

Quarto 2

Code-Chunk-Optionen und Querverweise

Daniela Palleschi

2023-05-30

Inhaltsverzeichnis

Update: Leistungspunkte	2
Wiederholung	2
Histogramm und Dichtediagramm	2
Heutige Ziele	4
Lust auf mehr?	4
1 Einrichtung	4
2 Code chunks	5
2.1 Chunk labels	5
2.1.1 Vorteile der Verwendung von Chunk-Labels	5
2.1.2 Chunk-Chunks sollten...	6
2.2 Chunk-Optionen	7
2.3 Globale Optionen	8
3 Tabellen	9
3.0.1 Tabellenbeschriftungen	10
3.0.2 Spaltennamen	11
4 Figuren	13
4.1 Figurenbeschriftung	13
4.2 Abbildungsbeschriftung	14
4.3 Größe der Figuren	15
5 Querverweise	16
6 Aufgaben	17

Fragen zum Bericht

Geht zu menti.com und gebt den Code auf dem nächsten Bildschirm ein

Update: Leistungspunkte

- Studienleistungen
 - 3LP
 - * 1LP: Hochladen des wöchentlichen Programmierungsskripts (mindestens 8 von den 13 Wochen)
 - * ~~1LP~~ **2LP**: zwei “in-class” Übungen (~~je 0,5LP~~) (**je 1LP**)
 - * ~~1LP: Hausarbeit (fällig am 15. August)~~

Wiederholung

Letzte Woche haben wir...

- einen Bericht über Eye-Tracking-Lesedaten von Biondo et al. (2022) erstellt
- bekannte und neue Diagrammtypen interpretiert
- bekannte Darstellungsformen reproduziert

Histogramm und Dichtediagramm

1. Was zeigen diese Diagramme?
 - Verteilung der Reaktionszeiten pro Genauigkeitsstufe
2. Was stellen die Spitzenwerte dar (z. B. Mittelwert, Median, Modus)?
 - die *Modus*-Reaktionszeit pro Genauigkeitsstufe
3. Gibt es einen (ungefähr) gleichen Anteil an richtigen (1) und falschen (0) Antworten? Wie kann man das feststellen?
 - Nein, es gibt viel mehr korrekte Antworten, das sieht man im Histogramm, das die Anzahl der Beobachtungen (y-Achse: Anzahl) pro Reaktionszeit-Bin (x-Achse) zeigt

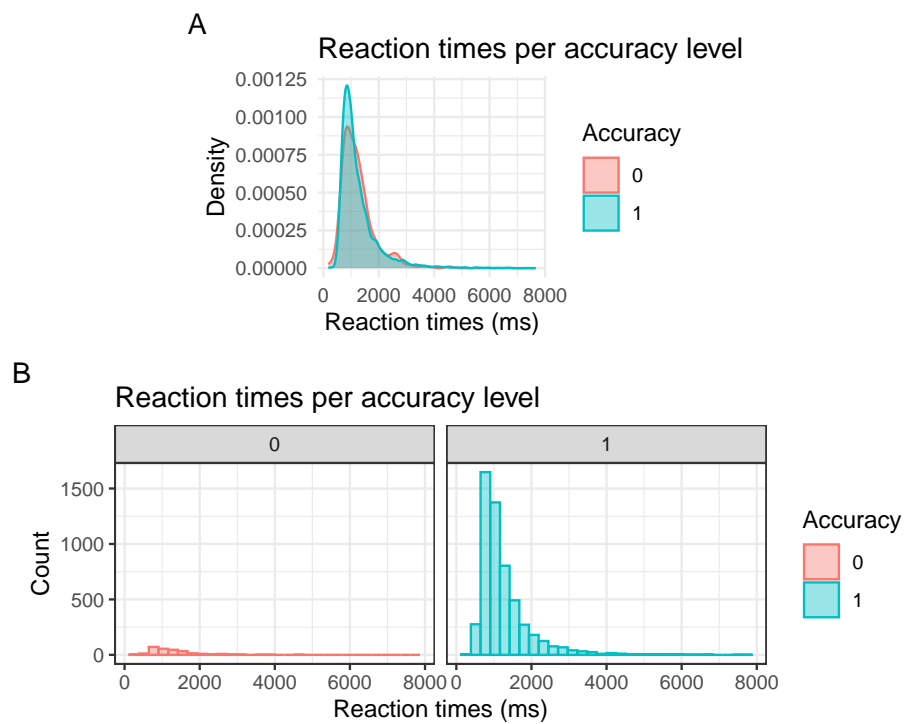


Abbildung 1: Dichte- und Histogrammdiagramme

Heutige Ziele

Heute werden wir...

