Datenvisualiserung 2

Visualisierung von Beziehungen

Daniela Palleschi

Mi. den 25.10.2023

Inhaltsverzeichnis

Wiederholung

Last week we learned...

- how to wrangle data using the `dplyr` package from the `tidyverse`
- learn to use uses the `pipe` (`|>`) to feed the result of one function into another function
- about functions that operate on rows
  - `filter()`, `arrange()`
- about functions that operate on columns
  - `rename()`, `mutate()`, `select()`, `relocate()`
- how to combine `dplyr` functions with plots from `ggplot2`

Wiederholung

- what do pipes connect? `|>`
  - `dplyr` verbs/functions
- what do pluses connect? `+`
  - `ggplot` layers

```r
data |>
  mutate(new_variable = height_cm/age) |>
  filter(new_variable > 5) |>
  ggplot() +
  aes(x = age, y = height) +
  geom_point()
```

Learning objectives

Today we will learn…

- how to plot two or more variables
  - with aesthetics and with facet grids
- how to use code chunk options
- how to save plots as files

Lesungen

Die **Pflichtlektüre** zur Vorbereitung auf dieses Thema ist Kap. 2 (Datenvisualisierung) aus Abschnitt 2.5 in Wickham et al. (2023).

Eine **ergänzende Lektüre** ist Ch. 3 (Data visualtion) in Nordmann & DeBruine (2022).

Set-up

Packages

```r
library(tidyverse)
library(patchwork)
library(ggthemes)
library(languageR)
```

- `tidyverse` family of packages
  - `ggplot2` for plots
  - `dplyr` for data wrangling
- `ggthemes` for colorblind-friendly color palettes
- `patchwork` for plot layouts
- `languageR` for linguistic datasets

ggplot theme

I set my preferred ggplot theme globally. This means that after I run this code, the plots will all use this theme.

Tabelle 1: english dataset variables of interest

variable	description
RTlexdec	Reaction times for a visual lexical decision (milliseconds)
RTnaming	Reaction times for the onset of a verbal word naming task (milliseconds)
WrittenFrequency	numeric vector with log frequency in the CELEX lexical database.
Word	a factor with 2284 words
AgeSubject	a factor with as levels the age group of the subject: young versus old.
WordCategory	a factor with as levels the word categories N (noun) and V (verb).
CV	factor specifying whether the initial phoneme of the word is a consonant (C) or a vowel (
CorrectLexdec	numeric vector with the proportion of subjects that accepted the item as a word in lexica

```
theme_set(theme_bw())
```

Data

We will use the `english` dataset from Baayen & Shafaei-Bajestan (2019).

- contains data from a lexical decision task in English
- let's back-transform the log transformed reaction times so they're in milliseconds
  - we use the `exp()` function to do this

```
df_english <-
  english |>
  mutate(RTlexdec = exp(RTlexdec),
         RTnaming = exp(RTnaming))
```

`english` dataset

Our variables of interest are:

Hypotheses

- what types of hypotheses might you have for such data?
  - our reaction time data are our *measure variables*
    - i.e., what we measure
  - all the other variables are possible *predictor variables*

    - i.e., we might predict that their value would influence our measure variables

- for example, what effect (if any) might word frequency have on lexical decision task reaction times? on naming times?
  - what about differences in reaction times between younger and older participants?
- what effect (if any) might word category have on reaction times?

1 Data visualisation

- visualising our data helps us visualise the relationship between variables in order to tell a story
- we typically visualise variables that we have a particular hypothesis for: predictor and measure variable(s)

1.1 Visualising distributions

- histograms, density plots, and bar plots for count values all visualise the *distribution* of observations
  - they tell us about how many times we observed certain values of a variable
  - we usually do this to get a feel for what our data look like
    - what is the range of our data, the mode, the overall distribution of values?

Task: visualising relationships

1. Create a plot that visualises the distribution of word written frequency.
2. Create a plot that visualises the distribution of nouns and verbs.

