Statistische Software (R) - Hausarbeit 2

Renate Schmeidler

21.06.2021

Aufgabe 2

Bei dem angegebenen Code liegt eine Endlos-Schleife vor:

- Der Wert von i wird zu Beginn auf 1 gesetzt und bei jedem Schleifendurchlauf um 1 erhöht.
- Damit ist die Bedingung "i > 0" stets erfüllt, und die Schleife bricht nicht ab, sondern wird theoretisch (d. h. solange kein Eingriff von außen o.ä. erfolgt) unendlich oft ausgeführt.

Aufgabe 3

a) Funktion zur Simulation des 4-fachen Münzwurfs

Gesucht ist eine Funktion, die das viermalige einer Münze mit R simuliert und als Ergebniswert zurückgibt, wie oft dabei "KOPF" erschienen ist.

```
set.seed(20210614)
outcomes <- c("KOPF", "ZAHL")

# This function simulates tossing a coin four times
# and counts the number of Heads ("KOPF") that appear.
# Necessary for using this function:
# outcomes <- c("KOPF", "ZAHL") must be defined.

# Arguments: No input arguments.
# Returns:   A numeric vector of length 1 indicating
#            the number of "KOPF" that appear during
#            simulating tossing a coin 4 times.

count_head_tossing_coin_four_times <- function() {

  number_head <- 0
  for (i in 1:4) {
    coin <- sample(outcomes, 1, prob = c(.5, .5))
    if (coin == "KOPF") {
      number_head <- number_head + 1
    }
  }
  return(number_head)
}
```

b) n-malige Wiederholung und Schätzung einer Wahrscheinlichkeit

Gesucht ist eine Funktion, die die Funktion aus a) n-mal ausführt.

```r
# This function executes n simulations of 'tossing a coin 4 times'
# and counts the number of "KOPF" for each simulation.
#
# Dependencies:
# Function count_head_tossing_coin_four_times() is required.
#
# Arguments:
# n:       A numeric vector of length 1 indicating the number
#          of repetitions of simulating 'tossing a coin 4 times'
# Returns: A numeric vector of length n indicating the numbers
#          of "KOPF" in each of those n simulations.

n_repeat_counting_head <- function(n) {

  if (!is.numeric(n))
    stop("Anzahl der Wiederholungen muss natürliche Zahl sein.")
  if (n < 0)
    stop("Negative Anzahl an Wiederholungen nicht sinnvoll.")

  outcome <- vector(mode = "double", length = n)
  for (i in 1:n) {
   outcome[i] <- count_head_tossing_coin_four_times()
  }

  return(outcome)
}
```

Diese Funktion soll nun mit n = 10000 aufgerufen werden, das heißt, die Simulation des vierfachen Münzwurf soll 10000 mal durchgeführt werden.
Das Ergebnis soll in der Variablen outcome gespeichert werden.

```r
n_times <- 10000
outcome <- n_repeat_counting_head(n_times)
```

Daraus soll ein Schätzwert für die Wahrscheinlichkeit des Ereignisses “Viermal ‘KOPF’ beim vierfachen Münzwurf” berechnet werden.

```r
four_head <- which(outcome == 4)
number_four_head <- length(four_head)

probability_four_head <- number_four_head / n_times
probability_four_head
```

```
## [1] 0.0644
```

Antwort: Aufgrund der von R durchgeführten Simulation kann die Wahrscheinlichkeit, beim viermaligen Werfen einer idealen Münze viermal “KOPF” zu erhalten, auf 0.0644 geschätzt werden.

c) Schätzung von zu erwartendem Gewinn bzw. Verlust

```r
gain_loss <- probability_four_head * (-50) + (1 - probability_four_head) * 1
gain_loss
```

```
## [1] -2.2844
```

```r
gain_loss_euro <- round(gain_loss, digits = 2)
gain_loss_euro
```

```
## [1] -2.28
```

Antwort: Bei der Durchführung der Wette ist im Mittel ein Gewinn von -2.28€ zu erwarten, wobei ein negatives Vorzeichen anzeigt, dass man in Wirklichkeit einen Verlust zu erwarten hat.

