

# Das ist ein Quarto-Dokument

Hier kommt dein Name

## Inhaltsverzeichnis

Einleitung . . . . .	1
Code-Chunks . . . . .	2
Aufgabe 1 . . . . .	2
Aufgabe 2 . . . . .	3
Aufgabe 3 . . . . .	4
Aufgabe 4 . . . . .	8
Aufgabe 5 . . . . .	8
Aufgabe 6 . . . . .	8

## Einleitung

Dieses Quarto-Dokument könnt ihr als PDF anzeigen lassen.

Um aus einem Quarto-Dokument ein PDF erstellen zu können, muss eine LaTeX-Distribution installiert sein. Am einfachsten geht das mit dem Paket TinyTex. Dazu muss man lediglich diese zwei Zeilen Code in dieser Reihenfolge ausführen. Diese Installation kann eine Weile dauern.

```
# install.packages("tinytex") # den «#» vor dem Befehl bei euch entfernen  
# tinytex::install_tinytex() # den «#» vor dem Befehl bei euch entfernen
```

Setzt man nun oben links im Editor das Häkchen bei «Render on Save» (rot), wird beim Speichern aus einem Quarto-Dokument ein PDF erzeugt - oder man klickt einfach auf den blauen Pfeil (blau), der mit «Render» beschriftet ist, vgl. Figure 1.

Die Abbildungen (vgl. Figure 1.) sind noch auf Englisch beschriftet, wir wollen das aber auf Deutsch. Dazu geht ihr in die YAML (das ist das Ding ganz oben, wo `title` usw. steht) und ändert bei `lang` «en» zu «de».

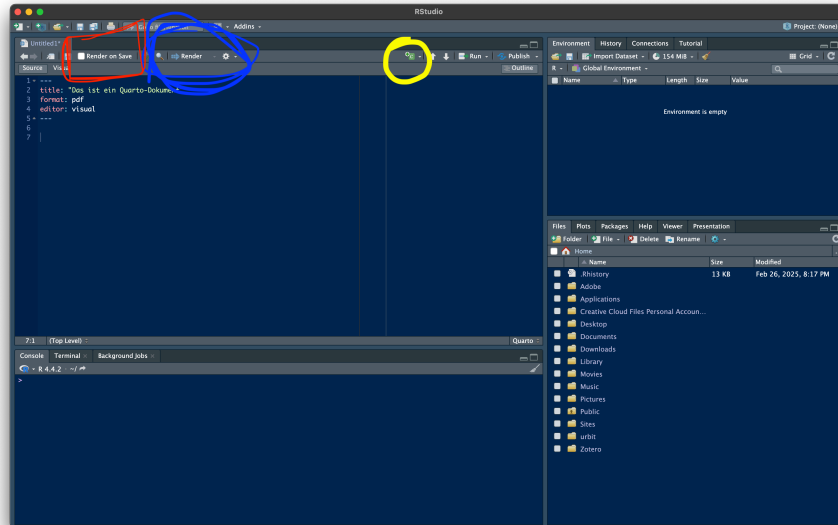


Figure 1: Ein geöffnetes Quarto-Dokument.

## Code-Chunks

Das hier ist ein Code-Chunk:

```
# Diese Bereiche sind sog. Code-Chunks. Hier kommt Code. In allen anderen
# Bereichen kommt normaler Text. In einem Quarto-Dokument kann man also
# normalen Fliesstext schreiben. In einem R-Skript kann man nur Text
# schreiben, wenn vorher ein «#» steht. Auch in den Code-Chunks in einem
# Quarto Dokument (also genau dieser Bereich hier) muss man einen «#»
# verwenden, wenn man Text schreiben will.
```

Mit dem Button (gelb) in Figure 1 erstellt ihr einen neuen Code-Chunk.

## Aufgabe 1

1. Erstellt einen neuen Code-Chunk. Ladet dort das `tidyverse`. Ihr kennt bereits zwei Befehle, um Pakete zu laden.
2. Gebt dem Code-Chunk das `label` «Pakete laden».
3. Erstellt unter dem `label` den `include`-Tag und den `message`-Tag.
4. Rendert das Dokument einmal mit (1) `include` auf «true» und `message` auf «true», dann mit (2) `include` auf «true» und `message` auf «false» und zum Schluss mit (3) `include` auf «false» und `message` auf «false». Schaut, was sich im gerenderten Dokument zwischen diesen drei Varianten verändert.

