Quarto 2

Cross-referencing und Code-Chunk-Optionen

Daniela Palleschi

2023-05-30

Inhaltsverzeichnis

Fragen zum Bericht

Go to menti.com and enter the code on the next screen

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

Last week you…

- created a report on eye-tracking reading data from Biondo et al. (2022)
- interpreted familiar and new plot types
- reproduced familiar plot types

histogram and density plot

- what do these plots show?
  - distribution of reaction times per accuracy level
- what do the peaks represent (e.g., mean, median, mode)?
  - the *mode* reaction time per accuracy level
- is there an (approximately) equal proportion of accurate (1) and inaccurate (0) responses? How can we tell?
  - no, there are many more accurate responses, we see this in the histogram which shows the number of observations (y-axis: count) per reaction time bin (x-axis)

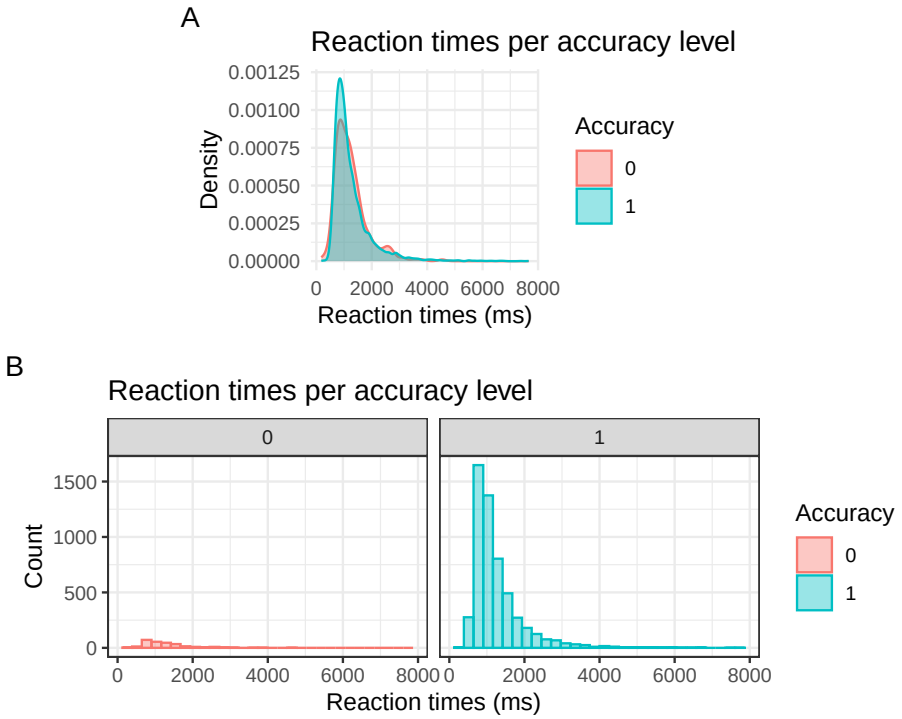

Abbildung 1: Dichte- und Histogrammdiagramme

Heutige Ziele

Today we will...

- learn how to use code chunk options
- learn how to control figure sizes
- learn how to add figure captions
- learn how to print formatted tables
- learn how to cross-reference

Lust auf mehr?

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

1 Einrichtung

1. New folder for this week
2. New Quarto document
3. Update YAML
4. Load packages

- `tidyverse`
- `knitr` (new)

```
pacman::p_load(tidyverse,
               knitr)
```

2 Code chunks

Shortcuts:

- `Cmd/Strg+Alt+I`: insert new chunk
- `Cmd/Strg+Enter`: run single line of code
- `Cmd/Strg+Shift+Enter`: run whole code chunk

- code chunks should be relatively self-contained
  - and focussed on a single task

2.1 Chunk labels

- we can give each code chunk specifications using `#|` *directly* under "'{r}
  - `#| label: simple-math` will label the chunk 'simple-math'

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

4 + 4
```
````

```
[1] 8
```

2.1.1 Advantages of using chunk labels

1. We can navigate to specific code chunks using the drop-down menu in the script editor
2. Graphics (i.e., plots) produced by chunks will have useful names that make it easier to find them later (more on this soon)

2.1.2 Chunk labels should...

- be short and informative
- contain no spaces (use `-` or `_`)
- be *unique* in a document (not repeated)

Unique chunk labels

Chunk labels must *always* be unique within a script!

- if not, you will get an error message when rendering and the document will not render

- you will get a informative error message in the 'Background jobs' pane if you have duplicate chunk labels, so ***always read the error message***!! They can be very helpful for debugging.