d) Funktion zur Simulation bis zum ersten Mal '4 mal KOPF' erscheint

```r
# This function executes repeated simulations of 'tossing a coin four times'
# until we get 'KOPF KOPF KOPF KOPF' the first time, and
# returns the number of executions needed for that.
#
# Dependencies:
# Function count_head_tossing_coin_four_times() is required.
#
# Arguments: No input arguments.
# Returns:   A numeric vector of length 1 indicating the number of
#            simulations of 'tossing a coin 4 times' that were needed
#            until there was 'KOPF KOPF KOPF KOPF' the first time.
#

n_until_four_head <- function() {

  k <- 0
  number_head <- -1

  while (number_head < 4) {
    k <- k + 1
    number_head <- count_head_tossing_coin_four_times()
  }

  return(k)
}
```

e) Zahl benötigter Versuche, bis erstmals '4 mal KOPF' erscheint

```r
first_four_head <- vector(mode = "numeric", length = 100)
for (i in seq_along(first_four_head)) {
  first_four_head[i] <- n_until_four_head()
}
```

Anzeigen der erhaltenen Anzahlen (nicht verlangt in Aufgabenstellung)

```r
first_four_head
```

```
##   [1]  3 38 10 22  1 60 12  2 24 30 12  1  4  3  5 18  2  4  7  1 13  6  9  4 27
##  [26]  2 22 35  5 24 31 15 20  5  4 27  6 18  6 27 26  2  6  4  4  5 27 32  4 48
##  [51]  3 15  5 19 27 11  5  4 47 21 10  4 14  6 45  9 27  4 31  2  4 39 14  1 12
##  [76]  8  3 16  7  7 31 36 26 18  1 12 15  1 14 10  9  5  6 11  5  7 36  2  2 14
```

Durchschnittliche Anzahl benötigter Versuche

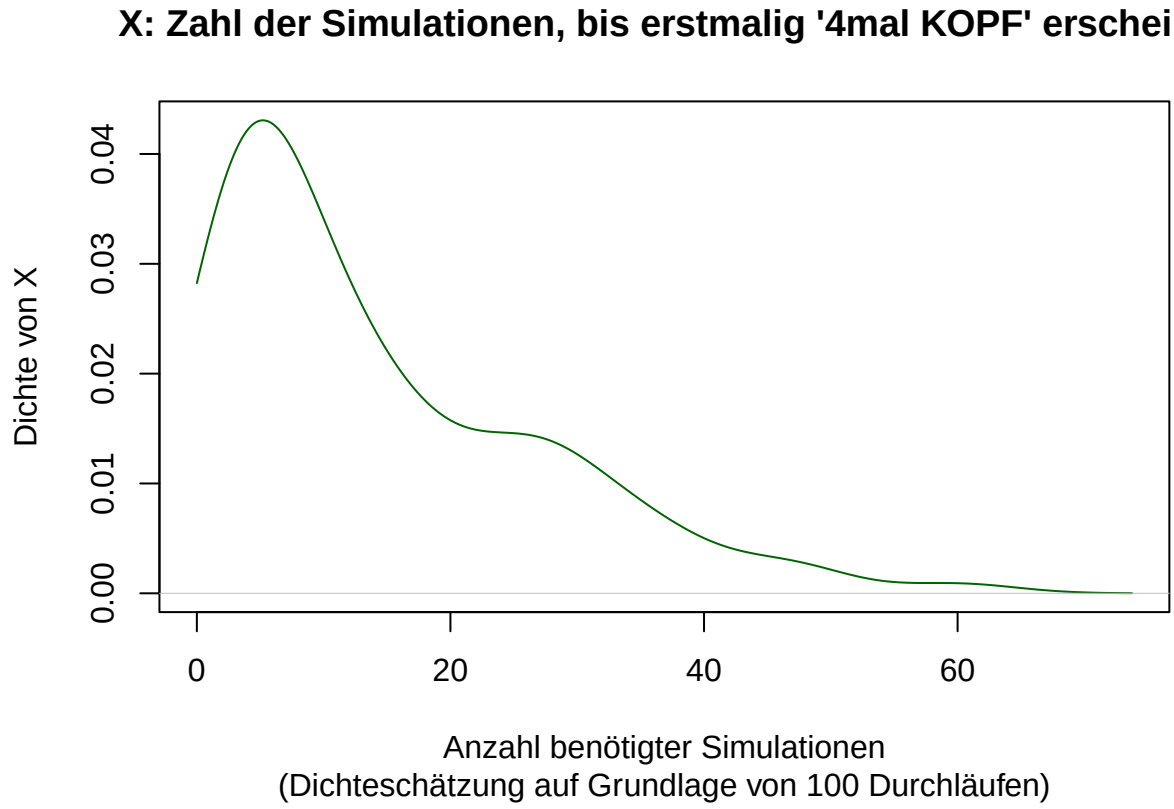
```r
mean(first_four_head)
```

```
## [1] 14.04
```

