R - Ein Einführungsskript

Lena Masch Kimon Kieslich Katharina Huseljić Marco Wähner
Johann-Sebastian Neef

November 2021

Inhaltsverzeichnis

Vorwort

Dieses Skript begleitet die Veranstaltung *DATA - Computergestützte Datenanalyse* für Studierende der Sozialwissenschaften an der Heinrich-Heine-Universität (HHU) Düsseldorf und ist in fachübergreifender Kooperation der Autoren zu dieser Veranstaltung entstanden. Dennoch soll das Skript nicht nur für Studierende der HHU, sondern gleichermaßen auch von allen interessierten Studierenden und R-Lernenden genutzt werden können. Im Folgenden wird kurz auf die Arbeit mit diesem Skript und die Verfügbarkeit der dafür notwendigen Datensätze eingegagen.

Dieses Skript bezieht sich neben ein paar kleineren Datensätzen zur Veranschaulichung vor allem auf die Daten des ALLBUS 2018. Diese Daten können frei über die GESIS bezogen werden, sofern sie für Forschung und Lehre genutzt werden. Dies umfasst auch empirische Forschungsvorhaben im Rahmen von Haus- und Abschlussarbeiten sowie Datenanalysen im Rahmen des Selbststudiums.

Darüber hinaus wird empfohlen, die Arbeit mit diesem Skript in einer strukturierten Ordnerumgebung auf dem Computer durchzuführen. Dafür kann beispielsweise ein eigener Ordner "R-Kurs" angelegt werden. In Unterordnern (z.B. "Daten") können dann die Datensätze abgelegt werden (s. Kapitel 2). Alternativ kann auch für jedes Kapitel ein eigener Ordner angelegt werden. Egal, für was sich entschieden wird: Strukturiertes Arbeiten ist für die Arbeit mit R unerlässlich.

Zusätzlich möchten wir darauf hinweisen, dass es sich beim vorliegenden Skript um ein anwendungsbezogenes Angebot handelt. Statistisches Grundwissen und Kenntnisse über Datenerhebung sind für die Arbeit mit sozialwissenschaftlichen Daten eminent wichtig. Wir präsentieren den interessierten Lesenden hier ausschließlich Möglichkeiten, wie sozialwissenschaftliche Datananalysen mit R umgesetzt werden.

Schließlich seien noch ein paar Worte zu den Zielen dieses Skriptes gesagt. Dieses Skript soll interessierten Lesenden die Möglichkeit geben, sich niederschwellig und praxisnah mit der statistischen Sprache `R` auseinanderzusetzen. `R` muss keine Gefahr für den Studienerfolg darstellen, sondern kann im Gegenteil viel Spaß bereiten. Wir hoffen, dass dieses Skript motivieren kann, sich mit den weitreichenden Möglichkeiten von R auseinanderzusetzen, und vielleicht dazu beiträgt, etwaige Ängste vor statistischer Datenanalyse abzubauen.

Düsseldorf, im November 2021

Lena Masch, Kimon Kieslich, Katharina Huseljić, Marco Wähner & Johann-Sebastian Neef

Zitierweise: Masch, L., Kieslich, K., Huseljić, K., Wähner, M., & Neef, J.-S. (2021). R - Ein Einführungsskript. Heinrich Heine University Düsseldorf. DOI: 10.24337/00001.

1 Einführung in R

1.1 Warum R?

In der empirisch-quantitativen Sozialforschung wurde lange Zeit mit SPSS oder Stata als Statistik-Software gearbeitet. Mittlerweile hat sich jedoch auch R, als de facto Programmiersprache, zur Datenanalyse in den Sozialwissenschaften etabliert. Einige Argumente sprechen für die Nutzung von R:

- R ist Open Source und daher als kostenfreie Software für alle zugänglich.
- R läuft auf verschiedenen Betriebssystemen, sowohl auf Windows als auch auf Mac Os und Linux.
- R hat eine große und sehr aktive Community. Die Community stellt vielfältige Lernressourcen kostenlos zur Verfügung und hilft bei Problemen in Online-Foren.
- R wird aktiv weiterentwickelt. Neue R-Versionen können in regelmäßigen Abständen heruntergeladen werden. Besonders wichtig sind dabei auch sogenannte "Packages", die oftmals die Durchführung bestimmter Analysen erleichtern oder die Anwendung neuer statistischer Verfahren ermöglichen.

1.2 Darstellung

Zunächst wird auf einige Konventionen der Darstellung für dieses Skript hingewiesen. Im vorliegenden Skript wird der Code in sogenannten "Code-Chunks" (engl. code chunks) angezeigt. In den Chunks stehen Kommentare, Befehle (Funktionen) sowie Ergebnisse. Kommentare in den Code-Chunks werden mit einer Raute `#` angezeigt. Das Ergebnis des Programms wird in diesem Skript jeweils mit zwei vorangestellten Rauten `##` angezeigt.

```
# Hier ist ein Code-Chunk mit Code und Ergebnis

1 + 1
## [1] 2
```

Dieses kurze Beispiel lässt sich also folgendermaßen interpretieren: R führt den Code `1 + 1` mit dem Ergebnis `2` aus.

1.3 Installation

1.3.1 Installation von R

Bevor die empirische Datenanalyse beginnen kann, muss R zunächst installiert werden. Auf der Website des Cran-Projekts („The Comprehensive R Archive Network") kann R heruntergeladen werden (https://cran.uni-muenster.de/). Unter *Download and Install R* wird das entsprechende Betriebssystem (Windows, Mac oder Linux) ausgewählt. Nun wird dem Hinweis *install R for the first time* gefolgt. Anschließend wird die aktuelle R-Version zum Download ausgewählt. Wenn die

Datei heruntergeladen wurde, wird das Programm mit einem Doppelklick gestartet und installiert. Die Standardeinstellungen können hierbei beibehalten werden. Nach der Installation kann R über die R-GUI verwendet werden.

1.3.2 R Konsole

R-GUI (*graphical user interface*) ist eine grafische Benutzeroberfläche und ermöglicht eine Schnittstelle zwischen dem Computer und R, um Befehle (oder Code) auszuführen.

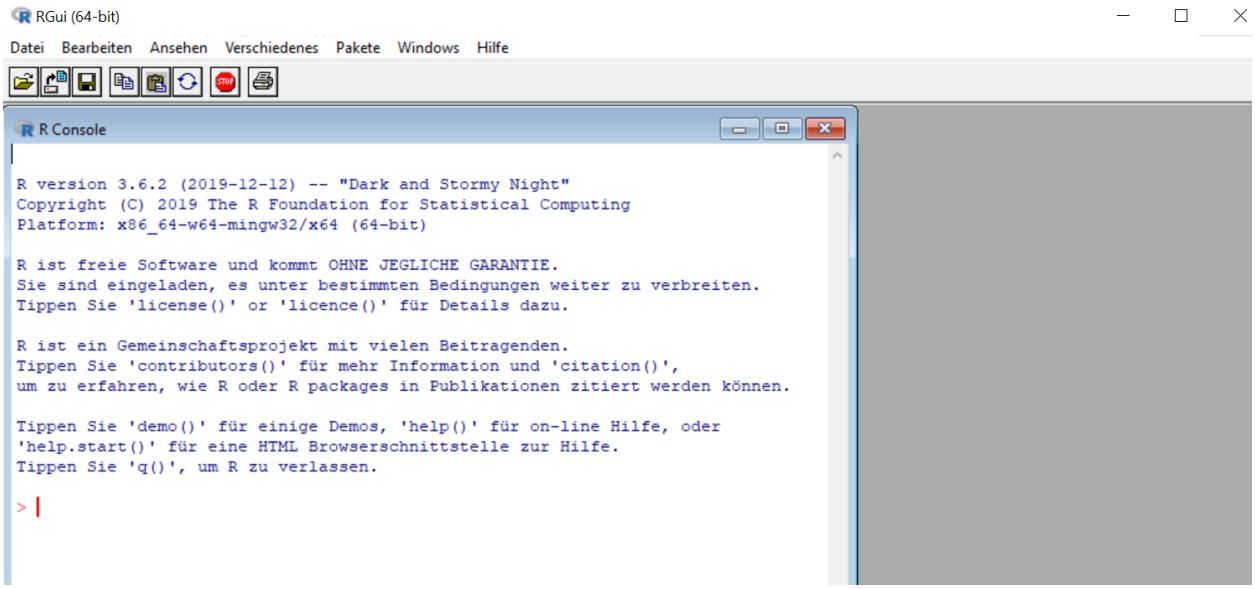


Abbildung 1: Beispiel für R-GUI

In der Konsole (engl. console) der R-GUI können die Befehle für eine Analyse direkt geschrieben und ausgeführt werden. Um einen ersten Eindruck von R zu bekommen, wird an eine in der Informatik gängige Programmier-Tradition angeknüpft und ein Programm mit dem Ergebnis "Hello World!" geschrieben. Das Programm kann zwar nicht sonderlich viel, es gibt aber immerhin eine kurze Begrüßung aus. Dafür wird die `print()` Funktion genutzt und mit `Enter` ausgeführt:

```
print("Hello World!")
## [1] "Hello World!"
```

1.3.3 Installation von RStudio

R kann auf diese Weise benutzt werden, allerdings bietet es sich an, ein weiteres Programm für eine einfache und komfortable Nutzung zu installieren: RStudio. RStudio ist eine IDE (*integrated development environment*) und grafische Benutzeroberfläche, das heißt, eine spezifische Entwicklungs- und Nutzungsumgebung für R. Es gibt zwar noch weitere IDEs und GUIs für R, jedoch hat sich RStudio über das letzte Jahrzehnt als die führende und mit Abstand am häufigsten genutzte Umgebung entwickelt. In der grundständigen Nutzung bietet RStudio einen integrierten Text-Editor zum Verfassen der Befehle (Code oder Sytnax). Auf diese Weise erleichtert es das Schreiben oder Speichern der Befehle, bevor diese über die Konsole ausgeführt werden. Außerdem stellt RStudio

einige hilfreiche Anwendungen zum Einstieg und fortgeschrittenen Nutzung zur Verfügung. RStudio kann ebenfalls kostenlos zur nicht kommerziellen Nutzung über folgenden Link heruntergeladen werden: https://rstudio.com/products/rstudio/download/ .

Dazu wird *RStudio Desktop* und die *Free-License* ausgewählt. Im Anschluss wird die Version für das entsprechende Betriebssystem (Windows, Mac oder Linux) heruntergeladen. Nach dem Download wird das Programm gestartet und installiert.

1.4 RStudio

Nach erfolgreicher Installation und Start, öffnet sich RStudio mit einem Programmfenster, das die Oberfläche in drei Bereiche einteilt: Arbeitsumfeld (workspace), Felder (panes) und Konsole (console). Diese Einteilung ist unter allen Betriebssystemen in der Standardeinstellung ähnlich und kann nach eigenen Vorlieben angepasst werden.

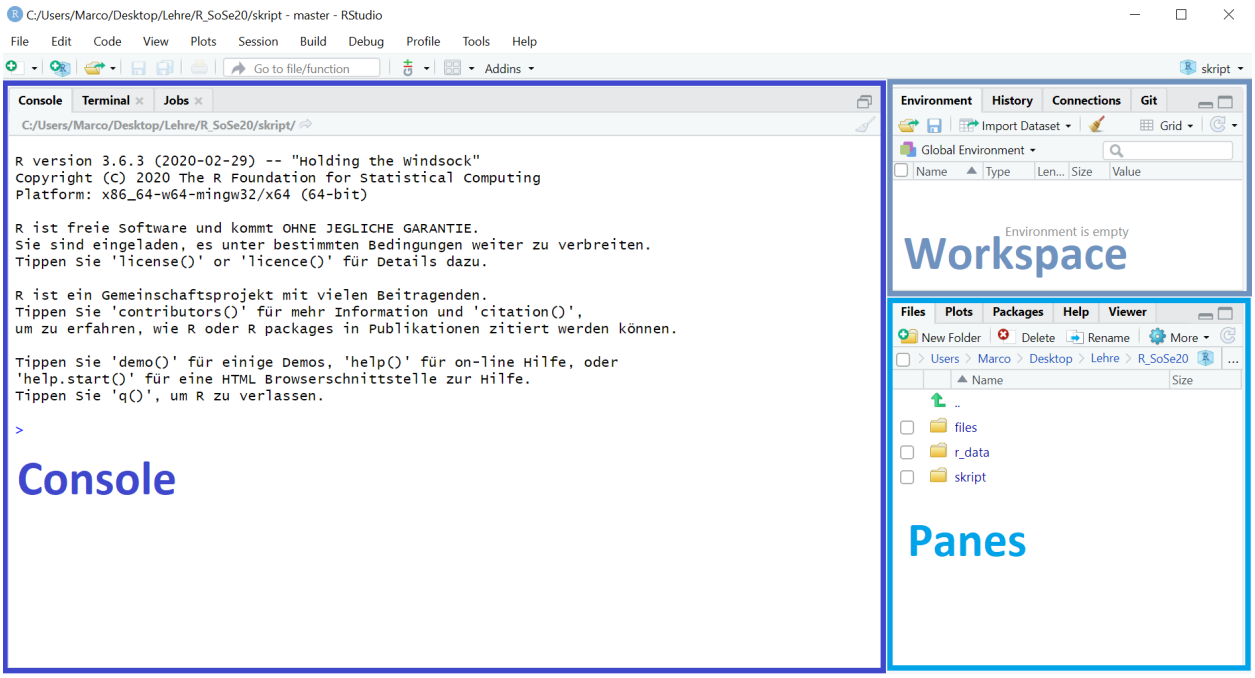


Abbildung 2: Startfenster RStudio

Workspace: Im Arbeitsumfeld ist insbesondere das *Environment* wichtig. In diesem Tab werden alle Objekte der Umgebung angezeigt, d.h. hier werden alle Datensätze und Variablen angezeigt, die sich im Arbeitsumfeld befinden.

Panes: "Felder" ist an dieser Stelle ein Sammelbegriff und bezeichnet verschiedene Tabs, von denen zunächst nur einige relevant sind: Unter *Files* werden beispielsweise die Dateien des Arbeitsverzeichnisses angezeigt. In *Plots* werden Grafiken (z.B. Diagramme) angezeigt und unter *Help* findet sich das Hilfefenster, das aufgerufen werden kann, um die Befehlsstruktur der Befehle anzuzeigen.

Console: Ergebnisse einer Analyse werden in der Konsole ausgegeben. Darüber hinaus ähnelt die Konsole der R-GUI. Auch in dieser Konsole können Befehle geschrieben und mit `Enter` direkt ausgeführt werden.

1.4.1 Skripte

Umfassende empirische Analysen (z.B. für Haus- und Abschlussarbeiten) erfordern in der Regel eine Reihe von statistischen Befehlen zur Datenaufbereitung und Datenanalyse. Eine solche Aneinanderreihung von Befehlen (auch Code oder Syntax) wird am besten im integrierten Text-Editor geschrieben: den sogenannten Skripten (engl. script). Ein solches Skript muss allerdings zu Beginn manuell geöffnet werden. Ein Skript öffnet sich als neues Fenster (links oben über der Konsole) über `File -> New File -> R Script`.

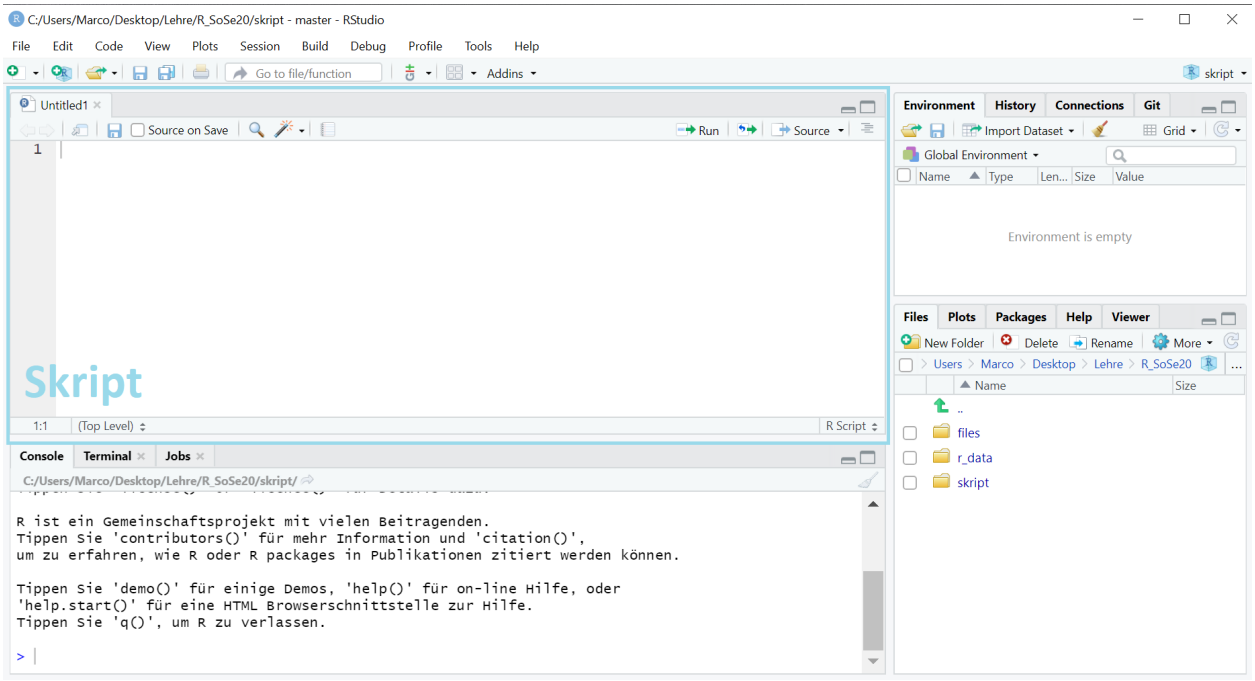


Abbildung 3: R Skript

Beim Schreiben des Codes in R-Skripten muss beachtet werden, dass der Code im Skript nicht automatisch durch das Drücken der `Enter-Taste` ausgeführt wird (wie es in der Konsole der Fall ist). Das Ausführen kann auf verschiedene Weisen geschehen. Für eine einzelne Zeile muss der Cursor zunächst in die entsprechende Zeile im Code gesetzt werden. Mit einem Klick auf *Run* wird der Code ausgeführt. Ebenso können mehrere Zeilen Code mit der Maus markiert werden und mit einem Klick auf *Run* ausgeführt werden. Das Ergebnis wird jeweils in der Konsole angezeigt. Der Code lässt sich auch alternativ mit dem Shortcut `STRG + ENTER` ausführen, nachdem er markiert wurde. Eine Übersicht zu allen Keyboard-Shortcuts ist unter `Tools -> Keyboard Shortcuts Help` aufgeführt.

Im Folgenden werden nun einige beispielhafte Anwendungen aufgezeigt, bevor die üblichen Schritte der Datenanalyse vertieft werden. R kann beispielsweise als Taschenrechner verwendet werden. Dies zeigt der folgende Code-Chunk.

```
# Eingabe im Skript
0.15 * 35.70
## [1] 5.355
```

R kann als Taschenrechner genutzt werden, um verschiedene Berechnungen durchzuführen. Es gibt eine Vielzahl an Operatoren. Dazu zählen:

- Addition `+`
- Subtraktion `-`
- Multiplikation `*`
- Division `/`
- Potenzieren `**`
- Wurzel `sqrt()`

Da R als Programmiersprache für die statistische Datenanalyse optimiert wurde, sind diese Operatoren und mathematische Regeln wie "Punkt vor Strich" implementiert. Dies zeigt das Ergebnis der Eingabe von `2 + 0.15 * 35.70` im nächsten Code-Chunk.

```
2 + 0.15 * 35.70
## [1] 7.355

4**2
## [1] 16

sqrt(16)
## [1] 4
```

An dieser Stelle sei auf einen häufigen Fehler beim Einstieg in R hingewiesen: Es kann vorkommen, dass R scheinbar nicht reagiert bzw. keine Ausgabe angezeigt wird. Das liegt oftmals daran, dass R auf einen weiteren Input wartet, da ein Befehl zuvor nicht abgeschlossen wurde. Meistens liegt es an kleinen Ungenauigkeiten im Code, z.B. daran, dass eine Klammer vergessen wurde oder eine Rechenoperation nicht zu Ende definiert wurde. Hier ein Beispiel:

```
# 3 +
```

Im obigen Beispiel erwartet R einen weiteren Input damit eine Berechnung ausgeführt wird. In der Konsole (links unten) wird dann statt eines `>` Symbols zu Anfang der Zeile ein `+` angezeigt. Dies verdeutlicht, dass in diesem Falle die Rechenoperation noch nicht abgeschlossen ist. Möchte man die Eingabe abbrechen, kann in der Konsole die Taste `Esc` gedrückt werden. Gerade bei längeren Code-Eingaben kann das Abbrechen des Ausführens einer Funktion durchaus sinnvoll sein, um im Anschluss den Fehler im Code zu suchen und vor der erneuten Ausführung zu korrigieren.

1.4.2 Funktionen

Der obige Code-Chunk zeigt neben den Rechenoperationen auch eine Funktion, bzw. wird eine Rechenoperation über eine Funktion ausgeführt. Die Quadratwurzel wird in diesem Fall über eine Funktion mit dem Namen `sqrt` erzeugt. In diesem Fall benötigt die Funktion mindestens eine Zahl, deren Quadratwurzel gezogen werden kann. Weiter oben wurde ebenfalls bereits eine Funktion genutzt: `print()`. Durch diese Funktion wurde der Inhalt der Klammern wiedergegeben.

Funktionen sind immer gleich aufgebaut: Auf einen Funktionsnamen wie `sqrt` folgt eine geschlossene Klammer `()`. Viele bereits in R implementierte Funktionen (z.B. aus base R) besitzen in der Regel mindestens ein oder mehrere Argumente. Argumente sind die notwendigen Inhalte, um eine Funktion auszuführen. Die Argumente werden in die Klammern geschrieben. Zu den Argumenten zählen auch

die Objekte, auf die sich die Funktion bezieht. In vielen Fällen ist mindestens ein Objekt erforderlich, um die Funktion auszuführen. Funktionen können allgemein als Befehle oder Handlungsanweisungen verstanden werden.

1.4.3 Objekte

Neben den Funktionen sind Zuweisungen (engl. assignment) ein wichtiges grundlegendes Prinzip zur Arbeit mit R. Durch Zuweisungen entstehen neue Objekte in R, die im Arbeitsumfeld gespeichert werden und auf die in einer Sitzung fortwährend zugegriffen werden kann. Objekte können in R viele verschiedene Formen annehmen. Technisch gesehen sind auch Funktionen Objekte in R. Für weitere Details empfehlen sich Einführungen zum Thema "Programmieren in R" (z.B. Grolemund 2014). Aus Sicht der sozialwissenschaftlichen Datenanalyse gehören zu den Objekten auch gängige Datenstrukturen wie z.B. Datensätze oder Variablen.

Eine Zuweisung wird im nächsten Code-Chunk veranschaulicht. Mit dem `<-` *assignment*-Zeichen, das aus einem „kleiner als"-Zeichen und einem „Minus"-Zeichen besteht, können Werte einem Vektor zugewiesen werden.

```
# Zuweisung
a <- 5

b <- 10

ab <- a * b

print(ab)
## [1] 50
```

R ermöglicht es, mit bereits erstellten Objekten weiter zu arbeiten. So wird der Vektor `ab` durch die Multiplikation der zuvor erstellten Vektoren `a` und `b` erstellt. Die Werte der Vektoren (`a, b, ab`) lassen sich jederzeit verändern und überschreiben.

Im nächsten Code-Chunk werden die Vektoren durch neue Zuweisungen überschrieben. Nach Ausführung der Zuweisungen wird der Unterschied sichtbar.

```
a <- 10

b <- 25

ab <- a * b

print(ab)
## [1] 250
```

Alle Objekte des Arbeitsumfeldes können über die Eingabe des Befehls `ls()` aufgelistet werden. Sie können mit mit dem Befehl `rm()` entfernt werden (remove). Alle Objekte aus dem Arbeitsumfeld können durch den Befehl `rm(list = ls())` entfernt werden. Alternativ können in RStudio mit dem "Besen" (Symbol im Arbeitsumfeld) Objekte aus dem Arbeitsumfeld entfernt werden.

```r
ls() #Auflistung der Objekte
## [1] "a"        "ab"       "b"        "img_path"
rm(a) # Entfernen des Objekts "a"
rm(list = ls()) # Entfernen aller Objekte
```

1.4.4 Kommentare

Um die statistische Datenanalyse transparent und nachvollziehbar zu gestalten, wird Code stets auskommentiert. Dies geschieht durch sogenannte Kommentare. Kommentare sollten kurz und prägnant sein und müssen stets mit einer Raute `#` beginnen, damit sie als Kommentare und nicht als Objekte gesucht (und in der Regel nicht gefunden) werden.

```r
# Wert der Variable a
a <- 10

# Wert der Variable b
b <- 25

# Ergebnis
ab <- a * b

# Output
print(ab)
## [1] 250
```

1.4.5 Speichern

Zum Speichern eines Skripts wird auf `Files -> Save As` geklickt. Anschließend muss ein Dateiname für das Skript und ein entsprechender Speicherort ausgewählt werden. Ein Skript hat die Dateiendung `.R`. Dateien mit dieser Dateiendung können ebenfalls als Textdateien von Text-Editoren (z.B. Notepad, Notepad++) geöffnet werden.

1.5 Packages

Ein großer Vorteil von R sind zusätzliche Pakete (engl. packages), die von zahlreichen Forschenden auf der Welt entwickelt werden und die grundlegenden Funktionen von R erweitern. Pakete stellen in der Regel ein Bündel an zusätzlichen Funktionen für ein spezifisches Anwendungsgebiet zur Verfügung. Eine Vielzahl an Funktionen zur statistischen Analyse sind bereits standardmäßig in R implementiert. Die `sqrt()`-Funktion (und viele weitere) sind dem sogenannten *Base R* zuzuordnen und stehen von Beginn an zur Verfügung (diese Funktionen müssen also nicht zusätzlich installiert werden).

Falls Packages genutzt werden sollen, müssen diese heruntergeladen und installiert werden. Um auf ein Package zuzugreifen, werden zwei Funktionen, nämlich `install.packages()` und `library()` benötigt:

- `install.packages()`: Zunächst wird das Package mit der `install.packages()`-Funktion aus dem Internet heruntergeladen. In der Klammer steht in Anführungszeichen der Name des Packages. Dieser Befehl muss nur einmal ausgeführt werden. Erst im Anschluss an ein neues R-Update müssen Pakete erneut installiert werden.
- `library()`: Nach dem Download muss das Package mit `library()` in die aktuelle Session geladen werden. Erneut wird der Name als Argument, allerdings diesmal ohne Anführungszeichen, eingefügt. Wichtig: Jedes Mal, wenn RStudio neu geöffnet wird, muss auch das benötigte Package über diesen Befehl in die aktuelle Session geladen werden, um auf die Funktionen aus dem Paket zugreifen zu können. Und eine Anmerkung: Als Alternative zum Befehl `library()` wird in manchen Skripten der Befehl `require()` verwendet. Auf den feinen Unterschied zwischen den beiden Befehlen wird an dieser Stelle nicht weiter eingegangen, sondern allgemein die Nutzung des Befehls `library()` empfohlen, wenn es darum geht, Pakete nach dem Installieren in das Arbeitsumfeld zu laden.

```
# Installation
install.packages("tidyverse")
# Package in Skript laden
library(tidyverse)
```

Eine Übersicht über die geladenen Pakete kann mit der Eingabe des Befehls `sessionInfo()` erhalten werden.

1.6 Hilfe

Mit `?` lässt sich die RStudio interne Hilfe aufrufen. Die Hilfe öffnet sich im unteren, rechten Feld unter "Help". Die Hilfe in RStudio ist immer ähnlich aufgebaut u.a. mit einer Beschreibung (engl.description) der Funktion, wie die Funktion verwendet wird (usage), welche Argumente benötigt werden (arguments) und einem kurzen Beispiel (examples):

```
# Hilfefunktion
?sqrt
?tidyverse
```

1.7 wissenschaftliche Notationen

R benutzt eine wissenschaftliche Notation. Dies wird in den späteren Kapiteln der statistischen Analysen deutlich. R gibt bis zu 16 Dezimalstellen aus, die allerdings nicht standardmäßig ausgeschrieben sind, sondern durch eine Exponentialschreibweise dargestellt werden. Wer die Exponentialschreibweise leicht irritierend findet, kann sie durch einen einfachen Befehl zu Beginn einer jeden Sitzung ausstellen. Der entsprechende Befehl lautet wie folgt: `options(scipen = 999)`.

2 Datenimport

Die sozialwissenschaftliche Datenanalyse beschäftigt sich oftmals mit individuellen Einstellungen und Verhalten - dafür werden entsprechende Daten benötigt.

Im folgenden Kapitel wird daher aufgeführt,

(a) wie der passende (frei verfügbare) Datensatz gefunden wird,

(b) Datensätze unterschiedlicher Formate in R eingelesen werden,

(c) ein erster Überblick über die Daten gewonnen wird.

2.1 Sozialwissenschaftliche Daten: Umfragen und Co.

In diesem Skript wird es hauptsächlich um die Analyse von Umfragen als Sekundärdaten gehen. Für die Suche nach Daten zu bestimmten Fragestellungen ist es im deutschsprachigen Raum u.a. möglich den GESIS-Datenbestandskatalog zu bemühen, um sich die Suche etwas zu erleichtern. Dort werden viele Datensätze zu unterschiedlichen Themen gesammelt.

Zu den gängigsten Datensätzen zählen:

- ALLBUS: Ein häufig genutzter Datensatz in den Sozialwissenschaften ist die Allgemeine Bevölkerungsumfrage der Sozialwissenschaften, kurz: ALLBUS. Auf der verlinkten Website gibt es nützliche Informationen zum Datensatz, wie beispielsweise die Themen, die in der jeweiligen Erhebung erfasst wurden sowie zum methodischen Vorgehen und der Stichprobenziehung. So wird z.B. erläutert, welcher Personenkreis die Grundgesamtheit für den ALLBUS 2018 darstellt. In diesem Falle sind das die in Deutschland in Privathaushalten lebenden Personen zum Zeitpunkt der Erhebung, die vor 2000 geboren wurden. Zudem finden sich Informationen, unter welchen Umständen mit systematischen Ausfällen zu rechnen ist (bspw. bei denjenigen, die sprachliche Schwierigkeiten mit dem Fragebogen hatten). Neben weiteren Informationen zu Ausschöpfungsquoten und Erhebungsverfahren findet sich dort auch der Fragebogen und Variablenreport.

- GLES: Es gibt auch Datensätze, die sich mit spezifischen Phänomenen auseinander setzen. Ein Beispiel ist die German Longitudinal Election Study (GLES), die verschiedene Umfragen zu den Themenfeldern der politischen Partizipation und Wahlforschung durchführt. Die GLES hat u.a. Vor- und Nachwahlstudien zu den Bundestagswahlen 2009, 2013, 2017 und 2021 erhoben. Auch hier sind detaillierte Informationen auf der Website der GESIS abrufbar.

- ISSP/ESS: Für internationale Vergleiche lohnt es sich einen Blick in das International Social Survey Programme (ISSP) oder den European Social Survey (ESS) zu werfen. Die Fragen des ISSP werden oftmals mit dem ALLBUS erhoben und auch in anderen Ländern auf ähnliche Weise erhoben. Der ESS ist eine eigenständige Umfrage, die seit 2002 im Zweijahresrhythmus in europäischen Ländern durchgeführt wird. So beinhaltet der ESS 2018 Daten aus 27 unterschiedlichen Ländern und eignet sich u.a. für Mehrebenenanalysen. Eine Auseinandersetzung mit den Erhebungsmethoden in den jeweiligen Ländern ist allerdings Voraussetzung für die Arbeit mit den Daten, da es stellenweise Unterschiede zwischen den Ländern (z.B. in der Stichprobenziehung und notwendigen Gewichtung) gibt.

- SOEP: Fortgeschrittene Studierende, die sich mit der Analyse von Paneldaten befassen, könnten zudem Freude daran haben, einen Blick ins sozio-ökonomische Panel (SEOP) zu werfen.

2.2 Datensätze in R importieren

2.2.1 Beispiel- und Testdaten

Im Repertoire von R gibt es einige kleine Datensätze, mit denen ohne Import gearbeitet werden kann. Einige davon sind bei der Online-Suche nach Fehlermeldungen und Hilfen in Beispiellösungen zu finden. Eine solche Online-Suche ist stets als einer der ersten Schritte zur Problemlösung empfehlenswert. Hierbei handelt es sich in der Regel allerdings nicht um Datensätze mit sozialwissenschaftlicher Ausrichtung. Eine Übersicht der integrierten Datensätze kann mit dem Befehl `data()` erfolgen. Als Beispieldatensatz ist dabei u.a. das *iris Dataset* beliebt. Mit dem `head()`-Befehl, kann der Aufbau des Datensatzes angezeigt werden. Dieser Befehl zeigt steht den "Kopf" oder Beginn des Datensatzes. Standardmäßig werden die ersten sechs Zeilen angezeigt, dies kann allerdings durch ein zusätzliches Argument geändert werden. Alternativ kann auch das Ende des Datensatzes über `tail()` betrachtet werden. Die Funktion `names()` zeigt die Variablen im Datensatz an. Die Funktion `summary()` kann auf einen Datensatz angewendet werden, wobei sie Lagemaße zu allen Variablen im Datensatz ausgibt.

```
head(iris) # Die ersten 6 Beobachtungen werden angezeigt.
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
tail(iris) # Die letzten 6 Beobachtungen werden angezeigt.
##     Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 145          6.7         3.3          5.7         2.5 virginica
## 146          6.7         3.0          5.2         2.3 virginica
## 147          6.3         2.5          5.0         1.9 virginica
## 148          6.5         3.0          5.2         2.0 virginica
## 149          6.2         3.4          5.4         2.3 virginica
## 150          5.9         3.0          5.1         1.8 virginica
names(iris) # Die Namen der Variablen im Datensatz.
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
summary(iris) # erste Lagemaße zu Variablen im Datensatz
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
```

```
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

Worum geht es in diesem Datensatz? Gemessen wurden Länge und Breite der Blätter von Schwertlilien, sowie die Spezies, zu der die entsprechende Schwertlilie gehört. Der Befehl `head()` gib die ersten Zeilen des Datensets aus. Je Zeile wurde im Beispielfall eine Pflanze ausgemessen und zugeordnet. In den Spalten stehen die gemessenen Variablen. So hat beispielsweise die erste Pflanze eine Sepallänge von 5,1 eine Sepalbreite von 3,5, eine Petallänge von 1,4 und eine Petalbreite von 0,2 und gehört zur Spezies Setosa. Weitere Informationen zu den integrierten Datensätzen können auch über die Hilfe gefunden werden, in diesem Falle durch die Eingabe von `?iris` in der Konsole.

> ***Information zu den Urhebern:*** An dieser Stelle soll auf den Ursprung des zunächst einmal unproblematisch wirkenden Datensatzes *iris* hingewiesen werden. Der Datensatz wurde im Jahr 1936 in einem Aufsatz zur linearen Diskriminanzanalyse von Ronald Fisher und Edgar Anderson in der Zeitschrift "Annals of Eugenics" veröffentlicht. Diese Zeitschrift war (wie viele andere Schriftstücke dieser Zeit) hochgradig menschenverachtend und beinhaltete viele rassistische und antisemitische Aufsätze von Fisher und Pearson.. Viele rassistische und antisemitische Aufsätze von Fisher und Pearson erschienen darin (Zum Überblick vgl. Joselson (2016)). Leider basieren viele Grundlagen der statistischen Analysen, die auch in diesem Kurs angesprochen werden, auf den Arbeiten von Fisher und Pearson. Diese Problematik wurde in der statistischen Datenanalyse viel zu lange ignoriert und so dient der iris-Datensatz nach wie vor häufig als Beispieldatensatz für Analysen. In Zukunft sollten alternative Datensätze verstärkt Anwendung finden. An dieser Stelle wird die Nutzung des Datensatzes beibehalten, um auf die Problematik aufmerksam zu machen und zu informieren.

Im nächsten Schritt soll zunächst ein *.RData* Datensatz (gapminder) in R eingelesen werden. *.RData* ist ein spezielles Datenformat in R, das für Datensätze genutzt wird, um diese zu speichern. R kann allerdings eine Vielzahl anderer gängiger Datenformate einlesen. Darunter fallen auch:

- .sav (SPSS-Format)
- .xlsx (Excel)
- .csv (comma-seperated-values)
- .txt (text file)

In der Regel müssen die unterschiedlichen Formate mit unterschiedlichen Befehlen importiert werden. Dies geschieht zum Teil mit `base R` und zum Teil mit zusätzlichen Paketen.

2.2.2 Festlegen des Arbeitsverzeichnisses

Für den Import von Daten muss im ersten Schritt ein Arbeitsverzeichnis (engl. working directory) festgelegt werden. Über `getwd()` lässt sich herausfinden, in welchem Arbeitsverzeichnis gearbeitet wird. Dieser Befehle zeigt das Arbeitsverzeichnis an.

```
## [1] "C:/Users/Kieslich/Desktop/R-Script"
```

Zurzeit ist es das Verzeichnis ein lokales Verzeichnis, das für das Verfassen des Skripts festgelegt wurde. Über den Befehl `setwd()` lässt sich das Arbeitsverzeichnis jederzeit ändern. An dieser Stelle kann auf verschiedene Weise vorgegangen werden. Entweder wird das Arbeitsverzeichnis auf den Ort, an dem sich der Datensatz befindet, geändert, oder der Datensatz wird in das aktuelle Arbeitsverzeichnis verschoben. Es ist allerdings unerlässlich zu wissen, in welchem Arbeitsverzeichnis der Datensatz liegt, mit dem gearbeitet werden soll. In diesem Falle soll also der Pfad des Ordners in R eingetragen werden, in dem der gapminder-Datensatz im .RData Format abgelegt wurde. Für weitere Informationen zum Datensatz kann die Online-Dokumentation `gapminder` aufgerufen werden. Es gibt ebenfalls ein R-Paket, das sich dem Datensatz widmet (`gapminder`) und über das der Datensatz verfügbar ist.

2.2.3 Exkurs R-Projekte

Neben dem manuellen Einstellen des Dateipfades mit dem Befehl `setwd()` ist es in R auch möglich mit sogenannten R-Projekten (engl. projects) zu arbeiten. R-Projekte haben den Vorteil, dass diese einen festen Ort bestimmen an dem die Daten, Skripte, Abbildungen usw. abgelegt werden und das Arbeitsverzeichnis direkt auf den Projektordner festlegt wird. So helfen R-Projects enorm bei der Strukturierung einzelner Arbeiten und auch einzelne Projekte auseinanderzuhalten. Außerdem muss nicht jedes Mal die Working-Directory neu eingestellt werden, sondern es kann einfach direkt das Projekt geöffnet werden. Wenn die Daten im R-Projekt liegen, können diese ohne weitere Spezifikation direkt eingelesen werden. R-Projects können in R-Studio über folgende Menüführung angelegt werden: File –> New Project. Anschließend muss über New oder Existing Directory der Ort angegeben werden, an dem das Projekt erstellt werden soll. R-Projects haben die Endung `.Rproj` und können mit einem Doppelklick geöffnet werden. Falls die Arbeitsumgebung gespeichert wurde, wird diese automatisch in der Environment angezeugt. Außerdem werden im Fenster rechts unten unter “Files” alle Daten angezeigt, die im entsprechenden R-Project Ordner verfügbar sind. Im Verlauf des Skriptes wird allerdings - aus Gründen der Nachvollziehbarkeit - weiterhin mit der manuellen Festlegung des Arbeitsverzeichnisses über `setwd()` gearbeitet.

2.2.4 .RData importieren

Nachdem das Arbeitsverzeichnis angepasst wurde, kann der gapminder Datensatz mit dem Befehl `load()`, in das Arbeitsumfeld geladen werden. Beim Arbeiten mit Arbeitsverzeichnissen auf Windows-Computern ist zu beachten, dass ein Backslash entweder durch zwei Backslashs oder durch einen Forward-Slash ersetzt werden muss. In MacOS und Linux sind keine Anpassungen notwendig. Im nächsten Code-Chunk wurde das Arbeitsverzeichnis des Datensatzes kopiert und jeder Backslash um einen weiteren Backslash erweitert, bevor der Datensatz geladen wurde.

```
# setwd(C:\\Users\\Documents\\DataKurs) für Windows
# setwd(C:/Users/Documents/DataKurs) für Windows
# setwd(/Users/Documents/DataKurs) für MacOS und Linux
# das Arbeitsverzeichnis ist mit setwd() individuell anzupassen

load("Daten/gapminder.RData") #import .RData-Format
```

Dass das Einladen des Datensatzes funktioniert hat, wird im globalen Arbeitsumfeld ("Global Environment") sichtbar, in dem der Datensatz nun auftaucht. Zusätzlich werden die Anzahl der Beobachtungen ("obs." von "observations") und die Anzahl der Variablen ("variables") des Datensatzes angezeigt.

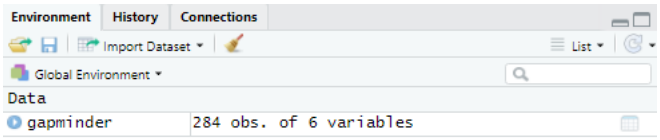
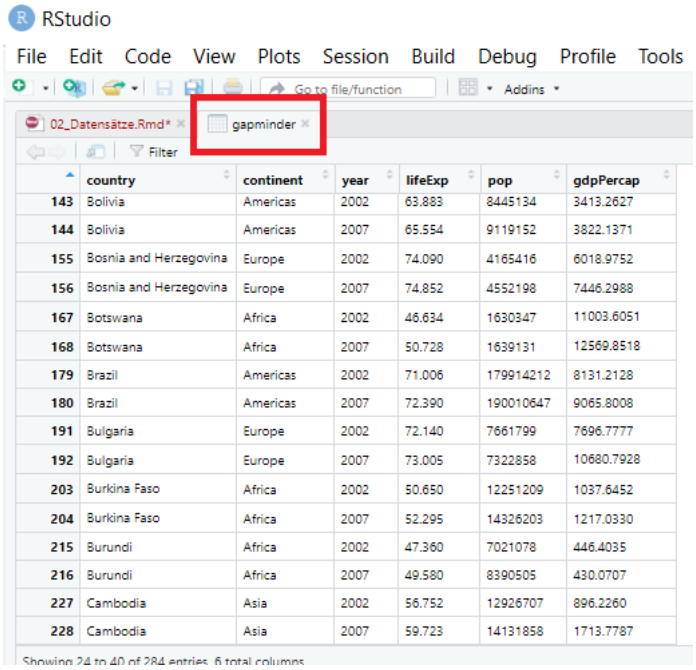


Abbildung 4: Datensatzes im Global Environment in R Studio

Hinweis: Der Datensatz wurde für eine bessere Übersicht gekürzt

Wenn der Datensatz im globalen Arbeitsumfeld angeklickt wird, wird er oben links in einem neuen Fenster neben dem Fenster, in dem der Code geschrieben wird, angezeigt. Ebenso kann der Datensatz über den Befehl `View()` angezeigt werden. Dieser Befehl empfiehlt sich allerdings nicht beim Arbeiten mit Big Data, für Datensätze der Umfrageforschung kann er in der Regel jedoch problemlos angewendet werden. Auf diese Weise kann die Datenmatrix inspiziert werden. Es ist jedoch nicht möglich, Änderungen in der Datenmatrix vorzunehmen.



	country	continent	year	lifeExp	pop	gdpPercap
143	Bolivia	Americas	2002	63.883	8445134	3413.2627
144	Bolivia	Americas	2007	65.554	9119152	3822.1371
155	Bosnia and Herzegovina	Europe	2002	74.090	4165416	6018.9752
156	Bosnia and Herzegovina	Europe	2007	74.852	4552198	7446.2988
167	Botswana	Africa	2002	46.634	1630347	11003.6051
168	Botswana	Africa	2007	50.728	1639131	12569.8518
179	Brazil	Americas	2002	71.006	179914212	8131.2128
180	Brazil	Americas	2007	72.390	190010647	9065.8008
191	Bulgaria	Europe	2002	72.140	7661799	7696.7777
192	Bulgaria	Europe	2007	73.005	7322858	10680.7928
203	Burkina Faso	Africa	2002	50.650	12251209	1037.6452
204	Burkina Faso	Africa	2007	52.295	14326203	1217.0330
215	Burundi	Africa	2002	47.360	7021078	446.4035
216	Burundi	Africa	2007	49.580	8390505	430.0707
227	Cambodia	Asia	2002	56.752	12926707	896.2260
228	Cambodia	Asia	2007	59.723	14131858	1713.7787

Showing 24 to 40 of 284 entries, 6 total columns

Abbildung 5: Datensätze in R Studio ansehen

Die Beobachtungsobjekte (Zeilen) tragen jeweils eine Ziffer als Identifikationsnummer (sog. "row names"). Die Variablen zeigen verschiedene Informationen. Von links nach rechts beginnend mit dem Land, um das es sich handelt, dem Kontinent, dem Jahr der Datenerfassung, der Lebenserwartung, Population und dem BIP pro Kopf.

Die Befehle `head()`, `nrow()` und `names()` können für einen ersten Überblick genutzt werden.