## Beispiel

```
library(dplyr)
```

```
# klassische Version mit output
```

```
library("tidyverse")
```

```
-- Attaching core tidyverse packages ----- tidyverse 2.0.0 --
```

```
v forcats   1.0.0      v readr     2.1.5
```

```
v ggplot2   3.5.1      v stringr  1.5.1
```

```
v lubridate 1.9.4      v tibble   3.2.1
```

```
v purrr     1.0.4      v tidyr    1.3.1
```

```
-- Conflicts ----- tidyverse_conflicts() --
```

```
x dplyr::filter() masks stats::filter()
```

```
x dplyr::lag()     masks stats::lag()
```

```
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become
```

```
# mit librarian, damit es installiert wird, wenn es nicht da ist und quiet für stilles ausführen
```

```
librarian::shelf("tidyverse", quiet = TRUE)
```

```
# Die beiden Versionen stören sich nicht, aber es braucht unnötig Zeit. Behalten Sie die, die
```

## Aufgabe 2

R hat den Beispieldatensatz `mtcars`. Dieser Datensatz muss daher nicht extra eingelesen werden, man kann ihn einfach verwenden.

```
mtcars |>  
  head()
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

Erklärt, was `head()` macht und was sich verändert, wenn man dort «3» in die Klammer schreibt.

**Antwort:** Der Befehl `head()` zeigt den Datensatz so an, wie er ist, aber nur die ersten 6 Zeilen. Wenn man in die Klammern 3 schreibt, werden die ersten 3 Zeilen angezeigt.

Wie heisst das Zeichen nach `mtcars`? Welches Zeichen gibt es noch, welches dasselbe macht und auch gleich heisst? Wie unterscheiden sie sich?

**Antwort:** Das Zeichen hinter `mtcars` ist der Pipe-Operator. Es gibt noch die ältere Version des Pipe-Operator `%>%` (den man auch den Magritter nennt). Der wichtigste Unterschied: Der `%>%` wird durch das `tidyverse`--Paket definiert (genauer `dplyr`::) und setzt damit voraus, dass `dplyr` beziehungsweise `tidyverse` geladen wurden. Der `|>` wird auch generischer Pipe-Operator genannt, weil der zu base-R gehört und darum immer funktioniert.<sup>1</sup>

### Aufgabe 3

Erklärt, was in den folgenden Code-Chunks passiert.

```
mtcars |>
  select(cyl, hp)|>
  head()
```

	cyl	hp
Mazda RX4	6	110
Mazda RX4 Wag	6	110
Datsun 710	4	93
Hornet 4 Drive	6	110
Hornet Sportabout	8	175
Valiant	6	105

**Antwort 1:** Es wird der Datensatz `mtcars` genommen, und dann werden die beiden Variablen `cyl` und `hp` mit `select()` ausgewählt und dann wird die Tabelle mit den ersten 6 Zeilen ausgegeben.

```
mtcars |>
  filter(hp >180)
```

---

<sup>1</sup>Fun-Fact: Der `%>%` ist älter. Das Prinzip des Pipens hat base-R erst später übernommen.

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Duster 360	14.3	8	360	245	3.21	3.570	15.84	0	0	3	4
Cadillac Fleetwood	10.4	8	472	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	10.4	8	460	215	3.00	5.424	17.82	0	0	3	4
Chrysler Imperial	14.7	8	440	230	3.23	5.345	17.42	0	0	3	4
Camaro Z28	13.3	8	350	245	3.73	3.840	15.41	0	0	3	4
Ford Pantera L	15.8	8	351	264	4.22	3.170	14.50	0	1	5	4
Maserati Bora	15.0	8	301	335	3.54	3.570	14.60	0	1	5	8

**Antwort 2:** Es wird der Datensatz `mtcars` genommen, und dann werden die Zeilen herausgefiltert (behalten) die einen `hp` über 180 haben. Der Datensatz wird am Ende wie vollständig angezeigt. Er hat noch 7 Zeilen. In der Spalte `hp` stehen nur Werte über 180.

