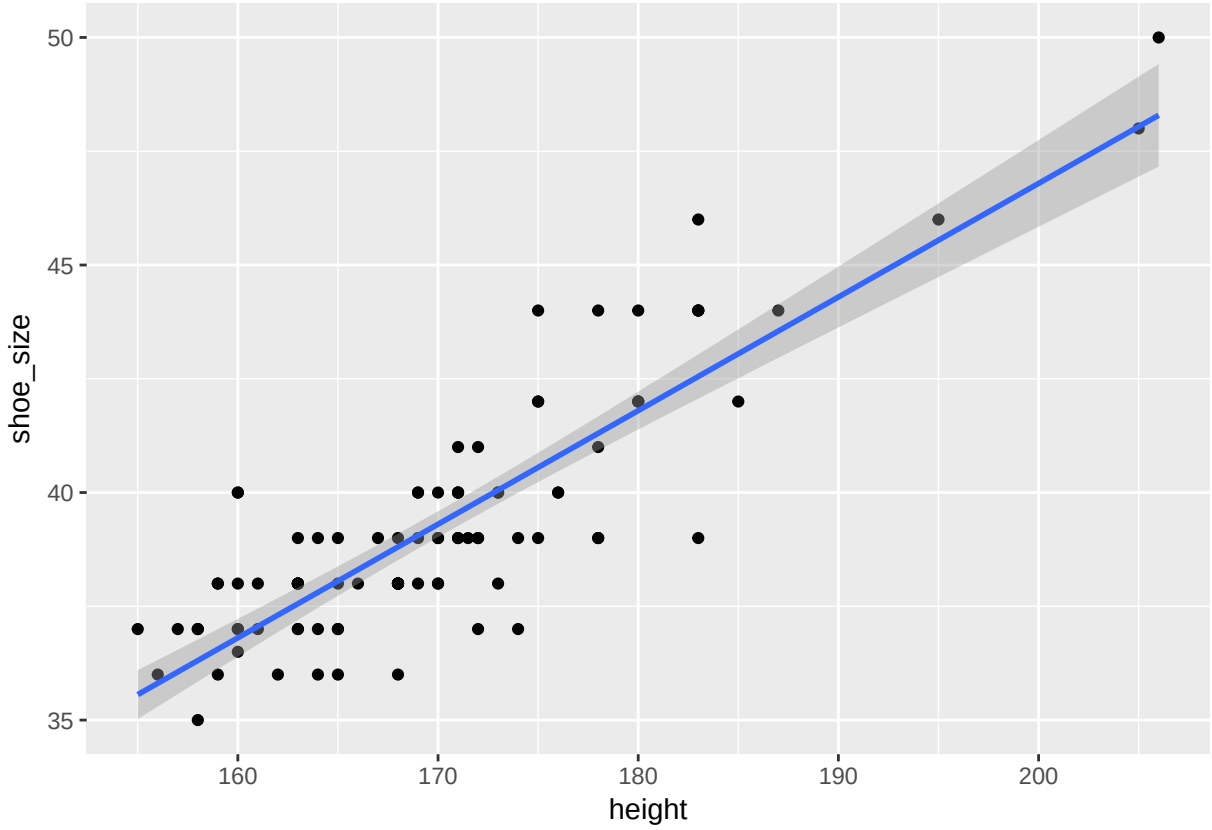
Wir weisen wieder der X-Achse und der Y-Achse eine Variable zu; handelt es sich in beiden Fällen um Zahlen, so wählt `ggplot2` automatisch ein Streudiagramm - d.h. Punkte als Geom (`geom = 'point'`). Wir sollten aber noch die Extremwerte herausnehmen:

```
wo_men %>%
  filter(height > 150, height < 210, shoe_size < 55) %>%
  qplot(x = height, y = shoe_size, data = .)
```



Der Trend ist deutlich erkennbar: Je größer die Person, desto länger die Füß´. Zeichnen wir noch eine Trendgerade ein.

```
wo_men %>%
  filter(height > 150, height < 210, shoe_size < 55) %>%
  qplot(x = height, y = shoe_size, data = .) +
  geom_smooth(method = "lm")
```



Synonym könnten wir auch schreiben:

```
wo_men %>%
  filter(height > 150, height < 210, shoe_size < 55) %>%
  ggplot() +
  aes(x = height, y = shoe_size) +
  geom_point() +
  geom_smooth(method = "lm")
```

Da `ggplot` als *ersten* Parameter die Daten erwartet, kann die Pfeife hier problemlos durchgereicht werden. *Innerhalb* eines `ggplot`-Aufrufs werden die einzelne Teile durch ein Pluszeichen `+` voneinander getrennt. Nachdem wir den Dataframe benannt haben, definieren wir die Zuweisung der Variablen zu den Achsen mit `aes` ("aes" wie "aesthetics", also das "Sichtbare" eines Diagramms, die Achsen etc., werden definiert). Ein "Smooth-Geom" ist eine Linie, die sich schön an die Punkte anschmiegt, in diesem Falls als Gerade (lineares Modell, `lm`).

Bei sehr großen Datensätze, sind Punkte unpraktisch, da sie sich überdecken ("overplotting"). Ein Abhilfe ist es, die Punkte nur "schwach" zu färben. Dazu stellt man die "Füllstärke" der Punkte über `alpha` ein: `geom_point(alpha = 1/100)`. Um einen passablen Alpha-Wert zu finden, bedarf es häufig etwas Probierens. Zu beachten ist, dass es mitunter recht lange dauert, wenn `ggplot` viele (>100.000) Punkte malen soll.