Antwort: Im Mittel wurden 14.04 Versuche benötigt, bis zum ersten Mal “4 mal KOPF” erschien.

f) Graphische Darstellung der Dichte der Verteilung

```
plot(density(first_four_head, from = 0),
     main = "X: Zahl der Simulationen, bis erstmalig '4mal KOPF' erscheint",
     sub = paste("(Dichteschätzung auf Grundlage von", length(first_four_head), "Durchläufen)"),
     xlab = "Anzahl benötigter Simulationen",
     ylab = "Dichte von X", col = "darkgreen")
```



Aufgabe 4

In dieser Aufgabe sollen Daten aus dem Datensatz vaccs.Rds visualisiert werden.

Vorbereitende Schritte für die gesamte Aufgabe 4:
Laden des ggplot2-Pakets und Einlesen der Daten.

```
library(ggplot2)
vaccs <- readRDS("vaccs.Rds")
```

a)

Es soll ein Liniendiagramm erstellt werden, das den Fortschritt der Covid-Impfungen in Bayern, wie er sich aus den Daten ergibt, darstellt.

```
# Erstellen eines Unterdatensatzes
# mit den Daten für Bayern:
vaccs_bayern_graph <- subset(vaccs, region == "DE-BY")
```

Vorbereitende Schritte: Im Diagramm soll für jeden der Impfstoffe jeweils die Menge der verabreichten Dosen gegen die Zeit aufgetragen werden.
Dabei erscheinen in der Beschriftung des Diagramms die Bezeichnungen der vorkommenden Variablen.

```
# Überprüfung der Variablenbezeichungen
str(vaccs_bayern_graph)
```

```
## 'data.frame':    236 obs. of  4 variables:
##  $ date     : Date, format: "2020-12-26" "2020-12-28" ...
##  $ impfstoff: Factor w/ 4 levels "astra","comirnaty",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ region   : Factor w/ 17 levels "DE-BB","DE-BE",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ dosen    : num  0 0 0 0 0 0 0 0 0 52800 ...
```

```
levels(vaccs_bayern_graph$impfstoff)
```

```
## [1] "astra"     "comirnaty" "johnson"   "moderna"
```

```
table(vaccs_bayern_graph$region)
```

```
##
##   DE-BB   DE-BE DE-BUND   DE-BW   DE-BY   DE-HB   DE-HE   DE-HH   DE-MV   DE-NI
##       0       0       0       0     236       0       0       0       0       0
##   DE-NW   DE-RP   DE-SH   DE-SL   DE-SN   DE-ST   DE-TH
##       0       0       0       0       0       0       0
```

Aus orthografischen Gründen und aus Gründen der besseren Lesbarkeit sind Umbenennungen zur Vorbereitung der Graphik sinnvoll.

