

Maik Thalmann  
[maik.thalmann@gmail.com](mailto:maik.thalmann@gmail.com)

# Tutorium zur Inferenzstatistik

Wintersemester 2019/20  
Version: 20. April, 2020

Georg-August-Universität Göttingen

---

## Inhaltsverzeichnis

<b>Vorbemerkungen</b> .....	5
Session Info .....	5
<b>1 Sitzung 1</b> .....	7
1.1 Allgemeines .....	7
1.2 Was kann man mit R alles anstellen? .....	10
1.3 RStudio-Optionen .....	15
<b>2 Sitzung 2</b> .....	16
2.1 Wiederholung .....	16
2.2 Vektoren .....	17
2.3 Zwei-Dimensionale Objekte .....	21
<b>3 Sitzung 3</b> .....	26
3.1 Wiederholung .....	26
3.2 Mehr zu Datenblättern .....	27
3.3 Daten exportieren .....	28
3.4 Daten einlesen .....	30
<b>4 Sitzung 4</b> .....	36
4.1 Wiederholung .....	36
4.2 Übung .....	36
4.3 Tabellenüberblick .....	39
4.4 Erste statistische Berechnungen .....	40
4.5 Nur Kurz: Plots .....	43
<b>5 Sitzung 5</b> .....	46
5.1 Wiederholung .....	46
5.2 Tabellen miteinander verbinden .....	47
5.3 Fehlende Werte .....	49
5.4 Faktoren .....	51

	Inhaltsverzeichnis	3
5.5	Nur Kurz: Plots .....	54
<b>6</b>	<b>Sitzung 6</b> .....	<b>57</b>
6.1	Wiederholung .....	57
6.2	Pakete und mehr Statistik .....	57
6.3	Eigene Funktionen schreiben .....	62
6.4	Tabellen transponieren .....	65
6.5	Nur Kurz: Plots .....	69
<b>7</b>	<b>Sitzung 7</b> .....	<b>71</b>
7.1	Wiederholung .....	71
7.2	Fehlertypen .....	72
7.3	$\chi^2$ -Test .....	72
7.4	<i>t</i> -test .....	79
7.5	Nur Kurz: Plots .....	87
<b>8</b>	<b>Sitzung 8</b> .....	<b>88</b>
8.1	Wiederholung .....	88
8.2	Base R: ANOVA .....	89
8.3	Alternative ANOVA-Befehle .....	93
8.4	Nur Kurz: Plots .....	99
<b>9</b>	<b>Sitzung 9</b> .....	<b>101</b>
9.1	Experimentanalyse .....	101
9.2	Daten einlesen und aufbereiten .....	102
9.3	Deskriptive Statistik .....	108
9.4	Plots .....	108
9.5	Inferenzstatistik .....	110
9.6	Kontraste .....	113
9.7	Nochmal Plots .....	118
<b>10</b>	<b>ggplot2</b> .....	<b>120</b>
10.1	Warum Plots: Anscombes Quartett .....	120
10.2	Beispiele .....	121
10.3	Einfache Plots .....	123
10.4	Plots erweitern .....	135
10.5	Plots verschönern .....	142
10.6	Plot speichern .....	150
10.7	Plots verbinden .....	150
10.8	stat_summary() .....	151
10.9	Fehlerbalken .....	153
10.10	Diagnostische Plots .....	155
10.11	Normalverteilung .....	158
10.12	Boxplot verbessern .....	160
<b>11</b>	<b>Extrasitzung</b> .....	<b>166</b>

4        Inhaltsverzeichnis

11.1 Einfach Textdateien speichern . . . . .	166
11.2 Funktionen aus dem tidyverse . . . . .	167
11.3 For-Loops . . . . .	172
11.4 Funktionen, die das Coden erleichtern . . . . .	173
11.5 Janitor-Paket . . . . .	176
11.6 Automatische Tabellen mit gtsummary . . . . .	178
11.7 Automatische Berichte . . . . .	179
11.8 Daten untersuchen . . . . .	182
11.9 Bildverarbeitung (und -bearbeitung) in R . . . . .	185
11.10 Popularity Contest . . . . .	185
11.11 Anhang: Die wichtigsten Funktionen . . . . .	187

---

## Vorbemerkungen

Dieses Buch ist zeitlich nach den Folien entstanden, die ich zum Unterrichten meiner Veranstaltung verwenden werde. Um gewährleisten zu können, dass der Inhalt der Folien und dieser Onlineressource deckungsgleich bleibt, werden beide auf Basis derselben R Markdown-Dateien erstellt – genauer im darauf basierenden bookdown-Format. Das hat einige Konsequenzen, die ästhetisch wenig zufriedenstellend sind, aber von mir billigend in Kauf genommen werden müssen). Ein offensichtlicher dieser Punkte betrifft die Abschnittsgliederung: Da Abschnitte im Präsentationsformat die Foliengrenzen markieren, ist dieses Buch leider vielerorts etwas unglücklich gegliedert (insbesondere in den Übungssektionen). Bis ich eine Lösung gefunden habe, die mir weiterhin erlaubt, in nur einem Dokument/Format zu arbeiten und mehrere Dateien generiert zu bekommen, muss das allerdings so bleiben.

Obwohl diese Resource eine Einführung in R ist, muss ich an dieser Stelle ein paar Voreinstellungen für die nachfolgenden Kapitel (insbesondere zu ladende Pakete und Schriftarteneinstellungen für Plots) vornehmen. Anfänger brauchen sich mit den folgenden Codezeilen nicht zu beschäftigen; ich wollte lediglich transparent vorgehen und offenlegen, welche externen Pakete ich verwende.

```
# load packages
Packages <- c("tidyverse", "gridExtra", "reshape2", "psych",
  "stringr", "afex", "ez", "car", "emmeans",
  # not really necessary
  "ggthemes", "ggpubr", "scico", "janitor",
  "magick", "ggnimate", "hrbrthemes")
xfun::pkg_attach(Packages, install = TRUE)
# base size for plots made with ggplot2
theme_set(theme_grey(base_size = 11))
# disable scientific number notation
options(scipen = 999)
```

## Session Info

```
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
```

6 Inhaltsverzeichnis

```
## Running under: macOS Catalina 10.15.1
##
## Locale: en_US.UTF-8 / en_US.UTF-8 / en_US.UTF-8 / C / en_US.UTF-8 / en_US.UTF-8
##
## Package version:
##   hrbrthemes_0.7.2     plotly_4.9.1      MASS_7.3-51.4
##   broom_0.5.2          ggsci_2.9        qqplotr_0.0.3
##   gganimate_1.0.4      magick_2.2       janitor_1.2.0
##   scico_1.1.0          ggpubr_0.2.4    magrittr_1.5
##   ggthemes_4.2.0        car_3.0-5       carData_3.0-3
##   ez_4.4-0              afex_0.25-1    lme4_1.1-21
##   Matrix_1.2-18        psych_1.8.12   reshape2_1.4.3
##   gridExtra_2.3         forcats_0.4.0  stringr_1.4.0
##   dplyr_0.8.3          purrr_0.3.3    readr_1.3.1
##   tidyrl_1.0.0          tibble_2.1.3   ggplot2_3.2.1
##   tidyverse_1.3.0       minqa_1.2.4    colorspace_1.4-1
##   ggsignif_0.6.0       rio_0.5.16     estimability_1.3
##   fs_1.3.1              rstudioapi_0.10 farver_2.0.1
##   mvtnorm_1.0-11        lubridate_1.7.4 xml2_1.2.2
##   codetools_0.2-16      splines_3.6.1  extrafont_0.17
##   mnormt_1.5-5          robustbase_0.93-5 knitr_1.26
##   zeallot_0.1.0         jsonlite_1.6   nloptr_1.2.1
##   Rttf2pt1_1.3.7        dbplyr_1.4.2   png_0.1-7
##   compiler_3.6.1        httr_1.4.1     emmeans_1.4.3.01
##   backports_1.1.5       assertthat_0.2.1 lazyeval_0.2.2
##   cli_1.1.0              tweenr_1.0.1   htmltools_0.4.0
##   prettyunits_1.0.2     tools_3.6.1    lmerTest_3.1-0
##   coda_0.19-3            gtable_0.3.0  glue_1.3.1
##   Rcpp_1.0.3              styler_1.2.0  cellranger_1.1.0
##   vctrs_0.2.0             nlme_3.1-142 extrafontdb_1.0
##   xfun_0.11              openxlsx_4.1.3 rvest_0.3.5
##   lifecycle_0.1.0        DEoptimR_1.0-8 scales_1.1.0
##   hms_0.5.2              parallel_3.6.1 rematch2_2.1.0
##   yaml_2.2.0              curl_4.3     gdtools_0.2.1
##   stringi_1.4.3           gifski_0.8.6 boot_1.3-23
##   zip_2.0.4              systemfonts_0.1.1 rlang_0.4.2
##   pkgconfig_2.0.3         evaluate_0.14  tidyselect_0.2.5 lattice_0.20-38
##   htmlwidgets_1.5.1        R6_2.4.1     tidyselect_0.2.5
##   bookdown_0.16            pillar_1.4.2  plyr_1.8.4
##   DBI_1.0.0              withr_2.1.2  generics_0.0.2
##   foreign_0.8-72           modelr_0.1.5 haven_2.2.0
##   abind_1.4-5              progress_1.2.2 mgcv_1.8-31
##   rmarkdown_1.18            data.table_1.12.6 crayon_1.3.4
##   readxl_1.3.1             xtable_1.8-4  grid_3.6.1
##   digest_0.6.23           viridisLite_0.3.0 reprex_0.3.0
##   munsell_0.5.0
```

# 1

---

## Sitzung 1

### 1.1 Allgemeines

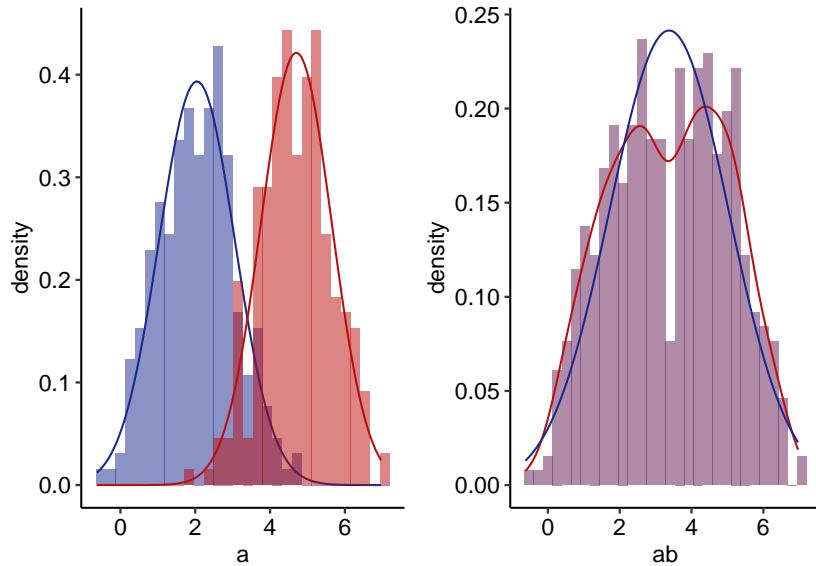
#### 1.1.1 Ein paar Worte zum Tutorium

- anfangs kaum Statistik, sondern eine Einführung in R
  - ich lade meine Foliensätze, die R-Scripte und die externen Dateien, die wir verwenden, bei StudIP hoch
- damit ihr was vom Tutorium habt, wäre es super, wenn ihr zu jeder Sitzung einen Computer mitbringen würdet (oder euch zusammenschließt und zu zweit einen Computer benutzt)
  - wenn ihr keinen Computer zur Verfügung habt, gibt es auch Apps für R, die sind aber kein guter Ersatz; vor allem, wenn wir in der dritten Sitzung anfangen, mit richtigen Daten zu arbeiten
  - IOS: <https://apps.apple.com/de/app/r-programming-compiler/id1158038782> (nicht getestet)
  - Android ???
- Wie ihr R auf euren Computer bekommt, zeige ich euch gleich

#### 1.1.2 R: Ein kurzer Überblick

- 1992 entwickelt, angelehnt an die Programmiersprache S
- Fokus liegt auf statistischen Berechnungen und Grafiken aller Art
- weit verbreitet, Anwendung vor allem in der Wissenschaft
- fortlaufende Weiterentwicklung durch die Nutzergemeinschaft (v.a. in Form von Paketen)
- kostenlos verwendbar

### 1.1.3 Ein Beispielplot



### 1.1.4 Installation

- Am wichtigsten: R selbst
  - <https://cran.r-project.org>
- Wir werden zwar hauptsächlich mit R arbeiten, aber da die Oberfläche wenig benutzerfreundlich ist, verwende ich außerdem Rstudio
  - <https://www.rstudio.com>
- Bitte alle beides installieren und zum Laufen bringen!

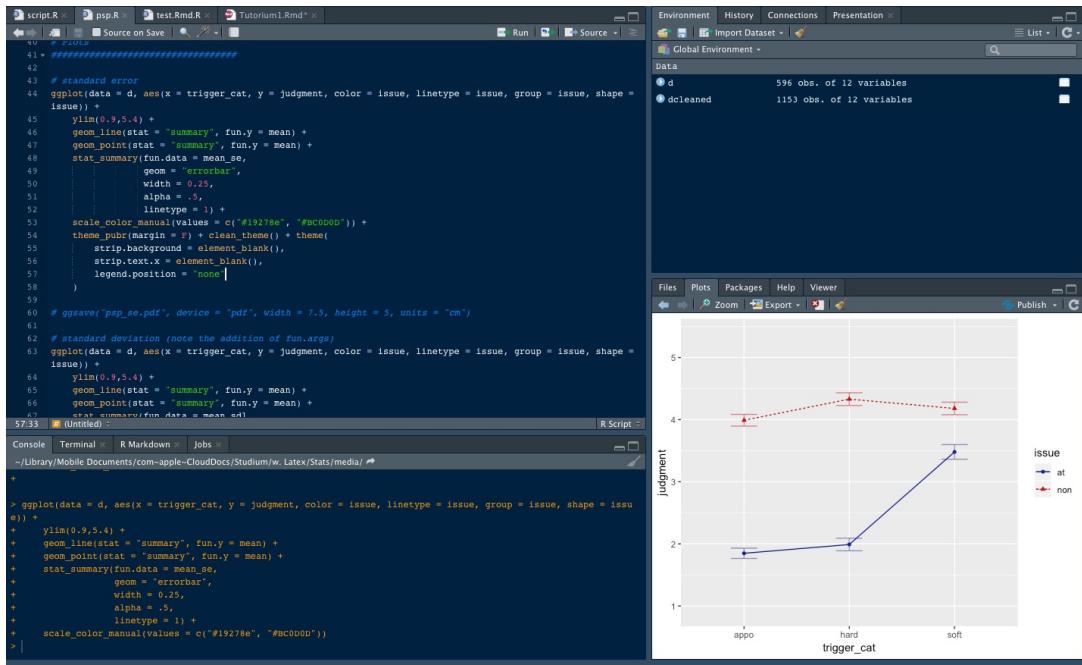
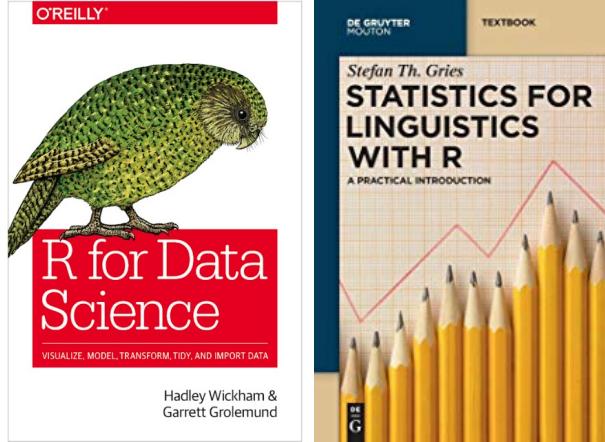


Abb. 1.1: Und so sieht das Ganze dann ungefähr aus

### 1.1.5 Tada!

### 1.1.6 Hilfestellung

<https://r4ds.had.co.nz> / <https://www.degruyter.com/view/product/203826>



FREE R READING MATERIAL			
A collection of books about the R programming language and Data Science, that you can read for free!			
If there is a book that you think I should add to the list, then please let me know <a href="#">@committedtotape</a>			
Show 20 <input checked="" type="checkbox"/> entries	Search: <input type="text"/>		
Title	Author(s)	Description	Keywords/Packages
R for Data Science	Garrett Grolemund, Hadley Wickham	This book will teach you how to do data science with R	tidyverse,ggplot2,dplyr,tidyr,readr
Hands-On Programming with R	Garrett Grolemund	This book will teach you how to program in R, with hands-on examples.	functions,programming,simulations
R Graphics Cookbook	Winston Chang	A practical guide that provides more than 150 recipes to help you generate high-quality graphs quickly	ggplot2,data visualization
Modern Dive	Chester Ismay, Albert Y. Kim	Statistical Inference via Data Science	tidyverse,infer,broom,inference,regression,sampling,hypothesis testing,confidence intervals
An Introduction to Statistical Learning with Applications in R	Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani	This book provides an introduction to statistical learning methods.	machine learning,supervised learning,unsupervised learning

Abb. 1.2: Mehr kostenlose Bücher und Resourcen: <https://committedtotape.shinyapps.io/freeR/>

### 1.1.7 Mehr Hilfe

### 1.1.8 Allgemeines

Wir werden hauptsächlich mit Zahlen arbeiten, darum ist das Folgende wichtig: - R verwendet den amerikanischen Dezimaltrenner (0.1666667 statt 0,1666667)

1 / 6

```
## [1] 0.1666667
```

Um Hilfe mit Funktionen zu bekommen: - ?Funktionsname - help(Funktionsname)

- Gebt mal Folgendes ein:

```
?mean  
help(mean)
```

## 1.2 Was kann man mit R alles anstellen?

### 1.2.1 Skripte

R ist eine Programmiersprache, die, wie alle anderen auch, textbasiert ist. D.h. man arbeitet in der Regel mit Skripten, in denen man seinen Code schreibt und diesen dann ausführt.

Hier ein Beispiel für ein (zugegeben sehr einfaches) Skript:

The screenshot shows the RStudio interface with three tabs at the top: 'psp.R', 'Tutorium1.Rmd\*', and 'Untitled1\*'. The 'Source' tab is selected. Below the tabs, there are several icons. In the main workspace, there is a code editor containing the following five lines of R code:

```

1 1+2
2 17-9
3 5*6
4 9/3
5 4^3

```

Abb. 1.3: Und so sieht das Ganze dann ungefähr aus

### 1.2.2 Skripte

Zwar könnetet ihr auch einfach in der Konsole in Rstudio euren Code schreiben, aber dann geht er euch ziemlich schnell verloren und ihr könnt nicht einfach dort wieder anfangen, wo ihr aufgehört habt.

### 1.2.3 Übung 1

- erstellt einen Order auf dem Desktop
- erstellt ein Skript in RStudio und speichert es in diesem Ordner
- macht eure Noizen am besten in diesem R-Skript
  - Fließtext als Kommentare nach dem #-Zeichen

```

# dies ist ein Kommentar, den R einfach ignoriert
print("Hello World") # ebenso mit dem hier

## [1] "Hello World"

```

### 1.2.4 R als Taschenrechner

```

1 + 2
## [1] 3
17 - 9
## [1] 8
5 * 6
## [1] 30

```

```
9 / 3
```

```
## [1] 3
```

```
4^3
```

```
## [1] 64
```

### 1.2.5 Mathematische Funktionen in R

- Quadratwurzel: sqrt()
- Summe: sum()
- Minimum: min()
- Maximum: max()
- Zahlenreihen von x bis y: x:y

```
1:5
```

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

### 1.2.6 Übung II

- addiert 500 und 5
- subtrahiert 19 von 3
- dividiert 144 durch 12
- addiert alle Zahlen von 1 bis 1000
- addiert 4 und 5 fünf mal hintereinander

### 1.2.7 Lösungen II

```
500 + 5
```

```
## [1] 505
```

```
19 - 3
```

```
## [1] 16
```

```
144 / 12
```

```
## [1] 12
```

```
sum(1:1000)
```

```
## [1] 500500
```

```
4 + 5;4 + 5;4 + 5;4 + 5;4 + 5
```

```
## [1] 9
```

```
## [1] 9
```

```
## [1] 9
```

```
## [1] 9
```

```
## [1] 9
```

### 1.2.8 Logische Operatoren

Logische Operatoren geben Wahrheitswerte (wie in der Semantik) wieder: entweder TRUE (oder T) oder FALSE (oder F)

- Vergleichsoperatoren
  - Gleich: ==
  - Ungleich: !=
  - Kleiner: <
  - Größer: >
  - Kleiner gleich: <=
  - Größer gleich: >=
- Logische Operatoren
  - Und/Konjunktion: &
  - Oder/Disjunktion: |
  - Negation: !

### 1.2.9 Beispiele: Logische Operatoren

```
4 == 4
## [1] TRUE
4 != 5
## [1] TRUE
17 / 9 >= 2 + 1
## [1] FALSE
TRUE & FALSE == FALSE
## [1] TRUE
TRUE | FALSE == TRUE
## [1] TRUE
TRUE | FALSE == FALSE
## [1] TRUE
```

### 1.2.10 R ist objektorientiert

- In R ist es oftmals wichtig, Berechnungen für die spätere Wiederverwertung zu speichern
- das geschieht, indem man ihnen einen
- diese Objekte erscheinen dann in der Arbeitsumgebung (eng. Environment)

```
a <- 9
b <- 81
```

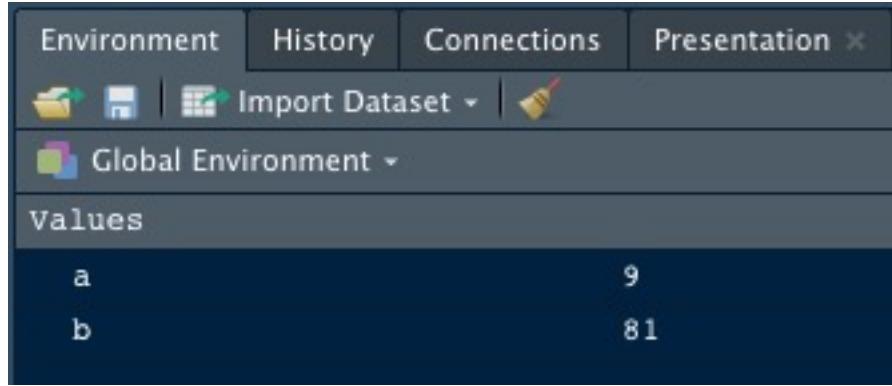


Abb. 1.4: Arbeitsumgebung

### 1.2.11 Beispiele

```
a <- 4 + 5
## [1] 9
a * 2
## [1] 18
b <- a^2
b
## [1] 81
(c <- a + b)
## [1] 90
```

### 1.2.12 Mehr zu Objekten

Wir können uns Objekte (oft auch **Varablen** genannt) als Mikrospeicherorte in unserem Makrospeicher (dem Skript selbst) vorstellen. Aus diesem Grund ist es wichtig, dass ihr Berechnungen und anderes, das ihr wiederverwenden wollt, als Objekt (mit sinnigem Namen) speichert

```
meinname <- "Maik Thalmann"
meineuni <- "Uni Göttingen"
paste(meinname, "studierte an der", meineuni)

## [1] "Maik Thalmann studierte an der Uni Göttingen"
```

Wenn ihr euch die Arbeitsumgebung anzeigen lassen möchtet, verwendet `ls()`, löschen könnt ihr sie mit `rm(list = ls())`

```
ls()
## [1] "a"        "b"        "c"        "meineuni" "meinname"
```

### 1.2.13 Datentypen in R

Ihr kennt bisher drei Datentypen:

- numeric: Zahlen
  - integer
  - double
- character: Zeichenketten
- logical: Wahrheitswerte

```
mode(a)
## [1] "numeric"
typeof(a)
## [1] "double"
typeof(FALSE)
## [1] "logical"
typeof(meinname)
## [1] "character"
```

## 1.3 RStudio-Optionen

### 1.3.1 Optionen, die das Leben erleichtern

Bitte stellt folgende Optionen in RStudio ein:

- Tools → Global Options → Code →
  - Editing
    - Soft-Wrap Source Files
  - Saving
    - Default Text Encoding: UTF-8
  - Display
    - Highlight Selected Line
    - Show syntax highlighting in console input

### 1.3.2 Das war's!

Danke fürs mitmachen und bis nächste Woche!

## Sitzung 2

### 2.1 Wiederholung

#### 2.1.1 Das Wichtigste vom letzten Mal

- R ist eine Programmiersprache für Statistik
  - kostenlos
  - nicht immer benutzerfreundlich, deshalb RStudio
- es gibt einiges zu beachten:
  - Ihr solltet immer mit einem Skipt arbeiten, nur in Ausnahmefällen in der Konsole
  - Hilfe bekommt ihr durch ?Funktion oder help(Funktion)
  - Wenn ihr Berechnungen macht und Daten aufzeichnet, bzw. einlest, und auf diese später wieder zugreifen möchtet, solltet ihr sie Objekten zuweisen
  - die Verschiedenen Datentypen:
    - numeric: Zahlen
    - character: Buchstabenfolgen in Anführungszeichen
    - logical: Wahrheitswerte

#### 2.1.2 Code vom letzten Mal

```
a <- 1:17  
b <- sum(a)  
b
```

```
## [1] 153
```

```
TRUE != FALSE
```

```
## [1] TRUE
```

```
(b & 3) >= 1 / 9
```

```
## [1] TRUE
```

```
b <- "überschreiben"
b == 153

## [1] FALSE
typeof(b)

## [1] "character"
```

## 2.2 Vektoren

### 2.2.1 Moment mal

Gerade haben wir folgenden Code gesehen:

```
a <- 1:17
```

Hatten wir beim letzten Mal nicht gesagt, dass x:y in R “von x bis y” bedeutet? Schaut mal, was in der Berechnung unten passiert.

```
1:3 + 10

## [1] 11 12 13
a <- 1:3 + 10

## [1] 11 12 13
```

Was passiert hier? Genau, wir können nicht nur einzelne Werte einem Objekt zuordnen, sondern auch eine Sequenz aus mehreren Werten unter einem Objektnamen zusammenfassen.

### 2.2.2 Vektoren

Wir haben nun gesehen, dass wir Elemente kombinieren können – diese komplexen Elemente nennt man Vektoren und sie funktionieren in etwa so geordnete Tupel in der Mengentheorie. Zahlenreihen können wir mit dem Doppelpunktoperator verbinden, bei anderen Datentypen verwenden wir c() (für “concatenate” – verketten)

```
a <- 7:14
b <- c(7, 8, 9, 10, 11, 12, 13, 14)
a == b

## [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE

c <- c("Als", "Gregor", "Samsa", "eines", "Morgens")
```

### 2.2.3 Subsetting

Wenn wir die Vektoren erstellt haben, können einzelne Elemente können per Subsetting herausgepickt werden. Dazu werden eckige Klammern mit positionalen Idizes verwendet ...

```
c[5] # zeigt das fünfte Element an
## [1] "Morgens"
(subset <- c[2:3]) # Elemente mit den Indizes von 2 bis 3
## [1] "Gregor" "Samsa"
... oder Bedingungen für das Enthaltensein in der Teilmenge:
b[b >= 10]
## [1] 10 11 12 13 14
```

#### 2.2.4 Dimensionen von Vektoren

Bei solch Komplexen Objekten interessiert uns oft die Anzahl der enthaltenen Elemente. Diese können wir uns mithilfe von `length()` ausgeben lassen:

```
a
## [1] 7 8 9 10 11 12 13 14
length(a)
## [1] 8
length(c)
## [1] 5
length(c(a, c))
## [1] 13
```

#### 2.2.5 Nochmal Subsetting

Wenn wir Subsetting mit der `length`-Funktion verbinden, erhalten wir das letzte Element des Vektors

```
a[length(a)]
## [1] 14
a[length(a) - 1] # vorletztes Element
## [1] 13
```

#### 2.2.6 Übungen

1. addiere zu jeder Zahl von 1 bis 700 die Zahl 13 und speichere das Ergebnis in einem Vektor ab.
2. speichere das 345. Element dieses Vektors in einem neuen Vektor und füge zum diesem Vektor außerdem das 666. Element des ersten Vektors hinzu

3. lass dir die letzten 22 elemente des langen vektors ausgeben
4. lass dir dasjenige element anzeigen, das die mitte des langen vektors bildet
5. addiere alle elemente im langen vektor auf

### 2.2.7 Lösungen

1. addiere zu jeder zahl von 1 bis 1000 die zahl 13 und speichere das ergebnis in einem vektor ab.

```
x <- 1:700
y <- x + 13
y
```

```
## [1] 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
## [19] 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
## [37] 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67
## [55] 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85
## [73] 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103
## [91] 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121
## [109] 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139
## [127] 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157
## [145] 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175
## [163] 176 177 178 179 180 181 182 183 184 185 186 187 188 189 189 190 191 192 193
## [181] 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211
## [199] 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229
## [217] 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247
## [235] 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265
## [253] 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283
## [271] 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301
## [289] 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319
## [307] 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337
## [325] 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355
## [343] 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373
## [361] 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391
## [379] 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409
## [397] 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427
## [415] 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445
## [433] 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463
## [451] 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481
## [469] 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499
## [487] 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517
## [505] 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535
## [523] 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553
## [541] 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571
## [559] 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589
## [577] 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607
## [595] 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625
## [613] 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643
## [631] 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661
## [649] 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679
## [667] 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697
## [685] 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713
```

### 2.2.8 Lösungen

2. speichere das 345. element dieses vektors in einem neuen vektor und füge zum diesem vektor außerdem das 666. Element des ersten Vektors hinzu

```

z <- y[345]
z <- c(z, y[666])
z

## [1] 358 679

3. lass dir die letzten 22 elemente des langen vektors ausgeben
y[(length(y) - 21):length(y)]

## [1] 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710
## [20] 711 712 713

y[length(y):(length(y) - 21)]

## [1] 713 712 711 710 709 708 707 706 705 704 703 702 701 700 699 698 697 696 695
## [20] 694 693 692

```

### 2.2.9 Lösungen

4. lass dir dasjenige element anzeigen, das die mitte des langen vektors bildet

```
y[length(y) / 2]
```

```
## [1] 363
```

5. addiere alle elemente im langen vektor auf

```
sum(y)
```

```
## [1] 254450
```

### 2.2.10 Heads und Tails

Oft ist es praktisch, sich die ersten und letzten Elemente eines Objektes anzeigen zu lassen. Wir haben verschiedene Möglichkeiten gesehen, wie man das erreichen kann.

Glücklicherweise hat R gesonderte Funktionen für genau solche Operationen und erspart uns somit Tipparbeit

```
y[1:6]
```

```
## [1] 14 15 16 17 18 19
```

```
head(y)
```

```
## [1] 14 15 16 17 18 19
```

```
y[(length(y) - 5):length(y)]
```

```
## [1] 708 709 710 711 712 713
```

```
tail(y)
```

```
## [1] 708 709 710 711 712 713
```

### 2.2.11 Listen

Manchmal ist es praktisch, wenn die Elemente eines Vektors selbst komplex sind und somit Elemente einer bestimmten Art enthalten. Dazu verwendet man `list`

```
kind <- c("Fara", "Lula", "Fu")
note <- c(2, 1, 5)
(klausur <- list(kind, note))

## [[1]]
## [1] "Fara" "Lula" "Fu"
##
## [[2]]
## [1] 2 1 5
```

### 2.2.12 Subsetting von Listen

Bei Objekten mit komplexen Elementen funktioniert das Subsetting etwas anders:

```
klausur[2] # gibt eine Liste aus

## [[1]]
## [1] 2 1 5

klausur[[2]] # gibt einen Vektor aus

## [1] 2 1 5

klausur[[1]][3] # gibt das dritte Element des ersten Vektors aus

## [1] "Fu"

klausur[1][3]

## [[1]]
## NULL
```

## 2.3 Zwei-Dimensionale Objekte

### 2.3.1 Über Vekten hinaus

Wir haben gesehen, dass eindimensionale Objekte erstellen können, die sich Vektoren (oder Listen) nennen. Was ist nun mit Objekten, die sowohl eine Länge wie auch eine Breite haben (read: Tabellen)?

```
kind

## [1] "Fara" "Lula" "Fu"

note

## [1] 2 1 5
```

```
(klausur <- cbind(kind, note))

##   kind   note
## [1,] "Fara" "2"
## [2,] "Lula" "1"
## [3,] "Fu"   "5"

is.matrix(klausur)

## [1] TRUE
```

### 2.3.2 Die Dimensionen von Tabellen

Die `length`-Funktion ist sehr irreführend bei Matritzen. Es ist besser, die `dim`-Funktion zu verwenden.

```
length(klausur)

## [1] 6

dim(klausur)

## [1] 3 2

klausur[1, ] # erste Zeile

##   kind   note
## "Fara" "2"

klausur[, 1] # erste Spalte

## [1] "Fara" "Lula" "Fu"

klausur[3, 2] # dritte Zeile, zweites Element

## note
## "5"
```

### 2.3.3 Subsetting von Matritzen

```
klausur[1, ] # erste Zeile

##   kind   note
## "Fara" "2"

klausur[, 1] # erste Spalte

## [1] "Fara" "Lula" "Fu"

klausur[3, 2] # dritte Zeile, zweites Element

## note
## "5"
```

### 2.3.4 Datenblätter

Zwar sind Matritzen geeignet, um Daten in Tabellenform zu repräsentieren. Eine wesentlich komfortablere Möglichkeit (und die, die wir im Verlaufe des Semesters fast ausschließlich verwenden werden) bieten data frames.

```
(klausur <- data.frame(kind, note))

##   kind note
## 1 Fara    2
## 2 Lula    1
## 3 Fu     5

is.data.frame(klausur)

## [1] TRUE

dim(klausur)

## [1] 3 2

length(klausur) # gibt Spaltenanzahl aus

## [1] 2
```

### 2.3.5 Subsetting von Datenblättern

```
klausur$note

## [1] 2 1 5

klausur[1, ]

##   kind note
## 1 Fara    2

klausur[, 2]

## [1] 2 1 5

(klausur2 <- subset(klausur, note > 3))

##   kind note
## 3   Fu    5
```

### 2.3.6 Datenblätter erweitern

```
geschlecht <- c("w", "w", "m")
note2 <- c(4, 3, 2)
(klausur <- cbind(klausur, note2, geschlecht))

##   kind note note2 geschlecht
## 1 Fara    2      4          w
## 2 Lula    1      3          w
## 3   Fu    5      2          m
```

```
dim(klausur)
## [1] 3 4
klausur$schnitt <- (klausur$note + klausur$note2) / 2
klausur
##   kind note note2 geschlecht schnitt
## 1 Fara    2     4          w      3.0
## 2 Lula    1     3          w      2.0
## 3 Fu      5     2          m      3.5
```

### 2.3.7 Übungen II

1. Fragt eure Sitznachbarn (links und rechts) nach
  - Vorname
  - Lieblingsessen
  - Geburtsmonat und erstellt für alle drei Faktoren Vektoren mit den jeweiligen Infos für Nachbar\_in 1 Nachbar\_in 2 und euch selbst
2. Erstellt aus diesen Vektoren einen data frame
3. Lasst euch das zweite Element der dritten Zeile ausgeben
4. Lasst euch die Zeile mit den Informationen über die Person links von euch anzeigen
5. Speichert die Spalte mit dem Lieblingsessen in einem neuen Vektor "mealprep" ab

### 2.3.8 Lösungen II

2. Erstellt aus diesen Vektoren einen data frame

```
vorname <- c("Marta", "Thomas", "Maik")
essen <- c("Schnitzel", "Auflauf", "Bier")
monat <- c(6, 8, 12)
(people <- data.frame(vorname, essen, monat))

##   vorname     essen monat
## 1 Marta Schnitzel     6
## 2 Thomas Auflauf     8
## 3 Maik   Bier     12
```

3. Lasst euch das zweite Element der dritten Zeile ausgeben

```
people[3, 2]

## [1] Bier
## Levels: Auflauf Bier Schnitzel
```

### 2.3.9 Lösungen II

4. Lasst euch die Zeile mit den Informationen über die Person links von euch anzeigen

```
people[1, ]  
##   vorname    essen monat  
## 1  Marta Schnitzel     6
```

5. Speichert die Spalte mit dem Lieblingsessen in einem neuen Vektor “mealprep” ab  
(mealprep <- people\$essen)

```
## [1] Schnitzel Auflauf    Bier  
## Levels: Auflauf Bier Schnitzel
```

### 2.3.10 Tüdelü

Bis zum nächsten Mal!

# 3

---

## Sitzung 3

### 3.1 Wiederholung

#### 3.1.1 Das Wichtigste vom letzten Mal

- Vektoren sind eindimensionale Tupel von Elementen

```
# vektoren erstellen
a <- 1:19
b <- a + 19
c <- c("svo", "sov", "ovs")
# vektoren ausgeben lassen
c

## [1] "svo" "sov" "ovs"

length(b)

## [1] 19
```

#### 3.1.2 Das Wichtigste vom letzten Mal

- Dataframes sind Tabellen (und damit 2-dimensional)

```
df <- data.frame(a, b)
dim(df)

## [1] 19  2

head(df, 1) # eine statt sechs zeilen als optionales argument

##   a   b
## 1 1 20

tail(df, 2)

##   a   b
## 18 18 37
## 19 19 38
```

```
df$c <- a + b
head(df, 1)
```

```
##   a   b   c
## 1 1 20 21
```

### 3.1.3 Das Wichtigste vom letzten Mal

```
head(df[, 1])
## [1] 1 2 3 4 5 6
df[df$a == 19, ]
##   a   b   c
## 19 19 38 57
df[df$a + df$b > 50, ]
##   a   b   c
## 16 16 35 51
## 17 17 36 53
## 18 18 37 55
## 19 19 38 57
subset(df, b > 35) # alle Reihen, auf die die Bedingung zutrifft
##   a   b   c
## 17 17 36 53
## 18 18 37 55
## 19 19 38 57
```

## 3.2 Mehr zu Datenblättern

### 3.2.1 Spalten umbenennen

```
names(df) <- c("low", "medium", "high")
head(df)
##   low medium high
## 1   1     20   21
## 2   2     21   23
## 3   3     22   25
## 4   4     23   27
## 5   5     24   29
## 6   6     25   31
```

### 3.2.2 Mehr Subsetting

- logische Bedingungen (beachtet hier das Komma! Wir arbeiten jetzt mit 2-dimensionalen Daten und müssen mit angeben, ob wir uns auf Zeilen oder Spalten beziehen)

```
df[df$low >= 17, ]  
  
##    low medium high  
## 17   17     36   53  
## 18   18     37   55  
## 19   19     38   57  
  
df[df$low + df$medium > 53, ]  
  
##    low medium high  
## 18   18     37   55  
## 19   19     38   57  
  
df[df$low %% 2 == 0 & df$high > 38, ] # modulo division -> rest = 0  
  
##    low medium high  
## 10   10     29   39  
## 12   12     31   43  
## 14   14     33   47  
## 16   16     35   51  
## 18   18     37   55
```

### 3.2.3 Noch mehr Subsetting

- Mengen: `%in%` fungiert wie  $\in$  aus der Mengentheorie

```
df[df$low %in% c(5, 6, 7), ]  
  
##    low medium high  
## 5    5     24   29  
## 6    6     25   31  
## 7    7     26   33  
  
df[df$low %in% c(5, 6, 7) | df$high %in% c(23, 25), ]  
  
##    low medium high  
## 2    2     21   23  
## 3    3     22   25  
## 5    5     24   29  
## 6    6     25   31  
## 7    7     26   33
```

## 3.3 Daten exportieren

### 3.3.1 Workspace

Wenn wir mir externen Daten (also solchen, die wir nicht in R generiert und/oder nicht nur in R verwenden wollen) arbeiten, müssen wir uns für einen Speicherort entscheiden. Ganz allgemein wird R ein Standardverzeichnis festlegen. Schaut mal eures nach.

```
getwd()
```

```
## [1] "/Users/Maik/Library/Mobile Documents/com~apple~CloudDocs/Studium/Arbeit am SDP/Tut Inferenzstatistik
```

### 3.3.2 Workspace verändern

Wenn ihr das Arbeitsverzeichnis verändern wollt, könnt ihr dies mit `setwd()` machen. Wie genau ihr die Pfade festlegt, könnt ihr entweder im Internet nachschauen (das variiert ziemlich zwischen Windows und Mac) oder per Rstudio über den Tab “Files”

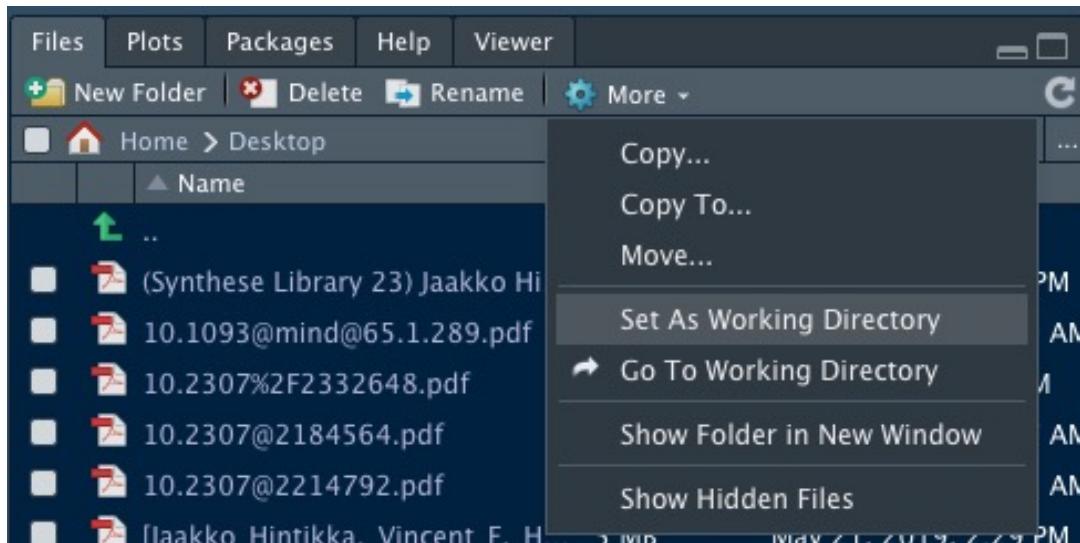


Abb. 3.1: RStudio zum Workspace festlegen

### 3.3.3 Workspace verändern

Selbst wenn ihr RStudio verwendet, um das Arbeitsverzeichnis zu verändern, solltet ihr die Codezeile aus der Konsole in eurer Skript kopieren, damit ihr das beim nächsten Mal nicht nochmal erledigen müsst, sondern einfach den Code ausführen könnt.

```
setwd("~/Desktop")
getwd()

## [1] "/Users/Maik/Desktop"
```

### 3.3.4 write.csv/write.table

```
write.csv(ObjektNameInR, "PFAD", WeitereParameter)
```

- weitere Parameter
  - Zeilennummerierung übernehmen: `row.names=TRUE/FALSE`

- Die Zeilennummerierung wird als erste Spalte übernommen. Die Spaltenbezeichnungen werden dementsprechend nach links verschoben, sodass die col.names manuell angepasst werden müssen.
- Dezimalstellentrennungszeichen bestimmen: dec=„Trennungszeichen“
- Separatoren definieren: sep = „Separator“
- Strings in Anführungszeichen einschließen: quote = TRUE/FALSE

### 3.3.5 Übersicht

```
# Zeilennummerierung von R wird nicht in die neue Tabelle übernommen
write.table(d, "tab.csv", row.names = F)
# Erstellt eine Tabelle, die Tabstops als Zellentrenner benutzt
write.table(d, "tab.csv", sep = "\t")
# Erstellt eine Tabelle mit Kommas als Dezimalstellentrenner
write.table(d, "tab.csv", dec = ",")
# Erstellt eine Tabelle mit Semikolons als Separatoren
write.table(d, "tab.csv", sep = ";")
# NAs werden als "Nicht vorhanden" bezeichnet
write.table(d, "tab.csv", na = "Nicht vorhanden")
```

### 3.3.6 Datenblatt speichern

Als nächstes können wir unseren Dataframe im Arbeitsverzeichnis unter dem angegebenen Namen abspeichern. Falls ihr andere Dateiformate verwenden wollt, schaut euch mal `write.table` an.

```
head(df)

##   low medium high
## 1   1     20   21
## 2   2     21   23
## 3   3     22   25
## 4   4     23   27
## 5   5     24   29
## 6   6     25   31

write.csv(df, "df_test.csv", row.names = F, quote = F)
```

## 3.4 Daten einlesen

### 3.4.1 Interlude: Dateiformate

Dateiformate

- Jede Datei wird in einem Dateiformat abgespeichert
- Endungen bestimmen die Art (z.B. ein Programm), mit welcher der Computer eine Datei öffnet
- Gängige Dateiendungen für unsere Zwecke -Comma-separated values: .csv
  - Tab-separated values: .tsv

- Text-Datei: .txt
- Unbestimmte Datei: .dat
- Excel: xls
  - R kann xls nur mit bestimmten Paketen lesen

### 3.4.2 `read.table()`

- Voraussetzung: einzulesende Datei befindet sich im Arbeitsverzeichnis bzw. in einem im Arbeitsverzeichnis liegenden Unterordner
- wichtige optionale Argumente von `read.table/read.csv`
  - `header = TRUE/FALSE`
  - `sep = "Separator für Einträge"`
    - oft verwendet: Komma, Semikolon, Leerzeichen oder Tabstopps
  - `dec = "Dezimalzahl trennungszeichen"`
    - beachte: R verwendet den Punkt und nicht das Komma als Dezimaltrenner
  - `na.strings = "Zeichenkette, die als NA definiert werden soll"` (oder Vektor mit mehreren Strings)
  - `skip = Zeile, bis zu der übersprungen werden soll`
  - `blank.lines.skip = TRUE`
  - `comment.char = "Kommentarzeichen"`

### 3.4.3 Ein Beispiel

Ich habe hier ein einfaches Beispiel, bei dem man keine weiteren Parameter angeben muss, damit ihr sehen könnt, wie es funktioniert.