```
#erster Überblick über den Datensatz
head(gapminder, n=20)
##         country continent year lifeExp       pop  gdpPercap
## 11  Afghanistan      Asia 2002  42.129  25268405   726.7341
## 12  Afghanistan      Asia 2007  43.828  31889923   974.5803
## 23      Albania    Europe 2002  75.651   3508512  4604.2117
## 24      Albania    Europe 2007  76.423   3600523  5937.0295
## 35      Algeria    Africa 2002  70.994  31287142  5288.0404
## 36      Algeria    Africa 2007  72.301  33333216  6223.3675
## 47       Angola    Africa 2002  41.003  10866106  2773.2873
## 48       Angola    Africa 2007  42.731  12420476  4797.2313
## 59    Argentina  Americas 2002  74.340  38331121  8797.6407
## 60    Argentina  Americas 2007  75.320  40301927 12779.3796
## 71    Australia   Oceania 2002  80.370  19546792 30687.7547
## 72    Australia   Oceania 2007  81.235  20434176 34435.3674
## 83      Austria    Europe 2002  78.980   8148312 32417.6077
## 84      Austria    Europe 2007  79.829   8199783 36126.4927
## 95      Bahrain      Asia 2002  74.795    656397 23403.5593
## 96      Bahrain      Asia 2007  75.635    708573 29796.0483
## 107  Bangladesh      Asia 2002  62.013 135656790  1136.3904
## 108  Bangladesh      Asia 2007  64.062 150448339  1391.2538
## 119     Belgium    Europe 2002  78.320  10311970 30485.8838
## 120     Belgium    Europe 2007  79.441  10392226 33692.6051
nrow(gapminder)
## [1] 284
names(gapminder)
## [1] "country"   "continent" "year"      "lifeExp"   "pop"       "gdpPercap"
```

2.2.5 SPSS Datensatz (.sav) importieren

Für den Import von .sav-Dateien oder .dta-Dateien, die typischerweise in SPSS bzw. in Stata genutzt werden, muss ein zusätzliches Paket verwendet werden, z.B. das Paket `haven`.

Zu Beginn wird dafür also das entsprechende Paket installiert. Das funktioniert über `install.packages("Paketname")` und über `library(Paketname)`. Ein Package muss nur einmalig installiert werden. Vor jeder Benutzung in einer Session muss es jedoch mit `library()` neu aktiviert werden.

```
# install.packages("haven") # wird durch Kommentierung nicht erneut ausgeführt
library(haven) # Aktivierung des Pakets für die aktuelle Session
```

Nach dem Laden des Pakets `haven` kann der Datensatz über die Funktion `read_spss()` aus dem Paket eingelesen werden. Dazu muss ein neuer Name für das Objekt erdacht werden, dem der Datensatz dann zugewiesen wird. Prinzipiell ist es möglich, einen beliebigen Namen zu wählen. Es gibt allerdings ein paar grundlegende Regeln (siehe Kapitel 3. Zudem empfiehlt es sich einen möglichst beschreibenden Namen zu wählen. Da nun ein ALLBUS-Datensatz aus dem Jahr 2018 eingelesen werden soll, wird hier der Objektname 'allbus2018' genutzt. Über einen `<-` wird dem

eingelesenen Datensatz dieser Name zugewiesen. Durch die Zuweisung erscheint der Datensatz als Objekt im Arbeitsumfeld und kann für weitere Analysen genutzt werden.

Für ein erfolgreiches Einlesen muss auch an dieser Stelle das Arbeitsverzeichnis berücksichtigt werden. Da der ALLBUS-Datensatz so wie weitere Datensätze dieses Skripts im Unterordner *Daten* abgelegt sind, muss R mitgeteilt werden, in welchem Unterordner der Datensatz zu finden ist. Deshalb wird der Pfad mit `read_spss(Ünterordner/Datensatz.sav")` spezifiziert. Alternativ könnte der Dateipfad über `setwd()` auf den Unterordner festgelegt werden. Dies ist jedoch nicht nötig, wenn der Unterordner beim Einlesen der Daten festgelegt wird. Es ist ebenfalls notwendig die Dateiendung mitzuführen, die in diesem Falle die Eigenschaften des SPSS-Datensatzes trägt (".sav"). Zudem wird an dieser Stelle der Befehl `as.data.frame()` hinzugefügt. Dieser Befehl ist optional, allerdings sorgt er dafür, dass der SPSS-Datensatz als ein klassischer R-Datensatz (data.frame) in R behandelt wird. Ohne diesen Zusatz wird der Datensatz durch das Paket `haven` als ein `tibble` eingelesen. Ein `tibble` ist eine moderne Form eines Datensatzes, die in den Paketen des tidyverse Verwendung findet und sich allgemein nur in wenigen Punkten von herkömmlichen Datensätzen unterscheidet. Es unterscheidet sich die Darstellung in der Konsole. Bei sehr großen Datensätzen bietet sich die Nutzung von tibbles an. Zudem werden Zeichenketten, z.B. in kategoriellen Variablen des Datensatzes, nicht automatisch als Faktoren klassifiziert. Weitere Informationen zu tibbles finden sich in R for Data Science (Wickham & Grolemund, 2016, Kap. 10) oder in der `vignette("tibble")`.

```
allbus2018 <- as.data.frame(read_spss("Daten/ALLBUS2018.sav"))# import .sav-Format
```

Im Arbeitsumfeld erscheint nun der Datensatz ALLBUS2018. Hierbei handelt es sich zum ersten Kennenlernen der Daten um eine gekürzte Fassung. Der komplette Datensatz beinhaltet deutlich mehr Variablen und wird in späteren Kapiteln noch Verwendung finden.

Da der Datensatz keine Labels besitzt, muss im Codebuch nachgeschaut werden, um welche Variablen es sich handelt. Es können allerdings Funktionen wie `attributes` genutzt werden, um Informationen zu Eigenschaften der einzelnen Variablen im Datensatz zu erhalten. Dies erfolgt über die Aufrufung der Variable durch ein Dollar-Zeichen im jeweiligen Datensatz: `attributes(Datensatz$Variablenname)` (mehr dazu in Kapitel 3). Der gekürzte Datensatz enthält neben soziodemographischen Informationen Variablen zum politischen Vertrauen:

pt01	VERTRAUEN: GESUNDHEITSWESEN	69
pt02	VERTRAUEN: BUNDESVERFASSUNGSGERICHT	70
pt03	VERTRAUEN: BUNDESTAG	71
pt04	VERTRAUEN: STADT-,GEMEINDEVERWALTUNG	72
pt08	VERTRAUEN: JUSTIZ	73
pt09	VERTRAUEN: FERNSEHEN	74
pt10	VERTRAUEN: ZEITUNGSWESEN	75
pt11	VERTRAUEN: HOCHSCHULEN,UNIVERSITAETEN	76
pt12	VERTRAUEN: BUNDESREGIERUNG	77
pt14	VERTRAUEN: POLIZEI	78
pt15	VERTRAUEN: POLITISCHE PARTEIEN	79
pt19	VERTRAUEN: KOMMISSION DER EU	80
pt20	VERTRAUEN: EUROPAEISCHES PARLAMENT	81

Abbildung 6: ausgewählte Variablen im Allbus 2018

Über den Befehl `head()` können die ersten Beobachtungen ausgeben werden. Im Falle des ALLBUS entsprechen die Beobachtungen den befragten Individuen. Über `nrow()` lässt sich die An-

zahl der Beobachtungen und über `names()` die Variablenbezeichnungen ausgeben. Die Funktion `attributes()` zeigt in diesem Falle die Eigenschaften der Variable "pt01" im Datensatz. Bei kleinen Datensätzen empfiehlt sich zudem die Funktion `summary()`. Diese kann auf den Datensatz angewendet werden und gibt eine Übersicht der Verteilung der Variablen. Zudem kann sie auf einzelne Variablen angewendet werden.

```
#erster Überblick über die Daten
head(allbus2018, n = 10)
##    eastwest educ sex age work pt01 pt02 pt03 pt04 pt08 pt09 pt10 pt11 pt12 pt14
## 1         1    3   1  62    1    6    6    7    6    5    4    5    6    6    7
## 2         2    5   2  64    1    6    6    5    5    6    6    5    7    6    7
## 3         1    1   1  22    1    5    6    4    4    4    2    2    4    4    5
## 4         2    3   1  59    4    7    4    5    5    6    7    7    7    4    6
## 5         2    5   2  30    1    5    6    4    6    4    1    3    4    4    4
## 6         1    5   1  41    1    5    6    4    5    5    3    4    5    3    6
## 7         1    5   2  43    2    6    6    4    5    4    5    6    5    4    4
## 8         1    5   1  39    1    3    6    5    4    6    3    5    6    5    6
## 9         1    5   2  40    2    6    6    6    5    6    4   NA    6    5    5
## 10        2    5   2  69    4    5    5    5    5    4    5    5    6    4    5
##    pt15 pt19 pt20
## 1     5    5    5
## 2     4    5    5
## 3     3    4    3
## 4     4    4    4
## 5     3    1    1
## 6     3    4    4
## 7     4   NA   NA
## 8     4    5    6
## 9     5    5    5
## 10    4    4    4
nrow(allbus2018)
## [1] 3477
names(allbus2018)
##  [1] "eastwest" "educ"     "sex"      "age"      "work"     "pt01"
##  [7] "pt02"     "pt03"     "pt04"     "pt08"     "pt09"     "pt10"
## [13] "pt11"     "pt12"     "pt14"     "pt15"     "pt19"     "pt20"
attributes(allbus2018$pt01)
## $label
## [1] "VERTRAUEN: GESUNDHEITSWESEN"
##
## $format.spss
## [1] "F2.0"
##
## $class
## [1] "haven_labelled" "vctrs_vctr"     "double"
##
## $labels
##       KEINE ANGABE GAR KEIN VERTRAUEN                  ..                  ..
```

```
##                 -9                  1                  2                  3
##                 ..                 ..                 ..  GROSSES VERTRAUEN
##                  4                  5                  6                  7
summary(allbus2018)
##     eastwest          educ            sex             age              work
##  Min.   :1.000   Min.   :1.000   Min.   :1.00   Min.   :18.00   Min.   :1.000
##  1st Qu.:1.000   1st Qu.:3.000   1st Qu.:1.00   1st Qu.:37.00   1st Qu.:1.000
##  Median :1.000   Median :3.000   Median :1.00   Median :53.00   Median :2.000
##  Mean   :1.313   Mean   :3.486   Mean   :1.49   Mean   :51.68   Mean   :2.367
##  3rd Qu.:2.000   3rd Qu.:5.000   3rd Qu.:2.00   3rd Qu.:65.00   3rd Qu.:4.000
##  Max.   :2.000   Max.   :7.000   Max.   :2.00   Max.   :95.00   Max.   :4.000
##                  NA's   :3                      NA's   :5
##       pt01            pt02            pt03            pt04
##  Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000
##  1st Qu.:4.000   1st Qu.:4.000   1st Qu.:3.000   1st Qu.:4.000
##  Median :5.000   Median :5.000   Median :4.000   Median :5.000
##  Mean   :4.637   Mean   :5.091   Mean   :4.058   Mean   :4.612
##  3rd Qu.:6.000   3rd Qu.:6.000   3rd Qu.:5.000   3rd Qu.:6.000
##  Max.   :7.000   Max.   :7.000   Max.   :7.000   Max.   :7.000
##  NA's   :5       NA's   :236     NA's   :91      NA's   :57
##       pt08            pt09            pt10            pt11            pt12
##  Min.   :1.000   Min.   :1.00   Min.   :1.000   Min.   :1.000   Min.   :1.000
##  1st Qu.:4.000   1st Qu.:2.00   1st Qu.:3.000   1st Qu.:5.000   1st Qu.:3.000
##  Median :5.000   Median :3.00   Median :4.000   Median :5.000   Median :4.000
##  Mean   :4.548   Mean   :3.35   Mean   :3.796   Mean   :5.136   Mean   :3.983
##  3rd Qu.:6.000   3rd Qu.:4.00   3rd Qu.:5.000   3rd Qu.:6.000   3rd Qu.:5.000
##  Max.   :7.000   Max.   :7.00   Max.   :7.000   Max.   :7.000   Max.   :7.000
##  NA's   :55      NA's   :30     NA's   :118     NA's   :268     NA's   :43
##       pt14            pt15            pt19            pt20
##  Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000
##  1st Qu.:4.000   1st Qu.:3.000   1st Qu.:3.000   1st Qu.:3.000
##  Median :5.000   Median :4.000   Median :4.000   Median :4.000
##  Mean   :5.053   Mean   :3.463   Mean   :3.568   Mean   :3.595
##  3rd Qu.:6.000   3rd Qu.:4.000   3rd Qu.:5.000   3rd Qu.:5.000
##  Max.   :7.000   Max.   :7.000   Max.   :7.000   Max.   :7.000
##  NA's   :14      NA's   :96      NA's   :300     NA's   :257
```

2.2.6 STATA Datensatz (.dta) importieren

Nicht nur SPSS-Datensätze können mit dem Paket `haven` eingelesen werden. Auch Datensätze aus STATA können über einen ähnlichen Weg importiert werden. Dazu wird der `read_stata()`Befehl verwendet. Auch diese Variante des ALLBUS-Datensatzes liegt mit der Endung .dta im Unterordner "Daten" im Order "RProjekt". Obwohl der gewählte Datensatz zur SPSS-Variante identisch sein sollte, werden hier die Eigenschaften erneut kurz überprüft. So wird die Anzahl der Beobachtungen mit `head()` kontrolliert, um einen ersten Überblick zu erhalten.

```r
# import .dta-Format
allbus2018stata <- as.data.frame(read_stata("Daten/ALLBUS_Statafile.dta"))

# erster Überblick über die Daten
head(allbus2018stata, 10)
##    eastwest educ sex age work pt01 pt02 pt03 pt04 pt08 pt09 pt10 pt11 pt12 pt14
## 1         1    3   1  62    1    6    6    7    6    5    4    5    6    6    7
## 2         2    5   2  64    1    6    6    5    5    6    6    5    7    6    7
## 3         1    1   1  22    1    5    6    4    4    4    2    2    4    4    5
## 4         2    3   1  59    4    7    4    5    5    6    7    7    7    4    6
## 5         2    5   2  30    1    5    6    4    6    4    1    3    4    4    4
## 6         1    5   1  41    1    5    6    4    5    5    3    4    5    3    6
## 7         1    5   2  43    2    6    6    4    5    4    5    6    5    4    4
## 8         1    5   1  39    1    3    6    5    4    6    3    5    6    5    6
## 9         1    5   2  40    2    6    6    6    5    6    4   -9    6    5    5
## 10        2    5   2  69    4    5    5    5    5    4    5    5    6    4    5
##    pt15 pt19 pt20
## 1     5    5    5
## 2     4    5    5
## 3     3    4    3
## 4     4    4    4
## 5     3    1    1
## 6     3    4    4
## 7     4   -9   -9
## 8     4    5    6
## 9     5    5    5
## 10    4    4    4
nrow(allbus2018stata)
## [1] 3477
names(allbus2018stata)
##  [1] "eastwest" "educ"     "sex"      "age"      "work"     "pt01"
##  [7] "pt02"     "pt03"     "pt04"     "pt08"     "pt09"     "pt10"
## [13] "pt11"     "pt12"     "pt14"     "pt15"     "pt19"     "pt20"
attributes(allbus2018stata$pt01)
## $label
## [1] "VERTRAUEN: GESUNDHEITSWESEN"
##
## $format.stata
## [1] "%12.0g"
##
## $class
## [1] "haven_labelled" "vctrs_vctr"     "double"
##
## $labels
##       KEINE ANGABE GAR KEIN VERTRAUEN                 ..                 ..
##                 -9                  1                  2                  3
##                 ..                 ..                 ..  GROSSES VERTRAUEN
```

```
##                  4                  5                  6                  7
summary(allbus2018stata)
##     eastwest          educ             sex             age
##  Min.   :1.000   Min.   :-9.000   Min.   :1.00   Min.   :-32.00
##  1st Qu.:1.000   1st Qu.: 3.000   1st Qu.:1.00   1st Qu.: 37.00
##  Median :1.000   Median : 3.000   Median :1.00   Median : 52.00
##  Mean   :1.313   Mean   : 3.476   Mean   :1.49   Mean   : 51.56
##  3rd Qu.:2.000   3rd Qu.: 5.000   3rd Qu.:2.00   3rd Qu.: 65.00
##  Max.   :2.000   Max.   : 7.000   Max.   :2.00   Max.   : 95.00
##       work            pt01             pt02             pt03
##  Min.   :1.000   Min.   :-9.000   Min.   :-9.000   Min.   :-9.000
##  1st Qu.:1.000   1st Qu.: 4.000   1st Qu.: 4.000   1st Qu.: 3.000
##  Median :2.000   Median : 5.000   Median : 5.000   Median : 4.000
##  Mean   :2.367   Mean   : 4.617   Mean   : 4.135   Mean   : 3.716
##  3rd Qu.:4.000   3rd Qu.: 6.000   3rd Qu.: 6.000   3rd Qu.: 5.000
##  Max.   :4.000   Max.   : 7.000   Max.   : 7.000   Max.   : 7.000
##       pt04             pt08             pt09             pt10
##  Min.   :-9.000   Min.   :-9.000   Min.   :-9.000   Min.   :-9.000
##  1st Qu.: 4.000   1st Qu.: 4.000   1st Qu.: 2.000   1st Qu.: 3.000
##  Median : 5.000   Median : 5.000   Median : 3.000   Median : 4.000
##  Mean   : 4.389   Mean   : 4.333   Mean   : 3.243   Mean   : 3.362
##  3rd Qu.: 6.000   3rd Qu.: 6.000   3rd Qu.: 4.000   3rd Qu.: 5.000
##  Max.   : 7.000   Max.   : 7.000   Max.   : 7.000   Max.   : 7.000
##       pt11             pt12             pt14             pt15
##  Min.   :-9.000   Min.   :-9.000   Min.   :-9.000   Min.   :-9.000
##  1st Qu.: 4.000   1st Qu.: 3.000   1st Qu.: 4.000   1st Qu.: 3.000
##  Median : 5.000   Median : 4.000   Median : 5.000   Median : 4.000
##  Mean   : 4.046   Mean   : 3.823   Mean   : 4.996   Mean   : 3.119
##  3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 6.000   3rd Qu.: 4.000
##  Max.   : 7.000   Max.   : 7.000   Max.   : 7.000   Max.   : 7.000
##       pt19             pt20
##  Min.   :-9.000   Min.   :-9.000
##  1st Qu.: 2.000   1st Qu.: 2.000
##  Median : 4.000   Median : 4.000
##  Mean   : 2.483   Mean   : 2.664
##  3rd Qu.: 4.000   3rd Qu.: 5.000
##  Max.   : 7.000   Max.   : 7.000
```

2.2.7 CSV Datensatz (.csv) importieren

Frei verfügbare Daten (Open Data) sind eine interessante und leicht zugängliche Datenressource für Makroindikatoren auf kommunaler, regionaler, nationaler oder gar europäischer Ebene. So haben viele Städte ein Open-Data-Portal, z.B. auch die Stadt Düsseldorf. In solchen Portalen stehen aggregierte Daten der Bevölkerung zur Verfügung. Solche Daten werden häufig im csv-Format bereitgestellt. In diesem Falle wurde bereits eine Datei mit dem Namen `Stadt.csv` angelegt. In dieser Datei sind Informationen zur Größe aller Mittel- und Großstädte in Deutschland gespeichert sowie eine kategorielle Zuordnung (Großstadt vs. mittelgroße Stadt). Die Funktion `read.csv()`

kann genutzt werden, um die Daten einzulesen. Hierbei kommen neben dem Dateinpfad, zwei weitere Argumente der Funktion zum Tragen. Zum fehlerfreien Einlesen sollte über die Argumenten "sep=" und "encoding=" angegeben, durch welches Symbol die Beobachtungen getrennt sind ("sep" = separator) und in welches Encoding-Format vorliegt. Zudem kann über das Argument "header" angegeben werden, ob Variablennamen in der ersten Zeile angezeigt werden. Die jeweiligen Argumente werden innerhalb der Funktion mit einem , getrennt. In diesem Fall liegt der Datensatz erneut im Unterordner "Daten", die Beobachtungen sind durch Semikolons getrennt und das Encoding wird auf "UTF-8" festgelegt. "UTF-8" ist ein gängiges Format und ermöglicht die korrekte Anzeige von Umlauten.

```
#import .csv-Format
stadt <- read.csv("Daten/Stadt.csv",
                  sep = ";",
                  encoding = "UTF-8")
```

2.2.8 Datensätze speichern

Über den `save()`-Befehl werden Datensätze gespeichert. Und auch hier muss an das Thema Arbeitsverzeichnis gedacht werden, um R mitzuteilen, an welchem Ort es die Daten speichern soll. Datensätze, die ursprünglich ein anderes Format hatten, können nun in .RData geändert und im Anschluss mit der `load()`-Funktion importiert werden.

```
# ohne Spezifizierung werden die Daten im aktuellen Arbeitsverzeichnis gespeichert
# (deshalb steht der Befehl als Kommentar):
# save(stadt, file = "Stadt.RData")

#Die Daten werden im Unterordner "Daten" gespeichert:
save(stadt, file = "Daten/Stadt.RData")
save(allbus2018, file = "Daten/allbus2018.RData")
```

Datensätze, die als .RData gespeichert werden, können ab sofort mit `load()` eingelesen werden.

```
load("Daten/allbus2018.RData")
```

Es sei angemerkt, dass es mit dem haven-Paket möglich ist, Daten auch als SPSS- oder Stata-Datensatz zu speichern. Dies könnte sinnvoll sein, falls der Datensatz Personen zugänglich gemacht werden soll, die auf die Nutzung dieser Programme angewiesen sind. Ebenso ist es möglich Daten als Excel-Dateien oder csv- und txt-Dateien zu speichern. Für die weitere Arbeit in R ist es jedoch am einfachsten und oftmals am platzsparendsten mit RData-Files zu arbeiten.

2.2.9 Exkurs: Daten importieren mit dem Rio-Package

Das Paket `rio` ermöglicht es, alle möglichen Formate mit einer Funktion in R zu öffnen. Im Paket "Rio" wurden unterschiedliche Funktionen generalisiert und in einer Funktion `import()` zusammengeführt. Es lohnt sich, die anderen Funktionen zwar im Kopf zu behalten, aber auch dieses Paket einmal über die verschiedenen Formate hinweg auszuprobieren.

```r
#install.packages(rio) # muss nur einmal ausgeführt werden
library(rio)
allbussav <- import("Daten/allbus2018.sav")
allbusdta <- import("Daten/ALLBUS_Statafile.dta")
stadt <- import("Daten/Stadt.csv", encoding = "UTF-8")

head(allbussav)
##   eastwest educ sex age work pt01 pt02 pt03 pt04 pt08 pt09 pt10 pt11 pt12 pt14
## 1        1    3   1  62    1    6    6    7    6    5    4    5    6    6    7
## 2        2    5   2  64    1    6    6    5    5    6    6    5    7    6    7
## 3        1    1   1  22    1    5    6    4    4    4    2    2    4    4    5
## 4        2    3   1  59    4    7    4    5    5    6    7    7    7    4    6
## 5        2    5   2  30    1    5    6    4    6    4    1    3    4    4    4
## 6        1    5   1  41    1    5    6    4    5    5    3    4    5    3    6
##   pt15 pt19 pt20
## 1    5    5    5
## 2    4    5    5
## 3    3    4    3
## 4    4    4    4
## 5    3    1    1
## 6    3    4    4
head(allbusdta)
##   eastwest educ sex age work pt01 pt02 pt03 pt04 pt08 pt09 pt10 pt11 pt12 pt14
## 1        1    3   1  62    1    6    6    7    6    5    4    5    6    6    7
## 2        2    5   2  64    1    6    6    5    5    6    6    5    7    6    7
## 3        1    1   1  22    1    5    6    4    4    4    2    2    4    4    5
## 4        2    3   1  59    4    7    4    5    5    6    7    7    7    4    6
## 5        2    5   2  30    1    5    6    4    6    4    1    3    4    4    4
## 6        1    5   1  41    1    5    6    4    5    5    3    4    5    3    6
##   pt15 pt19 pt20
## 1    5    5    5
## 2    4    5    5
## 3    3    4    3
## 4    4    4    4
## 5    3    1    1
## 6    3    4    4
head(stadt)
##                Stadt Einwohner Kategorie
## 1             Berlin    3644826         2
## 2            Hamburg    1841179         2
## 3            München    1471508         2
## 4               Köln    1085664         2
## 5 Frankfurt am Main     753056         2
## 6          Stuttgart     634830         2
```

Die kurze Inspektion der ersten Beobachtungen zeigt, dass die Daten wie zuvor eingelesen werden. Für den Import von csv-Dateien muss wie zuvor ein Argument zum Encoding `encoding = ÜTF-8"`

hinzugefügt werden.

An dieser Stelle sei noch angemerkt, dass es nach dem Import der Daten stets ratsam ist, die Struktur der Daten noch einmal zu überprüfen, um sicher zu gehen, dass der Import funktioniert hat. Sobald die Daten korrekt importiert wurden, kann mit der Datenaufbereitung gestartet werden.

3 Objekte und Funktionen

In den vorherigen beiden Kapiteln wurden Objekte und Funktionen bereits kurz angesprochen. Dieses Kapitel beschäftigt sich noch einmal ausführlich mit Funktionen und verschiedenen Objektarten in R, um die Grundlagen für eine erfolgreiche Datenanalyse zu legen. Dieses Kapitel dient damit dem tieferen Verständnis und bildet einen notwendigen Grundstein, bevor mit Datensätzen wie dem ALLBUS gearbeitet werden kann. Es wird auf verschiedene Objekttypen in R eingegangen. Dabei wird z.B. erläutert, wie Variablen und Datensätze angesteuert werden, wie Skalenniveaus erkannt und fehlende Werte identifiziert werden.

3.1 Back to the Basics

Zunächst stehen die verfügbaren Objekttypen von R im Zentrum. An dieser Stelle wird genauer erläutert, was Objekte (engl. objects) sind, wie sie gespeichert und umgewandelt werden. Objekte speichern verschiedene in R verfügbare Datentypen wie Vektoren, Listen, Matrizen oder auch Funktionen ab. Dieses Kapitel beschränkt sich auf Vektoren (engl. vectors) und Datensätze (engl. data.frames) als gängigste Objekttypen in der sozialwissenschaftlichen Analyse von Sekundärdaten.

Zur Veranschaulichung dieser beiden Datentypen werden im Folgenden verschiedene Objekte angelegt. Zur Erinnerung: Objekte entstehen durch eine Zuordnung mit `<-`. Im nächsten Code-Chunk werden drei Objekte durch Zuweisung angelegt: a, b, d. Es handelt sich in allen drei Fällen um Vektoren mit der Länge 1. Eine Anmerkung: Ein Objekt wird in der Regel nicht als "c" benannt, um Verwechselungen mit der Funktion `c()` zu vermeiden.

```
a <- 5
b <- "Hello World" # Zeichenketten müssen in Anführungszeichen gesetzt werden
d <- TRUE
```

Mit dem Befehl `class` kann die Klasse der Objekte angezeigt werden. Die Klasse bestimmt die grundlegenden Eigenschaften eines Vektors, z. B. ob Zahlen oder Buchstaben in den Daten abgespeichert werden. Alternativ ist der Befehl `str` ein nützlicher Befehl, da dieser die Struktur des Objekts aufzeigt. Mit dem Befehl `length` wird die Länge eines Objekts angezeigt.

```
class(a)
## [1] "numeric"
class(b)
## [1] "character"
class(d)
## [1] "logical"
str(a)
##  num 5
str(b)
##  chr "Hello World"
str(d)
##  logi TRUE
length(a)
## [1] 1
```

```
length(b)
## [1] 1
length(d)
## [1] 1
```

Die Ausgabe zeigt, dass drei verschiedenen Datenklassen in den Vektoren angelegt wurden. Objekt `a` ist numerisch, Objekt `b` ist ein sogenannter "character" bzw. Zeichenkette, Objekt "c" ist vom Typ "logical". Innerhalb eines Vektors müssen alle Werte die gleiche Datenklasse besitzen.

Achtung: R ist "case-sensitive"! Dies bedeutet, dass es einen Unterschied macht, ob der Name eines Objekts oder Funktion mit Klein- oder Großbuchstaben (a oder A) geschrieben wird. Zudem darf ein Name nicht mit einer Zahl beginnen und ebenso wenig darf er bestimmte Sonderzeichen beinhalten (z.B. *, #, !, $, @). Des Weiteren sollte auf die Nutzung von Umlauten verzichtet werden. Üblicherweise werden längere Namen für Variablen zur Leserlichkeit mit einem Unterstrich (my_variable), Bindestrich (my-variable), einem "Höcker" ("camel case", myVariable) oder einem Punkt (my.variable) geschrieben. Sie können allerdings nicht mit Sonderzeichen beginnen (_myvariable).

Objekte können auch wieder überspeichert werden, indem diesen ein neuer Wert oder eine Rechenfunktion zugeordnet wird. Achtung: Der ursprüngliche Zuordnungswert geht dabei verloren.

```
a <- 7
a
## [1] 7
a <- a + 2
a
## [1] 9
```

Ebenso ist es möglich, die Klasse eines Objektes zu ändern. Dies geschieht, indem das Objekt durch eine erneute Zuweisung überschrieben und eine neue Klasse festgelegt wird. In der Regel sollte die gewählte Klasse dem Skalenniveau der Variable (hier des Vektors) entsprechen. Dazu gibt es eine Reihe von Funktion: `as.numeric()`, `as.character()`, `as.factor()`, `as.logical()` und `as.data.frame()`. Im Code-Chunk wird der numerische Vektor `a` in eine Zeichenkette umgewandelt.

```
class(a) #Überprüfung der Klasse
## [1] "numeric"
a <- as.character(a) # Veränderung der Klasse
a #Ausgabe des Objekts
## [1] "9"
class(a) #Überprüfung der Klasse nach Zuweisung
## [1] "character"
```

Ein Vektor kann allerdings aus mehr als einer Beobachtung bestehen. Dies wird im nächsten Abschnitt erläutert.

3.2 Vektoren & Datensätze

In den Sozialwissenschaften sind oftmals Zusammenhänge zwischen zwei oder mehreren Faktoren auf Individual- oder Makroebene von Interesse. Bei der Arbeit mit Datensätzen handelt es sich

also in der Regel nicht nur um einzelne numerische Werte oder Zeichenketten, sondern um mehrere Ausprägungen zu einzelnen Beobachtungen. Ein Datensatz basiert in der Regel auf einer Datenmatrix, wobei die Variablen in den Spalten und die Beobachtungen in den Zeilen angeordnet sind. Die einzelnen Zeilen repräsentieren die Fälle (oftmals Versuchspersonen, Befragte oder auch Städte oder Länder) während die Spalten (Vektoren oder Variablen) die Faktoren von Erkenntnisinteresse abbilden. Ein Vektor besitzt Werte in einer festen Reihenfolge und alle Werte eines Vektors gehören zur gleichen Datenklasse.

Ein typisches Beispiel für einen Vektor (oder eine Variable) in einem Datensatz ist z.B. die Frage nach dem Alter der Befragten in einem Fragebogen. In R können viele Beobachtungen in einem einzigen Vektor abgespeichert werden. Dafür werden mehrere Werte mit der Funktion `c(x1, x2, x3, ...)` verkettet. Wichtig ist dabei, dass die einzelnen Werte durch ein `,` getrennt werden. Das "c" der Funktion steht für engl. "concatenate" und kann auch als "combine" interpretiert werden, da Werte über diese Funktion in einer festen Reihenfolge verbunden werden.

Hier ein Beispiel:

```
alter <- c(20,25,40,19,22,20,22,24,22,25) # alter wird erstellt
```

Mit der `class()` Funktion wird wiederum die Klasse des Vektors / der Vektoreinträge geprüft. Mit `length()` wird darüber hinaus ausgegeben, wie viele Angaben im Vektor vorhanden sind.

```
class(alter)
## [1] "numeric"
length(alter)
## [1] 10
```

Nun ist es möglich mit der entsprechenden Variable bzw. mit dem Vektor auch schon erste Berechnungen durchzuführen. Eine wichtige Funktion ist `mean()`. Diese berechnet den Mittelwert über alle Ausprägungen einer Variablen hinweg. Das heißt rechnerisch, dass die einzelnen Zahlenwerte addiert werden und durch die Anzahl der Einträge im Vektor geteilt wird.

```
mean(alter)
## [1] 23.9
```

Außerdem kann mit der Funktion `sum()` das Alter der Befragten summiert werden. Wenn diesen Wert wiederum durch 10 – also der Anzahl der Einträge im Vektor – dividiert wird, wird der obige Mittelwert ausgegeben.

```
sum(alter)
## [1] 239
sum(alter)/10
## [1] 23.9
sum(alter)/length(alter) #berechnet das Gesamtalter und teilt es
## [1] 23.9
                         #durch die Länge des Vektors (Anzahl der Befragten)
```

3.3 Datensätze erstellen

Ein Datensatz besteht normalerweise aus mehreren Vektoren der gleichen Länge. Ein weiteres Beispiel für einen Vektor, der in fast jeder Umfrage erhoben wird, ist das Geschlecht der Befragten. Auch hierfür lässt sich ein Vektor anlegen. In diesem Beispiel werden die Ausprägungen durch Zahlenwerte ausgedrückt. Hierbei steht eine `1` für `männlich`, eine `2` für `weiblich` und eine `3` für `divers`.

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

Nun wird die Klasse der Variable "geschlecht" überprüft:

```
class(geschlecht)
## [1] "numeric"
```

R erkennt die angelegte Geschlechtsvariable als numerischen Vektor, da bisher nur Zahlen hinterlegt wurden. Um den Zahlenwerten konkrete Bezeichnungen bzw. Beschriftungen zuzuordnen, muss der Vektor mit der Funktion `factor()` in einen Faktor umgewandelt werden.

```
geschlecht.faktor <- factor(geschlecht)
class(geschlecht.faktor)
## [1] "factor"
geschlecht.faktor
##  [1] 1 2 1 2 1 3 1 2 2 1
## Levels: 1 2 3
```

Aus der Ausgabe wird erkenntlich, dass ein Faktor "Levels" aufweist. Dies sind die Ausprägungen des Faktors. An dieser Stelle sind noch die eingetragenen Zahlenwerte hinterlegt. Wie die Levels eines Faktors neu beschriftet und zum Beispiel "divers", "Frau" und "Mann" als Werte hinterlegt werden, wird im nächsten Kapitel erläutert.

Im nächsten Schritt werden die beiden Vektoren zu einem Datensatz zusammengeführt. Dies geschieht über die Funktion `data.frame(x,y)`. Dafür wird wiederum ein neues Objekt erstellt. Anschließend wird erneut die Klasse des Objekts überprüft.

```
test_daten <- data.frame(alter, geschlecht.faktor)
class(test_daten)
## [1] "data.frame"
```

Achtung: Damit ein Datensatz gebildet werden kann, müssen alle Variablen gleich viele Fälle aufweisen. Ansonsten wird der Befehl gestoppt und eine Fehlermeldung erscheint.

Hinweis Ein Datensatz trägt optisch den Charakter einer Matrix. Zudem ist Matrixalgebra maßgeblich für die Ausführung vieler statistischer Verfahren und so sind Vektoren und Matrizen wesentlich für die Arbeit mit R. Neben Vektoren sind Matrizen sogar eigene Datentypen in R. Eine Matrix kann über den Befehl `matrix()` erstellt werden. Allerdings können in einer Matrix nur Vektoren des gleichen Typs (z.B. numerisch) gespeichert werden. Daher eignen sich Matrizen nicht für große Datensätze wie den ALLBUS, da diese eine Vielzahl an Variablen verschiedener Typen Klassen (Skalenniveaus) enthalten. Ein Datensatz (`data.frame()`) kann Vektoren verschiedener Klassen speichern und eignet sich somit für die Arbeit mit großen Umfragedaten.

3.4 Objekte in Datensätzen

3.4.1 Daten einlesen

Die Befehle zum Einlesen von Datensätzen wurden im vorherigen Kapitel erläutert (siehe Kapitel 2 für weitere Informationen). In diesem Schritt wird der komplette ALLBUS-Datensatz aus dem Erhebungsjahr 2018 eingelesen.

```
#install.packages("haven")
library(haven)
getwd()
## [1] "C:/Users/Kieslich/Desktop/R-Script"
#setwd()
allbus <- as.data.frame(read_spss("Daten/ALLBUS2018.sav"))
```

Die Funktion `names()` liefert einen Überblick über die im Datensatz vorhandenen Variablen.

```
names(allbus)
##  [1] "eastwest" "educ"     "sex"      "age"      "work"     "pt01"
##  [7] "pt02"     "pt03"     "pt04"     "pt08"     "pt09"     "pt10"
## [13] "pt11"     "pt12"     "pt14"     "pt15"     "pt19"     "pt20"
```

3.4.2 Variablen anwählen

Jede gründliche Datenarbeit beinhaltet eine Überprüfung der Variablen. In R ist es besonders wichtig, dass die Daten das richtige Skalenniveau aufweisen. Deshalb sollte zunächst eine (stichprobenartige) Prüfung der einzelnen Skalenniveaus (Klassen) durchgeführt werden. Ebenso empfiehlt es sich bei Fehlermeldungen die Klassen der Variablen zu überprüfen.

Um auf Variablen in einem Datensatz zuzugreifen, müssen die Variablen als Elemente des Datensatzes angesprochen werden. Die Funktion `class(eastwest)` führt zu folgendener Fehlermeldung: **"Error: object 'eastwest' not found"**

R weist mit der Fehlermeldung darauf hin, dass es das Objekt nicht finden kann. Mit Blick auf die Arbeitsumgebung fällt auf, dass die Variablen dort nicht als Objekte aufgeführt sind. Doch wo sind die Variablen dann?

Als Element des Datensatzes muss angegeben werden, dass R die entsprechende Variable im Datensatz finden kann. Dies wird durch ein $ Zeichen vorgenommen.

```
class(allbus$eastwest)
## [1] "haven_labelled" "vctrs_vctr"     "double"
head(allbus$eastwest)
## <labelled<double>[6]>: ERHEBUNGSGEBIET (WOHNGEBIET): WEST - OST
## [1] 1 2 1 2 2 1
##
## Labels:
##  value              label
```

```
##       1 ALTE BUNDESLAENDER
##       2 NEUE BUNDESLAENDER
```

Hinweis: Als Klasse wird hier `haven_labelled` angegeben. Dies ist eine spezifische Klasse für Daten, die mit dem haven-Paket geladen wurden. Unter der Ausgabe der Einzelwerte werden die Wertbeschriftungen angezeigt. Im Fall `eastwest` sind die Ausprägungen `1=Alte Bundesländer` und `2=Neue Bundesländer`.

3.4.3 Indexing

Mit Indexing können einzelne Zeilen, Spalten oder Zellen eines Datensatzes angewählt werden. Im folgenden Beispiel wird das Alter der fünften Person aus der Befragung ausgegeben. Dafür muss nach der Auswahl der Variablen in eckigen Klammern die Nummer der entsprechenden Zeile geschrieben werden.

```
allbus$age[5]
## <labelled<double>[1]>: ALTER: BEFRAGTE(R)
## [1] 30
##
## Labels:
##  value             label
##    -32 NICHT GENERIERBAR
```

Die Abfrage kann auch direkt auf den Datensatz bezogen werden. Dabei muss der Index aus zwei Zahlen bestehen. Wenn das Alter der im Datensatz an fünfter Stelle stehenden Person bezogen werden soll, muss zuerst ausfindig gemacht werden, welche Spalte des Datensatzes die Variable "Alter" darstellt (hier die vierte Spalte des Datensatzes). In den eckigen Klammern muss als erstes die Zeile, also die befragte Person, und danach die Spalte von Interesse (Variable) angegeben werden.

```
allbus[5,4] # die fünfte Zeile und vierte Spalte
## <labelled<double>[1]>: ALTER: BEFRAGTE(R)
## [1] 30
##
## Labels:
##  value             label
##    -32 NICHT GENERIERBAR
```

Es ist ebenso möglich, die Angabe zu Zeilen oder Spalten frei zu lassen. In diesem Falle werden alle Informationen der Zeilen oder Spalten angezeigt, d.h. je nach Auslassung alle Informationen zur ausgewählten Person im Datensatz oder das Alter aller Befragten.