```
mtcars |>
  group_by(cyl) |>
  summarise(durschn_hp = mean(hp))
```

```
# A tibble: 3 x 2
  cyl durschn_hp
<dbl> <dbl>
1     4      82.6
2     6     122.
3     8     209.
```

**Antwort 3:** Es wird der Datensatz `mtcars` genommen, und dann wird der Datensatz gruppiert. Es werden also wie Unterdatensätze gebaut für jede unterschiedliche Anzahl an Zylindern (`cyl`). Und dann wird für jeden der Unterdatensätze die Variable `hp` als Mittelwert (`mean()`) zusammengefasst (mit `summarise()`). Die Spalte in denen die Mittelwerte stehen, heisst `durschn_hp`. Die `hp` könnte man noch runden. Wie geht das?

```
mtcars |>
  group_by(cyl) |>
  summarise(durschn_hp = mean(hp)) |>
  mutate(ueber100hp = if_else(durschn_hp > 100, "Jep", "Nö"))
```

```
# A tibble: 3 x 3
  cyl durschn_hp ueber100hp
<dbl> <dbl> <chr>
1     4      82.6 Nö
2     6     122. Jep
3     8     209. Jep
```

**Antwort 4:** Die ersten drei Zeilen machen dasselbe wie in **Antwort 3**. In der letzten Zeile der Pipe wird eine dichotome Variable gebaut, die `ueber100hp` heisst und überall da ein “Jep” stehen hat, wo `durschn_hp` grösser als 100 ist und immer wenn `durschn_hp` unter 100 liegt, dann steht da ein “Nö”. Eine Tabelle mit der Gruppierungsvariablen `cyl`, den Summary-Variablen `durschn_hp` und `ueber_100hp` wird angezeigt (bei 5 `cyl` ist `ueber100hp` ein “Nö”, sonst “jep”).

```
mtcars_hp_tbl <- mtcars |>
  group_by(cyl)|>
  summarise(durschn_hp = mean(hp)) |>
  mutate(ueber100h = if_else(durschn_hp > 100, "Jep", "Nö"))
```

**Antwort 5:** Im Grunde passiert dasselbe wie in Antwort 4, nur dass in der ersten Pipe-Zeile geschrieben steht, dass die in der Pipe verarbeitete `mtcars` in das neue Datenobjekt (aka Tabelle) `mtcars_hp_tbl` geschrieben wird.