Bei noch größeren Datenmengen bietet sich an, den Scatterplot als "Schachbrett" aufzufassen, und das Raster einzufärben, je nach Anzahl der Punkte pro Schachfeld; zwei Geome dafür sind `geom_hex()` und

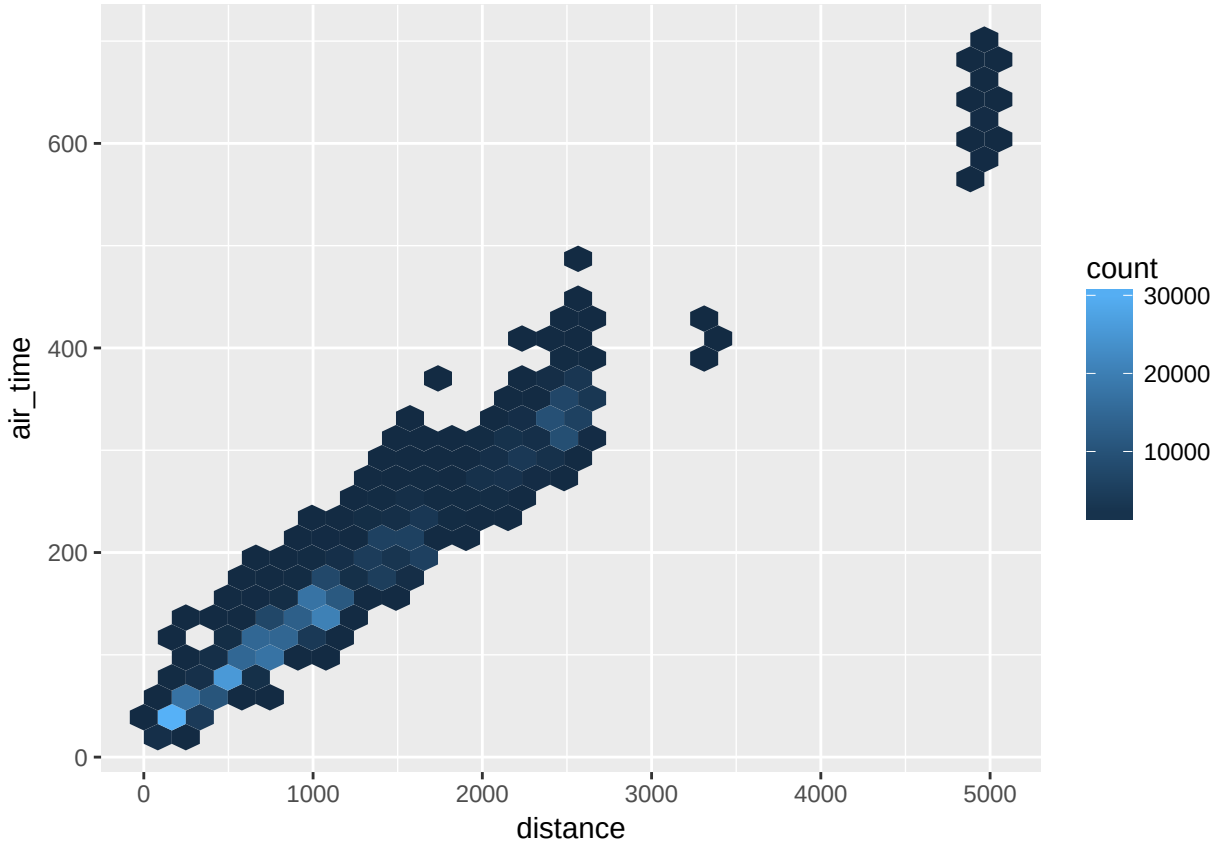
geom_bin2d().

```
data(flights, package = "nycflights13")
nrow(flights)  # groß!
```

```
## [1] 336776
```

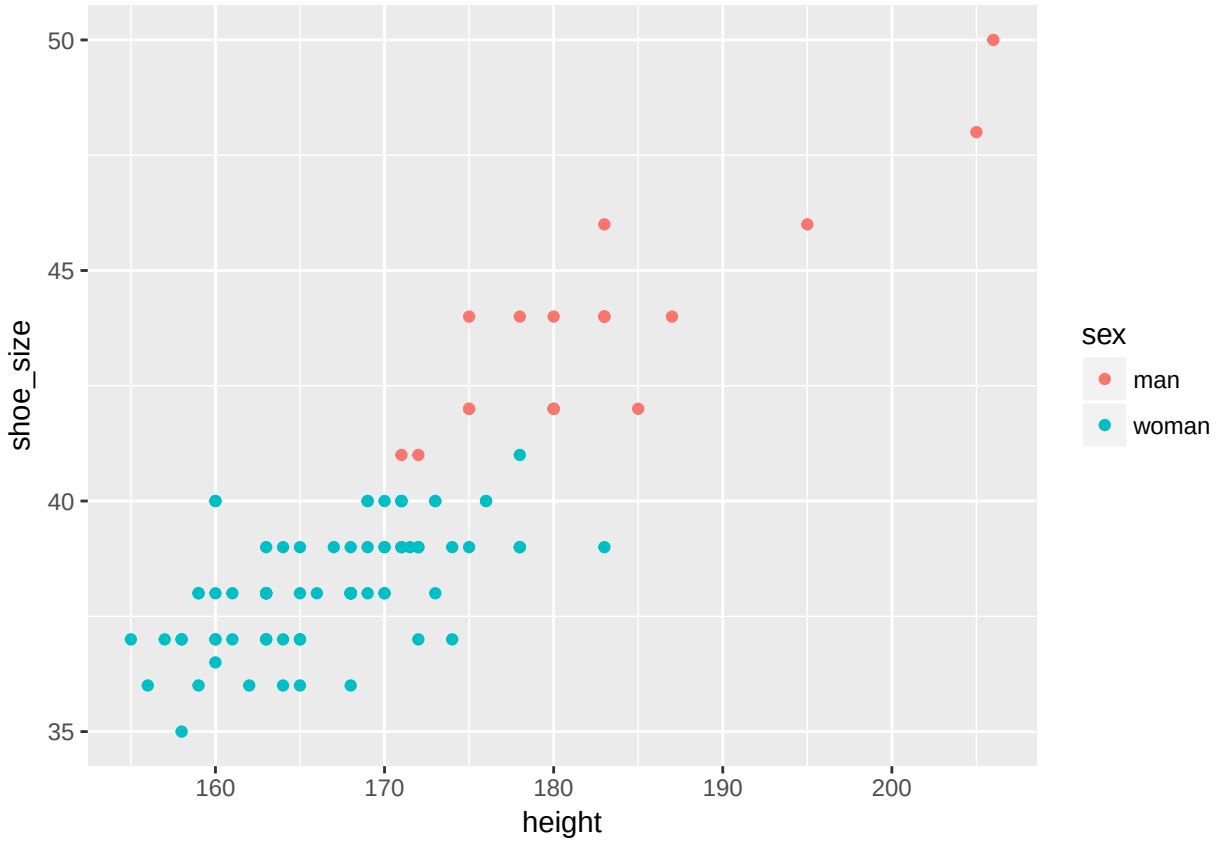
```
ggplot(flights) +
  aes(x = distance, y = air_time) +
  geom_hex()
```

```
## Warning: Removed 9430 rows containing non-finite values (stat_binhex).
```