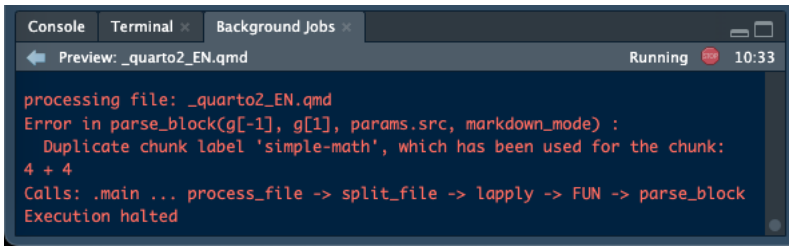

Abbildung 2: Error message when multiple code chunks have the same label `simple-math`

Aufgabe 2.1: Chunk labels

Beispiel 2.1.

1. Add a chunk label to your code chunk where you loaded packages
2. Add a code chunk using the keyboard shortcut `Cmd/Strg-Alt-I`, and add some simple math
3. Add a chunk label
4. Try out the chunk navigation bar at the bottom of the source window to jump between code chunks
5. Render

2.2 Chunk options

- chunk output can be formatted with **options** that tell R what to do with code when rendering your document
  - there are almost 60 options
  - the most important options control if your code chunk is executed when rendering and what results are printed in the output report:
- `eval: false` prevents code from being *printed* in rendered output
- `include: false` runs the code, but doesn't *show* the code or results in the final document
- `echo: false` prevents code, but not the output, from appearing in the rendered output
- `message: false` or `warning: false` prevents messages or warnings from appearing in the rendered output
- `results: hide` hides printed output; `fig-show: false` hides plots
- `error: true` renders the document even if errors are encountered

Will the following code chunk appear in the rendered output? Will the code be run?

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

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

Will the following code chunk appear in the rendered output? Will the code be run?

````
```{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()
```
````

The following table summarizes which types of output each option suppresses:

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

Option	Run code	Show code	Output	Plots	Messages	Warnings

- for the rest of the course, we will only use `eval`, `echo`, `include`, and `message`

2.3 Global options

- the chunk options just mentioned can also be set *globally* for your whole document by adding them to your YAML under `excute:`

```
title: "My report"
execute:
  echo: false
```

- and then subsequent code chunks can override the global setting in a case-by-case basis

Aufgabe 2.2: Chunk options

Beispiel 2.2.

1. Add a new code chunk
2. Give it a label
3. Change the `eval: false` and `echo: true`
4. Render

3 Figures

3.1 Figure label

- `label`s for code chunks that *print* a figure need to start with `fig-`
  - the figure will then have a number when printed

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

fig_flights
```
```

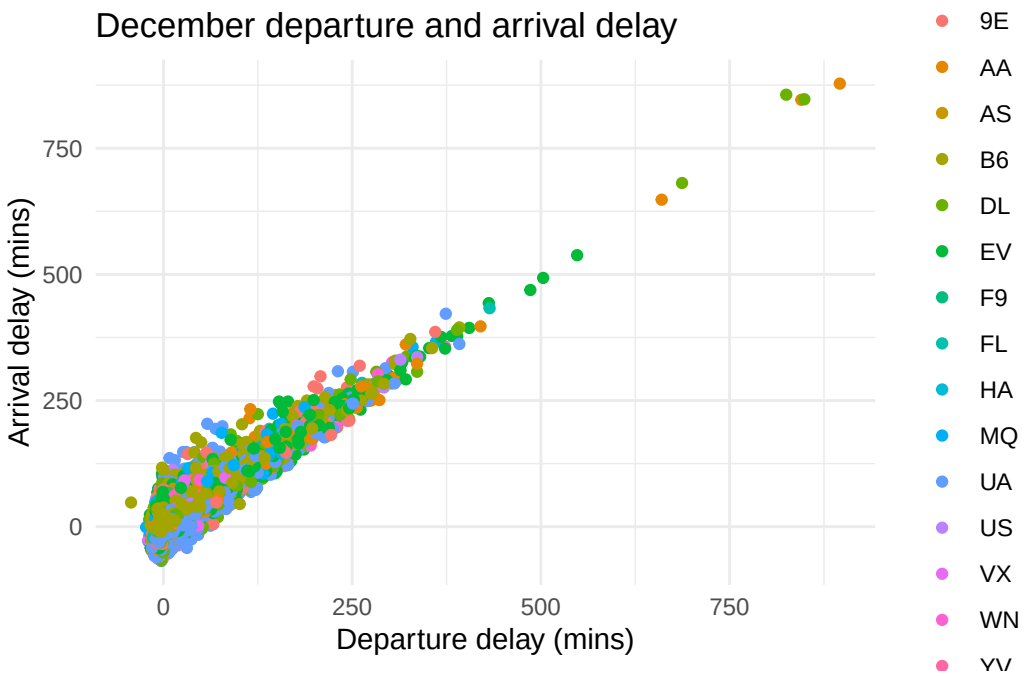

Abbildung 3: **?(caption)**

3.2 Figure caption

- `fig-cap:` adds a figure caption which will appear in the rendered document
  - always wrap the caption with quotation marks! `fig-cap: "..."`