- lernen, wie man Code-Chunk-Optionen verwendet
- lernen, wie man die Größe von Abbildungen kontrolliert
- lernen, wie man Abbildungsunterschriften hinzufügt
- lernen, wie man formatierte Tabellen druckt
- lernen, wie man Querverweise erstellt

Lust auf mehr?

- Ch. 29 ([Quarto](#)) in Wickham et al. (o. J.)
 - Absatz 29.5 ([Code Chunks](#))
 - Absatz 29.6 ([Figures](#))
 - Absatz 29.7 ([Tables](#))

1 Einrichtung

1. Neuer Ordner für diese Woche
2. Neues Quarto-Dokument
3. YAML aktualisieren
4. Pakete laden
 - `tidyverse`
 - `knitr` (neu)
 - `kableExtra` (neu)
5. Ladet die Daten der Datei `flight.csv` ein

```
pacman::p_load(tidyverse,  
               knitr,  
               kableExtra)  
  
df_flights <- read_csv(here::here("daten",  
                                  "flights.csv"))
```

2 Code chunks

Abkürzungen:

- Cmd/Strg+Alt+I: neuen Code-Chunk einfügen
- Cmd/Strg+Enter: eine einzelne Code-Zeile ausführen
- Cmd/Strg+Shift+Enter: ganzen Code-Chunk ausführen
- Code Chunks sollten relativ in sich geschlossen (*self-contained*) sein
 - und auf eine einzige Aufgabe fokussiert sein

2.1 Chunk labels

- wir können jedem Code-Chunk Spezifikationen geben, indem wir `#|` *direkt* unter “`{r}`”
 - `#| label: simple-math` wird den Chunk `simple-math` nennen

```
```{r}
#| label: simple-math

4 + 4
```
```

[1] 8

2.1.1 Vorteile der Verwendung von Chunk-Labels

1. Über das Dropdown-Menü im Skript-Editor können wir zu bestimmten Code Chunks navigieren
2. Grafiken (z. B. Plots), die von Chunks erzeugt werden, erhalten nützliche Namen, die das spätere Auffinden erleichtern (mehr dazu in Kürze)

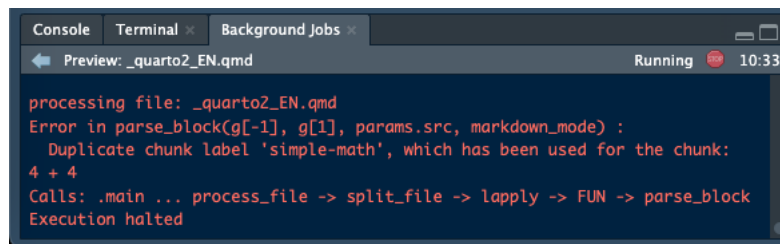
2.1.2 Chunk-Chunks sollten...

- kurz und informativ sein
- keine Leerzeichen enthalten (- oder _ verwenden) *einmalig* im Dokument sein (nicht wiederholt)

Eindeutige Chunk-Labels

Chunk-Labels müssen innerhalb eines Skripts *immer* eindeutig sein!

- Wenn dies nicht der Fall ist, erhalten wir beim Rendern eine Fehlermeldung und das Dokument wird nicht gerendert
- Wir erhalten eine informative Fehlermeldung im Fenster ‘Background Jobs’, wenn wir doppelte Chunk-Labels haben, also *immer die Fehlermeldung lesen!* Sie können bei der Fehlersuche sehr hilfreich sein.



```
Console Terminal x Background Jobs x
Preview: _quarto2_EN.qmd Running 10:33

processing file: _quarto2_EN.qmd
Error in parse_block(g[-1], g[1], params.src, markdown_mode) :
  Duplicate chunk label 'simple-math', which has been used for the chunk:
  4 + 4
Calls: .main ... process_file -> split_file -> lapply -> FUN -> parse_block
Execution halted
```

Abbildung 2: Error message when multiple code chunks have the same label `simple-math`

Aufgabe 2.1: Chunk labels

Beispiel 2.1.