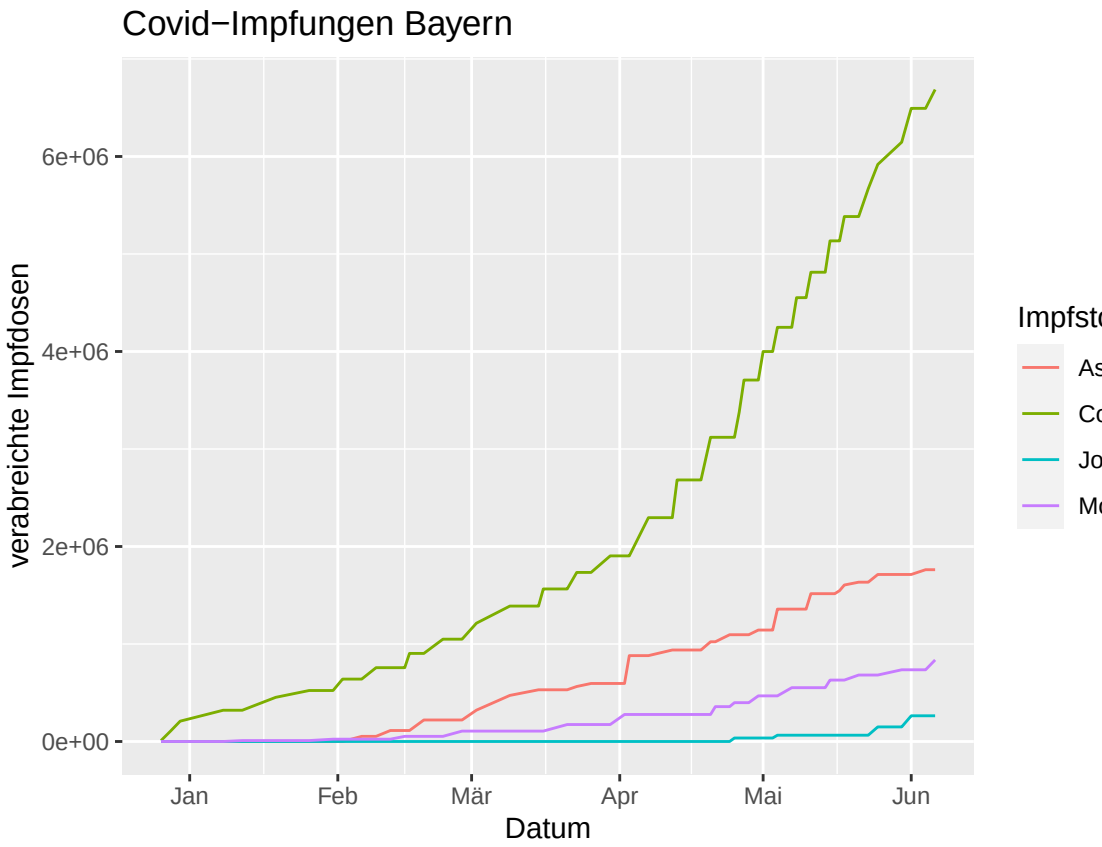
Da die Benennungen der Variablen danach nicht mehr dem Advanced R Style genügen, erfolgt die Umbenennung ausschließlich in dem nur für die Erstellung der Graphik vorgesehenen Datensatz vaccs_bayern_graph.

```
# Umbenennung von "DE-BY" zu "Bayern"
levels(vaccs_bayern_graph$region)[levels(vaccs_bayern_graph$region) == "DE-BY"] <- "Bayern"
```

```
# Umbenennug der Levels von "impfstoff":


# This function renames the levels of "impfstoff"
rename_levels_impfstoff <- function(df_vaccs){

  levels(df_vaccs$impfstoff)[levels(df_vaccs$impfstoff) == "astra"] <- "Astrazeneca"
  levels(df_vaccs$impfstoff)[levels(df_vaccs$impfstoff) == "comirnaty"] <- "Comirnaty"
  levels(df_vaccs$impfstoff)[levels(df_vaccs$impfstoff) == "johnson"] <- "Johnson"
  levels(df_vaccs$impfstoff)[levels(df_vaccs$impfstoff) == "moderna"] <- "Moderna"

  return(df_vaccs)
}

vaccs_bayern_graph <- rename_levels_impfstoff(vaccs_bayern_graph)
```

```
ggplot(data = vaccs) +
  geom_line(vaccs_bayern_graph, mapping = aes(x = date, y = dosen, color = impfstoff)) +
```

```
  labs(title = "Covid-Impfungen Bayern", x = "Datum", y = "verabreichte Impfdosen", color = "Impfstoff")
```



Erstellung der Graphik:

b)