Bei Gelingen sieht das dann so aus:

```
d <- read.csv("docs/data/tut3/example.csv")
head(d)

##           id      item judgment tester years months
## 1 3536fed0235c7b34a33ddf06fb91b390 schaffen4_at      5 mailin    5   1
## 2 3536fed0235c7b34a33ddf06fb91b390 schaffen1_non      5 mailin    5   1
## 3 3536fed0235c7b34a33ddf06fb91b390 cleft2_non      5 mailin    5   1
## 4 3536fed0235c7b34a33ddf06fb91b390 appRel3_non      5 mailin    5   1
## 5 3536fed0235c7b34a33ddf06fb91b390 appRel9_non      4 mailin    5   1
## 6 3536fed0235c7b34a33ddf06fb91b390 gewinnen2_non      4 mailin    5   1
##   gender     issue      trigger      itemid      lex
## 1       w at-issue      soft schaffen4 schaffen
## 2       w non-at-issue      soft schaffen1 schaffen
## 3       w non-at-issue      hard cleft2      cleft
## 4       w non-at-issue appositive RC appRel3 appRel
## 5       w non-at-issue appositive RC appRel9 appRel
## 6       w non-at-issue      soft gewinnen2 gewinnen
```

### 3.4.4 Vorsicht

Beachtet man die genannten Parameter nicht, bekommt man schnell unbrauchbaren Datensalat:

```
head(read.csv("docs/data/tut3/raw.txt"))

##          blablablabla.wejfuwejfqnqj.qfnwejfn
## 1      1\t1\ta\tdass der Peter den Hans mag.
## 2      1\t1\tb\tdass den Hans der Peter mag.
## 3  1\t1\tc\tdass der Peter sicherlich den Hans mag.
## 4 1\t1\td\tdass den Hans sicherlich der Peter mag.
## 5      1\t2\ta\tdass die Frau den Bäcker liebt.
## 6      1\t2\tb\tdass den Bäcker die Frau liebt.
```

### 3.4.5 Übungen

1. Lese die folgenden Dateien in R ein (Objektzuweisung nicht vergessen!):

- alldata.dat
- Beispiel.csv
- Tutorium\_dat3.txt
- NHIS\_2007\_data4.csv
- lungen.txt
- Tutorium\_4.dat
- raw.txt
- raw2.txt
- master.txt

### 3.4.6 Lösungen

```
# "alldata.dat"
d1 <- read.table("docs/data/tut3/alldata.dat", header = T)
head(d1)

##           id geo subject variant experiment item condition judgement
## 1 m-1987-EMusik bla      1      1        9       4      a       6
## 2 m-1987-EMusik bla      1      1        8       2      b       7
## 3 m-1987-EMusik bla      1      1        7      10      b       7
## 4 m-1987-EMusik bla      1      1        9       3      d       1
## 5 m-1987-EMusik bla      1      1        1       8      d     NA
## 6 m-1987-EMusik bla      1      1        8       9      a       6

# "Beispiel.csv"
d2 <- read.csv("docs/data/tut3/Beispiel.csv", sep = ";", header = T)
head(d2)

##   subject item condition judgement F1 F2
## 1      1    10      b         3   0   1
## 2      1     4      d         4   1   1
## 3      1    15      c         7   1   0
## 4      1     9      a         2   0   0
## 5      1    16      d         1   1   1
## 6      1    13      a         7   0   0
```

### 3.4.7 Lösungen

```
# "Tutorium_dat3.txt"
d3 <- read.table("docs/data/tut3/Tutorium_dat3.txt", header = T, sep = "\t")
head(d3)

##   ProbandIn Geschlecht Treatment Zeit1 Zeit2 Zeit3
## 1           f          1  19.7  19.7 20.5
## 2           f          2  14.2  16.9 17.3
## 3           f          3  11.2  14.2 17.2
## 4           f          1  18.8  20.3 28.5
## 5           f          2  14.8  25.3 30.2
## 6           f          3  11.1  12.1 16.2

# "NHIS_2007_data4.csv"
d4 <- read.csv("docs/data/tut3/NHIS_2007_data4.csv", header = T, sep = ";",
               dec = ",")
head(d4)

##   HHX FMX FPX SEX   BMI SLEEP educ height weight
## 1 16   1   2   1 33.36     8   16    74   260
## 2 20   1   1   1 26.54     7   14    70   185
## 3 69   1   2   2 32.13     7   9    61   170
## 4 87   1   1   1 26.62     8   14    68   175
## 5 88   1   1   2 27.13     8   13    66   168
## 6 99   1   1   2 99.99    98   12    98   998
```

### 3.4.8 Lösungen

```
# "lungen.txt"
d5 <- read.table("docs/data/tut3/lungen.txt", header = T, sep = "|",
                 dec = ",")
head(d5)

##   Lungenkapazität Alter Größe Raucher Geschlecht Kaiserschnitt
## 1       6.475      6 62.1   no     male      no
## 2      11.125     14 71.0   no     male      no
## 3       4.800      5 56.9   no     male      no
## 4       7.325     11 70.4   no     male      no
## 5       8.875     15 70.5   no     male      no
## 6       6.800     11 59.2   no     male      no

# "Tutorium_4.dat"
d6 <- read.table("docs/data/tut3/Tutorium_4.dat", sep = ";", dec = ",",
                  header = F, na.strings = c("umgeknickt", "verdorrt"))
head(d6)

##   V1 V2  V3  V4  V5
## 1  1  1 19.7 19.7 20.5
## 2  1  2 14.2 18.9 27.3
## 3  1  3 11.2 24.2 37.2
## 4  2  1 18.8 20.3 28.5
## 5  1  2 14.8 25.3 30.2
## 6  1  3 11.1 12.1 16.2
```

### 3.4.9 Lösungen

```
# "raw2.txt"
d8 <- read.table("docs/data/tut3/raw2.txt", sep = "\t", skip = 1,
  comment.char = "%")
head(d8)

##   V1 V2 V3          V4
## 1  1  1  a      dass der Peter den Hans mag.
## 2  1  1  b      dass den Hans der Peter mag.
## 3  1  1  c  dass der Peter sicherlich den Hans mag.
## 4  1  1  d  dass den Hans sicherlich der Peter mag.
## 5  1  2  a      dass die Frau den Bäcker liebt.
## 6  1  2  b      dass den Bäcker die Frau liebt.

# "raw.txt"
d7 <- read.table("docs/data/tut3/raw.txt", sep = "\t", skip = 1,
  header = F)
head(d7)

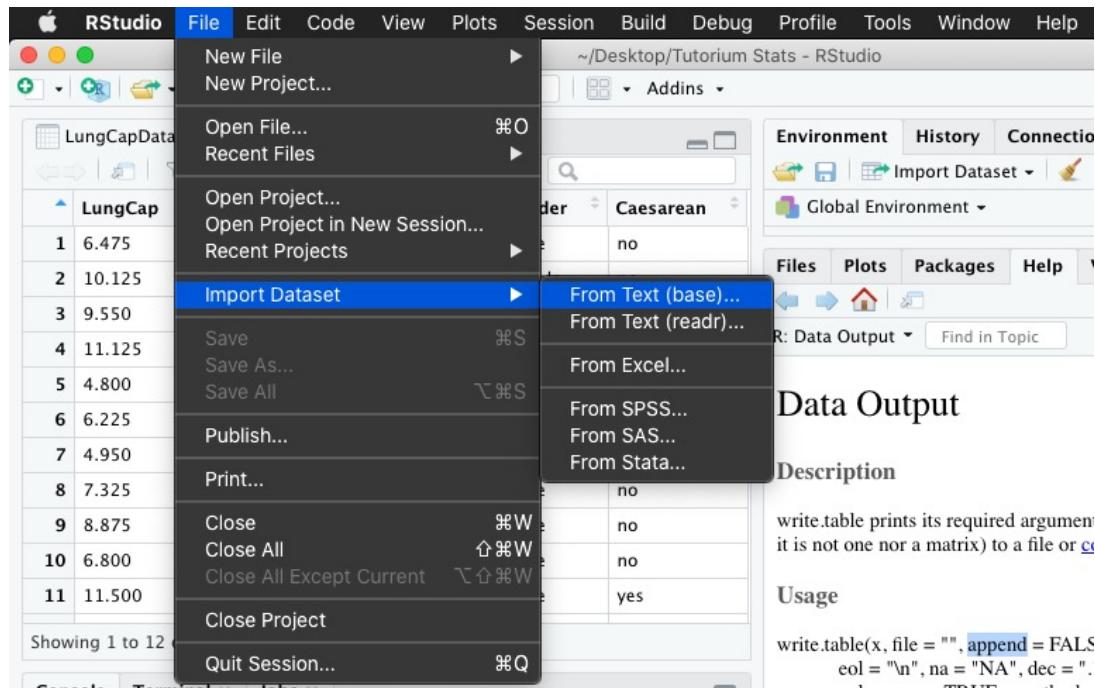
##   V1 V2 V3          V4
## 1  1  1  a      dass der Peter den Hans mag.
## 2  1  1  b      dass den Hans der Peter mag.
## 3  1  1  c  dass der Peter sicherlich den Hans mag.
## 4  1  1  d  dass den Hans sicherlich der Peter mag.
## 5  1  2  a      dass die Frau den Bäcker liebt.
## 6  1  2  b      dass den Bäcker die Frau liebt.
```

### 3.4.10 Lösungen

```
# "master.txt"
d9 <- read.table("docs/data/tut3/master.txt", sep = "&",
  comment.char = "$", dec = ",", header = T)
head(d9)

##   korinon pavalon tertiron anbolon velbidon
## 1 korinon    1.230 tertiron      ja      0
## 2 korinon    4.230 tertiron     nein      0
## 3 korinon    1.124 tertiron      ja      0
## 4 korinon    8.340 tertiron     nein      1
## 5 korinon    0.900 tertiron     nein      1
## 6 korinon    6.740 tertiron     nein      0
```

### 3.4.11 Daten einlesen mit GUI



### 3.4.12 Und Tschüss

Bis nächste Woche!

# 4

---

## Sitzung 4

### 4.1 Wiederholung

#### 4.1.1 Das Wichtigste vom letzten Mal

- Wir haben verschiedene Methoden besprochen, nur Teile einer Tabelle zu verwenden (“Subsetting”)
- Was bedeuten die unteren Befehle?

```
d[2:5, ]
d[d$wert == 14 | d$wert <= 2, ]
d[d$bub %in% c("bla", "bli"), ]
```

- Wir können Daten in R einlesen und aus R exportieren und so auf dem Computer speichern:

```
read.csv()
read.table()
write.csv()
write.table()
```

### 4.2 Übung

#### 4.2.1 Übung

1. Lest die Datei “workwithme.csv” als Objekt d ein und lasst euch die ersten 10 Zeilen anzeigen
2. Erstellt eine neue Spalte mit dem Namen “sum”, die die Zeilensumme der Spalten “aldi”, “lidl” und “penny” enthält
3. Erstellt eine neue Spalte “sumrank”: Wenn in der “sum”-Spalte ein Wert steht, der größer als 140 ist, dann enthält sie den Wert “high”, von 140 bis 125 “medium” und darunter “low”
4. Erstellt eine Spalte, die den Mittelwert der Werte aus den Spalten “aldi”, “lidl” und “penny” enthält

5. Erstellt eine Spalte, die den Monat des Jahres enthält. Benutzt dafür die `substring`-Funktion
6. Benennt die Spalten um
7. Speichert das Datenblatt als csv-Datei auf eurem Computer, benutzt die folgenden Parameter: Spaltentrenner = |, keine Anführungszeichen um Text, keine Zeilennamen. Benutzt die Hilfunktion, um die relevanten Argumente der `write.csv`-Funktion herauszufinden

#### 4.2.2 Lösung

1. Lest die Datei "workwithme.csv" als Objekt d ein und lasst euch die ersten 10 Zeilen anzeigen

```
d <- read.csv("docs/data/tut4/workwithme.csv")
head(d, 10) # alternative: d[1:10,]

##           time     aldi     lidl     penny   source     coeff
## 1 2018-01-01 49.41064 42.10573 44.69301 indirect 2.0289111
## 2 2018-01-02 51.51267 38.47846 55.16647 indirect 0.8333160
## 3 2018-01-03 44.75134 39.28813 46.71183 direct 2.1071974
## 4 2018-01-04 45.09636 42.36057 54.62832 indirect 2.1303495
## 5 2018-01-05 52.19682 36.08796 49.56332 indirect 0.7593176
## 6 2018-01-06 48.85863 39.54701 44.47574 direct 1.4346635
## 7 2018-01-07 44.31124 42.58359 49.07205 unknown 1.7395597
## 8 2018-01-08 49.72772 38.21966 37.36114 unknown 0.4596969
## 9 2018-01-09 49.90116 40.69519 64.80204 indirect 0.6560386
## 10 2018-01-10 51.66494 38.04120 68.73857 indirect 1.1486464
```

#### 4.2.3 Lösung

2. Erstellt eine neue Spalte mit dem Namen "sum", die die Zeilensumme der Spalten "aldi", "lidl" und "penny" enthält

```
d$sum <- d$aldi + d$lidl + d$penny
head(d)

##           time     aldi     lidl     penny   source     coeff      sum
## 1 2018-01-01 49.41064 42.10573 44.69301 indirect 2.0289111 136.2094
## 2 2018-01-02 51.51267 38.47846 55.16647 indirect 0.8333160 145.1576
## 3 2018-01-03 44.75134 39.28813 46.71183 direct 2.1071974 130.7513
## 4 2018-01-04 45.09636 42.36057 54.62832 indirect 2.1303495 142.0853
## 5 2018-01-05 52.19682 36.08796 49.56332 indirect 0.7593176 137.8481
## 6 2018-01-06 48.85863 39.54701 44.47574 direct 1.4346635 132.8814
```

#### 4.2.4 Lösung

3. Erstellt eine neue Spalte "sumrank": Wenn in der "sum"-Spalte ein Wert ist, der größer als 140 ist, dann enthält sie den Wert "high", von 140 bis 125 "medium" und darunter "low"

```
d$sumrank[d$sum > 140] <- "high"
d$sumrank[d$sum <= 140] <- "medium"
d$sumrank[d$sum <= 125] <- "low"
head(d[, 7:8])
```

```
##      sum sumrank
## 1 136.2094 medium
## 2 145.1576 high
## 3 130.7513 medium
## 4 142.0853 high
## 5 137.8481 medium
## 6 132.8814 medium
```

#### 4.2.5 Lösung

4. Erstellt eine Spalte, die den Mittelwert der Werte aus den Spalten “aldi”, “lidl” und “penny” enthält

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n a_i = \frac{a_1 + a_2 + \cdots + a_n}{n}$$

```
d$compmean <- (d$aldi + d$lidl + d$penny) / 3
head(d[, c(2, 3, 4, 7, 8, 9)])

##      aldi     lidl     penny      sum sumrank compmean
## 1 49.41064 42.10573 44.69301 136.2094 medium 45.40313
## 2 51.51267 38.47846 55.16647 145.1576 high 48.38587
## 3 44.75134 39.28813 46.71183 130.7513 medium 43.58377
## 4 45.09636 42.36057 54.62832 142.0853 high 47.36175
## 5 52.19682 36.08796 49.56332 137.8481 medium 45.94937
## 6 48.85863 39.54701 44.47574 132.8814 medium 44.29379
```

#### 4.2.6 Lösung

5. Erstellt eine Spalte, die den Monat des Jahres enthält. Benutzt dafür die `substring`-Funktion

```
d$month <- substring(d$time, 6, 7)
head(d[, c(1, 2, 3, 4, 7, 10)])

##      time     aldi     lidl     penny      sum month
## 1 2018-01-01 49.41064 42.10573 44.69301 136.2094 01
## 2 2018-01-02 51.51267 38.47846 55.16647 145.1576 01
## 3 2018-01-03 44.75134 39.28813 46.71183 130.7513 01
## 4 2018-01-04 45.09636 42.36057 54.62832 142.0853 01
## 5 2018-01-05 52.19682 36.08796 49.56332 137.8481 01
## 6 2018-01-06 48.85863 39.54701 44.47574 132.8814 01
```

#### 4.2.7 Lösung

6. Benennt die Spalten um

```
names(d) <- c("date", "markt1", "markt2", "markt3", "origin", "coef", "all", "level", "marktmean", "month")
```

7. Speichert das Datenblatt als csv-Datei auf eurem Computer, benutzt die folgenden Parameter: Spaltentrenner = |, keine Anführungszeichen um Text, keine Zeilennamen.

```
write.csv(d, "supermarktBS.csv", sep = "|", row.names = F, quote = F)
```

## 4.3 Tabellenüberblick

### 4.3.1 Struktur

```
dim(d) # alternativ: nrow(d), ncol(d)

## [1] 365 10

str(d)

## 'data.frame': 365 obs. of 10 variables:
## $ time : Factor w/ 365 levels "2018-01-01","2018-01-02",...: 1 2 3 4 5 6 7 8 9 10 ...
## $ aldi : num 49.4 51.5 44.8 45.1 52.2 ...
## $ lidl : num 42.1 38.5 39.3 42.4 36.1 ...
## $ penny : num 44.7 55.2 46.7 54.6 49.6 ...
## $ source : Factor w/ 3 levels "direct","indirect",...: 2 2 1 2 2 1 3 3 2 2 ...
## $ coeff : num 2.029 0.833 2.107 2.13 0.759 ...
## $ sum : num 136 145 131 142 138 ...
## $ sumrank : chr "medium" "high" "medium" "high" ...
## $ compmean: num 45.4 48.4 43.6 47.4 45.9 ...
## $ month : chr "01" "01" "01" "01" ...
```

### 4.3.2 Überblick über Zahlen

```
summary(d)

##      time          aldi          lidl          penny
## 2018-01-01: 1  Min.   :35.02  Min.   :33.93  Min.   :20.21
## 2018-01-02: 1  1st Qu.:46.57  1st Qu.:38.70  1st Qu.:38.29
## 2018-01-03: 1  Median :49.74  Median :40.04  Median :45.88
## 2018-01-04: 1  Mean    :49.80  Mean    :40.10  Mean    :45.49
## 2018-01-05: 1  3rd Qu.:52.78  3rd Qu.:41.33  3rd Qu.:51.93
## 2018-01-06: 1  Max.   :63.52  Max.   :47.83  Max.   :81.56
## (Other)   :359
##      source         coeff          sum        sumrank
## direct   :106  Min.   :-0.4144  Min.   :103.3  Length:365
## indirect:210  1st Qu.: 0.6830  1st Qu.:127.4  Class  :character
## unknown  :49   Median : 0.9468  Median :135.4  Mode   :character
##                  Mean   : 0.9746  Mean   :135.4
##                  3rd Qu.: 1.2882  3rd Qu.:142.7
##                  Max.   : 2.1701  Max.   :171.2
##
##      compmean        month
## Min.   :34.44  Length:365
## 1st Qu.:42.48  Class  :character
## Median :45.15  Mode   :character
## Mean   :45.13
## 3rd Qu.:47.56
## Max.   :57.07
##
```

### 4.3.3 Häufigkeiten

```
table(d$sumrank)

## 
##   high    low medium
##   122     65    178

# Verteilungen von summenwerten auf monate abgebildet
overview <- table(d$sumrank, d$month)
overview

## 
##      01 02 03 04 05 06 07 08 09 10 11 12
##   high    7 11 10 10 10  7 14 10 15 12  9  7
##   low     6  8  7  2  5  7  2  4  4  7  6  7
##   medium 18  9 14 18 16 16 15 17 11 12 15 17

(overview2 <- addmargins(overview))

## 
##      01 02 03 04 05 06 07 08 09 10 11 12 Sum
##   high    7 11 10 10 10  7 14 10 15 12  9  7 122
##   low     6  8  7  2  5  7  2  4  4  7  6  7 65
##   medium 18  9 14 18 16 16 15 17 11 12 15 17 178
##   Sum     31 28 31 30 31 30 31 31 30 31 30 31 365
```

### 4.3.4 Relative Häufigkeiten

```
overview3 <- prop.table(overview) * 100
overview3

## 
##      01          02          03          04          05          06          07
##   high  1.9178082 3.0136986 2.7397260 2.7397260 2.7397260 1.9178082 3.8356164
##   low   1.6438356 2.1917808 1.9178082 0.5479452 1.3698630 1.9178082 0.5479452
##   medium 4.9315068 2.4657534 3.8356164 4.9315068 4.3835616 4.3835616 4.1095890
## 
##      08          09          10          11          12
##   high  2.7397260 4.1095890 3.2876712 2.4657534 1.9178082
##   low   1.0958904 1.0958904 1.9178082 1.6438356 1.9178082
##   medium 4.6575342 3.0136986 3.2876712 4.1095890 4.6575342

round(prop.table(overview) * 100, digits = 2)

## 
##      01 02 03 04 05 06 07 08 09 10 11 12
##   high  1.92 3.01 2.74 2.74 2.74 1.92 3.84 2.74 4.11 3.29 2.47 1.92
##   low   1.64 2.19 1.92 0.55 1.37 1.92 0.55 1.10 1.10 1.92 1.64 1.92
##   medium 4.93 2.47 3.84 4.93 4.38 4.38 4.11 4.66 3.01 3.29 4.11 4.66
```

## 4.4 Erste statistische Berechnungen

### 4.4.1 Maße der zentralen Tendenz

Die wichtigsten statistischen Operationen sind als Funktionen in R enthalten. So auch das arithmetische Mittel und der Median

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n a_i = \frac{a_1 + a_2 + \cdots + a_n}{n}$$

$$\tilde{x} = \begin{cases} x_{\frac{n+1}{2}} & n \text{ odd} \\ \frac{1}{2} (x_{\frac{n}{2}} + x_{\frac{n}{2}+1}) & n \text{ even} \end{cases}$$

```
sample <- c(1, 1, 2, 2, 4, 6, 9)
mean(sample)

## [1] 3.571429
median(sample)

## [1] 2
```

#### 4.4.2 Meme My Stats



#### 4.4.3 Maße der Streuung

Gleiches gilt für die Streuungsmaße Standardabweichung und Median der absoluten Abweichung

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

$$MAD = \text{median}|x_i - \tilde{x}|$$

```
sample <- c(1, 1, 2, 2, 4, 6, 9)
sd(sample)

## [1] 2.992053
mad(sample, constant = 1)

## [1] 1
```

#### 4.4.4 Übung

1. Gibt es Werte in der “lidl”-Spalte, die mehr als 2 Standardabweichungen vom Mittelwert abweichen?
2. Erstellt eine Tabelle mit relativen Häufigkeiten für “source” abgebildet auf “sumrank”
3. Legt eine neue Spalte “coeffmedian” an, die den Median der Spalte “coeff” enthält
4. Legt eine neue Spalte an, die Informationen über das Jahresquartal enthält. Um die Monate (01, 02, 03, ...) in Zahlen (1, 2, 3, ...) umzuwandeln, könnt ihr `as.numeric` verwenden.

#### 4.4.5 Lösung

1. Gibt es Werte in der “lidl”-Spalte, die mehr als 2 Standardabweichungen vom Mittelwert abweichen?

```
d$lidl[d$lidl >= mean(d$lidl) + 2 * sd(d$lidl) |
d$lidl <= mean(d$lidl) - 2 * sd(d$lidl)]

## [1] 44.35339 35.54213 44.55232 35.88545 35.28163 46.54198 44.44375 35.81564
## [9] 33.93010 44.88954 44.88695 35.88933 44.45852 35.95163 44.34326 44.81850
## [17] 45.40808 47.83175 35.70551 35.95866 35.74292
```

#### 4.4.6 Lösung

2. Erstellt eine Tabelle mit relativen Häufigkeiten für “source” abgebildet auf “sumrank”

```
tab <- table(d$source, d$sumrank)
(propctab <- prop.table(tab) * 100)

##
##          high      low   medium
##  direct    11.506849  4.383562 13.150685
##  indirect  16.164384 10.684932 30.684932
##  unknown    5.753425  2.739726  4.931507
```

3. Legt eine neue Spalte “coeffmedian” an, die den Median der Spalte “coeff” enthält

```
d$coeffmedian <- median(d$coeff)
tail(d[, c(6, 11)])
```

```
##      coeff coeffmedian
## 360 1.3716517 0.9467506
## 361 0.1725648 0.9467506
## 362 0.9713443 0.9467506
## 363 0.8421736 0.9467506
## 364 0.6721796 0.9467506
## 365 1.2695549 0.9467506
```

#### 4.4.7 Lösung

4. Legt eine neue Spalte an, die Informationen über das Jahresquartal enthält. Um die Monate (01, 02, 03, ...) in Zahlen (1, 2, 3, ...) umzuwandeln, könnt ihr `as.numeric` verwenden.

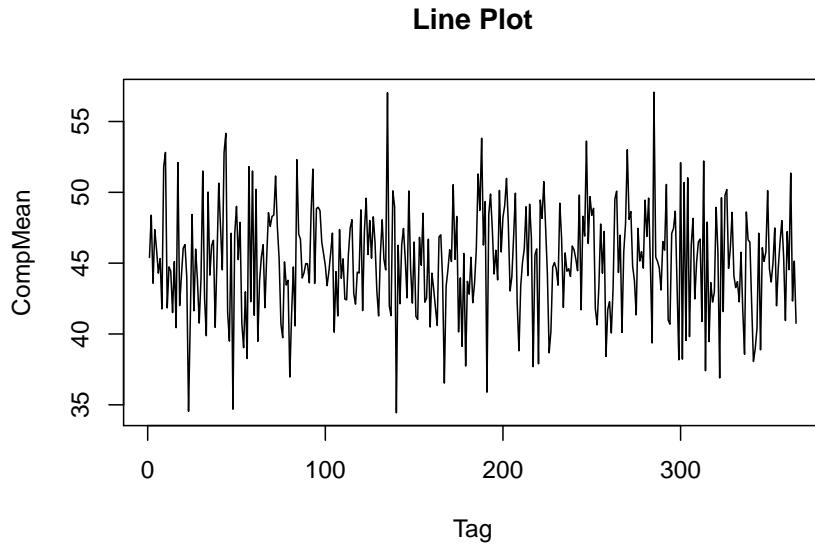
```
q <- c(1, 2, 3)
d$quarter[as.numeric(d$month) %in% q] <- 1
d$quarter[as.numeric(d$month) %in% (q + 3)] <- 2
d$quarter[as.numeric(d$month) %in% (q + 6)] <- 3
d$quarter[as.numeric(d$month) %in% (q + 9)] <- 4
# head(d[,c(10,length(d))], 3)
# tail(d[,c(10,length(d))], 3)
table(d$quarter)
```

```
## 
## 1 2 3 4
## 90 91 92 92
```

## 4.5 Nur Kurz: Plots

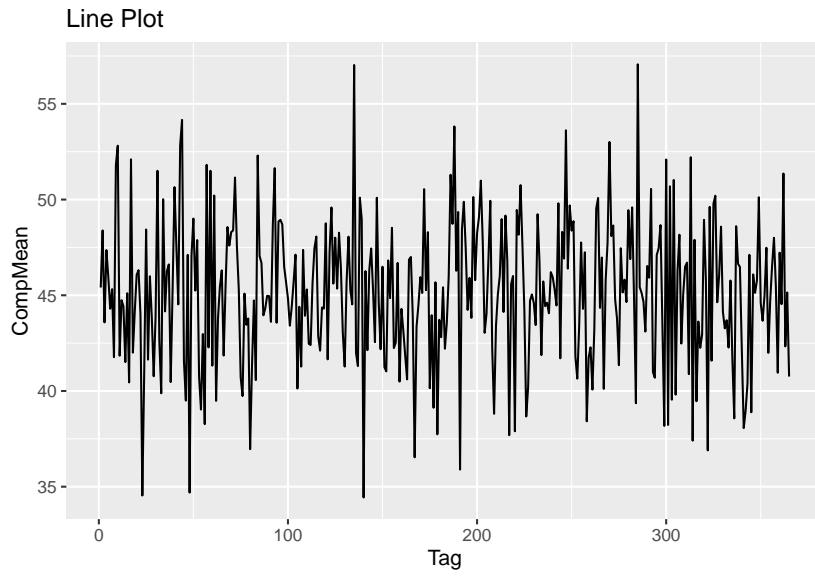
### 4.5.1 Ein Plot zum Abschluss

```
d$time <- as.ts(d$time) # convert to time series
plot(d$time, d$compmean, type = "l",
     xlab = "Tag", ylab = "CompMean", main = "Line Plot")
```



#### 4.5.2 Ein Plot zum Abschluss

```
library(ggplot2)
d$time <- as.ts(d$time) # convert to time series
ggplot(d, aes(x = time, y = compmean)) +
  geom_line() +
  labs(x = "Tag", y = "CompMean", title = "Line Plot")
```



#### 4.5.3 session no longer in session

Schönes Wochenende!

## Sitzung 5

### 5.1 Wiederholung

#### 5.1.1 Das Wichtigste vom letzten Mal

- Wir haben uns verschiedene Funktionen angesehen, die in der Statistik benutzt werden

```
# zentrale Tendenz  
mean()  
median()  
  
# Streuung  
sd()  
mad()
```

#### 5.1.2 Das Wichtigste vom letzten Mal

- Wir haben gelernt, Häufigkeiten zählen zu lassen

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

#### 5.1.3 Das Wichtigste vom letzten Mal

- Wir haben uns die Struktur von Datenblättern angesehen

```
head(diamonds, 3)  
  
##   carat      cut color clarity depth table price     x     y     z  
## 1 0.23    Ideal     E     SI2  61.5    55   326 3.98 2.43  
## 2 0.21  Premium     E     SI1  59.8    61   326 3.89 2.31  
## 3 0.23     Good     E     VS1  56.9    65   327 4.05 2.31  
  
str(diamonds)  
  
## 'data.frame': 53940 obs. of 10 variables:  
## $ carat : num 0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...  
## $ cut    : Ord.factor w/ 5 levels "Fair" < "Good" < ... : 5 4 2 4 2 3 3 3 1 3 ...
```

```
## $ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 2 5 ...
## $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7 3 4 5 ...
## $ depth   : num  61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
## $ table   : num  55 61 65 58 58 57 57 55 61 61 ...
## $ price   : int  326 326 327 334 335 336 336 337 337 338 ...
## $ x       : num  3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
## $ y       : num  3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
## $ z       : num  2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
```

### 5.1.4 Das Wichtigste vom letzten Mal

```
summary(diamonds)
```

```
##      carat          cut      color      clarity      depth
## Min. :0.2000    Fair     : 1610    D: 6775    SI1     :13065    Min.  :43.00
## 1st Qu.:0.4000  Good    : 4906    E: 9797    VS2     :12258    1st Qu.:61.00
## Median :0.7000  Very Good:12082   F: 9542    SI2     : 9194    Median :61.80
## Mean   :0.7979  Premium  :13791    G:11292    VS1     : 8171    Mean   :61.75
## 3rd Qu.:1.0400  Ideal    :21551    H: 8304    VVS2    : 5066    3rd Qu.:62.50
## Max.  :5.0100                    I: 5422    VVS1    : 3655    Max.   :79.00
##                               J: 2808    (Other): 2531
##      table          price         x         y
## Min.  :43.00    Min.   : 326    Min.   : 0.000  Min.   : 0.000
## 1st Qu.:56.00   1st Qu.: 950    1st Qu.: 4.710  1st Qu.: 4.720
## Median :57.00   Median : 2401   Median : 5.700  Median : 5.710
## Mean   :57.46   Mean   : 3933   Mean   : 5.731  Mean   : 5.735
## 3rd Qu.:59.00   3rd Qu.: 5324   3rd Qu.: 6.540  3rd Qu.: 6.540
## Max.  :95.00   Max.   :18823   Max.   :10.740  Max.   :58.900
##
##      z
## Min.  : 0.000
## 1st Qu.: 2.910
## Median : 3.530
## Mean   : 3.539
## 3rd Qu.: 4.040
## Max.   :31.800
##
```

## 5.2 Tabellen miteinander verbinden

### 5.2.1 Verbinden von zwei Tabellen

- Tabellen untereinander zusammenfügen:

```
rbind(Tabelle1, Tabelle2)
```

- Tabellen nebeneinander zusammenfügen:

```
cbind(Tabelle1, Tabelle2)
```

### 5.2.2 Übung

1. Lest die folgenden Tabellen in R ein

- Briefe.csv
  - Briefe2.csv
  - Tagebuecher.csv
  - Tagebuecher2.csv
2. Verbindet alle Tabellen sinnvoll miteinander

### 5.2.3 Lösung

1. Lest die Tabellen in R ein

```
d1 <- read.csv("docs/data/tut5/Briefe.csv", sep = ";", dec = ",")  
d2 <- read.csv("docs/data/tut5/Briefe2.csv", sep = ";", dec = ",")  
d3 <- read.csv("docs/data/tut5/Tagebuecher.csv", sep = ";", dec = ",")  
d4 <- read.csv("docs/data/tut5/Tagebuecher2.csv", sep = ";", dec = ",")  
  
head(d1, 3)  
  
##           Filename Segment   WC    WPS Sixltr Pronoun   I   We  
## 1 m-Rollet_Alexander-BR-1844_1906.txt      1 500 19.23  32.4  4.0 0.0 0.4  
## 2 m-Rollet_Alexander-BR-1844_1906.txt      2 500 18.52  32.8  7.6 3.6 0.6  
## 3 m-Rollet_Alexander-BR-1844_1906.txt      3 500 41.67  28.8  9.2 0.0 5.6  
##   Self You  
## 1  0.4 2.2  
## 2  4.2 1.6  
## 3  5.6 0.8  
  
head(d2, 3)  
  
##   Other Preps Affect Posemo Negemo Past Present Future  
## 1  3.6  9.4   3.2   2.4   0.8  1.2   1.6   0.2  
## 2  3.2  7.2   3.6   3.2   0.2  2.2   2.2   0.2  
## 3  3.2  7.2   2.0   2.0   0.0  2.4   2.8   1.4
```

### 5.2.4 Lösung

2. Verbindet alle Tabellen sinnvoll miteinander

```
dneu1 <- cbind(d1, d2) # schreibt die Tabellen nebeneinander  
dneu2 <- cbind(d3, d4) # schreibt die Tabellen nebeneinander  
dneuges <- rbind(dneu1, dneu2) # schreibt die Tabellen untereinander  
head(dneuges)  
  
##           Filename Segment   WC    WPS Sixltr Pronoun   I   We  
## 1 m-Rollet_Alexander-BR-1844_1906.txt      1 500 19.23  32.4  4.0 0.0 0.4  
## 2 m-Rollet_Alexander-BR-1844_1906.txt      2 500 18.52  32.8  7.6 3.6 0.6  
## 3 m-Rollet_Alexander-BR-1844_1906.txt      3 500 41.67  28.8  9.2 0.0 5.6  
## 4 m-Rollet_Alexander-BR-1844_1906.txt      4 500 31.25  36.2  5.8 0.0 1.2  
## 5 m-Rollet_Alexander-BR-1844_1906.txt      5 500 23.81  34.8  5.0 0.0 0.6  
## 6 m-Rollet_Alexander-BR-1844_1906.txt      6 500 25.00  36.0  8.4 4.8 0.6  
##   Self You Other Preps Affect Posemo Negemo Past Present Future  
## 1  0.4 2.2  3.6  9.4   3.2   2.4   0.8  1.2   1.6   0.2  
## 2  4.2 1.6  3.2  7.2   3.6   3.2   0.2  2.2   2.2   0.2  
## 3  5.6 0.8  3.2  7.2   2.0   2.0   0.0  2.4   2.8   1.4  
## 4  1.2 1.8  4.4  7.2   3.8   3.4   0.4  2.4   3.4   2.4  
## 5  0.6 2.2  3.4  6.6   3.4   3.4   0.0  1.6   2.2   0.6  
## 6  5.4 1.6  2.6  11.4  4.6   3.8   0.8  1.8   3.6   0.2
```

## 5.3 Fehlende Werte

### 5.3.1 NAs

Fehlende Werte führen dazu, dass Berechnungen ins Leere laufen

```
NA + 8
## [1] NA
sum(c(1, NA, 3, 5))
## [1] NA
```

Zwar ist dies bei intern erstellten Daten kein Problem, aber was ist, wenn Werte in einem Datenset fehlen wie hier:

```
d <- read.table("docs/data/tut5/alldata.dat", header = T)
head(d)

##           id geo subject variant experiment item condition judgement
## 1 m-1987-EMusik bla      1       1        9        4      a       6
## 2 m-1987-EMusik bla      1       1        8        2      b       7
## 3 m-1987-EMusik bla      1       1        7       10      b       7
## 4 m-1987-EMusik bla      1       1        9        3      d       1
## 5 m-1987-EMusik bla      1       1        1        8      d     NA
## 6 m-1987-EMusik bla      1       1        8        9      a       6
mean(d$judgement)
## [1] NA
```

### 5.3.2 NAs finden

`anyNA` gibt an, ob fehlende Werte enthalten sind:

```
anyNA(c(1, NA, 3, 5))
## [1] TRUE
anyNA(d$judgement)
## [1] TRUE
```

`is.na` geht das ganze Objekt durch, durchsucht es nach NAs und gibt deren Präsenz über Wahrheitswerte an:

```
is.na(c(1, NA, 3, 5))
```

```
## [1] FALSE  TRUE FALSE FALSE
```

Da Tabellen in der Regel sehr viele Werte enthalten, macht es hier Sinn, sich das Ergebnis der Suche als Tabelle ausgeben zu lassen:

```
table(is.na(d$judgement)) # summary statt table auch möglich
```

```
##
## FALSE  TRUE
## 3836     4
```

### 5.3.3 alle Zeilen mit NAs

```
d[!complete.cases(d), ]
```

	id	geo	subject	variant	experiment	item	condition	judgement
## 5	m-1987-EMusik	bla	1	1	1	8	d	NA
## 205	m-1983-EnglSport	bla	3	1	7	12	d	NA
## 2431	m-86-psych	???	31	6	7	12	c	NA
## 3284	m-1987-geschengl	???	42	7	8	22	d	NA

### 5.3.4 Prozent NA

Zeigt für jede Spalte an, wie viel Prozent der Daten NAs sind

```
(apply(is.na(d), 2, sum) / length(d[, 1])) * 100
```

	id	geo	subject	variant	experiment	item	condition
## judgement	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
##	0.1041667						

### 5.3.5 NAs überspringen

Möglichkeit 1:

```
mean(d$judgement[is.na(d$judgement) == F])
```

```
## [1] 4.64025
```

Möglichkeit 2: `na.rm` ist ein logisches Argument, das in vielen Funktionen zusätzlich angegeben werden kann, um fehlende Werte bei der Berechnung auszusparen

```
mean(d$judgement, na.rm = T)
```

```
## [1] 4.64025
```

```
sd(d$judgement, na.rm = T)
```

```
## [1] 2.435801
```

### 5.3.6 NAs entfernen

Alternativ kann man alle Zeilen, die fehlende Werte aufweisen, aus dem Datensatz entfernen

```
d <- na.omit(d)
mean(d$judgement)
```

```
## [1] 4.64025
```

## 5.4 Faktoren

### 5.4.1 Vektoren mit Faktorstufen

Vor allem wichtig in der statistischen Berechnung und bei Plots, weil Faktoren in R der Datentyp von kategorialen und ordinalen Variablen sind (für letzteres eigentlich `ordered = TRUE`, kann aber regelmäßig weggelassen werden)

```
calories <- c("high", "high", "low", "medium", "high", "medium")
levels(calories)

## NULL

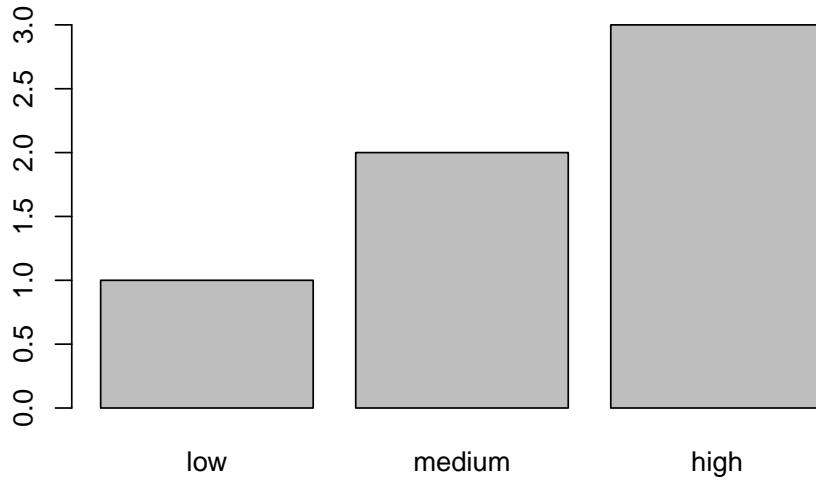
calories <- factor(calories)
levels(calories)

## [1] "high"    "low"     "medium"
calories <- factor(calories, levels = c("low", "medium", "high"))
levels(calories)

## [1] "low"     "medium"   "high"
```

### 5.4.2 Plots

```
# funktioniert nicht!
# plot(as.character(calories))
plot(calories)
```



### 5.4.3 Level bleiben erhalten

Selbst wenn alle Instanzen eines Levels wegfallen, bleibt das Level im Faktor enthalten:

```
calories <- calories[calories %in% c("low", "medium")]
calories
## [1] low   medium medium
## Levels: low medium high
```

Dies kann mithilfe von einer erneuten Anwendung von `factor` behoben werden. Alternativ kann auch die `droplevels`-Funktion verwendet werden.

```
calories <- factor(calories)
calories
## [1] low   medium medium
## Levels: low medium
```

#### 5.4.4 Konvertierungen

weitere wichtige Konvertierungen:

- `as.numeric()`
- `as.logical()`
- `as.data.frame()`

```
str(calories)
## Factor w/ 2 levels "low","medium": 1 2 2
calories <- as.character(calories)
str(calories)
## chr [1:3] "low" "medium" "medium"
calories <- as.factor(calories)
str(calories)
## Factor w/ 2 levels "low","medium": 1 2 2
```

#### 5.4.5 summary()

```
calories <- factor(c("high", "high", "low", "medium", "high", "medium"),
  levels = c("low", "medium", "high"))
summary(calories) # funktioniert wie table()

##    low medium   high
##      1      2      3

summary(as.character(calories))

##    Length   Class   Mode
##       6 character character
```

#### 5.4.6 Fakoren in eingelesenen Daten

R konvertiert strings, also alphabetische Zeichenketten, automatisch zu Faktoren, wenn sie in eingelesenen Daten vorkommen. Um dies zu verhindern, verwendet man `stringsAsFactors = F`

```
read.csv("data.csv", stringsAsFactors = F)
```

### 5.4.7 Übungen II

1. Lese die Datei "titanic.csv" in R ein und verschaffe dir einen Überblick über die Daten
2. Erstelle ein neues Objekt, das keine NAs enthält
3. Ersetze im Originalobjekt alle NAs durch den Mittelwert der jeweiligen Spalte

### 5.4.8 Lösungen II

1. Lese die Datei "titanic.csv" in R ein und verschaffe dir einen Überblick über die Daten

```
titanic <- read.csv("docs/data/tut5/titanic.csv")
head(titanic, 3)
```

```
##   PassengerId Pclass          Name     Sex   Age SibSp Parch
## 1           892      3 Kelly, Mr. James male 34.5    0    0
## 2           893      3 Wilkes, Mrs. James (Ellen Needs) female 47.0    1    0
## 3           894      2 Myles, Mr. Thomas Francis male 62.0    0    0
##   Ticket   Fare Cabin Embarked
## 1 330911 7.8292          Q
## 2 363272 7.0000          S
## 3 240276 9.6875          Q

str(titanic)

## 'data.frame': 418 obs. of 11 variables:
## $ PassengerId: int 892 893 894 895 896 897 898 899 900 901 ...
## $ Pclass      : int 3 3 2 3 3 3 2 3 3 ...
## $ Name        : Factor w/ 418 levels "Abbott, Master. Eugene Joseph",...: 210 409 273 414 182 370 85 58 5
## $ Sex         : Factor w/ 2 levels "female","male": 2 1 2 2 1 2 1 2 ...
## $ Age         : num 34.5 47 62 27 22 14 30 26 18 21 ...
## $ SibSp       : int 0 1 0 0 1 0 0 1 0 2 ...
## $ Parch       : int 0 0 0 0 1 0 0 1 0 0 ...
## $ Ticket      : Factor w/ 363 levels "110469","110489",...: 153 222 74 148 139 262 159 85 101 270 ...
## $ Fare        : num 7.83 7 9.69 8.66 12.29 ...
## $ Cabin       : Factor w/ 77 levels "", "A11", "A18", ...: 1 1 1 1 1 1 1 1 1 1 ...
## $ Embarked    : Factor w/ 3 levels "C", "Q", "S": 2 3 2 3 3 2 3 1 3 ...
```

### 5.4.9 Lösungen II

2. Erstelle ein neues Objekt, das keine NAs enthält

```
table(is.na(titanic))
```

```
##
## FALSE  TRUE
## 4511   87
```

```
table(is.na(titanic$Age))
```

```
##
## FALSE  TRUE
## 332    86
```

```
titanicsub <- na.omit(titanic)
dim(titanic) # or: nrow()

## [1] 418 11

dim(titanicsub)

## [1] 331 11
```

### 5.4.10 Lösungen II

3. Ersetze im Originalobjekt alle NAs durch den Mittelwert der jeweiligen Spalte

```
# durchschauen mit:
# titanic[!complete.cases(titanic),]
titanic[c(11, 23, 30, 34, 37, 153), ]

##   PassengerId Pclass          Name      Sex
## 11         902     3    Ilieff, Mr. Ylio male
## 23         914     1 Flegenheim, Mrs. Alfred (Antoinette) female
## 30         921     3        Samaan, Mr. Elias male
## 34         925     3 Johnston, Mrs. Andrew G (Elizabeth Lily" Watson) female
## 37         928     3           Roth, Miss. Sarah A female
## 153        1044    3       Storey, Mr. Thomas male
##   Age SibSp Parch     Ticket   Fare Cabin Embarked
## 11   NA    0     0  349220 7.8958      S
## 23   NA    0     0     PC 17598 31.6833      S
## 30   NA    2     0    2662 21.6792      C
## 34   NA    1     2 W./C. 6607 23.4500      S
## 37   NA    0     0  342712 8.0500      S
## 153 60.5    0     0    3701    NA      S

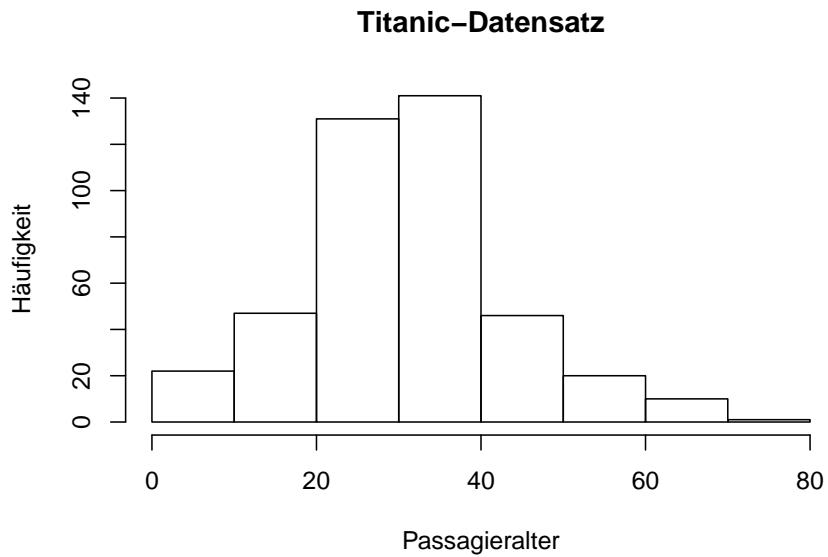
titanic$Age[is.na(titanic$Age)] <- mean(titanic$Age, na.rm = T)
titanic$Fare[is.na(titanic$Fare)] <- mean(titanic$Fare, na.rm = T)
anyNA(titanic)

## [1] FALSE
```

## 5.5 Nur Kurz: Plots

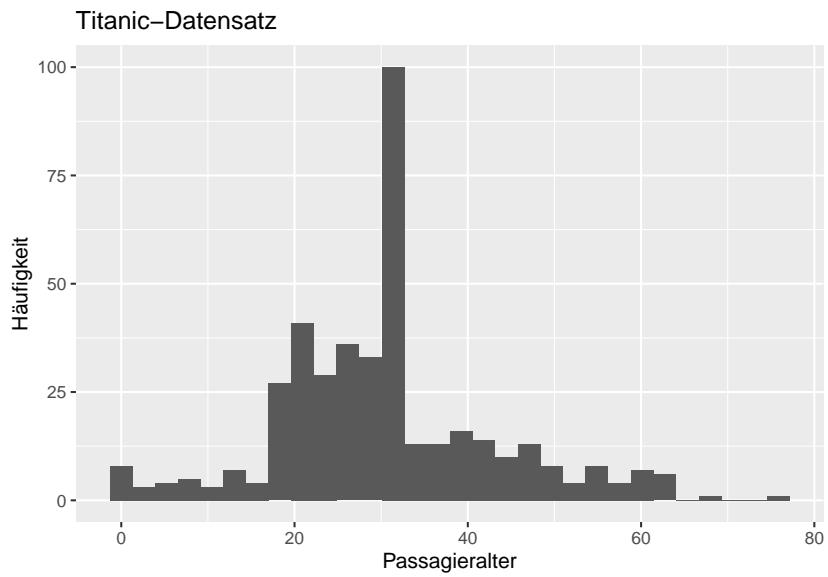
### 5.5.1 Ein Plot zum Abschluss

```
hist(titanic$Age, main = "Titanic-Datensatz",
      xlab = "Passagieralter", ylab = "Häufigkeit")
```



### 5.5.2 Ein Plot zum Abschluss

```
library(ggplot2)
ggplot(titanic, aes(x = Age)) +
  geom_histogram() +
  labs(title = "Titanic–Datensatz",
       x = "Passagieralter", y = "Häufigkeit")
```



**5.5.3 Friday Noon Fever**

Bis in einer Woche!

# 6

---

## Sitzung 6

### 6.1 Wiederholung

#### 6.1.1 Das Wichtigste vom letzten Mal

Tabellen verknüpfen

```
cbind()  
rbind()
```

Faktoren und NAs

```
a <- c(1, 2, 3, 4, 5, 4, 3, 5, NA, 4, 2, 1, 2, 3, 3, 6, 4)  
b <- factor(a)  
b  
  
## [1] 1    2    3    4    5    4    3    5    <NA> 4    2    1    2    3    3  
## [16] 6    4    4    5    6  
## Levels: 1 2 3 4 5 6  
  
anyNA(b)  
  
## [1] TRUE  
  
b <- na.omit(b)  
length(b) == length(a) - 1  
  
## [1] TRUE
```

### 6.2 Pakete und mehr Statistik

#### 6.2.1 Viel Tipparbeit

```
length(a)  
  
## [1] 16
```

```
mean(a)
## [1] 3.25
sd(a)
## [1] 1.437591
median(a)
## [1] 3
mad(a)
## [1] 1.4826
min(a)
## [1] 1
max(a)
## [1] 6
range(a)
## [1] 1 6
```

### 6.2.2 Wenig Tipparbeit (mit Paketen)

```
describe(a)
##    vars   n  mean    sd median trimmed  mad min max range skew kurtosis    se
## X1     1 16 3.25 1.44      3    3.21 1.48    1   6     5 0.09 -1.01 0.36
```

### 6.2.3 Pakete installieren

Pakete installieren (muss nur einmal gemacht werden):

```
install.packages("psych")
```

Pakete einbinden, damit die darin enthaltenen Funktionen verwendet werden können (muss in jeder neuen R-Session gemacht werden; am besten an den Anfang des Skripts schreiben)

```
library(psych)
```

```
describe(a)
```

```
##    vars   n  mean    sd median trimmed  mad min max range skew kurtosis    se
## X1     1 16 3.25 1.44      3    3.21 1.48    1   6     5 0.09 -1.01 0.36
```

### 6.2.4 Interlude: Mein Skiptanfang

Meine Skipte fangen in der Regel so an wie unten.