2 Visualising relationships

- to visualise relationships between variables, we need to have at least two variables mapped to a plot's aesthetics
- we've already done this by mapping colour or fill to a categorical variable, while mapping: + a continuous variable to the x-axis for histograms/density plots, or
  - a categorical variable to the y-axis for a barplot

Task: visualising relationships in distributions

1. Add another aesthetic to the plots you just created in order to show:
    - the distribution of WrittenFrequency scores for words with initial consonants and vowels
    - the distribution of nouns and verbs for words with initial consonants and vowels

2.1 Grouped continuous variable

- our histograms, density plots, and bar plots all shows the distribution of values of a *continuous* variable by different levels of a *categorical* variable

2.1.1 Stacked

- recall that by default these categories are "stacked" on top of each other

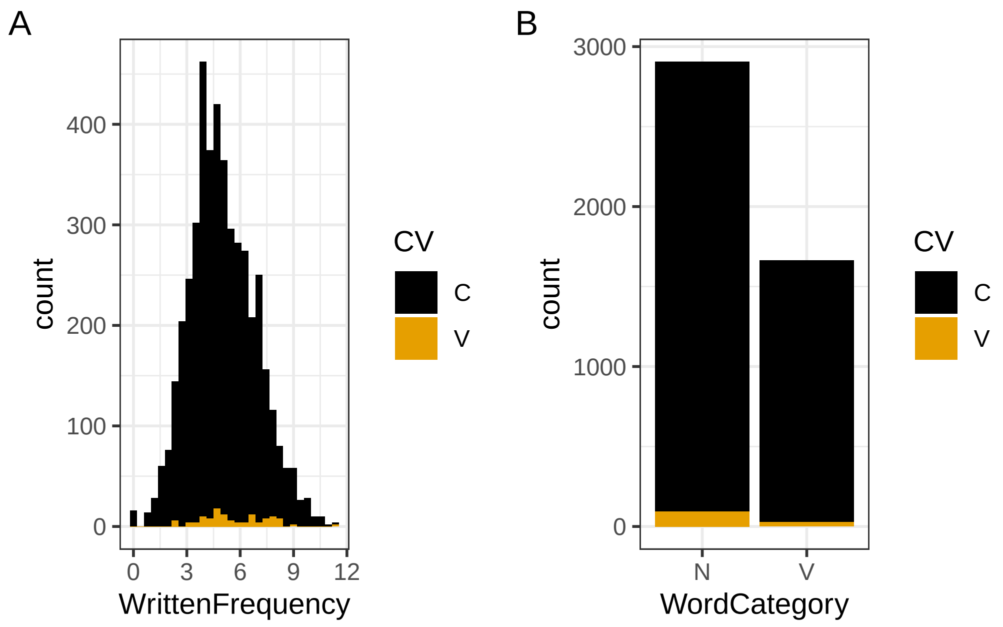


Abbildung 1: Visualising relationships in distributions

2.1.2 Dodged

- but that we can have them side-by-side by using the 'dodge' function

- I find this is more useful for bar plots

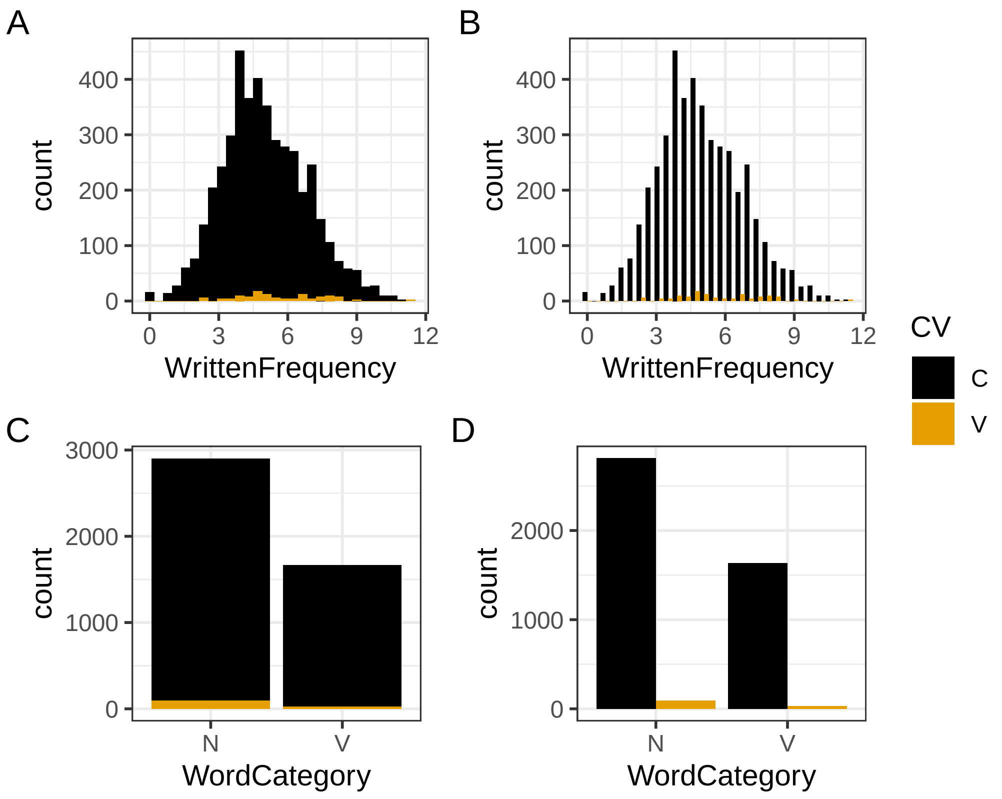
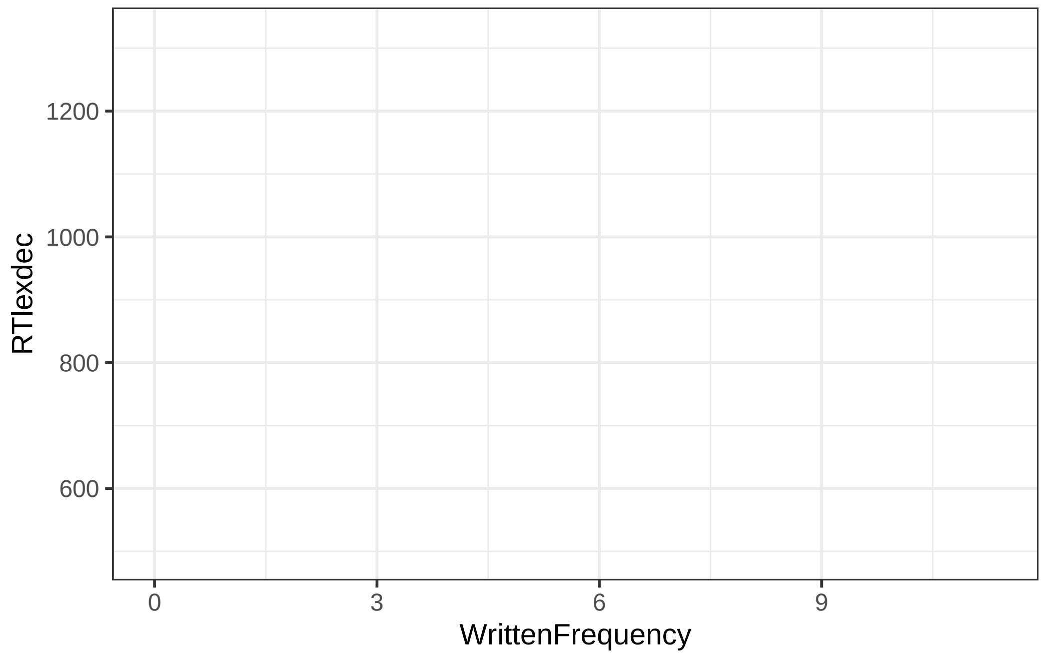