Wenn man dies verdaut hat, wächst der Hunger nach einer Aufteilung in Gruppen.

```
wo_men %>%
  filter(height > 150, height < 210, shoe_size < 55) %>%
  qplot(x = height, y = shoe_size, color = sex, data = .)
```

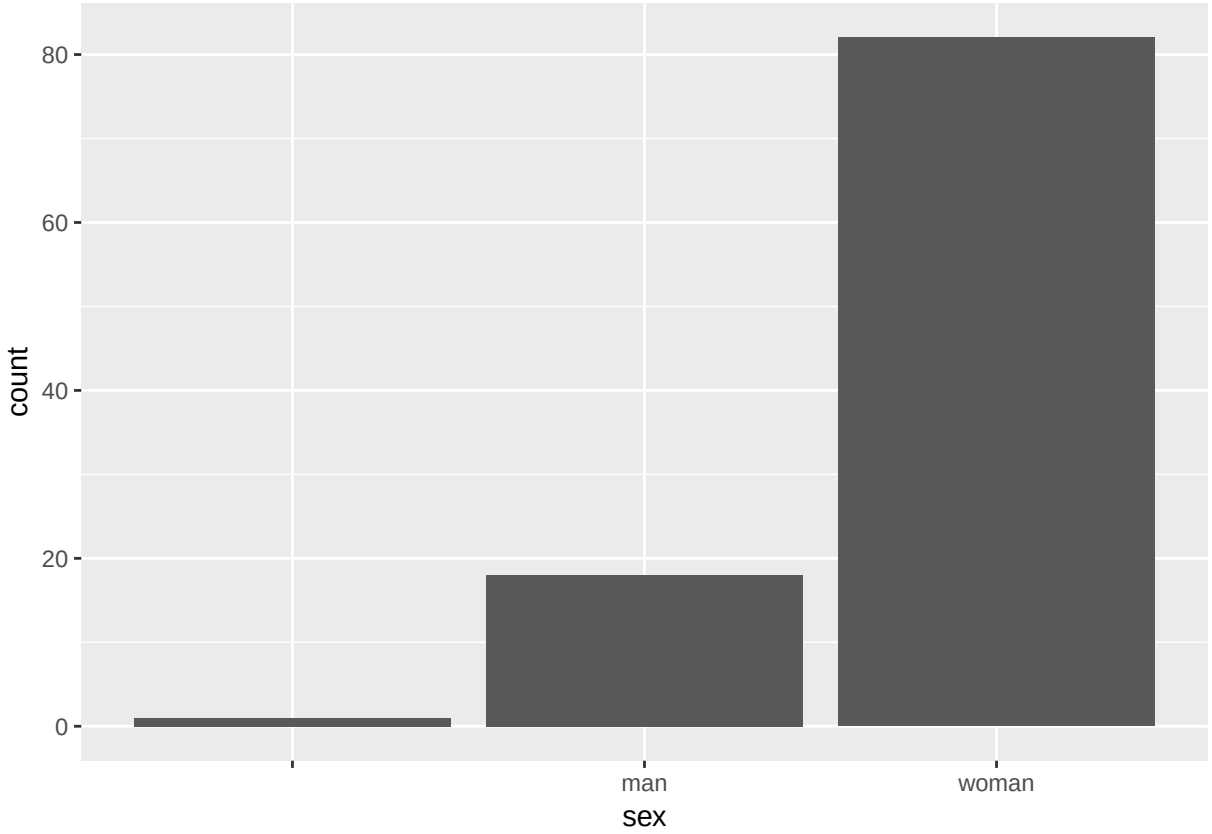


Mit `color = sex` sagen wir, dass die Linienfarbe (der Punkte) entsprechend der Stufen von `sex` eingefärbt werden sollen. Die genaue Farbwahl übernimmt `ggplot2` für uns.

Eine diskrete Variable

Bei diskreten Variablen, vor allem nominalen Variablen, geht es in der Regel darum, Häufigkeiten auszuzählen. Wie viele Männer und Frauen sind in dem Datensatz?