Einige der Pakete werdet ihr im Laufe dieses Semesters kennenlernen; andere sind für euch entweder nicht wichtig, oder die enthaltenen Funktionen übersteigen den (statistischen) Seminarstoff.

```
### %#####
# <>Titel> #
### %#####
# rm(list = ls()) # empty environment
options(scipen = 999) # no scientific number notation

library(ez)
library(tidyverse)
library(psych) # describeBy
library(gridExtra) # grid.arrange for plots
library(ggpubr) # theme_pubr(margin = F) + labs_pubr()
library(data.table)
library(ordinal) # used for the glmm
library(lme4)
library(ggthemes)
library(tikzDevice) # ggplot-latex
library(xtable) # table-latex
```

### 6.2.5 describeBy()

Gibt die wichtigsten deskriptiv-statistischen Berechnungen (in unserem Fall idR für eine abhängige Variable) wieder. Da man es jedoch oft mit Experimenten zu tun hat, in denen mindestens zwei Bedingungen einer unabhängigen Variable Einfluss auf die Messung nehmen, gibt es außerdem describeBy

```
describeBy(diamonds$price, diamonds$cut)

##
## Descriptive statistics by group
## group: Fair
##   vars   n    mean      sd median trimmed     mad min    max range skew
## X1    1 1610 4358.76 3560.39    3282 3695.65 2183.13 337 18574 18237 1.78
##   kurtosis    se
## X1    3.07 88.73
## -----
## group: Good
##   vars   n    mean      sd median trimmed     mad min    max range skew
## X1    1 4906 3928.86 3681.59 3050.5 3251.51 2853.26 327 18788 18461 1.72
##   kurtosis    se
## X1    3.04 52.56
## -----
## group: Very Good
##   vars   n    mean      sd median trimmed     mad min    max range skew
## X1    1 12082 3981.76 3935.86    2648 3243.22 2855.49 336 18818 18482 1.6
##   kurtosis    se
## X1    2.24 35.81
## -----
## group: Premium
##   vars   n    mean      sd median trimmed     mad min    max range skew
## X1    1 13791 4584.26 4349.2    3185 3822.23 3371.43 326 18823 18497 1.33
##   kurtosis    se
```

```
## X1      1.07 37.03
## -----
## group: Ideal
##   vars     n    mean      sd median trimmed     mad min    max range skew
## X1     1 21551 3457.54 3808.4    1810 2656.14 1630.86 326 18806 18480 1.84
##   kurtosis    se
## X1     2.98 25.94
```

### 6.2.6 describeBy()

Um den Output ein bisschen besser lesbar zu machen, eignen sich diese beiden optionalen Parameter:

- Anzahl der Nachkommastellen bestimmten: digits = Zahl
- Ouput der Funktion als Matrix: mat = TRUE

```
describeBy(diamonds$price, diamonds$cut, digits = 2, mat = T)
```

```
##      item   group1 vars     n    mean      sd median trimmed     mad min    max
## X11     1      Fair   1 1610 4358.76 3560.39 3282.0 3695.65 2183.13 337 18574
## X12     2      Good   1 4906 3928.86 3681.59 3050.5 3251.51 2853.26 327 18788
## X13     3 Very Good   1 12082 3981.76 3935.86 2648.0 3243.22 2855.49 336 18818
## X14     4 Premium   1 13791 4584.26 4349.20 3185.0 3822.23 3371.43 326 18823
## X15     5   Ideal   1 21551 3457.54 3808.40 1810.0 2656.14 1630.86 326 18806
##      range skew kurtosis    se
## X11 18237 1.78    3.07 88.73
## X12 18461 1.72    3.04 52.56
## X13 18482 1.60    2.24 35.81
## X14 18497 1.33    1.07 37.03
## X15 18480 1.84    2.98 25.94
```

### 6.2.7 describeBy()

Falls mehr als nur ein Faktor mit einbezogen werden sollen, muss das zweite Argument der Funktion eine list sein, die sämtliche dieser Faktoren enthält

```
describeBy(diamonds$price, list(diamonds$cut, diamonds$color),
           digits = 2, mat = T)
```

```
##      item   group1 group2 vars     n    mean      sd median trimmed     mad min
## X11     1      Fair      D   1 163 4291.06 3286.11 3730.0 3703.96 2043.02 536
## X12     2      Good      D   1 662 3405.38 3175.15 2728.5 2859.30 2674.61 361
## X13     3 Very Good      D   1 1513 3470.47 3523.75 2310.0 2774.01 2373.64 357
## X14     4 Premium      D   1 1603 3631.29 3711.63 2009.0 2881.00 1893.28 367
## X15     5   Ideal      D   1 2834 2629.09 3001.07 1576.0 1929.83 1184.60 367
## X16     6      Fair      E   1 224 3682.31 2976.65 2956.0 3164.34 2311.37 337
## X17     7      Good      E   1 933 3423.64 3330.70 2420.0 2799.42 2542.66 327
## X18     8 Very Good      E   1 2400 3214.65 3408.02 1989.5 2514.87 2014.11 352
## X19     9 Premium      E   1 2337 3538.91 3794.99 1928.0 2719.40 1712.40 326
## X110    10   Ideal      E   1 3903 2597.55 2956.01 1437.0 1904.70 1003.72 326
## X111    11      Fair      F   1 312 3827.00 3223.30 3035.0 3255.46 2404.04 496
## X112    12      Good      F   1 909 3495.75 3202.41 2647.0 2932.24 2403.29 357
## X113    13 Very Good      F   1 2164 3778.82 3786.12 2471.0 3052.83 2581.95 357
## X114    14 Premium      F   1 2331 4324.89 4012.02 2841.0 3605.12 2802.11 342
## X115    15   Ideal      F   1 3826 3374.94 3766.64 1775.0 2561.01 1481.12 408
## X116    16      Fair      G   1 314 4239.25 3609.64 3057.0 3537.06 2097.14 369
```

```

## X117 17 Good G 1 871 4123.48 3702.50 3340.0 3473.04 3177.21 394
## X118 18 Very Good G 1 2299 3872.75 3861.38 2437.0 3166.19 2610.86 354
## X119 19 Premium G 1 2924 4500.74 4356.57 2745.0 3738.69 2919.24 382
## X120 20 Ideal G 1 4884 3720.71 4006.26 1857.5 2948.86 1686.46 361
## X121 21 Fair H 1 303 5135.68 3886.48 3816.0 4489.27 2544.14 659
## X122 22 Good H 1 702 4276.25 4020.66 3468.5 3546.09 3231.33 368
## X123 23 Very Good H 1 1824 4535.39 4185.80 3734.0 3820.86 3670.92 337
## X124 24 Premium H 1 2360 5216.71 4466.19 4511.0 4551.71 4312.88 368
## X125 25 Ideal H 1 3115 3889.33 4013.38 2278.0 3121.10 2493.73 357
## X126 26 Fair I 1 175 4685.45 3730.27 3246.0 4015.75 1792.46 735
## X127 27 Good I 1 522 5078.53 4631.70 3639.5 4353.21 3814.73 351
## X128 28 Very Good I 1 1204 5255.88 4687.10 3888.0 4519.71 3925.18 336
## X129 29 Premium I 1 1428 5946.18 5053.75 4640.0 5303.56 5278.06 334
## X130 30 Ideal I 1 2093 4451.97 4505.15 2659.0 3643.44 3068.98 348
## X131 31 Fair J 1 119 4975.66 4050.46 3302.0 4280.01 2277.27 416
## X132 32 Good J 1 307 4574.17 3707.79 3733.0 4039.28 3546.38 335
## X133 33 Very Good J 1 678 5103.51 4135.65 4113.0 4574.19 3936.30 336
## X134 34 Premium J 1 808 6294.59 4788.94 5063.0 5833.48 4679.09 363
## X135 35 Ideal J 1 896 4918.19 4476.21 4096.0 4194.51 4272.85 340
## max range skew kurtosis se
## X11 16386 15850 1.90 3.81 257.39
## X12 18468 18107 2.03 5.42 123.41
## X13 18542 18185 1.91 3.92 90.59
## X14 18575 18208 1.78 2.96 92.70
## X15 18693 18326 2.55 6.92 56.37
## X16 15584 15247 1.88 4.00 198.89
## X17 18236 17909 1.91 4.09 109.04
## X18 18731 18379 1.99 4.21 69.57
## X19 18477 18151 1.92 3.36 78.50
## X110 18729 18403 2.58 7.18 47.32
## X111 17995 17499 2.04 4.85 182.48
## X112 18686 18329 2.07 5.34 106.22
## X113 18777 18420 1.73 2.88 81.39
## X114 18791 18449 1.46 1.65 83.10
## X115 18780 18372 1.92 3.25 60.89
## X116 18574 18205 1.80 3.01 203.70
## X117 18788 18394 1.63 2.80 125.45
## X118 18818 18464 1.56 2.17 80.53
## X119 18741 18359 1.28 0.89 80.57
## X120 18806 18445 1.59 1.93 57.33
## X121 18565 17906 1.46 1.58 223.27
## X122 18640 18272 1.58 2.25 151.75
## X123 18803 18466 1.39 1.54 98.01
## X124 18795 18427 1.13 0.62 91.94
## X125 18760 18403 1.58 2.19 71.91
## X126 18242 17507 1.73 2.67 281.98
## X127 18707 18356 1.16 0.39 202.72
## X128 18500 18164 1.16 0.43 135.08
## X129 18823 18489 0.89 -0.26 133.74
## X130 18779 18431 1.34 0.97 98.47
## X131 18531 18115 1.54 1.66 371.30
## X132 18325 17990 1.20 1.16 211.61
## X133 18430 18094 0.99 0.16 158.83
## X134 18710 18347 0.74 -0.44 168.47
## X135 18508 18168 1.20 0.72 149.54

```

## 6.3 Eigene Funktionen schreiben

### 6.3.1 Standardfehler-Funktion

Eine eigene Funktion lässt sich wie folgt schreiben:

```
name <- function(argument){Berechnung}
```

Hier ein Beispiel für den Standardfehler des Stichprobenmittelwerts:

$$s_{\bar{x}} = \frac{s}{\sqrt{n}}$$

```
std.error <- function(x) {
  sd(x) / sqrt(length(x))
}
sample <- c(1, 1, 2, 2, 4, 6, 9)
std.error(sample) == sd(sample) / sqrt(length(sample))

## [1] TRUE
std.error(sample)

## [1] 1.13089
```

### 6.3.2 Mehr Beispiele

```
# celsius nach kelvin konvertieren
celsius.kelvin <- function(temp) {
  temp + 273.15
}
# fahrenheit nach kelvin konvertieren
fahrenheit.celsius <- function(temp) {
  (temp - 32) * 5 / 9
}
# celsius nach fahrenheit konvertieren
celsius.fahrenheit <- function(temp) {
  temp * 9 / 5 + 32
}
celsius.kelvin(36)

## [1] 309.15
fahrenheit.celsius(96.8)

## [1] 36
celsius.fahrenheit(36)

## [1] 96.8
```

### 6.3.3 Übungen

1. Schreibt eine Funktion, die aus einer gegebenen Zahl ihr Quadrat berechnet

2. Schreibt eine Funktion, die den `describeBy`-Befehl auf ihre Argumente anwendet und die Matrix als data frame ausgibt
3. Schreibt eine Funktion, die den Mittelwert ihres Arguments berechnet (ohne `mean!`)

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n a_i = \frac{a_1 + a_2 + \cdots + a_n}{n}$$

4. Schreibt eine Funktion, die das erste und das letzte Element eines Vektors ausgibt
5. Schreibt eine Funktion, die den Mittelwert, die Standardabweichung und den Median eines Vektors berechnet und als Vektor ausgibt.

#### 6.3.4 Lösungen

1. Schreibt eine funktion, die aus einer gegebenen Zahl ihr Quadrat berechnet

```
my.square <- function(n) {
  n * n
}
my.squaretwo <- function(n) {
  n^2
}
my.square(10)

## [1] 100
my.squaretwo(10)

## [1] 100
```

#### 6.3.5 Lösungen

2. Schreibt eine Funktion, die den `describeBy`-Befehl auf ihre Argumente anwendet und die Matrix als data frame ausgibt

```
describeByDF <- function(x, y) {
  d <- describeBy(x, y, mat = T)
  d <- as.data.frame(d)
  d
}
descrip <- describeByDF(diamonds$price,
  list(diamonds$cut, diamonds$color))
is.data.frame(descrip)

## [1] TRUE
head(descrip)

##      item   group1 group2 vars     n      mean       sd median trimmed      mad
## X11    1      Fair      D   1 163 4291.061 3286.114 3730.0 3703.962 2043.023
## X12    2      Good      D   1 662 3405.382 3175.149 2728.5 2859.296 2674.610
## X13    3 Very Good      D   1 1513 3470.467 3523.753 2310.0 2774.014 2373.643
## X14    4   Premium      D   1 1603 3631.293 3711.634 2009.0 2880.995 1893.280
## X15    5      Ideal      D   1 2834 2629.095 3001.070 1576.0 1929.827 1184.597
## X16    6      Fair      E   1 224 3682.312 2976.652 2956.0 3164.344 2311.373
##      min     max range      skew kurtosis       se
```

```
## X11 536 16386 15850 1.902602 3.810324 257.38833
## X12 361 18468 18107 2.033159 5.424658 123.40566
## X13 357 18542 18185 1.914574 3.923597 90.59120
## X14 367 18575 18208 1.782423 2.964122 92.70398
## X15 367 18693 18326 2.554160 6.920157 56.37365
## X16 337 15584 15247 1.876280 4.002687 198.88590
```

### 6.3.6 Lösungen

3. Schreibt eine Funktion, die den Mittelwert ihres Arguments berechnet (ohne `mean!`)

```
mein.mittelwert <- function(x) {
  sum(x) / length(x)
}
mein.mittelwert(sample) == mean(sample)

## [1] TRUE
mein.mittelwert(sample)

## [1] 3.571429
```

### 6.3.7 Lösungen

4. Schreibt eine Funktion, die das erste und das letzte Element eines Vektors ausgibt

```
sample

## [1] 1 1 2 2 4 6 9

my.ends <- function(vector) {
  answer <- c(vector[1], vector[length(vector)])
  answer
}
my.ends2 <- function(x) {
  c(head(x, 1), tail(x, 1))
}
a <- 1:15
my.ends2(sample)

## [1] 1 9
my.ends(sample)

## [1] 1 9
```

### 6.3.8 Lösungen

5. Schreibt eine Funktion, die den Mittelwert, die Standardabweichung und den Median eines Vektors berechnet und als Vektor ausgibt.

```
my.desc <- function(vector) {
  mean <- mean(vector)
  sd <- sd(vector)
  median <- median(vector)
  desc <- c(mean, sd, median)
  desc
}
my.desc(sample)
```

```

## [1] 3.571429 2.992053 2.000000
standard <- c(mean(sample), sd(sample), median(sample))
my.desc(sample) == standard

## [1] TRUE TRUE TRUE
my.desc2 <- function(vector) {
  a <- c(mean(vector), sd(vector), median(vector))
}
my.desc2(sample) == my.desc(sample)

## [1] TRUE TRUE TRUE

```

## 6.4 Tabellen transponieren

### 6.4.1 reshape2

```

install.packages("reshape2")
library(reshape2)

melt() # wide zu long
cast() # long zu wide

```

### 6.4.2 Von Wide zu Long

Einlesen der Datei

```

br <- read.table("docs/data/tut6/Briefe.csv", header = T, sep = ";", dec = ",")
head(br)

##           Filename Segment   WC    WPS Sixltr Pronoun    I    We
## 1 m-Rollet_Alexander-BR-1844_1906.txt      1 500 19.23   32.4    4.0 0.0 0.4
## 2 m-Rollet_Alexander-BR-1844_1906.txt      2 500 18.52   32.8    7.6 3.6 0.6
## 3 m-Rollet_Alexander-BR-1844_1906.txt      3 500 41.67   28.8    9.2 0.0 5.6
## 4 m-Rollet_Alexander-BR-1844_1906.txt      4 500 31.25   36.2    5.8 0.0 1.2
## 5 m-Rollet_Alexander-BR-1844_1906.txt      5 500 23.81   34.8    5.0 0.0 0.6
## 6 m-Rollet_Alexander-BR-1844_1906.txt      6 500 25.00   36.0    8.4 4.8 0.6
##             Self You
## 1       0.4 2.2
## 2       4.2 1.6
## 3       5.6 0.8
## 4       1.2 1.8
## 5       0.6 2.2
## 6       5.4 1.6

```

### 6.4.3 Von Wide zu Long

Transponieren der Tabelle ins Long-Format:

```
brlong <- melt(br, id.vars = c("Filename", "Segment"))
head(brlong, 15)
```

	Filename	Segment	variable	value
## 1	m-Rollet_Alexander-BR-1844_1906.txt	1	WC	500
## 2	m-Rollet_Alexander-BR-1844_1906.txt	2	WC	500
## 3	m-Rollet_Alexander-BR-1844_1906.txt	3	WC	500
## 4	m-Rollet_Alexander-BR-1844_1906.txt	4	WC	500
## 5	m-Rollet_Alexander-BR-1844_1906.txt	5	WC	500
## 6	m-Rollet_Alexander-BR-1844_1906.txt	6	WC	500
## 7	m-Rollet_Alexander-BR-1844_1906.txt	7	WC	500
## 8	m-Rollet_Alexander-BR-1844_1906.txt	8	WC	500
## 9	m-Rollet_Alexander-BR-1844_1906.txt	9	WC	500
## 10	m-Rollet_Alexander-BR-1844_1906.txt	10	WC	500
## 11	m-Rollet_Alexander-BR-1844_1906.txt	11	WC	500
## 12	m-Rollet_Alexander-BR-1844_1906.txt	12	WC	500
## 13	m-Rollet_Alexander-BR-1844_1906.txt	13	WC	500
## 14	m-Rollet_Alexander-BR-1844_1906.txt	14	WC	500
## 15	m-Rollet_Alexander-BR-1844_1906.txt	15	WC	500

#### 6.4.4 Von Long zu Wide

```
brwide <- dcast(brlong, Filename + Segment ~ variable,
  value.var = "value")
head(brwide)
```

	Filename	Segment	WC	WPS	Sixltr	Pronoun	I	We
## 1	f-Bachmann_Ingeborg-BR-1948_61.txt	1	500	13.16	14.6	16.2	6.8	0.6
## 2	f-Bachmann_Ingeborg-BR-1948_61.txt	2	500	17.86	14.4	18.8	8.4	0.6
## 3	f-Bachmann_Ingeborg-BR-1948_61.txt	3	500	15.15	14.0	20.2	8.2	2.2
## 4	f-Bachmann_Ingeborg-BR-1948_61.txt	4	500	18.52	15.0	18.4	8.6	0.8
## 5	f-Bachmann_Ingeborg-BR-1948_61.txt	5	500	19.23	12.8	15.2	8.8	0.8
## 6	f-Bachmann_Ingeborg-BR-1948_61.txt	6	500	16.13	16.8	17.2	9.8	0.8
	Self You							
## 1	7.4 5.8							
## 2	9.0 5.6							
## 3	10.4 6.4							
## 4	9.4 6.0							
## 5	9.6 3.4							
## 6	10.6 3.8							

#### 6.4.5 Übungen II

1. Lest die Datei “implicated\_lies.csv” ein
2. Entscheidet, ob die Datensortierung long oder wide ist und transponiert die Daten ins jeweils andere Format
3. Speichert die Daten auf eurem Computer ab
4. Benutzt `describeBy`, um statistische Berechnungen für jedes Item separat zu bekommen

#### 6.4.6 Lösungen II

1. Lest die Datei “implicated\_lies.csv” ein

```
dlies <- read.csv("docs/data/tut6/implicated_lies.csv", header = T)
head(dlies)

##          id age stage af1 af2 af3 af4 af5 af6 aw1 aw2 aw3 aw4
## 1 Okcc8m19m6e46i2ojjomvjre91 53 adult  1  1  1  1  1  1  5  5  5  5
## 2 4fg5ka3ppabgnfd1k65ei6ar5 25 adult  1  1  1  1  1  1  5  5  5  5
## 3 77njs4ij7pstnrbe7d7ea033j5 24 adult  5  1  1  1  1  1  5  5  5  5
## 4 fauj5blp14kgovr221aac53p64 23 adult  1  1  1  1  1  1  5  5  5  5
## 5 ubqoiq93qbg3fcarse9e54qmi0 26 adult  1  1  5  1  1  1  5  5  5  5
## 6 398ovonqku33iadqkoorvju9s0 20 adult  1  1  1  1  1  1  5  5  5  5
##   aw5 aw6 gf1s gf2s gf3s gf4n gf5n gf6n gw1s gw2s gw3s gw4n gw5n gw6n pf1 pf2
## 1   5   5    2   1   1   5    5   1   5   5   5   5   5   5   5   5   5   5
## 2   5   5    3   5   1   5    1   2   5   5   5   5   5   5   5   5   5   1
## 3   5   5    1   4   4   4    4   1   5   5   5   5   5   5   5   5   4   2
## 4   5   5    1   1   1   1    1   4   2   5   5   4   5   5   5   5   1   1
## 5   5   5    1   1   1   1    4   2   1   2   5   5   5   5   5   5   2   4
## 6   4   5    2   2   2   3    4   3   5   NA  5   5   5   5   5   5   3   2
##   pf3 pf4 pf5 pf6 pw1 pw2 pw3 pw4 pw5 pw6
## 1   5   5    2   5   5   2    5   5   2   2
## 2   5   1    3   5   5   5    5   5   4   5
## 3   4   3    2   3   5   5    5   5   4   5
## 4   1   1    1   4   5   5    5   5   4   4
## 5   5   1    2   4   5   5    5   3   5   5
## 6   2   3    3   3   4   5    4   3   4   4
```

#### 6.4.7 Lösungen II

2. Entscheidet, ob die Datensortierung long oder wide ist und transponiert die Daten ins jeweils andere Format

```
dlieslong <- melt(dlies, id.vars = c("id", "stage", "age"),
  value.name = "judgment", variable.name = "item")
# sorting: dlieslong sorted first by the id column and then,
# provided id stays constant, by item
dlieslong <- dlieslong[order(dlieslong$id, dlieslong$item), ]
head(dlieslong, 8)
```

```
##          id stage age item judgment
## 21 02dfkhe0rsj9q8abs2skfc19h0 adult 31 af1     1
## 123 02dfkhe0rsj9q8abs2skfc19h0 adult 31 af2     1
## 225 02dfkhe0rsj9q8abs2skfc19h0 adult 31 af3     1
## 327 02dfkhe0rsj9q8abs2skfc19h0 adult 31 af4     1
## 429 02dfkhe0rsj9q8abs2skfc19h0 adult 31 af5     1
## 531 02dfkhe0rsj9q8abs2skfc19h0 adult 31 af6     1
## 633 02dfkhe0rsj9q8abs2skfc19h0 adult 31 aw1     5
## 735 02dfkhe0rsj9q8abs2skfc19h0 adult 31 aw2     5
```

#### 6.4.8 Lösungen

3. Speichert die Daten auf eurem Computer ab

```
write.csv(dlieslong, file = "implicated_lies_long.csv",
  row.names = F, quote = F, na = "keine Angabe")
```

### 6.4.9 Lösungen

4. Benutzt `describeBy`, um statistische Berechnungen für jedes Item separat zu bekommen

```
describeBy(dlieslong$judgment, dlieslong$item, mat = T, digits = 1)
```

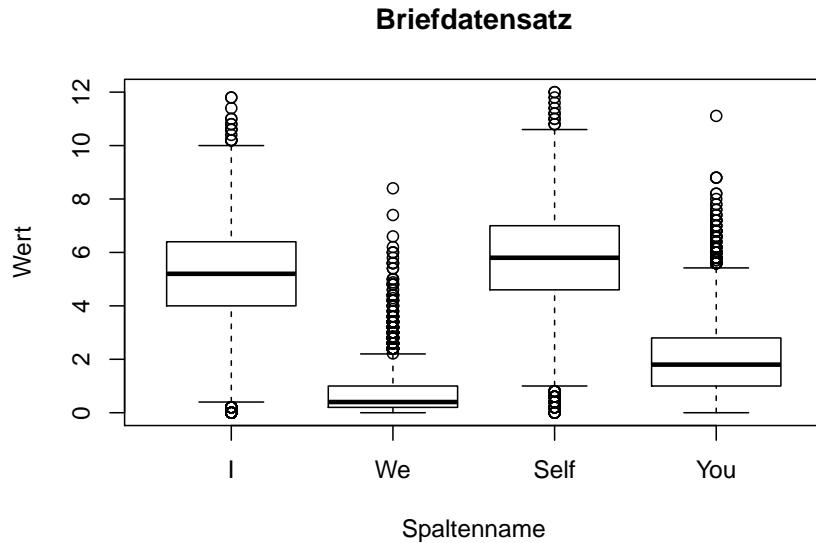
	item	group1	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew
##	X11	1	af1	1 101	2.0	1.5	1	1.7	0.0	1	5	4	1.3
##	X12	2	af2	1 100	1.2	0.8	1	1.0	0.0	1	5	4	4.4
##	X13	3	af3	1 100	1.1	0.6	1	1.0	0.0	1	5	4	6.2
##	X14	4	af4	1 101	1.1	0.4	1	1.0	0.0	1	4	3	5.3
##	X15	5	af5	1 101	1.1	0.6	1	1.0	0.0	1	5	4	5.8
##	X16	6	af6	1 101	1.1	0.3	1	1.0	0.0	1	2	1	3.3
##	X17	7	aw1	1 101	4.8	0.8	5	5.0	0.0	1	5	4	-3.7
##	X18	8	aw2	1 100	4.7	0.9	5	5.0	0.0	1	5	4	-3.3
##	X19	9	aw3	1 101	4.9	0.6	5	5.0	0.0	1	5	4	-5.6
##	X110	10	aw4	1 99	4.8	0.8	5	5.0	0.0	1	5	4	-3.6
##	X111	11	aw5	1 99	4.8	0.6	5	5.0	0.0	1	5	4	-5.0
##	X112	12	aw6	1 101	4.8	0.8	5	5.0	0.0	1	5	4	-3.9
##	X113	13	gf1s	1 33	1.6	1.0	1	1.4	0.0	1	5	4	1.7
##	X114	14	gf2s	1 33	1.8	1.2	1	1.6	0.0	1	5	4	1.4
##	X115	15	gf3s	1 33	2.1	1.3	2	1.9	1.5	1	5	4	0.9
##	X116	16	gf4n	1 33	3.6	1.5	4	3.7	1.5	1	5	4	-0.6
##	X117	17	gf5n	1 33	2.7	1.3	3	2.6	1.5	1	5	4	0.2
##	X118	18	gf6n	1 33	1.9	1.4	1	1.7	0.0	1	5	4	1.2
##	X119	19	gw1s	1 33	4.6	0.8	5	4.7	0.0	2	5	3	-1.7
##	X120	20	gw2s	1 32	5.0	0.2	5	5.0	0.0	4	5	1	-5.1
##	X121	21	gw3s	1 33	4.8	0.4	5	4.8	0.0	4	5	1	-1.1
##	X122	22	gw4n	1 33	5.0	0.0	5	5.0	0.0	5	5	0	NaN
##	X123	23	gw5n	1 33	4.8	0.9	5	5.0	0.0	1	5	4	-3.6
##	X124	24	gw6n	1 33	4.7	0.8	5	4.9	0.0	1	5	4	-3.7
##	X125	25	pf1	1 101	2.9	1.5	3	2.9	3.0	1	5	4	0.1
##	X126	26	pf2	1 101	3.2	1.5	3	3.3	3.0	1	5	4	-0.1
##	X127	27	pf3	1 101	2.1	1.3	2	1.9	1.5	1	5	4	1.1
##	X128	28	pf4	1 102	2.6	1.6	2	2.5	1.5	1	5	4	0.5
##	X129	29	pf5	1 99	2.3	1.3	2	2.2	1.5	1	5	4	0.6
##	X130	30	pf6	1 102	3.4	1.4	3	3.5	1.5	1	5	4	-0.3
##	X131	31	pw1	1 100	4.6	0.9	5	4.8	0.0	1	5	4	-2.4
##	X132	32	pw2	1 100	4.9	0.5	5	5.0	0.0	2	5	3	-4.4
##	X133	33	pw3	1 102	4.7	0.8	5	4.9	0.0	1	5	4	-3.5
##	X134	34	pw4	1 99	4.8	0.7	5	4.9	0.0	2	5	3	-3.0
##	X135	35	pw5	1 100	4.5	0.9	5	4.8	0.0	1	5	4	-2.2
##	X136	36	pw6	1 101	4.6	0.8	5	4.8	0.0	1	5	4	-2.7
##			kurtosis	se									
##	X11				0.0	0.2							
##	X12				18.4	0.1							
##	X13				38.8	0.1							
##	X14				30.2	0.0							
##	X15				33.5	0.1							
##	X16				9.3	0.0							
##	X17				13.3	0.1							
##	X18				9.6	0.1							
##	X19				32.9	0.1							
##	X110				12.2	0.1							
##	X111				26.3	0.1							
##	X112				13.9	0.1							
##	X113				2.5	0.2							
##	X114				0.8	0.2							
##	X115				-0.6	0.2							
##	X116				-1.2	0.3							
##	X117				-1.3	0.2							

```
## X118      0.0 0.2
## X119      1.9 0.1
## X120     25.2 0.0
## X121     -0.7 0.1
## X122      NaN 0.0
## X123     11.9 0.1
## X124     14.6 0.1
## X125     -1.4 0.2
## X126     -1.5 0.2
## X127     -0.1 0.1
## X128     -1.3 0.2
## X129     -0.8 0.1
## X130     -1.3 0.1
## X131      5.3 0.1
## X132     20.1 0.1
## X133     12.2 0.1
## X134      8.6 0.1
## X135      4.4 0.1
## X136      7.4 0.1
```

## 6.5 Nur Kurz: Plots

### 6.5.1 Ein Plot zum Abschluss

```
# boxplot der letzten vier spalten in briefe.csv
p1 <- boxplot(br[, 7:length(br)], main = "Briefdatensatz",
  xlab = "Spaltenname", ylab = "Wert")
```

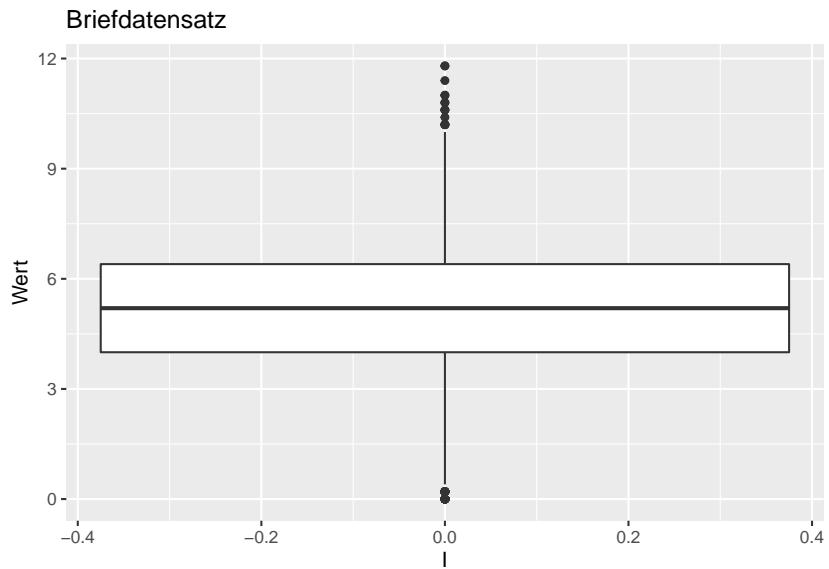


### 6.5.2 Ein Plot zum Abschluss

```
library(ggplot2)
p2 <- ggplot(br, aes(y = I)) +
  geom_boxplot() +
  labs(title = "Briefdatensatz",
       x = "I", y = "Wert")
```

### 6.5.3 Ein Plot zum Abschluss

p2



### 6.5.4 F\*ck die Uni

Schönes Wochenende!

## Sitzung 7

### 7.1 Wiederholung

#### 7.1.1 Das Wichtigste vom letzten Mal

Pakete installieren und laden

```
install.packages("PAKETNAME")
library(PAKETNAME)
```

Deskriptive Statistik

```
library(psych)
describeBy(d$judgment, list(d$wordorder, d$quantifier),
  digits = 2, mat = T)
```

#### 7.1.2 Das Wichtigste vom letzten Mal

Funktionen schreiben

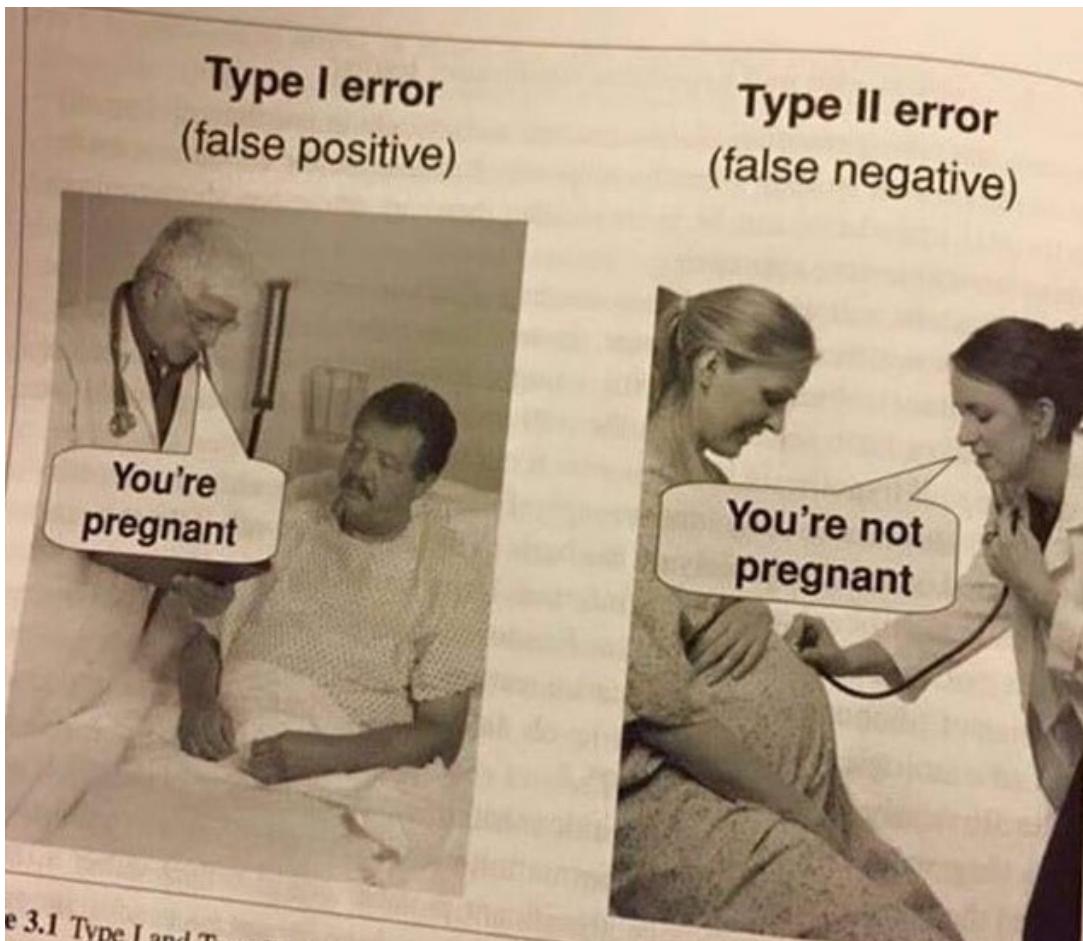
```
my.ends <- function(vector) {
  answer <- c(vector[1], vector[length(vector)])
  answer
}
```

Daten transponieren

```
library(reshape2)
cast(id + age + gender ~ condition,
  value.var = "judgment") # long zu wide
melt(d, id.vars = c("id", "age", "gender")) # wide zu long
```

## 7.2 Fehlertypen

### 7.2.1 Fehlertypen: Erinnerungshilfe



## 7.3 $\chi^2$ -Test

### 7.3.1 Daten

Übersicht zahlreicher Metriken von Studierenden an irgendeiner Hochschule.

Von der [Website](#):

The students data set consists of 8239 rows, each of them representing a particular student, and 16 columns, each of them corresponding to a variable/feature related to that particular student. These self-explaining variables are: stud.id, name, gender, age, height, weight, religion, nc.score, semester, major, minor, score1, score2, online.tutorial, graduated, salary.

```
d <- read.csv("docs/data/tut7/studentsmajor.csv")
head(d, 3)

##   stud.id      name gender age height weight religion nc.score
## 1 833917 Gonzales, Christina Female 19    160   64.8 Muslim     1.91
## 2 898539 Lozano, T'Hani Female 19    172   73.0 Other      1.56
## 3 379678 Williams, Hanh Female 22    168   70.6 Protestant  1.24
##   semester      major      minor score1 score2
## 1       1st Political Science Social Sciences NA NA
## 2       2nd Social Sciences Mathematics and Statistics NA NA
## 3       3rd Social Sciences Mathematics and Statistics 45 46
##   online.tutorial graduated salary
## 1             0          0     NA
## 2             0          0     NA
## 3             0          0     NA
```

### 7.3.2 Daten

Anforderungen des  $\chi^2$ -Tests:

- Kategoriale Daten
- Unabhängige Beobachtungen

```
str(d)
```

```
## 'data.frame': 8239 obs. of 16 variables:
## $ stud.id : int 833917 898539 379678 807564 383291 ...
## $ name   : Factor w/ 8174 levels "Aarvold, Cindi",...
## $ gender : Factor w/ 2 levels "Female","Male": 1 1 1 2 1 2 1 1 2 1 ...
## $ age    : int 19 19 22 19 21 19 21 21 18 18 ...
## $ height : int 160 172 168 183 175 189 156 167 195 165 ...
## $ weight : num 64.8 73 70.6 79.7 71.4 85.8 65.9 65.7 94.4 66 ...
## $ religion : Factor w/ 5 levels "Catholic","Muslim",...
## $ nc.score : num 1.91 1.56 1.24 1.37 1.46 1.34 1.11 2.03 1.29 1.19 ...
## $ semester : Factor w/ 7 levels ">6th","1st","2nd",...
## $ major   : Factor w/ 6 levels "Biology","Economics and Finance",...
## $ minor   : Factor w/ 6 levels "Biology","Economics and Finance",...
## $ score1  : int NA NA 45 NA NA NA 58 57 NA ...
## $ score2  : int NA NA 46 NA NA NA 62 67 NA ...
## $ online.tutorial: int 0 0 0 0 0 0 0 0 ...
## $ graduated : int 0 0 0 0 0 0 0 0 0 ...
## $ salary   : num NA NA NA NA NA NA NA NA NA ...
```

### 7.3.3 Häufigkeitstabelle

Kategoriale Variablen aufeinander abbilden

```
tab <- table(d$major, d$gender)
tab
```

			Female	Male
##				
##	Biology		959	638
##	Economics and Finance		461	863
##	Environmental Sciences		745	881
##	Mathematics and Statistics		276	949
##	Political Science		978	477
##	Social Sciences		691	321

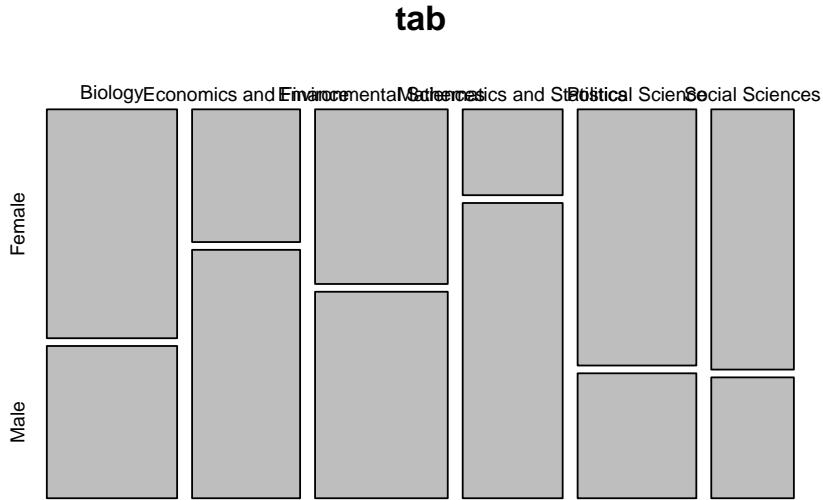
### 7.3.4 Relative Häufigkeiten

```
prop.table(tab, margin = 1) * 100

##
##                               Female      Male
## Biology                  60.05009 39.94991
## Economics and Finance    34.81873 65.18127
## Environmental Sciences   45.81796 54.18204
## Mathematics and Statistics 22.53061 77.46939
## Political Science        67.21649 32.78351
## Social Sciences           68.28063 31.71937
```

### 7.3.5 Grafische Übersicht

```
plot(tab)
```

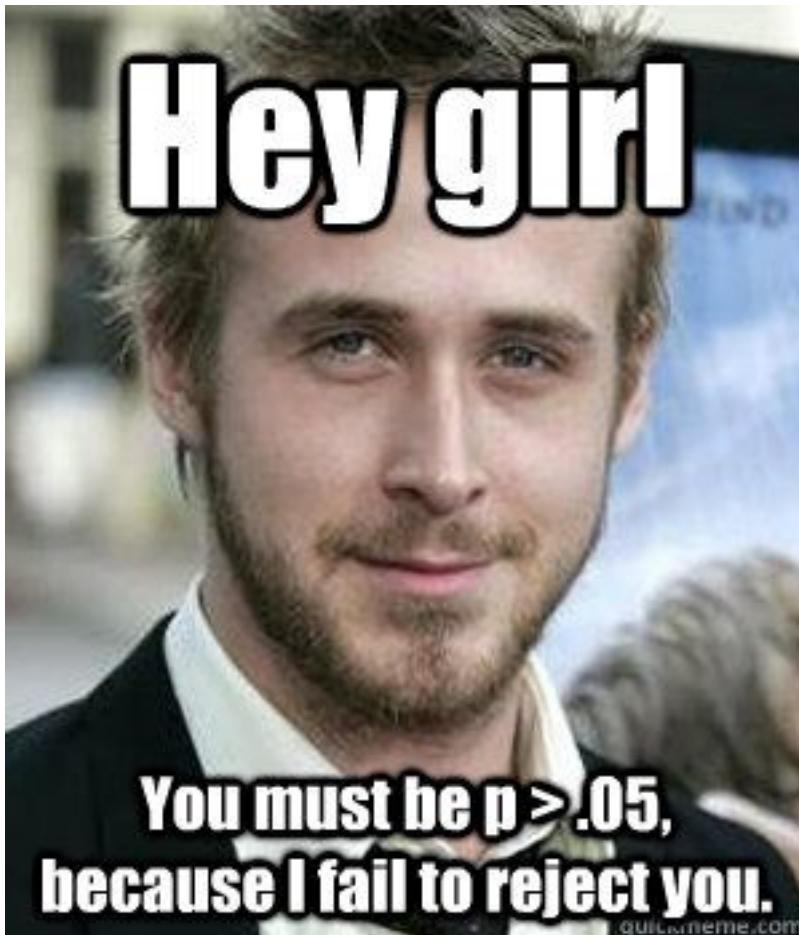


### 7.3.6 $\chi^2$ berechnen

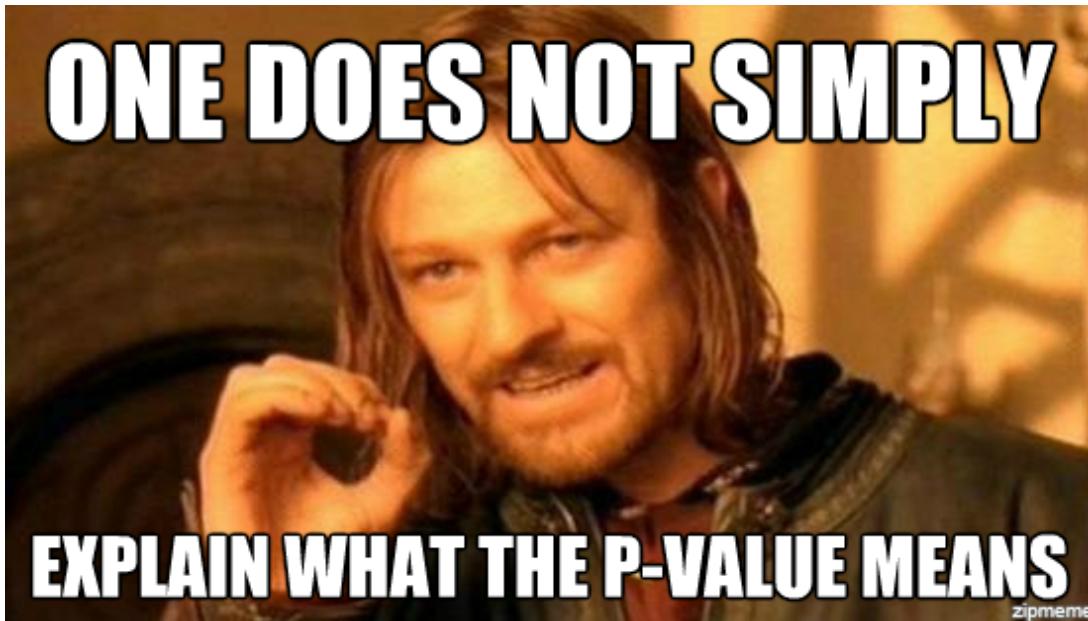
```
options(scipen = 999)
chisq.test(d$gender, d$major)

##
## Pearson's Chi-squared test
##
## data: d$gender and d$major
## X-squared = 875.44, df = 5, p-value < 0.0000000000000022
```

### 7.3.7 P-Werte



### 7.3.8 P-Werte



### 7.3.9 Übungen

1. Erstellt eine Tabelle, die das Studienfach nach Religionszugehörigkeit auszählt (einmal als absolute Häufigkeiten, einmal in Prozent)
2. Plottet die Tabelle
3. Berechnet den dazugehörigen  $\chi^2$ -Test

### 7.3.10 Lösungen

1. Erstellt eine Tabelle, die das Studienfach nach Religionszugehörigkeit auszählt (einmal als absolute Häufigkeiten, einmal in Prozent)

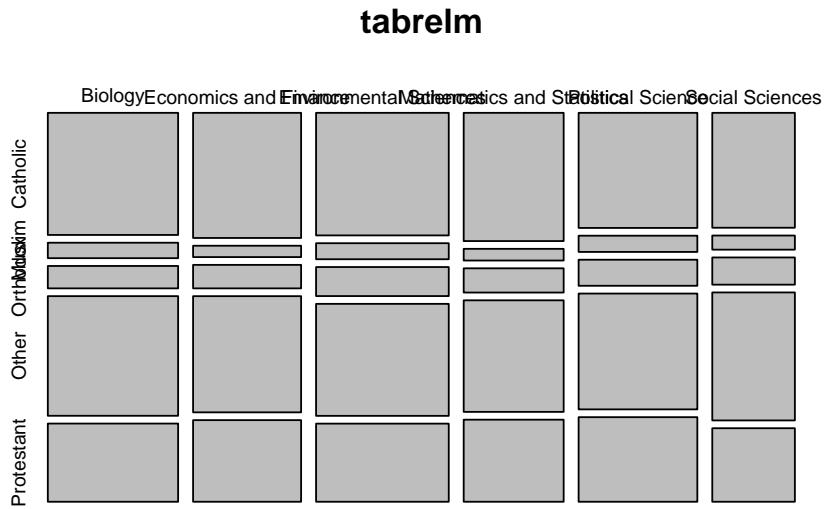
```
tabrelm <- table(d$major, d$religion)
prop.table(tabrelm, margin = 1) * 100
```

```
##
##                                     Catholic   Muslim Orthodox   Other Protestant
##   Biology                      34.126487  4.320601  6.261741 33.437696  21.853475
##   Economics and Finance        34.969789  3.172205  6.570997 32.477341  22.809668
##   Environmental Sciences      34.255843  4.489545  8.118081 31.303813  21.832718
##   Mathematics and Statistics  35.836735  3.265306  6.775510 31.183673  22.938776
##   Political Science           32.164948  4.536082  7.285223 32.371134  23.642612
##   Social Sciences              32.114625  3.952569  7.608696 35.770751  20.553360
```

### 7.3.11 Lösungen

2. Plottet die Tabelle

```
plot(tabrelm)
```



### 7.3.12 Lösungen

3. Berechnet den dazugehörigen  $\chi^2$ -Test

```
chisq.test(d$religion, d$major)
```

```
## 
## Pearson's Chi-squared test
## 
## data: d$religion and d$major
## X-squared = 23.881, df = 20, p-value = 0.2476
```

### 7.3.13 Übungen II

1. Erstellt ein Subset mit den drei größten Religionen
2. Erstellt eine Tabelle, die die Religionszugehörigkeit nach Geschlecht auszählt (einmal als absolute Häufigkeiten, einmal in Prozent)
3. Plottet die Tabelle
4. Berechnet den dazugehörigen  $\chi^2$ -Test

### 7.3.14 Lösungen II

1. Erstellt ein Subset mit den drei größten Religionen

```
dsub <- subset(d, religion %in% c("Catholic", "Other", "Protestant"))
# alternative_1: dsub <- d[d$religion %in% c("Catholic", "Other", "Protestant"), ]
# alternative_2: dsub <- d[d$religion == "Catholic" | d$religion == "Other" | d$religion == "Protestant", ]
# alternative: dsub$religion <- factor(dsub$religion )
dsub <- droplevels(dsub)
```

### 7.3.15 Zur Erinnerung: droplevels

```
a <- c("a", "b", "c")
a <- factor(a)
a

## [1] a b c
## Levels: a b c

a <- a[1:2]
a

## [1] a b
## Levels: a b c

a <- droplevels(a)
a

## [1] a b
## Levels: a b
```

### 7.3.16 Lösungen II

2. Erstellt eine Tabelle, die die Religionszugehörigkeit nach Geschlecht auszählt (einmal als absolute Häufigkeiten, einmal in Prozent)

```
tabrelg <- table(dsub$religion, dsub$gender)
prop.table(tabrelg, margin = 1) * 100

##
##          Female     Male
## Catholic 48.48051 51.51949
## Other    50.55804 49.44196
## Protestant 49.48341 50.51659
```

### 7.3.17 Lösungen II

3. Plottet die Tabelle

```
plot(tabrelg)
```

## tabrelg

	Catholic	Other	Protestant
Female			
Male			

### 7.3.18 Lösungen II

4. Berechnet den dazugehörigen  $\chi^2$ -Test

```
chisq.test(dsub$religion, dsub$gender)
```

```
## 
## Pearson's Chi-squared test
## 
## data: dsub$religion and dsub$gender
## X-squared = 2.3668, df = 2, p-value = 0.3062
```

## 7.4 t-test

### 7.4.1 Zwei Arten

```
t.test(WerteGruppe1, WerteGruppe2, paired = TRUE / FALSE,
       var.equal = TRUE) # var.equal = F = Welch's Test
```

Alternative:

```
t.test(AV ~ Gruppenvariable, data = d, paired = TRUE / FALSE,
       var.equal = TRUE)
```

paired = TRUE

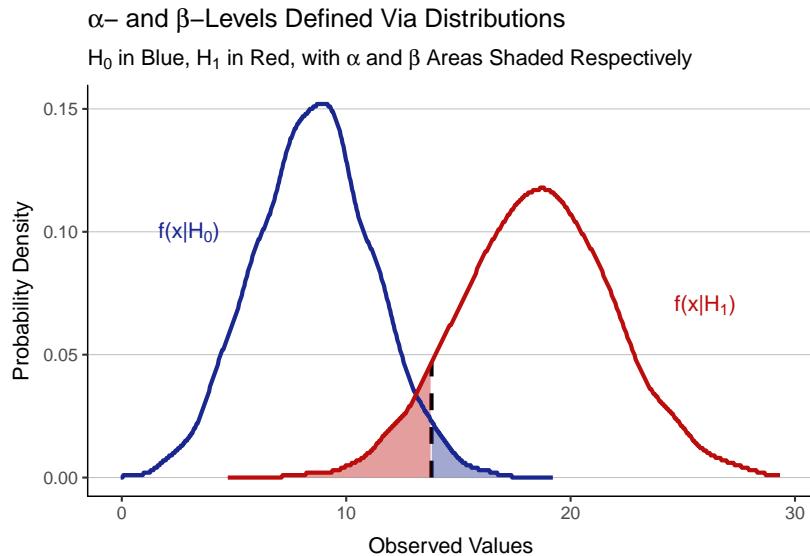
- Abhängige Beobachtungen/Stichproben: Oft Messwiederholung: within-subjects

paired = FALSE

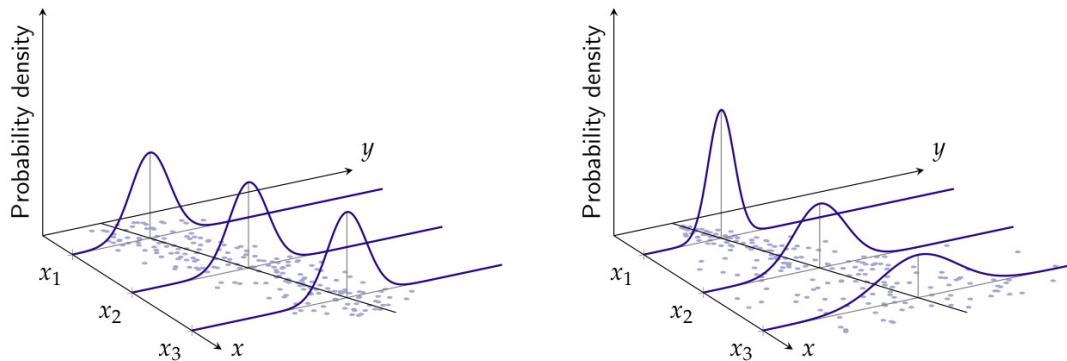
- Unabhängige Beobachtungen/Stichproben: between-subjects

- Voraussetzungen für die Anwendung des independent t-tests:
  - Unabhängige Variable sollte aus zwei kategorialverteilten Gruppen bestehen
  - Die Beobachtungen sollten unabhängig sein
  - Die abhängige Variable sollte innerhalb der beiden Gruppen ungefähr normalverteilt sein
  - Varianzhomogenität

#### 7.4.2 Interlude: Warum Normalverteilung?



#### 7.4.3 Interlude: Warum Varianzhomogenität/Homoskedastizität?



#### 7.4.4 Übung III

1. Lese die Daten, die in "student-mat.csv" enthalten, sind in R ein. Die relevanten Spalten, die wir uns im Folgenden genauer angucken wollen, sind "sex" und "G3". Erstelle ein entsprechendes Subset der Daten und verschaffe dir einen Überblick.
2. Überprüfe/Begründe, ob die folgenden Voraussetzungen gelten:
  - Unabhängige Variable sollte aus zwei kategorialverteilten Gruppen bestehen
  - Die Beobachtungen sollten unabhängig sein
  - Die abhängige Variable sollte innerhalb der beiden Gruppen ungefähr normalverteilt sein
  - Varianzhomogenität

#### 7.4.5 Lösungen III

1. Lese die Daten, die in "student-mat.csv" enthalten, sind in R ein, erstelle ein Subset und verschaffe dir einen Überblick.

```
d <- read.csv("docs/data/tut7/student-mat.csv")
dsub <- subset(d, select = c("sex", "G3"))
head(dsub)

##   sex G3
## 1 F  6
## 2 F  6
## 3 F 10
## 4 F 15
## 5 F 10
## 6 M 15

summary(dsub)

##   sex          G3
## F:208  Min.   : 0.00
## M:187  1st Qu.: 8.00
##           Median :11.00
##           Mean   :10.42
##           3rd Qu.:14.00
##           Max.   :20.00
```

#### 7.4.6 Lösungen III

2. Überprüfe/Begründe, ob die folgenden Voraussetzungen gelten:
  - Unabhängige Variable sollte aus zwei kategorialverteilten Gruppen bestehen
  - Die Beobachtungen sollten unabhängig sein

```
length(levels(dsub$sex)) == 2

## [1] TRUE
```

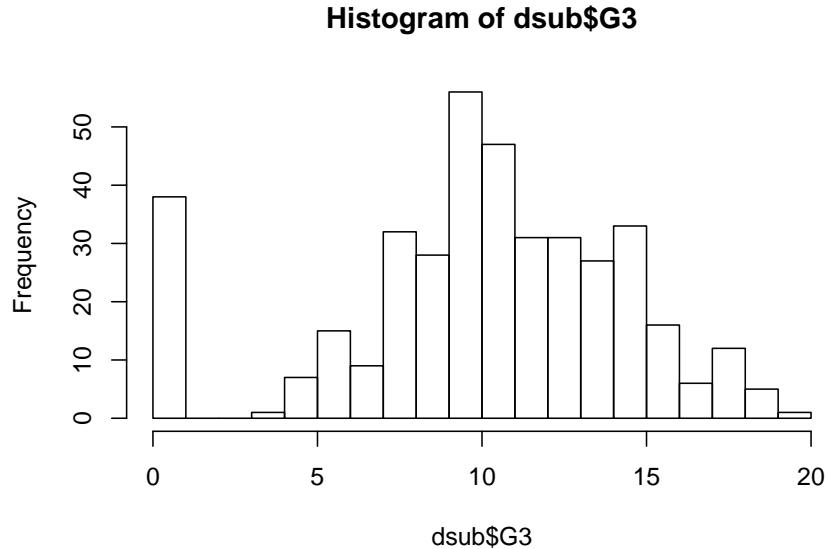
- Die Beobachtungen sollten unabhängig sein

Sex, so nehmen wir zumindest für die Zwecke dieser Übung an, ist binär- und disjunktverteilt. Die Anforderung ist also erfüllt, weil kein Schüler sowohl der Männer- als auch der Frauengruppe angehört.