1. Fügt ein Chunk-Label zu eurem Code-Chunk hinzu, in den ihr Pakete geladen habt
2. Fügt einen Code Chunk mit dem Tastaturkürzel `Cmd/Strg-Alt-I` hinzu und fügt einige einfache mathematische Berechnungen hinzu
3. Fügt ein Chunk-Label hinzu
4. Probiert die Chunk-Navigationsleiste am unteren Rand des Quelltextfensters aus, um zwischen Code Chunks zu springen
5. Rendert das Dokument

2.2 Chunk-Optionen

- Die Chunk-Ausgabe kann mit **Optionen** formatiert werden, die R mitteilen, was mit dem Code beim Rendern eures Dokuments geschehen soll
 - es gibt fast 60 Optionen!
 - Die wichtigsten Optionen steuern, ob euer Code-Chunk beim Rendern ausgeführt wird und welche Ergebnisse im Ausgabebericht gedruckt werden:
- `eval: false` verhindert, dass der Code in der gerenderten Ausgabe *ausgedruckt* wird
- `include: false` führt den Code aus, zeigt aber weder den Code noch die Ergebnisse im endgültigen Dokument an.
- `echo: false` verhindert, dass der Code, aber nicht die Ausgabe, in der gerenderten Ausgabe erscheint
- `message: false` oder `warning: false` verhindert, dass Meldungen oder Warnungen in der gerenderten Ausgabe erscheinen
- `results: hide` blendet die gedruckte Ausgabe aus; `fig-show: false` blendet Plots aus
- `error: true` rendert das Dokument, auch wenn Fehler aufgetreten sind

Wird der folgende Codeabschnitt in der gerenderten Ausgabe erscheinen? Wird der Code ausgeführt werden?

```
```${r}
#| eval: true
#| label: df-flights1
#| message: false

df_flights <- read_csv(here::here("daten", "flights.csv"))
```
```

Wird der folgende Codeabschnitt in der gerenderten Ausgabe erscheinen? Wird der Code ausgeführt werden?

```
```${r}
#| eval: false
#| label: df-flights2
#| message: false
```

```
fig_flights <- read_csv(here::here("daten", "flights.csv")) %>%
 filter(month == 12) %>%
 ggplot(aes(x = dep_delay, y = arr_delay, colour = carrier)) +
 geom_point() +
 theme_minimal()
```

```

Die folgende Tabelle fasst zusammen, welche Arten von Ausgaben jede Option *unterdrückt*:

| Option | Run code | Show code | Output | Plots | Messages | Warnings |
|----------------|----------|-----------|--------|-------|----------|----------|
| eval: false | X | | X | X | X | X |
| include: false | | X | X | X | X | X |
| echo: false | | X | | | | |
| results: hide | | | X | | | |
| fig-show: hide | | | | X | | |
| message: false | | | | | X | |
| warning: false | | | | | | X |

- Für den Rest des Kurses werden wir nur noch `eval`, `echo`, `include` und `message` verwenden

2.3 Globale Optionen

- die eben erwähnten Chunk-Optionen können auch *global* für euer gesamtes Dokument gesetzt werden, indem ihr sie in eurer YAML unter `execute:` hinzufügen

```
title: "My report"
format:
  html:
    toc: true
execute:
  echo: false
```

- und dann können nachfolgende Code-Chunks die globale Einstellung von Fall zu Fall außer Kraft setzen

💡 Aufgabe 2.2: Chunk options

Beispiel 2.2.

1. Einen neuen Codechunk hinzufügen
 - fügt einige einfache mathematische Berechnungen hinzu (z.B., $5 \cdot 13$)
2. Gib dem Code Chunk ein Label
3. Füge die Chunk-Labels `eval: false` und `echo: true` hinzu
4. Rendert das Dokument.
 - Wird der Code gedruckt?
 - Wird das Ergebnis gedruckt? Warum oder warum nicht?