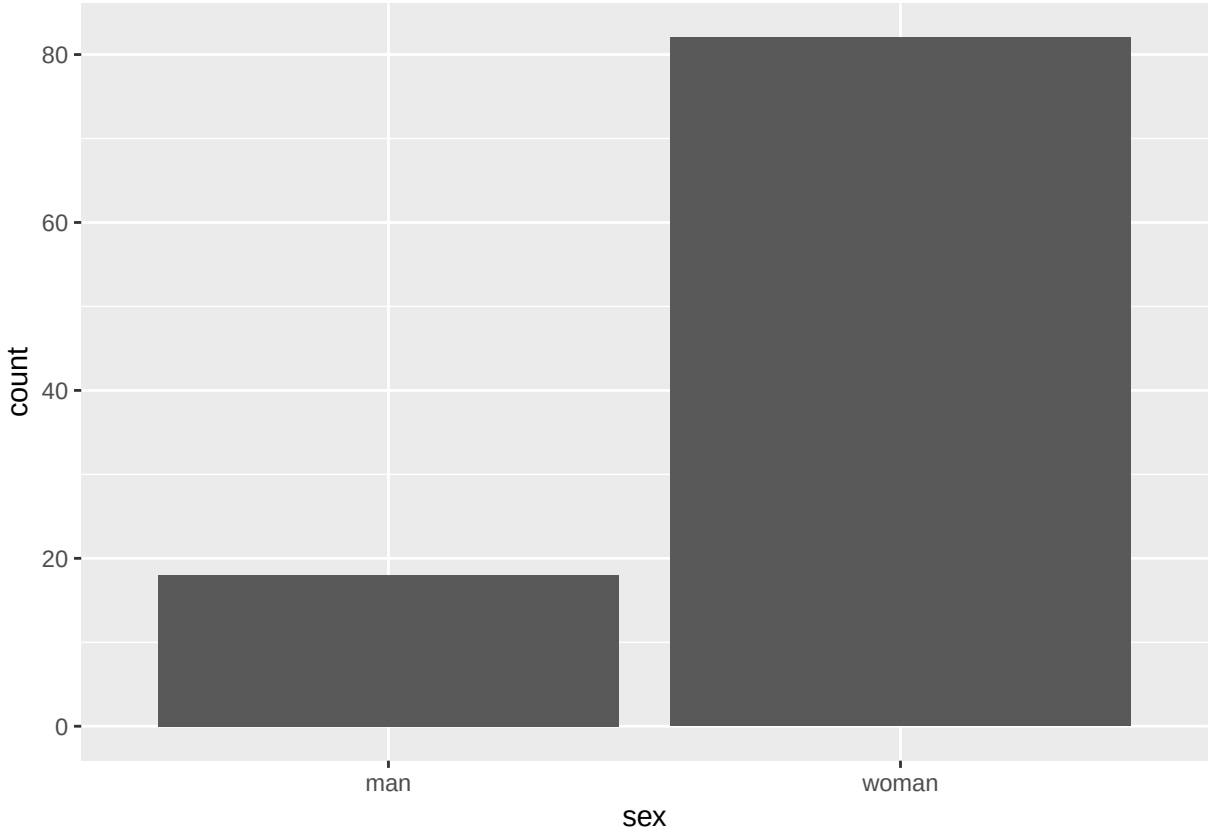
```
qplot(x = sex, data = wo_men)
```



Falls nur die X-Achse definiert ist und dort eine Faktorvariable oder eine Text-Variable steht, dann nimmt `qplot` automatisch ein Balkendiagramm als Geom.

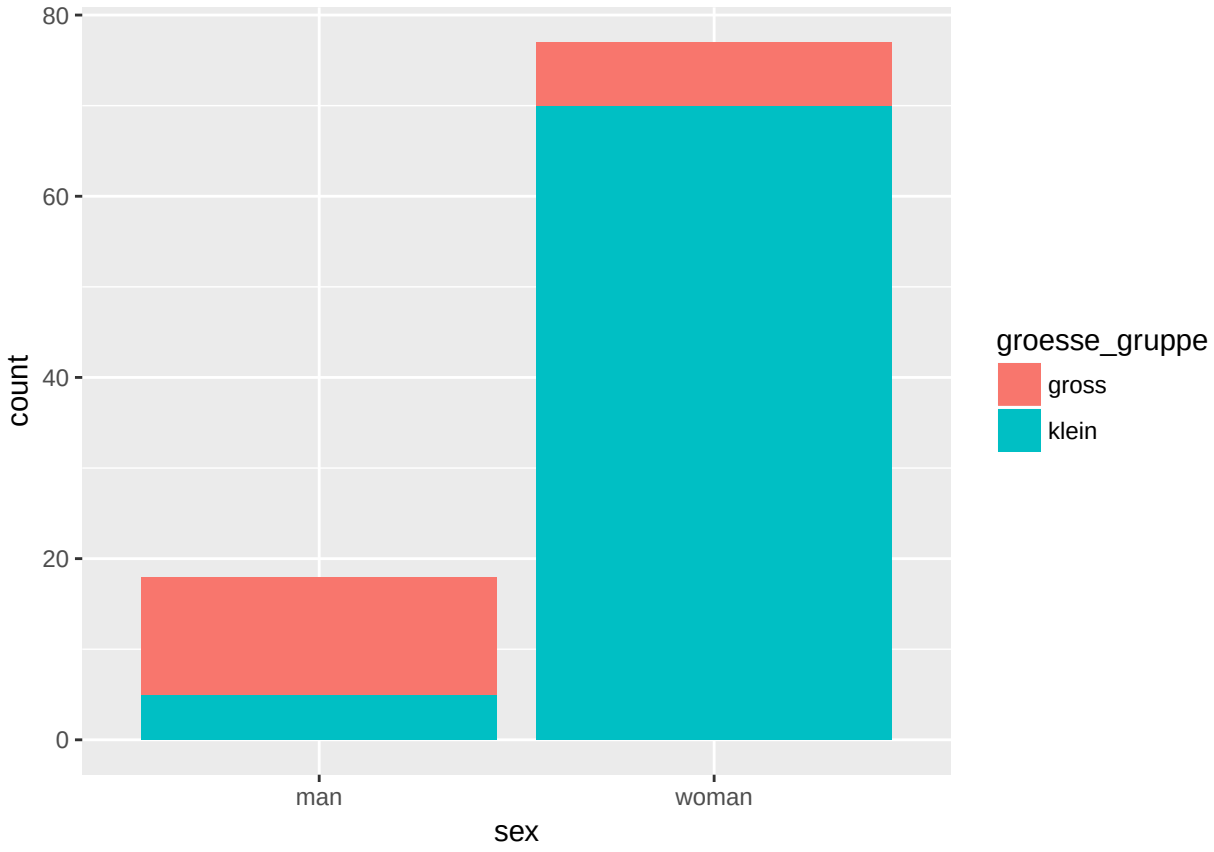
Entfernen wir vorher noch die fehlenden Werte:

```
wo_men %>%
  na.omit() %>%
  qplot(x = sex, data = .)
```