#### 7.4.7 Lösungen III

- Die abhängige Variable sollte innerhalb der beiden Gruppen ungefähr normalverteilt sein
- Histogramm

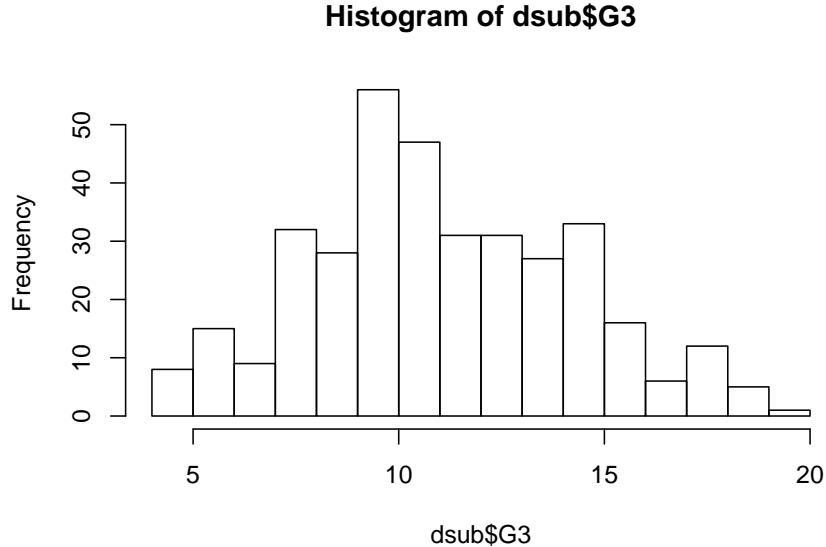
```
hist(dsub$G3, breaks = 20)
```



#### 7.4.8 Lösungen III

zu viele 0-Ergebnisse. Die sollten wir rausnehmen

```
dsub <- dsub[dsub$G3 != 0, ]  
hist(dsub$G3, breaks = 20)
```



#### 7.4.9 Lösungen III

- Varianzhomogenität

```
install.packages("car")

library(car)
leveneTest(G3 ~ sex, data = dsub)

## Levene's Test for Homogeneity of Variance (center = median)
##          Df F value Pr(>F)
## group     1  0.6145 0.4336
##            355
```

#### 7.4.10 Übung IV

Berechnet den *t*-test

#### 7.4.11 Lösung IV

Berechnet den *t*-test

```
t.test(dsub$G3[dsub$sex == "F"], dsub$G3[dsub$sex == "M"],
       paired = F, var.equal = TRUE)
```

Alternative Syntax:

```
t.test(G3 ~ sex, data = dsub, paired = F, var.equal = TRUE)
```

```
## 
## Two Sample t-test
```

```
##
## data: G3 by sex
## t = -1.9405, df = 355, p-value = 0.05311
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## -1.330667531 0.008920202
## sample estimates:
## mean in group F mean in group M
## 11.20541 11.86628
```

#### 7.4.12 Übung V

1. Lest die Datei "sleep.txt" ein und verschafft euch einen Überblick. Es geht hierbei um Folgendes: "Data which show the effect of two soporific drugs (increase in hours of sleep compared to control) on 10 patients."
2. Berechnet den entsprechenden *t*-Test

#### 7.4.13 Lösungen V

1. Lest die Datei "sleep.txt" ein und verschafft euch einen Überblick.

```
d <- read.table("docs/data/tut7/sleep.txt", sep = "|", header = T)
head(d)

##   extra group ID
## 1  0.7    1  1
## 2 -1.6    1  2
## 3 -0.2    1  3
## 4 -1.2    1  4
## 5 -0.1    1  5
## 6  3.4    1  6

str(d)

## 'data.frame': 20 obs. of 3 variables:
## $ extra: num  0.7 -1.6 -0.2 -1.2 -0.1 ...
## $ group: int  1 1 1 1 1 1 1 1 1 ...
## $ ID   : int  1 2 3 4 5 6 7 8 9 10 ...
```

#### 7.4.14 Lösungen V

2. Berechnet den entsprechenden *t*-Test

```
t.test(d$extra[d$group == "1"], d$extra[d$group == "2"],
paired = T, var.equal = TRUE)

t.test(extra ~ group, data = d, paired = T, var.equal = TRUE)

##
## Paired t-test
##
## data: extra by group
## t = -4.0621, df = 9, p-value = 0.002833
## alternative hypothesis: true difference in means is not equal to 0
```

```
## 95 percent confidence interval:
## -2.4598858 -0.7001142
## sample estimates:
## mean of the differences
## -1.58
```

#### 7.4.15 Lösungen V

- Berechnet den entsprechenden *t*-Test

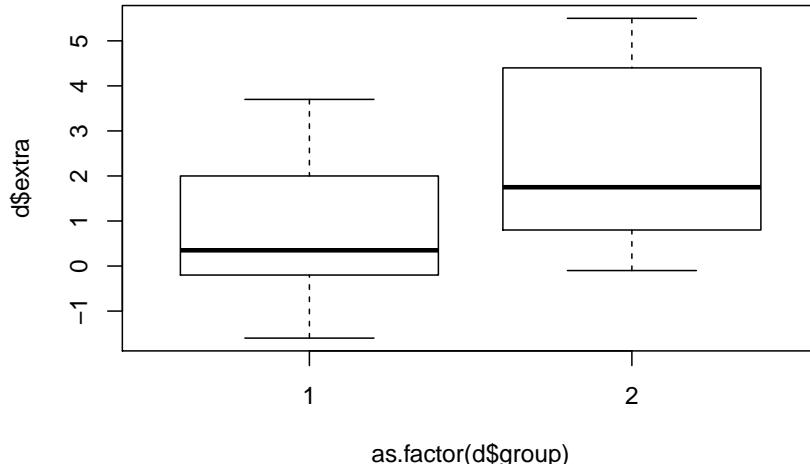
Hier habe ich die Alternativhypothese explizit angegeben. Erinnert ihr euch noch daran, was Thomas euch dazu erzählt hat?

```
t.test(d$extra[d$group == "1"], d$extra[d$group == "2"],
paired = T, alternative = "less", var.equal = TRUE)

##
##  Paired t-test
##
## data: d$extra[d$group == "1"] and d$extra[d$group == "2"]
## t = -4.0621, df = 9, p-value = 0.001416
## alternative hypothesis: true difference in means is less than 0
## 95 percent confidence interval:
## -Inf -0.8669947
## sample estimates:
## mean of the differences
## -1.58
```

#### 7.4.16 alternative = less

```
boxplot(d$extra ~ as.factor(d$group))
```



#### 7.4.17 Übungen VI

- Lest die Datei "grammar.dat" ein und verschafft euch einen Überblick. Es handelt sich hierbei um (ausdachte) Daten eines (ebenfalls ausgedachten) Experiments, in dem 10

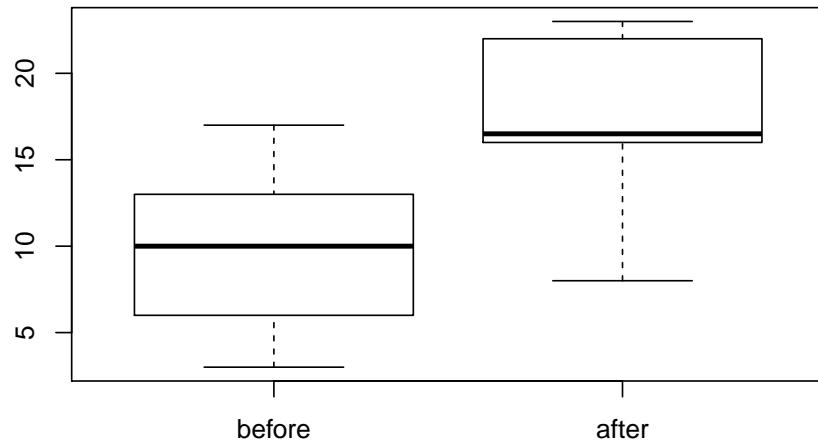
Studierende zweimal an einem Grammatiktest teilgenommen haben. Einmal ohne Training und einmal, zu einem späteren Zeitpunkt, mit vorangestelltem Grammatikunterricht.

2. Wie müsste der *t*-Test aussehen, der die Ergebnisse dieses Experiments inferenzstatistisch überprüfen soll?

#### 7.4.18 Lösungen VI

1. Lest die Datei "grammar.dat" ein und verschafft euch einen Überblick.

```
d <- read.table("docs/data/tut7/grammar.dat", sep = ";", header = T)
boxplot(d[, 1:2])
```



#### 7.4.19 Lösungen VI

2. Wie müsste der *t*-Test aussehen, der die Ergebnisse dieses Experiments inferenzstatistisch überprüfen soll?

```
t.test(d$before, d$after, alternative = "less",
       paired = T, var.equal = TRUE)
```

```
## 
##  Paired t-test
## 
##  data: d$before and d$after
##  t = -4.4853, df = 9, p-value = 0.0007604
##  alternative hypothesis: true difference in means is less than 0
##  95 percent confidence interval:
##      -Inf -4.493909
##  sample estimates:
##  mean of the differences
##                          -7.6
```

## 7.5 Nur Kurz: Plots

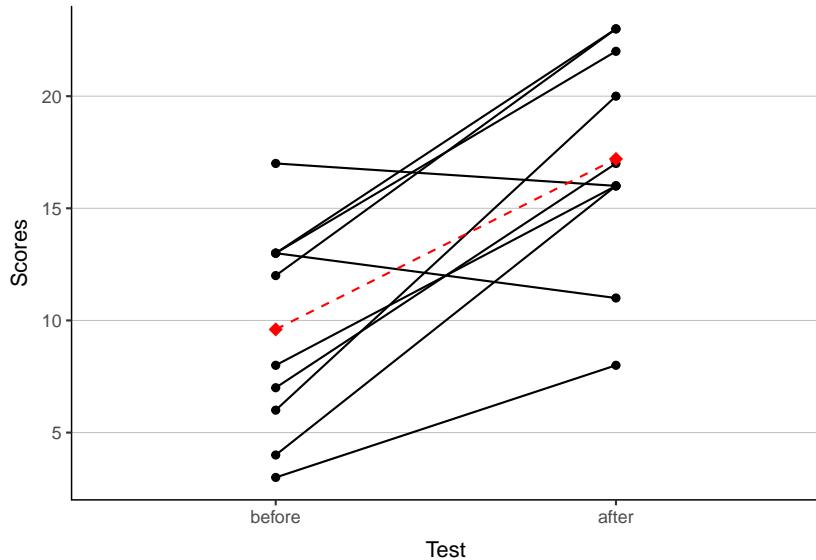
### 7.5.1 Ein Plot zum Abschluss

```
library(ggplot2)
library(reshape2)

dlong <- melt(d, id.vars = "id")
dlong$id <- factor(dlong$id)
p <- ggplot(dlong, aes(y = value, x = variable, group = id)) +
  geom_line() + geom_point() +
  stat_summary(fun.y = mean, geom = "line", color = "red",
    linetype = "dashed",
    mapping = aes(x = variable, y = value, group = 1)) +
  stat_summary(fun.y = mean, geom = "point", color = "red",
    shape = 18, size = 3,
    mapping = aes(x = variable, y = value, group = 1)) +
  labs(y = "Scores", x = "Test") + cleanup
```

### 7.5.2 Ein Plot zum Abschluss

```
# mittelwert in rot und gestrichelt
p
```



### 7.5.3 F\*ck die Uni

Schönes Wochenende!

## Sitzung 8

### 8.1 Wiederholung

#### 8.1.1 Das Wichtigste vom letzten Mal

$\chi^2$ -Test

```
chisq.test(x, y)
```

t-Test

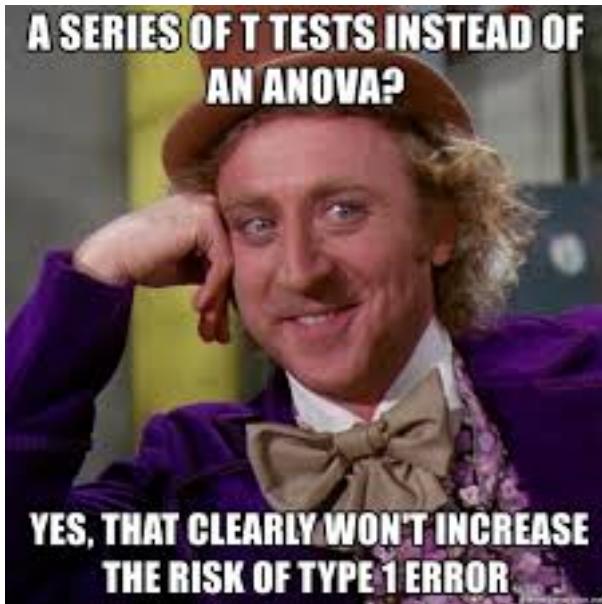
```
t.test(x, y, paired = FALSE, var.equal = T, data = d)
t.test(x, y, paired = TRUE, var.equal = T, data = d)
t.test(x, y, paired = TRUE, var.equal = T,
       alternative = "less", data = d) # "greater", "two.sided"
```

Alternative Syntax (lineares Modell):

```
t.test(AV ~ Gruppe, paired = FALSE / TRUE, var.equal = T, data = d)
```

## 8.2 Base R: ANOVA

### 8.2.1 Meme My Stats



### 8.2.2 Einleitendes

Die Varianzanalyse in R kann auf viele verschiedene Arten durchgeführt werden. Einige davon sind in base R enthalten, andere (idR komfortablere) werden durch externe Pakete zur Verfügung gestellt.

Ich zeige euch heute drei verschiedene davon:

```
aov()  
aov_ez()  
ezAnova()
```

### 8.2.3 Interlude: between und within

Beantwortet die unteren Fragen und gebt jeweils Beispiele (egal ob linguistischer Art oder nicht)

1. Was bedeutet es, wenn ein Faktor within-subjects ist?
2. Was ist mit between-subjects?
3. Wie steht's mit within-items?
4. Und mit between-items?

### 8.2.4 Interlude: between und within

Beantwortet die unteren Fragen und gebt jeweils Beispiele (egal ob linguistischer Art oder nicht)

1. Was bedeutet es, wenn ein Faktor within-subjects ist?
- Messung von Gedächtnisleistung: Jede Person einmal ausgeruht und einmal (an einem anderen Termin) nach einer durchzechten Nacht (außerdem Messwiederholung)
2. Was ist mit between-subjects?
- Geschlecht; Placebo vs richtiges Medikament
3. Wie steht's mit within-items?
- Item immer in zwei Ausführungen: Einmal mit Scrambling, einmal ohne
4. Und mit between-items?
- Itemset ist geteilt in Äußerungen mit Implikatur und Äußerungen ohne

### 8.2.5 aov()

Between-Design

```
summary(aov(AV ~ UVbetween, d))
```

Within-Design

```
summary(aov(AV ~ UVwithin + Error(ID - Variable / UVwithin), d))
```

Faktorinteraktion mit `*`

```
summary(aov(AV ~ UV1 * UV2, d))
```

Messwiederholung (within-Design)

```
summary(aov(AV ~ UV + Error(ID - Variable / UVwithin), d))
```

Messwiederholung (between-Design)

```
summary(aov(AV ~ UV + Error(ID - Variable), d))
```

### 8.2.6 aov()

Mixed-Design mit Messwiederholung

```
summary(aov(AV ~ UVwithin * UVbetween + Error(ID - Variable / UVwithin),
d))
```

Mixed-Design mit Messwiederholung und Interaktion

```
summary(aov(AV ~ UVwithin1 * UVwithin2 * UVbetween +
Error(ID - Variable / UVwithin1 * UVwithin2), d))
```

### 8.2.7 Übungen II

1. Stellt euch vor, ihr werdet beauftragt folgende Volksweisheit zu testen: "Bier auf Wein, das lass sein". Was wäre ein geeignetes Experimentdesign und wie müsste die entsprechende ANOVA aussehen?
2. Das erste Experiment liefert keine richtigen Ergebnisse. Euch kommt aber in den Sinn, dass der Spruch vielleicht von einem Mann stammt und der Rat daher vielleicht nur auf Männer zutrifft. Wie würdet ihr das testen und wie würde eine dazugehörige ANOVA aussehen?

### 8.2.8 Lösungen II

1. Stellt euch vor, ihr werdet beauftragt folgende Volksweisheit zu testen: "Bier auf Wein, das lass sein". Was wäre ein geeignetes Experimentdesign und wie müsste die entsprechende ANOVA aussehen?

```
aov(AV ~ Reihenfolge + Error(ID - Variable / Reihenfolge),
d)
```

2. Das erste Experiment liefert keine richtigen Ergebnisse. Euch kommt aber in den Sinn, dass der Spruch vielleicht von einem Mann stammt und der Rat daher vielleicht nur auf Männer zutrifft. Wie würdet ihr das testen und wie würde eine dazugehörige ANOVA aussehen?

```
aov(AV ~ Reihenfolge * Geschlecht + Error(ID - Variable / Reihenfolge),
d)
```

### 8.2.9 Übungen III

1. a. Lest die Datei „LungCapData.csv“ in R ein und verschafft euch einen Überblick über die Daten.  
b. Gibt es NAs?
2. Berechnet, ob der Faktor GENDER einen signifikaten Einfluss auf die Lungenkapazität hat.
3. Berechnet, ob der Faktor CAESAREAN einen signifikaten Einfluss auf die Lungenkapazität hat.
4. Berechnet, ob es eine Interaktion zwischen den Faktoren CAESAREAN, SMOKE für die über 18-Jährigen gibt.
5. Berechnet, ob es eine Interaktion zwischen den Faktoren CAESAREAN, SMOKE und GENDER für die über 18-Jährigen gibt.

### 8.2.10 Lösungen III

1. a. Lest die Datei „LungCapData.csv“ in R ein und verschafft euch einen Überblick über die Daten.

```
d <- read.csv("docs/data/tut8/LungCapData.csv", sep = ";")
head(d)

##   LungCap Age Height Smoke Gender Caesarean
## 1   6.475   6   62.1    no male      no
## 2  10.125  18   74.7   yes female     no
## 3   9.550  16   69.7   no female    yes
## 4  11.125  14   71.0    no male      no
## 5   4.800   5   56.9    no male      no
## 6   6.225  11   58.7   no female     no

summary(d)

##       LungCap           Age          Height         Smoke        Gender
##  Min.   : 0.507   Min.   : 3.00   Min.   :45.30   no :648   female:358
##  1st Qu.: 6.150  1st Qu.: 9.00   1st Qu.:59.90   yes: 77  male  :367
##  Median : 8.000  Median :13.00   Median :65.40
##  Mean   : 7.863  Mean   :12.33   Mean   :64.84
##  3rd Qu.: 9.800  3rd Qu.:15.00   3rd Qu.:70.30
##  Max.   :14.675  Max.   :19.00   Max.   :81.80
##       Caesarean
##       no :561
##       yes:164
##       
```

### 8.2.11 Lösungen III

1. b. Gibt es NAs?

```
# prozent NAs pro spalte
(apply(is.na(d), 2, sum) / length(d[, 1])) * 100

##   LungCap           Age          Height         Smoke        Gender Caesarean
##       0             0             0            0            0            0

# alle zeilen mit NAs
d[!complete.cases(d), ]

## [1] LungCap     Age          Height         Smoke        Gender      Caesarean
## <0 rows> (or 0-length row.names)
```

### 8.2.12 Lösungen III

2. Berechnet, ob der Faktor GENDER einen signifikaten Einfluss auf die Lungenkapazität hat.

```
summary(aov(LungCap ~ Gender, d))

##              Df Sum Sq Mean Sq F value    Pr(>F)
## Gender          1   148   147.96   21.47 0.00000426 ***
```

```
## Residuals    723   4983    6.89
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3. Berechnet, ob der Faktor CAESAREAN einen signifikanten Einfluss auf die Lungenkapazität hat.

```
summary(aov(LungCap ~ Caesarean, d))
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
## Caesarean	1	2	2.331	0.329	0.567
## Residuals	723	5128	7.093		

### 8.2.13 Lösungen III

4. Berechnet, ob es eine Interaktion zwischen den Faktoren CAESAREAN, SMOKE für die über 18-Jährigen gibt.

```
dsub <- subset(d, Age >= 18)
summary(aov(LungCap ~ Caesarean * Smoke, dsub))
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
## Caesarean	1	0.01	0.010	0.005	0.947
## Smoke	1	3.77	3.768	1.638	0.205
## Caesarean:Smoke	1	1.84	1.845	0.802	0.373
## Residuals	76	174.83	2.300		

5. Berechnet, ob es eine Interaktion zwischen den Faktoren CAESAREAN, SMOKE und GENDER für die über 18-Jährigen gibt.

```
summary(aov(LungCap ~ Caesarean * Smoke * Gender, dsub))
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
## Caesarean	1	0.01	0.01	0.005	0.941257
## Smoke	1	3.77	3.77	1.981	0.163565
## Gender	1	31.78	31.78	16.710	0.000112 ***
## Caesarean:Smoke	1	1.98	1.98	1.039	0.311355
## Caesarean:Gender	1	2.73	2.73	1.437	0.234571
## Smoke:Gender	1	3.04	3.04	1.598	0.210287
## Caesarean:Smoke:Gender	1	0.23	0.23	0.120	0.730253
## Residuals	72	136.92	1.90		
## ---					
## Signif. codes:	0 '***'	0.001 '**'	0.01 '*'	0.05 '.'	0.1 ' '

## 8.3 Alternative ANOVA-Befehle

### 8.3.1 afex-ANOVA

Zur Erinnerung: Die Reihenfolge der positionalen Argumente könnt ihr nur dann ignorieren, wenn ihr sie benennt. Wenn ihr sie nicht benennt, müsst ihr die untere Abfolge einhalten. Allerdings ist euer Code besser nachvollziehbar (für Kollegen, Reviewer, etc.), wenn ihr alle Argumente per Schlüsselwort angegeben (und nicht bloß positionale vorgeht).

```
install.packages("afex")
library(afex)
aov_ez(id = "ID-Variable",
       dv = "Abhängige Variable",
       data = d,
       within = c("UVwithin1", "UVwithin2"),
       between = c("UVbetween1", "UVbetween2"))
)
```

### 8.3.2 ez-ANOVA

```
install.packages("ez")
library(ez)
ezANOVA(data = d,
         dv = Abhängige - Variable,
         wid = ID - Variable,
         within = .(UVwithin1, UVwithin2),
         between = .(UVbetween1, UVbetween2))
```

### 8.3.3 Übungen IV

Jetzt beschäftigen wir uns mit einem Experiment, das im psycholinguistischen Experimentalpraktikum entstanden ist. Das Design ist wie folgt:

PRÄPOSITION\*LESART: 24 Items, 32 Proband\*innen; jede\*r Proband\*in sah ein Item genau einmal unter einer Bedingung, und alle Bedingungen gleich oft. Zuordnung nach lateinischem Quadrat.

1. a. Der Klempner wurde ins Gefängnis begleitet, weil er seine Mutter erschlagen hatte.
- b. Der Klempner wurde ins Gefängnis begleitet, weil ein Abwasserrohr geplatzt war.
- c. Der Klempner wurde zum Gefängnis begleitet, weil er seine Mutter erschlagen hatte.
- d. Der Klempner wurde zum Gefängnis begleitet, weil ein Abwasserrohr geplatzt war.

Lest die Datei “dzi.dat” ein, verschafft euch einen Überblick und berechnet die deskriptive Statistik und die richtige ANOVA in drei Ausführungen: base R, afex und ez.

### 8.3.4 Lösungen IV

Datei einlesen

```
d <- read.table("docs/data/tut8/psych.dat", sep = "\t", header = T)
```

Überblick verschaffen

```
head(d)
```

	gender	age	fach	subject	variant	experiment	item	condition
## 1	m	19	LADe-EngPhilologie	1	1	1	6	b
## 2	m	19	LADe-EngPhilologie	1	1	1	14	b
## 3	m	19	LADe-EngPhilologie	1	1	1	13	a
## 4	m	19	LADe-EngPhilologie	1	1	1	12	d
## 5	m	19	LADe-EngPhilologie	1	1	1	15	b

```

## 6      m 19 LADe-EngPhilologie      1      1      1    7      c
## judgement verb   gen prep
## 1          7 passive episodic ins
## 2          7 passive episodic ins
## 3          7 passive generic ins
## 4          7 modal episodic zum
## 5          7 passive episodic ins
## 6          7 modal generic zum

```

### 8.3.5 Lösungen IV

```
summary(d)
```

```

## gender      age      fach      subject      variant
## m:168  Min. :18.0  De-Gesch   :114  Min. : 1.0  Min. :1.00
## w:426  1st Qu.:19.0  WiPaed    : 95  1st Qu.: 8.0  1st Qu.:2.00
##               Median :20.0  LADe-Sport  : 54  Median :16.0  Median :4.00
##               Mean   :21.4  De        : 38  Mean   :16.4  Mean   :4.47
##               3rd Qu.:21.0  LADe-Gesch : 27  3rd Qu.:25.0  3rd Qu.:7.00
##               Max.  :33.0  De-AllgSprachw: 19  Max.  :32.0  Max.  :8.00
##               (Other)       :247
## experiment    item      condition judgement      verb
## Min.  :1  Min.  : 1.00  a:148  Min.  :1.000  modal  :314
## 1st Qu.:1  1st Qu.: 5.00  b:150  1st Qu.:5.000  passive:280
## Median :1  Median :12.00  c:146  Median :7.000
## Mean   :1  Mean   :11.54  d:150  Mean   :5.944
## 3rd Qu.:1  3rd Qu.:18.00                    3rd Qu.:7.000
## Max.  :1  Max.  :23.00                    Max.  :7.000
##
##      gen      prep
## episodic:300  ins:298
## generic  :294  zum:296
##
## 
## 
## 
## 
```

### 8.3.6 Lösungen IV

```
d[!complete.cases(d), ]
```

```

## [1] gender      age      fach      subject      variant      experiment
## [7] item        condition  judgement  verb       gen        prep
## <0 rows> (or 0-length row.names)
str(d)

## 'data.frame': 594 obs. of 12 variables:
## $ gender      : Factor w/ 2 levels "m","w": 1 1 1 1 1 1 1 1 1 ...
## $ age         : int 19 19 19 19 19 19 19 19 19 ...
## $ fach        : Factor w/ 19 levels "De","De-AllgSprachw",...
## $ subject     : int 1 1 1 1 1 1 1 1 1 ...
## $ variant     : int 1 1 1 1 1 1 1 1 1 ...
## $ experiment   : int 1 1 1 1 1 1 1 1 1 ...
## $ item         : int 6 14 13 12 15 7 22 11 16 4 ...
## $ condition   : Factor w/ 4 levels "a","b","c","d": 2 2 1 4 2 3 3 3 3 4 ...
```

```
## $ judgement : int 7 7 7 7 7 7 7 7 7 ...
## $ verb      : Factor w/ 2 levels "modal","passive": 2 2 2 1 2 1 2 1 2 1 ...
## $ gen       : Factor w/ 2 levels "episodic","generic": 1 1 2 1 1 2 2 2 2 1 ...
## $ prep      : Factor w/ 2 levels "ins","zum": 1 1 1 2 1 2 2 2 2 2 ...
```

### 8.3.7 Faktoren erstellen

```
d$subject <- factor(d$subject)
d$item <- factor(d$item)
```

### 8.3.8 Lösungen IV

Deskriptive Statistik

```
library(psych)

describeBy(d$judgement, list(d$gen, d$prep), mat = T, digits = 2)
```

	item	group1	group2	vars	n	mean	sd	median	trimmed	mad	min	max	range
## X11	1	episodic	ins	1	150	5.55	1.85	6	5.87	1.48	1	7	6
## X12	2	generic	ins	1	148	6.11	1.26	7	6.35	0.00	2	7	5
## X13	3	episodic	zum	1	150	6.08	1.34	7	6.35	0.00	1	7	6
## X14	4	generic	zum	1	146	6.04	1.39	7	6.32	0.00	1	7	6
		skew	kurtosis	se									
## X11	-1.11		-0.03	0.15									
## X12	-1.42		1.07	0.10									
## X13	-1.61		2.20	0.11									
## X14	-1.53		1.64	0.11									

### 8.3.9 Lösungen IV

Base R

```
aovsub <- summary(aov(judgement ~ prep * gen +
  Error(subject / (prep * gen)), d))
aovite <- summary(aov(judgement ~ (prep * gen) +
  Error(item / (prep * gen)), d))
```

### 8.3.10 Lösungen IV

Base R

```
aovsub
##
## Error: subject
##              Df Sum Sq Mean Sq F value Pr(>F)
## prep        1    5.4   5.428   0.412  0.526
## gen         1   12.8  12.754   0.967  0.334
## prep:gen    1    2.7   2.707   0.205  0.654
## Residuals  28  369.2  13.187
##
## Error: subject:prep
```

```

##          Df Sum Sq Mean Sq F value Pr(>F)
## prep      1   6.79   6.795   3.630 0.0667 .
## gen       1   0.30   0.302   0.161 0.6908
## prep:gen  1   0.00   0.002   0.001 0.9758
## Residuals 29  54.29   1.872
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Error: subject:gen
##          Df Sum Sq Mean Sq F value Pr(>F)
## gen       1  11.56  11.555   8.426 0.00687 **
## prep:gen  1  12.72  12.715   9.272 0.00481 **
## Residuals 30  41.14   1.371
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Error: subject:prep:gen
##          Df Sum Sq Mean Sq F value Pr(>F)
## prep:gen  1  10.93  10.929   4.563 0.0407 *
## Residuals 31  74.24   2.395
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Error: Within
##          Df Sum Sq Mean Sq F value Pr(>F)
## Residuals 466  721.1   1.547

aovitem

## 
## Error: item
##          Df Sum Sq Mean Sq F value Pr(>F)
## prep      1   2.55   2.554   0.638 0.4369
## gen       1   5.47   5.471   1.367 0.2607
## prep:gen  1  22.72  22.718   5.674 0.0309 *
## Residuals 15  60.06   4.004
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Error: item:prep
##          Df Sum Sq Mean Sq F value Pr(>F)
## prep      1   7.77   7.767   3.603 0.0759 .
## gen       1  10.85  10.854   5.035 0.0393 *
## prep:gen  1   1.99   1.986   0.921 0.3514
## Residuals 16  34.49   2.156
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Error: item:gen
##          Df Sum Sq Mean Sq F value Pr(>F)
## gen       1  10.33  10.329   2.462 0.135
## prep:gen  1   0.87   0.872   0.208 0.654
## Residuals 17  71.32   4.195
## 
## Error: item:prep:gen
##          Df Sum Sq Mean Sq F value Pr(>F)
## prep:gen  1  13.12  13.118   8.933 0.00787 **
## Residuals 18  26.43   1.468
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
```

```
## Error: Within
##          Df Sum Sq Mean Sq F value Pr(>F)
## Residuals 518 1055   2.037
```

### 8.3.11 Lösungen IV

afex

```
library(afex)
aov_ez(id = "subject", dv = "judgement", within = c("gen", "prep"),
       data = d)

## Anova Table (Type 3 tests)
##
## Response: judgement
##      Effect    df MSE      F ges p.value
## 1     gen 1, 31 0.30 9.07 ** .02    .005
## 2     prep 1, 31 0.37 4.64 * .01    .04
## 3 gen:prep 1, 31 0.52 4.85 * .02    .04
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '+' 0.1 ' ' 1

aov_ez(id = "item", dv = "judgement", within = c("gen", "prep"),
       data = d)

## Anova Table (Type 3 tests)
##
## Response: judgement
##      Effect    df MSE      F ges p.value
## 1     gen 1, 18 0.52  2.57 .04    .13
## 2     prep 1, 18 0.34  2.69 .03    .12
## 3 gen:prep 1, 18 0.19 9.09 ** .05    .007
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '+' 0.1 ' ' 1
```

### 8.3.12 Lösungen IV

ez

```
library(ez)
ezANOVA(data = d, wid = subject, dv = judgement, within = .(prep, gen))

## $ANOVA
##      Effect DFn DFd      F      p p<.05      ges
## 2     prep    1  31 4.641385 0.039094646    * 0.01370960
## 3     gen    1  31 9.073004 0.005128325    * 0.02213167
## 4 prep:gen  1  31 4.852868 0.035162122    * 0.02039948

ezANOVA(data = d, wid = item, dv = judgement, within = .(prep, gen))

## $ANOVA
##      Effect DFn DFd      F      p p<.05      ges
## 2     prep    1  18 2.694220 0.118069072    0.02947056
## 3     gen    1  18 2.574507 0.125999408    0.04226753
## 4 prep:gen  1  18 9.087690 0.007445464    * 0.05379394
```

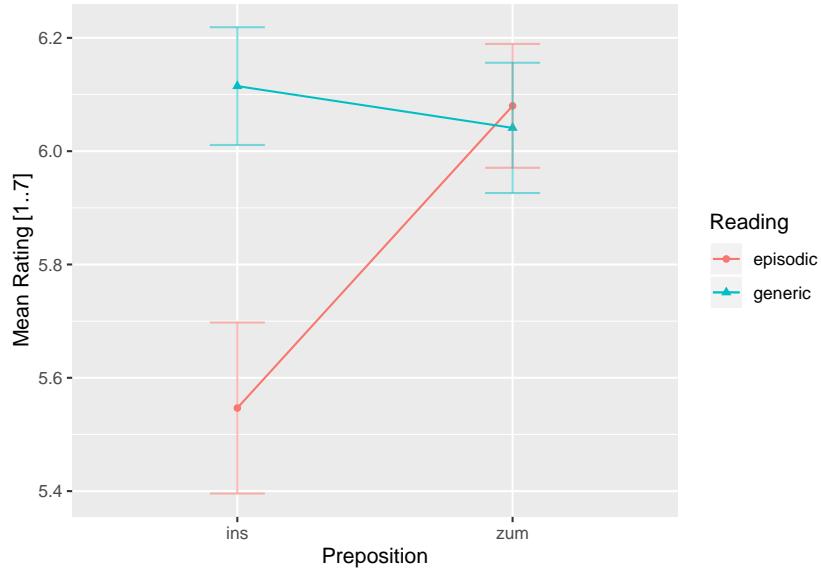
## 8.4 Nur Kurz: Plots

### 8.4.1 Ein Plot zum Abschluss

```
library(ggplot2)
p <- ggplot(data = d, aes(y = judgement, x = prep, shape = gen,
color = gen, group = gen)) +
  stat_summary(fun.y = mean, geom = "line") +
  stat_summary(fun.y = mean, geom = "point") +
  stat_summary(fun.data = mean_se, geom = "errorbar",
width = 0.2, alpha = .5, linetype = 1) +
  labs(y = "Mean Rating [1..7]", x = "Preposition",
shape = "Reading", color = "Reading")
```

### 8.4.2 Ein Plot zum Abschluss

p



### 8.4.3 Zum Nachdenken

Wie könnte man das folgende Meme in ein Experiment übersetzen? Wie sähe der Versuchsaufbau und wie die afex-ANOVA dazu aus?



mack  
@allbageldiet

▼

google maps: "12 minute walk"

me (gay, caffeinated): "8 minute walk"

#### 8.4.4 Der Glühwein ruft

Beste Grüße und bis nächste Woche!

# 9

---

## Sitzung 9

### 9.1 Experimentanalyse

#### 9.1.1 Der Plan

Heute will ich mit euch so tun, als würden wir gemeinsam ein komplettes Experiment analysieren. Das dient einerseits der Vorbereitung auf die Prüfungsleistung, wo ihr genau das machen müsst, andererseits ist es als kleine Wiederholungssitzung gedacht. Ihr könnt also überprüfen, welche Bereiche aus dem R-Handwerkszeug, das hier bereits Stoff war, schon bei euch sitzen und welche ihr euch vielleicht nochmal angucken solltet.

Genauer werden wir Folgendes machen:

- kurze Einführung in das Experimentdesign
- Rohdaten einlesen
- Daten für die Analyse aufbereiten
  - überflüssige Spalten entfernen
  - unabhängige Variablen codieren
  - Spalten zu Faktoren umwandeln
- Deskriptive Statistik
- (Visualisierung)
- (ein kleines bisschen) Inferenzstatistik

#### 9.1.2 Einführung ins Experiment

Yuqiu Chen, Mailin Antomo und ich haben ein Experiment zu folgender Fragestellung durchgeführt:

Warum gibt es einen Unterschied zwischen den beiden Satzpaaren:

1. a. Did somebody steal a car?  
b. # It was the duck [*PSP* who stole a car].*it*-Cleft
2. a. Did somebody steal a car?

b. I know [*PSP* that the duck stole a car].

*know*

Scheinbar verhalten sich nicht alle präspositionsauslösenden Ausdücke gleich in Bezug auf at-issuehood. Können wir diesen intuitiven Unterschied empirisch festigen und auf die soft-hard-Dichotomie von Präspositionsauslösern zurückführen?