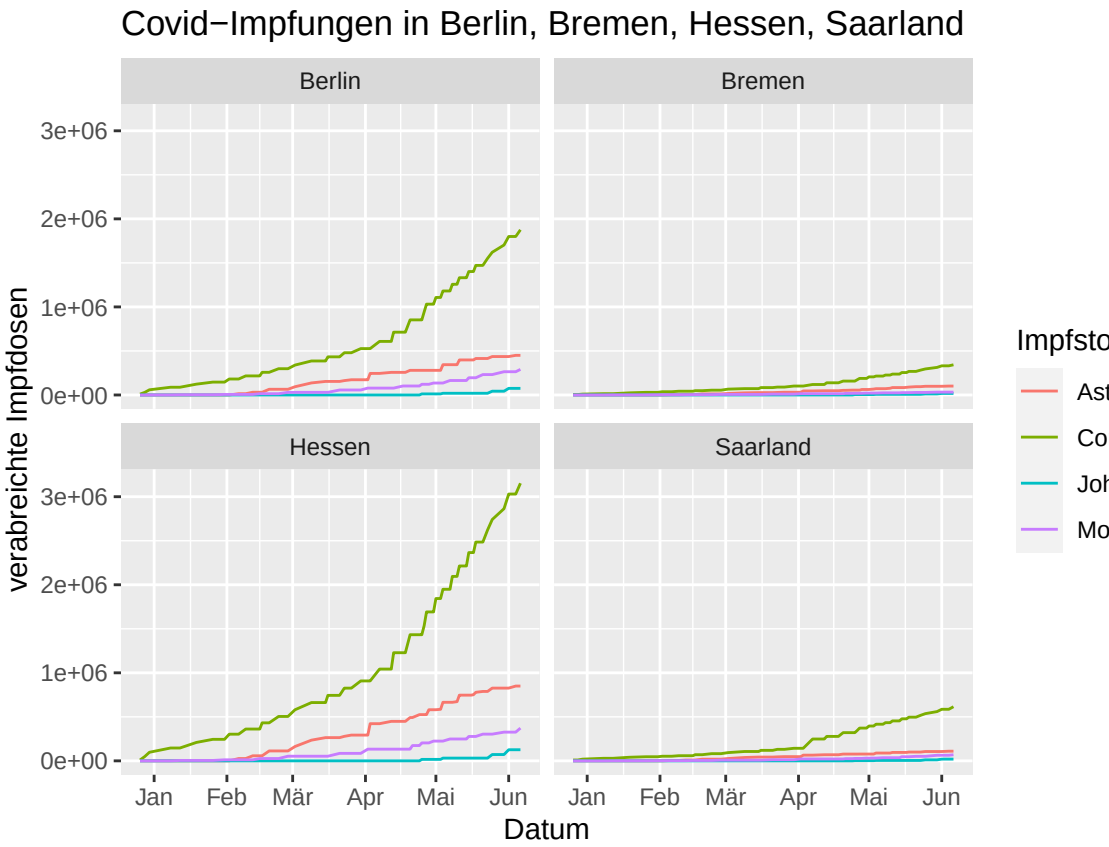
In ähnlichen Liniendiagramm wie in a) für Bayern soll der Fortschritt der Covid-Impfungen im Saarland, in Bremen, Hessen und Berlin, wie er sich aus den Daten ergibt, darstellt werden.

```
# Erstellen einese Unterdatensatzes
# mit den Daten für das Saarland und für Bremen,
# Hessen und Berlin:
vaccs_select_graph <- subset(vaccs, region %in% c("DE-SL", "DE-HB", "DE-HE", "DE-BE"))
```

Vorbereitende Schritte: Aus orthografischen Gründen und aus Gründen der besseren Lesbarkeit sind auch hier Umbenennungen zur Vorbereitung der Graphik sinnvoll.

```
# Umbenennung der Bezeichnungen der Bundesländer
levels(vaccs_select_graph$region)[levels(vaccs_select_graph$region) == "DE-SL"] <- "Saarland"
levels(vaccs_select_graph$region)[levels(vaccs_select_graph$region) == "DE-HB"] <- "Bremen"
levels(vaccs_select_graph$region)[levels(vaccs_select_graph$region) == "DE-HE"] <- "Hessen"
levels(vaccs_select_graph$region)[levels(vaccs_select_graph$region) == "DE-BE"] <- "Berlin"

# Umbenennung der Levels von "impfstoff"
vaccs_select_graph <- rename_levels_impfstoff(vaccs_select_graph)
```

```
ggplot(data = vaccs_select_graph) +
  geom_line(mapping = aes(x = date, y = dosen, color = impfstoff)) +
  labs(title = "Covid-Impfungen in Berlin, Bremen, Hessen, Saarland", x = "Datum", y = "verabreichte Imp
  facet_wrap(~ region, nrow = 2)
```