Wir könnten uns jetzt die Frage stellen, wie viele kleine und viele große Menschen es bei Frauen und bei den Männern gibt. Dazu müssen wir zuerst eine Variable wie "Größe gruppiert" erstellen mit zwei Werten: "klein" und "groß". Nennen wir sie `groesse_gruppe`

```
wo_men$groesse_gruppe <- car::recode(wo_men$height, "lo:175 = 'klein'; else = 'gross'")


wo_men %>%
  filter(height > 150, height < 210, shoe_size < 55) %>%
  na.omit -> wo_men2

qplot(x = sex, fill = groesse_gruppe, data = wo_men2)
```



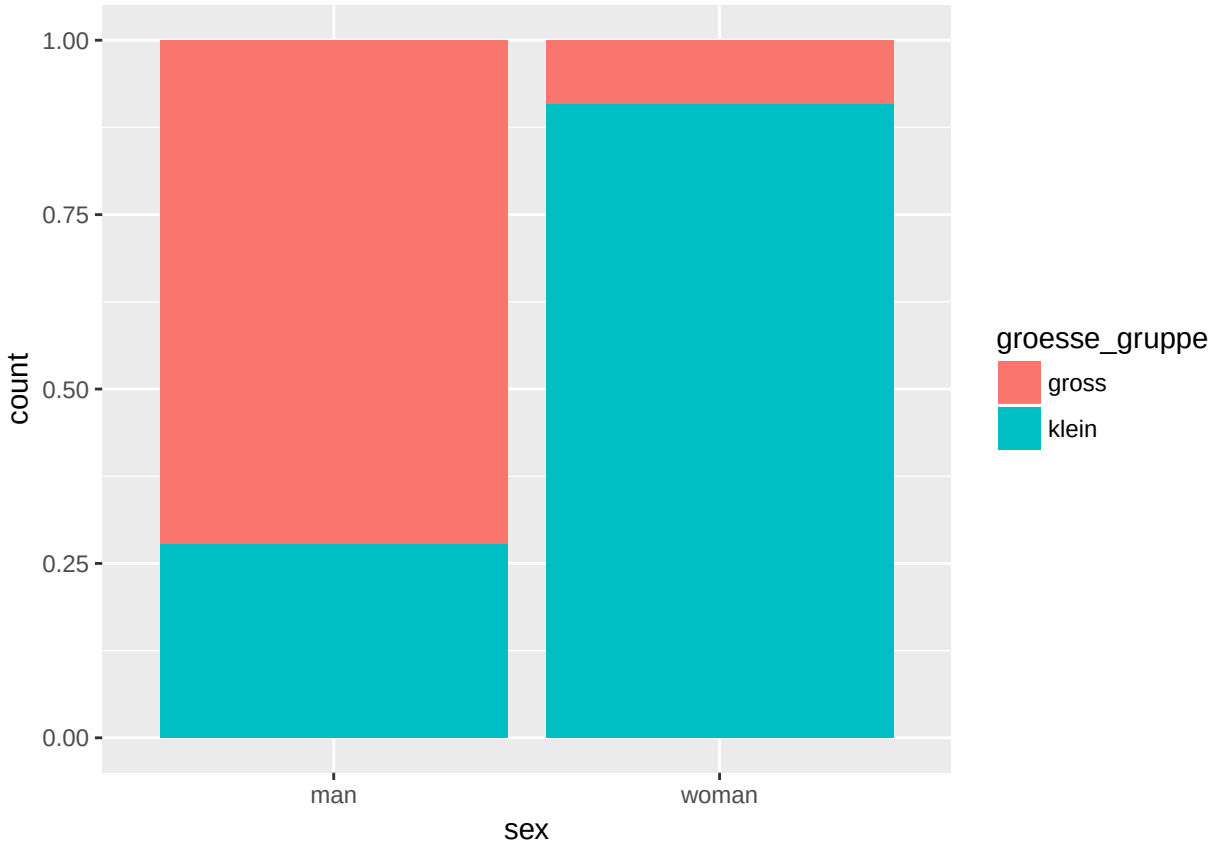
In Worten sagt der `recode`-Befehl hier in etwa: "Kodiere `wo_men$height` um, und zwar vom kleinsten (`lo`) Wert bis 170 soll den Wert `klein` bekommen, ansonsten bekommt eine Größe den Wert `gross`".

