Datenvisualiserung 1

Distributions

Daniela Palleschi

Clam00 3amte Wed 000, 25, 2023

Table of contents

Wiederholung

Last week we...

- installed R and RStudio
- created our first R script
- did simple arithmetic with objects and vectors

Wiederholung

```
x <- c(1,2,3)
y <- sum(1,2,3)
```

- what do the vectors `x` and `y` contain?
- The object `x` contains `1, 2, 3`.
- The object `y` contains `6`.

Heutige Ziele

Today we will learn...

- what dataframes are
- the difference between categorical and continuous data
- how to produce plots with `ggplot`
- choose the right plot for our data

Lust auf mehr?

- Chapter 2 (Data Visualisation) in Wickham et al. (2023), up until section 2.4
- Chapter 3 (Data Visualisation) in Nordmann & DeBruine (2022)

Vorbereitung

In your RProject folder…

- create a new folder called `moodle`
  - download the Moodle materials from today and save them there
- create a new folder in `notes` called `02-datenviz1`
- open a new `.R` script
  - save it in the new folder

0.0.1 Packages

- Pakete (installiert und) ladt
  - `tidyverse`
  - `languageR`
  - `ggthemes`
  - `patchwork`

```
# in the CONSOLE: install packages if needed
install.packages("tidyverse")
install.packages("languageR")
install.packages("ggthemes") # for customising our plots
install.packages("patchwork") # plot layouts
```

```
# Pakete laden
library(tidyverse)
library(languageR)
library(ggthemes)
library(patchwork)
```

1 Data frames

- data frames are a collection of variables, where
  - each variable is one column
  - each row is a single observation/data point
  - each cell in a row is linked
- data frames are just like spreadsheets, but are rectangular
- different words for data frames:
  - data frame
  - dataset
  - tibble (in the `tidyverse`)

1.1 Talking about datasets

- when we talk about our data, we use certan words to refer to different parts:
- a **variable**: a quantity, quality, or property you can measure
- a **value**: the state of a variable when you measure it
- an **observation**: set of measurements made under similar conditions
  - will contain several values each associated with a variable
  - an observation for a single variable is sometimes called a *data point*
- **tabular data** is a set of values, each associated with a variable and an observation
  - tabular data is *tidy* if each value is placed in its own *cell*, each variable in its own column, and each observation in its own row

1.2 Categorical and continuous variables

- how we visual the distribution of a variable depends on what type of data it represents: *categorical* or *numerical*
- a variable is *categorical* if it can take a small set of values that can be grouped together
- e.g., old/young, short/tall, grammatical/ungrammatical, L1/L2-speaker
- a variable is *numerical* (i.e., quantitative) if it can take on a wide range of numerical values
  - and it would make sense to add, subtract, compute the mean, etc.
  - can be *continuous* (decimal points make sense, e.g., 1.5cm)

  - or *discrete* (decimal points do *not* make sense, e.g., 1.5 children doesn't make sense)
  - age, height, reaction times, format frequencies
- we produce different plots depending on what type of variables we want to visualise

2 Lexical Decision Task

- our first dataset contains data from a lexical decision task (LDT)
- in the LDT, participants press a button to indicate whether a word is a real word or a pseudoword