```
allbus[5,] # die fünfte Zeile, alle Spalten
##   eastwest educ sex age work pt01 pt02 pt03 pt04 pt08 pt09 pt10 pt11 pt12 pt14
## 5        2    5   2  30    1    5    6    4    6    4    1    3    4    4    4
##   pt15 pt19 pt20
## 5    3    1    1
```

```
#allbus[,4] # alle Zeilen, die vierte Spalte
head(allbus[,4], n=50)
## <labelled<double>[50]>: ALTER: BEFRAGTE(R)
##  [1] 62 64 22 59 30 41 43 39 40 69 37 77 73 46 68 53 27 25 68 42 83 52 34 35 47
## [26] 68 46 55 35 55 25 80 37 40 36 30 83 45 39 57 70 37 85 27 78 44 78 51 81 75
##
## Labels:
##  value             label
##    -32 NICHT GENERIERBAR
```

Hinweis zur Darstellung der Ausgabe: Da mit dem Befehl `allbus[,4]` *alle* Fälle der vierten Spalte ausgegeben werden (N=3477), wurden für die Darstellung hier nur die ersten 50 Fälle zur Anschauung ausgegeben. Hierfür wird die Indizierung wiederum mit dem `head()` Funktion verbunden.

Anstatt alle Beobachtungen in der Konsole ausgeben zu lassen, empfiehlt es sich, statistische Lage- und Streuungsmaße zu analysieren (siehe Kapitel 6). Für einen ersten Eindruck über die Daten können – wie bereits oben verdeutlicht – die Funktionen `head()` oder `tail()` genutzt werden (siehe Kapitel 2).

Es gibt noch viele Erweiterungen der Index-Funktion. Auf diese Funktionen werden im nächsten Kapitel zur Datenaufbereitung näher eingegangen.

3.5 Fehlende Werte

Schließlich wird in diesem Kapitel noch ein Wertetyp vorgestellt: `NA`. Dies steht für `not available` - einen fehlenden Wert. Fehlende Werte kommen in Befragungen relativ häufig vor. Sie haben, je nach Frageformulierung und Antwortoption, unterschiedliche Bedeutungen. Beispielsweise kann es sein, dass eine Person eine Frage übersprungen hat, die Frage nicht beantworten konnte, wollte oder einfach keine Antwort wusste. In manchen Fällen kann die Ausprägung "weiß nicht" jedoch auch einen gültigen Wert mit Informationsgehalt darstellen. Dies ist stets abhängig von der erhobenen Variable. Fehlende Werte (engl. missings) werden in R als `NA` im Datensatz aufgeführt.

In R gibt es einen einfachen Befehl um zu prüfen, ob eine Variable überhaupt fehlende Werte enthält: `anyNA()`.

```
anyNA(allbus$sex)
## [1] FALSE
anyNA(allbus$pt03)
## [1] TRUE
```

Das Ergebnis ist entweder `TRUE` oder `FALSE`, d.h. ein logischer Wert. Im ALLBUS enthält die Variable Geschlecht demnach keine fehlenden Werte. Es gibt jedoch fehlende Werte bei der Angabe des Vertrauens in den Bundestag.

Um herauszufinden, wie viele Werte einer Variable tatsächlich fehlen, muss zunächst eruiert werden, welche Einträge einer Variable fehlend sind. Dazu wird die Funktion `is.na()` genutzt.

```
is.na(allbus18$pt03)
```

Die Ausgabe wird hier nicht ausgeführt, da diese sehr groß ist. Ein `TRUE` bedeutet, dass der entsprechende Eintrag ein fehlender Fall ist. Ein `FALSE` bedeutet, dass der Fall einen gültigen Wert besitzt.

3.6 Verkettung von Funktionen

In der Regel ist es zu vermeiden, in der Konsole lange Ausgaben der Daten auszugeben. Standardmäßig gibt R die ersten 1000 Beobachtungen aus, wenn eine Variable über `print()` oder die Eingabe des Variablennamens aufgerufen wird. Um einen Überblick über eine Variable mit vielen Fällen zu gewinnen, reicht es beispielsweise aus, sich die ersten 40 Ergebnisse anzeigen zu lassen. Dafür wird die Funktion `head()` benutzt und mit der vorherigen Funktion verknüpft.

Der folgende Code-Chunk fasst zwei Funktionen durch eine Verschachtelung zusammen. R führt dabei zunächst die Funktion in der inneren Klammer aus und danach die Funktion der äußeren Klammer. Zunächst wird daher ein vorheriger Befehl ausgeführt, der anzeigt, ob ein Wert einer Variablen ein fehlender Wert ist. Danach wird die head-Funktion darauf angewendet, sodass nur die ersten Fälle gezeigt werden. Wenn die ersten 40 Zeilen mit der `head()` Funktion ausgeben wollen, kann das Argument `n=40` ergänzt werden. An dieser Stelle kann eine beliebige Anzahl gewählt werden. Probieren Sie verschiedene Zahlenwerte aus.

```
head(is.na(allbus$pt03), n=40)
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [37] FALSE FALSE FALSE FALSE
```

Schließlich interessiert, wie viele fehlende Fälle in einer Variable vorhanden sind. Auch hier wird eine verkettete Funktion benutzt. Dazu wird die äußere Funktion `sum()` benutzt, welche alle logischen `TRUE` Werte addiert.

```
sum(is.na(allbus$pt03))
## [1] 91
```

Die Variable `pt03` (Vertrauen in den Bundestag) weist damit insgesamt 91 fehlende Fälle auf, d.h. 91 Befragte haben keine Angabe zu ihrem Vertrauen in den Bundestag gemacht.

3.7 Exkurs: Funktionen

Funktionen sind häufig als Verben aufzufassen, da ein Befehl ausgeführt wird. So sind die Namen der Funktionen oftmals Abkürzungen von Verben, die anzeigen, dass eine bestimmte Operation ausgeführt wird. Funktionen sind also Handlungsanweisungen. Hilfreiche Metaphern für das Verständnis einer Funktion sind Gebrauchsanleitungen (z.B. zum Aufbau von Möbelstücken) oder auch Rezepte zum Backen oder Kochen. Eine Funktion besitzt Argumente (Angaben zu benötigten Gegenständen, Zutaten, Inhalten) und einen Funktionskörper (Anleitung zur Vorgehensweise). Der Funktionskörper

führt die zu befolgenden Schritte in einer festgelegten Reihenfolge aus. Abweichungen von der Reihenfolge sind nicht möglich. Durch das Einsetzen von Objekten des Arbeitsumfeldes in die Funktion werden die Argumente besetzt und der Funktionskörper kann ausgeführt werden, sofern alle nötigen Argumente benannt werden. Funktionen können in R geschrieben werden. Konzeptionell ist die grundlegende Struktur einer Funktion wie folgt:

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

Im nächsten Code-Chunk wird zur Veranschaulichung eine neue Funktion erstellt. Diese Funktion wird nach der Zuweisung als Objekt im Arbeitsumfeld erscheinen. In diesem Beispiel lautet das Argument "x" als Konvention für einen numerischen Wert. Die Funktion addiert 2 zum Element "x" und gibt den neuen Wert wieder (return). Eine Wiedergabe (return) wird stets am Ende einer Funktion benötigt, damit die Funktion etwas (das Ergebnis) in der Konsole ausgeben kann. Ansonsten würde das Ergebnis der Funktion zwar vorliegen, aber nicht sichtbar oder für weitere Analysen nutzbar sein.

```
plus2 <- function(x) {  # Eine Funktion wird erstellt
  r = x + 2
  return(r)
}
```

Im nächsten Code-Chunk wird die Performanz der neuen Funktion getestet. Dazu wird ein Vektor "test" mit zehn Ausprägungen erstellt. Die Funktion wird für den Vektor angewendet und einem neuen Vektor mit dem Namen "xplus2" zugewiesen.

```
test <- 1:10 # Vektor "test" wird erstellt
test # Ausgabe des Test-Vektors
##  [1]  1  2  3  4  5  6  7  8  9 10
xplus2 <- plus2(test) #neue Funktion wird auf den Vektor angewendet
xplus2 #Ausgabe
##  [1]  3  4  5  6  7  8  9 10 11 12
```

Die Funktion scheint zu funktionieren. Die aufmerksame Leserschaft dieses Skripts wird bemerkt haben, dass die Implementation der Funktion "plus2" in gewisser Weise überflüssig ist, da ebenso die Rechenoperation `test + 2` verwendet werden kann. An dieser Stelle ging es vor allem darum, die Funktionsweise von Funktionen möglichst eingängig durch ein einfaches Beispiel zu erklären.

In R können also Funktionen geschrieben und genutzt werden. Für den Einstieg in R ist es allerdings zunächst wichtiger, Funktionen grundsätzlich zu verstehen, damit die bereits in R implementierten Funktionen ohne großes Kopfzerbrechen genutzt werden können. Als weiteres Beispiel für ein tieferes Verständnis empfiehlt sich z.B. eine Beschäftigung mit der Funktion `sample()`. Dazu kann der nächste Code-Chunk betrachtet werden. Was macht die Funktion? Welche Argumente sind notwendig und was passiert, wenn die default-Argumente verändert werden? Tipp: `?sample`. Inwiefern unterscheiden sich die Ergebnisse der beiden `sample`-Befehle des Code-Chunks?

```
set.seed(23) # Funktion legt eine Sequenz von Zufallszahlen fest (Replizierbarkeit)
sample(test, 5, replace = TRUE)
## [1] 8 3 8 9 7
sample(x = test, size = 5)
## [1] 2 1 7 5 9
```

An dieser Stelle sei noch angemerkt, dass bei einer Funktion mit mehreren Argumenten die Argumente in einer bestimmten Reihenfolge erwartet werden (siehe Hilfe zur Funktion). Bei Einhaltung der Reihenfolge kann die explizite Nennung der Argumente ausgelassen werden. Alternativ können die Argumente namentlich aufgerufen werden, ohne dabei an eine feste Reihenfolge gebunden zu sein. Bei Funktionen mit vielen Argumenten empfiehlt es sich die Argumente namentlich aufzurufen (z.B. size = 5). In einigen Funktionen werden standardmäßig Werte eingetragen (default). Die default-Einstellungen sind in der Regel in der Hilfe ersichtlich (ebenfalls unter usage). In diesem Beispiel wird die Funktion standardmäßig ohne Zurücklegen ausgeführt (`replace = FALSE`). Dies kann allerdings durch die Nennung der Argumente in der Funktion geändert werden, in diesem Falle `replace = TRUE`.

3.8 Exkurs: Die apply-Funktion

In allen bisherigen Beispielen wurde der Code Zeile für Zeile beziehungsweise Variable für Variable ausgeführt. Es kommt aber häufig vor, dass ein bestimmter Befehl nicht nur für ein Item, sondern für mehrere Items oder Item-Batterien (siehe Kapitel zur Indexbildung) ausgeführt werden soll. Damit eine bestimmte Funktion nicht für jede Zeile geschrieben werden muss, können in R Schleifen (Iterationen) verwendet werden. Durch Schleifen (engl. loops) können sich Funktionen 10-mal, 100-mal oder auch 1000-mal wiederholen - nur durch eine Zeile Code! Es gibt verschiedene Möglichkeiten, um Schleifen zu schreiben, wie z.B. eine “for”-Schleife (engl. for loop). Im kurzen Exkurs wird die “apply-Familie” vorgestellt, die unterschiedliche Möglichkeiten für Iterationen zur Verfügung stellt und aufwendigere Schleifen vermeidet. Zu den wesentlichen apply-Funktionen zählen:

- apply()
- tapply()
- lapply()
- sapply()

Die jeweiligen Funktionen unterscheiden sich durch a) der spezifizierung durch zusätzliche Argumente und b) unterschiedlichen Output (z.B. der Objekttyp). Die unterschiedliche Funktionsweise wird nun an den ALLBUS-Daten gezeigt.

Mit `apply()` kann eine beliebige Funktion auf alle Variablen im Datensatz angewandt werden. Dazu werden drei zusätzliche Argumente benötigt: Zunächst der Datensatz, der die entsprechenden Variablen beinhaltet. Danach muss spezifiziert werden, ob die Funktion für jede Zeile (1) oder für jede Spalte (2) ausgeführt wird. In der Regel soll die Funktion auf jede Spalte verwendet werden, sodass auf den Datensatz eine 2 folgt. Im Anschluss wird lediglich die Funktion geschrieben, die für jede Variable ausgeführt werden soll. Im Beispiel ist es die `class`-Funktion, die allerdings ohne Klammern genutzt wird.

```
#apply-Funktion
apply(allbus, 2, class)
##  eastwest      educ       sex       age      work      pt01      pt02      pt03
## "numeric" "numeric" "numeric" "numeric" "numeric" "numeric" "numeric" "numeric"
##      pt04      pt08      pt09      pt10      pt11      pt12      pt14      pt15
## "numeric" "numeric" "numeric" "numeric" "numeric" "numeric" "numeric" "numeric"
##      pt19      pt20
## "numeric" "numeric"
```

Die Variablen `pt01` bis `pt20` enthalten Items zum allgemeinen und politischen Vertrauen der Befragten. Soll eine Funktion nur für ausgewählte Variablen genutzt werden, dann muss das Argument zum Datensatz spezifiziert werden, zum Beispiel durch das Indexing.

```
apply(allbus[, 6:18], 2, class)
##      pt01      pt02      pt03      pt04      pt08      pt09      pt10      pt11
## "numeric" "numeric" "numeric" "numeric" "numeric" "numeric" "numeric" "numeric"
##      pt12      pt14      pt15      pt19      pt20
## "numeric" "numeric" "numeric" "numeric" "numeric"
apply(allbus[, 6:18], 2, mean, na.rm=TRUE)
##     pt01     pt02     pt03     pt04     pt08     pt09     pt10     pt11
## 4.636521 5.091330 4.057885 4.612281 4.547633 3.349869 3.796070 5.135556
##     pt12     pt14     pt15     pt19     pt20
## 3.983401 5.052556 3.463177 3.567831 3.595031
```

`tapply()`, `lapply()` und `sapply()` funktionieren im Prinzip ähnlich, allerdings ändert sich der Output der Variablen und ein zusätzliches Argument zur Zeile oder Spalte ist nicht mehr notwendig. Deutlich wird das bei `lapply()`, der Output hat nun die Klasse `list` (daher auch das "l"apply):

```
output <- lapply(allbus[, 6:9], mean, na.rm=TRUE)
class(output)
## [1] "list"
print(output)
## $pt01
## [1] 4.636521
##
## $pt02
## [1] 5.09133
##
## $pt03
## [1] 4.057885
##
## $pt04
## [1] 4.612281
```

`sapply()` unterscheidet sich lediglich im Output zu `lapply()`. Die neue Variable ist vom Typ Vektor mit der Klasse `numeric`:

```
output <- sapply(allbus[, 6:9], mean, na.rm=TRUE)
class(output)
## [1] "numeric"
print(output)
##     pt01     pt02     pt03     pt04
## 4.636521 5.091330 4.057885 4.612281
```

Nicht alle Funktionen können wie `class()` oder `mean()` innerhalb der apply-Familie genutzt werden. Manchmal ist es notwendig, hierzu eine eigene Funktion zu schreiben, z.B. beim `table()`-Befehl. Dabei müssen keine {}-Klammern gesetzt werden, die eigene Funktion wird mit FUN = eingeleitet:

```
lapply(allbus[, 6:9], FUN = function(x) table(x))
## $pt01
## x
##    1    2    3    4    5    6    7
##   88  173  429  785 1015  712  270
##
## $pt02
## x
##   1   2   3   4   5   6   7
##  68 121 275 587 720 872 598
##
## $pt03
## x
##   1   2   3   4   5   6   7
## 198 316 570 934 857 398 113
##
## $pt04
## x
##    1    2    3    4    5    6    7
##   80  166  425  803 1015  717  214
```

Durch `tapply()` können schließlich Werte für zwei (oder mehr) Gruppen getrennt angegeben werden, was insbesondere dann nützlich ist, wenn Mittelwerte miteinander verglichen werden (siehe Kapitel zu Mittelwertvergleiche). Es wird eine zusätzliche Gruppierungsvariable benötigt (im Beispiel die Variable `eastwest`). In der Funktion wird demnach der `mean()`-Befehl für Ost- und für Westdeutschland ausgeführt. Für die Variable pt01 unterscheidet sich der Mittelwert zwischen Ost- und Westdeutschland nur unwesentlich.

```
tapply(allbus$pt01, allbus$eastwest, mean, na.rm=TRUE)
##        1        2
## 4.625945 4.659633
```

Mit der apply-family können viele Zeilen Code gespart werden, was den Code nicht nur besser zu lesen, sondern auch weniger anfällig für Fehler macht. Der kurze Exkurs kann natürlich nicht sämtliche Funktionsweisen aufzeigen, sollte aber dennoch einen Überblick anbieten, was mit der apply-Funktion möglich ist.

4 Datenaufbereitung

Nur in seltenen Fällen können die Daten, so wie sie in einem Datensatz vorliegen, ohne weitere Aufbereitung für die Datenanalyse verwendet werden. Daher werden die Daten vor der Analyse vorbereitet; dieser Schritt wird Datenaufbereitung (engl. data preparation, data manipulation, data processing) genannt. Im folgenden Kapiteln werden die grundlegenden Schritte der Datenaufbereitung in R behandelt.

Bei der Datenaufbereitung in R sind zwei Vorgehnsweisen beliebt: Zum einen die standardmäßige Datenaufbereitung mit `Base-R` (ohne zusätzliche Pakete) und zum anderen die Nutzung des Pakets `dplyr` (aus dem `tidyverse`). In diesem Kapitel werden beide Varianten vorgestellt. Im späteren Kapiteln dieses Skriptes wird der Übersichtlichkeit halber nur `Base-R` verwendet. Fast alle Operationen sind aber grundsätzlich mit `dplyr` reproduzierbar. Zudem wird `Base-R` als Standard der Programmiersprache stets möglich sein. Obwohl sich Pakete wie `dplyr` aktuell großer Beliebtheit erfreuen, könnten diese Pakete über kurz oder lang aus der Mode geraten und unter Umständen auch nicht weiter aktualisiert werden. Insofern sind Grundkenntnisse von `Base-R` ratsam, auch wenn neuere Pakete die Datenaufbereitung erleichtern können.

```
# falls noch nicht geschehen, wird der R-Datensatz geladen
library(haven)

allbus <- as.data.frame(read_spss("Daten/ALLBUS2018.sav")) #Einlesen der Daten
allbus.kurz <- head(allbus, n = 20) #Kurzdatensatz wird erstellt
```

4.1 Bedingungen formulieren

Vor den einzelnen Schritten der Datenaufbereitung müssen noch einige Programmiergrundlagen geschaffen werden, und zwar dazu, wie Bedingungen formuliert werden. Während der Datenaufbereitung werden oft Bedingungssätze verwendet, um einen Befehl nur auf bestimmte Befragte oder Gruppen zu beziehen.

Beispiele:

- Es sollen nur Männer in einer Analyse betrachtet werden (Bedingung: Geschlecht gleich Mann).
- Eine Grafik soll nur Einstellungen aus Gruppe der 20- bis 39-Jährigen zeigen (Bedingung: Alter zwischen 20 und 39 Jahren, oder nicht kleiner als 20 Jahre und nicht älter als 39 Jahre).
- Fälle, d.h. Befragte, mit fehlenden Werten im politischen Vertrauen sollen ausgeschlossen werden.

Diese Beispiele werden in diesem Kapitel durchgegangen und in Aussagen übersetzt, die R versteht. Dies geschieht anhand sogenannter logischer Operatoren.

Bei der Benutzung von logischen Operatoren werden boolesche Werte zurückgegeben. Boolesche Werte bezeichen Werte, die wahr oder falsch sind (`TRUE` or `FALSE`). Eine Übersicht der wichtigsten logischen Operatoren findet sich im folgenden Code-Chunk:

```
a <- 3 # Vektor a wird erstellt
a
## [1] 3
b <- 4 # Vektor b wird erstelt
b
## [1] 4
a == b # a ist gleich b
## [1] FALSE
a != b # a ist ungleich b
## [1] TRUE
a < b  # a ist kleiner b
## [1] TRUE
a <= b # a ist kleiner/ gleich b
## [1] TRUE
a > b  # a ist größer b
## [1] FALSE
a >= b # a ist größer/gleich b
## [1] FALSE
a < 8 | b < 2 # oder
## [1] TRUE
a < 8 & b < 2 # und
## [1] FALSE
a %in% b # a stimmt mit b überein
## [1] FALSE
```

Im nächsten Abschnitt wird gezeigt, wie diese logischen Operatoren zur Datenaufbereitung genutzt werden.

4.1.1 Gleich- und Ungleichbedingungen

Oftmals ist es interessant, Daten für bestimmte Subgruppen der Bevölkerung zu analysieren. Ein Beispiel dafür sind getrennte Analysen für die Geschlechter. Dazu muss zunächst herausgefunden werden, welche Befragten in der Analyse Männer sind. Bei welchen Befragten ist also das Geschlecht gleich „Mann“? Um dies herauszufinden, werden die Bedingung in eine logische Aussage übersetzt, die R versteht. Diese Aussage beginnt in der Regel mit der Variable, hier also mit „Geschlecht”: `allbus.kurz$sex`. Darauf folgt die Bedingung, hier `== 1`.

Der Code `1` steht für die befragten Männer und muss im Codebook nachgeschlagen werden oder durch `attributes(allbus.kurz$sex)$labels` überprüft werden.

```
allbus.kurz$sex  == 1
##  [1]  TRUE FALSE  TRUE  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE  TRUE FALSE
## [13]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
```

Das Ergebnis der Aussage ist ein Vektor mit logischen Operatoren: `TRUE` und `FALSE`. Dieser Vektor zeigt, ob die Bedingung erfüllt und der Befragte somit ein Mann ist.

Da das Geschlecht hier binär abgefragt wurde, könnten die Männer auch mit einer anderen Bedingung gekennzeichnet werden. Und zwar: Bei welchen Befragten ist das Geschlecht nicht gleich "Frau"?

Um diese Bedingung in eine logische Aussage zu übersetzen, die R prüfen kann, wird eine Verneinung verwendet werden. Diese wird einfach vor der Aussagen mit einem `!` gekennzeichnet.

```
!allbus.kurz$sex  == 2
##  [1]  TRUE FALSE  TRUE  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE  TRUE FALSE
## [13]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
```

Außerdem kann anstatt der Verneinung das Zeichen für ungleich `!=` verwendet werden.

```
allbus.kurz$sex  != 2
##  [1]  TRUE FALSE  TRUE  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE  TRUE FALSE
## [13]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
```

Das Ergebnis ist in beiden Fällen identisch.

4.1.2 Größer- und Kleinerbedingungen

Eine nächste typische Unterscheidung ist die Aufteilung der Befragten in sogenante Altersgruppen. Welche Befragten sind älter als 30 Jahre? Oder anders gefragt, wessen Alter ist größer als 30? Auch diese Frage kann wieder in mehrere logische Aussagen überführt werden, die R interpretieren kann.

Die erste Möglichkeit besteht darin, die Personen auszuwählen, deren Alter größer als 30 ist - und zwar mit dem mathematischen Zeichen `>`.

```
allbus.kurz$age  > 30
##  [1]  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [13]  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE
```

Außerdem können die Personen ausgewählt werden, deren Alter größer oder gleich 31 ist, mit dem Zeichen `>=`. Das Ergebnis ist mit dem vorherigen Ergebnis identisch. Dies ist jedoch nur bei diskreten Variablen der Fall. Das heißt bei Variablen, die nur natürliche Zahlen `1,2,3,4,5 ...` als Wert haben können.

```
allbus.kurz$age  >= 31
##  [1]  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [13]  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE
```

Analog funktioniert es für Personen, die jünger als 50 Jahre alt sind.

```
allbus.kurz$age  < 50
##  [1] FALSE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE
## [13] FALSE  TRUE FALSE FALSE  TRUE  TRUE FALSE  TRUE
```

4.1.3 Und- und Oder-Bedingungen

Welche Befragten sind Männer, die älter als 30 sind? Das heißt, die beiden Bedingungen "Alter größer als 30" und "Geschlecht gleich Mann" müssen erfüllt sein. Um diese Bedingung in eine logische Aussage zu übersetzen, wird daher das Zeichen für "Und" `&` verwendet. Damit werden die beiden Bedingungen verbunden und festgelegt, dass beide gelten müssen.

```
allbus.kurz$age  >= 30 & allbus.kurz$sex  == 1
##  [1]  TRUE FALSE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE  TRUE FALSE
## [13]  TRUE  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE
```

Bedingungen können außerdem mit einem "Oder" `|` miteinander verbunden werden. Das "Oder" gibt an, dass nur eine der beiden Bedingungen gelten muss. Zum Beispiel sollen die besonders alten und die besonders jungen Befragten im Datensatz gekennzeichnet werden. Die Bedingung dazu laut, dass Personen betrachtet werden, die jünger als 25 oder älter als 65 Jahre sind.

```
allbus.kurz$age  < 25 | allbus.kurz$age > 65
##  [1] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE
## [13]  TRUE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE
```

4.1.4 Der in-Operator

Oft soll überprüft werden, ob Personen in einer bestimmten Gruppe sind, die vorher festgelegt wurde, z.B. in bestimmten Bundesländern leben oder zu bestimmten Berufsgruppen zählen. Um mühselige Verknüpfungen mit dem oder-Zeichen `|` zu vermeiden, gibt es die Möglichkeit eine Variable mit einer Gruppenzugehörigkeit zu vergleichen, die vorher als Vektor definiert wurde.

Beispiel: Es sollen Personen betrachtet werden, deren Vertrauen in das Gesundheitswesen besonders hoch (`6` oder `7`) oder besonders niedrig (`1` oder `2`) ist. Dazu wird zuerst ein Vektor mit der Gruppe definiert, die die gewünschte Eigenschaft hat (`c(1,2,6,7)`) und dieser wird dann in einem Objekt abgespeichert (`hoch.niedrig.vertrauen <- c(1,2,6,7)`). Im Anschluss wird mit dem `%in%` überprüft, ob die einzelnen Befragten in dieser Gruppe zu finden sind.

```
# Vergleichsverktor erstellen
hoch.niedrig.vertrauen <- c(1,2,6,7)

# Vergleich durchführen
allbus.kurz$pt01 %in% hoch.niedrig.vertrauen
##  [1]  TRUE  TRUE FALSE  TRUE FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE
## [13] FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
```

Außerdem gibt es für die Erstellung von langen Zahlen-Vektoren eine schnellere Schreibweise. Wenn beispielsweise Personen, die `30` bis `50` Jahre alt sind, betrachtet werden sollen, dann kann ein Vektor erstellt werden, der alle Alterszahlen beinhaltet, bei denen der Test dann ausschlagen soll.

Diesen Vektor könnte zum Beispiel mit diesem Befehl erstellt werden:

`c(30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50)`. Einfacher geht dies mit dem Doppelpunkt. Dabei muss nur Beginn und Ende des Vektors festgelegt werden: `30:50`. Das genaue Vorgehen sieht dann so aus:

```r
# Vergleichsvektor erstellen
altersgruppe <- 30:50

# Vergleich durchführen
allbus.kurz$age %in% altersgruppe
##  [1] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE
## [13] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE
```

Oder noch kürzer ohne vorherige Zuweisung:

```r
allbus.kurz$age %in% 30:50
##  [1] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE
## [13] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE
```

Wie eine solche Auswahl über eine Zuweisung als neuer Datensatz im Arbeitsumfeld gespeichert wird, zeigt der folgende Abschnitt.

4.2 Base-R

4.2.1 Spalten auswählen und umbenennen

Für eine Analyse werden selten alle Spalten eines Datensatzes benötigt. Damit die Analyse und deren einzelne Schritte übersichtlich bleiben, macht es Sinn, in einem ersten Schritt die wichtigsten Spalten auszuwählen.

Einzelne Spalten und Zeilen werden in R mit eckigen Klammern angesprochen (siehe Kapitel 3.4.3). Eine ganze Spalte kann ausgeben werden, in dem die erste Stelle in der eckigen Klammer frei bleibt. Für die Spalte 4 würde das so aussehen: `allbus.kurz[ ,4]`. Statt der Zahl der Spalte können auch den Namen verwendet werden, also `allbus.kurz[ ,"..."]`.

Mehrere Spalten können dann mit einem Vektor ausgewählt werden. Für Spalte 3 und 4 zum Beispiel `allbus.kurz[ ,c(3,4)]` oder mit den Namen der Spalten: `allbus.kurz[ ,c(„...",„...")]`.

Um die aktuellen Namen der Spalten in einem Datensatz zu sehen, wird der Befehl `names()` verwendet. Die Namen des ALLBUS-Datensatzes gibt daher der folgende Befehl aus: `names(allbus.kurz)`. Diese Namen können einfach mit einem neuen Vektor geändert werden.

Ein Workflow für die Zusammenstellung von Variablen in einem Datensatz könnte dann wie folgt aussehen. Es sollen nur die Variablen „Alter“, „Vertrauen in die Europäische Kommission” `pt19` und „Vertrauen in das Europäische Parlament"`pt20` in einem Datensatz abgespeichert werden.

```r
# Spalten auswählen
eu.vertrauen <- allbus.kurz[ ,c("age", "pt19", "pt20")]

# Neue Namen vergeben
names(eu.vertrauen) <- c("alter", "vertrauenEuKommission", "vertrauenEuParlament")

# Änderungen überprüfen
```

```
eu.vertrauen
##    alter vertrauenEuKommission vertrauenEuParlament
## 1     62                     5                    5
## 2     64                     5                    5
## 3     22                     4                    3
## 4     59                     4                    4
## 5     30                     1                    1
## 6     41                     4                    4
## 7     43                    NA                   NA
## 8     39                     5                    6
## 9     40                     5                    5
## 10    69                     4                    4
## 11    37                     4                    4
## 12    77                    NA                   NA
## 13    73                     1                    1
## 14    46                     3                    5
## 15    68                    NA                    4
## 16    53                     3                    3
## 17    27                     6                    6
## 18    25                     4                    4
## 19    68                     5                    6
## 20    42                     4                    4
```

4.2.2 Zeilen filtern

Zur Datenaufbereitung gehört auch, dass die Fälle, auf der die Analyse beruhen soll, ausgewählt werden. Zum Beispiel muss entschieden werden, welche Befragten aufgrund von fehlenden Werten ausgeschlossen werden sollen.

Im Folgenden sollen nur Personen betrachtet werden, die die beiden Fragen zum Vertrauen in die EU Institutionen beantwortet haben. Um herauszufinden, wer diese Fragen nicht beantwortet hat, hilft die Funktion `is.na()` aus dem vorherigen Kapitel. Für beide Variablen liefert dieser Befehl eine Liste mit Befragten, die die Frage nicht beantwortet haben. Diesen Personen wird in der Liste ein `TRUE` zugeordnet.

```
# Frage: Fehlt die Angabe zu Vertrauen in die EU-Kommission?
is.na(eu.vertrauen$vertrauenEuKommission)
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE
## [13] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE

# Frage: Fehlt die Angabe zu Vertrauen in das EU-Parlament?
is.na(eu.vertrauen$vertrauenEuParlament)
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

Mithilfe dieser Liste können nun Befragte herausgefiltert werden. Zur Erinnerung: In den eckigen Klammern stehen erst die Zahlen der Zeilen und dann der Spalten. Da alle Spalten behalten werden

sollen, bleibt der Platz nach dem Komma leer. Vor dem Komma wird die Bedingung aufgeführt, nach der gefiltert wird, nämlich: `!is.na(eu.vertrauen$vertrauenEuParlament)`.

Diese Bedingung ist nun verneint. Warum? Beim Filtern muss die Bedingung so formuliert werden, dass jene Befragten ein `TRUE` zugeordnet bekommen, die im Datensatz verbleiben sollen. `is.na()` ordnet den Befragten mit fehlenden Werten ein `TRUE` zu, daher muss die Bedingung mit `!` vereint werden. Nach der Verneinung erhalten die Befragten mit fehlenden Werten ein `FALSE` und verbleiben nach dem Filtern nicht im Datensatz.

```
eu.vertrauen[!is.na(eu.vertrauen$vertrauenEuParlament), ]
##    alter vertrauenEuKommission vertrauenEuParlament
## 1     62                     5                    5
## 2     64                     5                    5
## 3     22                     4                    3
## 4     59                     4                    4
## 5     30                     1                    1
## 6     41                     4                    4
## 8     39                     5                    6
## 9     40                     5                    5
## 10    69                     4                    4
## 11    37                     4                    4
## 13    73                     1                    1
## 14    46                     3                    5
## 15    68                    NA                    4
## 16    53                     3                    3
## 17    27                     6                    6
## 18    25                     4                    4
## 19    68                     5                    6
## 20    42                     4                    4
```

Der Datensatz hat nach dem Filtern nur noch 18 Zeilen. Es wurden also zwei Befragte aus dem Datensatz ausgeschlossen, die einen fehlenden Wert für `eu.vertrauen$vertrauenEuParlament` hatten.

Wenn die Fälle ausgeschlossen werden sollen, die entweder beim Vertrauen in das EU-Parlament oder beim Vertrauen in die EU-Kommission einen fehlenden Wert haben, wird das Zeichen für „Oder"`|` verwendet. Die Bedingung lautet dann: `!is.na(eu.vertrauen$vertrauenEuParlament) | !is.na(eu.vertrauen$vertrauenEuKommission`.

Auch hier werden die einzelnen Aussagen wieder mit `!` verneint - in diesem Fall mit einer Klammer, damit sich die Verneinung auf beide Variablen bezieht.

```
 eu.vertrauen[!(is.na(eu.vertrauen$vertrauenEuParlament)
                | is.na(eu.vertrauen$vertrauenEuKommission)), ]
##    alter vertrauenEuKommission vertrauenEuParlament
## 1     62                     5                    5
## 2     64                     5                    5
## 3     22                     4                    3
## 4     59                     4                    4
```

```
## 5     30                    1                    1
## 6     41                    4                    4
## 8     39                    5                    6
## 9     40                    5                    5
## 10    69                    4                    4
## 11    37                    4                    4
## 13    73                    1                    1
## 14    46                    3                    5
## 16    53                    3                    3
## 17    27                    6                    6
## 18    25                    4                    4
## 19    68                    5                    6
## 20    42                    4                    4
```

Der gefilterten Datensatz wird in dem Objekt `eu.vertrauen.ohne.na` abgespeichert.

```
eu.vertrauen.ohne.na <- eu.vertrauen[is.na(eu.vertrauen$vertrauenEuParlament) |
                                     is.na(eu.vertrauen$vertrauenEuKommission), ]
```

An dieser Stelle sei noch kurz eine Alternative zum Indexing erwähnt: der Befehl `subset()`. Hiermit lässt sich eine Teilmenge (Spalten und/oder Zeilen) des Datensatzes auswählen. Ebenso sei der Befehl `na.omit` an dieser Stelle erwähnt, mit dem alle fehlenden Werte in einem Datensatz entfernt werden können. Dieses Vorgehen ist allerdings oftmals nur bei kleinen Teildatensätzen (wie in diesem Beispiel) sinnvoll, da alle Beobachtungen mit fehlenden Werten über alle Variablen entfernt werden. Die Ergebnisse können verglichen werden.

```
# 1. Variante:
eu.vertrauen <- subset(allbus.kurz, !is.na(pt19) & !is.na(pt20),
                       select = c(age, pt19, pt20))
                # subset-Befehl zur Auswahl der Spalten und Zeilen
# 2. Variante:
eu.vertrauen <- subset(allbus.kurz, select = c(age, pt19, pt20))
                # subset zur Auswahl der Spalten
eu.vertrauen <- na.omit(eu.vertrauen)
                # fehlende Beobachtungen (Zeilen) werden aus dem Datensatz entfernt
```

4.2.3 Kategorien zusammenfassen

Im ALLBUS wird das Vertrauen in Institutionen mithilfe einer 7er-Skala abgefragt, wobei 1 für "gar kein Vertrauen" und 7 für "großes Vertrauen" steht. Gelegentlich (z.B. für die Übersichtlichkeit einer Grafik) kann es sinnvoll sein, die Befragten etwas gröber zu unterscheiden.

Deshalb werden im Folgenden die Befragten in drei Gruppen zusammengefasst. Befragte mit den Codes `1` oder `2` wird "wenig Vertrauen" bescheinigt. Jene mit den Codes `3`, `4` oder `5` ein "mittleres Vertrauen" und solche mit den Codes `6` oder `7` haben in dieser Betrachtung ein "hohes Vertrauen".

Als erstes wird die Variable erstellt und unter dem neuen Namen `pt01_kat` abgespeichert.

Danach müssen die einzelnen Werte der neuen Spalte angesprochen werden und festgelegt werden, wie sie verändert werden. Dazu wird zuerst die Spalte angewählt, also hier `allbus.kurz$pt01_kat` und dann wird mit der urprünglichen Variable `[allbus.kurz$pt01==1]` genau angegeben, welche Werte verändert werden sollen, Diesen wird anschließend mit dem Pfeil `<-` ihre neue Kategorie z.B. "Niedriges Vertrauen" zugeordnet.

Dies wird für alle `7` Werte wiederholt. Zum Schluss wird überprüft, ob alles funktioniert hat, indem die alte und die neue Variable ausgegeben lassen wird. Ein Vergleich der beiden Spalten des Datensatzes zeigt dann, ob die Transformation erfolgreich war.

```
# Im Vorfeld
attributes(allbus.kurz$pt01)$labels # Labels überprüfen
##       KEINE ANGABE GAR KEIN VERTRAUEN                  ..                  ..
##                   -9                   1                   2                   3
##                   ..                  ..                  ..  GROSSES VERTRAUEN
##                    4                   5                   6                   7
table(allbus.kurz$pt01) # Verteilung der Ursprungsvariable überprüfen
##
## 2 3 4 5 6 7
## 1 2 3 5 5 4

# 1. Eine neue Variable "kat" für "kategeoriell" wird im Datensatz erstellt
allbus.kurz$pt01_kat  <- NA # ein fehlender Wert wird für alle Zeilen zugewiesen

# 2. Neue Kategorien zuweisen
allbus.kurz$pt01_kat[allbus.kurz$pt01==1] <- "Niedriges Vertrauen"
allbus.kurz$pt01_kat[allbus.kurz$pt01==2] <- "Niedriges Vertrauen"
allbus.kurz$pt01_kat[allbus.kurz$pt01==3] <- "Mittleres Vertrauen"
allbus.kurz$pt01_kat[allbus.kurz$pt01==4] <- "Mittleres Vertrauen"
allbus.kurz$pt01_kat[allbus.kurz$pt01==5] <- "Mittleres Vertrauen"
allbus.kurz$pt01_kat[allbus.kurz$pt01==6] <- "Hohes Vertrauen"
allbus.kurz$pt01_kat[allbus.kurz$pt01==7] <- "Hohes Vertrauen"

# 3. Ergebnis mit dem Befehl "table" überprüfen
table(allbus.kurz$pt01_kat, allbus.kurz$pt01)
##
##                       2 3 4 5 6 7
##   Hohes Vertrauen     0 0 0 0 5 4
##   Mittleres Vertrauen 0 2 3 5 0 0
##   Niedriges Vertrauen 1 0 0 0 0 0
```

Die Prozedur kann effizienter gestaltet werden, indem die einzelnen Bedingungen für die neuen Kategorien zu jeweils einer Bedingung zusammengefasst wird. Dazu wird für jede Kategorie ein Vektor mit Codes der ursprünglichen Variable definiert. Mit `%in%` wird dann überprüft, ob sich der einzelne Befragte in dem Vektor befindet und deshalb der neuen Kategorie aus dieser Zeile zugeordnet wird.

```
# 1. Eine neue Variable erstellen und Werte zuweisen
allbus.kurz$pt01_kat  <- NA
# 2. Rekodierung durchführen
allbus.kurz$pt01_kat[allbus.kurz$pt01 %in% c(1,2)] <- "Niedriges Vertrauen"
allbus.kurz$pt01_kat[allbus.kurz$pt01 %in% c(3,4,5)] <- "Mittleres Vertrauen"
allbus.kurz$pt01_kat[allbus.kurz$pt01 %in% c(6,7)] <- "Hohes Vertrauen"

# 3. Rekodierung überprüfen
table(allbus.kurz$pt01_kat, allbus.kurz$pt01)
##
##                       2 3 4 5 6 7
##   Hohes Vertrauen     0 0 0 0 5 4
##   Mittleres Vertrauen 0 2 3 5 0 0
##   Niedriges Vertrauen 1 0 0 0 0 0
```

4.2.3.1 Der ifelse-Befehl Oftmals werden in der Datenaufbereitung dichotome Variablen erstellt, sogenannte Dummy-Variablen. Diese tragen in der Regel die Ausprägungen (0,1). Zur Erstellung von Dummy-Variablen, kann der `ifelse`-Befehl genutzt werden. Dieser Befehl unterscheidet zwischen zwei Kategorien. Er bedarf einer Bedingung. Wenn diese zutrifft (if), werden die Befragten in Gruppe 1 einsortiert (yes), Wenn sie nicht zutrifft (else), werden sie in Gruppe 2 einsortiert (no).

Ein Beispiel: Beim Alter sollen die Befragten zu zwei Gruppen zusammengefasst werden: Personen, die jünger als 40 Jahre sind, und Personen, die 40 Jahre alt oder älter sind. Die Bedingung für die Gruppe 1 lautet also: `age < 40`. An diese Bedingung werden dann im Befehl `ifelse()` die beiden Gruppennamen, durch Kommata getrennt, angehängt. Personen, auf die die Bedingung zutrifft, bekommen den Wert (1) oder "jünger als 40", die übrigen Personen erhalten den Wert (0) oder "40 oder älter".

Im Ergebnis kann es dann so aussehen:

```
# Inspizierung der Variable
table(allbus.kurz$age)
##
## 22 25 27 30 37 39 40 41 42 43 46 53 59 62 64 68 69 73 77
##  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  1  1  1

# Dummy-Variable mit 0 und 1
allbus.kurz$age_u40 <-ifelse(allbus.kurz$age < 40, 1, 0)
table(allbus.kurz$age_u40) #Überprüfung des Ergebnisses
##
##  0  1
## 14  6

# kategorielle Variable "as.character"
allbus.kurz$age_u40 <-ifelse(allbus.kurz$age < 40,
                             "jünger als 40","40 oder älter")
table(allbus.kurz$age_u40) #Überprüfung des Ergebnisses
##
```

```
## 40 oder älter jünger als 40
##            14             6
```

4.3 dplyr

4.3.1 Einfühung

Für die Datenaufbereitung mit dem Paket `dplyr` muss das Paket zunächst aufgerufen werden. Dies kann entweder durch die Installation und des anschließenden Ladens des Tidyverse, zu dem das Paket gehört, oder durch das Aufrufen des einzelnen Pakets geschehen.

```
library(dplyr)
```

Das Paket `dplyr` vereinfacht die Datenaufbereitung. Dies liegt vor allem an der Einführung des sogenannten Pipe-Operators `%>%` (dt. "Rohr"). Dieser Operator ermöglicht es, Befehle hintereinander und aufeinander aufbauend zu formulieren und so ein Objekt an die nächste Funktion weiterzugeben. Sprachlich entspricht der Pipe-Operator dem Ausdruck „und dannünd kann so im Code gelesen werden.