### 9.1.3 Einführung ins Experiment

Faktoren (unabhängige Variablen):

- TRIGGER: Hard Trigger (*it*-Clefts und Adverbien) vs Soft Trigger (fiktive Verben) vs Appositiver Relativsatz
- ISSUEHOOD: At-Issue vs Non-At-Issue
- STAGE: Erwachsene vs Kinder

Methode (abhängige Variable):

- Akzeptabilitätsrating (1 bis 5) des b-Satzes in Bezug auf die Frage in a

## 9.2 Daten einlesen und aufbereiten

### 9.2.1 Daten einlesen und Überblick verschaffen

```
d <- read.csv("docs/data/tut9/datapre.csv", sep = ";")
head(d)

##   V1          V2 V3 V4      V5          V6 V7
## 1 id 3536fed0235c7b34a33ddf06fb91b390 ip ::1 time(s) 1558796407.625 data
## 2 id 3536fed0235c7b34a33ddf06fb91b390 ip ::1 time(s) 1558796437.5646 data
## 3 id 3536fed0235c7b34a33ddf06fb91b390 ip ::1 time(s) 1558796471.0036 data
## 4 id 3536fed0235c7b34a33ddf06fb91b390 ip ::1 time(s) 1558796501.9807 data
## 5 id 3536fed0235c7b34a33ddf06fb91b390 ip ::1 time(s) 1558796538.352 data
## 6 id 3536fed0235c7b34a33ddf06fb91b390 ip ::1 time(s) 1558796576.8658 data
##          V8 V9 V10 V11 V12 V13 V14 V15          V16 V17 age months gender
## 1 schaffen4_at 5 NA 0 0 NA NA 0 mailin NA 5 1 w
## 2 schaffen1_non 5 NA 0 0 NA NA 0 mailin NA 5 1 w
## 3 cleft2_non 5 NA 0 0 NA NA 0 mailin NA 5 1 w
## 4 appRel3_non 5 NA 0 0 NA NA 0 mailin NA 5 1 w
## 5 appRel9_non 4 NA 0 0 NA NA 0 mailin NA 5 1 w
## 6 gewinnen2_non 4 NA 0 0 NA NA 0 mailin NA 5 1 w
```

### 9.2.2 Überblick II

```
summary(d)

##   V1          V2 V3 V4
##  id:1181  0up71acsusa12ruvaf31lod4h6 : 30  ip:1181  ::1:1181
##            28af308e70043f33346261c3fe3a79b8: 30
##            2oocalgtfmpphi3ro9s68fu990 : 30
##            3536fed0235c7b34a33ddf06fb91b390: 30
##            3gkj6ts9s53ytfhk8v1r4d07r1 : 30
```

```

##          3jtpb1nfmp15qb2fe95qkjc8j5      : 30
##          (Other)                      :1001
##          V5              V6              V7              V8
## time(s):1181  15.431.601.928.574: 1  data:1181  auch3_at      : 39
##                  15.431.602.323.662: 1          appRel10_at   : 26
##                  15.431.603.044.837: 1          appRel13_non  : 26
##                  15.431.603.431.064: 1          appRel9_non   : 26
##                  15.431.603.785.808: 1          gewinnen2_non : 26
##                  15.431.604.161.861: 1          schaffen2_at  : 26
##          (Other)                      :1175          (Other)       :1012
##          V9              V10             V11             V12             V13
## Min.   :1.000  Mode:logical  Min.   :0  Min.   :0  Mode:logical
## 1st Qu.:2.000  NA's:1181    1st Qu.:0  1st Qu.:0  NA's:1181
## Median :4.000               Median :0  Median :0
## Mean   :3.466               Mean   :0  Mean   :0
## 3rd Qu.:5.000               3rd Qu.:0 3rd Qu.:0
## Max.   :5.000               Max.   :0  Max.   :0
##
##          V14             V15             V16             V17             age
## Mode:logical  Min.   :0  Imke   :180  Mode:logical  Min.   : 4.00
## NA's:1181     1st Qu.:0  Linda  :176  NA's:1181    1st Qu.: 5.00
##                   Median :0  mailin :120  Median :18.00
##                   Mean   :0  Mailin : 88  Mean   :16.23
##                   3rd Qu.:0  Susanne:152 3rd Qu.:24.00
##                   Max.   :0  Y      :420  Max.   :65.00
##                   yuqiu  :45
##
##          months        gender
## Min.   : 0.000  m:292
## 1st Qu.: 0.000  w:889
## Median : 0.000
## Mean   : 2.767
## 3rd Qu.: 6.000
## Max.   :11.000
##
```

### 9.2.3 Was ist zu tun?

- leere und überflüssige Spalten entfernen
  - welche sind das?
- Faktoren codieren
  - V8 enthält alle Informationen für die Faktoren TRIGGER und ISSUENESS
  - die Spalte age enthält die Informationen für STAGE
- Spalten umbenennen

### 9.2.4 Leere und überflüssige Spalten entfernen

Spalten auswählen, die wir behalten wollen:

```
(apply(is.na(d), 2, sum) / length(d[, 1])) * 100
```

```

##      V1      V2      V3      V4      V5      V6      V7      V8      V9      V10     V11
##      0       0       0       0       0       0       0       0       0      100       0
##      V12     V13     V14     V15     V16     V17     age  months gender
##      0      100     100      0       0      100       0       0       0

```

```
d <- d[c(2, 8, 9, 16, 18, 19, 20)]
head(d)

##          V2      V8  V9    V16 age months gender
## 1 3536fed0235c7b34a33ddf06fb91b390 schaffen4_at 5 mailin 5 1 w
## 2 3536fed0235c7b34a33ddf06fb91b390 schaffen1_non 5 mailin 5 1 w
## 3 3536fed0235c7b34a33ddf06fb91b390 cleft2_non 5 mailin 5 1 w
## 4 3536fed0235c7b34a33ddf06fb91b390 appRel3_non 5 mailin 5 1 w
## 5 3536fed0235c7b34a33ddf06fb91b390 appRel9_non 4 mailin 5 1 w
## 6 3536fed0235c7b34a33ddf06fb91b390 gewinnen2_non 4 mailin 5 1 w
```

### 9.2.5 Leere und überflüssige Spalten entfernen

Alternative mit `subset`-Funktion:

```
d <- subset(d, select = c(V2, V8, V9, V16, age, months, gender))
```

### 9.2.6 Faktor Issueness

Was wir machen müssen, ist nach Mustern in der Itembenennung zu schauen, die wir verwenden können. Dazu bieten sich “`_at`” und “`_non`” an. Eine Funktion, die wir uns zunutze machen können, ist im Paket “`stringr`” enthalten.

```
install.packages("stringr")
library(stringr)
```

Sie heißt `str_detect` und funktioniert so:

```
str_detect(String, Muster)
```

Ein Beispiel:

```
fruit <- c("apple", "banana", "pear", "pineapple")
str_detect(fruit, "na")

## [1] FALSE TRUE FALSE FALSE
```

### 9.2.7 Übung

- Benutzt `str_detect` um eine neue Spalte “issueness” anzulegen, die “at-issue” enthält, wenn das Item den String “`_at`” enthält, und “non-at-issue” bei “`_non`”

### 9.2.8 Lösung

- Benutzt `str_detect` um eine neue Spalte “issueness” anzulegen, die “at-issue” enthält, wenn das Item den String “`_at`” enthält, und “non-at-issue” bei “`_non`”

```
d$issueness[str_detect(d$V8, "_at")] <- "at-issue"
d$issueness[str_detect(d$V8, "_non")] <- "non-at-issue"
head(d)
```

```

##          V2      V8  V9    V16 age months gender
## 1 3536fed0235c7b34a33ddf06fb91b390 schaffen4_at 5 mailin  5   1     w
## 2 3536fed0235c7b34a33ddf06fb91b390 schaffen1_non 5 mailin  5   1     w
## 3 3536fed0235c7b34a33ddf06fb91b390 cleft2_non  5 mailin  5   1     w
## 4 3536fed0235c7b34a33ddf06fb91b390 appRel3_non  5 mailin  5   1     w
## 5 3536fed0235c7b34a33ddf06fb91b390 appRel9_non  4 mailin  5   1     w
## 6 3536fed0235c7b34a33ddf06fb91b390 gewinnen2_non 4 mailin  5   1     w
##           issues
## 1      at-issue
## 2 non-at-issue
## 3 non-at-issue
## 4 non-at-issue
## 5 non-at-issue
## 6 non-at-issue

```

### 9.2.9 Faktor Trigger

Die Liste von Triggern sieht wie folgt aus:

- Soft: schaffen, entdecken, gewinnen
- Hard: cleft, auch, wieder
- Appositzer Relativsatz: appRel

### 9.2.10 Übung II

1. Benutzt dieselbe Methode, um den Faktor TRIGGER zu codieren.

Tipp: `str_detect` erlaubt auch Listen von Mustern, deren Elemente durch “|” (read: oder) innerhalb eines Strings getrennt sind:

```

fruit <- c("apple", "banana", "pear", "pineapple")
str_detect(fruit, "na|ar") # TRUE bei "na" und "ar"

## [1] FALSE TRUE  TRUE FALSE
fruit[str_detect(fruit, "na|ar")]

## [1] "banana" "pear"

```

### 9.2.11 Lösung II

1. Benutzt dieselbe Methode, um den Faktor TRIGGER zu codieren

```

d$trigger[str_detect(d$V8, "schaffen|entdecken|gewinnen")] <- "soft"
d$trigger[str_detect(d$V8, "cleft|auch|wieder")] <- "hard"
d$trigger[str_detect(d$V8, "appRel")] <- "appRel"
head(d)

```

```

##          V2      V8  V9    V16 age months gender
## 1 3536fed0235c7b34a33ddf06fb91b390 schaffen4_at 5 mailin  5   1     w
## 2 3536fed0235c7b34a33ddf06fb91b390 schaffen1_non 5 mailin  5   1     w
## 3 3536fed0235c7b34a33ddf06fb91b390 cleft2_non  5 mailin  5   1     w
## 4 3536fed0235c7b34a33ddf06fb91b390 appRel3_non  5 mailin  5   1     w
## 5 3536fed0235c7b34a33ddf06fb91b390 appRel9_non  4 mailin  5   1     w
## 6 3536fed0235c7b34a33ddf06fb91b390 gewinnen2_non 4 mailin  5   1     w

```

```
##      issuesness trigger
## 1      at-issue    soft
## 2 non-at-issue    soft
## 3 non-at-issue   hard
## 4 non-at-issue appRel
## 5 non-at-issue appRel
## 6 non-at-issue    soft
```

### 9.2.12 Faktor Stage

Jetzt müssen wir noch die letzte unabhängige Variable kodieren und der schwerste Teil ist geschafft

Übung:

1. Erstellt den Faktor "stage" im Datenblatt mit den Stufen "adult" und "child"

### 9.2.13 Lösung III

```
d$stage[d$age >= 18] <- "adult"
d$stage[d$age <= 6] <- "child"
head(d)

# V2          V8  V9    V16 age months gender
## 1 3536fed0235c7b34a33ddf06fb91b390 schaffen4_at 5 mailin 5 1 w
## 2 3536fed0235c7b34a33ddf06fb91b390 schaffen1_non 5 mailin 5 1 w
## 3 3536fed0235c7b34a33ddf06fb91b390 cleft2_non 5 mailin 5 1 w
## 4 3536fed0235c7b34a33ddf06fb91b390 appRel3_non 5 mailin 5 1 w
## 5 3536fed0235c7b34a33ddf06fb91b390 appRel9_non 4 mailin 5 1 w
## 6 3536fed0235c7b34a33ddf06fb91b390 gewinnen2_non 4 mailin 5 1 w

##      issuesness trigger stage
## 1      at-issue    soft child
## 2 non-at-issue    soft child
## 3 non-at-issue   hard child
## 4 non-at-issue appRel child
## 5 non-at-issue appRel child
## 6 non-at-issue    soft child
```

### 9.2.14 Letzte Schritte

Wir haben es fast geschafft! Es bleiben noch zwei Dinge zu tun.

Übung:

1. Benennt die Spalten sinnvoll um
2. Überprüft, ob die wichtigen Spalten Faktoren sind und holt das ggf nach

### 9.2.15 Lösung IV

1. Benennt die Spalten sinnvoll um

```
names(d) <- c("id", "item", "judgment", "tester", "years", "months",
"gender", "issueness", "trigger", "stage")
```

### 9.2.16 Lösung IV

2. Überprüft, ob die wichtigen Spalten Faktoren sind und holt das ggf nach

```
str(d)

## 'data.frame': 1181 obs. of 10 variables:
## $ id      : Factor w/ 41 levels "071362c17b307af0925ac2d94d697527",...: 6 6 6 6 6 6 6 6 6 ...
## $ item    : Factor w/ 59 levels "appRel1_at","appRel1_non",...: 52 47 29 8 20 43 57 23 33 2 ...
## $ judgment: int 5 5 5 5 4 4 3 4 5 4 ...
## $ tester   : Factor w/ 7 levels "Imke","Linda",...: 3 3 3 3 3 3 3 3 3 ...
## $ years    : int 5 5 5 5 5 5 5 5 5 ...
## $ months   : int 1 1 1 1 1 1 1 1 1 ...
## $ gender   : Factor w/ 2 levels "m","w": 2 2 2 2 2 2 2 2 2 ...
## $ issueness: chr "at-issue" "non-at-issue" "non-at-issue" "non-at-issue" ...
## $ trigger   : chr "soft" "soft" "hard" "appRel" ...
## $ stage    : chr "child" "child" "child" "child" ...

d$issueness <- factor(d$issueness)
d$trigger <- factor(d$trigger)
d$stage <- factor(d$stage)
str(d[, 8:10])

## 'data.frame': 1181 obs. of 3 variables:
## $ issueness: Factor w/ 2 levels "at-issue","non-at-issue": 1 2 2 2 2 2 2 1 2 2 ...
## $ trigger   : Factor w/ 3 levels "appRel","hard",...: 3 3 2 1 1 3 2 2 2 1 ...
## $ stage     : Factor w/ 2 levels "adult","child": 2 2 2 2 2 2 2 2 2 2 ...
```

### 9.2.17 (Optionale Kür)

Um die spätere Analyse ein bisschen einfacher (und die by-item-Plots aufschlussreicher) zu machen, codieren wir im Folgenden noch die Spalte mit den Itemids (also ohne die At-Issueness-Information) und die lexikalischen Trigger (Wert in der Itemspalte ohne Zahl und ohne At-Issueness)

```
# generate a column that holds the item ids without at-issues
# replace strings like "_at" with nothing
d$itemid <- factor(gsub("_.*$", "", d$item))

# code lexical triggers
# replace strings like "5_non" with nothing
d$lex <- factor(gsub("[[:digit:]]_.*$", "", d$item))

head(d, 4)

##           id      item judgment tester years months
## 1 3536fed0235c7b34a33ddf06fb91b390 schaffen4_at      5 mailin      5     1
## 2 3536fed0235c7b34a33ddf06fb91b390 schaffen1_non      5 mailin      5     1
## 3 3536fed0235c7b34a33ddf06fb91b390 cleft2_non       5 mailin      5     1
## 4 3536fed0235c7b34a33ddf06fb91b390 appRel3_non       5 mailin      5     1

##   gender  issueness trigger stage itemid    lex
## 1      w   at-issue    soft child schaffen4 schaffen
## 2      w non-at-issue   soft child schaffen1 schaffen
## 3      w non-at-issue    hard child cleft2    cleft
## 4      w non-at-issue appRel child appRel3 appRel
```

## 9.3 Deskriptive Statistik

### 9.3.1 describe()/describeBy()

Für die descriptive Statistik können wir der Einfachheit halber den `describeBy`-Befehl verwenden

Paket laden:

```
library(psych)
```

Statistik rechnen lassen:

```
describe(...)  
describeBy(..., ...)
```

### 9.3.2 Lösungen V

```
describeBy(d$judgment, list(d$stage, d$issueness, d$trigger),  
          mat = T, digits = 2)

##      item group1      group2 group3 vars   n mean    sd median trimmed mad min
## X11     1 adult      at-issue appRel   1 99 1.85 0.83    2  1.77 1.48  1
## X12     2 child      at-issue appRel   1 99 3.36 1.45    4  3.44 1.48  1
## X13     3 adult non-at-issue appRel   1 100 3.99 0.95   4  4.12 1.48  1
## X14     4 child non-at-issue appRel   1 100 3.71 1.32   4  3.86 1.48  1
## X15     5 adult      at-issue hard    1 100 1.99 1.01   2  1.85 1.48  1
## X16     6 child      at-issue hard    1 94  3.39 1.48   4  3.49 1.48  1
## X17     7 adult non-at-issue hard    1 97  4.33 1.01   5  4.53 0.00  1
## X18     8 child non-at-issue hard    1 98  3.80 1.32   4  3.96 1.48  1
## X19     9 adult      at-issue soft   1 100 3.48 1.18   4  3.51 1.48  1
## X110   10 child      at-issue soft   1 97  3.70 1.35   4  3.85 1.48  1
## X111   11 adult non-at-issue soft   1 100 4.18 1.01   4  4.38 1.48  1
## X112   12 child non-at-issue soft   1 97  3.84 1.34   4  4.01 1.48  1
##      max range skew kurtosis se
## X11     4   3  0.71 -0.13 0.08
## X12     5   4 -0.28 -1.34 0.15
## X13     5   4 -0.97  0.44 0.09
## X14     5   4 -0.64 -0.84 0.13
## X15     5   4  0.95  0.08 0.10
## X16     5   4 -0.27 -1.44 0.15
## X17     5   4 -1.65  2.24 0.10
## X18     5   4 -0.79 -0.66 0.13
## X19     5   4 -0.28 -1.15 0.12
## X110   5   4 -0.62 -0.94 0.14
## X111   5   4 -1.53  2.15 0.10
## X112   5   4 -0.82 -0.66 0.14
```

## 9.4 Plots

### 9.4.1 Plan

Im Folgenden zeige ich euch zwei Boxplots der Daten, die dieselben Informationen zeigen. Nämlich wie sich die unterschiedlichen Faktoren (bzw deren Stufen) auf die Bewertungen auswirken

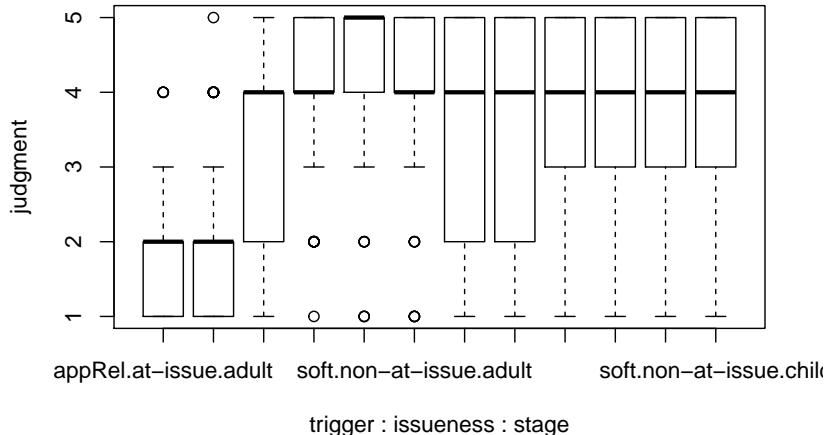
Zuerst kommt der `boxplot`-Befehl von R. Der Plot wird wenig übersichtlich sein.

Danach zeige ich euch einen Plot mit `ggplot2`, der wesentlich besser verständlich ist. Dazu brauchen wir Folgendes:

```
install.packages("ggplot2")
library(ggplot2)
```

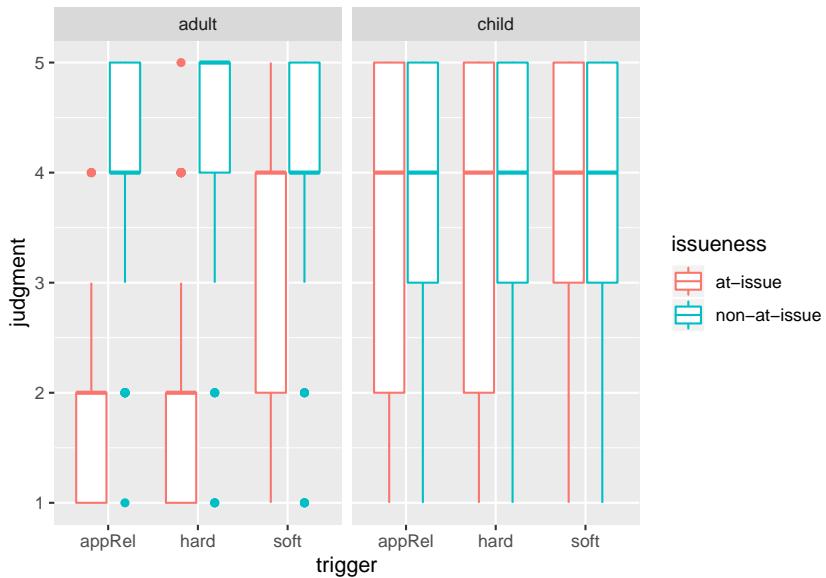
#### 9.4.2 Boxplot

```
boxplot(data = d, judgment ~ trigger:issueness:stage)
```



#### 9.4.3 Boxplot II

```
ggplot(data = d, aes(x = trigger, y = judgment, color = issueness)) +
  geom_boxplot() +
  facet_wrap(~stage)
```



## 9.5 Inferenzstatistik

### 9.5.1 Für einen Test entscheiden

Welchen Test müssen wir rechnen?

Wir müssen folgende Faktoren in unserer Analyse unterbringen und schauen, ob und wie sie sich auf die abhängige Variable ("judgment" – Akzeptabilitätsurteile) auswirken:

- ISSUENESS
- TRIGGER
- STAGE

Welcher Test bietet sich an?

### 9.5.2 Lösung

Die Varianzanalyse!

nächste Übung: schreibt einen Befehl für die Varianzanalyse mit dem `afex`-Paket. Falls ihr Hilfe braucht, schaut gerne in die vorherigen Folien oder gebt `?aov_ez` ein, nachdem ihr das Paket geladen habt.

```
library(afex)
aov_ez(id = "ID-Variable",
       dv = "Abhängige Variable",
       data = d,
```

```

    within = c("UVwithin1", "UVwithin2"),
    between = c("UVbetween1", "UVbetween2")
)

```

### 9.5.3 Lösung

```

# subjects
anovasub <- aov_ez("id", "judgment", d,
  within = c("issueness", "trigger"),
  between = "stage")
# items
anovaitem <- aov_ez("itemid", "judgment", d,
  within = c("issueness", "stage"),
  between = "trigger")

```

### 9.5.4 Lösung

```

# subjects
anovasub

## Anova Table (Type 3 tests)
##
## Response: judgment
##          Effect   df  MSE      F ges p.value
## 1        stage 1, 39 1.77  3.56 + .04   .07
## 2        issueness 1, 39 0.33 190.63 *** .31  <.0001
## 3     stage:issueness 1, 39 0.33 96.53 *** .19  <.0001
## 4        trigger 2, 78 0.39 17.35 *** .09  <.0001
## 5   stage:trigger 2, 78 0.39  7.36 ** .04   .001
## 6  issueness:trigger 2, 78 0.34 15.51 *** .07  <.0001
## 7 stage:issueness:trigger 2, 78 0.34  9.22 *** .04   .0003
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '+' 0.1 ' ' 1

```

### 9.5.5 Lösung

```

# items
anovaitem

## Anova Table (Type 3 tests)
##
## Response: judgment
##          Effect   df  MSE      F ges p.value
## 1        trigger 2, 26 0.22 15.80 *** .30  <.0001
## 2        issueness 1, 26 0.21 145.05 *** .64  <.0001
## 3   trigger:issueness 2, 26 0.21 12.64 *** .24  .0001
## 4        stage 1, 26 0.05 89.51 *** .21  <.0001
## 5   trigger:stage 2, 26 0.05 21.52 *** .12  <.0001
## 6  issueness:stage 1, 26 0.16 84.99 *** .45  <.0001
## 7 trigger:issueness:stage 2, 26 0.16  7.65 ** .13   .002
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '+' 0.1 ' ' 1

```

### 9.5.6 Für ausführlichere Wiedergabe der ANOVA

```
summary(anovasub)
```

```
##  
## Univariate Type III Repeated-Measures ANOVA Assuming Sphericity  
##  
##  
## (Intercept) 2950.73 1 69.078 39 1665.9223 < 2.2e-16 ***  
## stage 6.31 1 69.078 39 3.5631 0.0665353 .  
## issuueness 63.03 1 12.896 39 190.6296 < 2.2e-16 ***  
## stage:issuueness 31.92 1 12.896 39 96.5296 4.209e-12 ***  
## trigger 13.53 2 30.407 78 17.3550 5.823e-07 ***  
## stage:trigger 5.74 2 30.407 78 7.3608 0.0011793 **  
## issuueness:trigger 10.60 2 26.661 78 15.5099 2.133e-06 ***  
## stage:issuueness:trigger 6.30 2 26.661 78 9.2192 0.0002546 ***  
## ---  
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##  
##  
## Mauchly Tests for Sphericity  
##  
##  
## Test statistic p-value  
## trigger 0.93521 0.28007  
## stage:trigger 0.93521 0.28007  
## issuueness:trigger 0.91718 0.19347  
## stage:issuueness:trigger 0.91718 0.19347  
##  
##  
## Greenhouse-Geisser and Huynh-Feldt Corrections  
## for Departure from Sphericity  
##  
##  
## GG eps Pr(>F[GG])  
## trigger 0.93915 1.154e-06 ***  
## stage:trigger 0.93915 0.0015242 **  
## issuueness:trigger 0.92351 4.587e-06 ***  
## stage:issuueness:trigger 0.92351 0.0003904 ***  
## ---  
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##  
## HF eps Pr(>F[HF])  
## trigger 0.9850323 6.888931e-07  
## stage:trigger 0.9850323 1.256089e-03  
## issuueness:trigger 0.9673658 2.956515e-06  
## stage:issuueness:trigger 0.9673658 3.055378e-04
```

### 9.5.7 Exkurs: Sphärizitätskorrekturen

“In particular, RM ANOVA assumes sphericity. That is, the variances of the differences between all pairs of groups are equal. In repeated measures data, this would imply that the variance of e.g. time 1 and time 2 is the same as the variance of the difference of time 1 and time 3.” [Peter Flom \(2014\)](#)

### 9.5.8 Sphärizität in der afex-ANOVA

Die afex-ANOVA korrigiert automatisch die Freiheitsgrade (und damit den p-Wert) bei Sphärizitätsverletzungen. Dies kann wie folgt ausgestellt werden:

```
# subjects without sphericity correction
anovasub_nosphere <- aov_ez("id", "judgment", d,
  within = c("issueness", "trigger"),
  between = "stage",
  anova_table = list(correction = "none")) # <<
```

### 9.5.9 Vergleich beider Ergebnisse

```
anovasub

## Anova Table (Type 3 tests)
##
## Response: judgment
##          Effect      df  MSE      F ges p.value
## 1          stage 1, 39 1.77  3.56 + .04   .07
## 2          issueness 1, 39 0.33 190.63 *** .31  <.0001
## 3        stage:issueness 1, 39 0.33  96.53 *** .19  <.0001
## 4          trigger 1.88, 73.25 0.42 17.35 *** .09  <.0001
## 5        stage:trigger 1.88, 73.25 0.42  7.36 ** .04   .002
## 6    issueness:trigger 1.85, 72.03 0.37 15.51 *** .07  <.0001
## 7 stage:issueness:trigger 1.85, 72.03 0.37  9.22 *** .04   .0004
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '+' 0.1 ' ' 1
##
## Sphericity correction method: GG

anovasub_nosphere

## Anova Table (Type 3 tests)
##
## Response: judgment
##          Effect      df  MSE      F ges p.value
## 1          stage 1, 39 1.77  3.56 + .04   .07
## 2          issueness 1, 39 0.33 190.63 *** .31  <.0001
## 3        stage:issueness 1, 39 0.33  96.53 *** .19  <.0001
## 4          trigger 2, 78 0.39 17.35 *** .09  <.0001
## 5        stage:trigger 2, 78 0.39  7.36 ** .04   .001
## 6    issueness:trigger 2, 78 0.34 15.51 *** .07  <.0001
## 7 stage:issueness:trigger 2, 78 0.34  9.22 *** .04   .0003
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '+' 0.1 ' ' 1
```

## 9.6 Kontraste

### 9.6.1 Kontraste kodieren, aber wie?

```
install.packages("emmeans")

library(emmeans)
# referenztabelle für kontraste; specs sind die die Faktoren, dessen estimated marginal means (EMM) wir uns
(ref_id <- emmeans(anovasub,
  specs = c("stage", "trigger", "issueness")))
```

```

##  stage trigger issueness    emmean     SE   df lower.CL upper.CL
##  adult appRel at.issue      1.85 0.171 133     1.51    2.19
##  child appRel at.issue      3.34 0.169 129     3.01    3.68
##  adult hard  at.issue      1.99 0.171 133     1.66    2.33
##  child hard  at.issue      3.46 0.169 129     3.12    3.79
##  adult soft   at.issue      3.48 0.171 133     3.15    3.82
##  child soft   at.issue      3.65 0.169 129     3.31    3.98
##  adult appRel non.at.issue 3.99 0.171 133     3.66    4.33
##  child appRel non.at.issue 3.67 0.169 129     3.34    4.01
##  adult hard   non.at.issue 4.35 0.171 133     4.01    4.69
##  child hard   non.at.issue 3.84 0.169 129     3.51    4.18
##  adult soft   non.at.issue 4.18 0.171 133     3.85    4.52
##  child soft   non.at.issue 3.81 0.169 129     3.47    4.14
##
## Warning: EMMs are biased unless design is perfectly balanced
## Confidence level used: 0.95

```

### 9.6.2 Exkurs: Estimated Marginal Means

```

# let's generate a table with values
d <- tibble(
  reads = c("reads", "doesnt_read"),
  studies = c(1, 2),
  doesnt_study = c(3, 5)
)
d

## # A tibble: 2 x 3
##   reads      studies  doesnt_study
##   <chr>       <dbl>        <dbl>
## 1 reads         1            3
## 2 doesnt_read  2            5

# then lets transform the data into a tidy one (one row = one observation)
d <- d %>%
  pivot_longer(c("studies", "doesnt_study"),
               names_to = "prepares",
               values_to = "grade")
d

## # A tibble: 4 x 3
##   reads      prepares     grade
##   <chr>       <chr>       <dbl>
## 1 reads      studies      1
## 2 reads      doesnt_study 3
## 3 doesnt_read studies      2
## 4 doesnt_read doesnt_study 5

```

### 9.6.3 Exkurs cont'd

```

# then lets fit a model
fit <- aov(grade ~ reads * prepares, data = d)
# and compute estimated marginal means
# in a balanced design, EMMs just represent the cell means (here, because we only have one observation per ...
# in unbalanced design EMMs are adjusted so that all cell means are given equal weight
(ref <- emmeans(fit, c("reads", "prepares")))

## Warning in qt((1 - level)/adiv, df): NaNs produced

```

```
##  reads      prepares      emmean   SE df lower.CL upper.CL
##  doesnt_read doesnt_study     5 NaN  0     NaN     NaN
##  reads      doesnt_study     3 NaN  0     NaN     NaN
##  doesnt_read studies       2 NaN  0     NaN     NaN
##  reads      studies        1 NaN  0     NaN     NaN
##
## Confidence level used: 0.95
```

#### 9.6.4 Übung

Spezifiziert Vektoren für die folgenden Kontraste mithilfe der Referenztabelle:

- appositive RCs (at-issue) vs Hard Trigger (at-issue)
- Soft Trigger (at-issue) vs Hard Trigger (at-issue)
- At-issue, Kinder vs At-issue, Erwachsene

Beispiel: um adult\_appRel mit child\_appRel (und nichts anderes) zu vergleichen, verwenden wir folgende Spezifizierung:

```
# adult_appRel versus child_appRel
RCAvsC <- c(-1, 1, 0, 0, 0, 0, -1, 1, 0, 0, 0, 0)
```

#### 9.6.5 Lösung

Spezifiziert Vektoren für die folgenden Kontraste mithilfe der Referenztabelle:

- appositive RCs (at-issue) vs Hard Trigger (at-issue)
- Soft Trigger (at-issue) vs Hard Trigger (at-issue)
- At-issue, Kinder vs At-issue, Erwachsene

```
# kontraste spezifizieren
test_RCvHTat <- c(1, 1, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0)
test_STvHTat <- c(0, 0, -1, -1, 1, 1, 0, 0, 0, 0, 0, 0)
test_ATcvATA <- c(-1, 1, -1, 1, -1, 1, 0, 0, 0, 0, 0, 0)
```

#### 9.6.6 Kontraste berechnen

Bonferronikorrektur nur für *post hoc* Kontraste absolut nötig. Bei *a priori* spezifizierten (das heißt von der Hypothese gedeckten) Kontrasten kann diese Korrektur weggelassen werden.

```
# bonferroni-korrigierte tests berechnen
summary(contrast(ref_id, list(RCvHTat = test_RCvHTat,
  STvHTat = test_STvHTat,
  ATcvATA = test_ATcvATA),
adjust = "bonferroni", # Korrektur für multiple Tests
infer = TRUE # zeigt p-Werte und KonfInt
))

## contrast estimate   SE   df lower.CL upper.CL t.ratio p.value
## RCvHTat    -0.259 0.267 155.3   -0.906   0.387 -0.971  0.9997
## STvHTat     1.680 0.267 155.3    1.034   2.327  6.288 <.0001
## ATcvATA     3.123 0.555  53.1    1.752   4.495  5.630 <.0001
##
## Confidence level used: 0.95
## Conf-level adjustment: bonferroni method for 3 estimates
## P value adjustment: bonferroni method for 3 tests
```

### 9.6.7 Zum selbst Ausprobieren

Warning von Thomas (pc): "D.h. ich wuerde den Leuten sagen, dass der letzte Code-Schnipsel (wo Du dann *alle* Vergleiche anschauust) eine Mischung aus *prae-* und *post-hoc*-Betrachtung ist. Und dass das natuerlich fuer moralisch sproede bzw. vom publication bias und Befristung weichgeklopfte Gemueter gewisse Verlockungen/Gefahren birgt. You see?"

```
# alle paarweisen vergleiche für die beiden level von issueness berechnen
ref_big <- emmeans(anovasub, ~ stage * trigger | issueness)

## issueness = at.issue:
##   stage trigger emmean    SE df lower.CL upper.CL
##   adult appRel     1.85 0.171 133    1.51    2.19
##   child appRel     3.34 0.169 129    3.01    3.68
##   adult hard      1.99 0.171 133    1.66    2.33
##   child hard      3.46 0.169 129    3.12    3.79
##   adult soft      3.48 0.171 133    3.15    3.82
##   child soft      3.65 0.169 129    3.31    3.98
##
##   issueness = non.at.issue:
##   stage trigger emmean    SE df lower.CL upper.CL
##   adult appRel     3.99 0.171 133    3.65    4.33
##   child appRel     3.67 0.169 129    3.34    4.01
##   adult hard      4.35 0.171 133    4.01    4.69
##   child hard      3.84 0.169 129    3.51    4.18
##   adult soft      4.18 0.171 133    3.85    4.52
##   child soft      3.81 0.169 129    3.47    4.14
##
## Warning: EMMs are biased unless design is perfectly balanced
## Confidence level used: 0.95
```

### 9.6.8 Ergebnisse des großen Paarvergleichs

```
summary(contrast(ref_big, method = "pairwise",
  adjust = "bonferroni",
  infer = TRUE
))

## issueness = at.issue:
##   contrast          estimate    SE df lower.CL upper.CL t.ratio
##   adult,appRel - child,appRel -1.4980 0.241 131   -2.218  -0.7779 -6.220
##   adult,appRel - adult,hard   -0.1475 0.191 155   -0.718   0.4227 -0.771
##   adult,appRel - child,hard   -1.6099 0.241 131   -2.330   -0.8898 -6.684
##   adult,appRel - adult,soft   -1.6375 0.191 155   -2.208   -1.0673 -8.562
##   adult,appRel - child,soft   -1.8004 0.241 131   -2.520   -1.0802 -7.475
##   child,appRel - adult,hard   1.3505 0.241 131    0.630   2.0706  5.607
##   child,appRel - child,hard   -0.1119 0.187 155   -0.668   0.4446 -0.600
##   child,appRel - adult,soft   -0.1395 0.241 131   -0.860   0.5806 -0.579
##   child,appRel - child,soft   -0.3024 0.187 155   -0.859   0.2541 -1.620
##   adult,hard - child,hard    -1.4624 0.241 131   -2.182   -0.7423 -6.072
##   adult,hard - adult,soft    -1.4900 0.191 155   -2.060   -0.9198 -7.790
##   adult,hard - child,soft    -1.6529 0.241 131   -2.373   -0.9327 -6.863
##   child,hard - adult,soft    -0.0276 0.241 131   -0.748   0.6925 -0.115
##   child,hard - child,soft    -0.1905 0.187 155   -0.747   0.3660 -1.020
##   adult,soft - child,soft    -0.1629 0.241 131   -0.883   0.5573 -0.676
##   p.value
##   <.0001
```

```

## 1.0000
## <.0001
## <.0001
## <.0001
## <.0001
## 1.0000
## 1.0000
## 1.0000
## <.0001
## <.0001
## <.0001
## 1.0000
## 1.0000
## 1.0000
## issueness = non.at.issue:
## contrast   estimate    SE  df lower.CL upper.CL t.ratio
## adult appRel - child appRel  0.3194 0.241 131  -0.401  1.0395  1.326
## adult appRel - adult hard  -0.3525 0.191 155  -0.923  0.2177 -1.843
## adult appRel - child hard  0.1519 0.241 131  -0.568  0.8720  0.631
## adult appRel - adult soft  -0.1900 0.191 155  -0.760  0.3802 -0.993
## adult appRel - child soft  0.1868 0.241 131  -0.533  0.9069  0.776
## child appRel - adult hard -0.6719 0.241 131  -1.392  0.0483 -2.790
## child appRel - child hard -0.1675 0.187 155  -0.724  0.3890 -0.897
## child appRel - adult soft  -0.5094 0.241 131  -1.229  0.2107 -2.115
## child appRel - child soft  -0.1325 0.187 155  -0.689  0.4239 -0.710
## adult hard - child hard   0.5044 0.241 131  -0.216  1.2245  2.094
## adult hard - adult soft   0.1625 0.191 155  -0.408  0.7327  0.850
## adult hard - child soft   0.5393 0.241 131  -0.181  1.2594  2.239
## child hard - adult soft  -0.3419 0.241 131  -1.062  0.3782 -1.420
## child hard - child soft   0.0349 0.187 155  -0.522  0.5914  0.187
## adult soft - child soft   0.3768 0.241 131  -0.343  1.0969  1.565
## p.value
## 1.0000
## 1.0000
## 1.0000
## 1.0000
## 1.0000
## 0.0910
## 1.0000
## 0.5449
## 1.0000
## 0.5724
## 1.0000
## 0.4023
## 1.0000
## 1.0000
## 1.0000
## 
## Confidence level used: 0.95
## Conf-level adjustment: bonferroni method for 15 estimates
## P value adjustment: bonferroni method for 15 tests

```

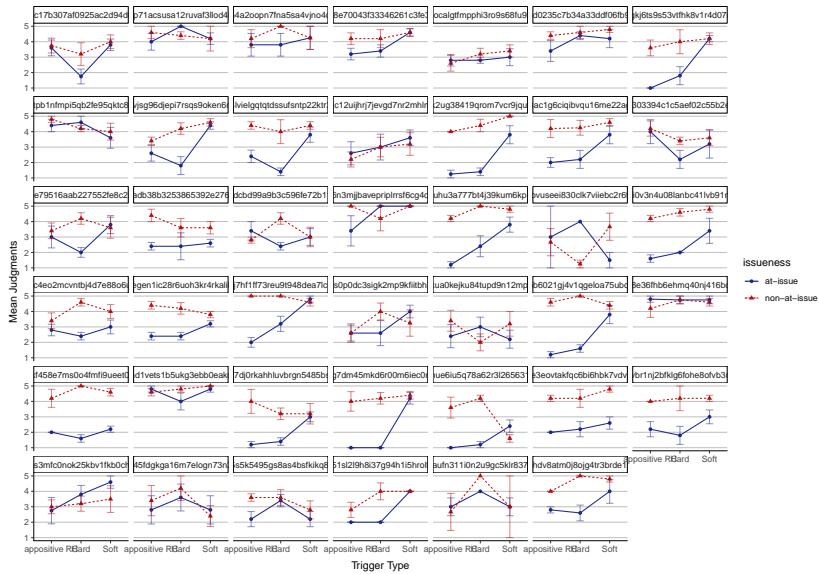
## 9.7 Nochmal Plots

### 9.7.1 by subject

```
subs <- ggplot(d, aes(y = judgment, x = trigger, color = issueness,
  shape = issueness, linetype = issueness),
  group = issueness) +
  stat_summary(fun.y = mean, geom = "line") +
  stat_summary(fun.y = mean, geom = "point") +
  stat_summary(fun.data = mean_se, geom = "errorbar", width = 0.25,
  alpha = .5, linetype = 1) +
  labs(y = "Mean Judgments", x = "Trigger Type") +
  scale_color_manual(values = c("#19278e", "#BC0D0D")) +
  scale_x_discrete(labels = c('appositive RBard', 'Hard', 'Soft')) +
  facet_wrap(~id) + cleanup
```

### 9.7.2 by subject

subs

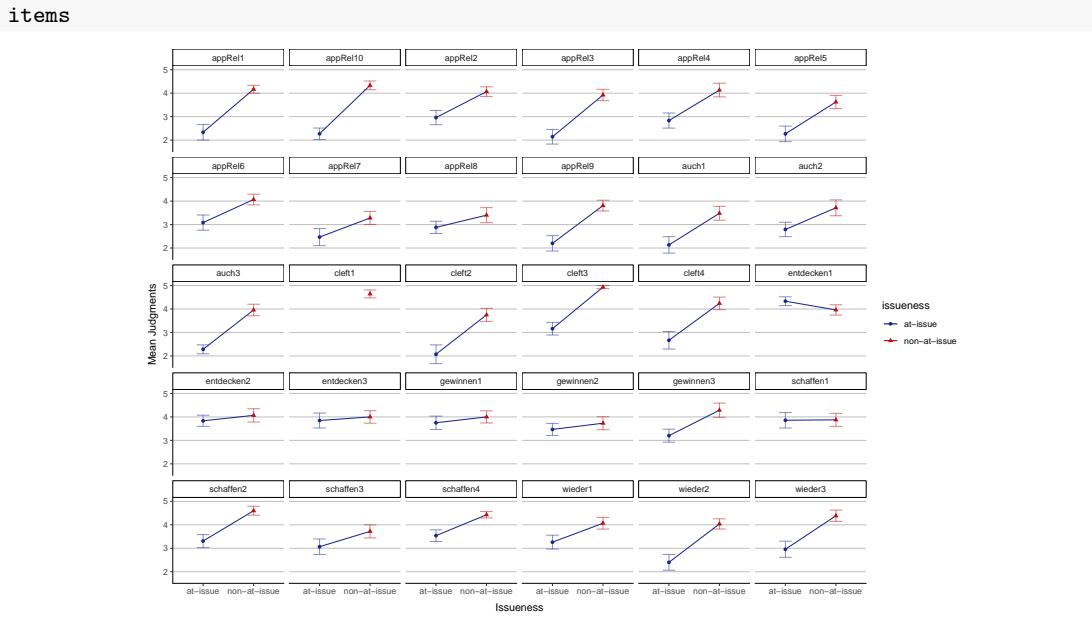


### 9.7.3 by item

```
items <- ggplot(d, aes(y = judgment, x = issueness, color = issueness,
  shape = issueness)) +
  stat_summary(fun.y = mean, geom = "line", aes(group = 1)) +
  stat_summary(fun.y = mean, geom = "point", aes(group = 1)) +
  stat_summary(fun.data = mean_se, geom = "errorbar", width = 0.25,
  alpha = .5, linetype = 1) +
  labs(y = "Mean Judgments", x = "Issueness") +
```

```
scale_color_manual(values = c("#19278e", "#BCDODD")) +
facet_wrap(~itemid) + cleanup
```

#### 9.7.4 by item



#### 9.7.5 Off to the rave

Bis zum nächsten Mal!

# 10

---

## ggplot2

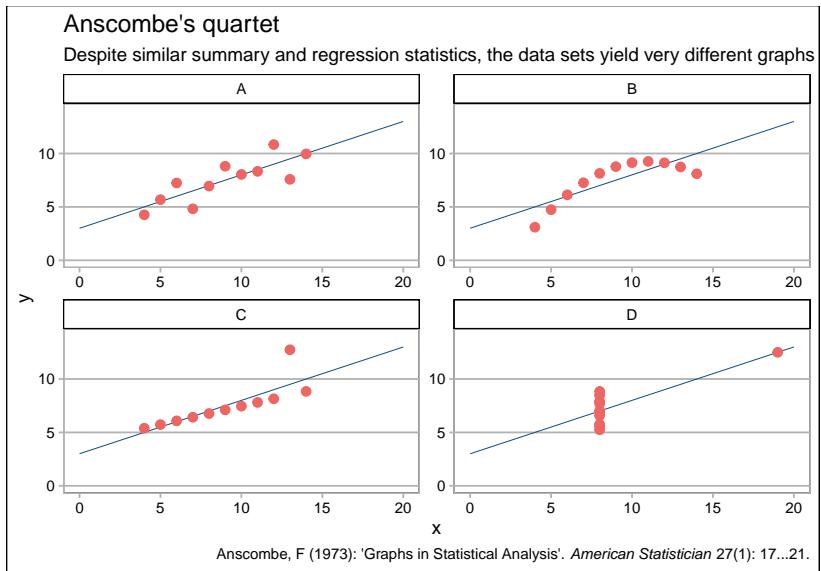
### 10.1 Warum Plots: Anscombes Quartett

#### 10.1.1 Scheinbar gleiche Daten

Diese vier Bedingungen sehen auf den ersten Blick ziemlich gleich aus:

```
## # A tibble: 4 x 9
##   set   mean_x  mean_y   sd_x   sd_y cor_xy intercept slope r_squared
##   <fct>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>      <dbl>  <dbl>      <dbl>
## 1 A       9     7.50   3.32   2.03  0.816      3     0.5     0.667
## 2 B       9     7.50   3.32   2.03  0.816     3.00    0.5     0.666
## 3 C       9     7.5    3.32   2.03  0.816     3.00    0.5     0.666
## 4 D       9     7.50   3.32   2.03  0.817     3.00    0.5     0.667
```

### 10.1.2 Aber!

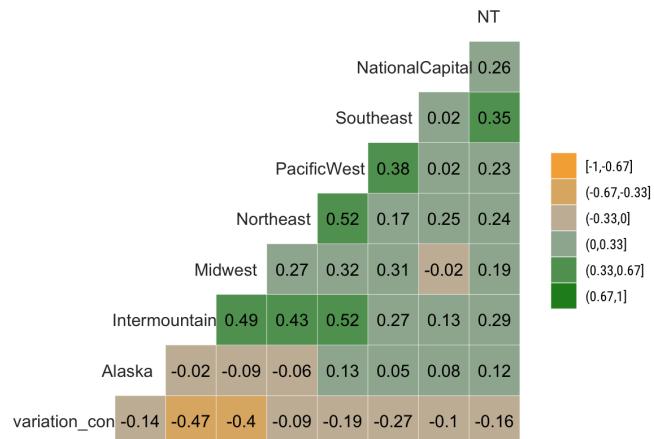


## 10.2 Beispiele

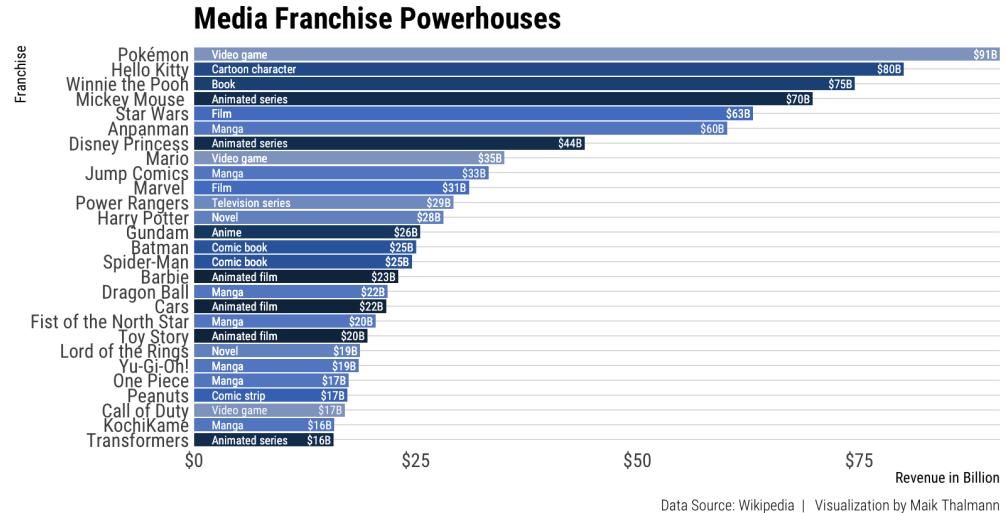
### 10.2.1 Beispiel 1

#### US National Parks

Correlation between annual variation of gas prices and number of visitors park visitors



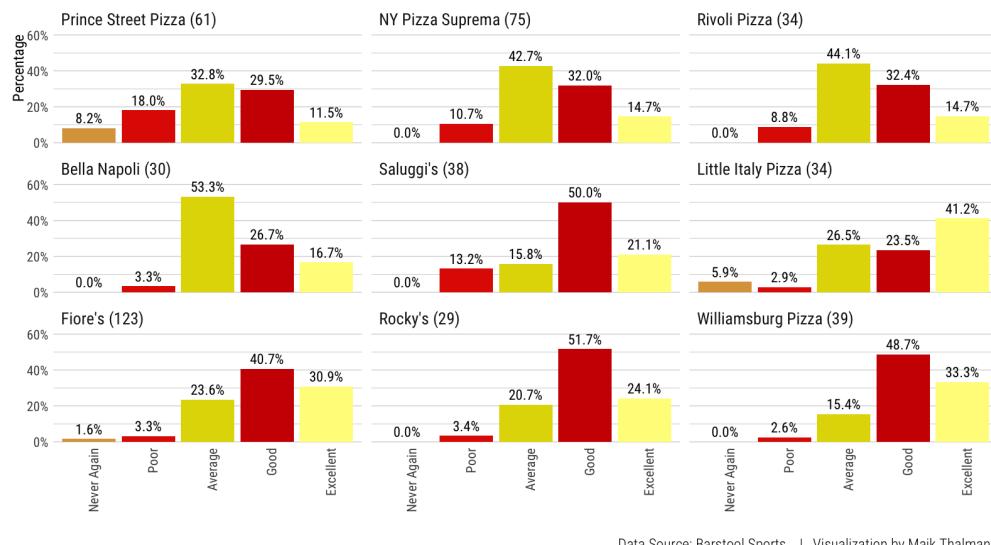
### 10.2.2 Beispiel 2



### 10.2.3 Beispiel 3

#### Pizza Ratings

Shown are the Top 9 Pizza Places With the Most Respondents



Data Source: Barstool Sports | Visualization by Maik Thalmann

## 10.3 Einfache Plots

### 10.3.1 Daten für den Basisplot

Hier schonmal der [Cheat Sheet zu ggplot2](#), eine weitere [Hilfeseite](#) und ein großartiges [Tutorial](#).

Paket laden und Daten präparieren (ich verwende hier “nur” 5000 Beobachtungen, weil es sonst in manchen Plots ziemlich voll wird; das kommt aber ganz auf die Art des Plots an):

```
library(ggplot2)
set.seed(1)
d <- as.data.frame(diamonds)
d <- d[sample(1:nrow(d), 5000), ]
head(d)

##      carat      cut color clarity depth table price     x     y     z
## 24388  0.41 Very Good    D    SI2  62.3    61   638 4.72 4.75 2.95
## 43307  0.50 Very Good    F    VS2  62.8    57  1402 5.05 5.08 3.18
## 4050   1.03     Fair     I    SI2  65.2    56  3530 6.42 6.35 4.16
## 11571  1.10    Ideal     I    SI1  62.1    57  5037 6.60 6.64 4.11
## 25173  1.51 Very Good    E    VS2  63.3    61 13757 7.24 7.17 4.56
## 32618  0.30    Ideal     H    VS2  62.1    55   457 4.30 4.33 2.68
```

Zu den Daten selbst: "A dataset containing the prices and other attributes of almost 54,000 diamonds."

### 10.3.2 ggplot()

Grundgerüst ist der `ggplot`-Befehl mit Datensatz- und `aes`-Angaben: Werte auf der *x*- und *y*-Achse. Dabei sollte die abhängige Variable, hier der Verkaufspreis, die *y*-Achse bilden, während die (erste) unabhängige Variable, hier das Gewicht der Diamanten, auf der *x*-Achse verortet ist.