Hier haben wir `qplot` gesagt, dass der die Balken entsprechend der Häufigkeit von `groesse_gruppe` füllen soll. Und bei den Frauen sind bei dieser Variablen die Werte `klein` häufig; bei den Männern hingegen die Werte `gross`.

Schön wäre noch, wenn die Balken Prozentwerte angeben würden. Das geht mit `qplot` (so) nicht; wir schwenken auf `ggplot` um[21].

```
wo_men2 %>%
  ggplot() +
  aes(x = sex, fill = groesse_gruppe) +
  geom_bar(position = "fill")
```

[21] Cleveland fände diese Idee nicht so gut.



Schauen wir uns die Struktur des Befehls `ggplot` näher an.

`wo_men2`: Hey R, nimm den Datensatz `wo_men2`

`ggpplot()` : Hey R, male ein Diagramm von Typ ggplot (mit dem Datensatz aus dem vorherigen Pfeifen-Schritt, d.h. aus der vorherigen Zeile, also `wo_men2`)!

`+`: Das Pluszeichen grenzt die Teile eines ggplot-Befehls voneinander ab.

`aes`: von "aethetics", also welche Variablen des Datensatzes den sichtbaren Aspekten (v.a. Achsen, Farben) zugeordnet werden.

`x`: Der X-Achse (Achtung, `x` wird klein geschrieben hier) wird die Variable `sex` zugeordnet.

`y`: gibt es nicht??? Wenn in einem ggplot-Diagramm *keine* Y-Achse definiert wird, wird ggplot automatisch ein Histogramm bzw. ein Balkendiagramm erstellen. Bei diesen Arten von Diagrammen steht auf der Y-Achse keine eigene Variable, sondern meist die Häufigkeit des entsprechenden X-Werts (oder eine Funktion der Häufigkeit, wie relative Häufigkeit).

`fill` Das Diagramm (die Balken) sollen so gefüllt werden, dass sich die Häufigkeit der Werte von `groesse_gruppe` darin widerspiegelt.

`geom_XYZ`: Als "Geom" soll ein Balken ("bar") gezeichnet werden.

Ein Geom ist in ggplot2 das zu zeichnende Objekt, also ein Boxplot, ein Balken, Punkte, Linien etc. Entsprechend wird gewünschte Geom mit `geom_bar`, `geom_boxplot`, geom_pointetc. `gewählt.`position = fill`:`position_fill`will sagen, dass die Balken alls eine Höhe von 100% (1) haben. Die Balken zeigen also nur die Anteile der Werte der`fill'-Variablen.

Die einzige Änderung in den Parametern ist `position ='fill'`. Dieser Parameter weist `ggplot` an, die Positionierung der Balken auf die Darstellung von Anteilen auszulegen. Damit haben alle Balken

die gleiche Höhe, nämlich 100% (1). Aber die "Füllung" der Balken schwankt je nach der Häufigkeit der Werte von `groesse_gruppe` pro Balken (d.h. pro Wert von `sex`).

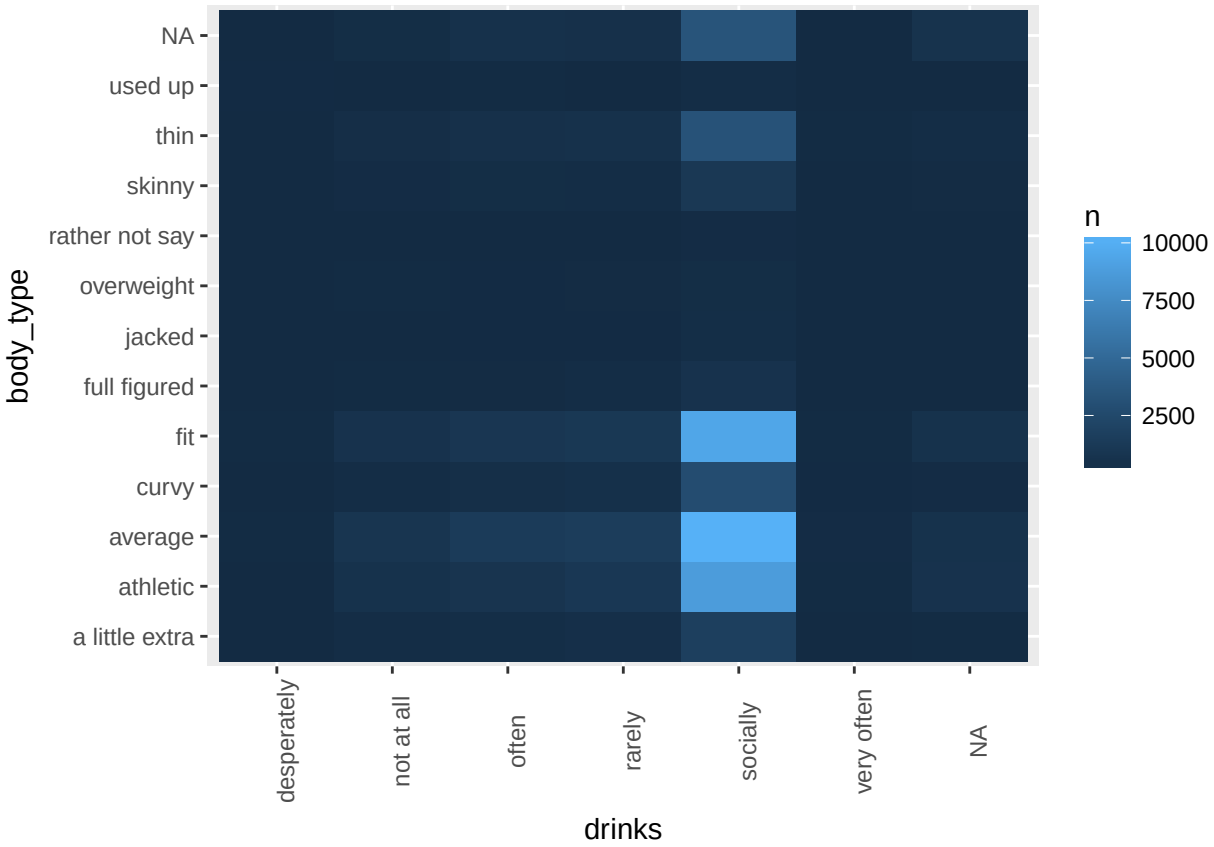
Wir sehen, dass die Anteile von großen bzw. kleinen Menschen bei den beiden Gruppen (Frauen vs. Männer) *unterschiedlich hoch* ist. Dies spricht für einen *Zusammenhang* der beiden Variablen; man sagt, die Variablen sind *abhängig* (im statistischen Sinne).