3 Tabellen

- wir können Tabellen so ausgeben, wie wir sie in der Konsole sehen
- wir können auch weitere Formatierungen mit der Funktion `kable()` aus dem Paket `knitr` hinzufügen

```
df_flights %>%  
  select(1:5) %>%  
  head()
```

```
# A tibble: 6 x 5  
  year month   day dep_time sched_dep_time  
  <dbl> <dbl> <dbl>   <dbl>         <dbl>  
1  2013     1     1     517           515  
2  2013     1     1     533           529  
3  2013     1     1     542           540  
4  2013     1     1     544           545  
5  2013     1     1     554           600  
6  2013     1     1     554           558
```

```
df_flights %>%  
  select(1:5) %>%  
  head() %>%  
  knitr::kable()
```

| year | month | day | dep_time | sched_dep_time |
|------|-------|-----|----------|----------------|
| 2013 | 1 | 1 | 517 | 515 |
| 2013 | 1 | 1 | 533 | 529 |
| 2013 | 1 | 1 | 542 | 540 |
| 2013 | 1 | 1 | 544 | 545 |
| 2013 | 1 | 1 | 554 | 600 |
| 2013 | 1 | 1 | 554 | 558 |

3.0.1 Tabellenbeschriftungen

- wir können der Tabelle auch eine Beschriftung hinzufügen, indem wir ein `label:`
 - und muss mit `tbl-` für Tabellen beginnen
- Wir können auch eine Tabellenbeschriftung hinzufügen (`tbl-cap:`), die über der Tabelle gedruckt wird, wenn wir das Dokument rendern hinzufügen

```

```{r}
#| output-location: fragment
#| label: tbl-flights
#| tbl-cap: "A table made with `knitr`."

df_flights %>%
 select(1:5) %>%
 head() %>%
 knitr::kable(
) %>%
 kable_styling(font_size = 25) # from kableExtra
```

```

```

```{r}
#| output-location: fragment
#| label: flights
#| tbl-cap: "A table made with `knitr`."

df_flights %>%
 select(1:5) %>%
 head() %>%
 knitr::kable(
) %>%

```

Tabelle 2: A table made with knitr.

year	month	day	dep_time	sched_dep_time
2013	1	1	517	515
2013	1	1	533	529
2013	1	1	542	540
2013	1	1	544	545
2013	1	1	554	600
2013	1	1	554	558

```
kable_styling(font_size = 25) # from kableExtra
...

```

### 3.0.2 Spaltennamen

- Zum Schluss drucken wir noch bessere Spaltennamen

```
```{r}
#| output-location: column-fragment
#| label: tbl-flights2
#| tbl-cap: "A table made with `knitr`."
df_flights %>%
  select(1:5) %>%
  head() %>%
  knitr::kable(
    col.names = c("Year", "Month", "Day",
                  "Dep. Time", "Sched. Dep. Time")
  ) %>%
  kable_styling(font_size = 25) # from kableExtra
...

```

Tabelle 3: A table made with knitr.

year	month	day	dep_time	sched_dep_time
2013	1	1	517	515
2013	1	1	533	529
2013	1	1	542	540
2013	1	1	544	545
2013	1	1	554	600
2013	1	1	554	558

Tabelle 4: A table made with knitr.

Year	Month	Day	Dep. Time	Sched. Dep. Time
2013	1	1	517	515
2013	1	1	533	529
2013	1	1	542	540
2013	1	1	544	545
2013	1	1	554	600
2013	1	1	554	558

💡 Aufgabe 3.1: Tables

Beispiel 3.1.

1. Kopiere den Code von der letzten Folie (aus dem HTML, PDF oder den Folien)
2. Fügt ein Tabellenetikett und eine Beschriftung ein
3. Rendert das Dokument
4. Ist die Tabelle gedruckt? Hat sie eine Tabellennummer?