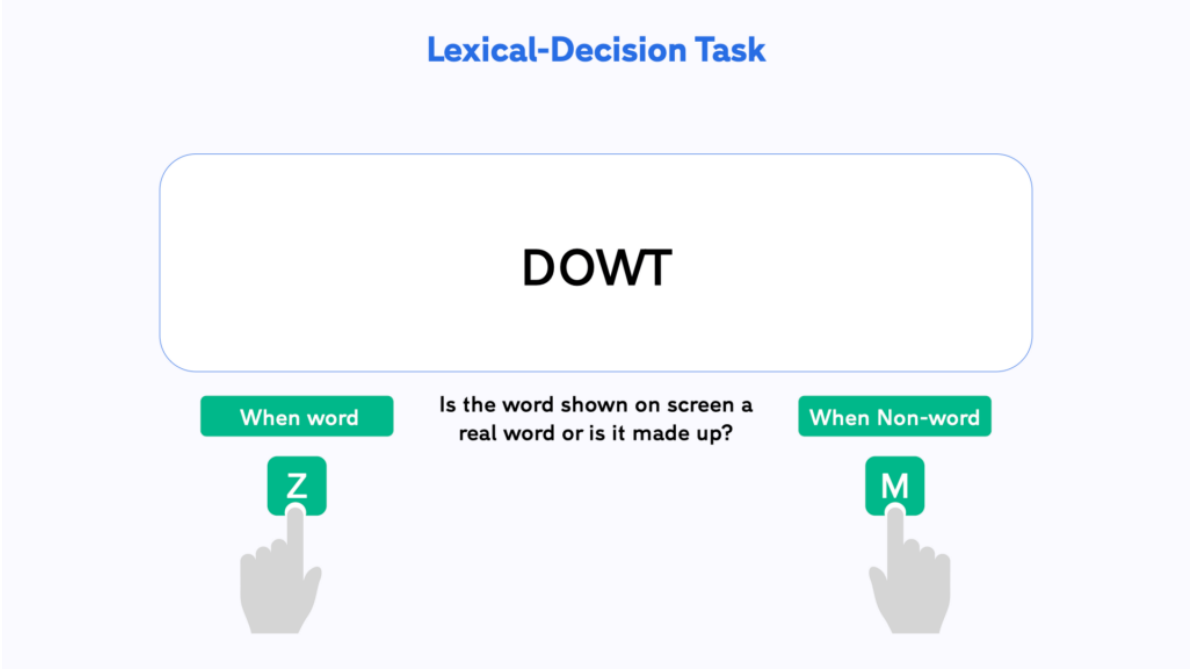
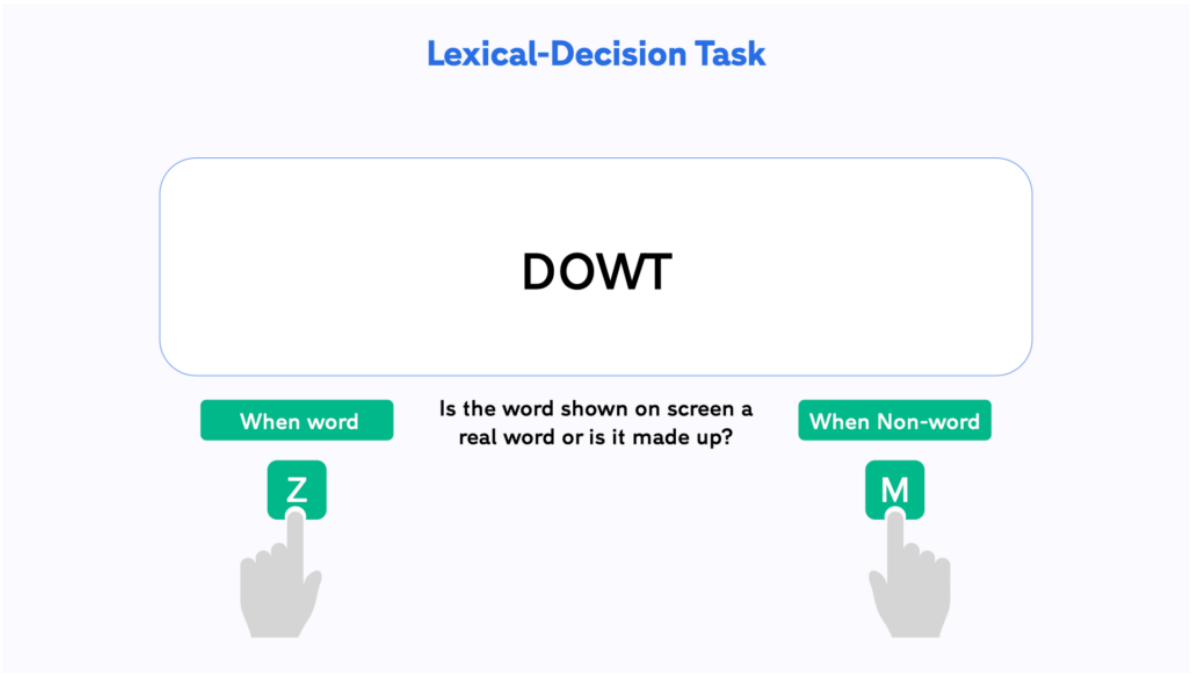


Figure 1: Source: https://www.testable.org/wp-content/uploads/2022/11/Lexical_decision_task-1024x576.png