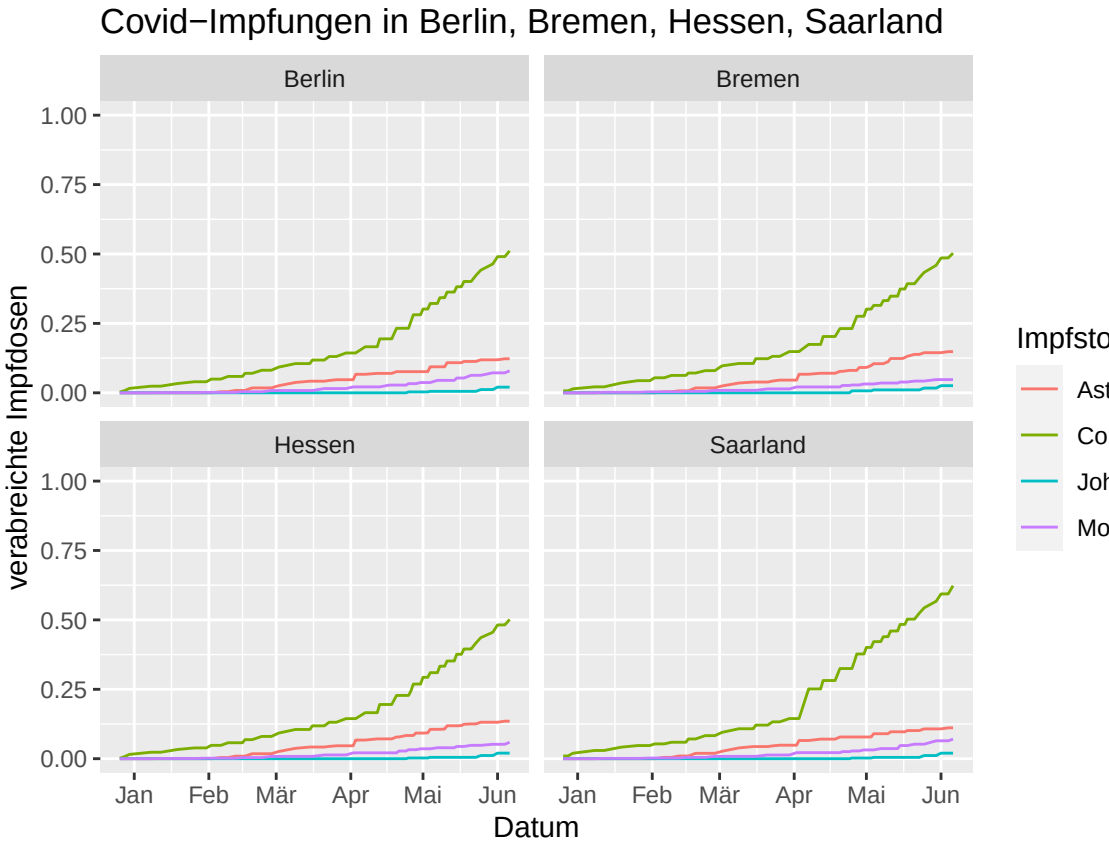
Erstellung der Graphik

c)

```
vrel <- vaccs_select_graph

vrel[vrel$region == "Berlin", ]$dosen <- vrel[vrel$region == "Berlin", ]$dosen / 3669491
vrel[vrel$region == "Bremen", ]$dosen <- vrel[vrel$region == "Bremen", ]$dosen / 681202
vrel[vrel$region == "Hessen", ]$dosen <- vrel[vrel$region == "Hessen", ]$dosen / 6288080
vrel[vrel$region == "Saarland", ]$dosen <- vrel[vrel$region == "Saarland", ]$dosen / 986887
```

Vorbereitende Schritte

```
ggplot(data = vrel) +
  geom_line(mapping = aes(x = date, y = dosen, color = impfstoff)) +
  labs(title = "Covid-Impfungen in Berlin, Bremen, Hessen, Saarland", x = "Datum", y = "verabreichte Imp
  scale_y_continuous(limits = c(0,1)) +
  facet_wrap(~ region, nrow = 2)
```



Erstellung der Graphik

Quellenangaben

Für die Programmierung:

- The Art of R Programming
- R for Data Science (https://r4ds.had.co.nz/index.html)
- Hilfefunktion von R
- https://methodenlehre.github.io/einfuehrung-in-R/grafiken-mit-ggplot2.html
- https://pandar.netlify.app/post/grafiken-mit-ggplot2/
- http://www.cookbook-r.com/Manipulating_data/Renaming_levels_of_a_factor/

Sowie

- https://de.wikipedia.org/wiki/Liste_der_deutschen_Bundesl%C3%A4nder_nach_Bev%C3%B6lkerung
- für das Englische:
  - https://www.linguee.de u. ä.