```
library(ggplot2)
mtcars |>
  ggplot(aes(x = mpg, y = hp)) +
  geom_point(size = 2, color = "#002fa7") +
  theme_bw() +
  labs(
    x = "Verbrauch in mpg",
    y = "Pferdestärke (PS)",
    title = "Verbrauch und PS") +
  geom_hline(
    yintercept = mean(mtcars$hp),
    linetype = "dotted",
    color = "blue") +
  annotate("text", x = max(mtcars$mpg) - 5, y = mean(mtcars$hp) + 10,
    label = paste("Durchschnitt:", round(mean(mtcars$hp)), "PS"),
    color = "midnightblue", hjust = 1)
```

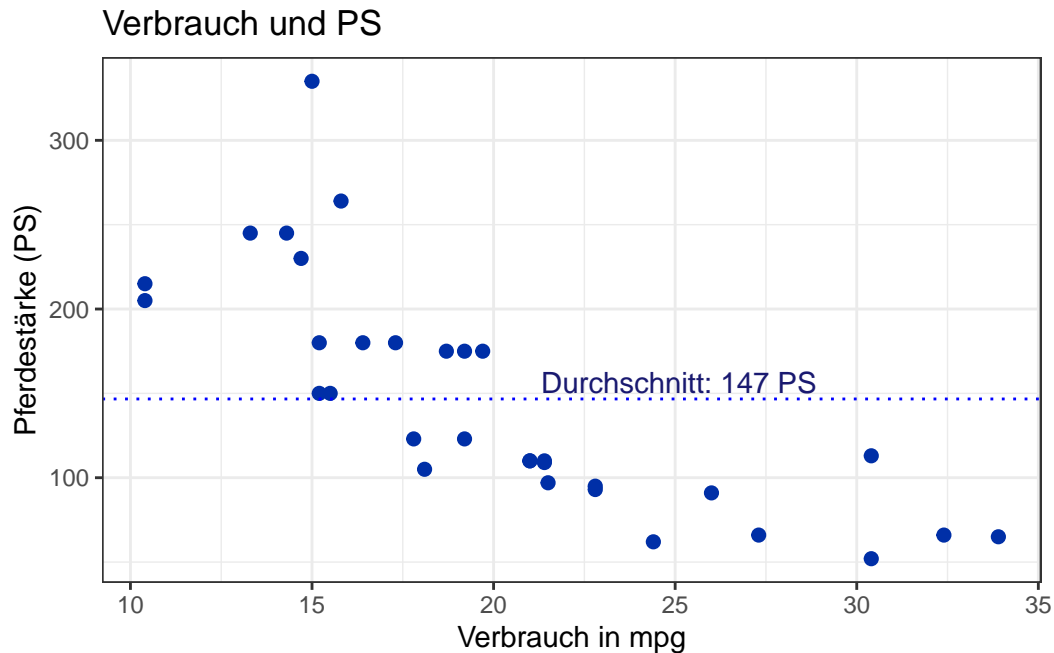


Figure 2: Was ist das für ein Diagramm?

**Antwort:** Hier wird erstmal das Paket `ggplot2` geladen (braucht man nicht, wenn oben `tidyverse` geladen wurde). Die Pipe sagt Folgendes:

Nimm den Datensatz `mtcars` und dann baue ein `ggplot`, wobei die aesthetics (`aes`) definiert sind für `x` durch die Variable `mpg` und `y` durch `hp`. Baue auf das Grundgerüst eine Datenvisualisierung (`geom`) als Punkte (`geom_point()`), wobei alle Punkte die größe 2 haben sollen und die Farbe blau (in hex-code `#002fa7`). Als Vorlage für die Grafik soll `theme_bw()` genommen werden (schwarz-weiss). Als Label sollen für `x` = "Verbrauch in mpg" und `y` = "Pferdestärke (PS)" angeschrieben werden. Der Titel der Grafik soll "Verbrauch und PS" sein. Dann soll in einem weiteren Layer mit `geom_hline` eine horizontale Linie eingezeichnet werden, die in `y` die Höhe des Mittelwertes von der Variable `hp` aus dem Datensatz `mtcars` hat (mit `Datensatz$Variable` werden in R Variablen aus einem bestimmten Datensatz angesprochen). Die Linie soll gepunktet sein und die Farbe "blue" haben. Dann soll in einem weiteren Layer eine Text-Beschriftung (`annotate("text")`) stehen mit der Position `x` = dem Maximalwert von `mtcars$mpg` (na gut ein bisschen tiefer, also -5) und `y` bei dem Durchschnitt (`mean`) von `mtcars$hp`, aber ein bisschen höher um + 10. Das Label, das da stehen soll, wird zusammengeklebt (`paste`) aus "Durchschnitt", dann dem gerundeten Mittelwert von `mtcars$hp` (`round(mean(mtcars$hp))`) und dahinter "PS". Als Farbe in dem Fall "midnightblue". Horizontal soll der Text rechtsbündig sein (`hjust = 1`).

Disclaimer: Es reicht auch, wenn Sie schreiben: Es wird der Datensatz `mtcars` genommen und die Daten in einem Scatterplot (oder Punktdiagramm) dargestellt.

## Aufgabe 4

Gehe in die YAML (das ist das Ding ganz oben, wo `title` usw. steht) und ändere `toc` auf «true». Was verändert sich nun im gerenderten Dokument?

**Antwort:** Jetzt erscheint im Zieldokument ein Inhaltsverzeichnis. Wer mag, ergänzt noch: `toc-title: "Inhaltsverzeichnis"`

## Aufgabe 5

Den Beispieldatensatz `mtcars` muss man nicht extra einlesen. Andere Datensätze müssen in R importiert werden. Im Unterordner «data» findet ihr den Datensatz zur Schweizer Nachwahlbefragung (Selects) aus dem Jahr 2019.