Auf den ersten Blick sieht die Verwendung des Operators komplizierter aus als die traditionelle Umsetzung in `Base-R`. Die Verwendung des Operators hat jedoch zwei Vorteile:

- Zu Beginn wird ein Datensatz oder eine Spalte eines Datensatzes bestimmt, auf welche sich dann alle folgenden Befehle beziehen werden.
- Es bedarf keiner komplizierten Verschachtelung von mehreren Befehlen, da die Befehle mithilfe des Operators aufeinader aufbauen. Ein Beispiel: Im vorherigen Kapitel wurde die Anzahl der fehlenden Fälle der Variable pt03 mit dem folgenden verketteten Befehl berechnet.

```
sum(is.na(allbus$pt03))
## [1] 91
```

Mithilfe des `%>%` Pipe-Operators kann diese Verkettung in dem folgenden Befehl aufgelöst werden. In den Kommentaren werden zur Verdeutlichung die einzelnen Schritte als Imperative beschrieben.

```
allbus$pt03 %>% # Nimm die Spalte allbus$pt03 und dann ...
  is.na() %>% # ... gib an, für welchen Befragten der Wert in dieser Spalte fehlt
              # und dann ...
  sum() # ... berechne die Summe der fehlenden Werte.
## [1] 91
```

4.3.2 Spalten auswählen und umbenennen

Außerdem ermöglicht `dplyr` die Nutzung eines Befehls `select()`. Mit diesem können Variablen aus einem Datensatz ausgewählt und umbenannt werden.

Um einzelne Spalten auszuwählen, werden in den Klammern die Namen der Spalten aufgelistet, die berücksichtigt werden sollen. Die Namen der Variablen brauchen hier keine Anführungszeichen, sie müssen jedoch durch Kommata getrennt werden.

```
allbus.kurz %>% # Nimm den Datensatz „allbus.kurz", und dann ...
select(age, pt19, pt20) # ... wähle die Variablen age, pt19, pt20 aus
##    age pt19 pt20
## 1   62    5    5
## 2   64    5    5
## 3   22    4    3
## 4   59    4    4
## 5   30    1    1
## 6   41    4    4
## 7   43   NA   NA
## 8   39    5    6
## 9   40    5    5
## 10  69    4    4
## 11  37    4    4
## 12  77   NA   NA
## 13  73    1    1
## 14  46    3    5
## 15  68   NA    4
## 16  53    3    3
## 17  27    6    6
## 18  25    4    4
## 19  68    5    6
## 20  42    4    4
```

Die Variablennamen können in diesem Befehl auch direkt angepasst werden. Welcher alte Name zu welchem neuen Namen gehört, wird durch das Gleichheitszeichen kenntlich gemacht.

```
allbus.kurz %>%
select(alter = age,
          vertrauenEuKommission  = pt19,
          vertrauenEuParlament  = pt20)
##    alter vertrauenEuKommission vertrauenEuParlament
## 1     62                    5                    5
## 2     64                    5                    5
## 3     22                    4                    3
## 4     59                    4                    4
## 5     30                    1                    1
## 6     41                    4                    4
## 7     43                   NA                   NA
## 8     39                    5                    6
## 9     40                    5                    5
## 10    69                    4                    4
## 11    37                    4                    4
## 12    77                   NA                   NA
```

```
## 13    73                    1                    1
## 14    46                    3                    5
## 15    68                   NA                    4
## 16    53                    3                    3
## 17    27                    6                    6
## 18    25                    4                    4
## 19    68                    5                    6
## 20    42                    4                    4
```

Am besten wird der neue Datensatz direkt durch eine Zuweisung abgespeichert, um später darauf zugreifen zu können.

```
allbus.eu <- allbus.kurz %>%
select(alter = age,
          vertrauenEuKommission  = pt19,
          vertrauenEuParlament  = pt20)
```

4.3.3 Zeilen filtern

Wenn mit `dplyr` bestimme Befragte aus der Analyse ausgeschlossen werden sollen, hilft der Befehl `filter()` weiter. In diesem Befehl können eine oder mehrere Bedingungen, die auf die Befragten zutreffen müssen, verwendet werden. Bei mehreren Bedingungen müssen diese durch ein Komma getrennt werden.

Da der Datensatz zu Beginn genannt wird, können die Variablennamen ohne die Angabe des Datensatzes aufgeführt werden. Alle Variablennamen im folgenden Code beziehen sich automatisch auf den ausgewählten Datensatz. Wenn diejenigen Befragten aus dem Datensatz allbus.eu herausgefiltert werden sollen, die eine der beiden Fragen zum Vertrauen in die EU nicht beantwortet haben, funktioniert es wie folgt:

```
allbus.eu %>%
    filter(! is.na(vertrauenEuKommission),
        ! is.na(vertrauenEuParlament))
##    alter vertrauenEuKommission vertrauenEuParlament
## 1     62                    5                    5
## 2     64                    5                    5
## 3     22                    4                    3
## 4     59                    4                    4
## 5     30                    1                    1
## 6     41                    4                    4
## 7     39                    5                    6
## 8     40                    5                    5
## 9     69                    4                    4
## 10    37                    4                    4
## 11    73                    1                    1
## 12    46                    3                    5
## 13    53                    3                    3
```

```
## 14    27                      6                      6
## 15    25                      4                      4
## 16    68                      5                      6
## 17    42                      4                      4
```

Auf eine ähnliche Weise kann der Datensatz auch auf Befragte beschränkt werden, die älter als 40 Jahre sind.

```
allbus.eu %>%
    filter(alter > 40)
##    alter vertrauenEuKommission vertrauenEuParlament
## 1     62                      5                      5
## 2     64                      5                      5
## 3     59                      4                      4
## 4     41                      4                      4
## 5     43                     NA                     NA
## 6     69                      4                      4
## 7     77                     NA                     NA
## 8     73                      1                      1
## 9     46                      3                      5
## 10    68                     NA                      4
## 11    53                      3                      3
## 12    68                      5                      6
## 13    42                      4                      4
```

Hinweis: Mit dem Paket `dplyr` können mehrere Befehle miteinander verbunden werden. So kann ein Filterbefehl direkt an den Befehl `select()` angehängt werden. Dies ist sinnvoll, da Zwischenergebnisse so nicht mehr als separates Element gespeichert werden müssen. Ebenso empfiehlt sich eine Zuweisung vorzunehmen, damit mit dem neuen Datensatz weitergearbeitet werden kann:

```
allbus.eu.ohne.na <- allbus.kurz %>%
select(alter = age,
          vertrauenEuKommission  = pt19,
          vertrauenEuParlament  = pt20) %>%
    filter(! is.na(vertrauenEuKommission),
        ! is.na(vertrauenEuParlament))
```

4.3.4 Kategorien zusammenfassen

Ähnlich wie bei `Base-R` muss zu Beginn der Datentransformation die neue Spalte festgelegt werden, in der die neuen Informationen abgespeichert werden. Dies übernimmt der Befehl `mutate()`. Dieser ergänzt eine neue Variable im Datensatz. Der Datensatz wird durch diesen Befehl "mutiert", d.h. um eine weitere Variable ergänzt und damit verändert und erweitert. In den Klammern des Befehls wird dann zuerst der neue Variablenname und dann nach einem `=` die Transformationsvorschrift festgelegt.

Wenn Kategorien von Variablen geändert und zusammengefasst werden sollen, kann einerseits der Befehl `ifelse()` aus `Base-R` innerhalb von `dplyr` angewendet werden. Es ist jedoch auch möglich,

den Befehl `case_when()` zu nutzen, vor allem wenn beabsichtigt ist, mehr als zwei Zielkategorien zu erstellen.

4.3.4.1 Der ifelse-Befehl innerhalb von dplyr

Dieser Befehl unterscheidet anhand einer Bedingung zwischen zwei Ausprägungen. Wenn die Bedingung zutrifft, werden die Befragten in Gruppe 1 einsortiert. Wenn sie nicht zutrifft, werden sie in Gruppe 2 einsortiert.

Beim Alter sollen die Befragten zu zwei Gruppen zusammengefasst werden: Personen jünger als 40 Jahre und Personen, die 40 Jahre alt oder älter sind. Die Bedingung für die Gruppe 1 lautet also erneut: `age < 40`. An diese Bedingung werden im Befehl `ifelse()` die beiden Gruppennamen, durch Kommata getrennt, angehängt. Personen, auf die die Bedingung zutrifft, bekommen erneut den Wert "jünger als 40", die übrigen Personen wird die Kategorie "40 oder älter" zugeordnet.

Im Ergebnis kann dann so aussehen:

```
allbus.kurz %>%
  mutate(age2 = ifelse(age < 40,
                       "jünger als 40",
                       "40 oder älter")) %>%
  select(age, age2)
##    age          age2
## 1   62 40 oder älter
## 2   64 40 oder älter
## 3   22 jünger als 40
## 4   59 40 oder älter
## 5   30 jünger als 40
## 6   41 40 oder älter
## 7   43 40 oder älter
## 8   39 jünger als 40
## 9   40 40 oder älter
## 10  69 40 oder älter
## 11  37 jünger als 40
## 12  77 40 oder älter
## 13  73 40 oder älter
## 14  46 40 oder älter
## 15  68 40 oder älter
## 16  53 40 oder älter
## 17  27 jünger als 40
## 18  25 jünger als 40
## 19  68 40 oder älter
## 20  42 40 oder älter
```

Die Transformation war erfolgreich, daher wird nun der ganze Datensatz in einem neuen Objekt abgespeichert, um ihn später weiter zu verwenden.

```
allbus.alter <- allbus.kurz %>%
  mutate(age2 = ifelse(age < 40,
                       "jünger als 40",
                       "40 oder älter"))
```

4.3.4.2 Der case_when-Befehl

Wenn mit `dplyr` eine Variable in mehr als zwei Kategorien umgewandelt werden soll, bietet sich der Befehl `case_when()` an. Mit diesem werden mehreren Bedingungen jeweils eine Kategorie zugeordnet. Die Bedingungen und die Kategorie werden durch eine Tilde `~` getrennt . Wenn also, wie im Base-R-Beispiel, im Folgenden die Befragten in drei Gruppen anhand ihrer Antwort zu `pt01` zusammengefasst werden sollen, wird bei dplyr der Befehl `case_when()` verwendet. Zur Erinnerung: Befragte mit den Codes `1` oder `2` soll "niedriges Vertrauen" bescheinigt werden. Jene mit den Codes `3`, `4` oder `5` haben "mittleres Vertrauen" und jene mit den Codes `6` oder `7` haben in dieser Betrachtung ein "hohes Vertrauen".

Wie auch schon beim `ifelse()`-Befehl, muss vor der Transformation mit `mutate()` eine neue Spalte im Datensatz erstellt werden.

```
allbus.kurz %>%
  mutate(pt01_neu = case_when(pt01 %in% c(1,2) ~ "Niedriges Vertrauen",
                              pt01 %in% c(3,4,5) ~ "Mittleres Vertrauen",
                              pt01 %in% c(6,7) ~ "Hohes Vertrauen")) %>%
  select(pt01, pt01_neu)
##    pt01            pt01_neu
## 1     6     Hohes Vertrauen
## 2     6     Hohes Vertrauen
## 3     5 Mittleres Vertrauen
## 4     7     Hohes Vertrauen
## 5     5 Mittleres Vertrauen
## 6     5 Mittleres Vertrauen
## 7     6     Hohes Vertrauen
## 8     3 Mittleres Vertrauen
## 9     6     Hohes Vertrauen
## 10    5 Mittleres Vertrauen
## 11    4 Mittleres Vertrauen
## 12    4 Mittleres Vertrauen
## 13    4 Mittleres Vertrauen
## 14    5 Mittleres Vertrauen
## 15    6     Hohes Vertrauen
## 16    7     Hohes Vertrauen
## 17    7     Hohes Vertrauen
## 18    7     Hohes Vertrauen
## 19    3 Mittleres Vertrauen
## 20    2 Niedriges Vertrauen
```

5 Deskriptive Statistiken

5.1 Häufigkeitsverteilungen

Ein Ziel der sozialwissenschaftlichen Forschung ist es, die Wirklichkeit zu beschreiben. Dazu werden oftmals Häufigkeitsverteilungen betrachtet. Neben einem genuinen Interesse an der sozialen Wirklichkeit, ist die Durchführung deskriptiver Analysen oftmals ein notwendiger Start bevor weitere (erklärende) Analysen durchgeführt werden. In diesem Kapitel wird mit der univariaten Analyse begonnen, also der Verteilung einer einzelnen Variablen. Diese können in tabellarischer oder grafischer Form sowie in statistischen Maßzahlen dargestellt werden.

Für die univariate Analyse werden die absolute Häufigkeit, die relative Häufigkeit und die prozentuale Häufigkeit genutzt. Dabei ist zu beachten, welches Skalenniveau und wie viele Ausprägungen die Variable von Interesse hat.

5.2 Absolute Häufigkeiten

Im ALLBUS wird erhoben, ob die Befragten in Ost- oder Westdeutschland leben (Variable `eastwest`). Die absoluten Häufigkeiten (je Ost oder West) werden mit `table()` angefordert:

```
# Häufigkeitsverteilung Ost/West
table(allbus2018$eastwest)
##
##    1    2
## 2387 1090
```

Die Häufigkeitsauszählung ergibt, dass 2387 Befragte aus West- und 1090 Befragte aus Ost-Deutschland kommen. Die Wertelabels können zusätzlich mit `attributes()` angefordert werden. Neben den Wertelabels fordert die Funktion auch die Variablenklasse und weitere Metainformationen an. Durch den Zusatz `$labels` wird die Ausgabe auf die Wertelabels beschränkt.

```
# Wertelabels
attributes(allbus2018$eastwest)$labels
## ALTE BUNDESLAENDER NEUE BUNDESLAENDER
##                  1                  2
```

Mit `sort()` werden die Ausprägungen aufsteigend sortiert

```
sort(table(allbus2018$eastwest))
##
##    2    1
## 1090 2387
```

Nun wird die Kategorie, die am wenigsten vorkommt, als erstes genannt. Mit dem Argument `decreasing = TRUE` wird die Reihenfolge umgekehrt:

```
sort(table(allbus2018$eastwest), decreasing = TRUE)
##
##    1    2
## 2387 1090
```

5.3 Relative Häufigkeiten

Die relative Häufigkeit stellt die absolute Häufigkeit in Bezug zur Fallzahl dar. Der `table()`-Befehl wird hierfür mit `prop.table()` erweitert:

```
prop.table(table(allbus2018$eastwest))
##
##         1         2
## 0.6865114 0.3134886
```

5.4 Prozentuale Häufigkeit

Die prozentuale Häufigkeit (=heißt "von hundert") wird ausgegeben, indem die relative Häufigkeit mit 100 multipliziert wird:

```
prop.table(table(allbus2018$eastwest))*100
##
##        1        2
## 68.65114 31.34886
```

Mit dem zusätzlichen Befehl `round()` und dem entsprechenden Argument `digits = 2` wird die prozentuale Häufigkeit auf zwei Nachkommastellen gerundet:

```
round((prop.table(table(allbus2018$eastwest))*100), digits = 2)
##
##     1     2
## 68.65 31.35
```

Der Anteil der Befragten aus Westdeutschland beträgt etwa 69 Prozent und aus Ostdeutschland etwa 31 Prozent. Damit sind Befragte aus Ostdeutschland in der Stichprobe überrepräsentiert (Stichwort Oversampling des ALLBUS, siehe Kapitel 13 zum Umgang mit Design-Gewichten).

Zusätzlich lassen sich mit `cumsum()` die kumulierten Anteile berechnen. Kumuliert bedeutet, dass die jeweiligen Anteile in der Summe zu 100 Prozent addiert werden:

```
cumsum(prop.table(table(allbus2018$eastwest))*100)
##         1         2
##  68.65114 100.00000
```

5.5 Häufigkeitstabellen

Die absoluten, relativen, prozentualen und kumulierten Häufigkeiten können aber auch in einem Objekt gespeichert werden. Dafür wird die `cbind()`-Funktion genutzt, um die einzelnen Vektoren in einem Objekt zusammenzuführen:

```
absolut <- table(allbus2018$eastwest)
relativ <- prop.table(absolut)
prozentual <- 100 * relativ
kumuliert <- cumsum(prozentual)

# in Objekt speichern

haeufigkeiten <- cbind(absolut, relativ, prozentual, kumuliert)
print(haeufigkeiten)
##   absolut   relativ prozentual kumuliert
## 1    2387 0.6865114   68.65114  68.65114
## 2    1090 0.3134886   31.34886 100.00000
```

5.6 Klassifizierung

Aufgrund der vielen Ausprägungen kann die Altersverteilung (nach Geburtsjahr) nicht sinnvoll per `table()`-Befehl untersucht werden. Um dennoch einen Überblick zur Verteilung zu erhalten, können die Geburtsjahre als Geburtskohorten klassifiziert werden. Diese neue Variable `age_class` soll die Klasse `factor` haben. Damit die Alterskategorien die richtige Reihenfolge aufweisen, muss mit der Variable `age_level` aber zunächst die Reihenfolge vorgegeben werden. Anschließend werden Bedingungen formuliert, um die Geburtsjahre den jeweiligen Kohorten zuzuordnen:

```
# Klassifizierung der Geburtsjahre
age_level <- c("unter 20", "20 bis 29", "30 bis 39",
               "40 bis 49", "50 bis 59", "60 bis 69", "über 70")

# Initiierung der Variable
allbus2018$age_class <- NA

allbus2018$age_class[allbus2018$age < 20] <- "unter 20"
allbus2018$age_class[allbus2018$age >= 20 & allbus2018$age < 30] <- "20 bis 29"
allbus2018$age_class[allbus2018$age >= 30 & allbus2018$age < 40] <- "30 bis 39"
allbus2018$age_class[allbus2018$age >= 40 & allbus2018$age < 50] <- "40 bis 49"
allbus2018$age_class[allbus2018$age >= 50 & allbus2018$age < 60] <- "50 bis 59"
allbus2018$age_class[allbus2018$age >= 60 & allbus2018$age < 70] <- "60 bis 69"
allbus2018$age_class[allbus2018$age > 70] <- "über 70"

# als Faktor mit vorgegebener Reihenfolge
allbus2018$age_class <- factor(allbus2018$age_class, levels = age_level)
```

Schließlich wird die Verteilung der Altersvariablen ausgegeben.

```
table(allbus2018$age_class)
##
##  unter 20 20 bis 29 30 bis 39 40 bis 49 50 bis 59 60 bis 69   über 70
##        63       412       511       532       727       641       540
```

6 Statistische Maßzahlen

6.1 Mittelwerte als Maße der zentralen Tendenz

Mittelwerte informieren über die "Mitte" einer Verteilung. Im Alltag spielt häufig der Durchschnitt eine wichtige Rolle, zum Beispiel das durchschnittliche Einkommen oder die durchschnittliche Lebenserwartung. In der Regel ist mit dem Durchschnitt das arithmetische Mittel gemeint. Es gibt allerdings auch andere Mittelwerte wie den Median oder den Modus.

6.1.1 Arithmetisches Mittel

Das arithmetische Mittel kann nur für (quasi-)metrische Variablen berechnet werden. Da das arithmetische Mittel alle Ausprägungen einer Variablen beinhaltet, beeinflussen insbesondere Extremwerte die Kennzahl. Das arithmetische Mittel wird in R mittels der `mean()`-Funktion berechnet. Da die `mean()`-Funktion eine Fehlermeldung ausgibt, wenn fehlende Fälle in den Daten vorhanden sind, muss zusätzlich das Argument `na.rm = TRUE` angefügt werden. Damit lässt sich das mittlere Lebensalter der Befragten im ALLBUS berechnen:

```
mean(allbus2018$age, na.rm = TRUE)
## [1] 51.67713
```

6.1.2 Median

Der zweite Mittelwert ist der Median. Das Skalenniveau, das der Median voraussetzt, ist mindestens ordinal. Da die Berechnung auf der Rangfolge der Variablenausprägungen basiert, ist der Median weniger anfällig für Extremwerte; die Abstände zwischen den jeweiligen Ausprägungen spielen bei der Berechnung keine Rolle. Ist die Anzahl der Ausprägungen ungerade, kann der Median problemlos ermittelt werden, weil die Reihe genau an diesem geteilt werden kann. Wenn die Anzahl der Ausprägungen gerade ist, dann liegt der Median im Intervall der zwei mittleren Zahlen. Auch die `median()`-Funktion muss bei fehlenden Werten mit dem zusätzlichen Argument `na.rm = TRUE` erweitert werden.

```
median(allbus2018$age, na.rm = TRUE)
## [1] 53
```

Der Median liegt mit 53 Jahren etwas höher als das arithmetische Mittel.

6.1.3 Modus

Die häufigste Ausprägung einer mindestens nominalskalierten Variable ist der Modus. Der Modus ist der häufigste Wert und damit "typisch" für eine Verteilung. Dazu reicht bereits der `table()`-Befehl aus, der bei einer metrischen Variable allerdings schnell unübersichtlich wird.

Um den Modalwert auszugeben, braucht es jedoch eine Verkettung von Funktionen. Für eine bessere Übersichtlichkeit, wird dazu die Wertetabelle als Objekt angelegt. Anschließend wird mit die `names()`-Funktion auf die Tabelle angewendet und mit einer Indizierung verknüpft. In der

Indizierung wird dabei eine Funktion beschrieben, die den Wert mit der häufigsten Ausprägung finden soll. Die untenstehende Funktion kann vereinfacht auch folgendermaßen beschreiben werden: Zeige mir den Namen der Ausprägung an, welche am häufigsten genannt wurde.

```
modal_tabelle <- table(allbus2018$age)
names(modal_tabelle)[which(modal_tabelle==max(modal_tabelle))]
## [1] "55"
```

Das Ergebnis zeigt, dass die häufigste Ausprägung der Altersvariablen im Datensatz 55 Jahre ist.

Soll die Ausprägung angezeigt werden, welche am seltensten vorkommt, so wird das `max` durch `min` ersetzt.

```
names(modal_tabelle)[which(modal_tabelle==min(modal_tabelle))]
## [1] "91" "95"
```

In diesem Fall gibt es zwei Ausprägungen (91 und 95 Jahre), die am seltesten in der Altersvariable vorkommen.

6.1.4 Quantile

Abschließend können Quantile als sogenanntes Lagemaß betrachtet werden. Wie schon beim Median muss eine Variable dabei mindestens ein ordinales Skalenniveau aufweisen und somit prinzipiell nach Größe sortiert werden können.

Beim Median wurden die Ausprägungen in zwei gleiche Hälften unterteilt, was einem 50-Prozent-Quantil entspricht. Es können allerdings auch 10-Prozent, 20-Prozent, 30-Prozent etc. Quantile berechnet werden. Manche Quantile haben eine bestimmte Bezeichnung wie das Quartil, das die Ausprägungen in vier gleiche Gruppen teilt. Per Standardeinstellung gibt die Funktion `quantile()` Quartile aus:

```
quantile(allbus2018$age, na.rm = TRUE) #quartile
##   0%  25%  50%  75% 100%
##   18   37   53   65   95
```

Die unteren 25 Prozent der Befragten sind bis 37 Jahre alt, die zweiten 25 Prozent sind bis 53 Jahre alt, die dritten 25 Prozent sind bis 65 Jahre alt und die oberen 25 Prozent sind bis 95 Jahre alt.

Weitere bekannte Quantile sind Dezile (10 Teile) und Perzentile (100 Teile). Dafür muss die Funktion mit `probs = seq` erweitert werden.

```
quantile(allbus2018$age, na.rm = TRUE, probs = seq(0, 1, 0.1)) #dezile
##   0%  10%  20%  30%  40%  50%  60%  70%  80%  90% 100%
##   18   27   34   41   48   53   57   62   68   76   95
```

Sollen Perzentile berechnet werden, so wird in der Sequenzierung der Wert `0.1` durch den Wert `0.01` ersetzt. Aufgrund der Größe des Outputs wird hier auf eine Darstellung des Ergebnisses verzichtet.

6.2 Streuungsmaße

Streuungsmaße sind Maßzahlen, die aufzeigen, wie sehr sich die einzelnen Beobachtungen in der Stichprobe voneinander unterscheiden. Die relevanten Maßzahlen in diesem Kapitel sind folgende:

- Spannweite
- Quartilsabstand
- Varianz
- Standardabweichung

6.2.1 Spannweite

Um die Spannweite (range) der Verteilung herauszufinden, berechnet man den Unterschied zwischen Maximal- und Minimalwert einer Verteilung.

Die Spannweite wird beispielhaft am Alter der befragten Personen gezeigt. Sie stellt dabei die Unterschiede zwischen der jüngsten (Minimum) und ältesten (Maximum) befragten Person dar. In R werden diese über die Befehle `min()` und `max()` angefordert. Zunächst wird daher der niedrigste und höchste Wert der Verteilung ausgegeben. Wiederum muss `na.rm=TRUE` ergänzt werden um fehlende Werte vor der Berechnung auszuschließen.

```
max(allbus$age, na.rm = TRUE)
## <labelled<double>[1]>: ALTER: BEFRAGTE(R)
## [1] 95
##
## Labels:
##  value             label
##    -32 NICHT GENERIERBAR
min(allbus$age, na.rm = TRUE)
## <labelled<double>[1]>: ALTER: BEFRAGTE(R)
## [1] 18
##
## Labels:
##  value             label
##    -32 NICHT GENERIERBAR
```

Die älteste Person im ALLBUS ist 95 Jahre alt und erwartungsgemäß entspricht das Alter der jüngsten Person im Datensatz 18 Jahre. Die Spannweite wird nun wie folgt berechnet:

```
max(allbus$age, na.rm = TRUE) - min(allbus$age, na.rm = TRUE) #berechnet Differenz
## [1] 77
```

Die Spannweite, die für das Alter berechnet wurde, ist 77 Jahre. Die älteste befragte Person ist 77 Jahre älter als die jüngste befragte Person.

Soll in einem Befehl nur die Extremwerte der Verteilung ausgeben werden, so kann auch einfach die `range()` Funktion benutzt werden.

```
range(allbus$age, na.rm = TRUE) # gibt Min/Max der Spannweite an
## <labelled<double>[2]>: ALTER: BEFRAGTE(R)
## [1] 18 95
##
## Labels:
##  value             label
##    -32 NICHT GENERIERBAR
```

Die Spannweite gibt lediglich Informationen über die Abstände zwischen den Extrempunkten einer Variablen. In vielen Fällen interessieren allerdings detailliertere Informationen zur Verteilung für die auf andere Maßzahlen zurückgegriffen werden muss.

6.2.2 Quartilsabstand

Die Berechnung des Quartilsabstandes als Streuungsmaß zur Überprüfung der Verteilung hat den Vorteil, dass dieser als Maßzahl unempfindlich für vorhandene Extremwerte in der Verteilung ist. Mit diesem wird also (wie bereits bei der Berechnung des Medians im Vergleich zum arithmetischen Mittel) das Problem beachtet, dass es in einer Verteilung starke Ausreißer geben kann, die zu verzerrten Ergebnissen beitragen.

Für den Quartilsabstand wird sich angeschaut, welche Werte innerhalb der mittleren 50 Prozent der Verteilung liegen und verzichtet gezielt auf die Extreme, um einen robusteren Eindruck der Verteilung zu bekommen. Dazu werden zunächst das 25. und das 75. Quartil bestimmt. Schließlich wird der Abstand zwischen diesen Quartilen berechnet.

Um diese in R zu berechnen, werden zunächst – wie bereits in Kapitel 6.1.4 beschrieben – die Quartile mit der `quantile()` Funktion bestimmt. Die Funktion `IQR()` gibt anschließend den Abstand der Quartilsgrenzen aus.

```
quantile(allbus$age, c(.25, .75), na.rm = TRUE)
## 25% 75%
##  37  65
IQR(allbus$age, na.rm = TRUE)
## [1] 28
```

Die berechnete Spannweite war bei der Variable Alter mit 77 Jahren relativ groß. Nachdem das 1. und 3. Quartil berechnet wurde – also die Werte der Altersverteilung unter denen 25% der Jüngsten (1. Quartil) und über denen 25% der Ältesten befragten Personen (3. Quartil) liegen – wird sichtbar, dass 50% der Personen im ALLBUS zwischen 37 und 65 Jahre alt sind. Die IQR (also der Abstand zwischen 1. und 3. Quartil) liegt damit bei 28 Jahren. Das heißt, dass 50 Prozent der im ALLBUS befragten Personen alterstechnisch 28 oder weniger Jahre auseinander liegen.

6.2.3 Varianz

Eine weitere Maßzahl, die in der Datenauswertung häufig Anwendung findet, ist die Varianz. Sie ist ein Maß für die Streuung einer Verteilung um ihren Mittelwert. Berechnet wird die Varianz als Quadrat der Abweichungen der individuellen Werte vom arithmetischen Mittel der Verteilung. Sie

nimmt damit bei großen Abweichungen sehr hohe und bei kleinen Abweichungen sehr geringe Werte an. Durch die Quadrierung der Werte kann der Varianzwert jedoch inhaltlich nicht wirklich sinnvoll interpretiert werden, da dieser nicht mehr auf dem originären Werteniveau betrachtet werden kann.

An der Altersverteilung im ALLBUS wird beispielhaft gezeigt, wie die Varianz in R berechnet wird. Die Funktion für die Berechnung wird in Base-R bereitgestellt.

```
var(allbus$age, na.rm = TRUE)
## [1] 311.2478
```

Die Varianz der Altersvariablen der Allbusbefragung 2018 beträgt 311,2478.

Anhand der Kennzahlen wird sichtbar, dass die Streuung der Altersvariable bei männlichen Befragten etwas höher ist.

```
var(allbus$age[allbus$sex == 1], na.rm = TRUE)
## [1] 320.6575
var(allbus$age[allbus$sex == 2], na.rm = TRUE)
## [1] 301.4771
```

6.2.4 Standardabweichung

Die Standardabweichung ist die Wurzel aus der Varianz und das wichtigste Streuungsmaß in der sozialwissenschaftlichen Datenanalyse. Sie gibt an, wie weit die einzelnen Werte im Durchschnitt vom arithmetischen Mittel entfernt sind. In R wird zum Berechnen der Standardabweichung die Funktion `sd()` genutzt.

```
sd(allbus$age, na.rm = TRUE)
## [1] 17.64222
```

Im ALLBUS 2018 weicht das Alter der Befragten durchschnittlich etwa 18 Jahre vom durchschnittlichen Alter eines Befragten ab. Die Zahl sagt damit aus, in welchem Bereich die Daten um den Mittelwert streuen und sind daher wichtig für das Verständnis der Verteilung.

Die Werte sind dabei abhängig von der Skalierung der jeweiligen Variable. Wenn beispielsweise das Alter der Befragten nicht in Jahren, sondern in Monaten angegeben wäre, dann würde dies in anderen (höhere) Werten für die Standardabweichung resultieren.

7 Datenvisualisierung

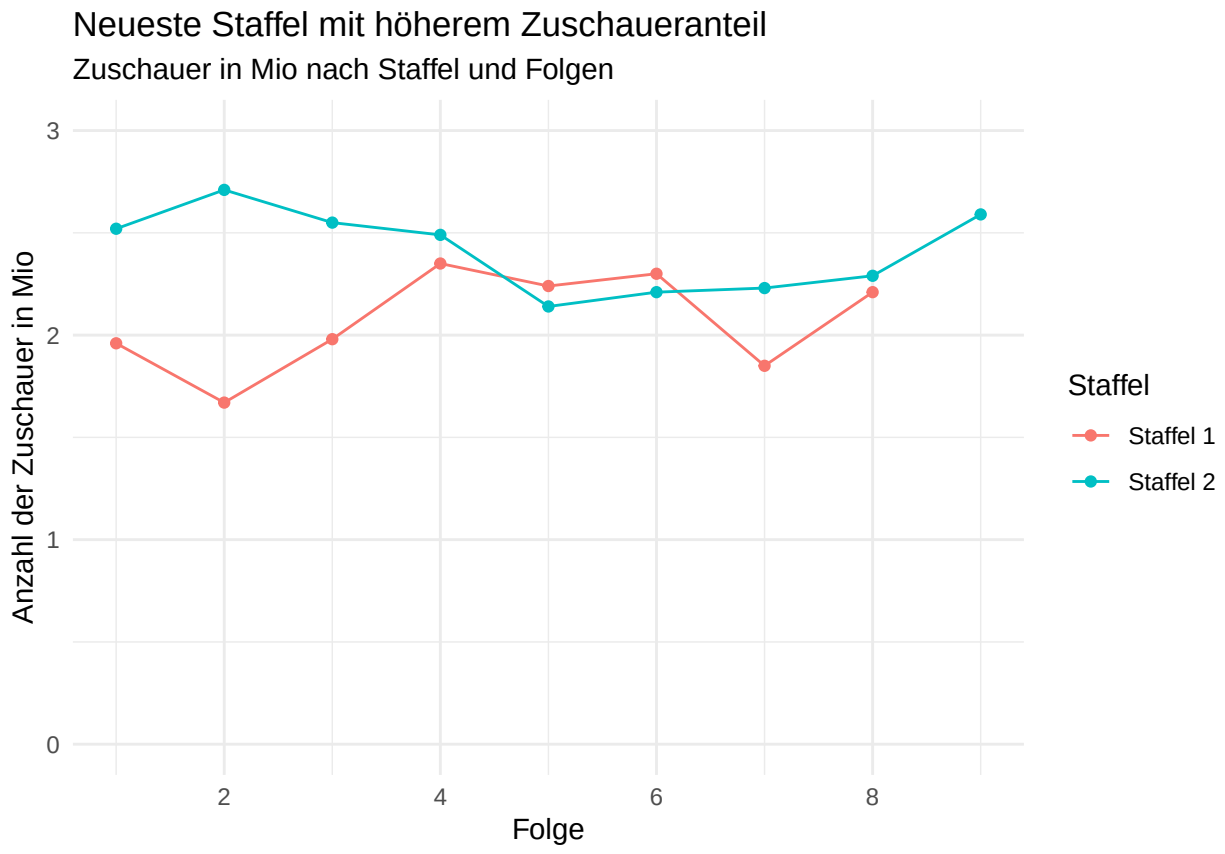
Als Teil der `tidyverse` bietet das Package `ggplot` die Möglichkeit, flexible und grafisch ansprechende Diagramme zu erstellen. Das gg in `ggplot` steht für "Grammar of Graphics". Das heißt, `ggplot` steht für ein Konzept / eine Grammatik, wie Grafiken aufgebaut werden können. Dieses Konzept ist für jede Art von Grafiken ähnlich.

Die grundsätzliche Idee ist, dass verschiedene Code-Schichten übereinanander geleget werden. Diese Ebenen beschreiben die Eigenschaften der Grafik. Die beiden wichtigsten Ebenen heißen:

- aesthetic mappings (aes): Sie definieren, welche Variablen in der Grafik dargestellt werden, z.B. x, y, col, fill
- geometric elements (geom): Sie definieren, welche geometrischen Formen gewählt werden, um die Daten darzustellen, z.B. geom_point, geom_line, geom_bar, geom_histogram

7.1 Basiselemente eines Diagramms

In den folgenden Analysen werden Quotendaten von zwei Staffeln einer Sendung untersucht. Das Ziel der kommenden Berechnungen ist die folgende Grafik.



Als Erstes müssen dazu die nötigen Daten angelegt werden.

Im Folgenden werden nun zunächst die Variablen Zuschaueranzahl in Millionen (alleMio), die Nummer der Folge (Folge) sowie die entsprechende Staffel (Staffel) angelegt. Diese Variablen werden später alle für die Erstellung der Grafik benötigt.

Schließlich werden die Daten ausgegeben, um eine Idee über die Struktur des Datensatzes zu erhalten.

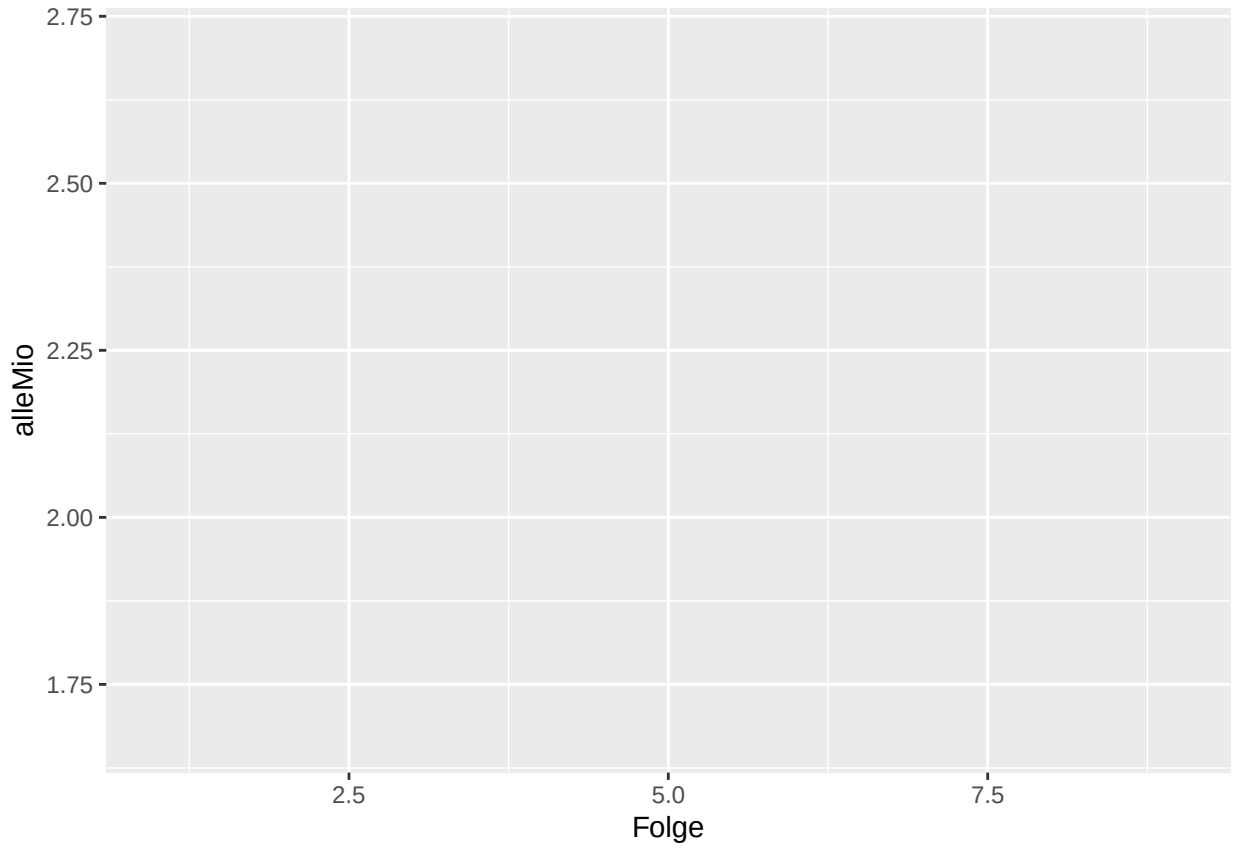
```
alleMio <- c(1.96, 1.67, 1.98, 2.35, 2.24,2.3, 1.85, 2.21, 2.52, 2.71, 2.55, 2.49,
             2.14, 2.21, 2.23, 2.29, 2.59)
Folge <- c(1, 2, 3, 4, 5, 6, 7, 8, 1, 2,3, 4,5, 6, 7, 8, 9)
Staffel <- c("Staffel 1", "Staffel 1",  "Staffel 1",
             "Staffel 1", "Staffel 1", "Staffel 1",
             "Staffel 1", "Staffel 1", "Staffel 2",
             "Staffel 2", "Staffel 2", "Staffel 2",
             "Staffel 2", "Staffel 2", "Staffel 2",
             "Staffel 2","Staffel 2")
Quoten <- data.frame(alleMio, Folge, Staffel)
print(Quoten)
##    alleMio Folge   Staffel
## 1     1.96     1 Staffel 1
## 2     1.67     2 Staffel 1
## 3     1.98     3 Staffel 1
## 4     2.35     4 Staffel 1
## 5     2.24     5 Staffel 1
## 6     2.30     6 Staffel 1
## 7     1.85     7 Staffel 1
## 8     2.21     8 Staffel 1
## 9     2.52     1 Staffel 2
## 10    2.71     2 Staffel 2
## 11    2.55     3 Staffel 2
## 12    2.49     4 Staffel 2
## 13    2.14     5 Staffel 2
## 14    2.21     6 Staffel 2
## 15    2.23     7 Staffel 2
## 16    2.29     8 Staffel 2
## 17    2.59     9 Staffel 2
```

7.1.1 Achsen definieren

Im ersten Schritt der Grafikerstellung müssen die grundlegenden Eigenschaften des Plots definiert werden. Zuallererst sind dies die Daten, auf die sich die Grafik beziehen wird, also hier der Datensatz `Quoten`. Dieser wird als erstes innerhalb des Befehls `ggplot()` definiert.

Im Anschluss wird innerhalb des Befehls `ggplot()` ein `aes()` eingefügt. Dort werden die sogenannten aesthetic mappings festgelegt. Das heißt, es wird angegeben, welche Variable auf welcher Achse dargestellt werden soll. Anhand des Beispiel-Plots oben ist klar, welche Variable auf der x-Achse und welche auf der y-Achse erscheinen soll.

```
ggplot(Quoten,
       aes(x= Folge,
           y = alleMio))
```

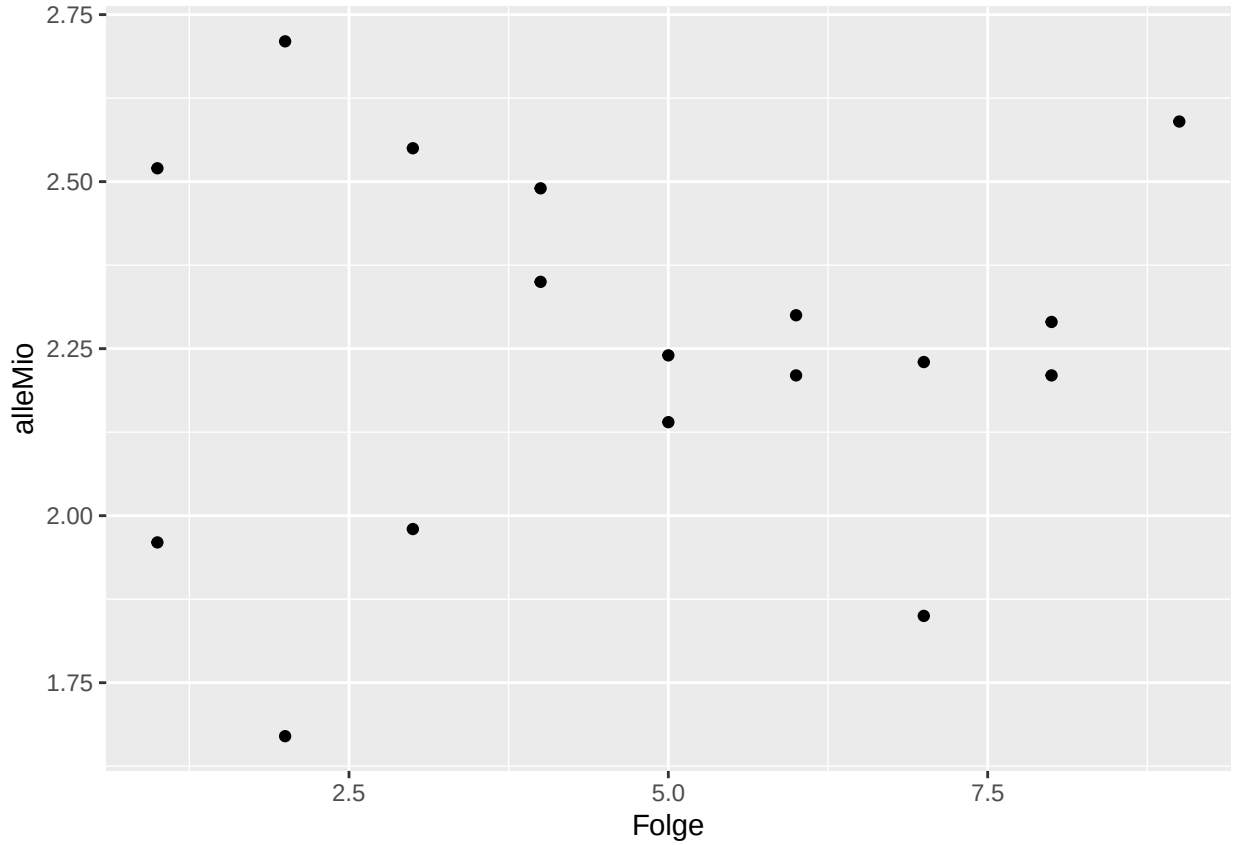


Das Ergebnis ist ein leerer Plot auf dem nur die beiden Achsen abgebildet sind. Dieser wird nun mit weiteren Befehlen erweitert.