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

fig_flights
```
````

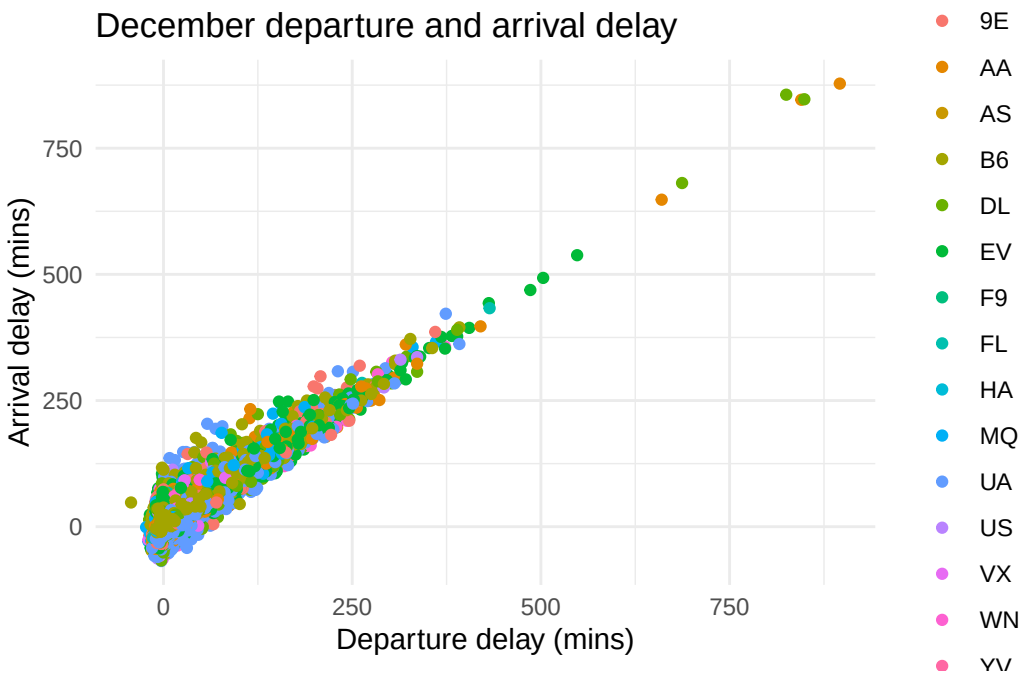

Abbildung 4: Departure delay by arrival delay for December 2013. Airline is indicated via point colour.

3.3 Figure sizing

- a big challenge of graphics in Quarto is getting the figures the right size and shape
- five main chunk options that can be helpful:
  - `fig-width`: sets the width of the figure in inches (e.g., `fig-width = 4`)
  - `fig-height`: sets the height of the figure in inches (e.g., `fig-height = 4`)
  - `fig-asp`: sets the aspect-ratio of your figure (if you set only height or width; e.g., `fig-asp = 0.618`)
  - `out-width`: sets the width of the figure in percentage to line width (e.g., `out-width = "70%"`)
  - `out-height`: sets the height of the figure in percentage to line width (e.g., `out-height = "30%"`)
  - `fig-align: centre` centres the figure on the output page

```
```{r}
#| label: fig-flights-dec120-3
```
```

```
#| fig-cap: "Departure delay by arrival delay for December 2013. Airline is indicated via
#| out-width: "70%"
#| fig-asp: .618
#| fig-align: center
#| output-location: fragment

fig_flights
```

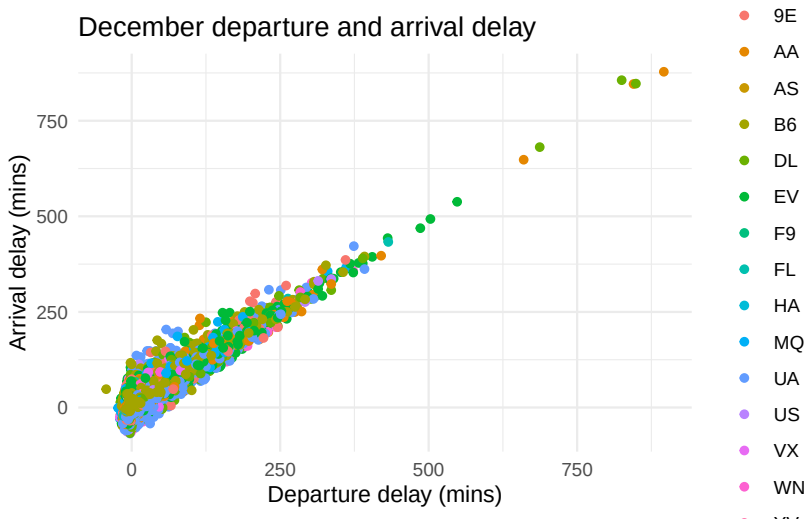

Abbildung 5: Departure delay by arrival delay for December 2013. Airline is indicated via point colour.