2.1 LDT variables

- common variables collected in a lexical decision task experiment are:
  - reaction time
  - accuracy (correct/incorrect)
  - word category (e.g., real/pseudo, noun/verb)
  - word frequency
- additional variables that might be collected could be:
  - participant demographics (e.g., age, L1/L2, gender)

3 `lexdec` dataset

- `languageR` is a companion package for the textbook Baayen (2008)
  - contains linguistic datasets, e.g., `lexdec`
- the `lexdec` dataset contains data for a lexical decision task in English
  - we will be working with variables such as reaction times and accuracy

3.1 `lexdec` variables

- a list of some of the variables is included in Table 1

Table 1: Data dictionary for `df_lexdec`: Lexical decision latencies elicited from 21 subjects for 79 English concrete nouns, with variables linked to subject or word.

variable	description
Subject	a factor for the subjects
RT	a numeric vector for reaction times in milliseconds
Trial	a numeric vector for the rank of the trial in the experimental list.
Sex	a factor with levels F (female) and M (male).
NativeLanguage	a factor with levels English and Other, distinguishing between native and nonnative speake

3.2 LDT research questions

- before we conduct an experiment, we have research questions that we want to answer with the data
  - today we'll address the following question:
    - do the reaction times differ between native and non-native speakers?

3.3 Load the data

- our data is available in the `lanaugeR` package we've already loaded
  - to print the data, just type the name of the dataset and run it
- below we only see a few variables, but you should see more in your console

```
lexdec
```

```
  Subject       RT Trial Sex NativeLanguage Correct PrevType PrevCorrect
1      A1 6.340359    23   F        English correct     word     correct
2      A1 6.308098    27   F        English correct  nonword     correct
3      A1 6.349139    29   F        English correct  nonword     correct
4      A1 6.186209    30   F        English correct     word     correct
5      A1 6.025866    32   F        English correct  nonword     correct
6      A1 6.180017    33   F        English correct     word     correct
```

- how many variables do we have? observations?

3.3.1 Save data as an object

- to save the data in our Environment, we have to assign it a name
  - let's call it `df_lexdec`, which means "data frame lexical decision"

```
df_lexdec <- lexdec
```

- now we see it in our Enrivonment
  - double-click on it to view it in the Editor pane

3.4 Relevant variables

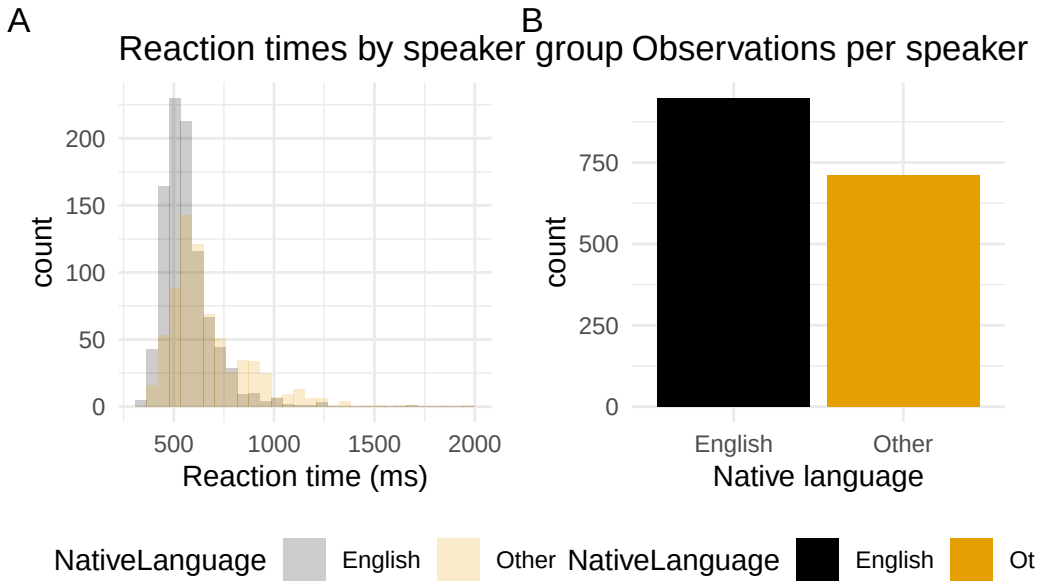
- Among the variables we have are:
  1. **Subject**: participant ID
  2. **RT**: logged reaction times
  3. **NativeLanguage**: the native language of the participant
  4. **Word**: what word was presented
  5. **Class**: if the word was animal or plant

Aufgabe 3.1: `?lexdec`

Example 3.1.
Find out what the other variables represent by running `?lexdec` in the console.