Abbildung 2: Visualising relationships in distributions

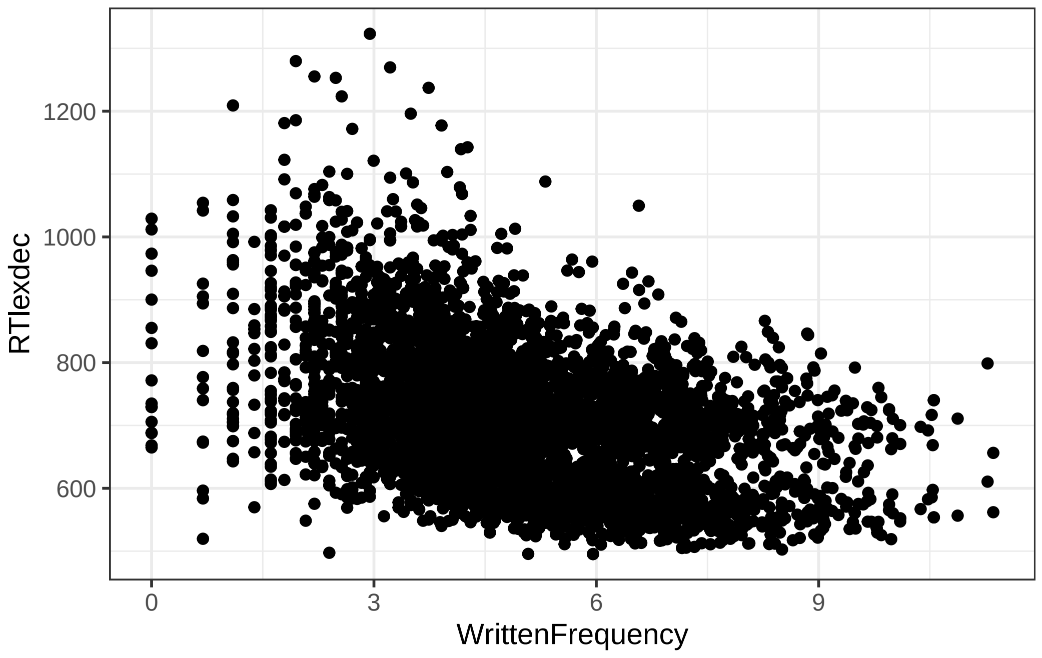
2.2 Two continuous variables

- we often want to see the effect one continuous variable has on another
- for example, in our `english` dataset we have the variables `WrittenFrequency` and `RTlexdec`
  - what type of relationship do you think these two variables will have?
  - i.e., do you think that words with lower WrittenFrequency will tend of have longer or shorter reaction times in a lexical decision task?
  - how might we visualise such a relationship?

```
# + geom_?
df_english |>
  ggplot() +
  aes(x = WrittenFrequency, y = RTlexdec)
```



```
df_english |>
  ggplot() +
  aes(x = WrittenFrequency, y = RTlexdec) +
  geom_point()
```

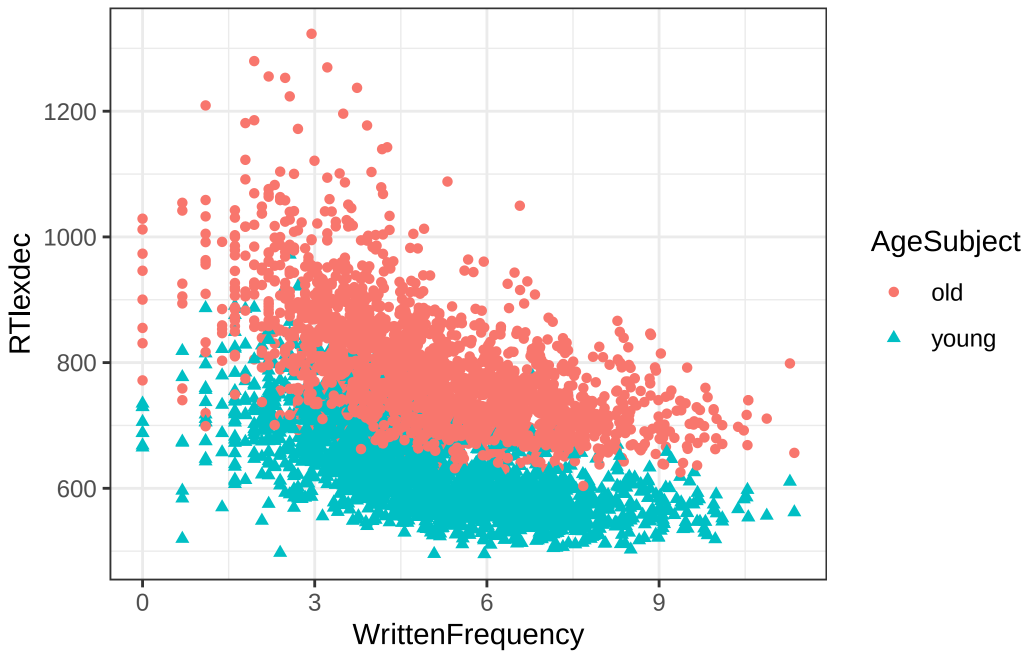


- take a moment to look at this plot and come up with an interpretation
  - what effect does written frequency of a word have on reaction times in a lexical decision task?
  - Complete the sentence: "Words with a higher word frequency elicited ________________ reaction times"
- where was there more variation in the reaction times? Where was there less variation?