7.1.2 Datenpunkte

Auf diesem ersten Plot mit den Achsen wird nun eine weitere Daten-Schicht abgebildet. Mit dem Befehl `geom_point()` können Datenpunkte als zweite Schicht hinzugefügt werde. Der Befehl wird einfach mit einem `+` an den ersten Befehl angehängt. Die Werte der Datenpunkte orientieren sich an der vorher definierten Achsen-Schicht - das heißt der x-Wert ist hier die einzelne Folge und der y-Wert die Anzahl aller Zuschauer.

```
ggplot(Quoten,
       aes(x= Folge,
           y = alleMio)) +
  geom_point()
```

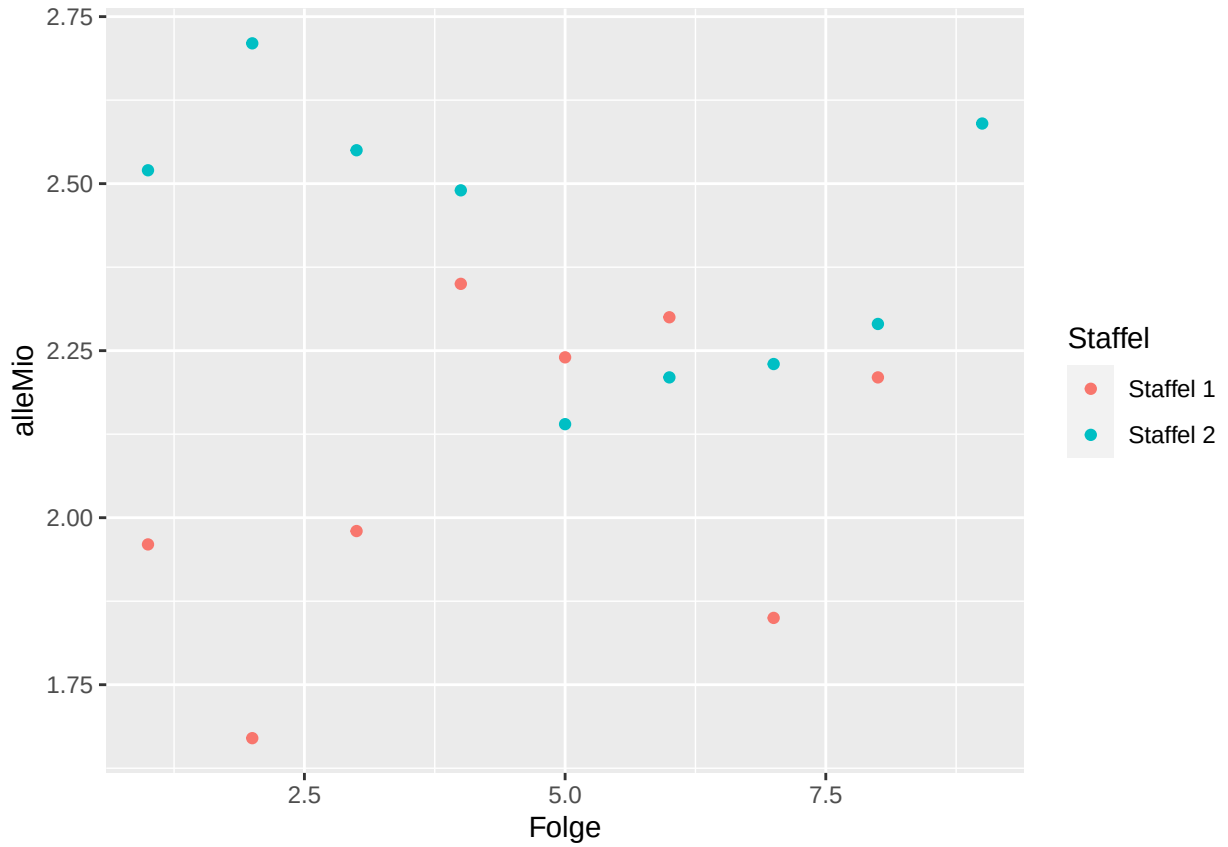


Das Ergebnis ist ein Plot, der im Groben den Trend der Zuschauerzahlen anzeigt. Eine wichtige Information fehlt jedoch: Es ist noch nicht klar, welcher Datenpunkt zu welcher Staffel gehört und es kann noch nicht gesagt werden, welche Staffel besser abgeschnitten hat. Dieses Problem wird als Nächstes behoben.

7.1.3 Gruppenvergleiche

Mit dem Argument `aes()` können nicht nur die Achsen der Grafik festgelegt werden, sondern es kann auch bestimmt werden, welche Variablen farbig als Gruppen dargestellt werden sollen. Die Gruppierungsvariable kann je nach Darstellungsform (`geom_`) entweder mit dem Argument `color =` oder dem Argument `fill =` definiert werden. Diese färben den Graphen in zwei oder mehr Farben ein, um Gruppen unterscheidbar zu machen. Für das hier verwendete `geom_point()` wird das Argument `color =` benötigt. Da die Zuschauerzahlen der ersten und zweiten Staffel unterschieden werden sollen, wird die Variable `Staffel` im Argument `color =` eingetragen.

```
ggplot(Quoten,
       aes(x= Folge,
           y = alleMio,
           color = Staffel)) +
  geom_point()
```

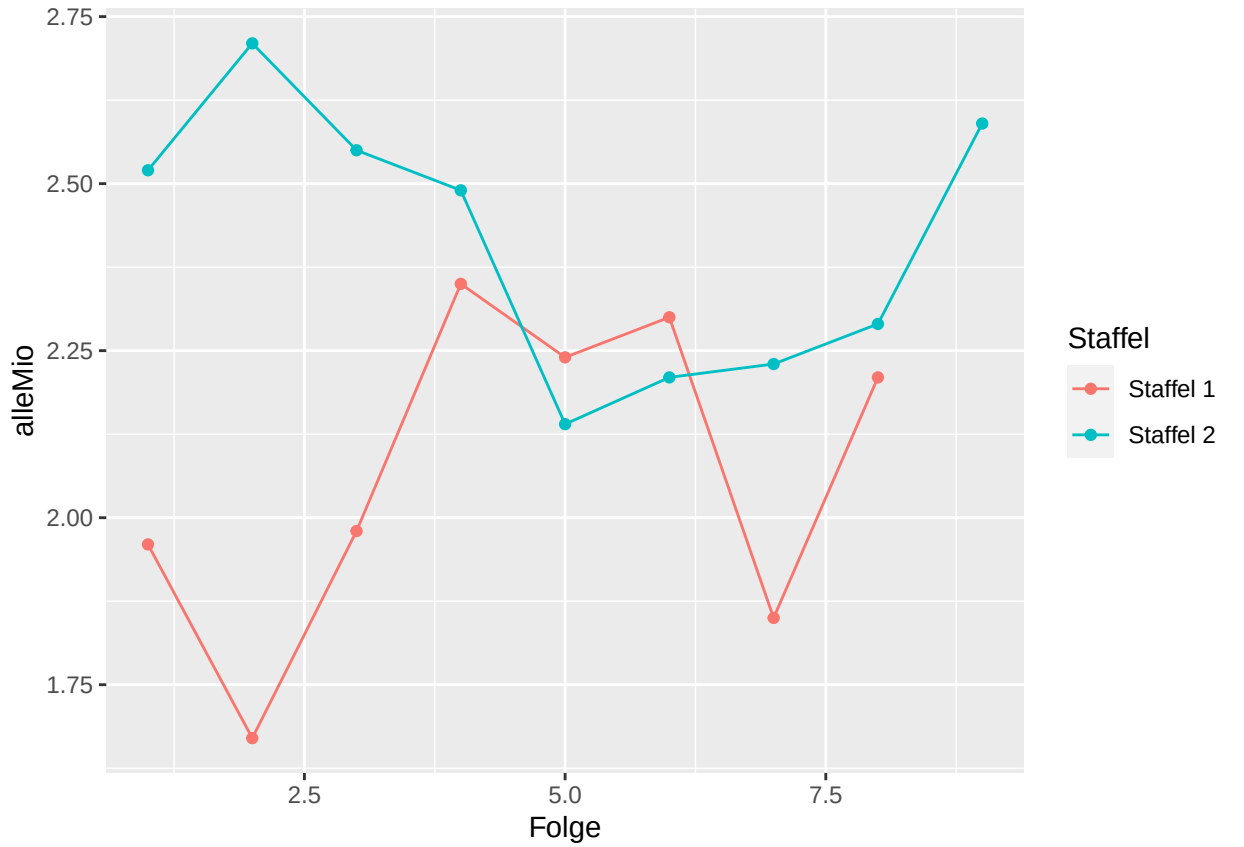


Das Ergebnis ist ein Plot, der schon interpretiert werden könnte. Dennoch sollte der Plot noch etwas einfacher gestalten werden, so dass er für die Lesenden schneller zu erfassen ist. Im ersten Schritt werden die Punkte mit Linien verbunden, um den Trend verständlicher zu machen.

7.1.4 Linien

Um die einzelnen Datenpunkte mit einer Linie zu verbinden, wird eine weitere Schicht mit geometrischen Elementen benötigt. Ähnlich wie beim Befehl `geom_point()`, kann zusätzlich noch der Befehl `geom_line()` angehängt werden. Der Befehl `geom_line()` verbindet dann die einzelnen Datenpunkte mit einer Linie. Die beiden Farb-Gruppen werden in der Regel automatisch bei den Linien unterschieden.

```
ggplot(Quoten,
       aes(x= Folge,
           y = alleMio,
           color = Staffel)) +
  geom_point() +
  geom_line()
```



7.1.5 Achsen anpassen

Die bisherigen Diagramme machen den Inhalt schwer erfassbar, da die Achsen noch nicht die optimalen Eingeschaften haben. Die y-Achse zum Beispiel startet nicht bei `0`, was die Trends verzerrt und Unterschiede zwischen den einzelnen Folgen größer erscheinen lässt, als sie eigentlich sind. Dies kann mit dem `scale_`-Befehl angepasst werden. Der Befehlsname besteht aus mehreren Teilen. Als Erstes wird die Achse angegeben, auf die sich der Befehl bezieht (hier die y-Achse) mit `scale_y_`. Daran wird die Eigenschaft der Achse (diskret, kontinuierlich) angehängt. Demnach wird wird hier `scale_y_continuous()` verwendet.

Dieser Befehl hat einige mögliche Argumente, mit denen die Achsen angepasst werden können. Zu den wichtigsten zählen `breaks =`(Sprünge auf der Achse kennzeichnen) und `limits =` (Anfang und Ende der Achse definieren). Weitere Unterbefehle können in der Dokumentation des `tidyverse` nachgelesen werden.

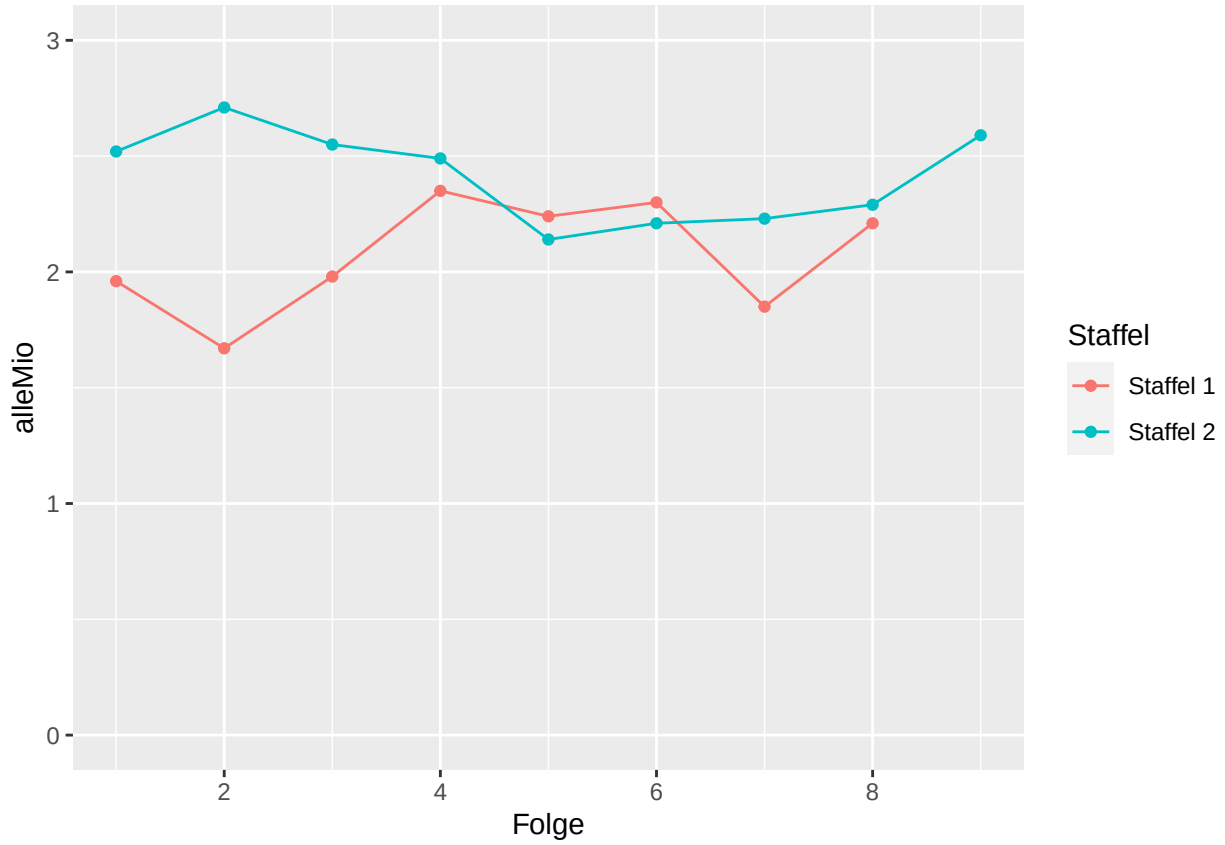
Um die y-Achse nun bei `0` beginnen zu lassen, wird der Befehl `scale_y_continuous(limits = c(0,3))` verwendet. Der Vektor `c(0,3)` gibt dabei den Beginn und das Ende der Skala an.

Die Sprünge und Hilfslinen auf der x-Achse sind ebenfalls missverständlich, da zum Beispiel die Werte `2.5` und `7.5` vermerkt sind. Für diese Werte liegen keine Folgen vor, sondern nur für ganze Zahlen. Daher sollte die Skala ganzzahlig, hier in Zweier-Sprüngen angeordnet sein. Der Befehl `scale_x_continuous(breaks = c(0,2,4,6,8, 10))` ermöglicht dies. `scale_x_continuous` macht deutlich, dass es sich um die x-Achse handelt, die angepasst werden soll. Das Argument

`breaks =` verändert die Sprünge auf der Achse und der Vektor `c(0,2,4,6,8, 10))` gibt an, welche Sprünge vorgenommen werden sollen.

Die beiden Befehle werden als weitere Schichten im `ggplot()`-Befehl ergänzt.

```
ggplot(Quoten,
       aes(x= Folge,
           y = alleMio,
           color = Staffel)) +
  geom_point() +
  geom_line() +
  scale_y_continuous(limits = c(0,3)) +
  scale_x_continuous(breaks = c(0,2,4,6,8))
```



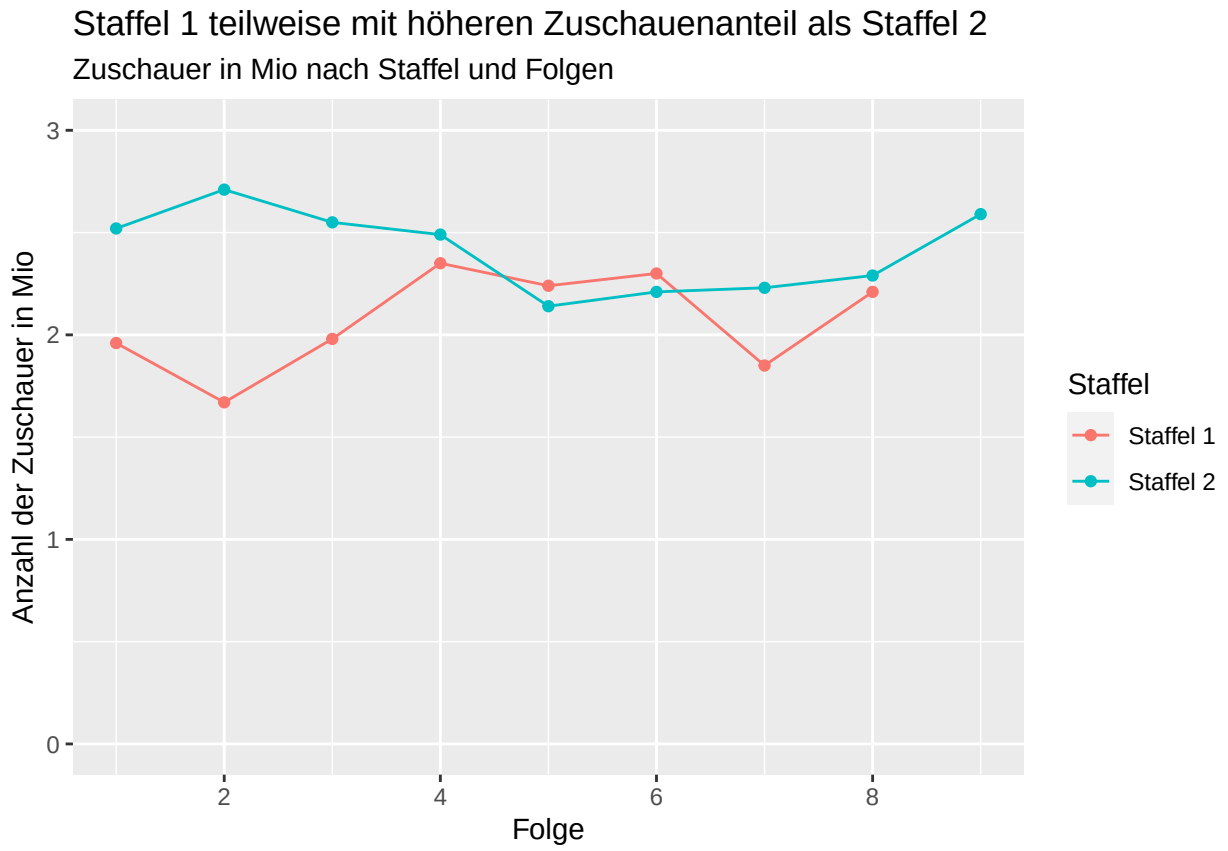
Nun können die Trends der beiden Staffeln besser erfasst werden. Es wird deutlich, dass die Anzahl der Zuschauer stabiler ist, als es im vorherigen Diagramm suggeriert wurde.

7.1.6 Beschriftungen im Diagramm

Um die Informationen der Grafik möglichst schnell erfassbar zu machen, bietet es sich an, einen Titel und Achsenbeschriftungen zu verwenden. Dazu gibt es den Befehl `labs()`. Hierfür werden die einzelnen Beschriftungen als Text in Anführungsstrichen den jeweiligen Argumenten zugeordnet. Wenn die x-Achse beschriftet werden soll, wird das Argument `x =` "Folge" verwendet. Ein Titel

kann mit `title =` “Staffel 1 teilweise mit höheren Zuschauenanteil als Staffel 2” hinzugefügt werden. Die Überschrift der Legende kann in unserem Bespiel mit `color =` “Staffel” angepasst werden, da `color =` diese Legende erstellt hatte.

```
ggplot(Quoten,
       aes(x= Folge,
           y = alleMio,
           color = Staffel)) +
  geom_point() +
  geom_line() +
  scale_y_continuous(limits = c(0,3)) +
  scale_x_continuous(breaks = c(0,2,4,6,8)) +
  labs(x = "Folge",
       y = "Anzahl der Zuschauer in Mio",
       color = "Staffel",
       title = "Staffel 1 teilweise mit höheren Zuschauenanteil als Staffel 2",
       subtitle = "Zuschauer in Mio nach Staffel und Folgen")
```

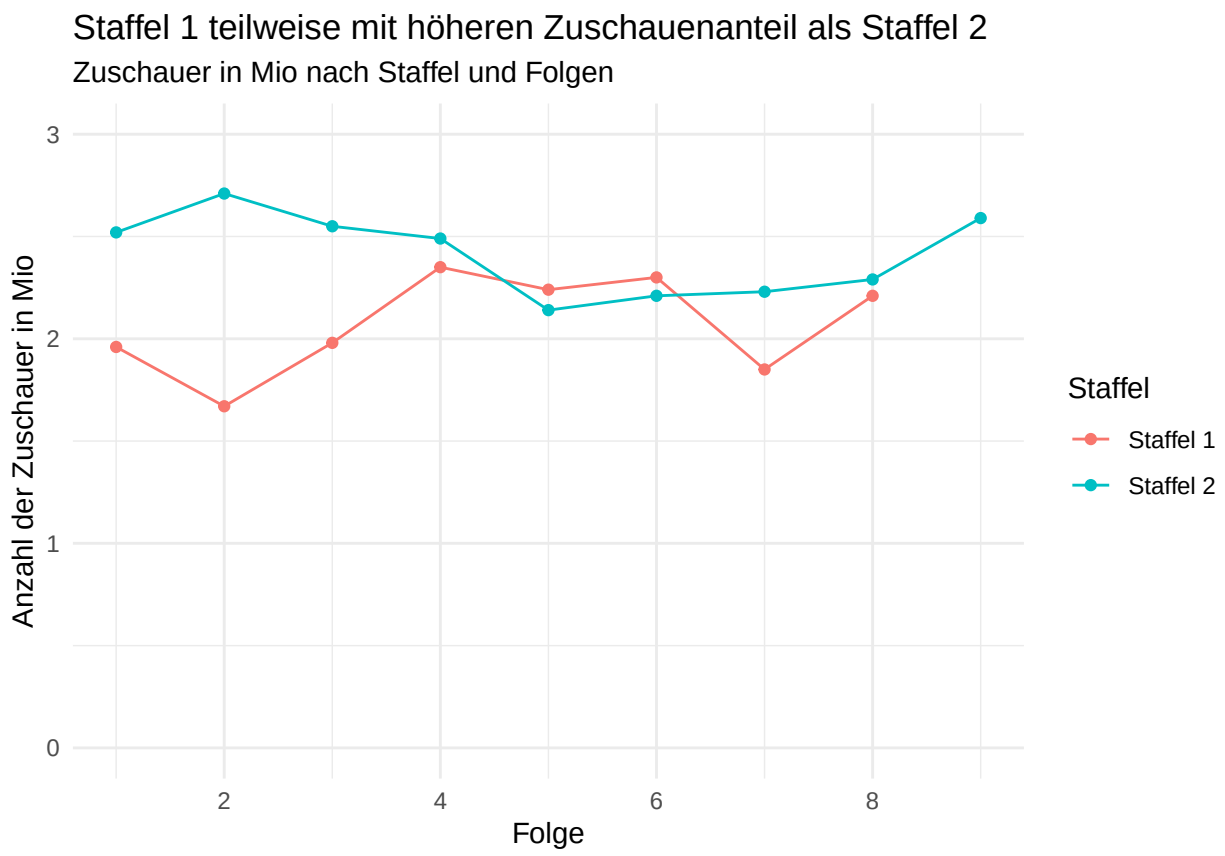


7.1.7 Designeinstellungen

Zum Schluss kann ein klares Design das Verständnis der Grafik vereinfachen. `ggplot` bietet viele Designs unter dem Befehl `theme_` an, z.B. die beiden Designs `theme_minimal()` und `theme_light()`. Diese werden als weitere Schicht auf unser Diagramm mit dem jeweiligen Befehl aufgetragen.

In diesem Beispiel wird das minimalistische Theme verwendet, welches mit dem Befehl `theme_minimal()` aktiviert wird.

```
ggplot(Quoten,
       aes(x= Folge,
           y = alleMio,
           color = Staffel)) +
  geom_point() +
  geom_line() +
  scale_y_continuous(limits = c(0,3)) +
  scale_x_continuous(breaks = c(0,2,4,6,8)) +
  labs(x = "Folge",
       y = "Anzahl der Zuschauer in Mio",
       color = "Staffel",
       title = "Staffel 1 teilweise mit höheren Zuschauenanteil als Staffel 2",
       subtitle = "Zuschauer in Mio nach Staffel und Folgen") +
  theme_minimal()
```



7.2 Säulendiagramme

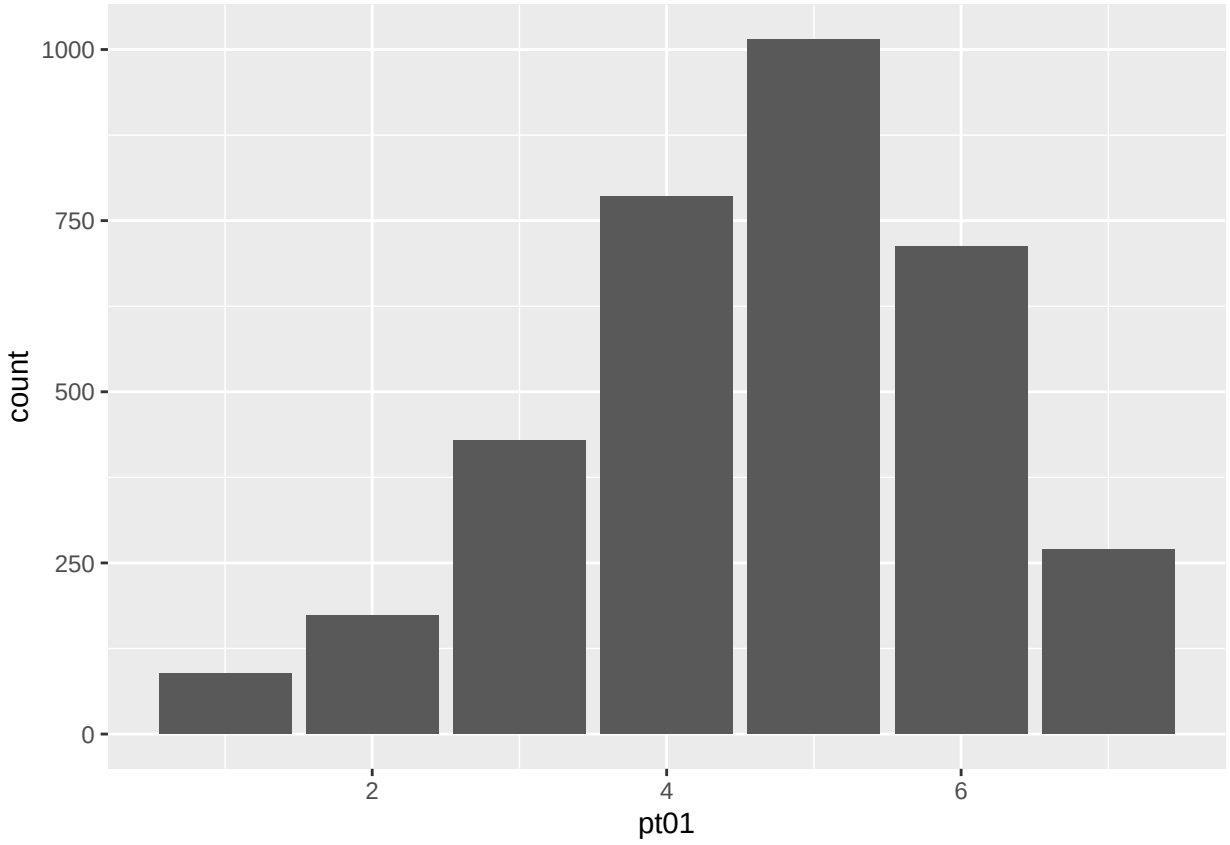
7.2.1 Grundlagen

Um absolute Häufigkeiten einer diskreten Variable darzustellen, empfiehlt sich ein Säulendiagramm. Im Folgenden wird das Vertrauen in das Gesundheitssystem aus den ALLBUS-Daten dargestellt

```
library(haven)
allbus <- as.data.frame(read_spss("Daten/ALLBUS2018.sav"))
```

Mit dem Befehl `ggplot(allbus, aes(x = pt01))` wird festgelegt, welche Daten in dem Diagramm verwendet werden sollen (`allbus`) und welche Variable dargestellt werden soll (`aes(x = pt01)`). Hier ist nur eine Spezifizierung der x-Achse nötig, da das Säulendiagramm nur aus einer Variablen besteht. Die y-Achse wird automatsich erstellt. Wenn der Befehl `geom_bar()` angehängt wird, wird ein Säulendiagramm mit den absoluten Häufigkeiten ausgegeben.

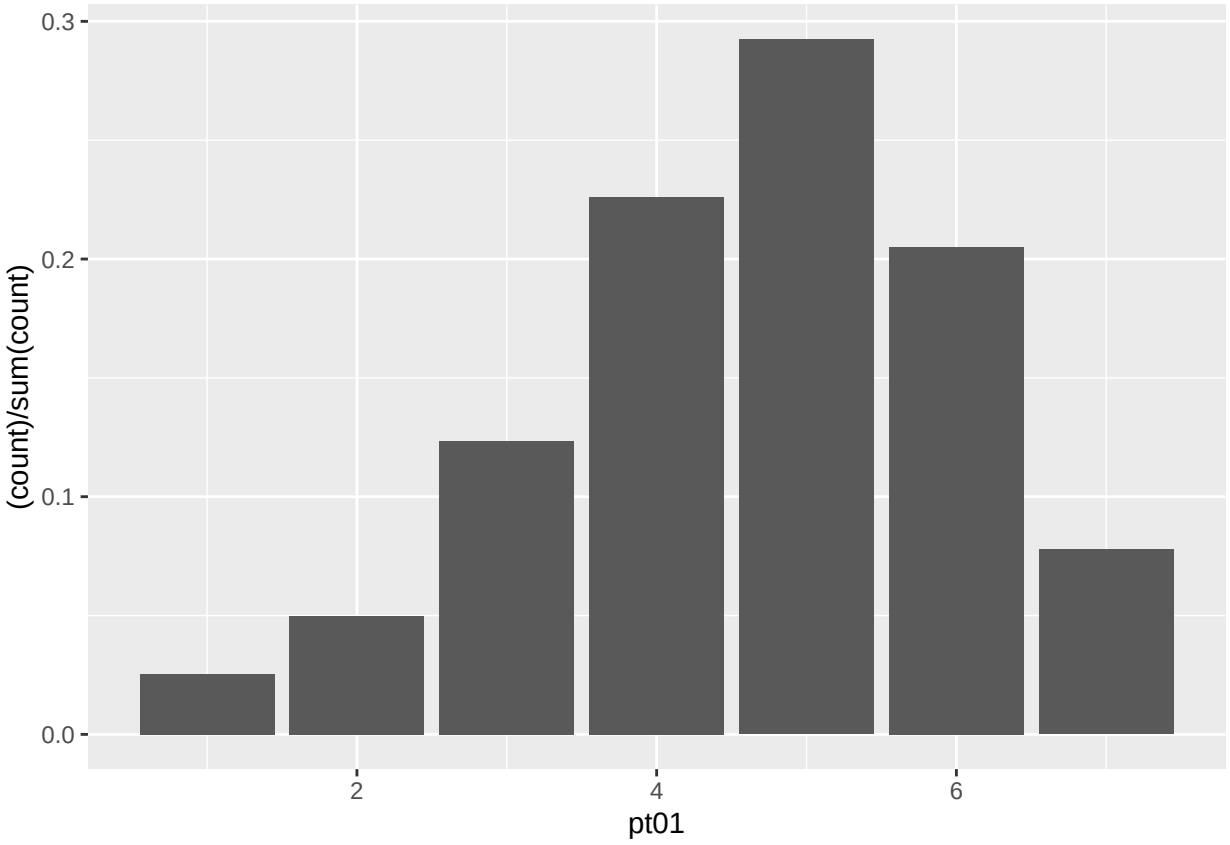
```
ggplot(allbus,
       aes(x = pt01)) +
  geom_bar()
```



7.2.2 Relative Häufigkeiten

Als Grundeinstellung wird durch `geom_bar()` eine y-Achse mit der Anzahl der Befragten, also den absoluten Häufigkeiten erstellt. Diese Einstellung kann geändert werden, in dem eine andere Berechnung der y-Achse manuell definiert wird. Mit `aes(y = (..count..)/sum(..count..))` wird angegeben, wie die y-Achse berechnet werden soll, nämlich indem für jede Kategorie die Anzahl der Befragten (`..count..`) durch die Summe aller Befragten `sum(..count..)` geteilt wird; dies entspricht der relativen Häufigkeit.

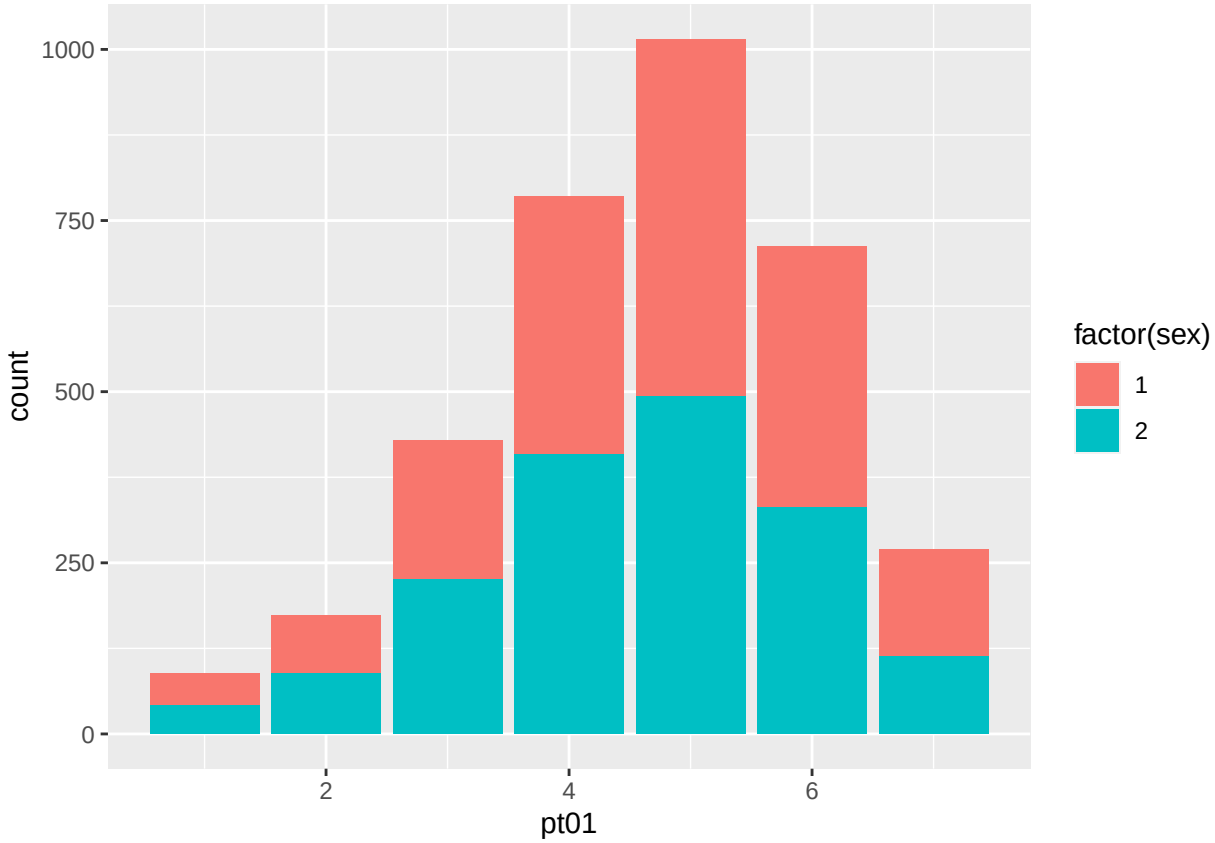
```
ggplot(allbus,
       aes(x = pt01)) +
  geom_bar(aes(y = (..count..)/sum(..count..)))
```



7.2.3 Absolute Häufigkeit nach Gruppen

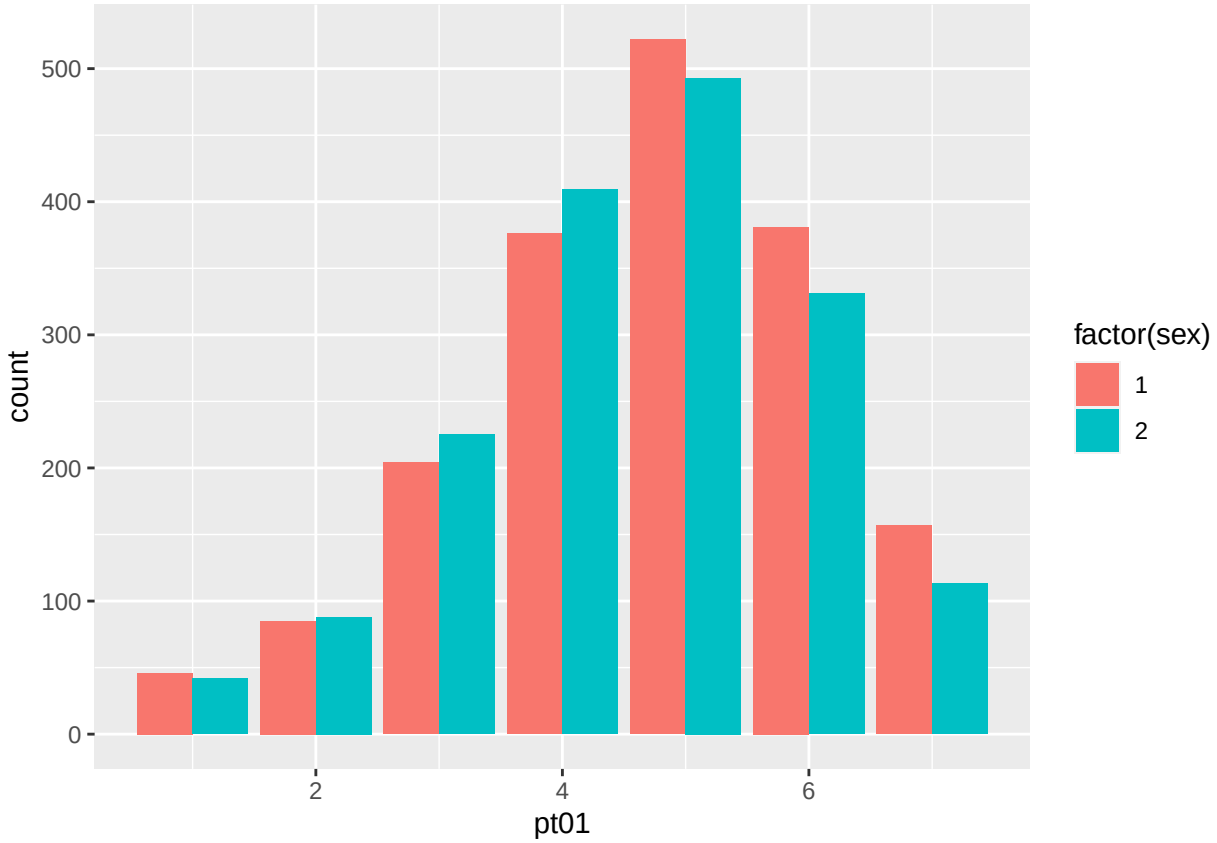
Außerdem können die absoluten Häufigkeiten nach Gruppen ausgegeben werden. Zum Bespiel das Vertrauen in das Gesundheitssytem nach Geschlecht. Dazu wird das Argument `fill =` in dem Befehl `aes()` verwendet. In diesem Argument wird angegeben, nach welcher Gruppe die Häufigkeiten aufgeteilt werden sollen. Mit dem Befehl `factor()` wird sichergestellt, dass die Gruppierungsvariable ein Faktor ist. Das ist nötig, weil die Aufteilung nach Gruppen nur mit Variablen, die als Faktoren definiert sind, möglich ist.

```
ggplot(allbus,
       aes(x = pt01,
           fill = factor(sex))) +
  geom_bar()
```



Um die Verteilungen von Männern und Frauen besser vergleichen zu können, kann die Position der Säulen angepasst werden. Das Argument `position = "dodge"`wird verwendet, um die Säulen nebeneinander darzustellen.

```
ggplot(allbus,
       aes(x = pt01,
           fill = factor(sex))) +
  geom_bar(position = "dodge")
```

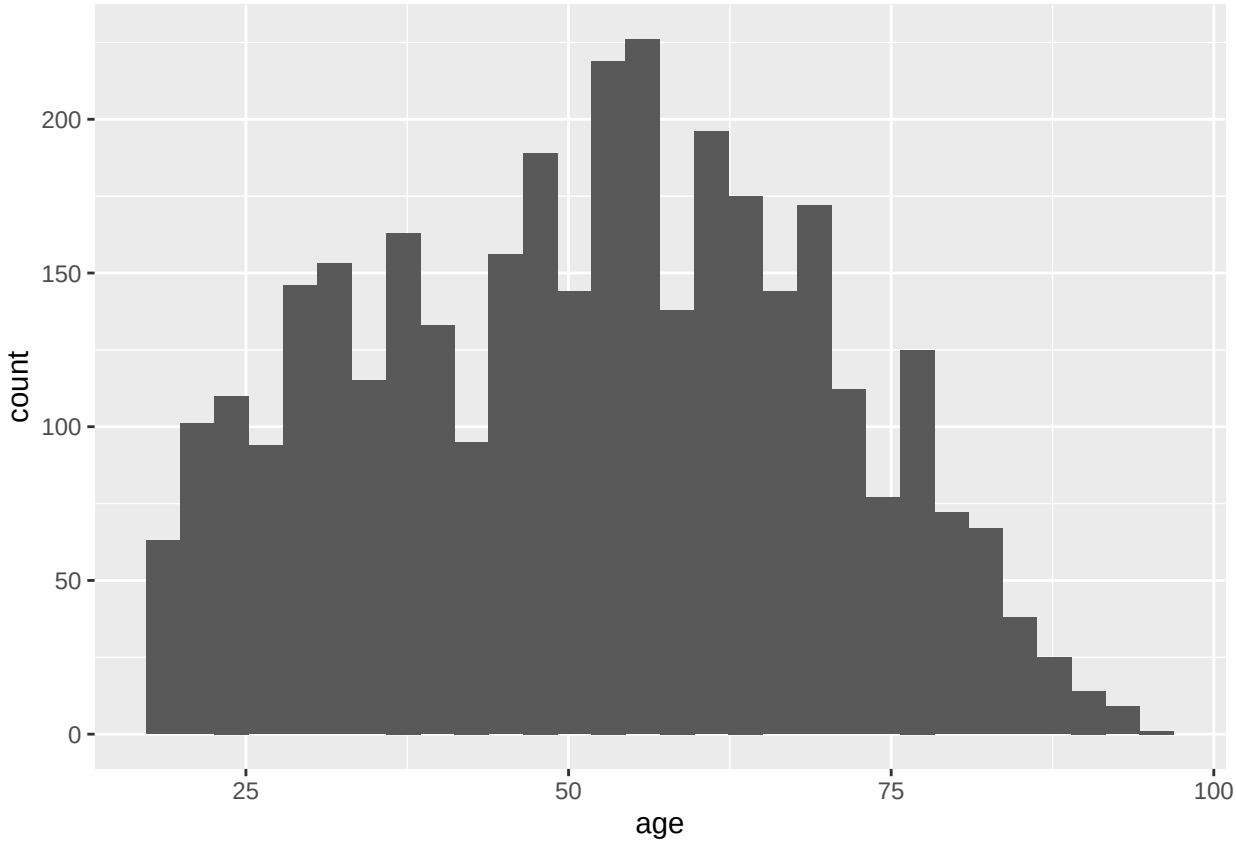


7.3 Histogramm

Ein Histogramm ist eine weitere Möglichkeit, um Häufigkeiten darzustellen. Histogramme lohnen sich vor allem für metrische Variablen mit vielen einzelnen Ausprägungen wie dem Alter. Die einzelnen Ausprägungen werden im Histogramm nach Häufigkeit zu Balken zusammengefasst.