> Je unterschiedlicher die "Füllhöhe", desto stärker sind die Variablen (X-Achse vs. Füllfarbe) voneinander abhängig (bzw. desto stärker der Zusammenhang).

Zwei diskrete Variablen

Arbeitet man mit nominalen Variablen, so sind Kontingenztabellen Täglich Brot. Z.B.: Welche Produkte wurden wie häufig an welchem Standort verkauft? Wie ist die Verteilung von Alkoholkonsum und Körperform bei Menschen einer Single-Börse. Bleiben wir bei letztem Beispiel.

```
data(profiles, package = "okcupiddata")

profiles %>%
  count(drinks, body_type) %>%
  ggplot +
  aes(x = drinks, y = body_type, fill = n) +
  geom_tile() +
  theme(axis.text.x = element_text(angle = 90))
```



Was haben wir gemacht? Also:

```
Nehme den Datensatz "profiles" UND DANN
Zähle die Kombinationen von "drinks" und "body_type" UND DANN
Erstelle ein ggplot-Plot UND DANN
Weise der X-Achse "drinks" zu,
    der Y-Achse "body_type" und der Füllfarbe "n" UND DANN
Male Fliesen UND DANN
Passe das Thema so an, dass der Winkel für Text der X-Achse auf 90 Grad steht.
```

Was sofort ins Auge sticht, ist dass "soziales Trinken", nennen wir es mal so, am häufigsten ist, unabhängig von der Körperform. Ansonsten scheinen die Zusammenhäng nicht sehr stark zu sein.

Zusammenfassungen zeigen

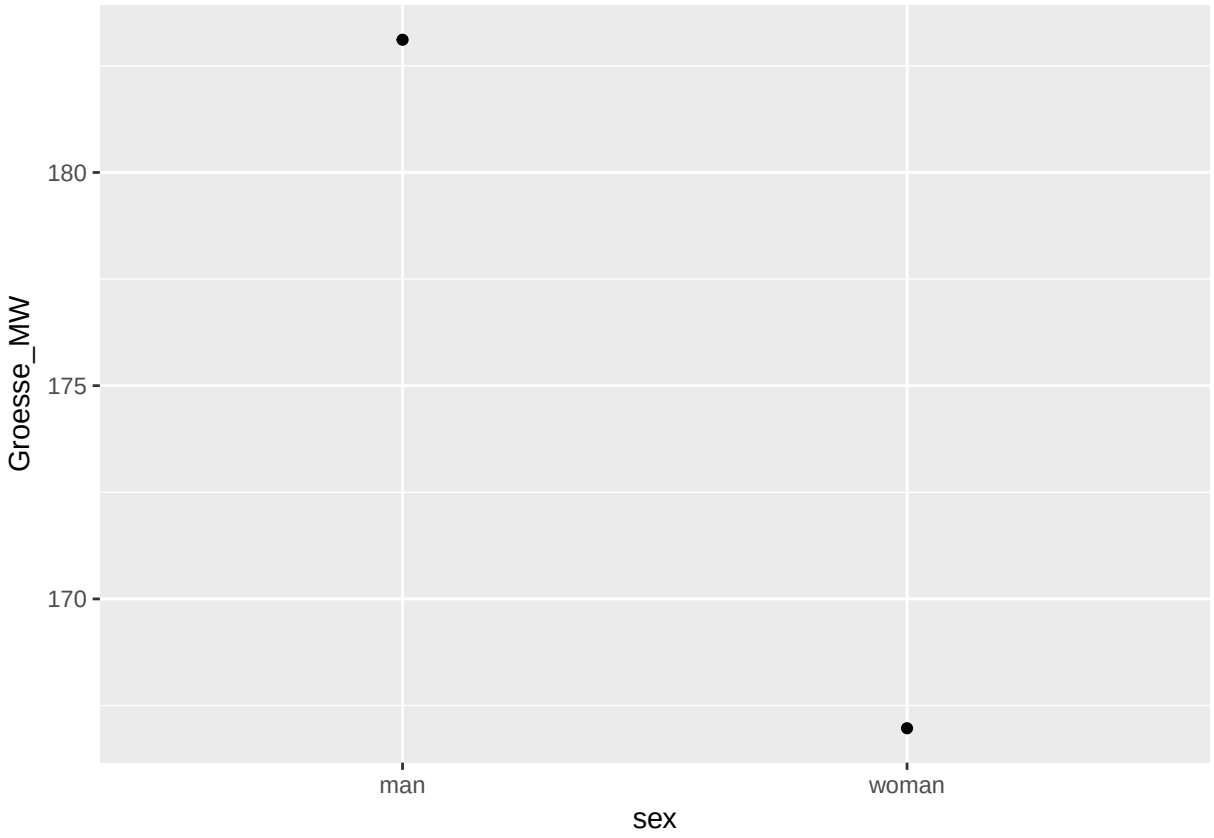
Manchmal möchten wir *nicht* die Rohwerte einer Variablen darstellen, sondern z.B. die Mittelwerte pro Gruppe. Mittelwerte sind eine bestimmte *Zusammenfassung* einer Spalte; also fassen wir zuerst die Körpergröße zum Mittelwert zusammen - gruppiert nach Geschlecht.

```
wo_men2 %>%
  group_by(sex) %>%
  summarise(Groesse_MW = mean(height)) -> wo_men3


wo_men3
```

```
## # A tibble: 2 x 2
##      sex Groesse_MW
##   <fctr>      <dbl>
## 1    man   183.1111
## 2  woman   166.9675
```