2.3 Adding more variables

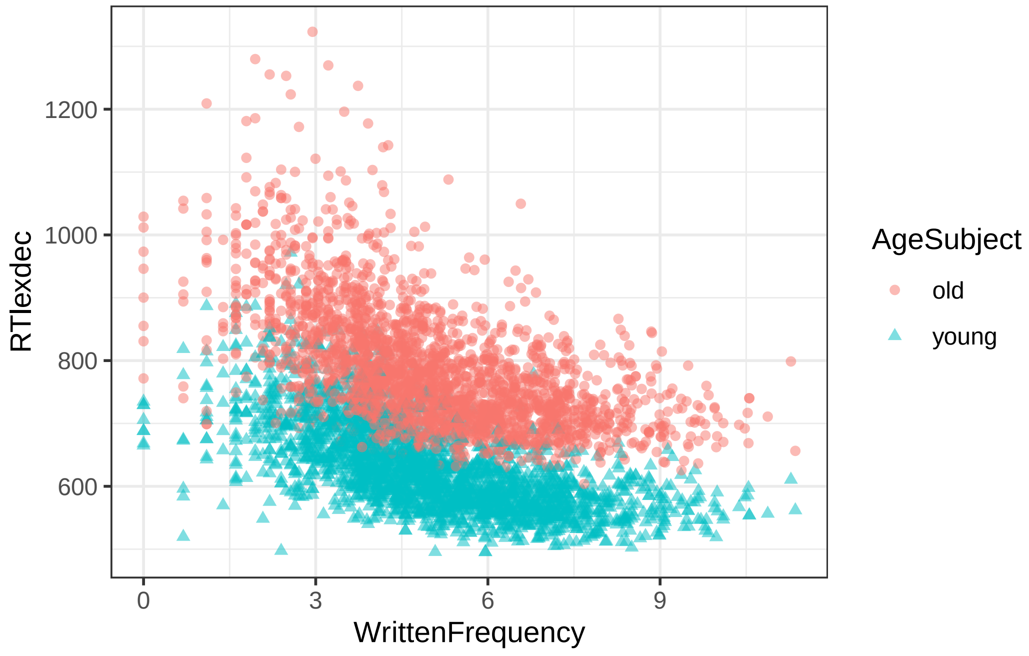
- recall that we can use other aesthetics like `fill` or `colour`
  - for `geom_point()`, it's also helpful to use `shape`

```r
df_english |>
  ggplot() +
  aes(x = WrittenFrequency, y = RTlexdec,
      colour = AgeSubject,
      shape = AgeSubject) +
  geom_point()
```



- there's a lot of overlap in the middle of the plot
  - how can we change the opacity of the points?

```r
df_english |>
  ggplot() +
  aes(x = WrittenFrequency, y = RTlexdec,
      colour = AgeSubject,
      shape = AgeSubject) +
  geom_point(alpha = .5)
```



- describe the relationship between age group and reaction times

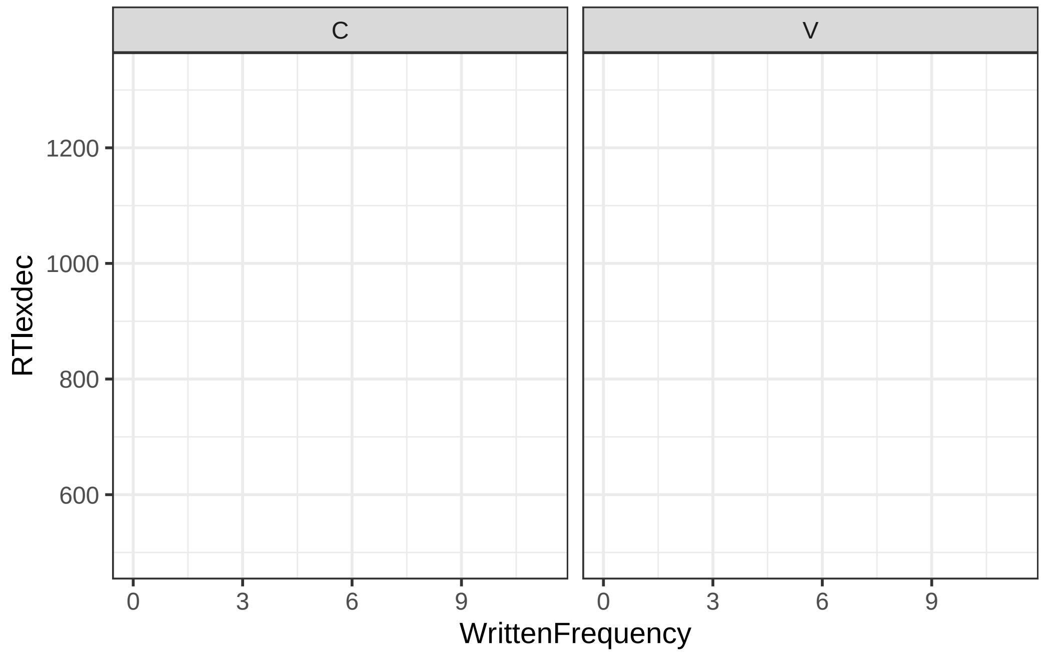
Aufgabe 2.1: `Adding another variable`

Beispiel 2.1.
How might you include a fourth variable in the plot above? Try adding `CV`. Does the plot still tell a clear story?