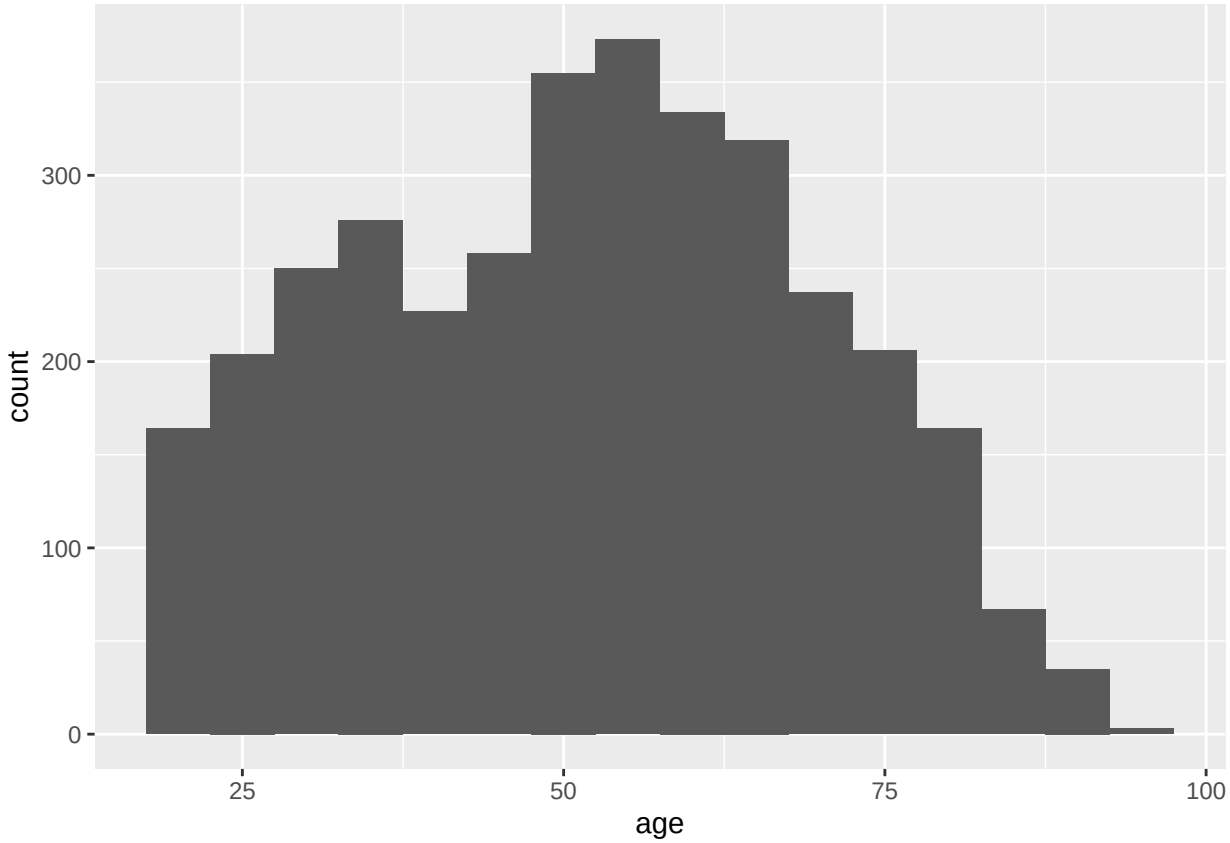
Der Befehl funktioniert analog zu den Säulendiagrammen. Es wird in `ggplot()` definiet, welche Daten verwendt werden sollen und welche Variable (hier `age`) im Histogramm dargestellt werden soll. Danach wird mit dem Befehl `geom_histogram()` das entsprechende Histogramm ausgegeben.

```
ggplot(allbus,
       aes(x = age)) +
  geom_histogram()
```



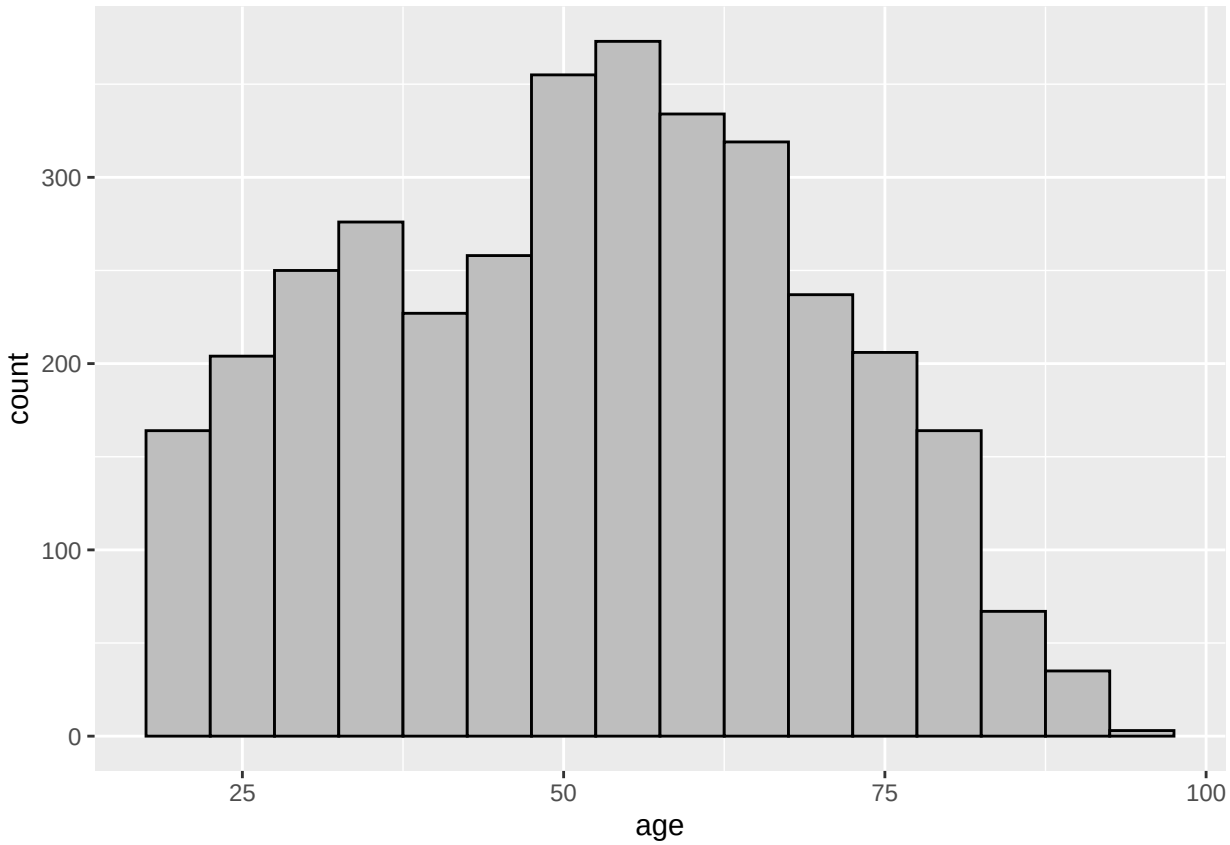
Das Argument `binwidth =` ändert die Art der Zusammenfassung. Hier kann definiert werden, wie breit jede Kategorie sein soll. Wenn dieser Wert zum Beispiel auf `5` gesetzt wird, werden immer fünf Jahre zu einem Balken zusammengefasst, also zum Beispiel die Stufen 18, 19, 20, 21, 22 im ersten Balken.

```
ggplot(allbus,
       aes(x = age)) +
  geom_histogram(binwidth = 5)
```



In diesem Diagramm verschwimmen die einzelnen Kategorien. Das Histogramm ist einfacher zu verstehen, wenn sich die einzelnen Kategorien / Altersklassen voneinander unterscheiden. Das können die Argumente `fill = "grey"` und `color = "black"` im Befehl `geom_histogram()` erreichen. `fill =` ändert die Füllung der Histogrammflächen, während `color =` den Rand der Flächen ändert.

```
ggplot(allbus,
       aes(x = age)) +
  geom_histogram(binwidth = 5,
                 fill = "grey", color = "black")
```


count
300
200
100
0
25
50
75
100
age

8 Mittelwertvergleiche

8.1 Einführung in die Inferenzstatistik

Nachdem sich die letzten Kapitel auf deskriptive Auswertungen von Daten bezogen haben, beziehen sich dieses und folgende Kapitel auf inferenzstatistische Methoden der Datenauswertung. Grundgedanke der Inferenzstatistik ist es, von den erhobenen Daten auf eine **Grundgesamtheit** zu schließen. Das bedeutet, dass mit den erhobenen Daten Aussagen getreffen werden sollen, die nicht nur für die befragten Personen gültig sind, sondern für eine vordefinierte Gruppe an Menschen. Im Falle des ALLBUS ist dies die Bevölkerung in Deutschland ab 18 Jahren. Die Grundgesamtheit kann je nach Studienanlage variieren. Sollen beispielsweise Aussagen über die Studierendenschaft einer bestimmten Universität getroffen werden, so würde diese die Grundgesamtheit bilden. In der empirischen Praxis müssten Forschende dann eine Stichprobe aus der Studierendenschaft befragen.

Und auch bei Inhaltsanalysen ist es wichtig zu wissen, für welche Grundgesamtheit eine Aussage getroffen werden soll. Wenn sich Forschende beispielsweise damit beschäftigten, wie über ein bestimmtes Thema (z.B. Migration nach Deutschland) in den (Massen-)Medien berichtet wird, so ist die Grundgesamtheit prinzipiell jeder Nachrichtenbeitrag über das entsprechende Thema.

Während bei klassischen Befragungsstudien und Inhaltsanalysen die Grundgesamtheit eine große Rolle spielt, gibt es auch Erhebungsmethoden bei deinen eine repräsentative Stichprobe eine eher untergeordnete Rolle spielt. Dies betrifft vor allem Experimentalstudien sowie Messinstrumententwicklungen. In Experimentalstudien wird der Effekte einer oder mehrerer Faktoren gezielt manipuliert. Durch eine zufällige Zuweisung (Randomisierung) der Probanden auf Experimental- und Kontrollgruppe sind alle anderen (Stör-)Variablen in der Regel zu vernachlässigen, da sie gleichermaßen in den beiden Gruppen auftreten sollten. Gemessene Effekte sind dann allein auf den Stimulus zurückzuführen. Wichtig hierbei ist jedoch, dass die Gruppen sich in ihrer Zusammensetzung nicht wesentlich voneinander unterscheiden. Eine typische Aussage wäre dann, dass Kondition A (Stimulus oder Treatment) im Vergleich zu Kondition B (Kontrollgruppe oder Placebo-Treatment) einen anderen Effekt auslöst.

Die Grundlagen, wie man eine vernünftige Stichprobe zieht und welche Schritte bei der Datenerhebungen beachtet werden sollten, werden in diesem Skript nicht weiter erläutert. Hintergrundinformation können in Lehr- und Handbüchern zu sozialwissenschaftlichen Methoden gefunden werden (z.B. Diekmann, 2007; Döring & Bortz, 2016; Schnell, Hill & Esser, 2018). In der empirischen Praxis muss der Auswahl des Instruments und der Stichprobe viel Aufmerksamkeit geschenkt werden. Die Aussagekraft einer Studie kann massiv darunter leiden, dass Aussagen nicht auf eine Grundgesamtheit übertragen werden können. Vor jeder empirischen Studie müssen sich Forschende die Frage stellen: Sollen Aussage über eine spezifische Grundgesamtheit getroffen werden? Wenn ja, wie ist die Grundgesamtheit definiert?

Die ersten inferenzstatistischen Verfahren, denen sich dieses Skript widmet, sind **Mittelwertvergleiche**. Mittelwertvergleiche überprüfen, ob sich (mindestens intervallskalierte) Variablen (signifikant) voneinander unterscheiden. In diesem Skript werden drei Verfahren vorgestellt: **T-Test bei unabhängigen Stichproben**, **T-Test bei verbundenen Stichproben** und die **Einfaktorielle Varianzanalyse**.

Auch in diesem Kapitel wird mit dem ALLBUS-Datensatz gearbeitet. Allerdings ist dabei anzumerken, dass bei jeglichen Aussagen über die Grundgesamtheit das komplexe Stichprobendesign des Datensatzes (Gewichtungen) berücksichtigt werden müssen. Da hier zunächst aber das

Grundprinzip der Tests vorgestellt werden soll, wird an dieser Stelle auf eine Gewichtung der Analysen verzichtet. Eine Erläuterung zu Gewichten und deren praktische Umsetzung in R findet sich in Kapitel 13 dieses Skripts.

Im ALLBUS-Datensatz wurde die ostdeutsche Bevölkerung höher gewichtet als die westdeutsche Bevölkerung. In den jeweiligen Teilgruppen ist jedoch eine Bevölkerungsrepräsentatitivität gewährleistet. Das heißt, dass zum einen Vergleiche zwischen ostdeutschen und westdeutschen Personen ohne Gewichtung berechnet werden dürfen sowie jeweils mit den Teilstichproben ohne Gewichtung gerechnet werden darf (z.B. Analysen nur für die westdeutsche Bevölkerung).

```
library(haven)
allbus <- as.data.frame(read_spss("ALLBUS2018.sav"))
```

8.2 t-Test für unabhängige Stichproben

8.2.1 Einführung

Ein T-Test für unabhängige Stichproben überprüft, ob es Unterschiede in einer Variable **zwischen (genau) zwei Gruppen** gibt. Die Gruppenvariable muss dabei ein **dichotomer Faktor** sein und die Ausprägungen müssen überschneidungsfrei sein; d.h. kein Befragter darf beide Ausprägungen aufweisen. Die Testvariable muss intervallskaliert sein, z.B. eine metrische Größe (intervall- oder ratioskaliert) oder eine Zustimmungsabfrage auf einer quasi-metrischen Skala.

Bezogen auf den ALLBUS-Datensatz kann z.B. folgende Frage gestellt werden: Unterscheiden sich Westdeutsche und Ostdeutsche in ihrem Vertrauen in das europäische Parlament?

Konkret werden dabei nun die mittleren Zustimmungswerte verglichen und statistisch überprüft, ob ein Unterschied vorliegt und ob dieser groß genug ist um **überzufällig** zu sein. Der t-Wert wird dabei mit folgender Formel berechnet:

$$t = \frac{\overline{X_1} - \overline{X_2}}{\sqrt{(\frac{s_p^2}{n_1} + \frac{s_p^2}{n_2})}}$$

Der errechnete t-Wert ist daher abhängig von der Mittelwertdifferenz der beiden Gruppen, der gewichteten Stichprobenvarianz in der Testvariable sowie der Stichprobengröße. Diesem Test liegt die Annahme zugrunde, dass die Differenz der Mittelwerte annähernd normalverteilt ist. Die empirischen t-Werte folgen dann den Werten der t-Verteilung. Die Studentische t-Verteilung nähert sich mit zunehmender Fallzahl einer Normalverteilung an. Daher besitzt die t-Verteilung ähnliche Eigenschaften wie die Normalverteilung. Dies lässt sich zur Interpretation der t-Werte nutzen.

Der beobachtete t-Wert ist ein standardisierter Wert, der eine kritische Größe überschreiten kann und dadurch Signifikanz anzeigt. In der Regel deuten Werte größer als +2 und kleiner als -2 darauf hin, dass ein Unterschied nicht zufällig entdeckt wird und zwar mit einer fünfprozentigen Irrtumswahrscheinlichkeit (die genauen Werte liegen bei großen Stichproben bei -1.96 und +1.96). In der Regel kann die Daumenregel "+/-2" bei Tests basierend auf einer Stichprobe von mindestens 30 Fällen und einem angenommenen Signifikanzniveau von 5% angewendet werden. Heutzutage müssen diese kritischen Werte zur Bestimmung der Signifikanz nicht mehr in Tabellen statistischer Nachschlagewerke abgelesen werden, sondern werden von R automatisch berechnet und als sogenannter Signifikanzwert, auch **p-Wert** genannt, angezeigt. Mit diesem Wert lässt sich die Signifikanz des beobachteten t-Werts interpretieren.

8.2.2 Das Vorgehen des statistischen Testens

Beim statistischen Testen gibt es verschiedene Stufen, die nacheinander abgearbeitet werden sollten. In der Literatur wird dies mitunter in vier bis sieben Stufen unterteilt. Hier liegt der Fokus auf dem grundsätzlichen Vorgehen.

1. Formulieren der Nullhypothese (HO).
2. Formulieren der Alternativhypothese (H1, auch Arbeitshypothese). Hierbei ist zu berücksichtigen, ob lediglich ein Unterschied angenommen wird oder ob es sich um einen gerichteten Unterschied oder Zusammenhang handelt. Daher entscheidet dieser Schritt, ob ein ein- oder beidseitiger Test durchgeführt wird. Diese Entscheidung ist abhängig von den aufgestellten Hypothesen. Wird ein Unterschied ohne Richtungsvermutung angenommen, dann wird ein beidseitiger Test durchgeführt. Bestehen Annahmen über einen gerichteten Zusammenhang (z.B. "Kinder sind kleiner als Erwachsene"), so kann ein einseitiger Test durchgeführt werden. Allerdings sind beidseitige Tests oftmals die Regel. Ein- und beidseitige Tests legen fest, ob die Irrtumswahrscheinlichkeiten in beide Richtungen und nur in eine Richtung eingeräumt werden.
3. Auswahl der Daten für den Test (Fallzahl berücksichtigen).
4. Festlegen der geeigneten Irrtumswahrscheinlichkeit.
5. Testdurchführung.
6. Interpretation des Ergebnisses. Vergleich des Ergebnisses des beobachteten t-Tests mit dem kritischen Wert entsprechend der zuvor aufgestellten Kriterien (abhängig vom festgelegten alpha, ein- oder beidseitigem Test und der zugrundeliegenden Fallzahl).

8.2.3 Durchführung

Als Beispielthese wird angenommen, dass sich das Vertrauen in Institutionen für Personen in Ost- und Westdeutschland unterscheidet. Hierbei wird lediglich von einem Unterschied ausgegangen. Es werden keine Annahmen über die Richtung des Zusammenhangs getroffen. Um die These zu prüfen, werden zunächst rein deskriptiv die Mittelwerte nach ostdeutscher und westdeutscher Bevölkerung ausgegeben. Dafür wird die Variable `eastwest` in einen Faktor `westost` umcodiert. Mit `tapply()` können dann die Mittelwerte für beide Gruppen getrennt ausgegeben werden.

```
attributes(allbus$eastwest)
## $label
## [1] "ERHEBUNGSGEBIET (WOHNGEBIET): WEST - OST"
##
## $format.spss
## [1] "F1.0"
##
## $display_width
## [1] 10
##
## $class
## [1] "haven_labelled" "vctrs_vctr"     "double"
##
## $labels
```

```
## ALTE BUNDESLAENDER NEUE BUNDESLAENDER
##                  1                  2
table(allbus$eastwest)
##
##    1    2
## 2387 1090
allbus$westost <- factor(allbus$eastwest,
                         labels = c("Westdeutschland", "Ostdeutschland"))
table(allbus$westost)
##
## Westdeutschland  Ostdeutschland
##            2387            1090

tapply(allbus$pt20, allbus$westost, mean, na.rm=TRUE)
## Westdeutschland  Ostdeutschland
##        3.712366        3.329960
```

Alternativ können Mittelwerte auch mit `dplyr` nach Gruppen bestimmt werden. Dazu muss zuerst angegeben werden, welche Gruppen unterschieden werden sollen, dies geschieht mit dem Befehl `group_by()`. Dann werden die Daten mit dem Befehl `summarise()` aggregiert und hier die Kennzahl für das arithmetische Mittel ausgegeben.

```
# dplyr Variante
library(tidyverse)
allbus %>%
group_by(westost) %>%
summarise(mean_pt20 = mean(pt20, na.rm = TRUE))
## # A tibble: 2 x 2
##   westost         mean_pt20
##   <fct>               <dbl>
## 1 Westdeutschland      3.71
## 2 Ostdeutschland       3.33
```

Rein deskriptiv wird erkenntlich, dass Personen in Westdeutschland anscheinend ein wenig mehr Vertrauen in das europäische Parlament haben als Personen in Ostdeutschland, allerdings könnte es sich auch um einen zufälligen Unterschied handeln. Um dies zu überprüfen folgt nun der Inferenztest - der t-Test. Der t-Test ist standardmäßig in base R implementiert.

Ein t-Test kann über die Funktion `t-test()` durchgeführt werden. Die Logik eines Inferenztests ist dabei folgende: Es wird eine sog. **Nullhypothese** überprüft. Eine Nullhypothese besagt, dass es *keinen* Unterschied (oder keinen Zusammenhang) gibt. In diesem Fall lautet die Nullhypothese: Das Vertrauen in das europäische Parlament unterscheidet sich **nicht** zwischen West- und Ostdeutschen. Es wird ein beidseitiger Test durchgeführt. Dieser ist die Standardoption im Argument `alternative = two.sided` der Funktion `t-test()` . In den Sozialwissenschaften liegt die allgemein akzeptierte Irrtumswahrscheinlichkeit in der Regel bei 5%, sodass für ein Irrtumswahrscheinlichkeit unter 5 Prozent geprüft wird.

```
t.test(allbus$pt20 ~ allbus$westost, na.rm=TRUE)
##
##  Welch Two Sample t-test
##
## data:  allbus$pt20 by allbus$westost
## t = 6.9701, df = 1882.3, p-value = 4.365e-12
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.2748059 0.4900062
## sample estimates:
## mean in group Westdeutschland  mean in group Ostdeutschland
##                      3.712366                      3.329960
```

Im Folgenden wir die Ergebnisansicht Schritt für Schritt durchgegangen. Zunächst wird erkenntlich, dass ein *Welch Two Sample t-Test* berechnet wurde. Danach wird hinter `data` aufgeführt, dass die Variable `pt20` zwischen `westost` verglichen wurde. Als nächstes folgen die Werte, die für die eigentliche Interpretation benötigt werden. Der t-Wert beträgt ungefähr `6.97`. Der Wert `df` zeigt die sogenannten Freiheitsgrade (engl. degrees of freedom) an, welche für die Berechnung des p-Wertes benötigt werden.

Schließlich zeigt der p-Wert `4.365e-12` an . Dies liegt an der wissenschaftlichen Notation, die standardmäßig in R ausgegeben wird. Das `e-12` bedeutet dabei, dass der Wert mit 10^{-12} multipliziert wird. Die Ausgabe kann durch die Eingabe von `option(scipen = 999)` vor der Ausführung des Tests geändert werden. Mit diesem zusätzlichen Befehl erscheint der p-Wert als `0.000000000004365` in der Ausgabe. Der p-Wert stellt den **Signifikanzwert** dar. In den Sozialwissenschaften arbeitet man i.d.R. mit sogenannten Signifikanzniveaus. Das sind konventionell festgelegte Schwellenwerte, unter denen von einer statistischen Signifikanz gesprochen werden darf. In den Sozialwissenschaften liegt dies meist bei `p<.05`. Der Wert wird - bezogen auf die Nullhypothese - folgendermaßen interpretiert: Der p-Wert gibt die Wahrscheinlichkeit an, dass die Nullhypothese zutrifft (also keine Unterschiede bestehen). In obigen Beispiel ist die Wahrscheinlichkeit dafür sehr gering. Es ist also höchst unwahrscheinlich, dass sich die Mittelwerte in der Grundgesamtheit **nicht** unterscheiden.

Andersherum lässt sich also folgendes sagen: Das Vertrauen in das europäische Parlament unterscheidet sich signifikant zwischen Personen aus Ost- und Westdeutschland. Die Irrtumswahrscheinlichkeit liegt dabei unter 5% ($p<.05$).

Weiterhin wird die sog. Alternativhypothese angezeigt. Dies ist das Gegenteil der Nullhypothese, welche besagt, dass die "wahre" Mittelwertdifferenz ungleich 0 ist (= es gibt Unterschiede). Schließlich wird noch das 95% Konfidenzintervall angegeben. Dieses beruht auf einer statistischen Berechnung und besagt, dass der Mittelwertunterschied zwischen Personen in Ost- und Westdeutschland mit 95% Wahrscheinlichkeit zwischen den angegebenen Werten liegt (hier `0.2748059` und `0.4900062`).

Auch hier bedarf es einiges an Hintergrundwissen über Teststatistik. Inhärent für jedes inferenzstatistische Verfahren ist, dass Fehler in die Messung einkalkuliert werden. Es wird davon ausgegangen, dass beispielsweise ein gemessener Stichprobenmittelwert auch ungefähr so in der Grundgesamtheit auftritt. Jedoch unterliegt die Schätzung Schwankungen. Das Konfidenzintervall gibt nun an, dass nach der statistischen Berechnung aber mit 95% Wahrscheinlichkeit davon ausgegangen werden kann, dass der Mittelwert der Grundgesamtheit zumindest in dem errechneten Intervall liegt. Dabei ist die Breite des Intervalls abhängig von der Anzahl der Befragten sowie der Streuung der Messung.

Ein gutes Beispiel um Fehler in Schätzungen zu erklären sind Wahlumfragen, wie z.B. aus dem Politbarometer. Dort werden die Prozentwerte prognostiziert, die eine Partei bei einer Wahl zum jetzigen Stand erreichen würde. Diese Werte beruhen auf Umfragedaten, die ebenfalls einer Ungenauigkeit unterliegen. Wenn Aussagen über die Grundgesamtheit getroffen werden, muss diese Ungenauigkeit mit einbezogen werden. Wenn die Grünen beispielsweise auf 18% in den Umfragen kommen, heißt das eigentlich: Der Stichprobenwert liegt bei 18%. Bezöge man ein Konfidenzintervall mit ein, würde sich die Aussage ändern in: Die Grünen würden am Wahlsonntag zum jetzigen Stand mit 95% Wahrscheinlichkeit zwischen 16% und 20% der Stimmen kommen. Das Konfidenzintervall gibt also Aussagen über Wertebereiche, in denen eine Verteilung hochwahrscheinlich liegt.

Zuletzt werden in der Ausgabe die oben bereits berechneten Mittelwerte für die beiden Gruppen angezeigt.

Ein statistischer Bericht der Auswertung könnte zusammenfassend folgendermaßen lauten: Personen in Ost- und Westdeutschland unterscheiden sich signifikant voneinander in ihrem Vertrauen in das europäische Parlament, t(1882.3)=6.971, p<.05. Westdeutsche (M=3.71) weisen dabei ein höheres Vertrauen auf als Ostdeutsche (M=3.33).

8.2.4 Prüfung der Annahmen eines Tests

Parametrische statistische Tests liegen häufig eine Reihe von Annahmen zugrunde. Bevor man einen t-Test für unabhängige Stichproben eigentlich rechnet, sollte ein Levene-Test auf Varianzgleichheit der beiden Gruppen durchgeführt werden. Dies ist ein Test auf Varianzhomogenität. Je nachdem ob Varianzhomogenität vorliegt, d.h. die Varianzen der beiden Stichproben gleich / ähnlich sind, wird eine Anpassung der t-Test Berechnung vorgenommen. Standardmäßig geht R davon aus, dass keine Varianzgleichheit vorliegt und diese Annahme verletzt wird.

Um einen Levene-Test zu berechnen, muss ein Package installiert werden. Hier wird das `car` Package genutzt.

```
library(car)
test <- as.numeric(allbus$pt20)
leveneTest(test, allbus$westost)
## Levene's Test for Homogeneity of Variance (center = median)
##         Df F value  Pr(>F)
## group    1  5.3562 0.02071 *
##       3218
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Auch hier wird wieder eine Nullhypothese getestet (“die Varianzen sind gleich”). Das Ergebnis `Pr(>F)` ist analog zum `p-Wert` zu interpretieren. Das heißt auch hier gilt die Grenze von `.05`. Die Ausgabe zeigt, dass eine Signifikanz vorliegt; die Varianzen sind demnach nicht homogen, sondern heterogen. Dies ist eine wichtige Information für die Durchführung des t-Tests aus dem Schritt zuvor. In diesem Fall muss der vorige t-Test jedoch nicht angepasst werden. Die Standardoption der `t-test` Funktion ist, dass Varianz heterogen sind und dass die Voraussetzung für Varianzhomogenität verletzt ist. Das lässt sich am der default-Einstellung des Arguments `var.equal = FALSE` ablesen.

Hätte der Levene-Test auf Varianzhomogenität hingedeutet, hätte die obige Funktion mit `var.equal = TRUE` ergänzt werden müssen.

```
t.test(allbus$pt20 ~ allbus$westost, na.rm=TRUE, var.equal = TRUE)
```

Die Interpretation würde aber wie oben stattfinden - einzig die Werte für `df`, `p` sowie die Konfidenzintervalle würden sich verändern.

8.2.5 Zusammenfassung

Mit einem t-Test für unabhängige Stichproben kann man die Mittelwerte von genau zwei Gruppen statistisch miteinander vergleichen. Die Nullhypothese lautet dabei: Es gibt keinen Mittelwertunterschied zwischen den Gruppen; die Alternativhypothese: Es gibt einen Mittelwertunterschied zwischen den beiden Gruppen. Zunächst wird rechnerisch ein Levene-Test durchgeführt um auf Varianzhomogenität zu prüfen. Sind die Varianzen homogen (kein signifikanter p-Wert im Levene-Test), so wird ein `two sample t-test` durchgeführt. In der Funktion wird dann das Argument `var.equal=TRUE` ergänzt. Sind die Varianzen heterogen, wird der korrigierte `Welch-Test` durchgeführt. Zeigt der t-Test einen signifikanten p-Wert ($p<.05$) so wird die Nullhypothese abgelehnt. Die Mittelwerte unterscheiden sich mit 95% Sicherheit auch in der Grundgesamtheit voneinander.

8.3 t-Test für abhängige Stichproben

Beim t-Test für unabhängige Stichproben werden zwei überschneidungsfreie Gruppen in ihren Mittelwerten auf einer intervallskalierten Variable verglichen. In manchen Fällen möchte man aber auch die Mittlerwerte zweier Variablen über das gesamte Sample vergleichen. Klassischerweise wird ein t-Test für abhängige Stichproben bei Vorher-Nachher-Messungen angewendet. Beispiel: Es soll die emotionale Wirkung eines traurigen Films (Stimulus) auf das Publikum überprüft werden. Im Aufbau der Studie wird der emotionale Zustand der Teilnehmenden an zwei Zeitpunkten mit der gleichen Variable abgefragt - vor dem Stimulus und nach dem Stimulus. Es soll nun überprüft werden, ob sich der emotionale Zustand der Teilnehmenden signifikant verändert hat.

Als Beispiel wird hier ein frei erfundener Datensatz mit 30 Testpersonen generiert. Traurigkeit ist in diesem Beispiel auf einer 7er Skala gemessen.

```
traurigkeit_vorher <- c(2, 3, 2, 2, 1, 4, 2, 3, 5, 2, 2, 3, 2, 3, 4,
                        1, 1 ,2, 2, 1, 3, 3, 5, 2, 1, 1, 2, 1, 3, 4)
traurigkeit_nachher <- c(3, 3, 4, 4, 2, 3, 4, 5, 7, 4, 3, 3, 2, 5, 4,
                         2, 1 ,4, 5, 2, 2, 4, 5, 2, 3, 5, 3, 2, 3, 5)
```

Für die Beantwortung dieser Frage wird ebenfalls ein t-Test benutzt. Mit `paired = TRUE` wird bestimmt, dass es sich um den Test für abhängige Stichproben handelt. An dieser Stelle muss jedoch kein Levene-Test durchgeführt werden, da die gleichen Befragten in beiden Variablen geantwortet haben.

```
t.test(traurigkeit_nachher, traurigkeit_vorher, paired=TRUE)
##
##  Paired t-test
##
## data:  traurigkeit_nachher and traurigkeit_vorher
```

```
## t = 5.1128, df = 29, p-value = 1.854e-05
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.6399781 1.4933552
## sample estimates:
## mean of the differences
##                1.066667
```

Der Output kann analog zu dem des t-Tests für unabhängige Stichproben interpretiert werden. Es wird angezeigt, dass der Traurigkeitsmittelwert in der Nachher-Messung mit dem der Vorher-Messung verglichen wird. Der t-Wert beträgt 5.11 und ist mit $p<.05$ signifikant. Das heißt, dass die Nullhypothese (= es gibt keine Mittelwertunterschiede zwischen Vorher- und Nachhermessung) abgelehnt wird und die Alternativhypothese (= der Mittelwertunterschied ist ungleich 0) angenommen wird. Der Mittelwertunterschied beträgt in unserem Beispiel 1.066. Das 95%-Konfidenzintervall ist hier um einiges größer und umfasst die Werte zwischen 0.64 und 1.49; dies liegt v.a. daran, dass wir deutlich weniger Befragte als im vorigem Beispiel haben. Der "wahre" Mittelwertunterschied für die Grundgesamtheit liegt also mit 95% Wahrscheinlichkeit in dem Bereich. Die Grundgesamtheit müsste natürlich in einer richtigen Studie zuvor sorgfältig beschrieben werden und die Stichprobe auf dessen Grundlage gezogen werden.

8.4 Einfaktorielle Varianzanalyse (ANOVA)

Bei t-Tests werden immer genau zwei Mittelwerte miteinander verglichen. Allerdings können selbstverständlich (Forschungs-)Fragen gestellt werden, die sich auf einen Unterschied von mehr als zwei Gruppen beziehen. Bezogen auf den ALLBUS-Datensatz könnte demnach eine Frage lauten: Unterscheidet sich das Vertrauen in den Bundestag nach dem schulischen Bildungsgrad der Befragten? Gefragt wird also nach dem Vergleich von - in einem verkürzten Beispiel - drei Bildungsgraden (Hauptschulabschluss, Mittlere Reife, (Fach-)Abitur).

8.4.1 Vorbereitung

Zunächst muss wieder überprüft werden, welche Eigenschaften die Variable Schulabschluss im Datensatz aufweist.

```
attributes(allbus$educ)
## $label
## [1] "ALLGEMEINER SCHULABSCHLUSS"
##
## $format.spss
## [1] "F2.0"
##
## $display_width
## [1] 6
##
## $class
## [1] "haven_labelled" "vctrs_vctr"     "double"
```

```
##
## $labels
##       KEINE ANGABE     OHNE ABSCHLUSS VOLKS-,HAUPTSCHULE     MITTLERE REIFE
##                 -9                  1                  2                  3
## FACHHOCHSCHULREIFE     HOCHSCHULREIFE  ANDERER ABSCHLUSS      NOCH SCHUELER
##                  4                  5                  6                  7
```

Die Ergebnisse zeigen, dass der Bildungsgrad in sieben Stufen abgefragt wird. Für die Überprüfung der Forschungsfrage wird demnach eine Reduktion der Gruppen auf drei Ausprägungen benötigt. Personen ohne Schulabschluss, mit einem anderen Abschluss und Personen, die noch zur Schule gehen, werden von der Analyse ausgeschlossen. Personen mit Fachhochschulreife und Hochschulreife werden zu einer Kategorie zusammengefasst.

```
library(tidyverse)
allbus <- allbus %>%
  mutate(schulabschluss = case_when(educ == 2 ~ "Hauptschulabschluss",
                                    educ == 3 ~ "Mittlere Reife",
                                    educ %in% c(4,5) ~ "(Fach-)Abitur"))
table(allbus$schulabschluss)
##
##       (Fach-)Abitur Hauptschulabschluss      Mittlere Reife
##                1377                 810                1195
```

Schließlich wird für dieses Beispiel noch der Datensatz gefiltert, sodass nur Personen aus dem westdeutschen Erhebungsgebiet im Datensatz vorhanden bleiben. Dies hat mit der bereits besprochenen Gewichtung zu tun: Personen aus Ostdeutschland sind im Datensatz überrepräsentiert. Wenn Aussagen auf die Grundgesamtheit übertragen werden sollen, muss eigentlich eine Gewichtung vorgenommen werden. Da diese erst an späterer Stelle im Skript ausführlicher besprochen wird, wird in diesem Beispiel nur die westdeutsche Bevölkerung analysiert. Innerhalb der beiden Populationen ist die Datenerhebung repräsentativ und Aussagen können auf die Grundgesamtheit übertragen werden.

```
allbus_west <- allbus %>%
  filter(westost == "Westdeutschland")
```

8.4.2 Hintergrund

Für einen Vergleich der Mittelwerte wird ein anderes Testverfahren benötigt: die (einfaktorielle) Varianzanalyse (ANOVA). Auch bei dieser wird wieder die Nullhypothese auf Gleichheit überprüft: "Die Mittelwerte der Gruppen unterscheiden sich nicht". Die Testvariablen müssen wiederum intervallskaliert sein und die Gruppierungsvariable als *Faktor* vorliegen.

Bei einer ANOVA wird ein F-Wert berechnet, der in der Ausgabe wiederum mit einem p-Wert verknüpft wird. Warum werden aber nicht mehrere t-Tests berechnet?

Bei einem t-Test werden Aussagen mit einer Irrtumswahrscheinlichkeit von 5% getroffen. Möchte man nun mehrere t-Test berechnet, so addieren sich diese Irrtumswahrscheinlichkeiten auf (sog.

Alphafehler-Kumulation). Je mehr Einzelwerte miteinander verglichen werden, desto größer ist die Alphafehler-Kumulation. In der Berechnung der ANOVA wird dem Rechenschaft getragen und es werden Alphafehler-Korrekturen einberechnet.

Der F-Wert berechnet sich aus dem Anteil der erklärten Varianz des Modells im Verhältnis zur unerklärten Varianz:

$$F = \frac{MS_M}{MS_R}$$

- MS(M) steht dabei für die *erklärte Abweichung des Modells* (Mean Square Model) und wird aus der Summe der quadrierten Abweichung zwischen Mittelwert der Fakorstufen und Gesamtmittelwert berechnet.
- MS(R) bezeichnet die *nicht erklärte Abweichung des Modells* (Mean Square Residuals) und wird aus der Summe der quadrierten Abweichungen innerhalb der Faktorstufen berechnet.

Die Quadratsummen sind dabei wiederum sensibel auf die Samplegröße insgesamt und der Samplegröße der einzelnen Faktorstufen. Eine größere Stichprobe führt zu genaueren Schätzungen.

8.4.3 Durchführung

Zunächst findet wieder eine Prüfung auf Varianzhomogenität statt. Auch hier wird der Levene-Test angewendet. Zur Erinnerung. Die Hypothese lautet: Das Vertrauen in den Bundestag unterscheidet sich nach dem schulischen Bildungsgrad der Befragten.

```
leveneTest(allbus_west$pt03 ~ allbus_west$schulabschluss)
## Levene's Test for Homogeneity of Variance (center = median)
##         Df F value Pr(>F)
## group    2  2.1723 0.1142
##       2253
```

Im Beispiel ist Varianzhomogenität gegeben, da `p>.05` ist. Es kann demnach eine "klassische" ANOVA berechnet werden. Hierfür wird in R die Funktion `aov()` benutzt. Das Ergebnis der ANOVA wird in einem Objekt abgespeichert.

Hinweis: Bei Varianzheterogenität muss der Befehl (oneway.test) verwendet werden (Welch-Test). Ansonsten ist der one-way Test analog zu den folgenden Ausführungen zu interpretieren.