3.5 Ultimate goal

- our ultimate goal today is to produce the following visualisation of the data
  - the plot shows the distribution (`count`) of reaction times and native language of the participants



4 Creating plots with `ggplot2`

- the `tidyverse` is a collection of packages that facilitate data tidying and visualisation
  - when we load the `tidyverse`, this collection of packages is automatically loaded
- the `ggplot2` package is a `tidyverse` package that builds plots in layers

`ggplot2` Schichten

4.1 Layer 1: empty canvas

- the first layer with the function `ggplot()` is like an empty canvas

```
ggplot(data = df_lexdec)
```

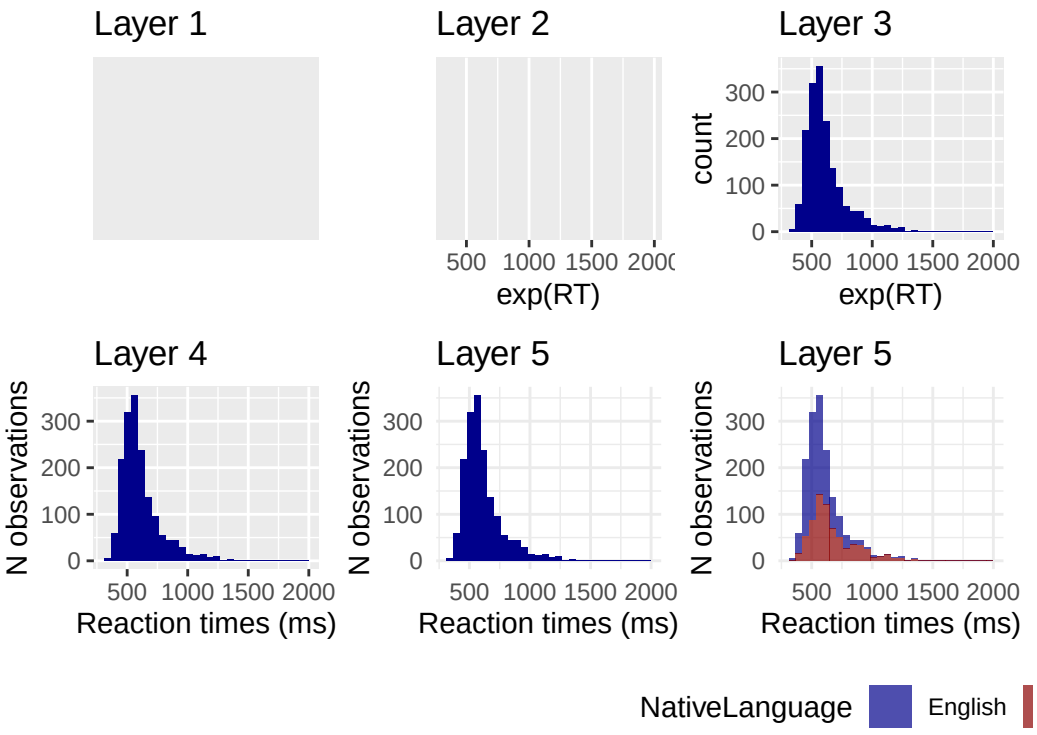
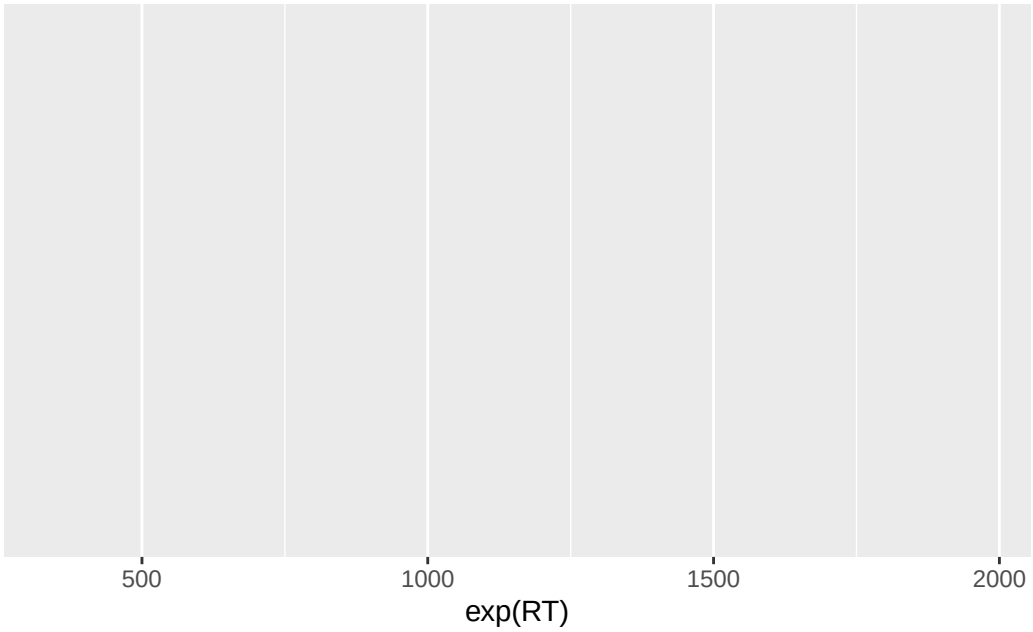