```
df_selects <- readr::read_csv(here::here("data/Selects2019.csv"))
```

Erkläre, was hier passiert.

**Antwort:** Es wird der Datensatz “Selects2019.csv” importiert mit der Funktion `read_csv()` aus dem Paket `readr::`. Das `here::here()` sorgt dafür, dass der Pfad zum Datensatz bei allen und immer von derselben Stelle aus gesucht und gefunden wird.

Was macht `readr::`? Braucht es das?

**Antwort:** Es wird hier ein für R “fremdes” Datenformat eingelesen (Kommaseparierte Datei ‘csv’). Dazu braucht es das Paket `readr::`. Es ginge auch mit `read.csv` aus base-R, aber das ist schlampiger und weniger konsistent - also nicht zu empfehlen (vor allem nicht in Western Europe mit dem Komma als Dezimaltrenner.).

## Aufgabe 6

Der Datensatz ist nun in «df\_selects» gespeichert. Anstatt wie oben «mtcars» zu schreiben, schreibst du nun «df\_selects».

1. Berechne das Durchschnittsalter der Befragten. Tipp: Du brauchst dafür einen Pipe-Operator und zwei Funktionen, `mean()` und `summarise()`.

```
df_selects |>
  summarise(durschn_alter = mean(alter))
```



```
# A tibble: 1 x 1
  durschn_alter
      <dbl>
1         51.0
```

2. Berechne, wie viele Männer und Frauen befragt wurden. Tipp: Du brauchst dafür einen Pipe-Operator und eine Funktion, `count()`.

```
df_selects |>
  count(gender)
```

```
# A tibble: 2 x 2
  gender      n
  <chr> <int>
1 Female  3465
2 Male    3199
```

(Der Datensatz ist sehr binär. :-)

3. Berechne, wie viele Frauen unter 30 befragt wurden. Tipp: Du brauchst dafür zwei Pipe-Operatoren und zwei Funktionen, `filter()` und `count()`.

```
df_selects |>
  filter(gender == "Female" & alter < 30) |>
  count()
```

```
# A tibble: 1 x 1
      n
  <int>
1    541
```

4. Berechne das Durchschnittsalter für die Wähler jeder Partei. Tipp: Du brauchst dafür zwei Pipe-Operatoren und drei Funktionen, `group_by()`, `summarise()` und `mean()`.

```
df_selects |>
  group_by(partei) |>
  summarise(avg_age = mean(alter, na.rm = TRUE))
```

```
# A tibble: 8 x 2
  partei                avg_age
  <chr>                <dbl>
1 CVP/PDC - Christian Democratic People's Party 54.9
2 FDP/PLR - Liberal Radical Party             56.3
3 GLP/PVL - Green Liberal Party               46.0
4 GPS/PES - Green Party                      48.7
5 Others                                     52.9
6 SP/PS - Social Democratic Party            54.2
7 SVP/UDC - Swiss People's Party             56.1
8 <NA>                                       46.7
```

- Wähle mit `select()` die Spalten für das Alter, das Geschlecht und die Religion aus. Erstelle mit `mutate()` eine neue Spalte/Variable, die den Namen «religioesTRUE» hat. Die Variable «religioesTRUE» soll 1 sein, wenn die Person irgendeine Religion hat und 0, wenn sie keine hat. Verwende dazu `if_else()`.

```
Religioes <- df_selects |>
  select(alter, gender, religion) |>
  mutate(religioesTRUE = if_else(religion == "None", 0, 1))

Religioes
```

```
# A tibble: 6,664 x 4
  alter gender religion  religioesTRUE
  <dbl> <chr>  <chr>          <dbl>
1    69 Male   Catholic        1
2    32 Female None           0
3    91 Female Catholic        1
4    56 Female Catholic        1
5    52 Male   Catholic        1
6    53 Male   None           0
7    34 Female Catholic        1
8    21 Male   Protestant       1
9    72 Female None           0
10   26 Male   Catholic        1
# i 6,654 more rows
```