```
anova1 <- aov(allbus_west$pt03 ~ allbus_west$schulabschluss)
```

Zunächst erfolgt wieder eine Analyse der deskriptiven Mittelwerte um eine Voreinschätzung über mögliche Mittelwertunterschiede zu erhalten.

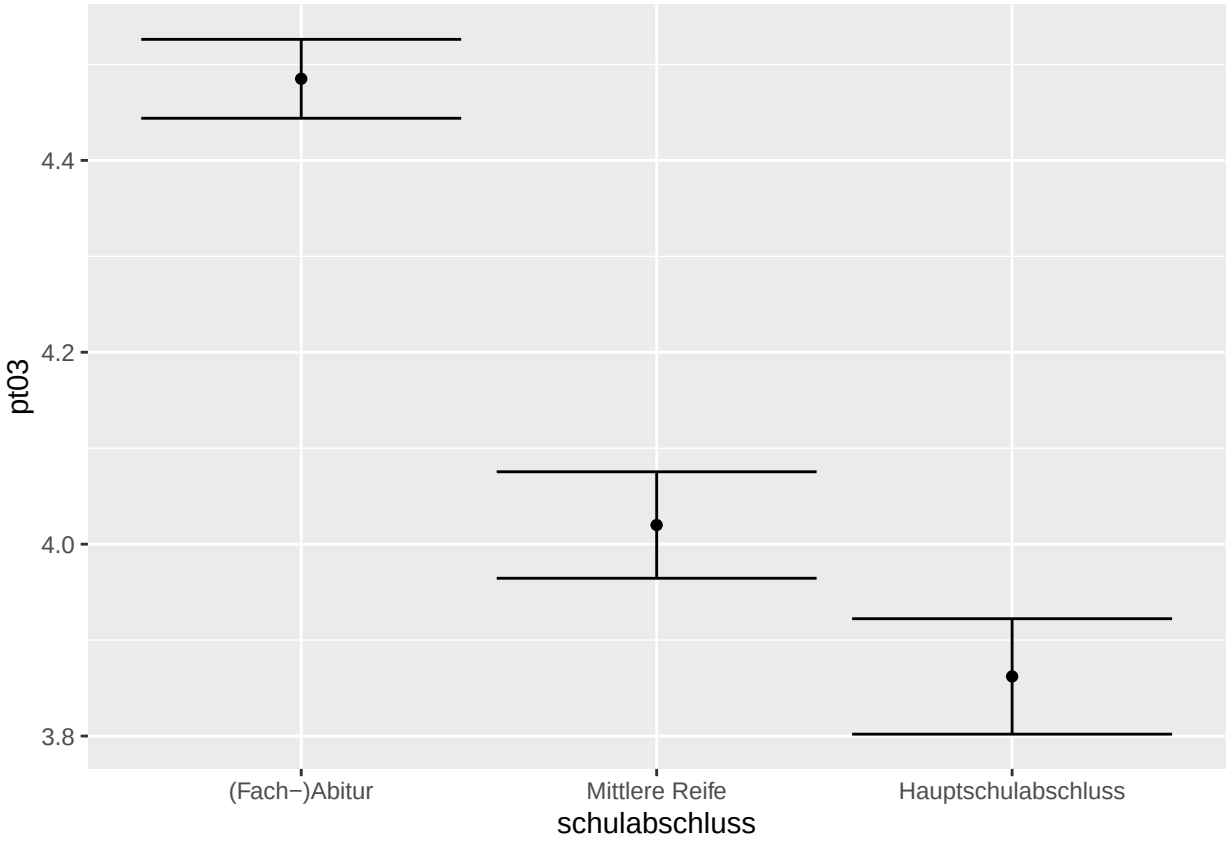
```
model.tables(anova1, "means")
## Tables of means
## Grand mean
```

```
##
## 4.184397
##
##  allbus_west$schulabschluss
##     (Fach-)Abitur Hauptschulabschluss Mittlere Reife
##             4.485               3.862           4.02
## rep      1002.000             602.000         652.00

# alternativer base R Code
mean(allbus_west$pt03, na.rm = TRUE)
## [1] 4.189841
tapply(allbus_west$pt03, allbus_west$schulabschluss, mean, na.rm = TRUE)
##       (Fach-)Abitur Hauptschulabschluss      Mittlere Reife
##            4.485030            3.862126            4.019939
```

Diese können auch mit `ggplot` visualisiert werden. Dafür wird ein Visualisierungsdatensatz angelegt. Mit der Funktion `na.omit` können fehlende Fälle aus dem Datensatz gelöscht werden, da diese sonst im Plot angezeigt werden. Schließlich werden die Ausprägungen der Schulabschluss-Variable zugeordnet und das Skalenniveau der Vertrauensvariable in numerisch verändert.

In der Plotfunktion wird anschließend als Darstellungsart ein *Punktdiagramm* gewählt, welches den Mittelwert der einzelnen Ausprägungen abträgt sowie über die weitere Ebene (layer) `geom_errorbar()` die Standardfehler der Mittelwerte.

```
vis_data <- allbus_west %>%
  select(schulabschluss, pt03)
vis_data <- na.omit(vis_data)
vis_data$schulabschluss <- ordered(vis_data$schulabschluss,
                         levels = c("(Fach-)Abitur", "Mittlere Reife",
                                    "Hauptschulabschluss"))
vis_data$pt03 <- as.numeric(vis_data$pt03)

ggplot(vis_data, aes(x=schulabschluss, y=pt03)) +
  geom_point(stat="summary", fun="mean") +
  geom_errorbar(stat="summary", fun.data="mean_se")
```



Die Tabelle zeigt sowohl den Gesamtmittelwert an (4.18) als auch die Mittelwerte für die Befragten der einzelnen Bildungsstufen. Es kann der Tabelle weiterhin entnommen werden, dass die Befragten mit Abiturabschluss tendenziell ein größeres Vertrauen in den Bundestag haben als Befragte mit einem niedrigeren Bildungsabschluss. Dies kann auch sehr gut in der Visualisierung nachvollzogen werden.

In einem nächsten Schritt wird überprüft, ob diese Vermutung statistisch haltbar ist. Dazu wird die `summary()` Funktion benutzt.

```
anova1
## Call:
##    aov(formula = allbus_west$pt03 ~ allbus_west$schulabschluss)
##
## Terms:
##                 allbus_west$schulabschluss Residuals
## Sum of Squares                     170.718  4312.573
## Deg. of Freedom                          2      2253
##
## Residual standard error: 1.383527
## Estimated effects may be unbalanced
## 131 observations deleted due to missingness
summary(anova1)
##                              Df Sum Sq Mean Sq F value Pr(>F)
## allbus_west$schulabschluss    2    171   85.36   44.59 <2e-16 ***
```

```
## Residuals                  2253   4313    1.91
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 131 observations deleted due to missingness
```

Im Output werden mehrere Werte ausgegeben: für Freiheitsgrade (df), die Summe der quadrierten Abweichungen (Sum Sq) sowie der quadrierten Mittelwertabweichungen (Mean Sq) und den daraus berechneten F-Wert. Die Ergebnisse sind getrennt nach Testvariablen (erklärte Varianz) und Residuen (Residuals). Der F-Wert und der daraus resultierende p-Wert berechnet sich aus diesen Werten. Außerdem zeigt R an, dass 131 Fälle aus der Analyse ausgeschlossen wurden, da fehlende Werte vorlagen.

Für die Interpretation der Ergebnisse ist der p-Wert von Interesse (`Pr(>F)`). Ist dieser unter .05 kann von einer statistischen Signifikanz gesprochen werden und die Nullhypothese wird abgelehnt. R zeigt zur Vereinfachung in der F-Wert Testausgabe Sternchen bei signifikanten Werten an. In diesem Fall ist der p-Wert signifikant Es gibt also signifikante Mittelwertunterschiede zwischen den Gruppen. Allerdings zeigt uns die Statistik nicht an, welche Mittelwerte sich signifikant voneinander unterscheiden - es wird lediglich anzeigt, dass es Mittelwertunterschiede über das gesamte Modell gibt.

8.4.4 Post-Hoc Tests

Für die Überprüfung, ob sich die Mittelwerte einzelner Gruppen signifikant voneinander unterscheiden, werden Post-Hoc Tests durchgeführt. In der Logik der Interpretation, sind diese sehr ähnlich zu den t-Tests. Allerdings wurde hier eine Korrektur des Alpha-Fehlers mit einberechnet. Es existieren sehr viele verschiedene Post-Hoc Korrekturen, die je nach Samplezusammentstellung angewendet werden sollten. In diesem Skript wird mit dem *Bonferroni-Test* gearbeitet (weitere Tests können über das `p.adj` Argument durchgeführt werden).

Post-hoc Tests werden mit der Funktion `pairwise.t.test()` angefordert.

```
pairwise.t.test(allbus_west$pt03, allbus_west$schulabschluss,
                p.adj = "bonferroni")
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  allbus_west$pt03 and allbus_west$schulabschluss
##
##                    (Fach-)Abitur Hauptschulabschluss
## Hauptschulabschluss < 2e-16       -
## Mittlere Reife      8.9e-11       0.13
##
## P value adjustment method: bonferroni
```

Das Ergebnis zeigt eine Matrix mit den einzelnen p-Werten für die Vergleiche. Die Ergebnisse zeigen, dass die p-Werte im Vergleich zwischen Befragten mit Abitur und Hauptschulabschluss sowie mittlere Reife hoch signifikant sind ($p<.05$). Zwischen Personen mit Hauptschulabschluss

und mittlere Reife Abschluss besteht allerdings kein signifikanter Mittelwertunterschied, da p=0.13 (p>.05).

Hinweis: Je nach Post-Hoc Korrektur kann die Interpretation der Ergebnisse anders ausfallen. Der Bonferroni-Test gilt als sehr konservativ. Wendet man beispielsweise den Holm-Test im oben genannten Beispiel an, findet man signifikante Unterschiede auch zwischen Personen mit Realschul- und Hauptschulabschluss.

8.4.5 Zusammenfassung

In Bezug auf die Fragestellung, ob sich das mittlere Vertrauen in den Bundestag in der westdeutschen Bevölkerung nach dem schulischen Bildungsgrad unterscheidet, wurde Folgendes herausgefunden. Es gibt signifikante Unterschiede bzgl. des Vertrauens in den Bundestag zwischen Personen mit unterschiedlichen Bildungsabschlüssen, F(2253)=44.59, p<.05. Der Post-Hoc Test mit Bonferroni-Korrektur zeigt, dass Personen mit (Fach-)Abitur mehr Vertrauen in den Bundestag haben (M=4.48) als Personen mit einem Abschluss der mittleren Reife (M=4.02) oder Hauptschulabschluss (M=3.86). Personen mit den letztgenannten Schulabschlüssen unterscheiden sich dagegen nicht signifikant in ihrem Vertrauen in den Bundestag voneinander.

8.4.6 Weiteres zur Varianzanalyse

Dies stellt nur eine (vereinfachte) Einführung in die breite Thematik der Varianzanalyse dar. In der empirischen Forschungspraxis können viele weitere Optionen eine wichtige Rolle spielen. Dies umfasst beispielsweise rechnerische Zusätze wie Kontraste und Effektstärken. Außerdem gibt es verschiedene Arten der Varianzanalyse, wie zum Beispiel Mixed Designs, Varianzanalysen mit Messwiederholungen, usw. Maike Luhmann gibt im Buch *R für Einsteiger* (Luhmann, 2015) einen guten Überblick über weitere Verfahren. Ebenso ist *Discovering Statistics Using R* (Field, Miles & Field, 2012) empfehlenswert, um einen vertiefenden Einblick zu erhalten.

9 Kreuztabellen und statistische Unabhängigkeit

Dieses Kapitel widmet sich Kreuztabellen und dem Chi-Quadrat-Test als Verfahren zur Prüfung auf statistische Unabhängigkeit. Die zentrale Fragestellung hinter Kreuztabellen lautet: Gibt es Unterschiede in der **Verteilung** zwischen zwei Variablen. Dabei sollten die Variablen eine geringe Anzahl an Ausprägungen aufweisen und daher in der Regel nominalskaliert sein.

9.1 Wiederholung: Deskripitve Statistik

Kreuztabellen wurden bereits in Kapitel 5 vorgestellt. Kreuztabellen sind eine Darstellung der Ausprägungen zweier Variablen. Jede Ausprägung der Variable X wird dabei mit jeder Ausprägung der Variable Y abgeglichen und die Gesamtanzahl der Nennungen in die einzelne Zelle übertragen. Als Beispiel wird die reduzierte Schulabschlussvariable aus dem vorigen Kapitel mit dem Geschlecht der Befragten in Zusammenhang gesetzt.

```
library(haven)
allbus <- as.data.frame(read_spss("ALLBUS2018.sav"))

allbus$westost <- factor(allbus$eastwest,
                         labels = c("Westdeutschland", "Ostdeutschland"))

library(tidyverse) # alternativ (dplyr)
allbus <- allbus %>%
  mutate(schulabschluss = case_when(educ == 2 ~ "Hauptschulabschluss",
                                    educ == 3 ~ "Mittlere Reife",
                                    educ %in% c(4,5) ~ "(Fach-)Abitur"))
```

```
absolut <- table(allbus$schulabschluss, allbus$westost)
absolut
##
##                      Westdeutschland Ostdeutschland
##   (Fach-)Abitur                 1020            357
##   Hauptschulabschluss            621            189
##   Mittlere Reife                 669            526
```

Häufig interessiert darüber hinaus nun, wie viele Fälle insgesamt in der Ausprägung einer Variablen vorhanden sind. Dazu können mit `addmargins()` die Summenwerte hinzugefügt werden.

```
absolut_summe <- addmargins(absolut)
absolut_summe
##
##                      Westdeutschland Ostdeutschland  Sum
##   (Fach-)Abitur                 1020            357 1377
##   Hauptschulabschluss            621            189  810
##   Mittlere Reife                 669            526 1195
##   Sum                           2310           1072 3382
```

Leider gibt auch diese Tabelle wenig Aufschluss über die eigentliche *Verteilung der Werte*. Um diese zu erhalten, werden Spaltenprozentwerte auf Grundlage der relativen Häufigkeiten in den Spalten berechnet. Eine relative Häufigkeitstabelle in Prozent wird mit `100*prop.table(x)` berechnet, wobei `x` für die originäre Tabelle steht. Die Zeilenprozente können anschließend durch das Argument `1` ausgegeben werden. Spaltenprozentwerte können analog durch das Argument `2` ausgewählt werden. Sofern die Variablen in eine abhängige und eine unabhängige Variable unterschieden werden können, wird die unabhängige Variable standardmäßig in die Spalten geschrieben. Anschließend können die Dezimalstellen können mit dem Befehl `round()` und der Angabe einer beliebigen Anzahl an Dezimalstellen zusätzlich gerundet werden. Dies ergibt folgende Funktion:

```
spaltenprozente <- round(100*prop.table(absolut, 2),2)
spaltenprozente
##
##                       Westdeutschland Ostdeutschland
##   (Fach-)Abitur                 44.16          33.30
##   Hauptschulabschluss           26.88          17.63
##   Mittlere Reife                28.96          49.07
```

Nun sind die Werte besser interpretierbar. In Westdeutschland haben circa 44% der Befragten das Abitur abgelegt, während dies in Ostdeutschland nur 33% sind. In Ostdeutschland lebende Personen haben zu 49% die "Mittlere Reife" abgelegt und nur 18% haben einen Hauptschulabschluss. In Westdeutschland haben 29% einen Abschluss der Mittleren Reife und 27% einen Hauptschulabschluss.

9.2 Test auf statistische Unabhängigkeit

Nun soll überprüft werden, ob die empirischen Werte zufällig beobachtet wurden oder ob sie auf eine Abhängigkeit zwischen den beiden Variablen hindeuten. Die Frage ist also, ob das Bildungsniveau der Befragten statistisch unabhängig vom Erhebungsort (Ost- und Westdeutschland) ist, oder anders ausgrdrückt, ob die Bildungsabschlüsse in Ost- und Westdeutschland ungleich verteilt sind. Um dies statistisch zu prüfen, wird der Chi-Quadrat-Test auf statistische Unabhängigkeit genutzt.

Der Chi-Quadrat-Wert basiert auf der Differenz zwischen den beobachteten Häufigkeiten und den bei statistischer Unabhängigkeit erwarteten Häufigkeiten der Variablen. Die erwartete Häufigkeit für jede Zelle errechnet sich aus dem Produkt der Randverteilungen geteilt durch die Größe der Gesamtstichprobe. Auf Basis eines kritischen Chi-Quadrat-Werts wird anschließend die Signifikanz des Tests bestimmt (p-Wert). Der kritische Chi-Quadrat-Wert zur Bestimmung der Signifikanz ist von der Anzahl der Freiheitsgrade und des Testniveaus abhängig. In der Regel gilt eine Irrtumswahrscheinlichkeit von 5% als akzeptabel in den Sozialwissenschaften.
Die Freiheitsgrade für einen solchen Chi-Quadrat-Test werden durch die Anzahl der Zeilen und Spalten einer Kreuztabelle bestimmt:

(Anzahl der Zeilen - 1) x (Anzahl der Spalten - 1). In diesem Falle gibt es zwei Freiheitsgrade ((3-1) x (2-1)). Der kritische Chi-Quadrat für einen Test mit zwei Freiheitsgraden und einer angenommenen Irrtumswahrscheinlichkeit von 5% liegt bei 5.99. Ein Chi-Quadrat-Wert, der diesen kritischen Wert überschreitet, wird durch den p-Wert als signifikant markiert.

$$\chi^2 = \sum_{j=1}^{m} \frac{(N_j - n_{oj})^2}{n_{0j}}$$

Der Chi-Quadrat-Test überprüft dabei die Nullhypothese ("die Verteilung ist unbhängig von einander"). Wird ein signifikanter Testwert erzielt, so wird die Nullhypothese abgelehnt und die Alternativhypothese ("die Verteilung ist nicht unabhängig") angenommen.

9.3 Zusammenhangsmaße

Darüber hinaus interessiert meistens auch noch die Zusammenhangsstärke. Während der Chi-Quadrat-Test nur eine Aussage darüber trifft, ob eine statistische Unabhängigkeit wahrscheinlich ist, gibt ein Zusammenhangsmaß an, wie stark der Zusammenhang zwischen zwei Variablen ist. Es existieren mehrere Zusammenhangsmaße, welche je nach Skalenniveau der Variablen Verwendung finden:

- Zwei dichotome Variablen: phi (ϕ)
- Eine dichotome und eine nominale Variable, zwei nominale Variablen, eine ordinale und eine dichotome oder nominale Variable: **Cramer's V**
- Zwei mindestens ordinalskalierte Variablen oder eine ordinale und eine mindestens intervallskalierte Variable: **Spearman's rho** (ρ) oder **Kendall's tau-b** (τ-b) oder **tau-c**(τ-c)
- Zwei mindestens intervallskalierte Variablen: **Pearson's r**

Die Zusammenhangsmaße haben gemeinsam, dass die Ergebnisse immer zwischen 0 und 1 bzw. -1 und 1 liegen. Je höher der Betrag des Wertes, desto stärker ist der Zusammenhang. In den Sozialwissenschaften kann man ab einer Zusammenhangsstärke von .20 von einem interpretierbaren Zusammenhang sprechen. Zusammenhänge ab einer Stärke von .50 sind als stark einzustufen. Zeigt das Zusammenhangsmaß exakt 1, so besteht ein perfekter Zusammenhang.

Achtung: Nur, weil statistisch ein Zusammenhang zwischen Variablen besteht, muss dies nicht gleich bedeuten, dass eine Aussage darüber auch logisch Sinn ergibt bzw. ein tatsächlicher Zusammenhang besteht!

Pearson's Chi-Quadrat-Test ist standardmäßig in base R mit der Funktion `chisq.test()` implementiert. Um das Zusammenhangsmaß Cramer's-V zu erhalten, können verschiedene Pakete geladen werden. Es eignet sich das Paket `vcd`, da beide Tests im `vcd` Package gefunden werden.

Mit der Funktion `assocstats()` kann der Chi-Quadrat-Test auf Unabhängigkeit sowie das Zusammenhangsmaß Cramer's-V ausgegeben werden.

```
chisq.test(absolut)
##
##  Pearson's Chi-squared test
##
## data:  absolut
## X-squared = 131.13, df = 2, p-value < 2.2e-16

# install.packages("vcd")
library(vcd)
## Loading required package: grid
assocstats(absolut)
##                     X^2 df P(> X^2)
```

```
## Likelihood Ratio 128.93  2        0
## Pearson          131.13  2        0
##
## Phi-Coefficient   : NA
## Contingency Coeff.: 0.193
## Cramer's V        : 0.197
```

Die Ergebnisse zeigen, dass χ^2 nach Pearson einen Wert von 131.13 aufweist und mit p<.05 signifikant ist. Cramer's V gibt darüber hinaus an, dass die Zusammenhangsstärke bei .197 liegt. Dies deutet auf einen schwachen Zusammenhang hin.

Es lässt sich also feststellen, dass die gefundenen Unterschiede in der Verteilung der Bildungsabschlüsse zwischen Ost- und Westdeutschland keine zufällige Variation abbilden und mit hoher Wahrscheinlichkeit ein Zusammenhang zwischen den beiden Variablen besteht.

10 Korrelationen

In diesem Kapitel werden Zusammenhangsmaße zwischen Variablen behandelt. Zu den wichtigsten Zusammenhangsmaßen für metrisch- und ordinalskalierte Variablen zählen Pearson's r, Kendall's tau und Spearman's Rho.

```
# Pakete laden und Daten einlesen
library(tidyverse)
library(haven)

allbus <- as.data.frame(read_spss("ALLBUS2018.sav"))
allbus_kurz <- head(allbus, n = 20)
```

10.1 Pearson's r

Für zwei Variablen mit einem metrischen Skalennivau kann Pearson's r als Maßzahl für den Zusammenhang zwischen den beiden Variablen genutzt werden. Zur Berechnung des Korrelationskoeffizienten wird zunächst die Kovarianz zwischen den Variablen X und Y ermittelt:

$$cov(x;y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x}) * (y_i - \overline{y})$$

Die Kovarianz ist abhängig von der Skalierung der Variablen, deshalb wird eine Normierung durchgeführt, die den Korrelationskoeffizienten r ergibt. Dazu wird die Kovarianz duch die Standardabweichung von X und Y geteilt:

$$r = \frac{cov(x;y)}{s_x * s_y}$$

Der nun berechnete Koeffizient nimmt einen Wert im Bereich $-1; +1$ an. Im folgenden Beispiel wird ein Zusammenhang zwischen dem Vertrauen in den Bundestag und dem Vertrauen in die politischen Parteien berechnet. Die Hypothese lautet: Je stärker eine Person in politische Parteien vertraut, desto stärker ist ihr Vertrauen in den Bundestag. Da kein gerichteter Zusammenhang getestet werden kann, lässt sich ebenso annehmen: Je stärker eine Person dem Bundestag vertraut, desto stärker vertraut sie auch den politischen Parteien.

In diesem Beispiel wird beiden Variablen ein metrisches Skalenniveau unterstellt, sodass Pearson's r berechnet werden kann.

10.1.1 Vorbereitungen

Zunächst werden die benötigten Variablen deskriptiv analysiert, bevor der Zusammenhang untersucht wird. Das sind aus dem ALLBUS-Datensatz die Variablen `pt15` für das Vertrauen in politische Parteien und `pt03` für das Vertrauen in den Bundestag. Mit dem Argument `useNA = 'ifany'` wird zusätzlich angezeigt, wie viele Befragte die Frage nicht beantwortet haben. Mit dem `summary()`-Befehl werden zudem einige deskriptive Statistiken ausgegeben, die ein erstes Bild über die Verteilung geben.

```
table(allbus$pt15, useNA = "ifany")
##
##    1    2    3    4    5    6    7 <NA>
##  260  488  868 1090  556  104   15   96
summary(allbus$pt15, useNA = "ifany")
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   1.000   3.000   4.000   3.463   4.000   7.000      96
table(allbus$pt03, useNA = "ifany")
##
##    1    2    3    4    5    6    7 <NA>
##  198  316  570  934  857  398  113   91
summary(allbus$pt03, useNA = "ifany")
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   1.000   3.000   4.000   4.058   5.000   7.000      91
```

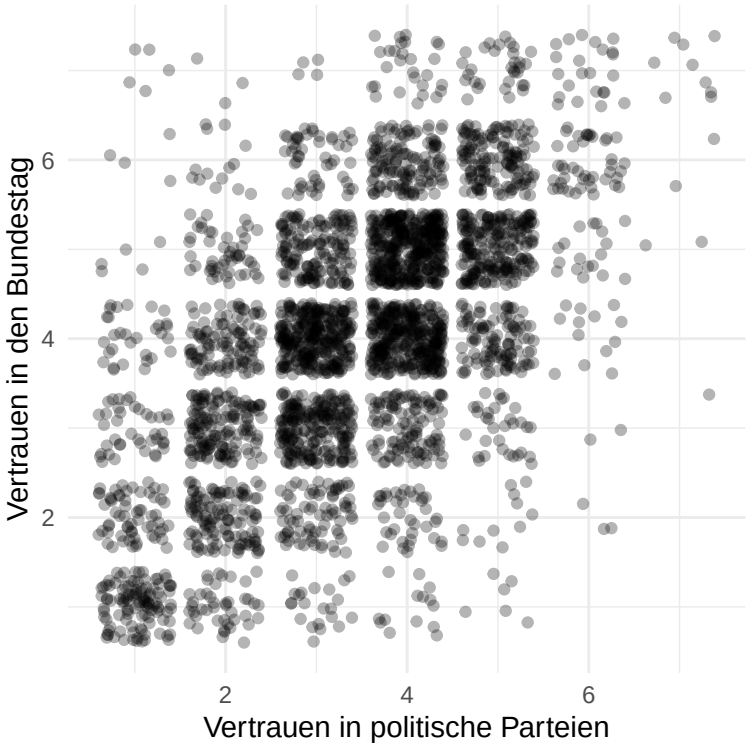
Die Verteilung zeigt, dass jeweils etwas mehr als 90 Befragte die Frage nicht beantwortet haben, das müssen wir in folgenden Analysen beachten.

10.1.2 Grafische Darstellung

Um einen ersten Eindruck über den Zusammenhang zwischen den Variablen zu erhalten, wird ein Streudiagramm erstellt. Die einzelnen Punkte des Diagramms stehen für die Befragten, die beide Fragen beantwortet haben. Das Diagramm können wir mit `ggplot` erstellen. Dort verwenden wir dann zusätzlich bekannterweise `geom_point()`. Die Punkte liegen aufgrund der diskreten Skala alle übereinader und sind so nicht zu interpretieren. Daher wird das Argument `position = "jitter"` genutzt, um die Punkte leicht nach rechts und links schwingen zu lassen. Das Argument `alpha = 0.3` macht die einzelnen Punkte transparenter. Das ist hier aufgrund der hohen Fallzahl empfehlenswert.

Der Plot sieht wie folgt aus:

```
ggplot(allbus, aes(x= pt15,
                   y = pt03)) +
  geom_point(position = "jitter", alpha = 0.3) +
  labs(x = "Vertrauen in politische Parteien",
       y = "Vertrauen in den Bundestag") +
  theme_minimal()
```



Es zeigt sich, dass wenige Befragte gleichzeitig ein niedriges Vertrauen in die politischen Parteien und ein hohes Vertrauen in den Bundestag haben (Bereich oben links im Diagramm). Personen, die ein hohes Vertrauen in politische Parteien haben, haben in der Mehrzahl auch ein hohes Vetrauen in den Bundestag (Bereich oben rechts im Diagramm). Die grafische Analyse stützt die Annahme über einen positiven Zusammenhang zwischen den Variablen.

10.1.3 Korrelation berechnen

Das erstellte Diagramm weißt darauf hin, dass es einen Zusammenhang zwischen dem Vertrauen in politische Parteien und dem Vertrauen in den Bundestag gibt.

Um diesen Zusammenhang in einer Maßzahl zu beschreiben, wird nun Pearson's r berechnet. Um auch die Signifikanz des Zusammenhangs ausgegeben zu bekommen, wird der Befehl `cor.test()` genutzt. In diesem Befehl werden – durch ein Komma getrennt – die beiden Variablen angeben, für die der Zusammenhang berechnet wird.

```
cor.test(allbus$pt15,
         allbus$pt03)
##
##  Pearson's product-moment correlation
##
## data:  allbus$pt15 and allbus$pt03
## t = 41.087, df = 3323, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
```

```
##  0.5574159 0.6025157
## sample estimates:
##       cor
## 0.5804107
```

Für die Interpretation des Ergebnis sind nun drei Punkte wichtig:

- **Stärke des Zusammenhangs:** Wie stark eine Variable mit der anderen Variable zusammenhängt, wird in der Höhe des Koefizienten angegeben. Je höher dieser Wert ist, desto stäker wird eine Variable durch die andere Variable bestimmt. Der Wert kann zwischen `0` und `+-1` liegen. Mit `0.58` ist der Zusammenhang zwischen den beiden Variablen hier relativ hoch und positiv. Ein Wert von `+1` zeigt einen perfekten positiven Zusammenhang auf, ein Wert von `-1` einen perfekten negativen Zusammen und ein Wert von `0` deutet auf keinen Zusammenhang hin.
- **Richtung des Zusammenhangs:** Hier kommt es auf das Vorzeichen des Koeffizienten an. Bei einem `+` sprechen wir von einem positiven Zusammenhang, während wir bei einem `-` von einem negativen Zusammenhang sprechen. In unserem Beispiel ist der Zusammenhang positiv: Je mehr eine Person in politische Parteien vertraut, desto mehr vertraut sie in den Bundestag (oder andersrum!). Wenn der Zusammenhang negativ wäre, würden die Richtung des Zusammenhangs wie folgt beschrieben werden: Je mehr eine Person in politische Parteien vertraut, desto weniger vertraut sie in den Bundestag (oder andersrum).
- **Signifikanztest:** Hier wird überprüft, ob auch in der Grundgesamtheit von einem Zusammenhang zwischen den beiden Variablen ausgehen können. Wenn der `p-value` (p-Wert) kleiner als `0.05` (0,05) ist, können wir davon ausgehen, dass ein Zusammenhang zwischen den beiden Variablen auch in der Grundgesamtheit vorliegt. Dies ist in dem vorliegenden Beispiel gegeben.

Alles in allem scheint unsere Analyse die Annahme, dass Personen, die politischen Parteien vertrauen, auch dem Bundestag vertrauen, Unterstützung in den Daten zu finden.

10.2 Spearman's Rho und Kendall's tau

Um den Zusammenhang zwischen zwei Variablen zu berechnen, für die höchstens ein ordinales Skalenniveau vorliegt, können die Kennwerte Spearman's Rho oder Kendall's tau berechnet werden. Auch diese Werte können mit dem Befehl `cor.test()` berechnet werden.

In der Forschung besteht eine Kontroverse über die korrekte Zuordnung von Likert-Skalen zu Skalenniveaus. Streng genommen sind sie ordinalskaliert, oftmals jedoch werden sie in den Sozialwissenschaften als quasi-metrisch behandelt. In diesem Beispiel könnte man argumentieren, dass die Abstände zwischen den einzelnen Antwortkategorien nicht gleich groß sein müssen.

Die Korrelationsmaße Spearman's Rho und Kendalls tau können ebenso berechnet werden. Diese Kennwerte können mit der Spezifizierung des Arguments `method = "kendall"` oder `method = "spearman"` in dem Befehl `cor.test()` ausgegeben werden.

```
# cor.test(allbus$pt15,
#          allbus$pt03,
#          method = "kendall")
```

```
#cor.test(allbus$pt15,
#         allbus$pt03,
#         method = "spearman")
```

Der Wert für Kendall's tau ist kleiner als der Wert für Spearman's Rho. Dass sich beide Werte unterscheiden, liegt an der unterschiedlichen Berechnung der beiden Maße. Spearman's Rho ist ähnlich hoch wie der Wert für Pearson's r. Hier sei anzumerken, dass das korrekte Korrelationsmaß im Vorfeld der Analyse ausgewählt werden sollte und ein Vergleich von Korrelationsmaßen in der Regel nicht sinnvoll ist. Eine Beschreibung der Unterschiede zwischen den beiden Korrelationsmaßen und ihre Anwendungsgebiete, findet sich bei Field (2009).

10.3 Die Korrelationsmatrix

Ebenso können bivariate Korrelationen zeitgleich für mehrere Variablen in einer sogenannten Korrelationsmatrix dargestellt werden. In diesem Falle lässt sich eine Korrelationsmatrix mit dem Befehl `cor` erstellen. Die Funktion erwartet einen Vektor, eine Matrix oder einen Datensatz als Argument. In diesem Falle wird zunächst ein Datensatz aus mehreren Variablen von Interesse gebildet. Dieser Datensatz wird dann als Argument in die Funktion gegeben, um eine Korrelationsmatrix zu erhalten. Das Argument `use` beschreibt den Umgang mit fehlenden Werten. Das Argument `method` legt fest, welcher Korrelationskoeffizient berechnet wird. Im nächsten Code-Chunk wird Pearson's r für alle paarweisen Beobachtungen eines Datensatzes zum politischen Vertrauen berechnet.

```
vertrauen <- subset(allbus, select = pt03:pt15) # Datensatz zum pol. Vertrauen
cor(vertrauen, use = "pairwise.complete.obs", method = "pearson")
##           pt03      pt04      pt08      pt09      pt10      pt11      pt12
## pt03 1.0000000 0.4018623 0.5418264 0.3695336 0.4158966 0.3527202 0.7012329
## pt04 0.4018623 1.0000000 0.3817825 0.2386091 0.2701777 0.3061913 0.3624219
## pt08 0.5418264 0.3817825 1.0000000 0.3040348 0.3622916 0.3919584 0.5107834
## pt09 0.3695336 0.2386091 0.3040348 1.0000000 0.6690724 0.2531800 0.3726665
## pt10 0.4158966 0.2701777 0.3622916 0.6690724 1.0000000 0.3842970 0.4094042
## pt11 0.3527202 0.3061913 0.3919584 0.2531800 0.3842970 1.0000000 0.3740649
## pt12 0.7012329 0.3624219 0.5107834 0.3726665 0.4094042 0.3740649 1.0000000
## pt14 0.3944567 0.3566765 0.5011352 0.2752920 0.3092543 0.4009544 0.4296461
## pt15 0.5804107 0.3369428 0.4364346 0.4198514 0.4473540 0.3089789 0.6366093
##           pt14      pt15
## pt03 0.3944567 0.5804107
## pt04 0.3566765 0.3369428
## pt08 0.5011352 0.4364346
## pt09 0.2752920 0.4198514
## pt10 0.3092543 0.4473540
## pt11 0.4009544 0.3089789
## pt12 0.4296461 0.6366093
## pt14 1.0000000 0.3888976
## pt15 0.3888976 1.0000000
```

11 Indexbildung

Bisher wurde weitestgehend mit Variablen gearbeitet, bei denen einzelne Items (Indikatoren) die Konstrukte gemessen und damit die Information von Interesse geliefert haben (z.B. Alter, Geschlecht, Bildung). So wird die Bildung oftmals mit einer Frage nach dem Schulabschluss einer Person erfasst. Auch die daraus folgende Zuordnung zu einer Bildungsgruppe (z.B. niedrig, mittel, hoch) ist über diese einzelne Frage im Fragebogen erhoben worden.

In den Sozialwissenschaften wird sich allerdings nicht nur mit Variablen beschäftigt„ die mit einer einzelnen Messung abgebildet werden. Das Vertrauen in das politische System ist zum Beispiel ein Konstrukt, das häufig über mehrere Items erhoben wird. Um ein solches Konstrukt angemessen darzustellen, werden mehrere Einzelindikatoren zu einem Index zusammengefasst.

Das Zusammenfassen von einzelnen Items (Indikatoren oder Variablen) hat dabei mehrere Vorteile:

1. Erfassung mehrerer theoretischer Merkmale eines latenten Konstrukts: Bei Phänomenen, die sich aus mehreren Aspekten der sozialen Realität zusammensetzen, werden häufig Indizes genutzt. Diese erklären häufig mehr als die einzelnen Indikatoren. Allerdings sollte stets berücksichtigt werden, welche Nuancen durch einzelne Indikatoren abgebildet werden.

2. Reduktion sozialer Erwünschtheit: Soziale Erwünschtheit kann dazu beitragen, dass einige gesellschaftliche Phänomene durch einzelne Indikatoren nicht gut gemessen werden können. Daher wird häufig auf mehrere Einzelfragen beim Messen dieser Phänomene zurückgegriffen.

3. Reduktion des Messfehlers: Ein weiterer Vorteil ist eine Reduktion des Messfehlers. Durch die Verwendung mehrerer Indikatoren gleichen sich die Messfehler der einzelnen Indikatoren aus und die Messung gewinnt an Validität (Latcheva & Davidov, 2019).

11.1 Vorbereitung

Um eine Skala oder einen Index zu bilden, müssen die Variablen in einem inhaltlichen Sinnzusammenhang stehen. Ansonsten wären inhaltliche Schlüsse aus nachfolgenden Analysen kaum möglich. Ein erster Hinweis, dass die Variablen zu einem latenten Konstrukt gehören können, ist gegeben, wenn sich das Antwortverhalten der einzelnen Indikatoren ähnelt.

In der Umfrageforschung werden oftmals sogannente Fragebatterien eingesetzt, die mit mehreren Items ein latentes Konstrukt erfassen. Dieses Vorgehen wird im ALLBUS mehrfach angewendet, um Einstellungen zu erfassen. So lohnt sich beispielsweise ein Blick auf die Items mit den Kürzeln px01 bis px10.

Auf den ersten Blick lässt sich annehmen, dass die Einzelindikatoren “Wir sollten endlich wieder Mut zu einem starken Nationalgefühl haben”, “Ich bin stolz, ein Deutscher/eine Deutsche zu sein” und “Deutschland ist durch die vielen Ausländer in einem gefährlichen Maß überfremdet”, zu einem zugrundeliegenden Konstrukts gehören könnten und bestimmte Einstellungen (in diesem Fall rechtsextreme Einstellungen) abbilden.

Für die statistische Konsistenzprüfung und vor allem eine spätere Indexbildung sollten alle Variablen des Konstrukts in die gleiche Richtung codiert sein. Es sollte also im Vorfeld sichergestellt werden, dass die höheren Werte auf den gemessenen (manifesten) Variablen auch (wie in dem Beispiel) die stärkeren rechtsextremen Einstellungen abbilden. In diesem Beispiel ist dies der Fall, somit bedarf es keiner vorherigen Rekodierung der Variablen.

11.2 Konsistenz im Antwortverhalten (Cronbach's Alpha)

Wichtig für die Indexbildung ist es, dass die Befragten ein konsistentes Antwortverhalten auf den Variablen aufweisen. Würden die Items "Ich bin stolz, ein Deutscher/eine Deutsche zu sein" und "Deutschland ist durch die vielen Ausländer in einem gefährlichen Maß überfremdet" von den Befragten nicht mehrheitlich ähnlich beantwortet werden, gäbe es Gründe daran zu zweifeln, dass die Variablen dasselbe Konstrukt messen. Es gibt verschiedene Ansätze, mit denen herausgefunden werden kann, ob die einzelnen Fragestellungen ein zugrundeliegendes Konstrukt abbilden.

Die meistgenutzte Möglichkeit, die interne Konsistenz eines Messinstrumentes zu schätzen, ist das Bestimmen von Cronbach's Alpha.

Hinweis: Itembatterien in gängigen Fragebögen nutzen meist etablierte Messungen. Es ist dennoch unerlässlich zu überprüfen, ob die ausgewählten Items ein zugrundeliegendes Konstrukt darstellen.

Die Maßzahl Cronbachs Alpha gibt an, ob die einzelnen Indikatoren eine interne Konsistenz aufweisen. Es ergibt sich aus der Anzahl der Komponenten und den Korrelationen zwischen den jeweiligen Komponenten.

Dabei wird lediglich die interne Konsistenz gemessen. Alpha kann nicht dabei helfen, herauszufinden, ob die Einzelindikatoren eindimensional auf ein Konstrukt laden oder ob es sich gegebenfalls um ein Konstrukt mit mehreren Dimensionen handelt. Um dies zu überprüfen, müssten Hauptkomponenten- und (konfirmatorische) Faktoranalysen durchgeführt werden.

Die Berechnung kann mit dem Befehl `alpha()` aus dem Paket psych durchgeführt werden.

```
# Pakete laden und Daten einlesen
library(haven)
# install.packages("psych")
library(psych)
allbus <- as.data.frame(read_spss("ALLBUS2018komplett.sav"))
```

Nun können die entsprechenden Variablen, die zu einem Index zusammengefasst werden sollen, aus dem Datensatz herausgesucht werden. Für dieses Skript interessieren die Variablen zu rechtsextremistischen Einstellungen aus dem ALLBUS 2018. Diese tragen die Bezeichnungen px01, px02, px03, px04, px05, px06, px07, px08, px09 und px10. Um übersichtlich zu arbeiten, wird ein separater Datensatz generiert, der nur die Variablen zu rechtsextremistischen Einstellungen speichert, die an dieser Stelle von Interesse sind. Dies ist kein notwendiger Schritt, sondern dient der Übersichtlichkeit.

```
extremismus <- subset(allbus, select = px01:px10)
```

Es kann nun getestet werden, ob befragte Personen konsistent auf diese Items geantworte haben. Um Cronbachs Alphaz zu berechnen, wird die Funktion `alpha()` verwendet.

```
alpha(extremismus)
##
## Reliability analysis
## Call: alpha(x = extremismus)
##
##   raw_alpha std.alpha G6(smc) average_r S/N    ase mean  sd median_r
```

```
##       0.82      0.82    0.83      0.31 4.4 0.0045  2.3 0.7      0.3
##
##  lower alpha upper     95% confidence boundaries
## 0.81 0.82 0.83
##
##  Reliability if an item is dropped:
##      raw_alpha std.alpha G6(smc) average_r S/N alpha se var.r med.r
## px01      0.82      0.82    0.82      0.33 4.5   0.0045 0.012  0.32
## px02      0.81      0.81    0.81      0.32 4.2   0.0048 0.017  0.31
## px03      0.81      0.81    0.83      0.32 4.3   0.0047 0.018  0.31
## px04      0.79      0.79    0.81      0.30 3.8   0.0051 0.019  0.29
## px05      0.81      0.81    0.82      0.31 4.1   0.0048 0.019  0.31
## px06      0.79      0.79    0.81      0.29 3.7   0.0054 0.020  0.27
## px07      0.79      0.79    0.81      0.30 3.8   0.0052 0.019  0.30
## px08      0.79      0.79    0.80      0.29 3.7   0.0053 0.016  0.30
## px09      0.79      0.79    0.80      0.29 3.7   0.0052 0.016  0.30
## px10      0.81      0.81    0.83      0.32 4.2   0.0048 0.020  0.30
##
##  Item statistics
##         n raw.r std.r r.cor r.drop mean   sd
## px01 3250  0.47  0.46  0.40   0.33  4.0 1.07
## px02 3230  0.58  0.57  0.52   0.45  3.8 1.13
## px03 3223  0.52  0.53  0.44   0.39  1.6 1.02
## px04 3200  0.67  0.67  0.63   0.56  1.7 1.07
## px05 3161  0.56  0.57  0.50   0.44  1.6 1.07
## px06 3228  0.73  0.70  0.66   0.61  2.9 1.42
## px07 3231  0.67  0.67  0.61   0.56  1.9 1.17
## px08 3046  0.71  0.71  0.69   0.61  1.8 1.15
## px09 3093  0.70  0.70  0.68   0.60  1.7 1.03
## px10 3240  0.54  0.56  0.48   0.43  1.4 0.92
##
## Non missing response frequency for each item
##         1    2    3    4    5 miss
## px01 0.04 0.04 0.23 0.28 0.41 0.07
## px02 0.05 0.09 0.18 0.37 0.30 0.07
## px03 0.68 0.15 0.09 0.06 0.02 0.07
## px04 0.66 0.14 0.10 0.08 0.02 0.08
## px05 0.70 0.13 0.07 0.08 0.02 0.09
## px06 0.22 0.20 0.16 0.25 0.17 0.07
## px07 0.53 0.19 0.16 0.07 0.04 0.07
## px08 0.57 0.17 0.14 0.07 0.04 0.12
## px09 0.63 0.16 0.14 0.05 0.02 0.11
## px10 0.81 0.08 0.05 0.04 0.03 0.07
```

Ein hoher Wert von Cronbachs Alpha deutet auf eine hohe Konsistenz im Antwortverhalten hin. Dabei gilt ein Wert ab 0,8 als empfehlenswert. Allerdings werden Werte ab 0,7 häufig bereits als akzeptabel angesehen (Krebs & Menold, 2019, S. 495). Dies gilt vor allem, falls das Konstrukt

nur mit einer geringen Anzahl von Items erfasst wurde. Ist der berechnete Wert größer als dieser Grenzwert, kann davon ausgegangen werden, dass eine ausreichende Konsistenz im Antwortverhalten vorliegt. Nur wenn eine akzeptable Konsistenz vorhanden ist, sollte ein Index für weitere Analysen gebildet werden.

Der Wert, der in R als "raw alpha" ausgegeben wird, liegt bei 0,82 und ist damit als ausreichend konsistent zu bewerten. Die Variablen sind damit für eine Indexbildung geeignet.

11.3 Vorgehen bei der Indexbildung

Nachdem getestet wurde, ob das Antwortverhalten konsistent ist, kann mit der Indexbildung begonnen werden.

Es gibt verschiedene Wege einen Index zu bilden. In diesem Skript werden Möglichkeiten der Indexbildung vorgestellt, die alle Variablen, die ein Konstrukt messen sollen, gleichmäßig in die Konstruktion von Indizes einbezieht: der Mittelwertindex sowie der additive Index. Bei der Erstellung eines Indizes auf diese Weise werden alle Einzelindikatoren des Konstrukts gleichermaßen in den Index einbezogen, da keine Gewichtungen einzelner Indikatoren vorliegen. Itembatterien werden in der Regel ohne Gewichtungen einzelner Indikatoren zu einem Index zusammengefasst.

11.3.1 Mittelwertindex

Der Mittelwertindex wird errechnet, in dem der Mittelwert aus den Einzelitems des Konstrukts "rechtsextremistische Einstellungen" gebildet wird.

Ein Mittelwert einer Zeile über mehrere Spalten kann nun mit der Funktion `rowMeans()` gebildet werden. Durch eine Zuweisung wird dem Datensatz ein solcher Mittelwert als neue Variable mit dem Namen "ext_meani" hinzugefügt. Fehlende Werte werden hierbei nicht entfernt, damit die ursprüngliche Länge des Vektors beibehalten wird. So kann der neue Mittelwert an den ALLBUS-Datensatz angehängt werden.