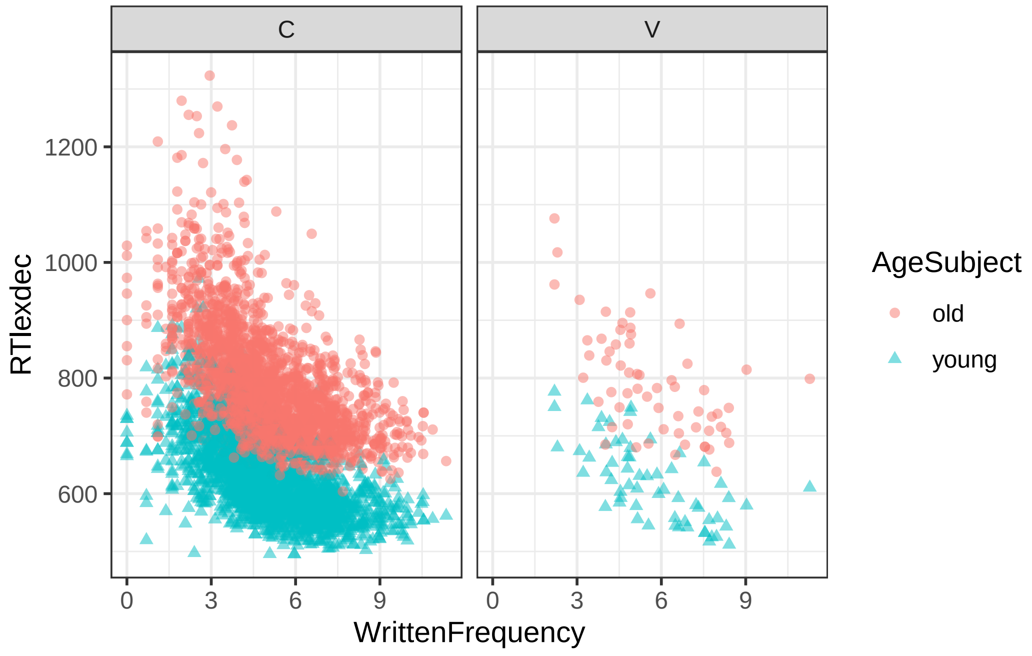
2.4 Facet grids

- in general, if you want to plot more than three variables it's a good idea to split categorical variables into *facets*
  - facets are subplots that displays subsets of the data
- we can use `facet_wrap()`, which takes a formula as its argument
  - this formula includes `~` and the name of a cateogircal variable, e.g., `~CV`

```r
# + geom_?
df_english |>
  ggplot() +
  aes(x = WrittenFrequency, y = RTlexdec,
      colour = AgeSubject,
      shape = AgeSubject) +
  facet_wrap(~CV)
```



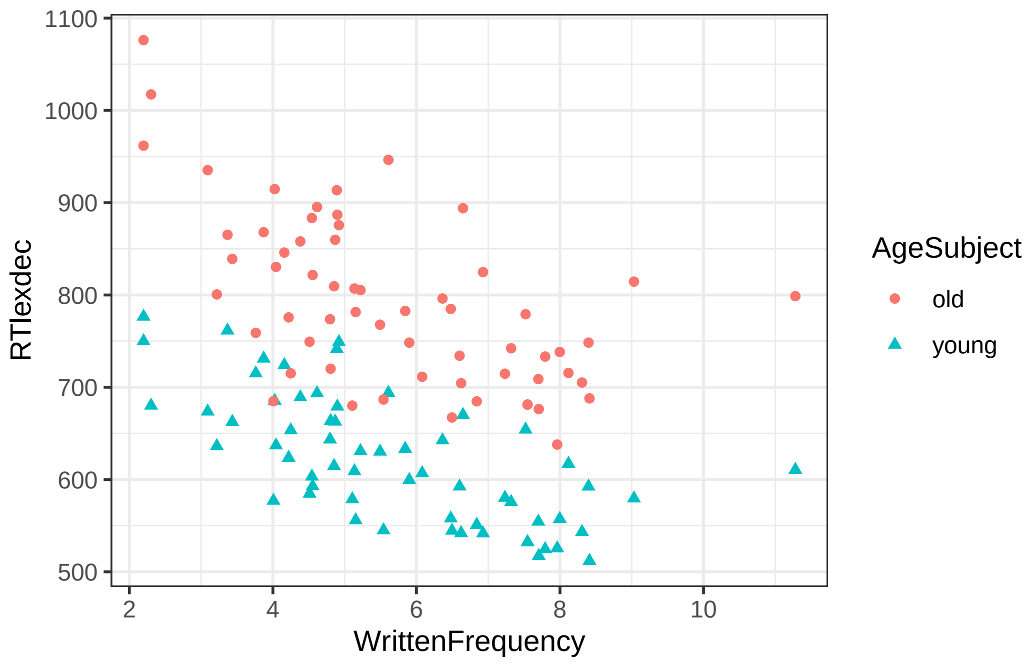
```r
df_english |>
  ggplot() +
  aes(x = WrittenFrequency, y = RTlexdec,
      colour = AgeSubject,
      shape = AgeSubject) +
  facet_wrap(~CV) +
  geom_point(alpha = .5)
```



3 Wrangled data

- We can also wrangle our data before feeding it into `ggplot()`
  - This is useful when we don't want to make permanent changes to the data, but we just want to plot for example a subset of the data
- maybe we only want to look at words that begin with a vowel, how might we do this with a `dplyr` verb?