4 Figuren

4.1 Figurenbeschriftung

- Beschriftungen (`labels`) für Code Chunks, die eine Abbildung *ausdrucken*, müssen mit `fig-` beginnen
 - die Figur hat dann eine Nummer, wenn sie gedruckt wird

```
```{r}
#| label: fig-flights-dec120

fig_flights
```
```

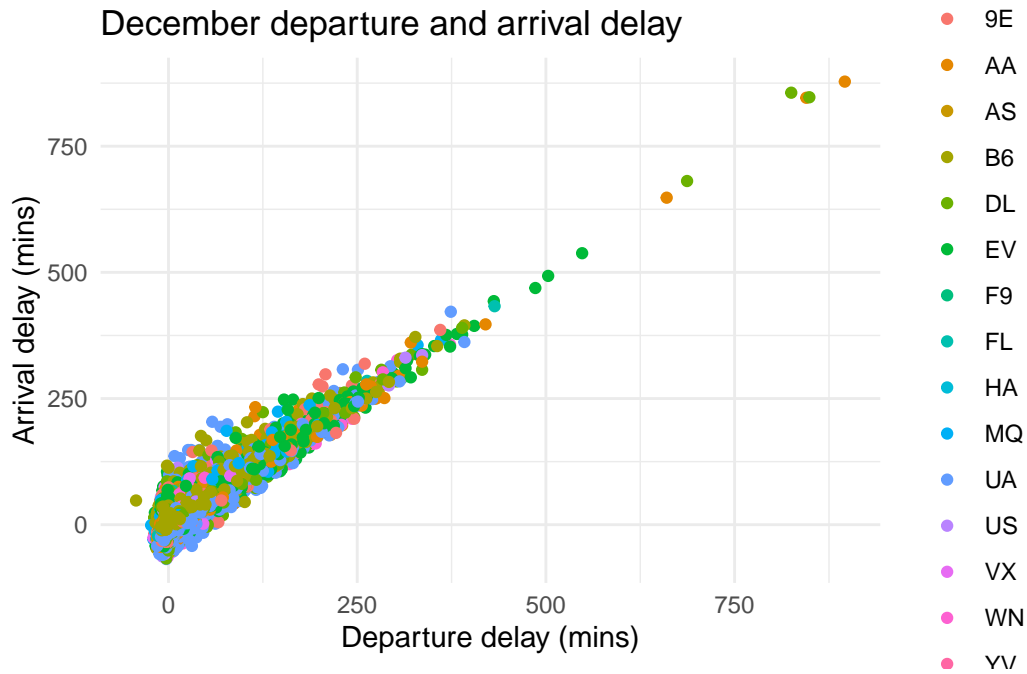


Abbildung 3: ?(caption)

4.2 Abbildungsbeschriftung

- `fig-cap`: fügt eine Bildunterschrift ein, die im gerenderten Dokument erscheint
 - die Beschriftung immer mit Anführungszeichen umschließen! z.B., `fig-cap: "..."`

```

```{r}
#| label: fig-flights-dec120-2
#| fig-cap: "Departure delay by arrival delay for December 2013. Airline is indicated via
fig_flights
```