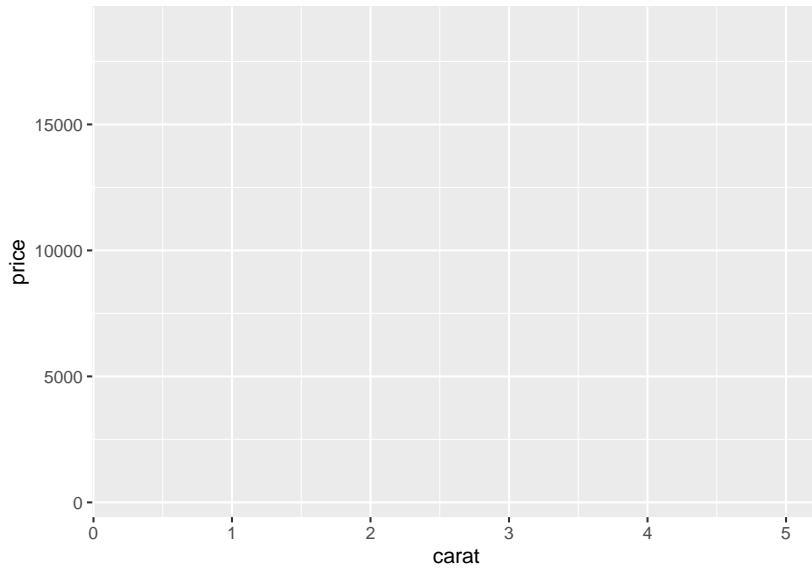
Ich speichere diese Angaben unter dem Namen `p` ab, damit ich später weniger Tipparbeit habe und euch einfach verschiedene Variationen desselben Plots zeigen kann. Nötig ist das aber nicht unbedingt.

```
p <- ggplot(d, aes(x = carat, y = price))
```

### 10.3.3 ggplot()

Lassen wir uns dieses Objekt anzeigen, sehen wir allerdings nur die Achsen.

```
p
```



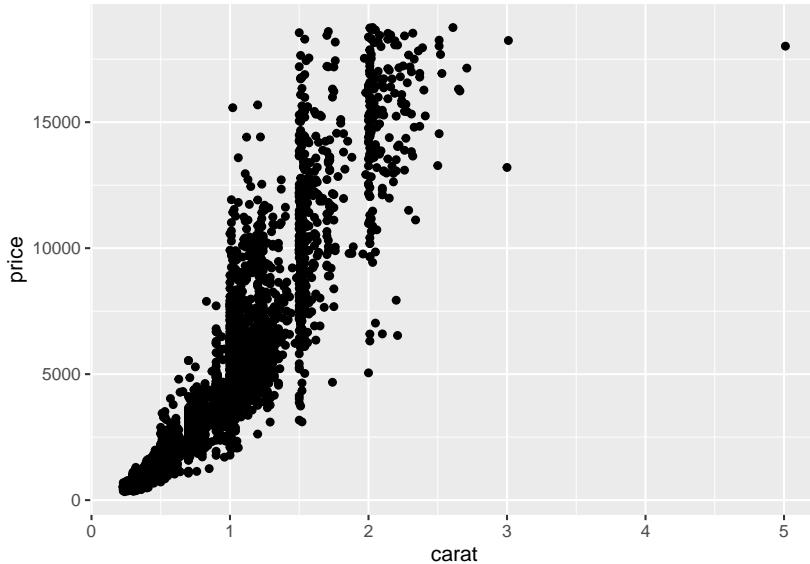
### 10.3.4 Geometrien

Das liegt daran, dass wir `ggplot` noch nicht gesagt haben, was für eine Art Plot wir haben wollen. Dies machen wir mithilfe sogenannter Geometrien, die wir einfach per `+` unserem Objekt hinzufügen können. Gängig sind vor allem Scatterplots (`geom_point`), Linienplots (`geom_line`) und Balkendiagramme (`geom_bar/geom_histogramm`)

### 10.3.5 Geometrien

Hier die Punktewolke (also der Scatterplot):

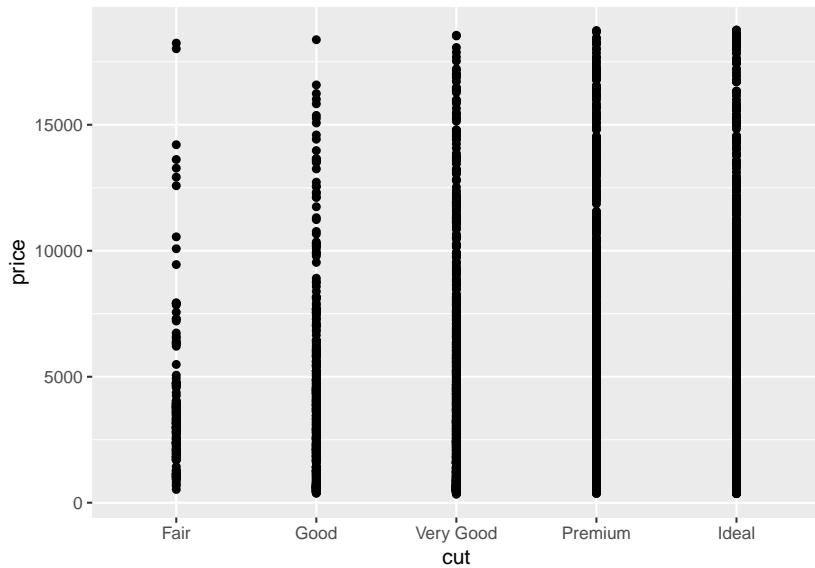
```
p + geom_point()
```



### 10.3.6 Geometrien

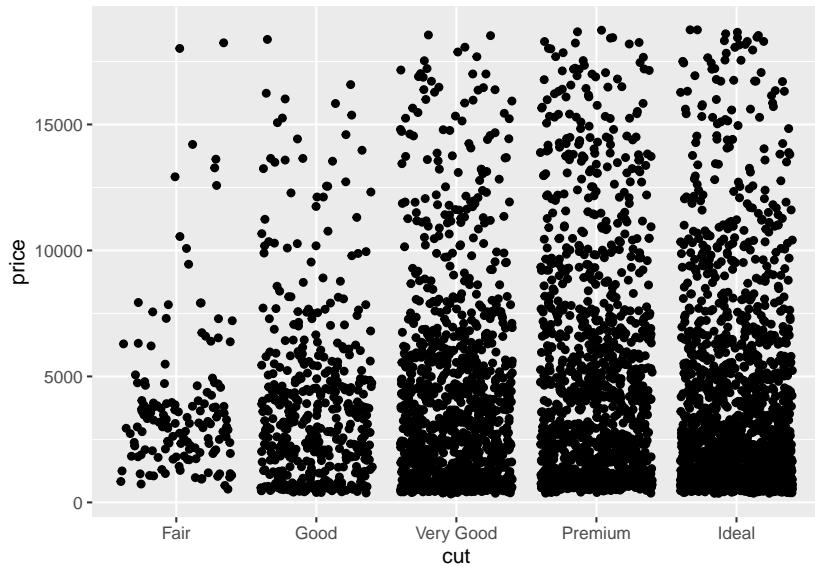
Falls sich Punkte überlagern: `geom_jitter` rüttelt ein wenig an den Punkten, damit sie sich nicht länger überschneiden

```
ggplot(d, aes(x = cut, y = price)) + geom_point()
```



### 10.3.7 Geometrien

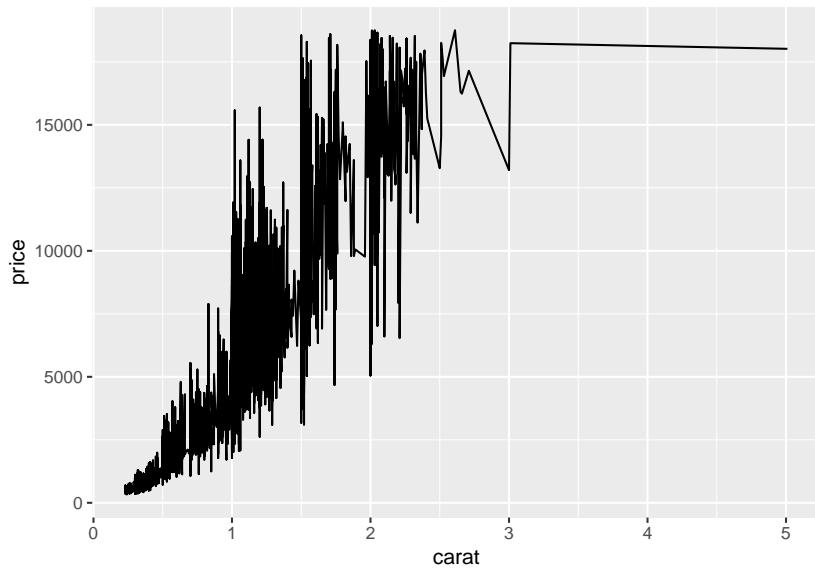
```
ggplot(d, aes(x = cut, y = price)) + geom_jitter()
```



### 10.3.8 Geometrien

Hier ein Linienplot:

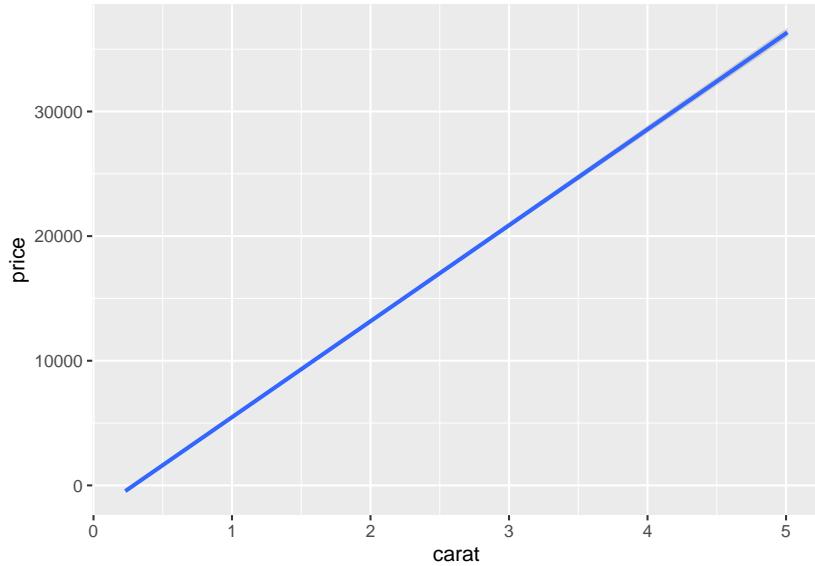
```
p + geom_line()
```



### 10.3.9 Geometrien

Hier eine Trendlinie (LM bzw Korrelation):

```
p + geom_smooth(method = "lm")
```

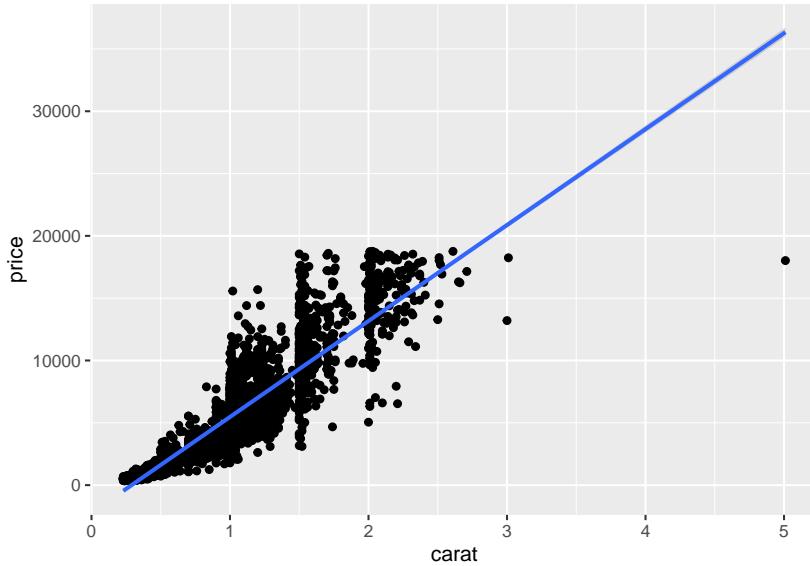


128 10 ggplot2

### 10.3.10 Geometrien

Linie + Punkte

```
p + geom_point() + geom_smooth(method = "lm")
```



### 10.3.11 Mehr Spalten

Was ist, wenn ihr mehr als nur zwei Spalten unterbringen will? → `aes` erweitern.

Weitere Möglichkeiten:

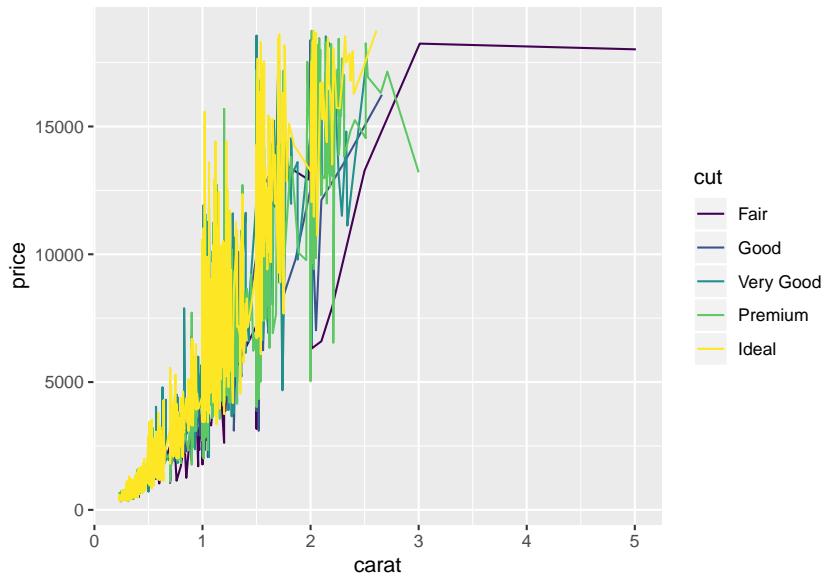
- `fill`
- `linetype` (kurz: `lty`)
- `size`
- `group`
- `alpha`

```
p <- ggplot(d, aes(x = carat, y = price, color = cut))
```

### 10.3.12 Mehr Spalten

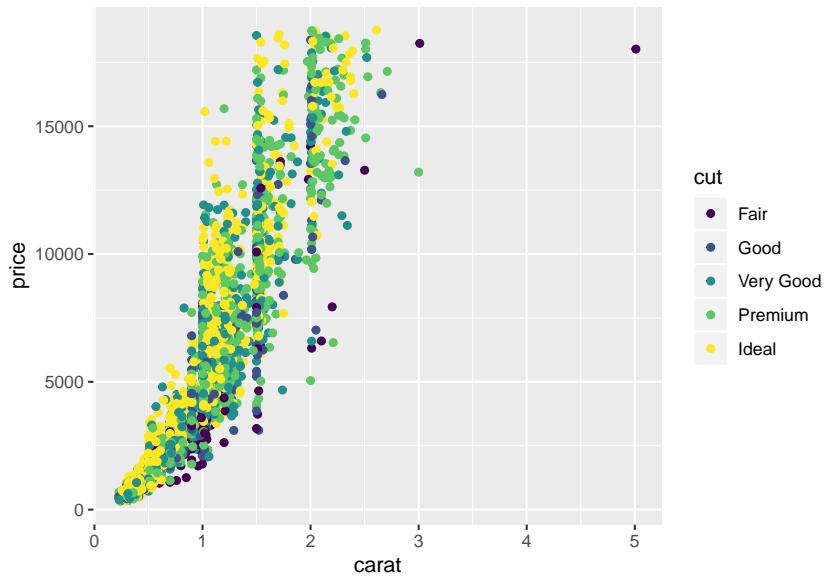
Was ist, wenn ihr mehr als nur zwei Spalten unterbringen will? → `aes` erweitern:

```
p + geom_line()
```



### 10.3.13 Mehr Spalten

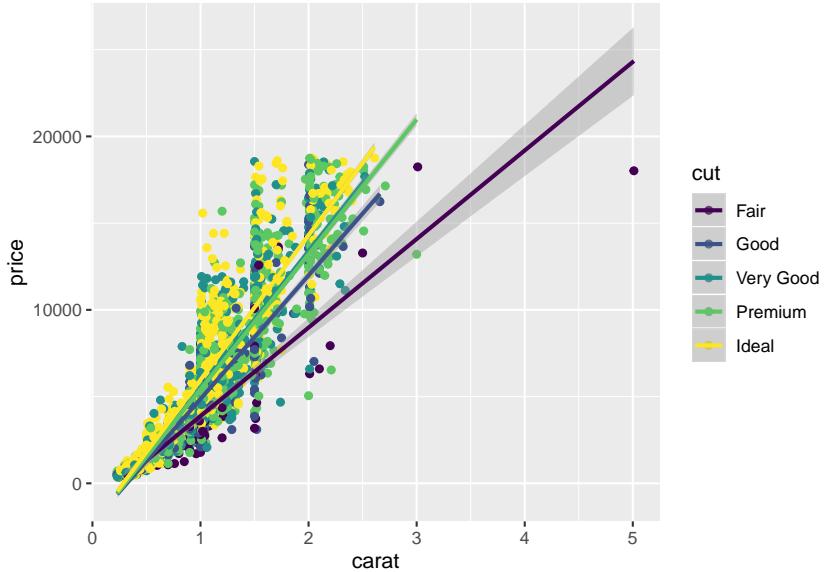
```
p + geom_point()
```



130 10 ggplot2

### 10.3.14 Mehr Spalten

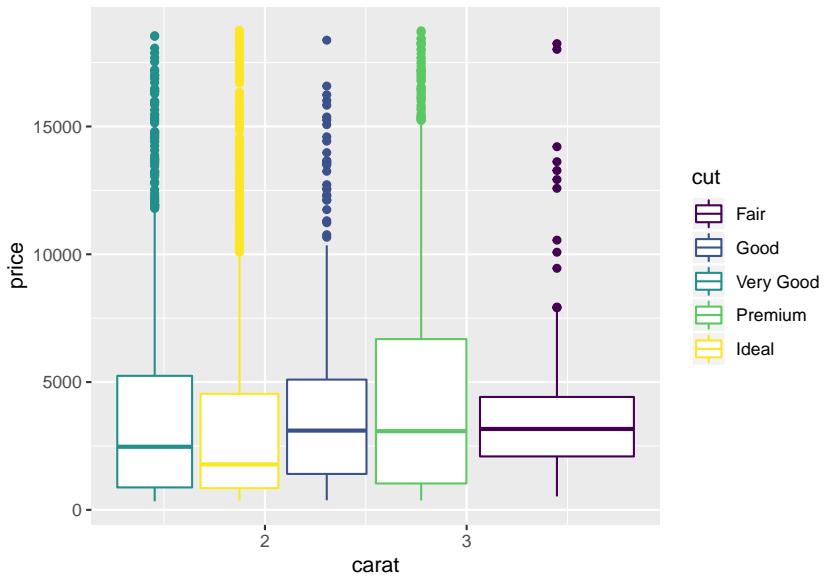
```
p + geom_point() + geom_smooth(method = "lm")
```



### 10.3.15 Mehr Spalten

Oder als Boxplot:

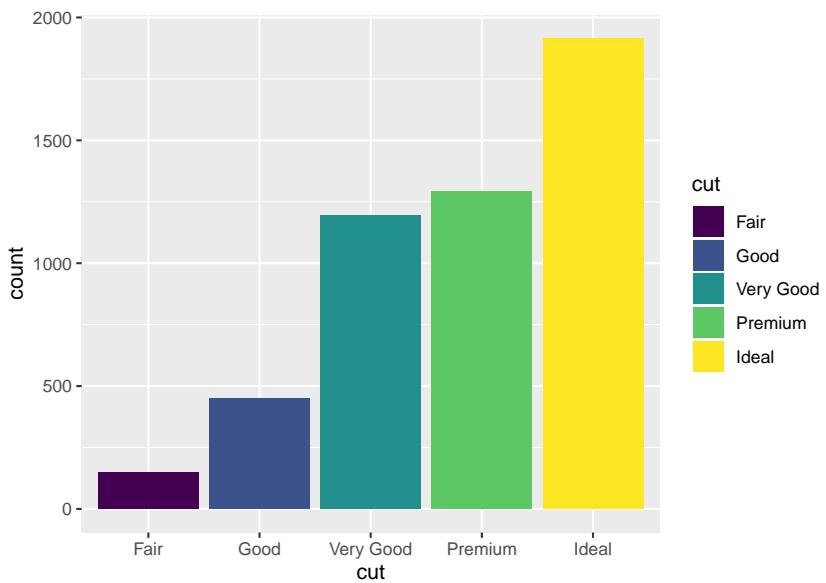
```
p + geom_boxplot()
```



### 10.3.16 Häufigkeiten

`fill` statt `color` weil wir nicht bloß die Umrandung der Balken einfärben wollen

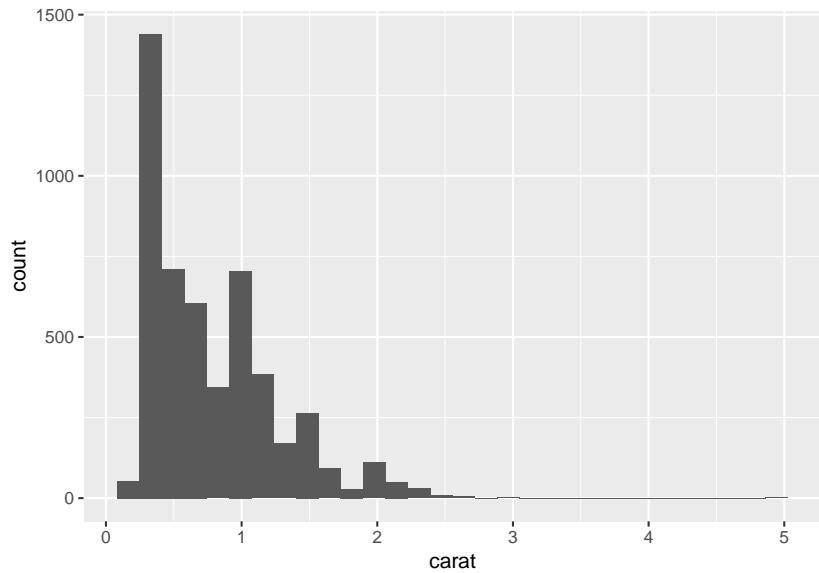
```
d$cut <- factor(d$cut)
ggplot(d, aes(x = cut, fill = cut)) + geom_bar()
```



### 10.3.17 Häufigkeiten

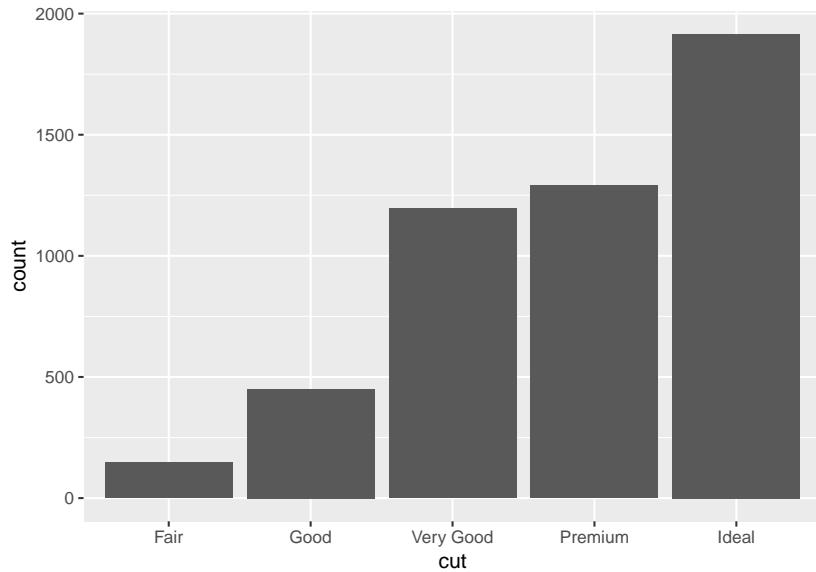
Zum Unterschied zwischen Histogrammen und Barplots: Ohne weitere Argumente sind Barplots für diskrete Daten und Histogramme für kontinuierliche Daten ausgelegt (ich zeige euch aber gleich, dass man es auch anders machen kann)

```
ggplot(d, aes(x = carat)) + geom_histogram()
```



### 10.3.18 Häufigkeiten diskret

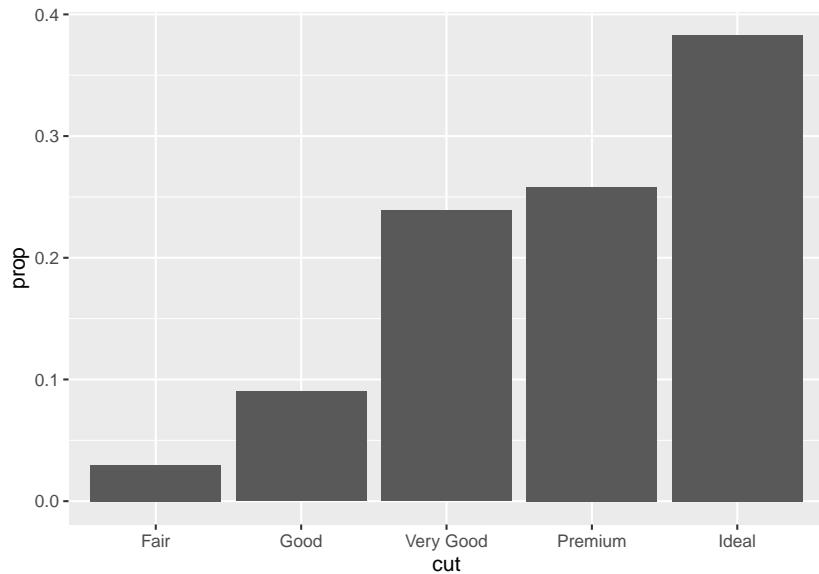
```
ggplot(d, aes(x = cut)) +  
  geom_histogram(stat = "count")
```



### 10.3.19 Prozente

`group = 1` damit Prozente nicht innerhalb der Schliffkategorien berechnet werden (dann hätten wir überall 100%!)

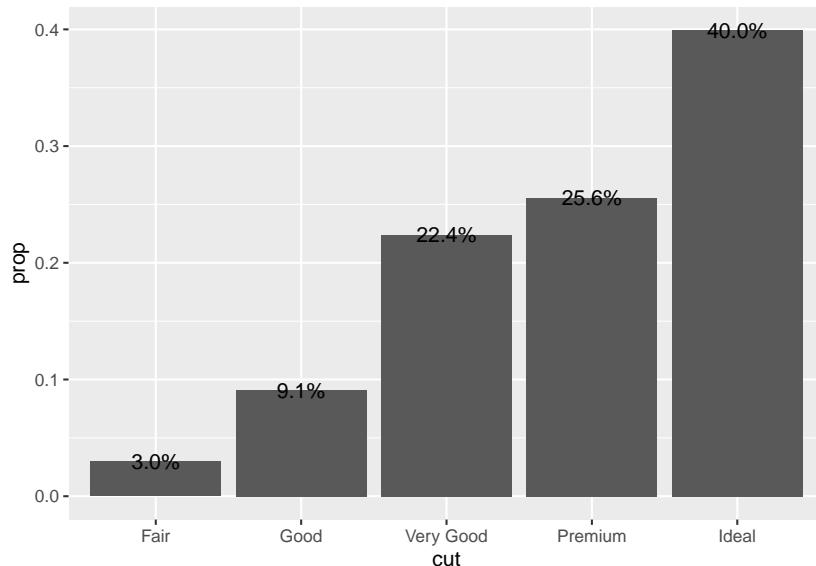
```
ggplot(d, aes(x = cut, y = ..prop.., group = 1)) +
  geom_histogram(stat = "count")
```



### 10.3.20 Prozente mit Label

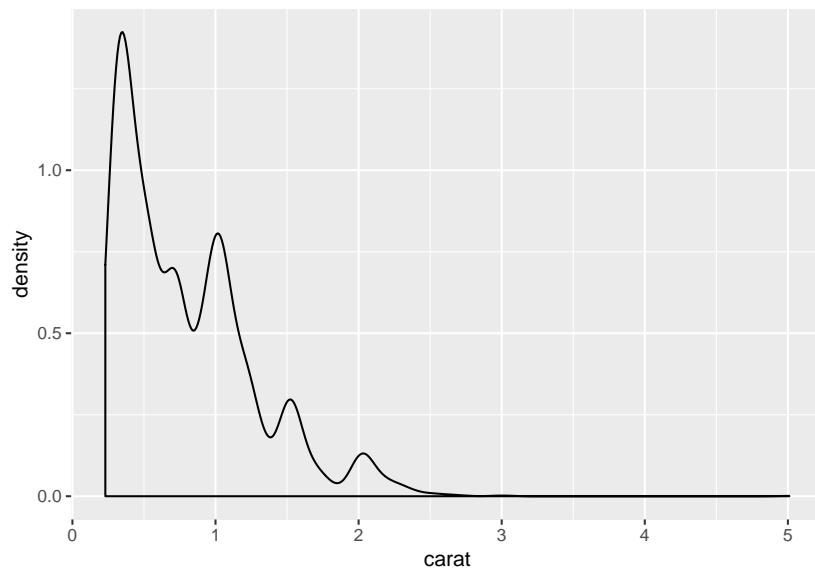
`label` ändert die Beschriftung

```
ggplot(diamonds, aes(x = cut, y = ..prop.., group = 1)) +
  geom_histogram(stat = "count") +
  geom_text(aes(label = scales::percent(..prop..)),
            stat = "count")
```



### 10.3.21 Häufigkeiten

```
ggplot(d, aes(x = carat)) + geom_density()
```



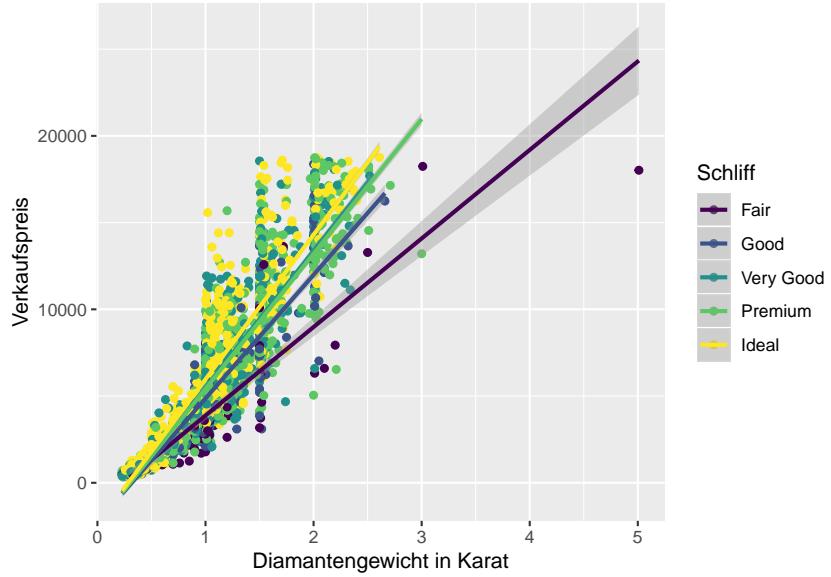
## 10.4 Plots erweitern

### 10.4.1 Beschriftungen

```
p <- ggplot(d, aes(x = carat, y = price, color = cut)) +  
  geom_point() + geom_smooth(method = "lm")
```

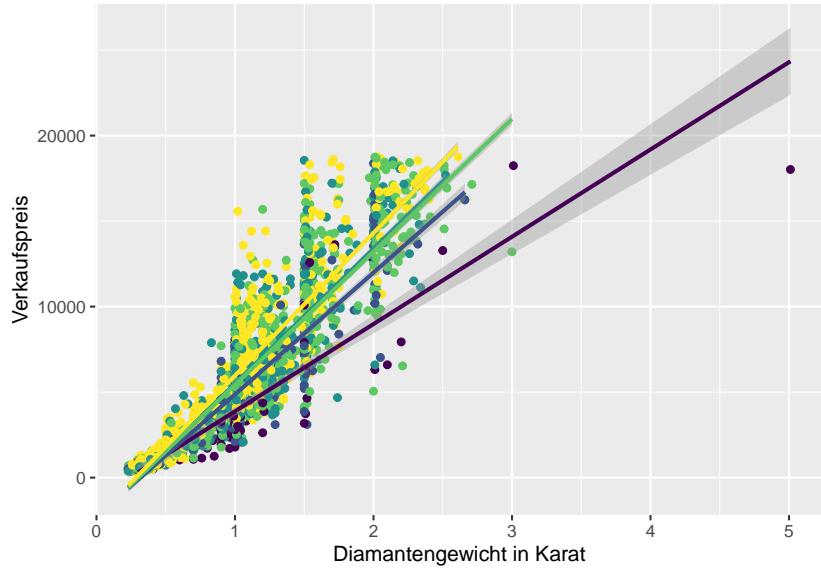
### 10.4.2 Beschriftungen

```
p + labs(x = "Diamantengewicht in Karat",  
         y = "Verkaufspreis",  
         color = "Schliff")
```



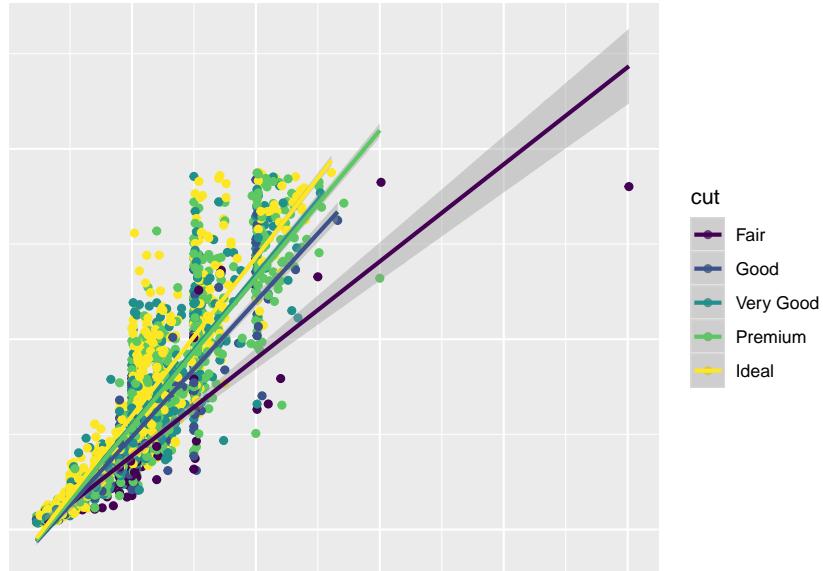
#### 10.4.3 Beschriftungen

```
p + labs(x = "Diamantengewicht in Karat",
          y = "Verkaufspreis") +
  guides(color = F)
```



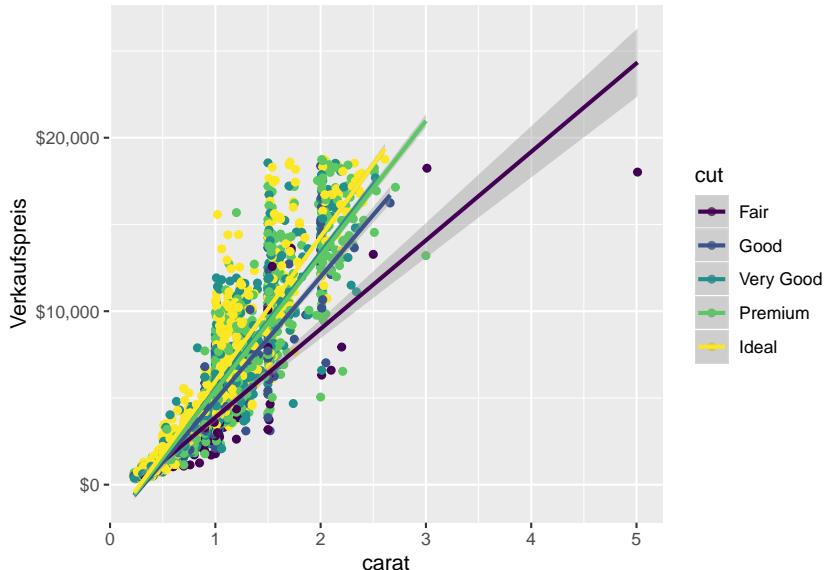
#### 10.4.4 Beschriftungen

```
library(ggpubr)
p + clean_theme()
```



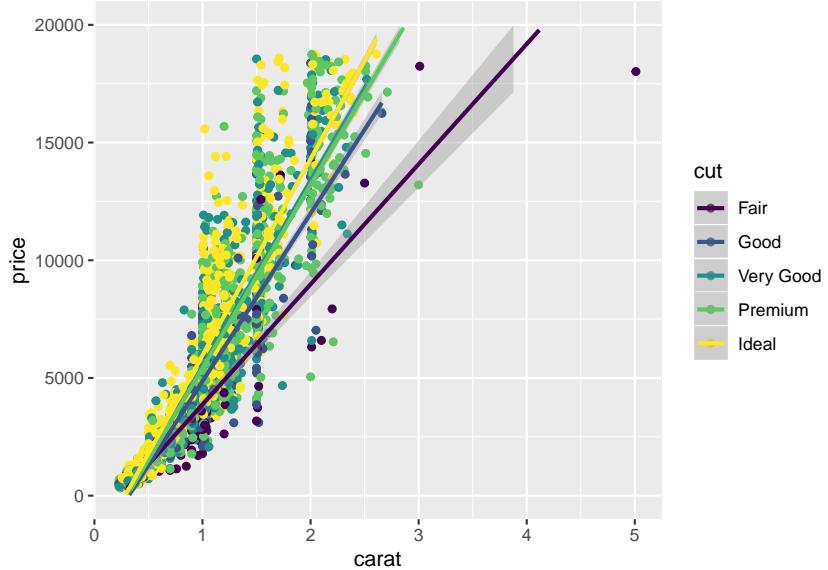
#### 10.4.5 Beschriftungen

```
p + scale_y_continuous(name = "Verkaufspreis", labels = scales::dollar)
```



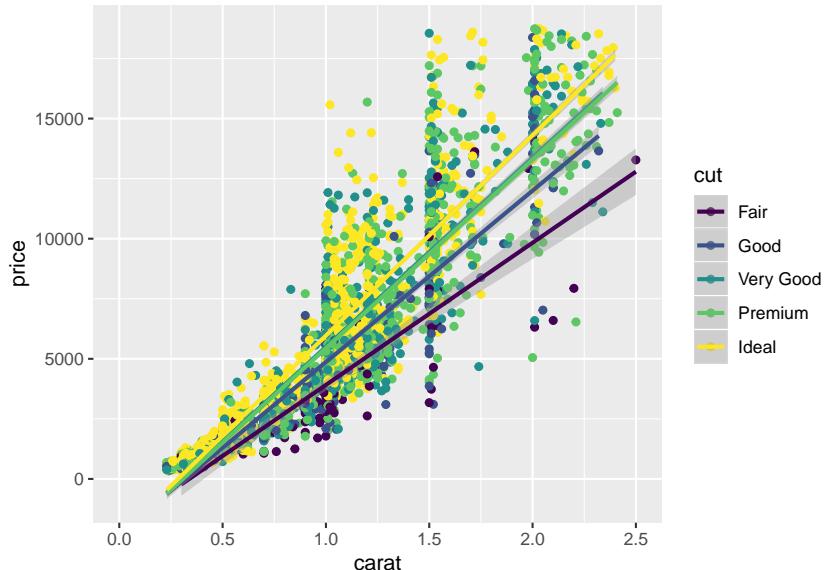
#### 10.4.6 Achsen beschränken

```
p + ylim(0, 20000)
```



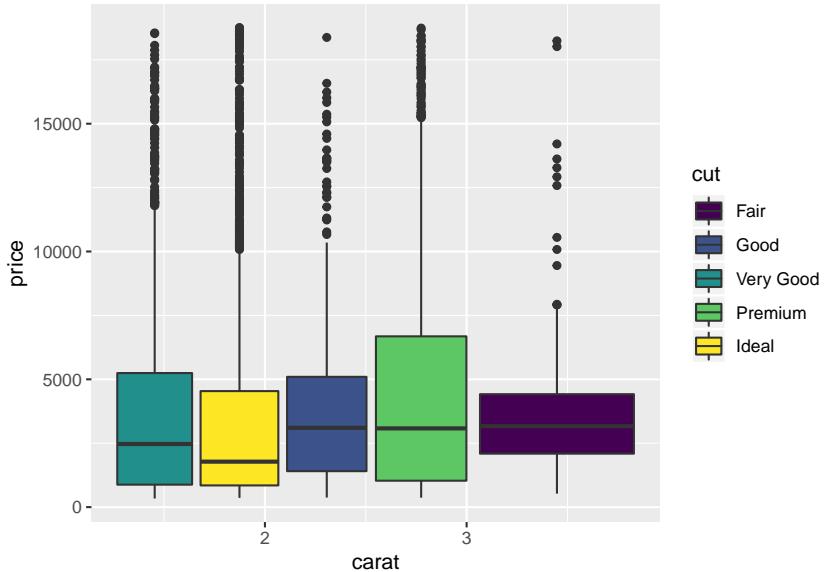
#### 10.4.7 Achsen beschränken

```
p + xlim(0, 2.5)
```



#### 10.4.8 Achsen beschränken

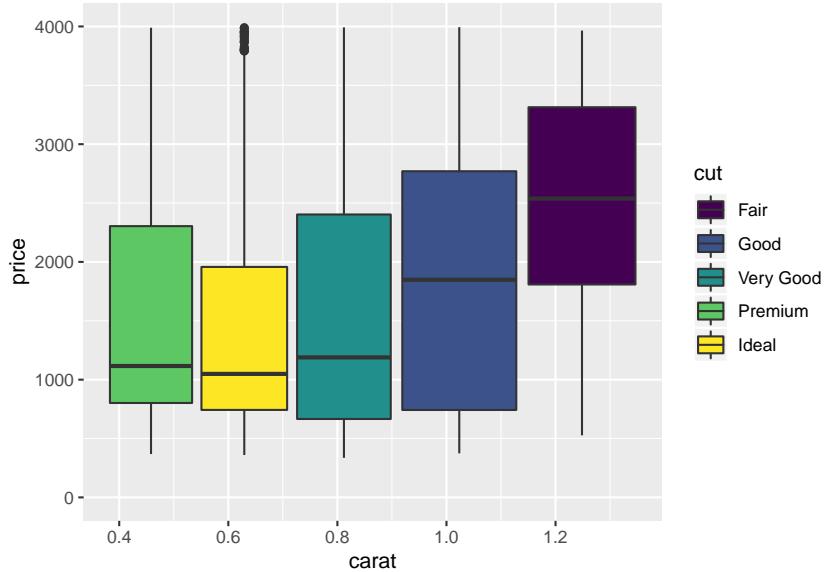
```
p <- ggplot(d, aes(x = carat, y = price, fill = cut)) +  
  geom_boxplot()
```



#### 10.4.9 Achsen beschränken

Merkt ihr was? Die Daten verändern sich!

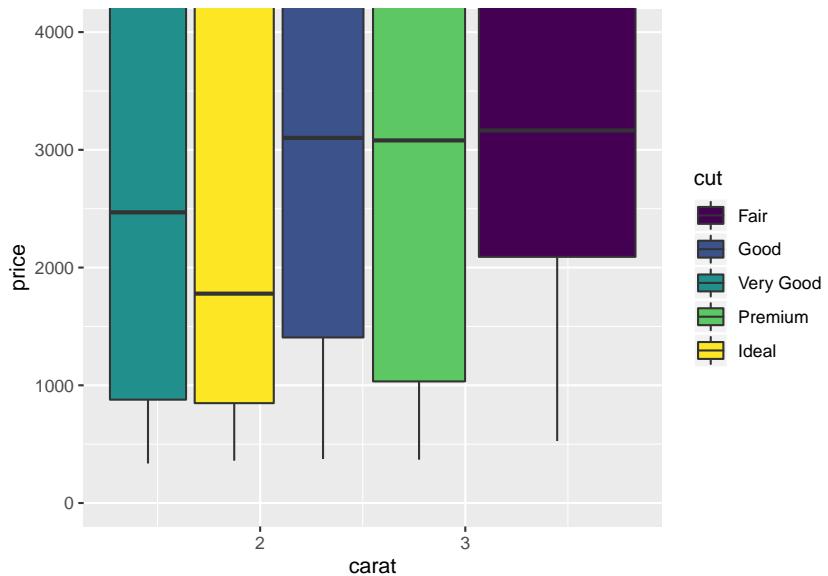
```
p + ylim(1, 4000)
```



#### 10.4.10 Achsen beschränken

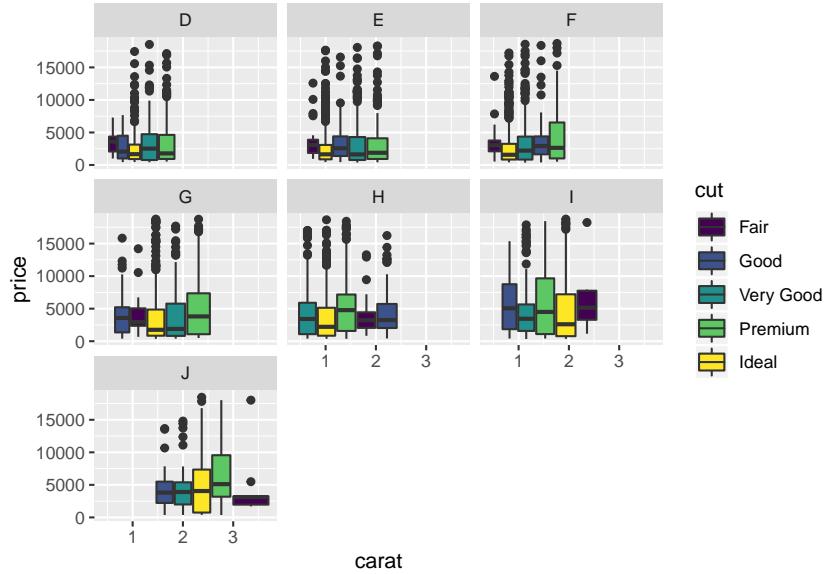
Besser:

```
p + coord_cartesian(ylim = c(1, 4000))
```



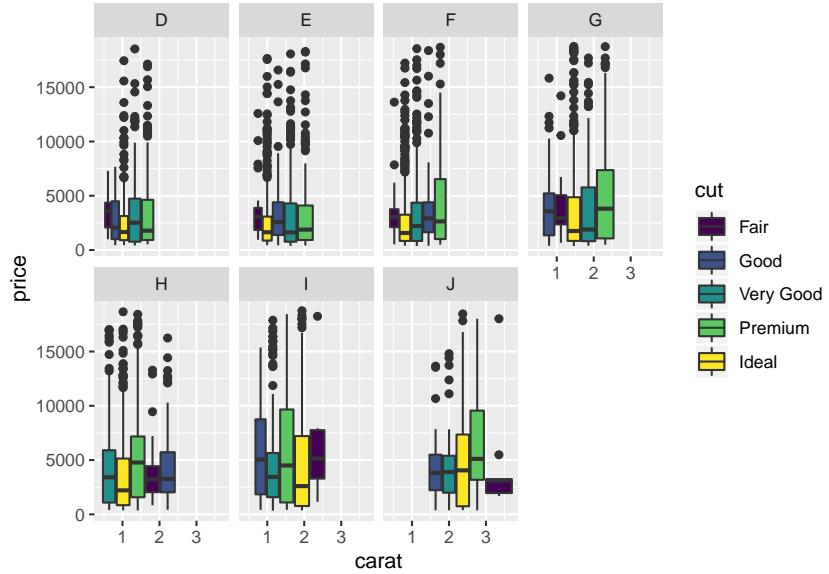
### 10.4.11 Facetten

```
p + facet_wrap(~color)
```



### 10.4.12 Facetten

```
p + facet_wrap(~color, ncol = 4)
```



## 10.5 Plots verschönern

### 10.5.1 Daten für den Basisplot

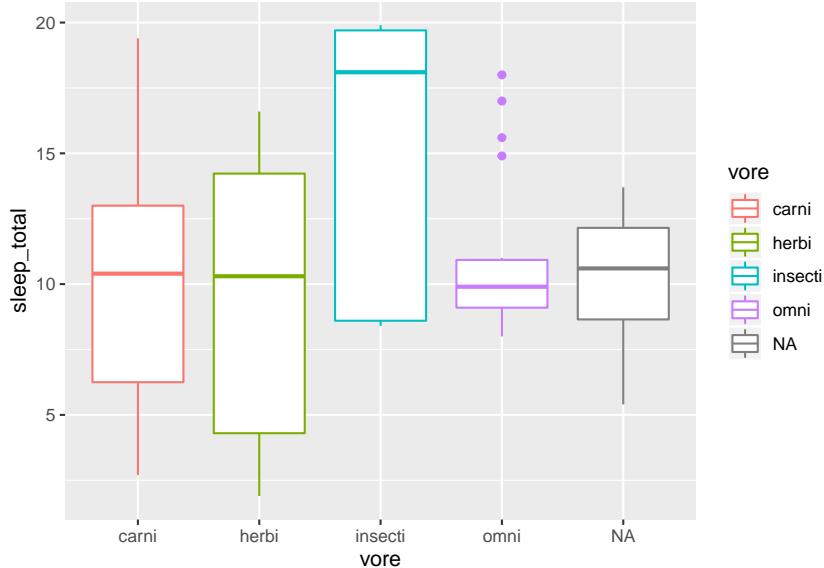
```
library(ggplot2)

d <- as.data.frame(msleep) # from ggplot2 package
head(d)

##           name   genus  vore      order conservation
## 1       Cheetah Acinonyx carni Carnivora          lc
## 2 Owl monkey     Aotus  omni Primates        <NA>
## 3 Mountain beaver Aplodontia herbi Rodentia          nt
## 4 Greater short-tailed shrew    Blarina  omni Soricomorpha          lc
## 5            Cow     Bos herbi Artiodactyla domesticated
## 6 Three-toed sloth Bradypus herbi Pilosa        <NA>
## sleep_total sleep_rem sleep_cycle awake brainwt bodywt
## 1      12.1        NA         NA  11.9      NA 50.000
## 2      17.0        1.8        NA  7.0  0.01550  0.480
## 3      14.4        2.4        NA  9.6      NA  1.350
## 4      14.9        2.3  0.1333333  9.1  0.00029  0.019
## 5       4.0        0.7  0.6666667 20.0  0.42300 600.000
## 6      14.4        2.2  0.7666667  9.6      NA  3.850
```

### 10.5.2 Basisplot

```
p <- ggplot(d, aes(y = sleep_total, x = vore, color = vore)) +
  geom_boxplot()
p
```

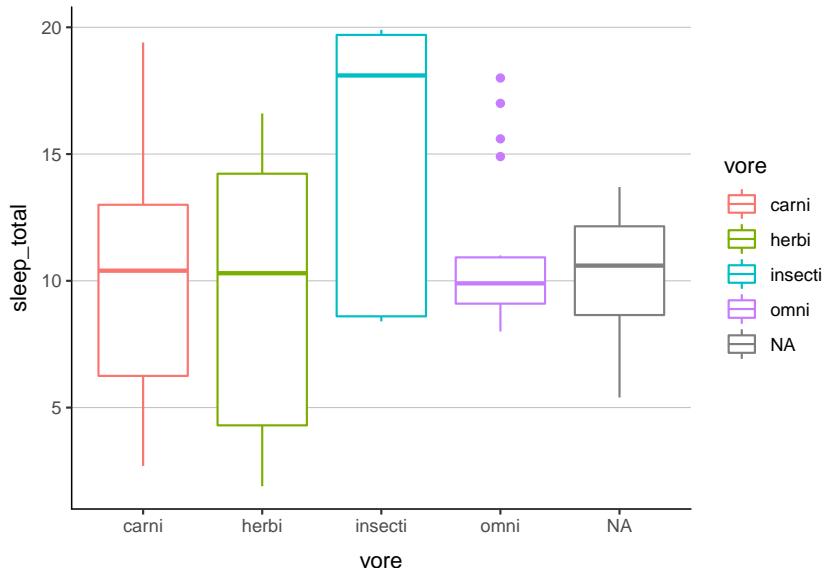


### 10.5.3 Möglichkeit 1

```
cleanup = theme(axis.line.x = element_line(color = "black"),
  axis.line.y = element_line(color = "black"),
  panel.background = element_rect(fill = "transparent", colour = NA),
  panel.grid.major = element_line(size = .2, color = "gray"),
  panel.grid.major.x = element_blank(),
  legend.background = element_rect(fill = "transparent"),
  axis.title.x = element_text(vjust = -1),
  axis.title.y = element_text(vjust = 2),
  plot.margin = unit(c(.5, .5, .5, .5), "cm"),
  strip.background = element_rect(colour = "black", fill = "transparent"),
  legend.key = element_blank()
)
```