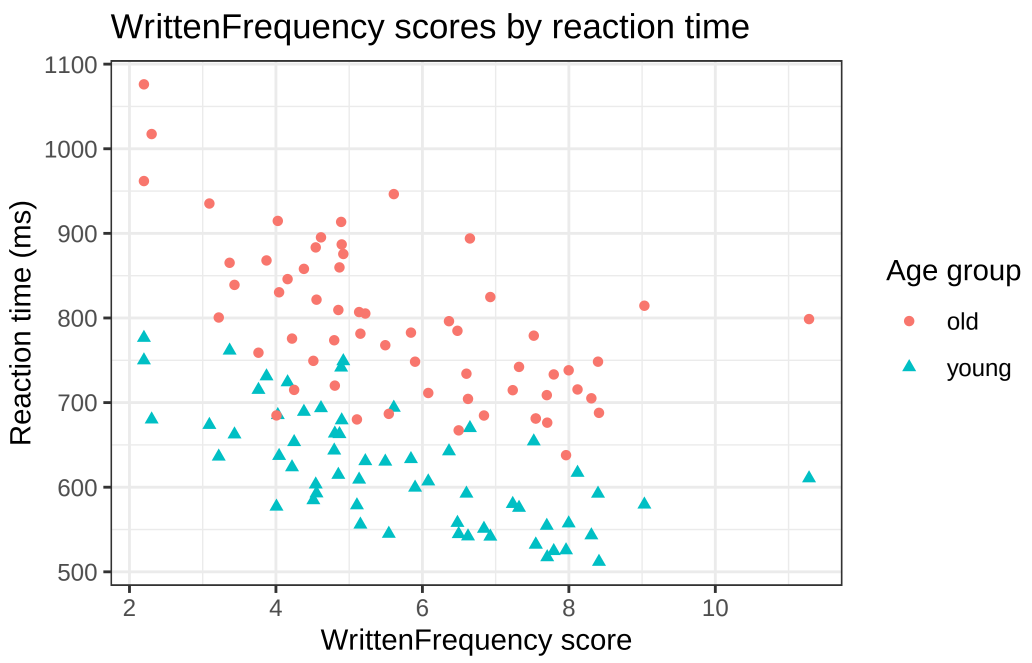
```
df_english |>
  filter(CV == "V") |>
  ggplot() +
  aes(x = WrittenFrequency, y = RTlexdec,
      colour = AgeSubject,
      shape = AgeSubject) +
  geom_point()
```



Plot annotation

The point of data visualisation is to communicate something about your data. In order to do this, we need to faciliate the reader's understanding of what our model is showing. Right now, our plots don't have a title, and the axis labels correspond to the variable names in our data, which may not be interpretable to an outsider.
All this to say, remember to give useful labels to your plots. Let's add a title, and x- and y-axis labels. There are several different ways to do this, but I find the cleanest way is to use the `labs()` ggplot-layer, which takes as its arguments `title = ""`, `x = ""`, and `y = ""`. If you also have other aesthetics (e.g., `fill` or `shape`), you can add labels to those to make sure your legend also has a reader-friendly title.

```r
df_english |>
  filter(CV == "V") |>
  ggplot() +
  aes(x = WrittenFrequency, y = RTlexdec,
      colour = AgeSubject,
      shape = AgeSubject) +
  labs(title = "WrittenFrequency scores by reaction time",
       x = "WrittenFrequency score",
       y = "Reaction time (ms)",
       colour = "Age group",
       shape = "Age group") +
  geom_point()
```



4 Quarto code chunk settings

- long code chunks can lead to very messy output documents
- usually only the plot is important to the reader, not the code that produced it
- we can control the presentation and evaluation of code chunks through code chunk options
    - these begin with `#|`

Tabelle 2: Most common chunk options

option	values	function
#| echo:	true/false	should this code chunk be printed when rendering?
#| eval:	true/false	should this code chunk be run when rendering?