```

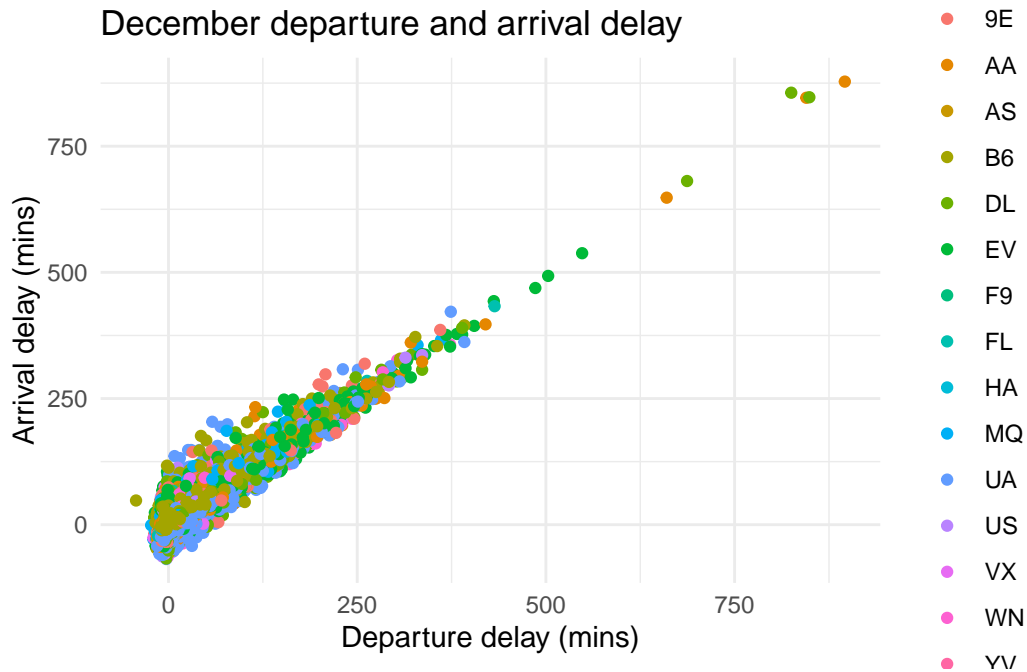


Abbildung 4: Departure delay by arrival delay for December 2013. Airline is indicated via point colour.

4.3 Größe der Figuren

- Eine große Herausforderung bei Grafiken in Quarto ist es, die Figuren in die richtige Größe und Form zu bringen
- Es gibt fünf Hauptoptionen, die hilfreich sein können:
 - `fig-width`: legt die Breite der Figur in Zoll fest (z.B. `fig-width = 4`)
 - `fig-height`: legt die Höhe der Figur in Zoll fest (z. B. `fig-height = 4`)
 - `fig-asp`: legt das Seitenverhältnis der Figur fest (wenn Sie nur Höhe oder Breite festlegen; z. B. `fig-asp = 0,618`)
 - `out-width`: legt die Breite der Figur in Prozent der Zeilenbreite fest (z. B. `out-width = "70%"`)
 - `out-height`: setzt die Höhe der Figur in Prozent zur Zeilenbreite (z.B. `out-height = "30%"`)
 - `fig-align`: `centre` zentriert die Abbildung auf der Ausgabeseite

Wie wird die Größe dieser Figur aussehen?

```

```{r}
#| label: fig-flights-dec120-3
#| fig-cap: "Departure delay by arrival delay for December 2013. Airline is indicated via
#| out-width: "60%"
#| fig-asp: .618
#| fig-align: center
#| output-location: fragment

fig_flights
```

```

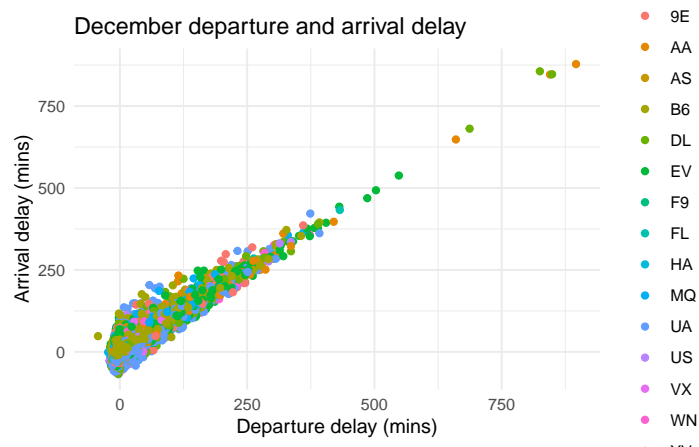


Abbildung 5: Departure delay by arrival delay for December 2013. Airline is indicated via point colour.

5 Querverweise

- Wir können auch im Text auf Diagramme oder Tabellen verweisen, indem wir @ gefolgt vom label eingeben
 - z.B.: Dies ist ein Text, der @fig-flights-dec120 beschreibt.

Also der Text:

@fig-flights-dec120-3 zeigt die Abflug- und Ankunftsverspätungen für Dezember 2013. @fig-flights-dec120 zeigt ebenfalls diese Daten, hat aber keine

Beschriftung. Die Datei `@fig-flights-dec120-2` zeigt ebenfalls diese Daten und hat eine Beschriftung, ist aber nicht vergrößert.

Wird gedruckt:

Abbildung 5 zeigt die Abflug- und Ankunftsverspätungen für Dezember 2013. Abbildung 3 zeigt ebenfalls diese Daten, hat aber keine Beschriftung. Abbildung 4 zeigt ebenfalls diese Daten und hat eine Beschriftung, ist aber nicht vergrößert.