```
allbus$ext_meani <- rowMeans(extremismus, na.rm = FALSE)
```

Der Wert, den die Befragten nun in der neuen Variable zugewiesen bekommen, zeigt, wie rechtextremistisch die einzelnen Befragten im Mittel auf die Fragen geantwortet haben.

11.3.2 Additiver Index

Die zweite Möglichkeit einen Index zu bilden, ist die additive Indexbildung. Dabei wird die Summe der Einzelvariablen gebildet. Auch hier zählt jeder Einzelindikator gleichermaßen. Für die additive Indexbildung wird die Funktion `rowSums()` genutzt. Auch hier wurden Befragte mit fehlenden Werten nicht von der Analyse ausgeschlossen. Dies ist in diesem Fall jedoch problematischer als beim Mittelwertindex. Falls ein Befragter auf ein einzelnes Item nicht geantwortet hat, wird beim Mittelwertindex der Konstruktmittelwert aus dem Mittelwert aller anderen Items berechnet. Bei zehn Items, also zum Beispiel aus den anderen neun Items. Somit ist das Konstrukt noch recht gut abgebildet.

Bei einem additiven Index hingegen fehlt dieser Wert in der Aufsummierung. Personen mit fehlenden Werten bekommen also einen deutlich geringeren Wert im Konstrukt zugewiesen. Dies sollte bei der Bildung von additiven Indizes immer beachtet werden.

```
allbus$ext_addi <- rowSums(extremismus, na.rm = FALSE)
```

Die Spannweite des additiven Indexes ist dabei nicht mehr zwischen den ursprünglichen Werten zwischen 1 und 5, sondern wird mit der Anzahl der Variablen gestreckt. Es ergeben sich dadurch unterschiedliche Mittelwerte und Minima und Maxima der beiden Indizes. Beide gebildeten Skalen sind metrisch und können für weitere Analysen des Konstrukts "rechtsextremistische Einstellungen" verwendet werden.

```
summary(allbus$ext_meani)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   1.000   1.700   2.100   2.204   2.600   5.000     582
summary(allbus$ext_addi)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   10.00   17.00   21.00   22.04   26.00   50.00     582
```

12 Lineare Regressionen

Dieses Kapitel führt in die Methodik der linearen Regressionen ein. Lineare Regressionen sind geeignet, um ein theoretisches Erklärungsmodell empirisch zu testen. Hierbei geht es darum, die Ausprägungen der abhängigen Variable (AV) auf eine oder mehrere unabhängige Variablen (UV) zurückzuführen (vgl. lateinisch "regredi", engl. "regress", franz. "régresser" etc.). Dabei liegen die kausalen Annahmen je nach Forschungsdesign und Datenbeschaffenheit oftmals nur theoretisch vor, statistisch handelt es sich in vielen Fällen streng genommen nur um Assoziationen oder Zusammenhänge, die in den Daten vorliegen und aufgedeckt können. Während Experimente geeignet sind, kausale Annahmen zu testen, können anhand von Umfragen Zusammenhänge in der Bevölkerung aufgezeigt werden. In solchen Fällen kontrollieren Regressionsmodelle kontrollieren typischerweise für Drittvariablen (confounding variables), um verzerrte Schätzungen durch die Auslassung theoretisch relevanter Variablen zu vermeiden (omitted variable bias). Vor einer jeden Analyse sollten die erwarteten theoretischen Zusammenhänge festgehalten werden. Zusätzlich lohnt es sich, die vermuteten Zusammenhänge zwischen den Variablen graphisch in einem Diagramm darzustellen.

12.1 Vorbereitung

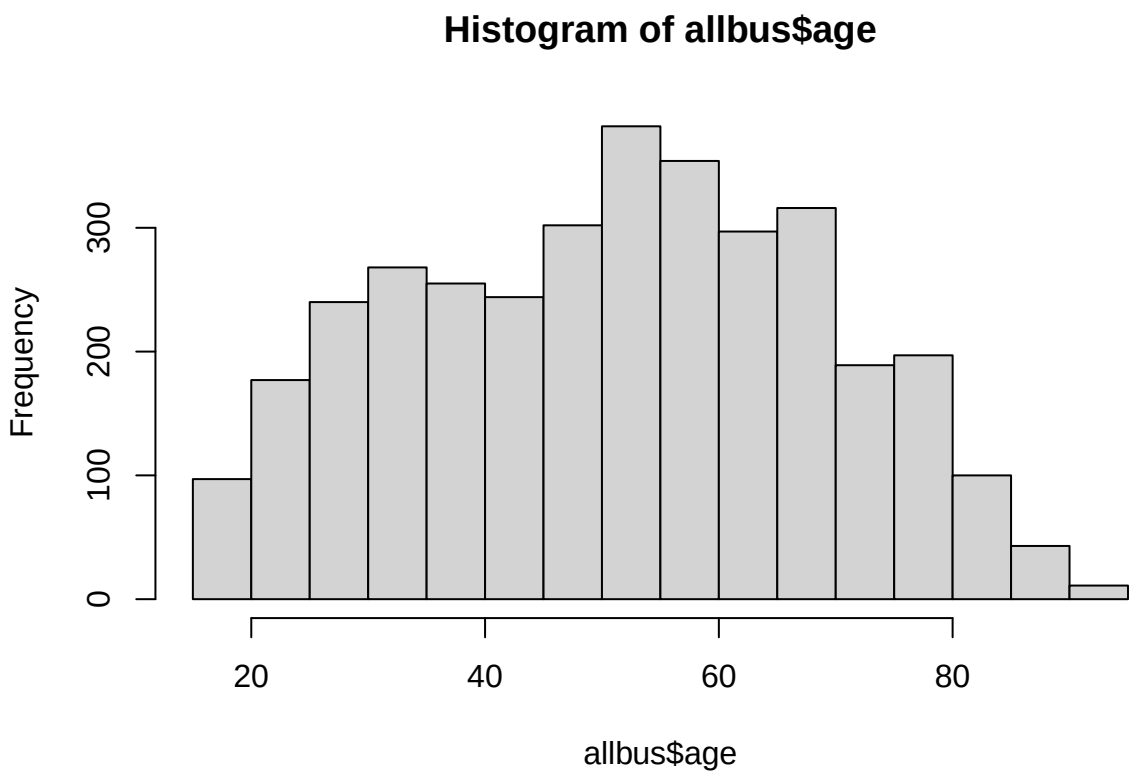
Um eine lineare Regression durchzuführen, werden erneut die Daten des ALLBUS-Datensatzes geladen und aufbereitet.

```
# Die notwendigen Pakete werden installiert, wenn noch nicht vorhanden
if (!require(haven)) install.packages("haven")

# das Paket zum Einlesen des Datensatzes wird in die Sitzung geladen
library(haven)

# Der Datensatz "ALLBUS2018" wird dem Objekt "allbus" zugewiesen
allbus <- as.data.frame(read_spss("ALLBUS2018.sav"))
names(allbus)
##  [1] "eastwest" "educ"     "sex"      "age"      "work"     "pt01"
##  [7] "pt02"     "pt03"     "pt04"     "pt08"     "pt09"     "pt10"
## [13] "pt11"     "pt12"     "pt14"     "pt15"     "pt19"     "pt20"
```

In diesem Beispiel soll zunächst der Einfluss des Alters (age) auf das Vertrauen in die Bundesregierung (pt12) untersucht werden. Beide Variablen werden mit `as.numeric()` als metrische Variablen gespeichert und die Verteilung anhand eines Histogrammes visualisiert und geprüft.

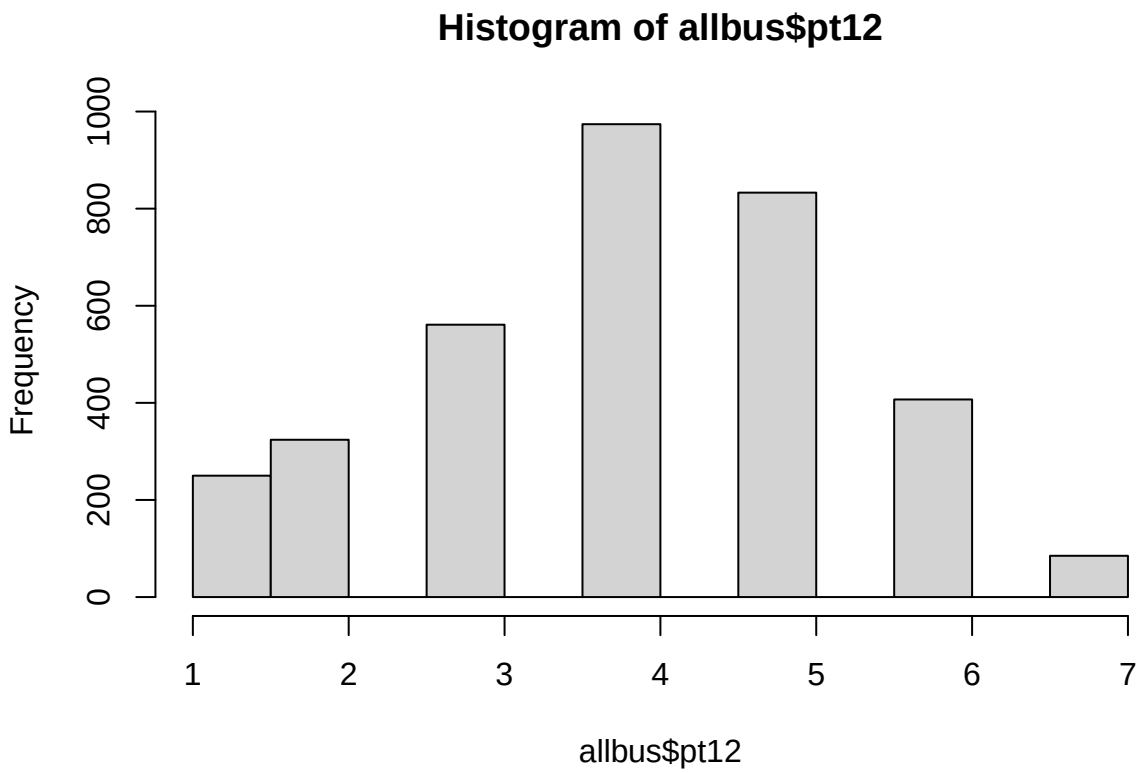
```
# Die Variablen (UV) zur Analyse werden im Datensatz gesucht,
# ggf. umbenannt oder erstellt

attributes(allbus$age) # Eigenschaften des Alters
## $label
## [1] "ALTER: BEFRAGTE(R)"
##
```

```
## $format.spss
## [1] "F3.0"
##
## $display_width
## [1] 10
##
## $class
## [1] "haven_labelled" "vctrs_vctr"     "double"
##
## $labels
## NICHT GENERIERBAR
##                -32
hist(allbus$age) # Verteilung des Alters
```




```
allbus$age <- as.numeric(allbus$age)

attributes(allbus$pt12) #Eigenschaften des pol. Vertrauens
## $label
## [1] "VERTRAUEN: BUNDESREGIERUNG"
##
## $format.spss
## [1] "F2.0"
```

```
##
## $class
## [1] "haven_labelled" "vctrs_vctr"     "double"
##
## $labels
##       KEINE ANGABE GAR KEIN VERTRAUEN                  ..                  ..
##                  -9                   1                   2                   3
##                  ..                  ..                  ..  GROSSES VERTRAUEN
##                   4                   5                   6                   7
hist(allbus$pt12) #Verteilung des pol. Vertrauens
```




```
allbus$pt12 <- as.numeric(allbus$pt12)
    #die ordinale Variable wird als numerische Variable gespeichert
str(allbus$pt12)
##  num [1:3477] 6 6 4 4 4 3 4 5 5 4 ...
```

In diesem Fall ist keine weitere Aufbereitung nötig und es kann eine lineare Regression mit dem Befehl `lm()` durchgeführt werden.

12.2 Die bivariate lineare Regression

Der Befehl `lm()` steht für "linear model" (lineares Modell). Hierbei wird die abhängige Variable auf die linke Seite der Gleichung und die unabhängige Variable hinter einem Tilde-Zeichen (~) auf die rechte Seite der Gleichung gestellt. Zusätzlich wird der Datensatz als Argument in der Funktion genannt. Das Regressionsmodell kann als Objekt zugewiesen und mit einem `summary()`-Befehl aufgerufen werden.

```
#Regression mit Alter
model1 <- lm(pt12 ~ age, data = allbus)
summary(model1)
##
## Call:
## lm(formula = pt12 ~ age, data = allbus)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.01796 -0.98122  0.02298  1.02088  3.05552
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.03686    0.07763  51.998   <2e-16 ***
## age         -0.00105    0.00142  -0.739     0.46
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.462 on 3427 degrees of freedom
##   (48 observations deleted due to missingness)
## Multiple R-squared:  0.0001595,  Adjusted R-squared:  -0.0001322
## F-statistic: 0.5467 on 1 and 3427 DF,  p-value: 0.4597
```

Der summary-Befehl zeigt die Regressionsgleichung, die Verteilung der Residuen, die Koeffizienten und Informationen zum Gesamtmodell (F-Test als Gütekriterium, R-Quadrat und korrigiertes R-Quadrat).

In diesem Fall hat entgegen der Annahme das Alter der Befragten keinen Einfluss auf das politische Vertrauen, d.h. zumindest keinen linearen Einfluss als metrische Variable. Ein Einfluss für bestimmte Alterskohorten wäre weiterhin denkbar. Dies wurde allerdings an dieser Stelle nicht überprüft, da es nicht über die metrische Variable getestet werden kann. Daher Als Nächstes wird das Erhebungsgebiet in den Fokus gerückt (eastwest). Aufgrund struktureller und historischer Gegebenheiten kann angenommen werden, dass sich weiterhin in Ost- und Westdeutschland Unterschiede hinsichtlich des politischen Vertrauens in die Regierung finden.

Neben metrischen Variablen kann die lineare Regression auch kategorielle Variablen in die Regression als sogenannte Dummy-Variablen als unabhängige Variablen aufnehmen. Hierbei ist es wichtig, dass aus den Kategorien (k) k-1 Dummy-Variablen in die Regression aufgenommen und eine Kategorie als Referenz ausgelassen wird. Alle Ergebnisse werden im Vergleich zur ausgelassenen Kategorie interpretiert. Im Folgenden wird aus der Variable "eastwest" mittels `as.factor` ein Faktor gebildet ("east"). Wenn Faktorvariablen in die Regression aufgenommen werden, werden in

R automatisch entsprechende Dummy-Variablen für den Faktor aufgenommen, sofern die Variable als Faktor gespeichert ist. Es ist möglich die Referenzkategorie für einen Faktor zu ändern und damit festzulegen, welche Dummy-Variablen in die Regressionanalyse aufgenommen werden.

```
#Kodierung
attributes(allbus$eastwest)
## $label
## [1] "ERHEBUNGSGEBIET (WOHNGEBIET): WEST - OST"
##
## $format.spss
## [1] "F1.0"
##
## $display_width
## [1] 10
##
## $class
## [1] "haven_labelled" "vctrs_vctr"     "double"
##
## $labels
## ALTE BUNDESLAENDER NEUE BUNDESLAENDER
##                  1                  2
allbus$east <- as.factor(allbus$eastwest)
table(allbus$east)
##
##    1    2
## 2387 1090
table(allbus$east, allbus$eastwest)
##
##        1    2
##   1 2387    0
##   2    0 1090
```

Durch diese Rekodierung kann die neue Variable “ost” in ein Regressionsmodell als unabhängige Variable aufgenommen werden.

```
#Regression mit East/West
model2 <- lm(pt12 ~ east, data = allbus)
summary(model2)
##
## Call:
## lm(formula = pt12 ~ east, data = allbus)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0735 -1.0735 -0.0735  0.9265  3.2132
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)  4.07346    0.02999 135.809  < 2e-16 ***
## east2       -0.28662    0.05351  -5.357 9.04e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.456 on 3432 degrees of freedom
##   (43 observations deleted due to missingness)
## Multiple R-squared:  0.008291,   Adjusted R-squared:  0.008002
## F-statistic: 28.69 on 1 and 3432 DF,  p-value: 9.042e-08
```

Model 2 zeigt nun den Einfluss des Erhebungsgebiets. Der Koeffizient für die Variable "east" ist hochsignifikant. Dies lässt sich wie folgt interpretieren: Befragte in Ostdeutschland besitzen durchschnittlich ein geringeres Vertrauen in die Regierung als Befragte in Westdeutschland und zwar ist das Vertrauen im Durschnitt -0,29 Punkte geringer. Die Interpretation dieser Dummy-Variablen kann nur im Vergleich zur Referenzkategorie (der ausgelassenen Kategorie) erfolgen und daher muss die ausgelassene Kategorie stets in der Interpretation genannt werden. Der folgende Code-Chunk zeigt wie die Referenzkategorie für die Faktorvariable geändert wird. Das Ergebnis kann erneut interpretiert werden, sofern der Einfluss im Vergleich zur Referenzkategorie benannt wird. Wie unterscheidet sich das Ergebnis der Regression? Was hat sich geändert? Was ist gleich geblieben?

```
# Wechsel der Referenzkategorie und Speicherung unter neuem Namen
allbus$west <- relevel(allbus$east, ref= "2")
#Regression mit East/West und geänderter Referenzkategorie
model2 <- lm(pt12 ~ west, data = allbus)
summary(model2)
##
## Call:
## lm(formula = pt12 ~ west, data = allbus)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0735 -1.0735 -0.0735  0.9265  3.2132
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.78684    0.04431  85.459  < 2e-16 ***
## west1        0.28662    0.05351   5.357 9.04e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.456 on 3432 degrees of freedom
##   (43 observations deleted due to missingness)
## Multiple R-squared:  0.008291,   Adjusted R-squared:  0.008002
## F-statistic: 28.69 on 1 and 3432 DF,  p-value: 9.042e-08
```

12.3 Die multivariate lineare Regression

In Modell 1 und Modell 2 handelt es sich jeweils um bivariate Regressionen, da nur eine unabhängige Variable in das Modell aufgenommen wurde. Nun wird eine multiple Regression mit mehreren unabhängigen Variablen geschätzt. Modell 3 benutzt die Variablen "age" und "east" zur Erklärung des Vertrauens in die Regierung.

```
# Multiple Regression
model3 <- lm(pt12 ~ age + east, data = allbus)
summary(model3)
##
## Call:
## lm(formula = pt12 ~ age + east, data = allbus)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0865 -1.0676 -0.0630  0.9341  3.2285
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.0939972  0.0780717  52.439  < 2e-16 ***
## age         -0.0004187  0.0014190  -0.295    0.768
## east2       -0.2856940  0.0537482  -5.315 1.13e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.456 on 3426 degrees of freedom
##   (48 observations deleted due to missingness)
## Multiple R-squared:  0.008338,   Adjusted R-squared:  0.007759
## F-statistic:  14.4 on 2 and 3426 DF,  p-value: 5.906e-07
```

In Modell 3 hat das Erhebungsgebiet weiterhin einen signifikanten Einfluss. Der F-Test zeigt, dass das Gesamtmodell signifikant ist. Dies ist in der Regel der Fall, sobald eine unabhängige Variable im Modell signifikant ist. Der F-Test sagt aus, dass die Vorhersage des politischen Vertrauens durch das Modell verbessert wird. Die Vorhersage des Modells in diesem Falle zur Erklärung des Vertrauens wird dabei mit der bestmöglichen Schätzung ohne Einbeziehung der unabhängigen Variablen verglichen, d.h. einer Vorhersage über den Mittelwert der abhängigen Variable (Field, 2009).

Für multiple Regressionen sollte das korrigierte R-Quadrat herangezogen werden, da dies für die Aufnahme zusätzlicher Prädiktoren korrigiert. In diesem Falle liegt das korrigierte R-Quadrat bei 0,008. Wenn dieser Wert mit 100 multipliziert wird, kann ausgesagt werden, dass lediglich 0,8 Prozent der Varianz des politischen Vertrauens durch das Modell erklärt werden kann. Dies ist ein sehr geringer Wert. Insofern zeigt es, dass signifikante Einflüsse nicht immer mit einer substanziell starken Erklärungskraft einhergehen.

Die Modellannahmen sind stets zu überprüfen und sind daher an dieser Stelle kurz genannt. Ein ausführlicher Überblick findet sich beispielsweise in Field (2009).

R bietet eine visuelle Inspektion der Residuen, die über die Plot-Funktion `plot()` und das jeweils geschätzte Model als Objekt aufgerufen werden kann, z.B. `plot(model3)`. Dies ist ein einfacher und schneller Weg, zentrale Annahmen der lineraren Regression grafisch zu überprüfen (z.B. Normalverteilung der Residuen über einen Q-Q-Plot, Homoskedastizität der Residuen sowie eine Übersicht möglicher einflussreicher Ausreißer auf die Regressionsgerade (leverage)).

12.4 Weitere Formen der Regression

In der statistischen Datenanalyse können neben linearen Regressionen über die Methode der kleinsten Quadrate (engl. ordinary least squares, kurz OLS) eine Vielzahl weiterer Schätzverfahren angewendet werden. Die Angemessenheit der Modelle wird durch die Beschaffenheit der vorliegenden Daten bestimmt. Für einen vertiefenden Einstieg sollten statistische Lehrbücher herangezogen und der Besuch weiterführender Veranstaltungen in Betracht gezogen werden. An dieser Stelle sei nur kurz auf die Durchführung in R hingewiesen. Der Befehl `glm` steht für "generalized linear model" und ermöglicht die Schätzung einer Regression mit verschiedenen Schätzverfahren. So kann neben anderen Verfahren ebenso eine lineare Regression (OLS) durchgeführt werden. Dabei wird über das Argument `family` das Schätzverfahren festgelegt, , hier "gaussian". Der Befehl für eine lineare Regression lautet demnach wie folgt:

```
# Multiple Regression mit dem Befehl "glm"
model4 <- glm(pt12 ~ age + east, data = allbus, family = gaussian)
summary(model4)
##
## Call:
## glm(formula = pt12 ~ age + east, family = gaussian, data = allbus)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0865  -1.0676  -0.0630   0.9341   3.2285
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.0939972  0.0780717  52.439  < 2e-16 ***
## age         -0.0004187  0.0014190  -0.295    0.768
## east2       -0.2856940  0.0537482  -5.315 1.13e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 2.120221)
##
##     Null deviance: 7325.0  on 3428  degrees of freedom
## Residual deviance: 7263.9  on 3426  degrees of freedom
##   (48 observations deleted due to missingness)
## AIC: 12313
##
## Number of Fisher Scoring iterations: 2
```

Um eine binär logistische Regression zu schätzen, kann ebenfalls der Befehl `glm` genutzt werden. Logistische Regressionen finden häufig in den Sozialwissenschaften Anwendung, wenn die abhängige Variable zwei Ausprägungen besitzt, d.h. dichotom skaliert ist. In der Regel geht es darum, das Eintreten eines Ereignisses (1) im Vergleich zum Nicht-Eintreten (0) zu erklären. Gängige Beispiele sind die Wahlabsicht für eine bestimmte Partei (1) vs. die Nicht-Wahlabsicht dieser Partei (0), die Wahlbeteilung, Teilnahme an Demonstrationen uvm. Im nächsten Code-Chunk zeigt sich ein Befehl zur Ausführung einer logistischen Regression. Dazu wird das Vertrauen in die Bundesregierung (pt12) dichotomisiert und zwar in kein/mäßiges Vertrauen (0) und großes Vertrauen (1). Die Variable wurde auf einer 7er-Skala gemessen. Die Werte 5 bis 7 werden für die Ausprägung "großes Vertrauen" zusammengefasst; die Werte 1-4 bilden die Kategorie "kein / mäßiges Vertrauen". Zur Schätzung einer logistischen Regression wird das Argument `family = binomial` innerhalb des Regressionsbefehls genutzt.

```
allbus$pt12_dummy <- ifelse(allbus$pt12 >= 5, 1, 0)
  # dichotomes Vertrauensvariable wird erstellt

model5 <-  glm(pt12_dummy ~ age + east, data = allbus, family = binomial)
summary(model5)
##
## Call:
## glm(formula = pt12_dummy ~ age + east, family = binomial, data = allbus)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0348  -1.0237  -0.9031   1.3374   1.4924
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.3320127  0.1100741  -3.016  0.00256 **
## age         -0.0007303  0.0020055  -0.364  0.71576
## east2       -0.3148088  0.0772316  -4.076 4.58e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4572.3  on 3428  degrees of freedom
## Residual deviance: 4554.9  on 3426  degrees of freedom
##   (48 observations deleted due to missingness)
## AIC: 4560.9
##
## Number of Fisher Scoring iterations: 4
```

Da die Koeffizienten der logistischen Regression nur schwerlich interpretiert werden können (abgesehen von ihrer Richtung und Signifikanz), werden häufig Odds Ratios berichtet. Diese sind allerdings in ihrer Interpretation umstritten (Best & Wolf, 2010). Die Odds Ratios und ihre Konfidenzintervalle für die Koeffizienten der obigen logistischen Regression werden wie folgt ausgegeben:

```
exp(cbind(Odds_Ratios = coef(model5), confint(model5)))
## Waiting for profiling to be done...
##             Odds_Ratios     2.5 %    97.5 %
## (Intercept)   0.7174782 0.5779945 0.8899318
## age           0.9992700 0.9953486 1.0032060
## east2         0.7299284 0.6270267 0.8487748
```

Aus den Koeffizienten können ebenfalls die Wahrscheinlichkeiten für das Eintreten des Ereignisses (1) berechnet werden. Allgemein ist es zu empfehlen, Wahrscheinlichkeiten anstelle der Odds Ratios zu berechnen. Dies lässt sich leicht in R durchführen, erfordert allerdings ein gewisses Grundverständnis der logistischen Regression sowie Kenntnisse in R. Daher wird auf eine ausführliche Beschreibung in diesem Einführungsskript verzichtet.

13 Gewichtung

In den Sozialwissenschaften wird – wie bereits erläutert – oftmals mit Befragungsdaten gearbeitet. Dabei wird zunächst eine Grundgesamtheit definiert aus der dann eine Stichprobe gezogen wird. Die Stichprobe sollte mit Bedacht gezogen werden und die Grundgesamtheit möglichst genau repräsentieren. Je nach Ansatz der Studie geschieht die Sitchprobenziehung (engl. Sampling) nach einem unterschiedlichen Vorgehen.

Die meisten Befragungen repräsentieren – ob intendiert oder nicht – bestimmte Befragtengruppen über und andere unter. Um diese Ungleichverteilungen zu berücksichtigen, wird in der empirischen Praxis mit Gewichten gearbeitet. Diese Gewichte werden als zusätzliche Variablen im Datensatz aufgeführt. Werden die Gewichte in die Analyse einbezogen, erhalten Personen mit unterrepräsentierten Eingeschaften ein höheres Gewicht in den Analysen als Personen mit überrepräsentierten Eigenschaften und Personen mit überrepräsentierten Eigenschaften erhalten ein niedrigeres Gewicht in den Analysen.

Ein Beispiel: Im ALLBUS-Datensatz wurde die ostdeutsche Bevölkerung höher repräsentiert als die westdeutsche Bevölkerung. Diese Auswahlmethode wurde absichtlich gewählt und stellt ein sogenanntes Designgewicht dar. Das Oversampling geschieht, damit genügend Fälle im Datensatz vorhanden sind, um komplexe Analysen für Ost- und Westdeutschland getrennt durchzuführen und so jeweils repräsentative Ergebnisse für die beiden Erhebungsgebiete zu erhalten. Um repräsentative Aussagen über die gesamte Bevölkerung zu treffen, muss daher dieses Überrepräsentierung der Personen aus ostdeutschen Bundesländern ausgeglichen werden. Im Variablenreport des ALLBUS heißt es: "Befragte in Ostdeutschland werden seit 1991 zu einem größeren Anteil in die Stichprobe einbezogen als es ihrem Anteil an der Grundgesamtheit entspräche (Oversampling). Dieses Oversampling soll auch für kleinere Bevölkerungsgruppen in Ostdeutschland noch statistisch vertretbare Analysen ermöglichen." (GESIS-Leibniz-Institut Für Sozialwissenschaften, 2019, S. iii)

13.1 Vorbereitung

Zunächst wird erneut der Datensatz eingelesen.

```
library(haven)
allbus <- as.data.frame(read_spss("ALLBUS2018komplett.sav"))
```

Es gibt verschiedene Möglichkeiten in R Gewichtungen zu berücksichtigen. Eine Möglichkeit bietet das Paket "survey". Mit dem Paket "survey" können in R auch komplexe Auswahlverfahren berücksichtigt werden und eine Vielzahl an statistische Verfahren durchgeführt werden. Das Paket wird mit dem folgenden Code geladen.

```
# install.packages("survey")
library(survey)
```

Die Gewichtungsvariable im ALLBUS-Datensatz ist unter `wghtpew` abgespeichert. Näheres zur Variable findet sich im Variablenreport (GESIS-Leibniz-Institut Für Sozialwissenschaften, 2019, S. 800). In diesem Fall handelt es sich um ein Designgewicht. Zunächst werden sich die Eigenschaften der Gewichtungsvariable angeschaut.

```
attributes(allbus$wghtpew)
## $label
## [1] "PERSONENBEZOGENES OST-WEST-GEWICHT"
##
## $format.spss
## [1] "F14.12"
##
## $display_width
## [1] 14
##
## $class
## [1] "haven_labelled" "vctrs_vctr"     "double"
##
## $labels
## NEUE BUNDESLAENDER ALTE BUNDESLAENDER
##          0.5448076          1.2078591
summary(allbus$wghtpew)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.5448  0.5448  1.2079  1.0000  1.2079  1.2079
table(allbus$wghtpew)
##
## 0.544807595224425  1.20785912073958
##              1090              2387
```

13.2 Erstellung des Umfrage-Designs

Zunächst wird das Umfrage-Designs des Datensatzes `allbus` festgelegt und in einem neuen Objekt `allbus.w` gespeichert. Die Funktion `svydesign()` wird herangezogen, um das Design festzulegen. Die Funktion besitzt verschiedene Argumente von denen mindestens drei Argumente festgelegt werden müssen (`ids`, `data`, `weights`). Das Argument `ids` kann für Erhebungscluster genutzt werden, das Argument `data` legt den Datensatz fest, für den die Gewichtung vorgenommen werden soll. Über das Argument `weights` der Funktion `svydesign()` wird festgelegt, welche Variable eines Datensatzes zur Gewichtung herangezogen werden soll.

```
allbus.w <- svydesign(ids =~ 1, data = allbus, weights =~ wghtpew)
```

13.3 Berücksichtigung des Umfrage-Designs

Nachdem ein Objekt erstellt wurde, das die Informationen zum Survey-Design enthält, kann diese Information in einer Vielzahl von Funktionen des Pakets `survey` genutzt werden, um auf diese Weise gewichtete Ergebnisse zu erhalten. Die folgenden Befehle beziehen sich demnach nicht auf den ursprünglichen ALLBUS-Datensatz, sondern auf das neu erstellte Objekt, welches das Umfrage-Design berücksichtigt.

Als Beispiel wird nun eine deskritpive Auswertung ohne und mit Gewichtung gegenübergestellt. Die Funktion `svytable` ermöglicht dabei die Berücksichtigung der Gewichte für Häufigkeiten und

Kreuztabellen. Als Testvariable wird hier der Bildungsabschluss der Befragten gewählt. Zunächst wird eine Verteilungstabelle für beide Datensätze (ohne und mit Gewichtung) angelegt. Die Ausprägungen der Variablen werden für die Tabelle dabei noch mittels `rownames()` beschriftet. Nachdem beide Tabelle erstellt wurden, werden diese mit der Funktion `cbind()` zusammengeführt, die jeweiligen Spalten benannt und die Tabelle ausgegeben (siehe detailliertes Vorgehen dazu in Kapitel 4).

```
educ.table <- prop.table(table(allbus$educ))
rownames(educ.table) <- c("Ohne Schulabschluss", "Volks-, Hauptschule",
                          "Mittlere Reife", "Fachhochschulreife",
                          "Hochschulreife",
                          "Anderer Abschluss", "Noch Schüler")

educ.table.w <- prop.table(svytable(~educ, design = allbus.w))
rownames(educ.table.w) <- c("Ohne Schulabschluss", "Volks-, Hauptschule",
                            "Mittlere Reife", "Fachhochschulreife",
                            "Hochschulreife",
                            "Anderer Abschluss", "Noch Schüler")

vergleich <- cbind(educ.table, educ.table.w)
colnames(vergleich) <- c("ohne Gewichtung", "mit Gewichtung")
vergleich
##                      ohne Gewichtung mit Gewichtung
## Ohne Schulabschluss     0.014392631    0.015478438
## Volks-, Hauptschule     0.233160622    0.245596516
## Mittlere Reife          0.343983880    0.315147687
## Fachhochschulreife      0.086931491    0.093566122
## Hochschulreife          0.309441566    0.317132990
## Anderer Abschluss       0.005757052    0.006191375
## Noch Schüler            0.006332758    0.006886871
round(vergleich, 4)*100
##                      ohne Gewichtung mit Gewichtung
## Ohne Schulabschluss            1.44           1.55
## Volks-, Hauptschule           23.32          24.56
## Mittlere Reife                34.40          31.51
## Fachhochschulreife             8.69           9.36
## Hochschulreife                30.94          31.71
## Anderer Abschluss              0.58           0.62
## Noch Schüler                   0.63           0.69
```

Die Ergebnisse zeigen, dass sich die deskriptiven Verteilungen leicht zwischen der Variante ohne und mit Gewichtung unterscheiden; dies ist besonders auffällig beim Bildungsabschluss "mittlere Reife".

Gewichtete deskriptive Statistiken können ebenfalls mit dem `survey`-Paket berechnet werden. Um beispielsweise die gewichtete Mittelwerte zu erhalten, kann die Funktion `svymean()` benutzt werden. Im Folgenden wird wiederum der ungewichtete und gewichtete Mittelwert der Variable "Vertrauen in das europäische Parlament" `pt20` berechnet.

```r
mean(allbus$pt20, na.rm = TRUE)
## [1] 3.595031
svymean(~ pt20, allbus.w, na.rm = TRUE)
##        mean     SE
## pt20 3.6487 0.0264
```

Auch hier zeigen sich leichte Unterschiede zwischen ungewichteten und gewichteten Werten.

Das `survey`-Paket bietet eine Vielzahl an weiteren Möglichleiten um Gewichtungen in statistische Verfahren mit einzubeziehen. So können Gewichte auch in t-Tests (`svyttest`) oder linearen Regressionen (`svyglm`) berücksichtigt werden. Mehr dazu findet sich in der Dokumentation des `survey`-Pakets. Für Tidyverse-Nutzende gibt es außerdem das Package `srrvyr`, welches analog zum `survey` Package aufgebaut ist.

14 Weitere Literaturempfehlungen

Ziel des vorliegenden Skripts ist es, einen ersten Einstieg in die statistische Datenanalyse mit R für Studierende der Sozialwissenschaften zu bieten. In fortgeschrittenen Kursen sowie im Selbststudium können die Kenntnisse weiter vertieft werden. An dieser Stelle sei daher auf ein paar weiterführende Ressourcen verwiesen. Dabei handelt es sich lediglich um eine Auswahl zu einzelnen Themengebieten. Eine eigene Recherche wird empfohlen.

- Zur Datenanalyse mit dem `tidyverse`:
  R for Data Science
- Zum Programmieren in R:
  Offizielles Manual
  Hands-On Programming with R
- Zur Datenvisualiserung:
  R Graphics Cookbook
  Data Visualization with R
  Data Visualization. A Practical Introduction
  Fundamentals of Data Visualization
- Zur Durchführung statistischer Verfahren:
  Discovering Statistics Using R
  R für Einsteiger
  R in Action
  UCLA. Institute for Digital Research & Education. Statistical Consulting
- Zur quantitativen Textanalyse:
  Quanteda Tutorials
  Text Mining with R. A Tidy Approach
- Zum schnellen Überblick (Cheat Sheets):
  ggplot2
  dplyr
  quanteda
  und viele mehr
- Zur Behebung von Fehlermeldungen:
  Google
  stackoverflow
  stats.stackexchange

Autoreninformation

Lena Masch ist wissenschaftliche Mitarbeiterin an der Humboldt-Universität zu Berlin. Zuvor war sie als wissenschaftliche Mitarbeiterin in der Politikwissenschaft und Soziologie an der Heinrich-Heine-Universität Düsseldorf tätig. In Forschung und Lehre liegen ihre Interessen in der politischen Psychologie und empirischen Sozialforschung.

Kimon Kieslich ist wissenschaftlicher Mitarbeiter und Doktorand am Lehrstuhl Kommunikations- und Medienwissenschaft I der Heinrich-Heine-Universtität Düsseldorf. Seinen Bachelor- und Masterabschluss erlangte er im Fach Kommunikationswissenschaft an der Westfälischen Wilhelms-Universtität Münster. Kimon Kieslich Forschungsinteressen liegen in der öffentlichen Wahrnehmung und Bewertung von Künstlicher Intelligenz, der medialen Darstellung und Erzeugung von Angst sowie den Effekten populistischer Kommunikation.

Katharina Huseljić ist wissenschaftliche Mitarbeiterin und Doktorandin am Lehrstuhl Soziologie V. Sie studierte im Bachelor Soziologie, Politikwissenschaft und Kommunikations- und Medienwissenschaften an der Heinrich Heine Universität in Düsseldorf, sowie im Master Soziologie und empirische Sozialforschung an der Universität zu Köln. Ihr Forschungsschwerpunkte liegen in den Bereichen Stadtsoziologie, politische Partizipation sowie Legitimitätswahrnehmungen und der Stabilität von Demokratie.

Marco Wähner ist wissenschaftlicher Mitarbeiter am Lehrstuhl für Soziologie II an der Heinrich-Heine-Universität Düsseldorf und Kollegiat im NRW Forschungskolleg Online-Partizipation. Sein Forschungsinteresse liegt in der Untersuchung von politischer Beteiligung im Internet und Methoden der empirischen Sozialforschung. Seit 2019 hat er außerdem eine Lehrtätigkeit in der Datenanalyse mit SPSS oder R an der HHU.

Sebastian Neef studierte Sozialwissenschaften an der Heinrich-Heine-Universität Düsseldorf und lehrt dort seit einigen Jahren den Umgang mit Statistikprogrammen. Aktuell arbeitet er in einer Düsseldorfer Digitalagentur als Data Analyst und beschäftigt sich mit Marketing-Daten von Unternehmen.

Quellenverzeichnis

Best, H. & Wolf, C. (2010). Logistische Regression. In C. Wolf & H. Best (Hrsg.), *Handbuch der sozialwissenschaftlichen Datenanalyse* (Band 80, S. 827–854). Wiesbaden: VS Verlag für Sozialwissenschaften. https://doi.org/10.1007/978-3-531-92038-231

Diekmann, A. (2007). *Empirische Sozialforschung: Grundlagen, Methoden, Anwendungen* (rororo Rowohlts Enzyklopädie) (18. Aufl., vollst. überarb. und erw. Neuausg., [1. Aufl. der Neuausg.]., Band 55678). Reinbek bei Hamburg: Rowohlt-Taschenbuch-Verl.

Döring, N. & Bortz, J. (2016). *Forschungsmethoden und Evaluation in den Sozial- und Humanwissenschaften*. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-41089-5

Field, A. (2009). *Discovering Statistics Using SPSS* (ISM (London, England)). SAGE Publications.

Field, A., Miles, J. & Field, Z. (2012). *Discovering statistics using R* (Repr.). London: SAGE Publ.

GESIS-Leibniz-Institut Für Sozialwissenschaften. (2019). Allgemeine Bevölkerungsumfrage der Sozialwissenschaften ALLBUS 2018. GESIS Data Archive. https://doi.org/10.4232/1.13250

Joselson, N. (2016). Eugenics and Statistics, Discussing Karl Pearson and R. A. Fisher. Blog Post. Verfügbar unter: https://njoselson.github.io/Fisher-Pearson/

Krebs, D. & Menold, N. (2019). Gütekriterien quantitativer Sozialforschung. In N. Baur & J. Blasius (Hrsg.), *Handbuch Methoden der empirischen Sozialforschung* (S. 489–504). Wiesbaden: Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-21308-434

Latcheva, R. & Davidov, E. (2019). Skalen und Indizes. In N. Baur & J. Blasius (Hrsg.), *Handbuch Methoden der empirischen Sozialforschung* (Band 25, S. 893–905). Wiesbaden: Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-21308-462

Luhmann, M. (2015). *R für Einsteiger: Einführung in die Statistiksoftware für die Sozialwissenschaften* (4., überarb. Aufl.). Weinheim [u.a.]: Beltz.

Schnell, R., Hill, P. B. & Esser, E. (2018). *Methoden der empirischen Sozialforschung* (De Gruyter Studium) (11., überarbeitete Auflage.). Berlin; Boston: De Gruyter Oldenbourg.

Wickham, H. & Grolemund, G. (2016). *R for data science: Import, tidy, transform, visualize, and model data* (First edition.). Beijing; Boston; Farnham; Sebastopol; Tokyo: O'Reilly.