  - and are placed directly below ```` ```{r}``` ````

- important code chunk options:

4.1 Using code chunks

- why do we not see the result of this plot?

~~~
```{r}
#| eval: false
df_english |>
  ggplot() +
  aes(x = RTlexdec, y = RTnaming,
      colour = AgeSubject,
      shape = AgeSubject) +
  geom_point()
```
~~~

5 Saving plots

- we often want to use our plots in a document that is not created in RStudio
  - for example in a thesis or a paper written in LaTeX
- to do this we need to load in our figures as an accepted file type, such as `jpeg` or `png`
- we can do this with the `ggsave()` function
- Can you guess what types of arguments `ggsave()` needs in order to save our plots? Some are required, some are optional.

5.1 `ggsave()`

At minimum `ggsave()` takes as its arguments:

1. the name of the plot in your Environment that you wish to save
2. the file name you wish to save your plot under
    - it's a good idea to create a folder where you will save your plots, and to include the filepath in the name

5.1.1 `ggsave()` optional argumets

- some optional arguments are:
    - `width =`: how wide you want your plot to be in cm, mm, inches, or pixels
    - `height =`:
    - `dpi =`: desired resolution (numerical, or a set of strings: "retina" = 320, "print" = 300, or "screen" = 72)
    - `bg =`: background colour, e.g., "black"

Warnung

Always set code chunks that save files to your machine to `eval: false`!!! Otherwise, every time that you run your script, the file will be re-written locally.

Aufgabe 5.1: `ggsave()`

Beispiel 5.1.

1. Copy the code below into a code chunk and run it. Look at your 'Files' tab, what has changed?

~~~
```{r}
#| eval: false
ggsave(
  # required:
  "figures/04-dataviz2/fig_lexdec_rt.png",
  plot = fig_lexdec_rt,
  # optional:
  width = 2000,
  height = 1000,
  units = "px",
  scale = 1,
  dpi = "print")
```
~~~

2. Try playing with the scale and dpi. What changes?
3. Try changing the units, width, and height values. What changes?

6 Exercises

1. a. Plot dodged barplots of AgeSubject (x-axis) by CV (facets).
   b. Change your code chunk options for the last plot so that the code, but not the plot, is printed in the output.
2. a. Filter the data to include only older participants, and plot RTlexdec (x-axis) by RTnaming (y-axis). Map CV onto colour and shape.
   b. Change your code chunk options for the last plot so that the plot, but not the code, is printed in the output.
3. Save the last plot locally, and set the code chunk to *not* run during rendering.

Session Info

Hergestellt mit R version 4.3.0 (2023-04-21) (Already Tomorrow) und RStudioversion 2023.9.0.463 (Desert Sunflower).

```
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 knitr_1.44       languageR_1.5.0  ggthemes_4.2.4
 [5] patchwork_1.1.3  lubridate_1.9.2  forcats_1.0.0    stringr_1.5.0
 [9] dplyr_1.1.3      purrr_1.0.2      readr_2.1.4      tidyr_1.3.0
[13] tibble_3.2.1     ggplot2_3.4.3    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_0.6.33     magrittr_2.0.3    evaluate_0.21
 [9] grid_4.3.0        timechange_0.2.0  fastmap_1.1.1     jsonlite_1.8.7
[13] httr_1.4.6        rvest_1.0.3       fansi_1.0.4       viridisLite_0.4.2
[17] scales_1.2.1      cli_3.6.1         rlang_1.1.1       munsell_0.5.0
[21] withr_2.5.0       yaml_2.3.7        tools_4.3.0       tzdb_0.4.0
[25] colorspace_2.1-0  webshot_0.5.4     pacman_0.5.1      vctrs_0.6.3
[29] R6_2.5.1          lifecycle_1.0.3   pkgconfig_2.0.3   pillar_1.9.0
[33] gtable_0.3.4      glue_1.6.2        systemfonts_1.0.4 xfun_0.39
[37] tidyselect_1.2.0  rstudioapi_0.14   farver_2.1.1      htmltools_0.5.5
[41] labeling_0.4.3    rmarkdown_2.22    svglite_2.1.1     compiler_4.3.0
```

Literaturverzeichnis

Baayen, R. H., & Shafaei-Bajestan, E. (2019). *languageR: Analyzing Linguistic Data: A Practical Introduction to Statistics*. https://CRAN.R-project.org/package=languageR

Nordmann, E., & DeBruine, L. (2022). *Applied Data Skills*. Zenodo. https://doi.org/10.5281/zenodo.6365078

Wickham, H., Çetinkaya-Rundel, M., & Grolemund, G. (2023). *R for Data Science* (2. Aufl.).