4 Tables

- we can print tables as we see them in the console
- we can also add further formatting using the `kable()` function from the `knitr` package

```
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
```

```r
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

4.0.1 Table captions

- we can also add a caption to the table using by adding a `label` and `tbl-cap`

````
```{r}
#| output-location: fragment
#| label: tbl-flights
#| tbl-cap: "A table made with `knitr`. The first 6 rows of the first 5 columns from the f
df_flights %>%
  select(1:5) %>%
  head() %>%
  knitr::kable(
  )
```
````

Tabelle 3: A table made with `knitr`. The first 6 rows of the first 5 columns from the flights.csv dataset are printed.

year	month	day	dep_time	sched_dep_time
2013	1	1	517	515

year	month	day	dep_time	sched_dep_time
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

4.0.2 Column names

- lastly, let's fix the column names

````
```{r}
#| output-location: fragment
#| label: tbl-flights2
#| tbl-cap: "A table made with `knitr`. The first 6 rows of the first 5 columns from the f
df_flights %>%
  select(1:5) %>%
  head() %>%
  knitr::kable(
    col.names = c("Year", "Month", "Day", "Dep. Time", "Sched. Dep. Time")
  )
```
````

Tabelle 4: A table made with `knitr`. The first 6 rows of the first 5 columns from the flights.csv dataset are printed.

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

5 Cross-referencing

- we can also refer to plots or tables in-text by typing `@` followed by the `label`
  - e.g., This is some text describing `@fig-flights-dec120`.

So the text:

> `@fig-flights-dec120-3` shows the departure and arrive delays for December 2013. `@fig-flights-dec120` also shows this data, but doesn't have a caption. `@fig-flights-dec120-2` also shows this data, and does have a caption, but is not sized.

Will print:

> Abbildung 5 shows the departure and arrive delays for December 2013. Abbildung 3 also shows this data, but doesn't have a caption. Abbildung 4 also shows this data, and does have a caption, but is not sized.

6 Aufgaben

Create a copy of your report from last week, and:

1. Change the ***global chunk*** options (in the YAML) so that messages and code are not printed in the output file by default.
    - hint: you do this with `execute` and `include: false`
2. Change the ***global chunk*** options (in the YAML) so that all figures have `fig-out: 6` and `fig-align: center`
3. Use `knitr::kable()` to print tables of whichever summary you printed.
    - add a `label` and `tbl-caption`
4. Change the ***code-chunk settings*** code chunks that produced your barplot and scatterplot, so that:
    - the code is printed
    - the plots have `label`s and `caption`s
5. Refer to the barplot you created in-text using `@`. When you render the document, does it say 'Abbildung 1'?

Didn't do a report? Then just copy the code from the solutions shared on Moodle.

Heutige Ziele

Heute haben wir...

- learn how to use code chunk options
- learn how to control figure sizes
- learn how to add figure captions
- learn how to print formatted tables
- learn how to cross-reference

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;  l

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] knitr_1.42      patchwork_1.1.2 here_1.0.1      lubridate_1.9.2
 [5] forcats_1.0.0   stringr_1.5.0   dplyr_1.1.2     purrr_1.0.1
 [9] readr_2.1.4     tidyr_1.3.0     tibble_3.2.1    ggplot2_3.4.2
[13] tidyverse_2.0.0

loaded via a namespace (and not attached):
```

```
 [1] utf8_1.2.3       generics_0.1.3   stringi_1.7.12   hms_1.1.3
 [5] digest_0.6.31    magrittr_2.0.3   evaluate_0.21    grid_4.3.0
 [9] timechange_0.2.0 fastmap_1.1.1    rprojroot_2.0.3  jsonlite_1.8.4
[13] fansi_1.0.4      scales_1.2.1     cli_3.6.1        rlang_1.1.1
[17] crayon_1.5.2     bit64_4.0.5      munsell_0.5.0    withr_2.5.0
[21] yaml_2.3.7       tools_4.3.0      parallel_4.3.0   tzdb_0.4.0
[25] colorspace_2.1-0 pacman_0.5.1     vctrs_0.6.2      R6_2.5.1
[29] lifecycle_1.0.3  bit_4.0.5        vroom_1.6.3      pkgconfig_2.0.3
[33] pillar_1.9.0     gtable_0.3.3     glue_1.6.2       xfun_0.39
[37] tidyselect_1.2.0 rstudioapi_0.14  farver_2.1.1     htmltools_0.5.5
[41] rmarkdown_2.21   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/