### 10.5.4 cleanup in action

p + cleanup



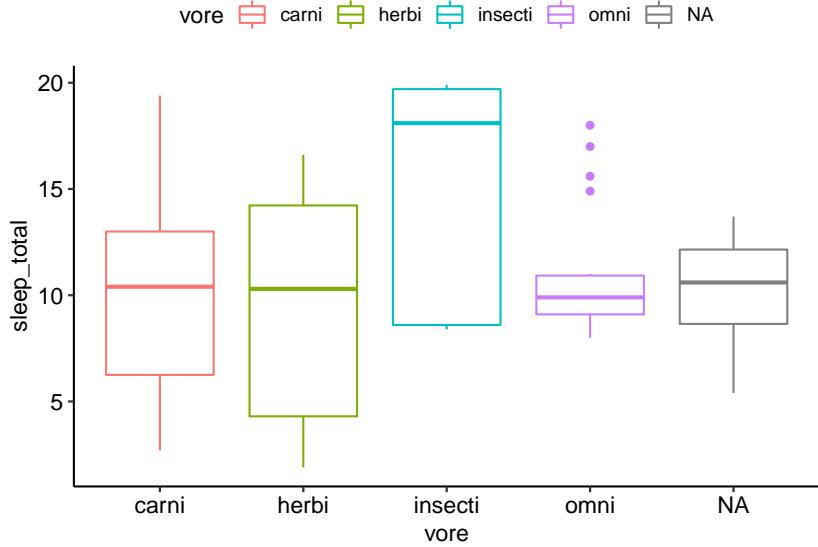
### 10.5.5 Möglichkeit 2

```
install.packages("ggpubr")
library(ggpubr)
+theme_pubr() + clean_theme()
```

### 10.5.6 theme\_pubr()

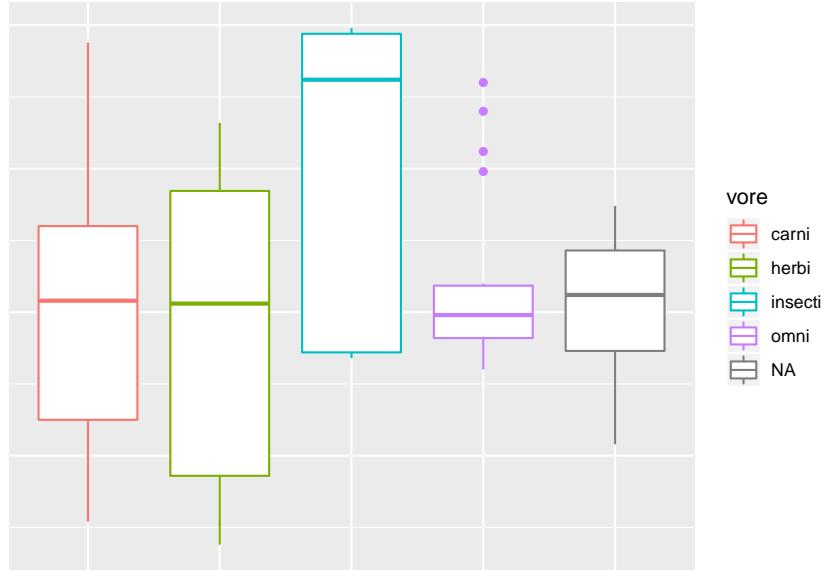
144 10 ggplot2

```
p + theme_pubr()
```



#### 10.5.7 clean\_theme()

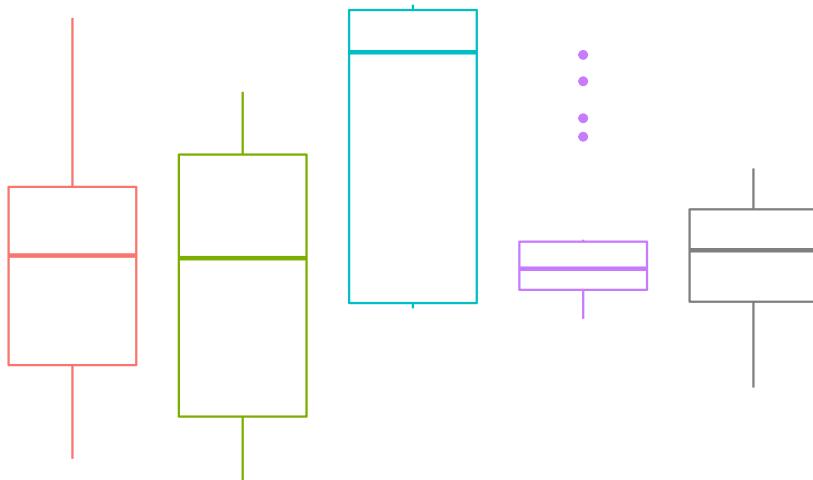
```
p + clean_theme()
```



### 10.5.8 theme\_pubr() + clean\_theme()

```
p + theme_pubr() + clean_theme()
```

vore    carni    herbi    insecti    omni    NA



### 10.5.9 ggthemes

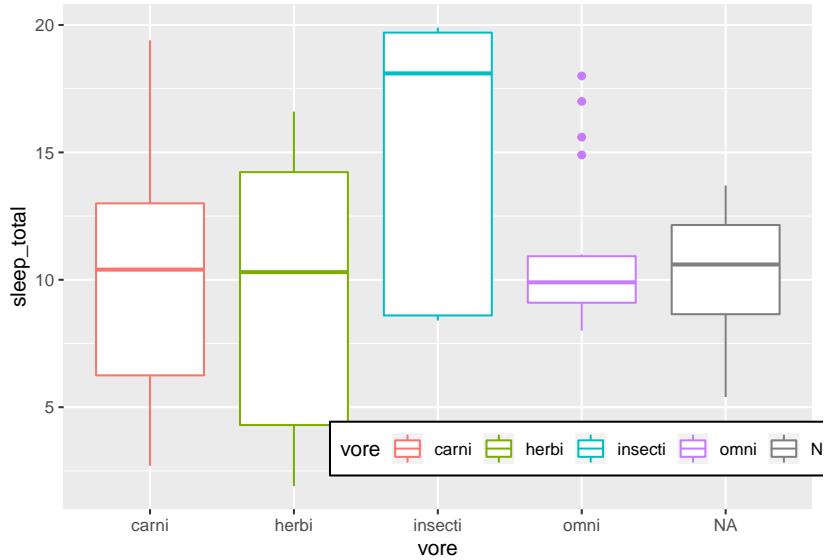
```
library(ggthemes)
+theme_base() # looks like basic R plot
+theme_tufte()
+theme_minimal()
+theme_gray()
+theme_few()
+theme_wsj()
+theme_economist()
+theme_fivethirtyeight()
+theme_stata()
+theme_hc()
+theme_solarized()
+theme_pander()
+theme_void()
+theme_linedraw()
# to change the font or its size in a theme, use
+theme_bw(base_family = "Times", base_size = "10")
```

### 10.5.10 Schriftarten

```
+theme_bw(base_family = "Times", base_size = "10")
"Courier"
"Times"
"Palatino"
"AvantGarde"
"Helvetica"
```

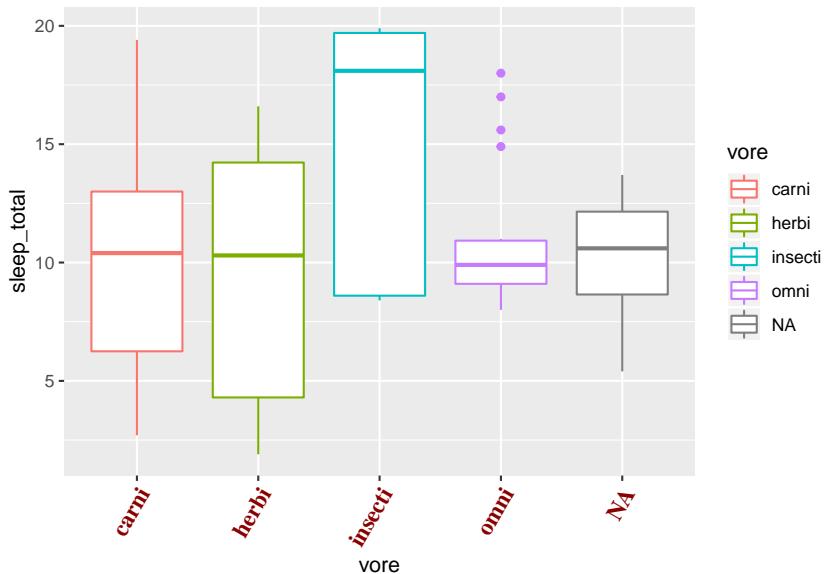
### 10.5.11 Legende verändern

```
p + theme(legend.position = c(0.7, 0.12),
  legend.background = element_rect(linetype = "solid", color = "black"),
  legend.direction = "horizontal")
```



### 10.5.12 Achsenbeschriftung

```
p + theme(axis.text.x = element_text(angle = 60, hjust = 1,
  colour = "darkred", size = 12,
  family = "serif", face = "bold"))
```



### 10.5.13 Farbreferenz

Mögliche Farben: <http://sape.inf.usi.ch/quick-reference/ggplot2/colour>

Vordefinierte Farbpaletten: [https://github.com/EmilHvitfeldt/r-color-palettes/blob/master/R\\_EADME.md](https://github.com/EmilHvitfeldt/r-color-palettes/blob/master/R_EADME.md)

Mein Favorit: <https://www.data-imaginist.com/2018/scico-and-the-colour-conundrum/>

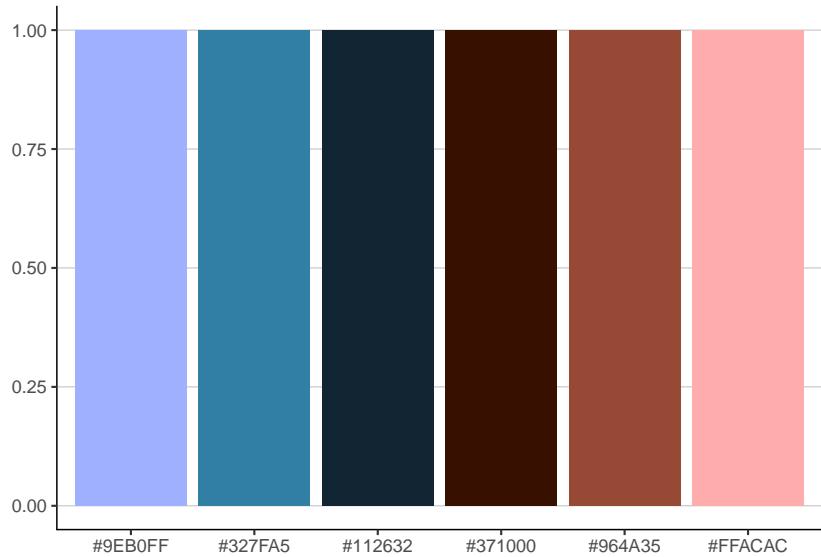
### 10.5.14 scico

```
# install.packages("scico")
library(scico)
scico_palette_show()
```



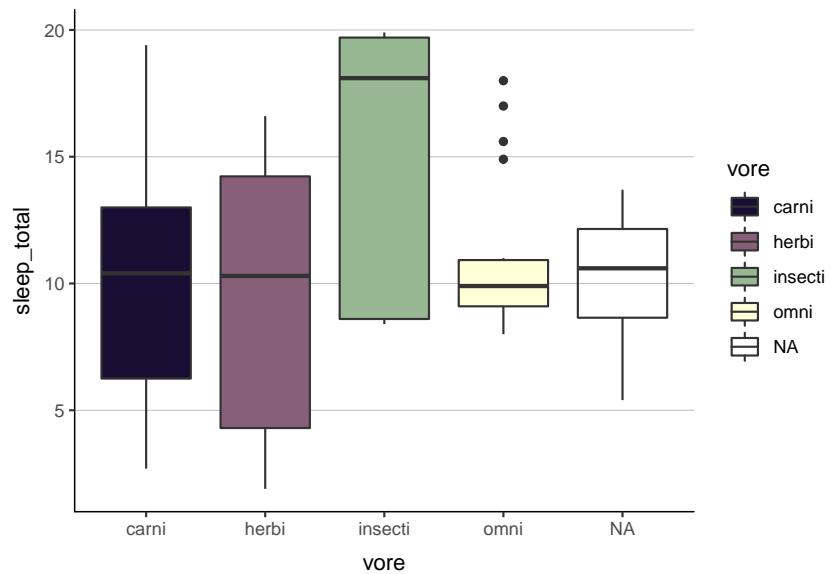
### 10.5.15 Palettenvorschau

```
# palette erstellen (6 farben aus palette "berlin")
COL <- scico(6, palette = "berlin")
# und plotten
ggplot() + aes(x = factor(COL, levels = COL), fill = factor(COL, levels = COL)) +
  geom_bar() +
  scale_fill_manual(values = COL) +
  guides(fill = F) + labs(x = NULL, y = NULL) +
  cleanup
```



### 10.5.16 scico in action

```
ggplot(d, aes(y = sleep_total, x = vore, fill = vore)) +
  geom_boxplot() +
  scale_fill_scico_d(palette = "tokyo") +
  cleanup
```



```
150      10 ggplot2
```

## 10.6 Plot speichern

### 10.6.1 Plots speichern

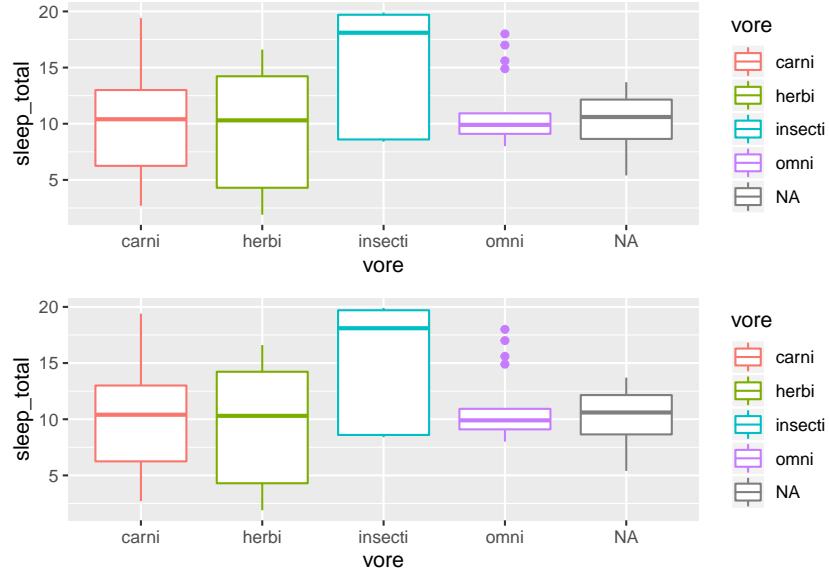
Ohne Angabe des zu speichernden Objekts speichert `ggsave` den zuletzt angezeigten Plot Ohne "path" wird die Datei Arbeitsverzeichnis (`getwd`) gespeichert

```
ggsave("boxplot.pdf", device = "pdf", width = 8, height = 5,  
       units = "cm", path = getwd())
```

## 10.7 Plots verbinden

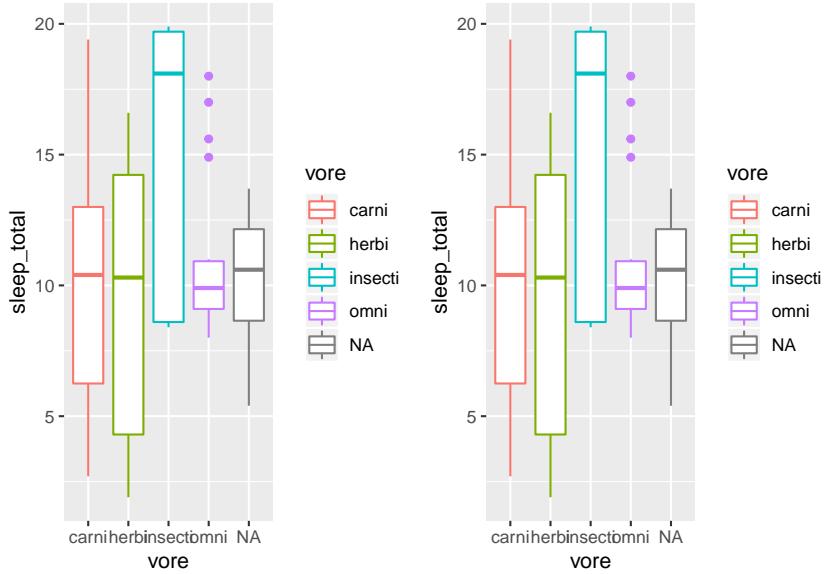
### 10.7.1 Mehrere Plots verbinden

```
library(gridExtra)  
  
grid.arrange(p, p)
```



### 10.7.2 nrow/ncol

```
grid.arrange(p, p, nrow = 1)
```



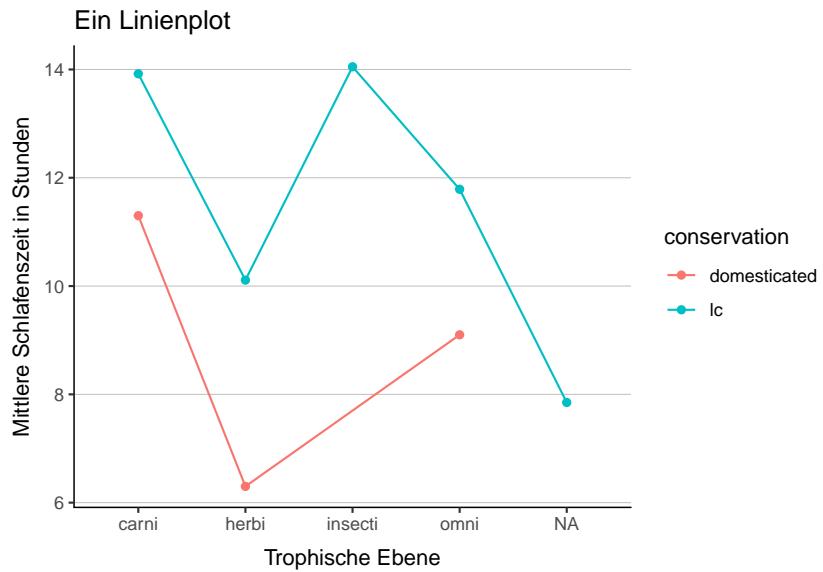
## 10.8 stat\_summary()

### 10.8.1 Line Plot

```
d <- subset(d, d$conservation %in% c("domesticated", "lc"))
p <- ggplot(d, aes(x = vore, y = sleep_total,
  color = conservation, group = conservation)) +
  stat_summary(fun.y = mean, geom = "line") +
  stat_summary(fun.y = mean, geom = "point") +
  labs(x = "Trophische Ebene",
       y = "Mittlere Schlafenszeit in Stunden",
       title = "Ein Linienplot")
```

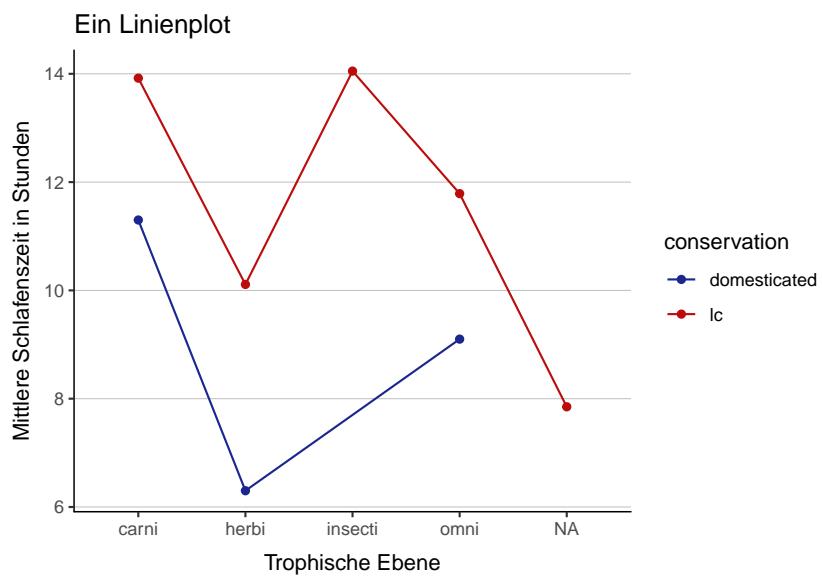
### 10.8.2 Line Plot

```
p + cleanup
```



### 10.8.3 Line Plot

```
p <- p + cleanup + scale_color_manual(values = c("#19278e", "#BC0D0D"))
```

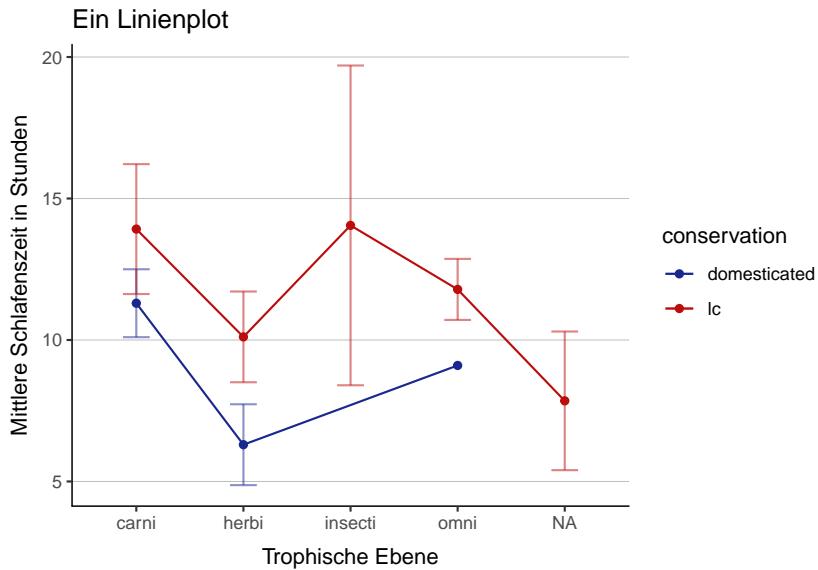


## 10.9 Fehlerbalken

### 10.9.1 Fehlerbalken

Standardfehler

```
p + stat_summary(fun.data = mean_se, geom = "errorbar",
  width = 0.25, alpha = .5, linetype = 1)
```



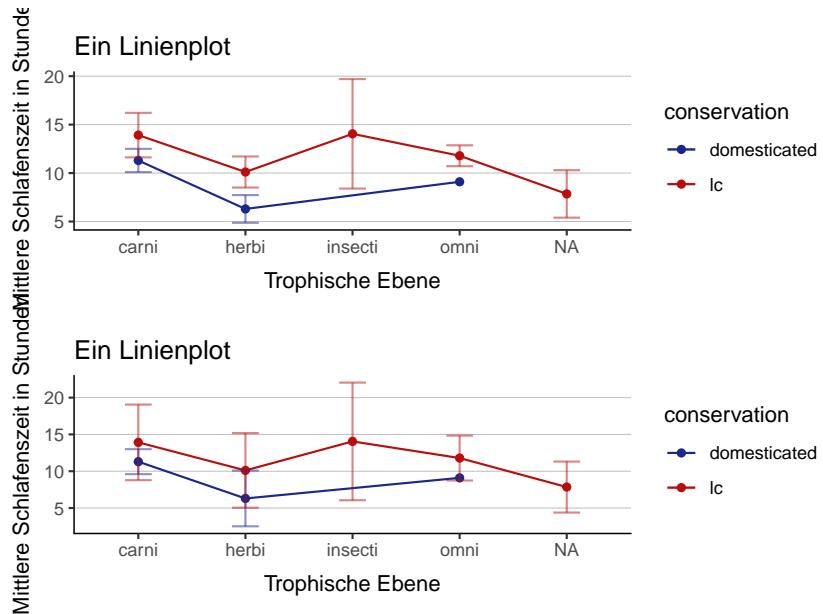
### 10.9.2 Andere Fehlerbalken

Standardfehler vs Standardabweichung

```
p1 <- p + stat_summary(fun.data = mean_se, geom = "errorbar",
  width = 0.25, alpha = .5, linetype = 1)
p2 <- p + stat_summary(fun.data = mean_sdl,
  geom = "errorbar", width = 0.25,
  alpha = .5, linetype = 1)
```

### 10.9.3 Standardfehler vs Standardabweichung

```
grid.arrange(p1, p2, nrow = 2)
```



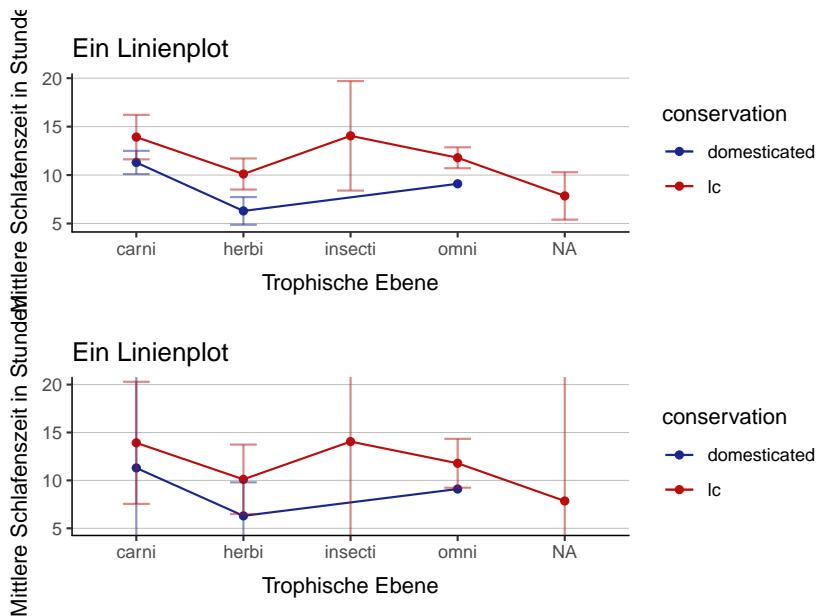
#### 10.9.4 Andere Fehlerbalken

95% Konfidenzintervall

```
p3 <- p + stat_summary(fun.data = mean_cl_normal,
  geom = "errorbar", width = 0.25,
  alpha = .5, linetype = 1) +
  coord_cartesian(ylim = c(5, 20))
```

#### 10.9.5 Standardfehler vs 95% Konfidenzintervall

```
grid.arrange(p1, p3, nrow = 2)
```



## 10.10 Diagnostische Plots

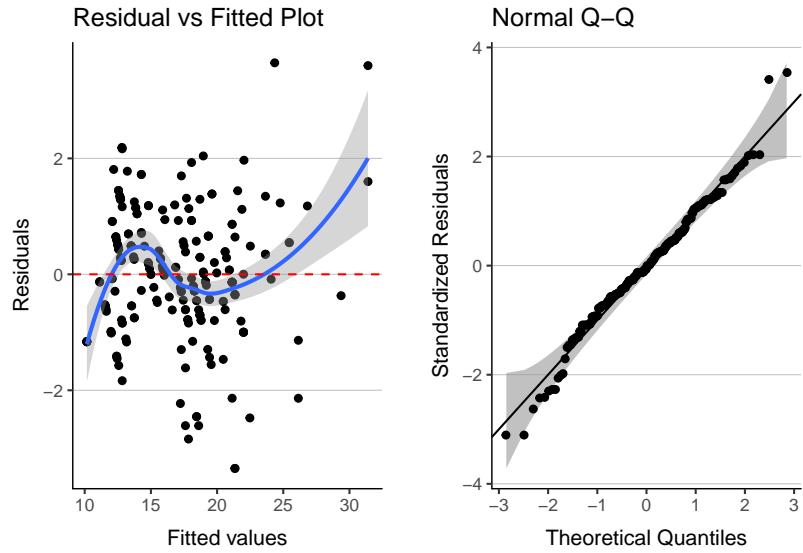
### 10.10.1 Diagnostikfunktion

```
library(qqplotr) # stat_qq_band
diagPlot <- function(model) {
  p1 <- ggplot(model, aes(.fitted, .resid)) + geom_point()
  p1 <- p1 + stat_smooth(method = "loess")
  p1 <- p1 + geom_hline(yintercept = 0, col = "red",
    linetype = "dashed")
  p1 <- p1 + labs(x = "Fitted values", y = "Residuals",
    title = "Residual vs Fitted Plot")
  p2 <- ggplot(model, aes(qqnorm(.stdresid)[[1]], .stdresid))
  p2 <- p2 + stat_qq_band(mapping = aes(sample = qqnorm(.stdresid)[[1]]),
    alpha = .6)
  p2 <- p2 + geom_point(na.rm = TRUE) + geom_abline()
  p2 <- p2 + labs(x = "Theoretical Quantiles",
    y = "Standardized Residuals",
    title = "Normal Q-Q")
  return(list(rvfPlot = p1, qqPlot = p2))
}
```

### 10.10.2 Plots

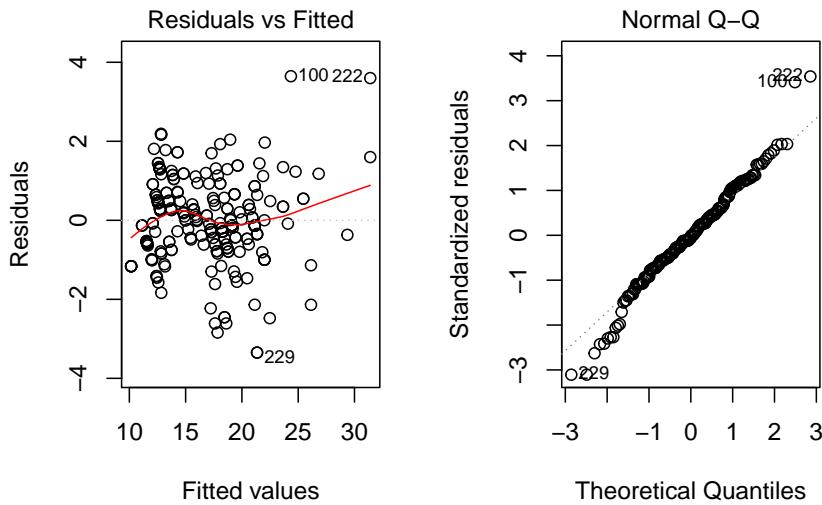
```
diagPlts <- diagPlot(lm(cty ~ displ * hwy, data = mpg))
plot <- grid.arrange(grobs = diagPlts, ncol = 2)
```

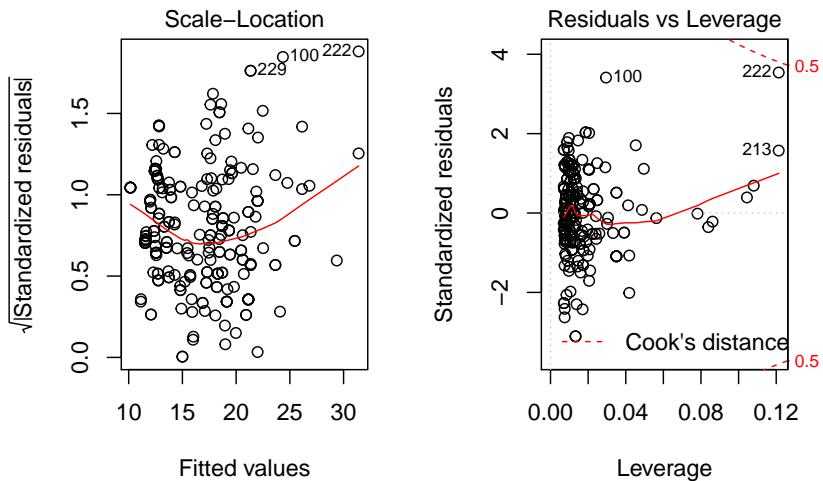
156 10 ggplot2



### 10.10.3 Alternative: Base R

```
par(mfrow = c(1, 2))
plot(lm(cty ~ displ * hwy, data = mpg))
```





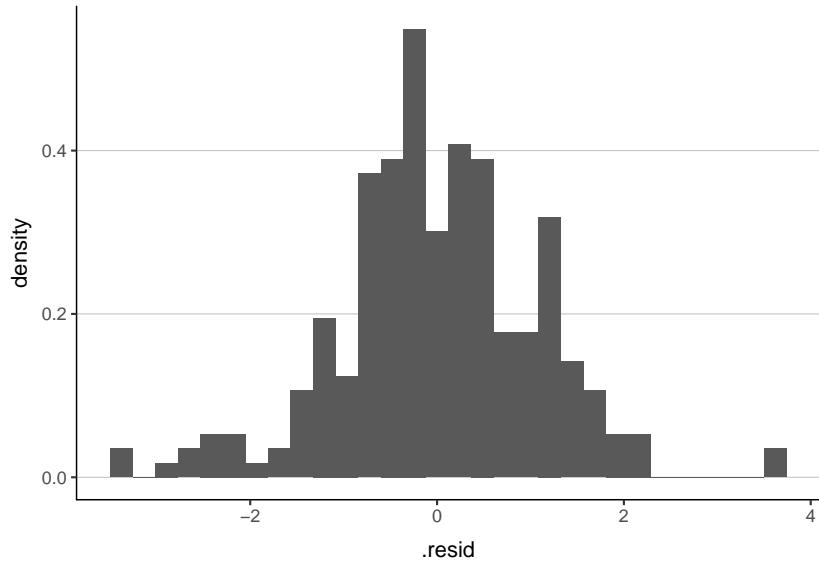
#### 10.10.4 Weitere Alternative

Von mir nicht getestet! Erstellt die bekannten diagnostischen Plots.

```
install.packages("broom")
library(broom)
plot_residuals(lm(cty ~ displ * hwy, data = mpg))
```

#### 10.10.5 Ganz simpel: Residuenhistogramm

```
mod <- lm(cty ~ displ * hwy, data = mpg)
# alternativer plot: hist(resid(mod)); test: shapiro.test(resid(mod))
ggplot(mod, aes(x = .resid)) +
  geom_histogram(aes(y = ..density..)) + cleanup
```



## 10.11 Normalverteilung

### 10.11.1 Normalverteilung: Daten

```
# sicherstellen, dass ihr die gleichen "zufällig"
# erstellten daten bekommt
set.seed(1)

# normalverteilte daten generieren
a <- rnorm(250, mean = 2, sd = 1)
b <- rnorm(250, mean = 4.5, sd = 1)
# datenblatt erstellen
d <- data.frame(a, b)
# daten kombinieren und neues datenblatt
ab <- c(a, b)
d2 <- data.frame(ab)
```

### 10.11.2 shapiro.test

```
shapiro.test(a)

##
##  Shapiro-Wilk normality test
##
## data: a
## W = 0.9964, p-value = 0.8398

shapiro.test(b)

##
```

```
## Shapiro-Wilk normality test
##
## data: b
## W = 0.9968, p-value = 0.897
shapiro.test(ab)

##
## Shapiro-Wilk normality test
##
## data: ab
## W = 0.98647, p-value = 0.0001351
```

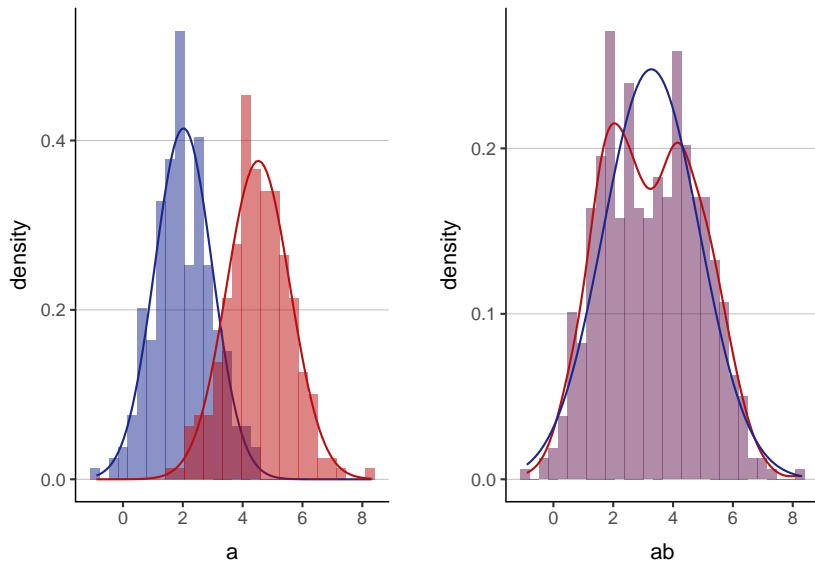
### 10.11.3 Normalverteilung: Daten

```
# beide Verteilungen getrennt
p1 <- ggplot(d) + geom_histogram(mapping = aes(x = a, y = ..density..),
  fill = "#19278e", alpha = .5) +
  stat_function(fun = dnorm,
    args = list(mean = mean(d$a), sd = sd(d$a)),
    color = "#19278e", geom = "line") +
  geom_histogram(mapping = aes(x = b, y = ..density..),
  fill = "#BCODOD", alpha = .5) +
  stat_function(fun = dnorm,
    args = list(mean = mean(d$b), sd = sd(d$b)),
    color = "#BCODOD", geom = "line") +
  cleanup

# beide Verteilungen kombiniert
p2 <- ggplot(d2) + geom_histogram(mapping = aes(x = ab, y = ..density..),
  fill = "#621b54", alpha = .5) +
  stat_density(aes(x = ab, y = ..density..),
    color = "#BCODOD",
    adjust = 1,
    geom = "line") +
  stat_function(fun = dnorm,
    args = list(mean = mean(d2$ab), sd = sd(d2$ab)),
    color = "#19278e", geom = "line") +
  cleanup
```

### 10.11.4 Normalplots

```
grid.arrange(p1, p2, ncol = 2)
```



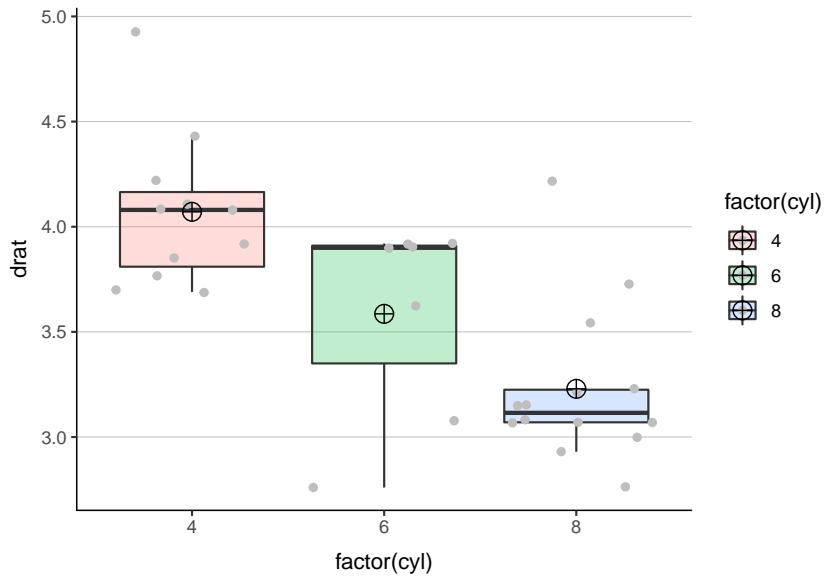
## 10.12 Boxplot verbessern

### 10.12.1 Mittelwerte in Boxplots

```
p <- ggplot(mtcars, aes(x = factor(cyl), y = drat, fill = factor(cyl))) +
  geom_boxplot(alpha = .25, outlier.alpha = 0) +
  geom_jitter(color = "grey") +
  stat_summary(fun.y = mean, geom = "point", shape = 10,
    size = 4, position = position_dodge(0.75)) +
  cleanup
```

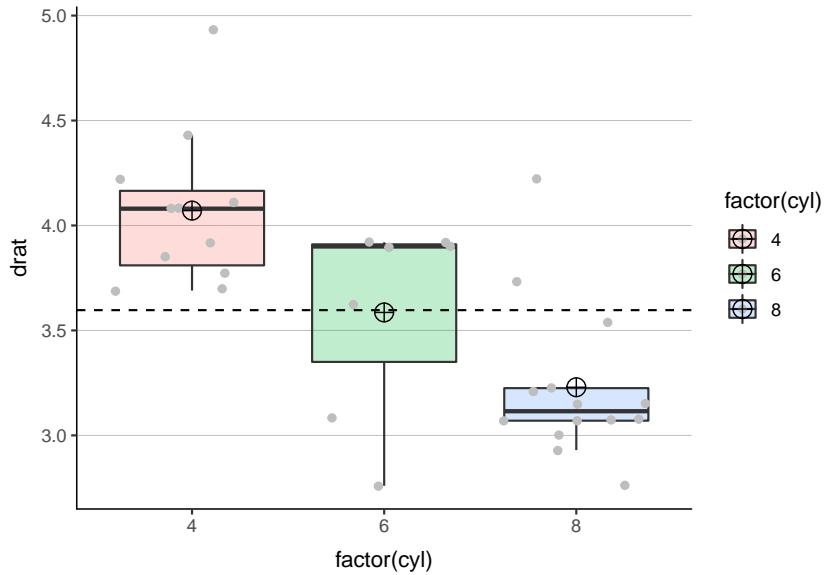
### 10.12.2 Mittelwerte in Boxplots

```
p
```



### 10.12.3 Mittelwerte in Boxplots

```
p + geom_hline(yintercept = mean(mtcars$drat), lty = 2)
```



162 10 ggplot2

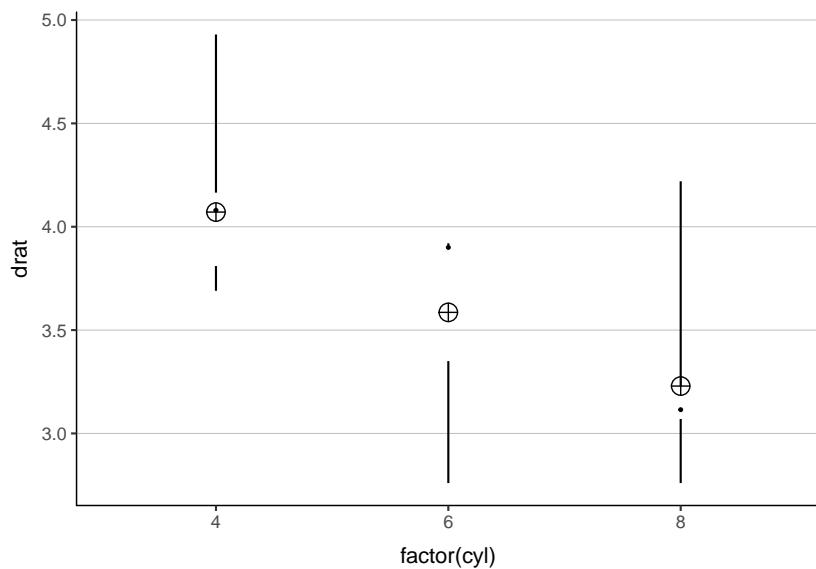
#### 10.12.4 Tufte-Boxplots

```
library(ggthemes)
+geom_tufteboxplot()

p <- ggplot(mtcars, aes(x = factor(cyl), y = drat)) +
  geom_tufteboxplot() +
  stat_summary(fun.y = mean, geom = "point", shape = 10,
    size = 4, position = position_dodge(0.75)) +
  cleanup
```

#### 10.12.5 Tufte-Boxplots

p



#### 10.12.6 Übung

1. Baut den folgenden Plot (“class\_cl.csv”) nach.



### 10.12.7 Lösung

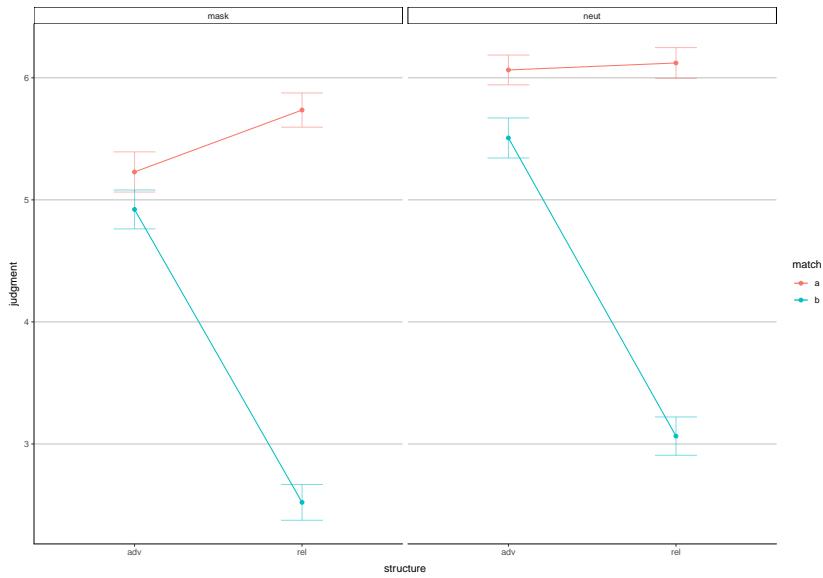
- Baut den folgenden Plot mit "class\_cl.csv" nach.

```
d <- read.csv("docs/data/tutgg/class_cl.csv")
d <- d[d$truth == "f", ]

p <- ggplot(d, aes(x = delivery, fill = judgment)) +
  geom_histogram(stat = "count", position = "dodge") +
  labs(title = "Lie 3AFC Data",
       y = "Absolute Frequency",
       x = "Delivery") +
  facet_wrap(~id) +
  cleanup
```

### 10.12.8 Übung II

- Baut den folgenden Plot mit "complex.csv" nach.

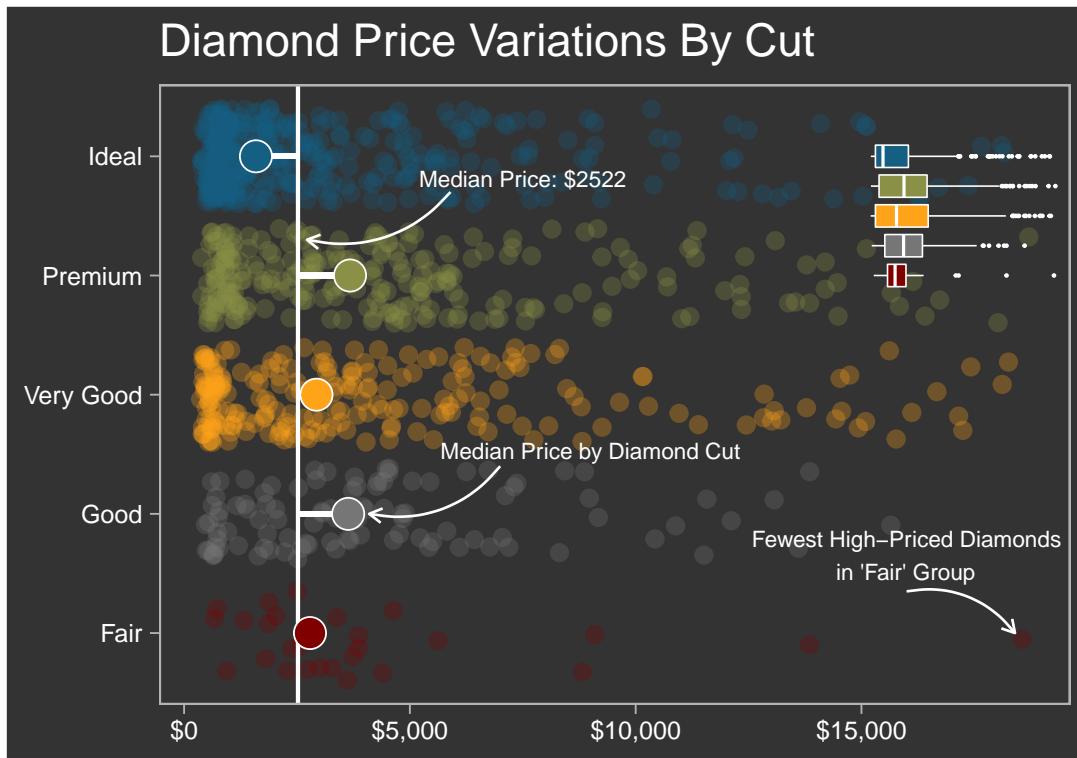


### 10.12.9 Lösung II

1. Baut den folgenden Plot mit "complex.csv" nach.

```
d <- read.csv("docs/data/tutgg/complex.csv")
p <- ggplot(d, aes(x = structure, y = judgment,
  color = match, group = match)) +
  stat_summary(fun.y = mean, geom = "line") +
  stat_summary(fun.y = mean, geom = "point") +
  stat_summary(fun.data = mean_se, geom = "errorbar",
    width = 0.25, alpha = .5, linetype = 1) +
  facet_wrap(~gender) +
  cleanup
```

### 10.12.10 Bonus: Pimp My Boxplot



### 10.12.11 Paper, die ihr lesen solltet:

Weissgerber et al. (2015): Beyond Bar and Line Graphs: Time for a New Data Presentation Paradigm

Cumming et al. (2007): Error bars in experimental biology

### 10.12.12 Arbeitsstreik

Wir sehen uns nächste Woche!