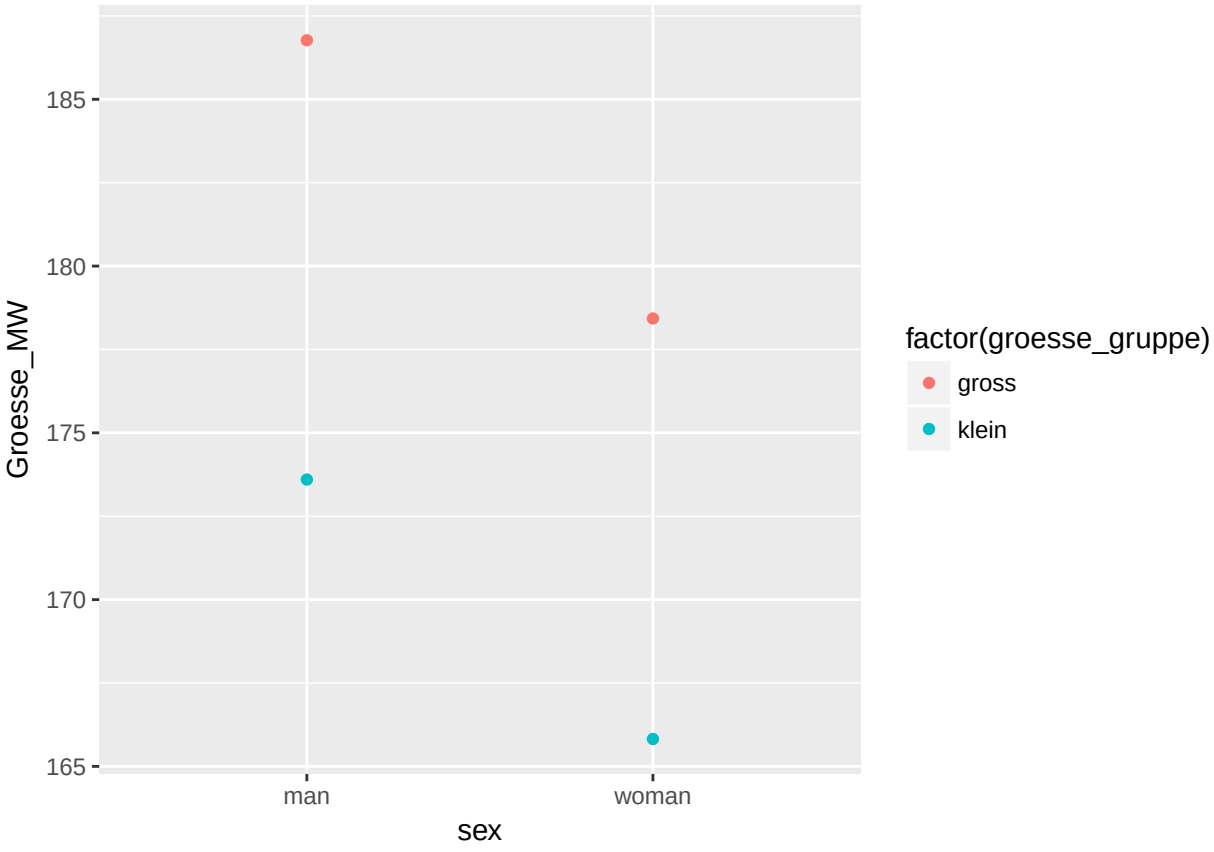
Diese Tabelle schieben wir jetzt in `ggplot2`; natürlich hätten wir das gleich in einem Rutsch durchpfeifen können.

```
wo_men3 %>%
  qplot(x = sex, y = Groesse_MW, data = .)
```



Das Diagramm besticht nicht durch die Tiefe und Detaillierung. Wenn wir noch zusätzlich die Mittelwerte nach `Groesse_Gruppe` ausweisen, wird das noch überschaubar bleiben.

```
wo_men2 %>%
  group_by(sex, groesse_gruppe) %>%
  summarise(Groesse_MW = mean(height)) %>%
  qplot(x = sex, color = factor(groesse_gruppe), y = Groesse_MW, data = .)
```



Verweise

- Edward Tufte gilt als Grand Seigneur der Datenvisualisierung; er hat mehrere lesenswerte Bücher zu dem Thema geschrieben [@1930824130; @1930824165; @1930824149].
- William Cleveland, ein amerikanischer Statistiker ist bekannt für seine grundlegenden, und weithin akzeptierten Ansätze für Diagramme, die die wesentliche Aussage schnörkellos transportieren [@Cleveland].
- Die Auswertung von Umfragedaten basiert häufig auf Likert-Skalen. Ob diese metrisches Niveau aufweisen, darf bezweifelt werden. Hier findet sich einige vertiefenden Überlegungen dazu und zur Frage, wie Likert-Daten ausgewertet werden könnten: https://bookdown.org/Rmadillo/likert/.

Hinweis

Erstellt von Sebastian Sauer