6 Aufgaben

Erstellt eine Kopie eures Berichts von letzter Woche und:

1. Ändert die *globalen Chunk*-Optionen (in der YAML) so, dass Meldungen und Code standardmäßig nicht in der Ausgabedatei ausgegeben werden.
 - Hinweis: Sie tun dies mit `execute` und `include: false`.
2. Ändert die *globalen Chunk*-Optionen (in der YAML) so, dass alle Figuren `fig-out: 6` und `fig-align: center` haben
3. Benutzt `knitr::kable()`, um die Tabellen zu drucken, die ihr gedruckt habt.
 - Füge ein `label` und `tbl-caption` hinzu
4. Ändert die *Code-Chunk-Einstellungen* der Code-Chunks, die Euren Barplot und Scatterplot erzeugt haben, so, dass:
 - der Code gedruckt wird
 - die Diagramme `Label` und `Caption` haben
5. Bezieht euch auf den Balkenplot, den ihr im Text mit `@` erstellt habt. Wenn ihr das Dokument rendert, steht da dann ‘Abbildung 1’?

Du hast keinen Bericht erstellt? Dann kopiere einfach den Code aus den auf Moodle geteilten Lösungen.

Heutige Ziele

Heute haben wir...

- lernen, wie man Code Chunk Optionen verwendet
- lernen, wie man die Größe von Abbildungen kontrolliert
- lernen, wie man Bildunterschriften hinzufügt
- lernen, wie man formatierte Tabellen druckt
- lernen, wie man Querverweise erstellt

Session Info

Hergestellt mit R version 4.3.0 (2023-04-21) (Already Tomorrow) und RStudioversion 2023.3.0.386 (Cherry Blossom).

```
sessionInfo()
```

R version 4.3.0 (2023-04-21)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Ventura 13.2.1

Matrix products: default

BLAS: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib;

locale:

[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: Europe/Berlin

tzcode source: internal

attached base packages:

[1] stats graphics grDevices utils datasets methods base

other attached packages:

[1] kableExtra_1.3.4.9000 knitr_1.42 patchwork_1.1.2
[4] here_1.0.1 lubridate_1.9.2 forcats_1.0.0
[7] stringr_1.5.0 dplyr_1.1.2 purrr_1.0.1
[10] readr_2.1.4 tidyr_1.3.0 tibble_3.2.1
[13] ggplot2_3.4.2 tidyverse_2.0.0

loaded via a namespace (and not attached):

[1] utf8_1.2.3 generics_0.1.3 xml2_1.3.4 stringi_1.7.12
[5] hms_1.1.3 digest_0.6.31 magrittr_2.0.3 evaluate_0.21
[9] grid_4.3.0 timechange_0.2.0 fastmap_1.1.1 rprojroot_2.0.3
[13] jsonlite_1.8.4 httr_1.4.6 rvest_1.0.3 fansi_1.0.4
[17] viridisLite_0.4.2 scales_1.2.1 cli_3.6.1 rlang_1.1.1
[21] crayon_1.5.2 bit64_4.0.5 munsell_0.5.0 withr_2.5.0
[25] yaml_2.3.7 tools_4.3.0 parallel_4.3.0 tzdb_0.4.0
[29] colorspace_2.1-0 webshot_0.5.4 pacman_0.5.1 vctrs_0.6.2
[33] R6_2.5.1 lifecycle_1.0.3 bit_4.0.5 vroom_1.6.3
[37] pkgconfig_2.0.3 pillar_1.9.0 gtable_0.3.3 glue_1.6.2

[41] systemfonts_1.0.4 xfun_0.39 tidyselect_1.2.0 rstudioapi_0.14
[45] farver_2.1.1 htmltools_0.5.5 svglite_2.1.1 rmarkdown_2.21
[49] labeling_0.4.2 compiler_4.3.0

Literaturverzeichnis

- Biondo, N., Soilemezidi, M., & Mancini, S. (2022). Yesterday Is History, Tomorrow Is a Mystery: An Eye-Tracking Investigation of the Processing of Past and Future Time Reference during Sentence Reading. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 48(7), 1001–1018. <https://doi.org/10.1037/xlm0001053>
- Wickham, H., Çetinkaya-Rundel, M., & Grolemund, G. (o. J.). *R for Data Science* (2. Aufl.). <https://r4ds.hadley.nz/>