# 11

---

## Extrasitzung

### 11.1 Einfach Textdateien speichern

#### 11.1.1 Daten

```
library(ggplot2)

d <- as.data.frame(msleep) # from ggplot2 package
head(d)

##           name   genus vore      order conservation
## 1       Cheetah Acinonyx carni Carnivora        lc
## 2 Owl monkey     Aotus omni Primates      <NA>
## 3 Mountain beaver Aplodontia herbi Rodentia        nt
## 4 Greater short-tailed shrew Blarina omni Soricomorpha    lc
## 5          Cow     Bos herbi Artiodactyla domesticated
## 6 Three-toed sloth Bradypus herbi     Pilosa      <NA>
## sleep_total sleep_rem sleep_cycle awake brainwt bodywt
## 1       12.1       NA         NA  11.9      NA 50.000
## 2       17.0       1.8         NA   7.0  0.01550  0.480
## 3       14.4       2.4         NA   9.6      NA  1.350
## 4       14.9       2.3 0.1333333  9.1 0.00029  0.019
## 5        4.0       0.7 0.6666667 20.0 0.42300 600.000
## 6       14.4       2.2 0.7666667   9.6      NA  3.850
```

#### 11.1.2 sink

- Auch im Arbeitsverzeichnis ( $\rightarrow \text{getwd}$ )
- `cat` erzeugt den Textstring in seinem Argument auch in der Textdatei, nützlich für Titel, wenn mehrere Berechnungen in dieselbe Datei sollen
- `\n` erzeugt einen Zeilenumbruch
- Mit `with` erspart man sich ein kleines bisschen Schreibarbeit, weil man den Datensatz selbst nicht immer wieder innerhalb der Funktion eingeben muss
- Wichtig: `sink` immer mit `sink` beenden, wenn die Textdatei alle Infos enthält, die man abspeichern wollte

```
library(psych)
sink("descriptive_stats.txt")
cat("\n\n#####")
with(d,
  describeBy(price, cut))
cat("\n\n#####")
sink()
```

## 11.2 Funktionen aus dem tidyverse

### 11.2.1 Tidyverse

```
https://www.tidyverse.org
install.packages("tidyverse")

library(tidyverse)
```

### 11.2.2 Funktionen, die man kennen sollte

```
mutate() # spalten erstellen oder verändern
summarise() # mehrere werte auf einen reduzieren (zB mean)
group_by() # datenblatt gruppieren (wie by() )
%>% # "pipe"
```

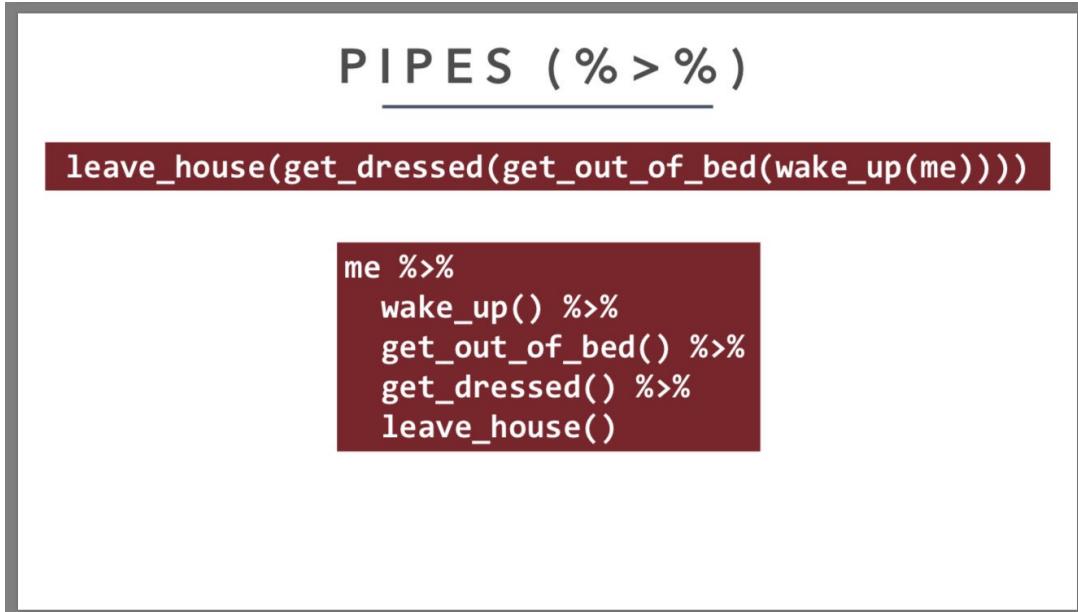
### 11.2.3 Pipe %>%

### 11.2.4 Beispiele

```
# Spalte für Faktor Issueness anlegen
d <- mutate(d, issue = case_when(
  str_detect(d$V8, "_at") ~ "at-issue",
  str_detect(d$V8, "_non") ~ "non-at-issue"))

# Spalten zu Faktoren konvertieren
d <- mutate_at(d,
  c("issue", "trigger", "itemid", "lex"),
  funs(factor(.)))

# neue Spalte und Faktoren mit Pipes
d <- d %>%
  mutate(issue = case_when(
    str_detect(V8, "_at") ~ "at-issue",
    str_detect(V8, "_non") ~ "non-at-issue")) %>%
  mutate_at(c("issue", "trigger", "itemid", "lex"),
    funs(factor(.)))
```

Abb. 11.1: <https://twitter.com/andrewweiss/status/1173743447171354624>

### 11.2.5 Weitere Beispiele

Mit dem altbekannten Diamantendatenset:

```
d <- diamonds
head(d)

##   carat      cut color clarity depth table price     x     y     z
## 1 0.23    Ideal   E    SI2   61.5    55   326 3.95 3.98 2.43
## 2 0.21  Premium   E    SI1   59.8    61   326 3.89 3.84 2.31
## 3 0.23     Good   E    VS1   56.9    65   327 4.05 4.07 2.31
## 4 0.29  Premium   I    VS2   62.4    58   334 4.20 4.23 2.63
## 5 0.31     Good   J    SI2   63.3    58   335 4.34 4.35 2.75
## 6 0.24 Very Good   J   VVS2   62.8    57   336 3.94 3.96 2.48
```

### 11.2.6 Vorkommen zählen

```
# reihen zählen und nach color und cut aufspalten
d %>% count(color, cut) %>%
  head

## # A tibble: 6 x 3
##   color cut     n
##   <ord> <ord> <int>
## 1 D     Fair     163
## 2 D     Good    662
## 3 D     Very Good 1513
## 4 D     Premium  1603
```

```
## 5 D      Ideal      2834
## 6 E      Fair       224
```

### 11.2.7 Zählen und Sortieren

```
d %>% count(color, cut) %>%
  arrange(n) %>%
  head()

## # A tibble: 6 x 3
##   color cut     n
##   <ord> <ord> <int>
## 1 J     Fair    119
## 2 D     Fair    163
## 3 I     Fair    175
## 4 E     Fair    224
## 5 H     Fair    303
## 6 J     Good    307
```

### 11.2.8 Sortieren und Spalten auswählen

```
d %>% arrange(desc(price)) %>%
  dplyr::select(price, carat, color, cut) %>%
  head()

## # A tibble: 6 x 4
##   price carat color      cut
##   <dbl>  <dbl> <ord> <ord>
## 1 18823  2.29  I   Premium
## 2 18818  2.00  G   Very Good
## 3 18806  1.51  G   Ideal
## 4 18804  2.07  G   Ideal
## 5 18803  2.00  H   Very Good
## 6 18797  2.29  I   Premium
```

### 11.2.9 Gruppieren und rechnen

```
d %>% group_by(color) %>%
  summarise(mprice = mean(price),
            medprice = median(price))

## # A tibble: 7 x 3
##   color mprice medprice
##   <ord>   <dbl>    <dbl>
## 1 D      3170.    1838
## 2 E      3077.    1739
## 3 F      3725.    2344.
## 4 G      3999.    2242
## 5 H      4487.    3460
## 6 I      5092.    3730
## 7 J      5324.    4234
```

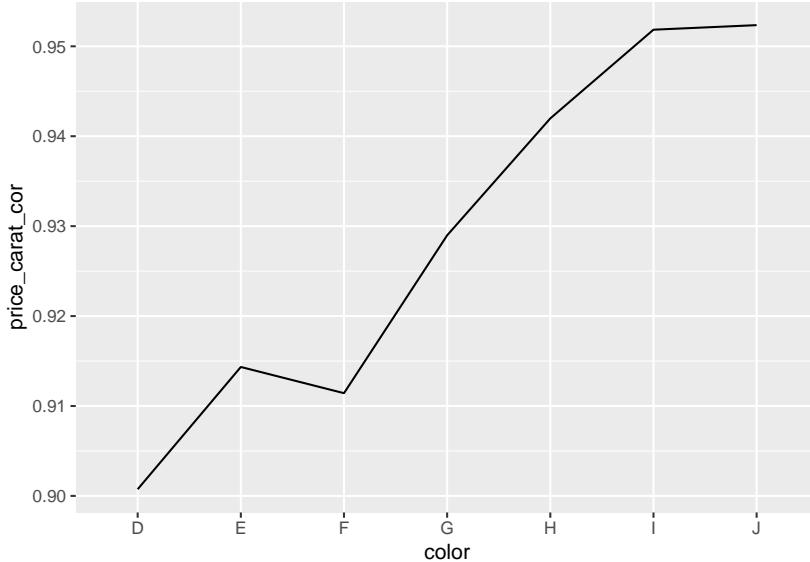
### 11.2.10 Gruppieren, berechnen, filtern

```
# nach schliff gruppieren
# spalte mit mittlerem preis hinzufügen
# und maximale verkaufspreise anzeigen lassen
d %>% group_by(cut) %>%
  mutate(mprice = mean(price)) %>%
  filter(price == max(price))

## # A tibble: 5 x 11
## # Groups:   cut [5]
##   carat cut      color clarity depth table price     x     y     z mprice
##   <dbl> <ord>    <ord> <ord>  <dbl> <dbl> <int> <dbl> <dbl> <dbl> <dbl>
## 1  2.01 Fair      G     SI1    70.6   64 18574  7.43  6.64  4.69  4359.
## 2  2.8   Good     G     SI2    63.8   58 18788  8.9   8.85  0     3929.
## 3  1.51 Ideal     G     IF     61.7   55 18806  7.37  7.41  4.56  3458.
## 4  2     Very Good G     SI1    63.5   56 18818  7.9   7.97  5.04  3982.
## 5  2.29 Premium   I     VS2    60.8   60 18823  8.5   8.47  5.16  4584.
```

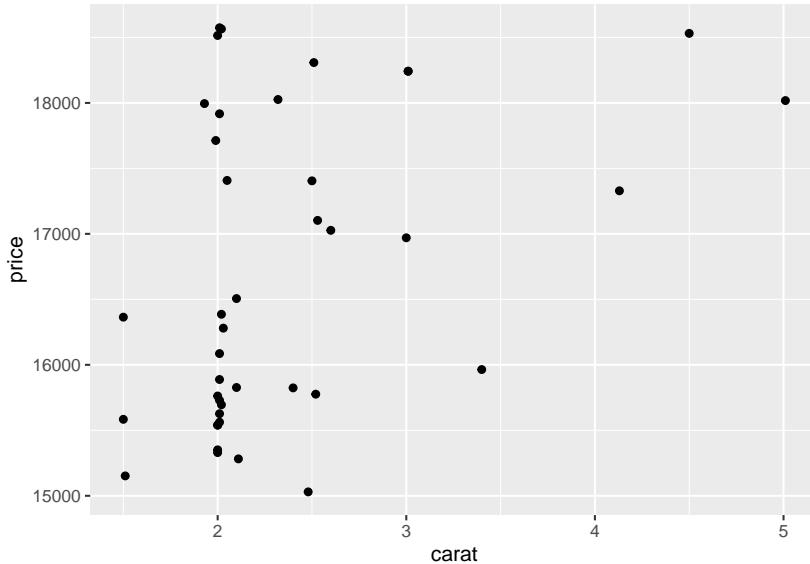
### 11.2.11 Gruppenkorrelationen plotten

```
# nach farbe gruppieren
# und korrelation zwischen preis und gewicht berechnen
# und plotten
d %>% group_by(color) %>%
  summarise(price_carat_cor = cor(price, carat)) %>%
  ggplot(aes(color, price_carat_cor, group = 1)) +
  geom_path()
```



### 11.2.12 Filtern und plotten

```
# nur zeilen mit preis > 15000 und schliff == Fair
# im plot darstellen
d %>% filter(price > 15000, cut == "Fair") %>%
  ggplot(aes(carat, price)) +
  geom_point()
```



### 11.2.13 Ein Tutorial

```
vignette("dplyr")
```

### 11.2.14 Speed-Up

Bei größeren Datensätzen (Korpusdaten, EGG, Eye-Tracking) kann es vorkommen, dass die inhärenten Funktionen von R und dem `tidyverse` zu langsam sind. Wem eine solche Situation häufiger unterkommt, sollte es sich überlegen, die unteren beiden Pakete in seinen Workflow zu integrieren. Beide bieten `tidyverse`-Syntax mit der Geschwindigkeit von `data.table`, einem Paket, das zwar wesentlich schneller als andere Lösungen ist, aber einiges an Komplexität mit sich bringt.

1. <https://github.com/TysonStanley/tidyfast>
2. <https://github.com/markfairbanks/tidytable>

### 11.3 For-Loops

#### 11.3.1 Loops

```
# ausgabe aller geraden zahlen zwischen 1 und 15
for (i in 1:15) { # 1:15 = zahlenreihe 1 bis 15
  if (i %% 2) { # wenn die jeweilige zahl durch 2 teilbar ist, ...
    next} # dann gehe zum nächsten Schritt und
  print(i) # gib sie aus
}

## [1] 2
## [1] 4
## [1] 6
## [1] 8
## [1] 10
## [1] 12
## [1] 14
```

#### 11.3.2 Loops

```
# summe: x_1 + x_2 + ... + x_n
my.sum <- function(vector) {
  sum <- 0 # ist nötig, damit wir immer wieder neu bei
  # null beginnen, wenn wir die funktion nochmal benutzen wollen
  for (i in vector) {sum <- sum + i
  }
  sum
}
a <- c(1, 2, 3, 4, 5, 6)
my.sum(a)

## [1] 21
sum(a)

## [1] 21
my.sum(c(1, 2, 3, 4, 5, 6))

## [1] 21
```

#### 11.3.3 Übung

1. Schreibe eine Funktion "my.mean", die den Mittelwert eines Vektors berechnet
2. Schreibe eine Funktion "my.var", die die Varianz eines Vektors berechnet

#### 11.3.4 Lösungen

1. Schreibe eine Funktion "my.mean", die den Mittelwert eines Vektors berechnet

```
# mittelwert:  $(x_1 + x_2 + \dots + x_n)/n$ 
my.mean <- function(vector) {
  sum <- 0 # wie oben
  for (i in vector) {
    sum <- sum + i
  }
  sum / length(vector)
}
a <- c(1, 2, 3, 4, 5, 6)
my.mean(a) == mean(a)

## [1] TRUE
my.mean(a)

## [1] 3.5
```

### 11.3.5 Lösungen

2. Schreibe eine Funktion "my.var", die die Varianz eines Vektors berechnet

```
# varianz:
#  $((x_1 - \text{mittelwert})^2 + (x_2 - \text{mittelwert})^2 + \dots + (x_n - \text{mittelwert})^2)/n$ 
my.var <- function(vector) {
  sum <- 0
  meanie <- my.mean(vector)
  for (i in vector) {
    sum <- sum + (i - meanie)^2
  }
  sum / (length(vector) - 1)
}
my.var(a) == var(a)

## [1] TRUE
my.var(a)

## [1] 3.5
```

## 11.4 Funktionen, die das Coden erleichtern

### 11.4.1 Daten

```
d <- InsectSprays
head(d)

##   count spray
## 1     10     A
## 2      7     A
## 3     20     A
## 4     14     A
## 5     14     A
## 6     12     A

str(d)
```

```
## 'data.frame':    72 obs. of  2 variables:
## $ count: num  10 7 20 14 14 12 10 23 17 20 ...
## $ spray: Factor w/ 6 levels "A","B","C","D",...: 1 1 1 1 1 1 1 1 1 1 ...
```

### 11.4.2 by()

```
by(d$count, d$spray, mean)
```

```
## d$spray: A
## [1] 14.5
## -----
## d$spray: B
## [1] 15.33333
## -----
## d$spray: C
## [1] 2.083333
## -----
## d$spray: D
## [1] 4.916667
## -----
## d$spray: E
## [1] 3.5
## -----
## d$spray: F
## [1] 16.66667
```

### 11.4.3 with()

Struktur:

- `with(Variable, Befehl/Funktion)`

Der Befehl/die Funktion innerhalb von `within` wird mit der angegebenen Variable durchgeführt

### 11.4.4 with()

Hier die Daten, mit denen ich die Funktionsweise von `with` illustrieren werde:

```
head(ToothGrowth)
```

```
##   len supp dose
## 1 4.2   VC  0.5
## 2 11.5  VC  0.5
## 3 7.3   VC  0.5
## 4 5.8   VC  0.5
## 5 6.4   VC  0.5
## 6 10.0  VC  0.5
```

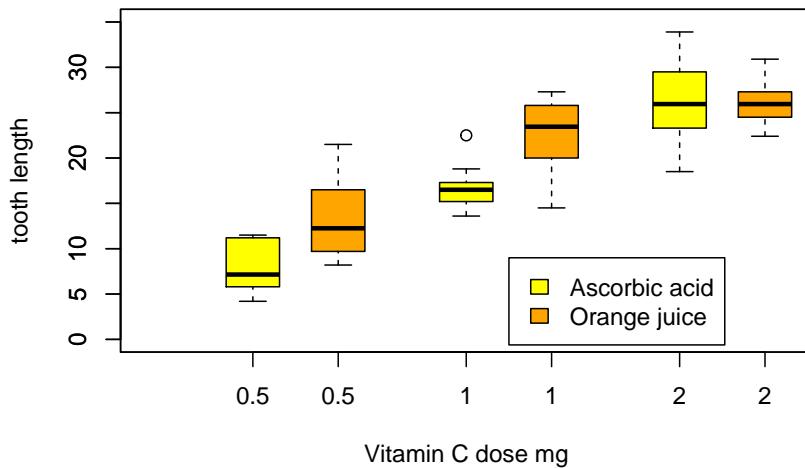
```
str(ToothGrowth)
```

```
## 'data.frame':    60 obs. of  3 variables:
## $ len : num  4.2 11.5 7.3 5.8 6.4 10 11.2 11.2 5.2 7 ...
## $ supp: Factor w/ 2 levels "OJ","VC": 2 2 2 2 2 2 2 2 2 2 ...
## $ dose: num  0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 ...
```

### 11.4.5 with()

```
with(ToothGrowth, {
  boxplot(len ~ dose, boxwex = 0.25, at = 1:3 - 0.2,
    subset = (supp == "VC"), col = "yellow",
    main = "Guinea Pigs' Tooth Growth",
    xlab = "Vitamin C dose mg",
    ylab = "tooth length", ylim = c(0, 35))
  boxplot(len ~ dose, add = TRUE, boxwex = 0.25, at = 1:3 + 0.2,
    subset = supp == "OJ", col = "orange")
  legend(2, 9, c("Ascorbic acid", "Orange juice"),
    fill = c("yellow", "orange")))
})
```

**Guinea Pigs' Tooth Growth**



### 11.4.6 attach()

- Alle folgenden Befehle werden mit der angegebenen Variable ausgeführt
- Eignet sich, wenn mit wenigen Tabellen/Variablen gleichzeitig gearbeitet wird
- Eine Variable, die mit attach ausgewählt wurde, kann mit detach wieder entfernt werden, wenn beispielsweise eine neue Variable verwendet werden soll

```
attach(d)
mean(count)

## [1] 9.5

sd(count)

## [1] 7.203286
```

```
median(count)
## [1] 7
mad(count, constant = 1)
## [1] 5
detach(d)
```

### 11.4.7 apply()-Familie

tapply(Vektorvariable/Spaltenvariable, INDEX = Vektorvariable/Spalte, Befehl)

- für Tabellen/Data-frames
- Als zweites Argument muss eine Spalte angegeben werden, nach deren Inhalt die Funktion ausgeführt wird
  - Es werden quasi Subsets nach dem Inhalt der angegebenen Spalte erstellt, über die dann einzeln der Befehl angewandt wird.
- Alternative Befehle: sapply, lapply

### 11.4.8 apply()-Familie

```
# Berechnet Mittelwert und Median der Insektenanzahl
# für den jeweiligen Typ von Insektenspray
tapply(d$count, d$spray, mean)
```

```
##          A          B          C          D          E          F
## 14.500000 15.333333  2.083333  4.916667  3.500000 16.666667
tapply(d$count, d$spray, median)

##          A          B          C          D          E          F
## 14.0 16.5  1.5  5.0  3.0 15.0
```

### 11.4.9 Unterschied by() und tapply()

## 11.5 Janitor-Paket

### 11.5.1 Installieren und laden

```
install.packages("janitor")
library(janitor)
```

### 11.5.2 Bessere Häufigkeitstabellen

- Alternative zu table
- Gibt Ergebnisse als data frame aus

```
d <- diamonds
tab <- tabyl(d, color, cut)
tab # absolute Häufigkeiten

##   color Fair Good Very Good Premium Ideal
##   D    163   662    1513   1603   2834
##   E    224   933    2400   2337   3903
##   F    312   909    2164   2331   3826
##   G    314   871    2299   2924   4884
##   H    303   702    1824   2360   3115
##   I    175   522    1204   1428   2093
##   J    119   307     678    808    896

adorn_percentages(tab, denominator = "row") # relative Häufigkeiten

##   color      Fair      Good Very Good Premium Ideal
##   D 0.02405904 0.09771218 0.2233210 0.2366052 0.4183026
##   E 0.02286414 0.09523323 0.2449730 0.2385424 0.3983873
##   F 0.03269755 0.09526305 0.2267868 0.2442884 0.4009642
##   G 0.02780730 0.07713425 0.2035955 0.2589444 0.4325186
##   H 0.03648844 0.08453757 0.2196532 0.2842004 0.3751204
##   I 0.03227591 0.09627444 0.2220583 0.2633714 0.3860199
##   J 0.04237892 0.10933048 0.2414530 0.2877493 0.3190883
```

### 11.5.3 Bessere Häufigkeitstabellen

- denominator-Argument verändert, wie Häufigkeiten berechnet werden
- adorn\_pct\_formatting: \*100 plus %
- zusätzlich möglich: adorn\_totals

```
adorn_pct_formatting(adorn_percentages(tab, denominator = "col"))
```

```
##   color Fair Good Very Good Premium Ideal
##   D 10.1% 13.5%    12.5% 11.6% 13.2%
##   E 13.9% 19.0%    19.9% 16.9% 18.1%
##   F 19.4% 18.5%    17.9% 16.9% 17.8%
##   G 19.5% 17.8%    19.0% 21.2% 22.7%
##   H 18.8% 14.3%    15.1% 17.1% 14.5%
##   I 10.9% 10.6%    10.0% 10.4% 9.7%
##   J  7.4%  6.3%     5.6%  5.9% 4.2%
```

```
adorn_pct_formatting(adorn_percentages(tab, denominator = "all"))
```

```
##   color Fair Good Very Good Premium Ideal
##   D 0.3% 1.2%    2.8% 3.0% 5.3%
##   E 0.4% 1.7%    4.4% 4.3% 7.2%
##   F 0.6% 1.7%    4.0% 4.3% 7.1%
##   G 0.6% 1.6%    4.3% 5.4% 9.1%
##   H 0.6% 1.3%    3.4% 4.4% 5.8%
##   I 0.3% 1.0%    2.2% 2.6% 3.9%
##   J 0.2% 0.6%    1.3% 1.5% 1.7%
```

### 11.5.4 Leere Zeilen und Spalten entfernen

```

q <- data.frame(v1 = c(1, NA, 3),
  v2 = c(NA, NA, NA),
  v3 = c("a", NA, "b"))
q

##   v1 v2   v3
## 1  1  NA    a
## 2  NA NA <NA>
## 3  3  NA    b

remove_empty(q, c("rows", "cols"))

##   v1 v3
## 1  1  a
## 3  3  b

```

### 11.5.5 Spalten mit konstanten Werten entfernen

```

(a <- data.frame(good = 1:3, boring = "the same"))

##   good   boring
## 1     1 the same
## 2     2 the same
## 3     3 the same
remove_constant(a)

##   good
## 1     1
## 2     2
## 3     3

```

### 11.5.6 clean\_names()

```

# Create a data.frame with dirty names
test_df <- as.data.frame(matrix(ncol = 6))
names(test_df) <- c("firstName", "abc@!*", "% successful (2009)",
  "REPEAT VALUE", "REPEAT VALUE", "")
test_df

##   firstName abc@!* % successful (2009) REPEAT VALUE REPEAT VALUE
## 1          NA          NA          NA          NA          NA NA

clean_names(test_df)

##   first_name abc percent_successful_2009 repeat_value repeat_value_2 x
## 1          NA          NA          NA          NA          NA NA

```

## 11.6 Automatische Tabellen mit gtsummary

### 11.6.1 Automatische Tabellen mit gtsummary

```

library(gtsummary)
options(
  gtsummary.add_p.test.continuous_by2 = "t.test",
  gtsummary.add_p.test.continuous = "aov"
)
d <- diamonds
sum_tbl <- d %>%
  select(price, cut, carat) %>%
  mutate(cut = factor(cut, ordered = F)) %>%
  filter(cut %in% c("Premium", "Fair", "Good")) %>%
  droplevels() %>%
 tbl_summary(by = cut,
  statistic = list(all_continuous() ~ "{mean} ({sd})"),
  label = list(carat ~ "Carat",
    price ~ "Price")) %>%
add_p()

```

### 11.6.2 Automatische Tabellen

```
sum_tbl
```

Characteristic Fair, N = 1610 <sup>1</sup> Good, N = 4906 <sup>1</sup> Premium, N = 13791 <sup>1</sup> p-value <sup>2</sup>				
Price	4359 (3560)	3929 (3682)	4584 (4349)	<0.001
Carat	1.05 (0.52)	0.85 (0.45)	0.89 (0.52)	<0.001

<sup>1</sup>Statistics presented: mean (SD)

<sup>2</sup>Statistical tests performed: One-way ANOVA

### 11.6.3 correlation-Paket

```

library(correlation)
d %>%
  select(price, carat, depth) %>%
  correlation()

## Parameter1 | Parameter2 |      r |      95% CI |      t |      df |      p |   Method | n_Obs
## -----
## price     |      carat |  0.92 | [ 0.92,  0.92] | 551.41 | 53938 | < .001 | Pearson | 53940
## price     |      depth | -0.01 | [-0.02,  0.00] | -2.47 | 53938 |  0.013 | Pearson | 53940
## carat     |      depth |  0.03 | [ 0.02,  0.04] |  6.56 | 53938 | < .001 | Pearson | 53940

```

## 11.7 Automatische Berichte

### 11.7.1 report-Paket

Hier zu finden: <https://github.com/easystats/report>

```
install.packages("devtools")
devtools::install_github("easystats/report")

library(report)
```

### 11.7.2 Dataframes

```
d <- diamonds
d %>%
  select(price, carat, cut) %>%
  report_sample()

## # Descriptive Statistics
##
## Characteristic | Summary
## -----
## Mean price (SD) | 3932.8 (3989.4)
## Mean carat (SD) | 0.8 (0.5)
## cut [Fair], % | 3.0
## cut [Good], % | 9.1
## cut [Very Good], % | 22.4
## cut [Premium], % | 25.6
## cut [Ideal], % | 40.0

report_sample(d, group_by = "cut", select = c("price", "carat"))

## # Descriptive Statistics
##
## Characteristic | Fair (n=1610) | Good (n=4906) | Very Good (n=12082) | Premium (n=13791) | Ideal (n=)
## -----
## Mean price (SD) | 4358.8 (3560.4) | 3928.9 (3681.6) | 3981.8 (3935.9) | 4584.3 (4349.2) | 3457.5 (3
## Mean carat (SD) | 1.0 (0.5) | 0.8 (0.5) | 0.8 (0.5) | 0.9 (0.5) | 0.1 (0.0)
```

### 11.7.3 t-Test

```
d %>%
  filter(cut == "Fair" | cut == "Ideal") %>%
  droplevels() %>%
  t.test(carat ~ cut, data = .) %>%
  report()
```

The Welch Two Sample t-test suggests that the difference of carat by cut (mean in group Fair = 1.05, mean in group Ideal = 0.70) is significant (difference = 0.34, 95% CI [0.32, 0.37],  $t(1781.99) = 26.00$ ,  $p < .001$ ) and can be considered as very large (Cohen's d = 1)., The Welch Two Sample t-test suggests that the difference of carat by cut (mean in group Fair = 1.05, mean in group Ideal = 0.70) is significant (difference = 0.34, 95% CI [0.32, 0.37],  $t(1781.99) = 26.00$ ,  $p < .001$ ) and can be considered as very large (Cohen's d = 1)., The Welch Two Sample t-test suggests that the difference of carat by cut (mean in group Fair = 1.05, mean in group Ideal = 0.70) is significant (difference = 0.34, 95% CI [0.32, 0.37],  $t(1781.99) = 26.00$ ,  $p < .001$ ) and can be considered as very large (Cohen's d = 1). and The Welch Two Sample t-test suggests that the difference of carat by cut (mean in group Fair = 1.05, mean in group Ideal = 0.70) is significant (difference = 0.34, 95% CI [0.32, 0.37],  $t(1781.99) = 26.00$ ,  $p < .001$ ) and can be considered as very large (Cohen's d = 1).

### 11.7.4 ANOVA (base)

```
stdaoov <- aov(price ~ color * cut, data = d)
stdaoov %>%
  report()
```

The ANOVA suggests that:

- The main effect of color is significant ( $F(6, 53905) = 294.07$ ,  $p < .001$ ) and can be considered as small (partial omega squared = 0.03).
- The main effect of cut is significant ( $F(4, 53905) = 159.36$ ,  $p < .001$ ) and can be considered as small (partial omega squared = 0.01).
- The interaction between color and cut is significant ( $F(24, 53905) = 4.53$ ,  $p < .001$ ) and can be considered as very small (partial omega squared = 0.00).

### 11.7.5 ANOVA (afex)

```
library(afex)
d$id <- 1:length(d$carat)
afexaoov <- aov_ez("id", dv = "price", between = c("color", "cut"), data = d)
afexaoov$aov %>%
  report()
```

The ANOVA suggests that:

- The main effect of color is significant ( $F(6, 53905) = 294.07$ ,  $p < .001$ ) and can be considered as small (partial omega squared = 0.03).
- The main effect of cut is significant ( $F(4, 53905) = 159.36$ ,  $p < .001$ ) and can be considered as small (partial omega squared = 0.01).
- The interaction between color and cut is significant ( $F(24, 53905) = 4.53$ ,  $p < .001$ ) and can be considered as very small (partial omega squared = 0.00).

```
afexaoov$aov %>%
  report() %>%
  table_short()

## Parameter | Sum_Squares |   df | Mean_Square |      F |     p | Omega_Sq_partial
## -----
## color      | 2.68491e+10 |     6 | 4.47485e+09 | 294.07 | 0.00 |          0.03
## cut        | 9.69968e+09 |     4 | 2.42492e+09 | 159.36 | 0.00 |          0.01
## color:cut  | 1.65346e+09 |    24 | 6.88940e+07 |  4.53 | 0.00 |          0.00
## Residuals  | 8.20271e+11 | 53905 | 1.52170e+07 |      |      |
```

### 11.7.6 LMM

```
d %>%
  filter(cut == "Fair" | cut == "Ideal") %>%
  droplevels() %>%
  lmer(price ~ carat + cut + (1 | carat), data = .) %>%
  report()
```

We fitted a linear mixed model (estimated using REML and nloptwrap optimizer) to predict price with carat and cut (formula = price ~ carat + cut). The model included carat as random effects (formula = ~1 | carat). Standardized parameters were obtained by fitting the model on a standardized version of the dataset. Effect sizes were labelled following Funder's (2019) recommendations. The model's total explanatory power is substantial (conditional R<sup>2</sup> = 0.87) and the part related to the fixed effects alone (marginal R<sup>2</sup>) is of 0.57. The model's intercept, corresponding to price = 0, carat = 0 and cut = Fair, is at -1192.60 (SE = 273.06, 95% CI [-1727.79, -657.41], p < .001). Within this model:

- The effect of carat is positive and can be considered as very large and significant (beta = 6346.99, SE = 165.26, 95% CI [6023.09, 6670.89], std. beta = 1.67, p < .001).
- The effect of cut.L is positive and can be considered as small and significant (beta = 1129.24, SE = 26.68, 95% CI [1076.95, 1181.54], std. beta = 0.30, p < .001).

## 11.8 Daten untersuchen

### 11.8.1 Das dlookr-Paket

```
# install.packages("dlookr")
library(dlookr)
```

### 11.8.2 Datenblatt diagonalstizieren

```
# check for missing data and distinct values per column
d %>%
  diagnose()

## # A tibble: 11 x 6
##   variables types  missing_count missing_percent unique_count unique_rate
##   <chr>     <chr>      <int>           <dbl>        <int>       <dbl>
## 1 carat     numeric      0             0          273    0.00506
## 2 cut       ordered       0             0            5    0.0000927
## 3 color     ordered       0             0            7    0.000130
## 4 clarity   ordered       0             0            8    0.000148
## 5 depth     numeric      0             0           184    0.00341
## 6 table     numeric      0             0           127    0.00235
## 7 price     integer      0             0          11602   0.215
## 8 x         numeric      0             0            554    0.0103
## 9 y         numeric      0             0            552    0.0102
## 10 z        numeric      0             0            375    0.00695
## 11 id       integer      0             0          53940    1

# descriptive statistics of all numeric variables (min, max, mean, etc.)
d %>%
  diagnose_numeric()

## # A tibble: 8 x 10
##   variables min      Q1      mean    median      Q3      max zero minus outlier
##   <chr>     <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <int> <int> <int>
## 1 carat      0.2     0.4     0.798  7.00e-1  1.04e0   5.01e0     0     0    1889
## 2 depth      43      61      61.7   6.18e+1  6.25e1   7.90e1     0     0    2545
```

```

## 3 table      43      56      57.5  5.70e+1  5.90e1  9.50e1      0      0      605
## 4 price     326     950    3933.  2.40e+3  5.32e3  1.88e4      0      0    3538
## 5 x          0       4.71    5.73  5.70e+0  6.54e0  1.07e1      8      0      32
## 6 y          0       4.72    5.73  5.71e+0  6.54e0  5.89e1      7      0      29
## 7 z          0       2.91    3.54  3.53e+0  4.04e0  3.18e1     20      0      49
## 8 id         1     13486.  26970.  2.70e+4  4.05e4  5.39e4      0      0      0

# values and distribution of all categorical variables
d %>%
  diagnose_category()

## # A tibble: 20 x 6
##   variables levels      N  freq ratio  rank
##   <chr>     <ord> <int> <dbl> <dbl> <int>
## 1 cut       Ideal    53940 21551 40.0    1
## 2 cut       Premium  53940 13791 25.6    2
## 3 cut       Very Good 53940 12082 22.4    3
## 4 cut       Good    53940  4906  9.10   4
## 5 cut       Fair    53940  1610  2.98   5
## 6 color    G       53940 11292 20.9    1
## 7 color    E       53940  9797 18.2    2
## 8 color    F       53940  9542 17.7    3
## 9 color    H       53940  8304 15.4    4
## 10 color   D       53940  6775 12.6    5
## 11 color   I       53940  5422 10.1    6
## 12 color   J       53940  2808  5.21   7
## 13 clarity SI1    53940 13065 24.2    1
## 14 clarity VS2    53940 12258 22.7    2
## 15 clarity SI2    53940  9194 17.0    3
## 16 clarity VS1    53940  8171 15.1    4
## 17 clarity VVS2   53940  5066  9.39   5
## 18 clarity VVS1   53940  3655  6.78   6
## 19 clarity IF     53940  1790  3.32   7
## 20 clarity I1     53940   741  1.37   8

# show outlier information for all numerical variables
d %>%
  diagnose_outlier()

##   variables outliers_cnt outliers_ratio outliers_mean   with_mean without_mean
## 1 carat        1889    3.50203930    2.153684 7.979397e-01    0.748738
## 2 depth        2545    4.71820541    61.204794 6.174940e+01   61.776373
## 3 table         605     1.12161661    64.842975 5.745718e+01   57.373404
## 4 price        3538    6.55913978   14944.776427 3.932800e+03  3159.807111
## 5 x            32      0.05932518    7.239375 5.731157e+00   5.730262
## 6 y            29      0.05376344    9.773448 5.734526e+00   5.732353
## 7 z            49      0.09084168    4.054694 3.538734e+00   3.538265
## 8 id           0       0.00000000      NaN 2.697050e+04  26970.500000

```

### 11.8.3 Normalverteilung

```

# shapiro-wilk test
d %>%
  select(price, carat, color, cut) %>%
  group_by(color) %>%
  normality()

## # A tibble: 14 x 5
##   variable color statistic p_value sample

```

```

##   <chr>  <ord>    <dbl>    <dbl>  <dbl>
## 1 price    D     0.742 1.76e-66  5000
## 2 price    E     0.728 1.59e-67  5000
## 3 price    F     0.782 3.14e-63  5000
## 4 price    G     0.805 3.96e-61  5000
## 5 price    H     0.838 1.04e-57  5000
## 6 price    I     0.853 5.91e-56  5000
## 7 price    J     0.889 6.11e-41  2808
## 8 carat    D     0.875 4.51e-53  5000
## 9 carat    E     0.879 2.10e-52  5000
## 10 carat   F     0.910 2.20e-47  5000
## 11 carat   G     0.904 1.37e-48  5000
## 12 carat   H     0.924 1.18e-44  5000
## 13 carat   I     0.935 4.00e-42  5000
## 14 carat   J     0.955 1.64e-28  2808

```

#### 11.8.4 Deskriptive Statistik

```

# tidy alternative for psych::describe
d %>%
  describe() %>%
  select(-starts_with("p")) # don't show percentiles

## # A tibble: 8 x 9
##   variable   n    na   mean     sd se_mean    IQR skewness kurtosis
##   <chr>     <int> <dbl> <dbl>  <dbl> <dbl> <dbl>    <dbl>
## 1 carat      53940  0  5.73  1.12  0.00483  1.83  0.379  -0.618
## 2 depth      53940  0  5.73  1.12  0.00483  1.83  0.379  -0.618
## 3 table      53940  0  5.73  1.12  0.00483  1.83  0.379  -0.618
## 4 price      53940  0  5.73  1.12  0.00483  1.83  0.379  -0.618
## 5 x          53940  0  5.73  1.12  0.00483  1.83  0.379  -0.618
## 6 y          53940  0  5.73  1.12  0.00483  1.83  0.379  -0.618
## 7 z          53940  0  5.73  1.12  0.00483  1.83  0.379  -0.618
## 8 id         53940  0  5.73  1.12  0.00483  1.83  0.379  -0.618

# tidy alternative for psych::describeBy
d %>%
  group_by(cut) %>%
  describe(price, carat) %>%
  select(-starts_with("p")) # don't show percentiles

## # A tibble: 10 x 10
##   variable cut      n    na   mean     sd se_mean    IQR skewness kurtosis
##   <chr>    <ord> <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>    <dbl>
## 1 price    Fair    1610  0  4.36e+3 3.56e+3 8.87e+1 3155.   1.78   3.09
## 2 price    Good   4906  0  3.93e+3 3.68e+3 5.26e+1 3883.   1.72   3.05
## 3 price    Very ~ 12082  0  3.98e+3 3.94e+3 3.58e+1 4461.   1.60   2.24
## 4 price    Premi~ 13791  0  4.58e+3 4.35e+3 3.70e+1 5250.   1.33   1.07
## 5 price    Ideal   21551  0  3.46e+3 3.81e+3 2.59e+1 3800.   1.84   2.98
## 6 carat    Fair    1610  0  1.05e+0 5.16e-1 1.29e-2  0.5    1.69   5.34
## 7 carat    Good   4906  0  8.49e-1 4.54e-1 6.48e-3  0.51   1.03   1.23
## 8 carat    Very ~ 12082  0  8.06e-1 4.59e-1 4.18e-3  0.61   0.994  0.896
## 9 carat    Premi~ 13791  0  8.92e-1 5.15e-1 4.39e-3  0.79   0.862  0.431
## 10 carat   Ideal   21551  0  7.03e-1 4.33e-1 2.95e-3  0.66   1.34   1.63

```

### 11.8.5 Komplette Diagnoseberichte (standardmäßig als PDF)

```
# generate pdf report for various diagnosis tasks
# d %>%
#   diagnose_report()
# generate pdf report for exploratory data analysis
# (including correlations and transformations)
# d %>%
#   eda_report()
```

## 11.9 Bildverarbeitung (und -bearbeitung) in R

### 11.9.1 magick

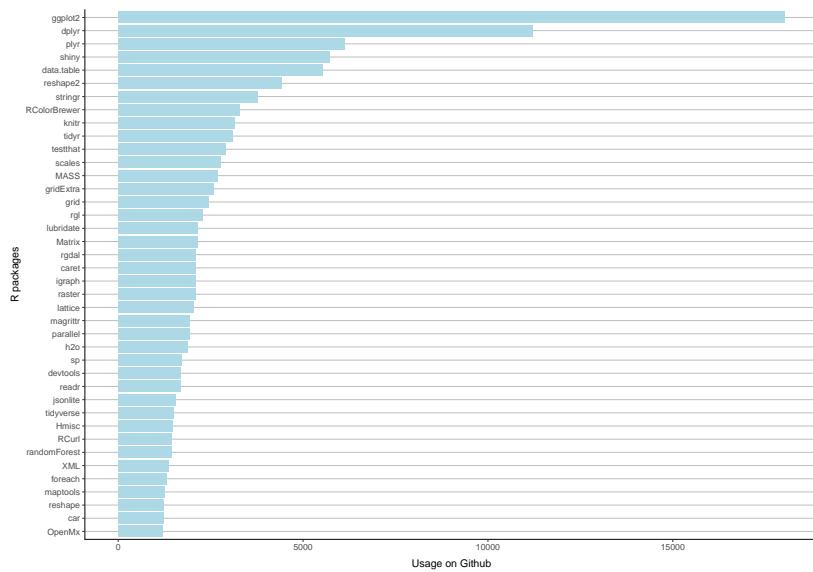
<https://cran.r-project.org/web/packages/magick/vignettes/intro.html>

```
# install.packages("magick")
library(magick)
```

## 11.10 Popularity Contest

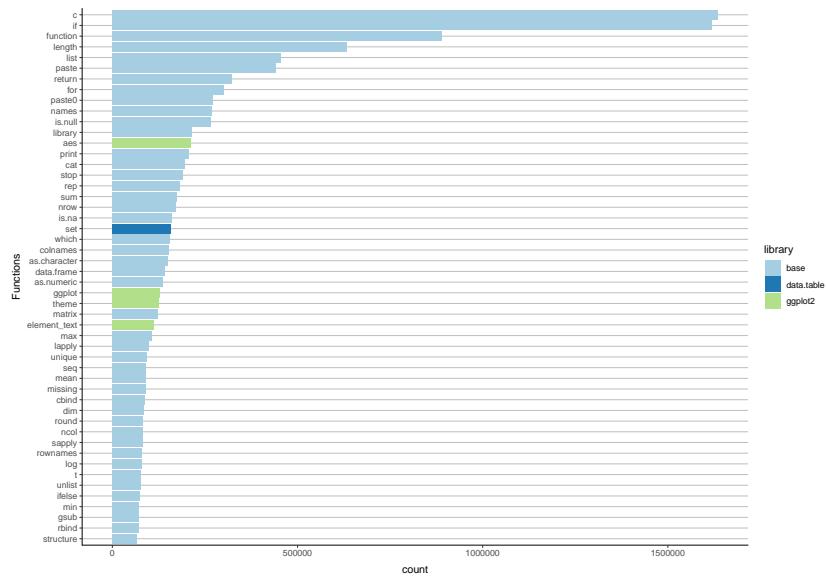
### 11.10.1 Die populärsten Pakete

Hier geklaut: <https://github.com/v-kozhevnikov>



### 11.10.2 Die populärsten Pakete

Hier geklaut: <https://github.com/v-kozhevnikov>



### 11.10.3 fin



## 11.11 Anhang: Die wichtigsten Funktionen

### 11.11.1 Allgemein

Hier geklaut: [http://www.sr.bham.ac.uk/~ajrs/R/r-function\\_list.html](http://www.sr.bham.ac.uk/~ajrs/R/r-function_list.html)

```

builtins() # List all built-in functions
options() # Set options to control how R computes & displays results
?NA # Help page on handling of missing data values
abs(x) # The absolute value of "x"
append() # Add elements to a vector
c(x) # A generic function which combines its arguments
cat(x) # Prints the arguments
cbind() # Combine vectors by row/column (cf. "paste" in Unix)
diff(x) # Returns suitably lagged and iterated differences
gl() # Generate factors with the pattern of their levels
grep() # Pattern matching
identical() # Test if 2 objects are *exactly* equal
jitter() # Add a small amount of noise to a numeric vector
julian() # Return Julian date
length(x) # Return no. of elements in vector x
ls() # List objects in current environment
mat.or.vec() # Create a matrix or vector
paste(x) # Concatenate vectors after converting to character
range(x) # Returns the minimum and maximum of x
rep(1,5) # Repeat the number 1 five times
rev(x) # List the elements of "x" in reverse order
seq(1,10,0.4) # Generate a sequence (1 -> 10, spaced by 0.4)
sequence() # Create a vector of sequences
sign(x) # Returns the signs of the elements of x
sort(x) # Sort the vector x
order(x) # list sorted element numbers of x
tolower(), toupper() # Convert string to lower/upper case letters
unique(x) # Remove duplicate entries from vector
system("cmd") # Execute "cmd" in operating system (outside of R)
vector() # Produces a vector of given length and mode
formatC(x) # Format x using 'C' style formatting specifications
floor(x), ceiling(x), round(x), signif(x), trunc(x) # rounding functions
Sys.getenv(x) # Get the value of the environment variable "x"
Sys.putenv(x) # Set the value of the environment variable "x"
Sys.time() # Return system time
Sys.Date() # Return system date
getwd() # Return working directory
setwd() # Set working directory
?files # Help on low-level interface to file system
list.files() # List files in a give directory
file.info() # Get information about files

# Built-in constants:
pi,letters,LETTERS # Pi, lower & uppercase letters, e.g. letters[7] = "g"
month.abb,month.name # Abbreviated & full names for months

```

### 11.11.2 Mathe

Hier geklaut: [http://www.sr.bham.ac.uk/~ajrs/R/r-function\\_list.html](http://www.sr.bham.ac.uk/~ajrs/R/r-function_list.html)

```

log(x), logb(), log10(), log2(), exp(), expm1(), log1p(), sqrt()    # Fairly obvious
cos(), sin(), tan(), acos(), asin(), atan(), atan2()                 # Usual stuff
cosh(), sinh(), tanh(), acosh(), asinh(), atanh()                   # Hyperbolic functions
union(), intersect(), setdiff(), setequal()                          # Set operations
+, -, *, /, ^, %%, %%/%%                                         # Arithmetic operators
<, >, <=, >=, ==, !=                                           # Comparison operators
eigen()      # Computes eigenvalues and eigenvectors
deriv()      # Symbolic and algorithmic derivatives of simple expressions
integrate()  # Adaptive quadrature over a finite or infinite interval.
sqrt(), sum()
?Control      # Help on control flow statements (e.g. if, for, while)
?Extract      # Help on operators acting to extract or replace subsets of vectors
?Logic        # Help on logical operators
?Mod          # Help on functions which support complex arithmetic in R
?Paren        # Help on parentheses
?regex         # Help on regular expressions used in R
?Syntax       # Help on R syntax and giving the precedence of operators
?Special      # Help on special functions related to beta and gamma functions

```

### 11.11.3 Statistik

Hier geklaut: [http://www.sr.bham.ac.uk/~ajrs/R/r-function\\_list.html](http://www.sr.bham.ac.uk/~ajrs/R/r-function_list.html)

```

help(package=stats)    # List all stats functions
?Chisquare            # Help on chi-squared distribution functions
?Poisson               # Help on Poisson distribution functions
help(package=survival) # Survival analysis
cor.test()             # Perform correlation test
cumsum(); cumprod(); cummin(); cummax()   # Cumulative functions for vectors
density(x)             # Compute kernel density estimates
ks.test()               # Performs one or two sample Kolmogorov-Smirnov tests
loess(), lowess()       # Scatter plot smoothing
mad()                  # Calculate median absolute deviation
mean(x), weighted.mean(x), median(x), min(x), max(x), quantile(x)
rnorm(), runif()        # Generate random data with Gaussian/uniform distribution
splinefun()             # Perform spline interpolation
smooth.spline()         # Fits a cubic smoothing spline
sd()                    # Calculate standard deviation
summary(x)              # Returns a summary of x: mean, min, max etc.
t.test()                # Student's t-test
var()                   # Calculate variance
sample()                # Random samples & permutations
ecdf()                  # Empirical Cumulative Distribution Function
qqplot()                # quantile-quantile plot

```