Figure 2: Example of layers in a ggplot figure

4.2 Layer 2: mapping aesthetics

- next we tell `ggplot()` how to visually represent our variables
  - we add the `+` to the end of our line of code, and on a new line of code use the function `aes()` to define our *aethetics*
- our first aesthetic maps reaction times (`RT`) on the x-axis (the bottom of the plot)
  - we wrap the logged `RT` in the `exp()` function to get RTs in milliseconds (for reasons we won't discuss)

```
ggplot(data = df_lexdec) +
  aes(x = exp(RT))
```



Aufgabe 4.1: Mapping aesthetics

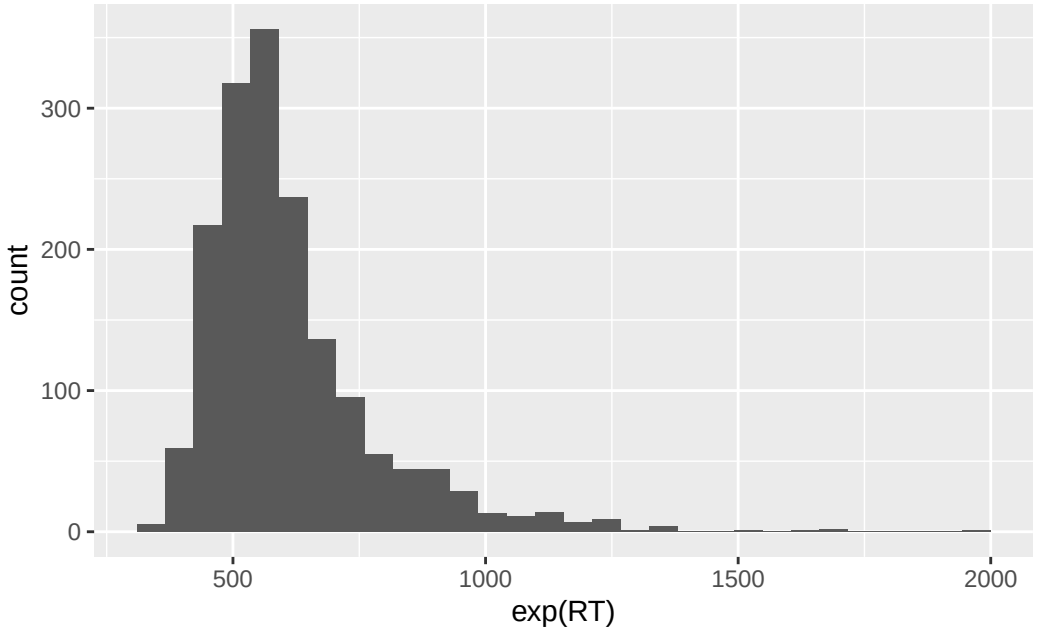
Example 4.1.

1. Produce the plot so far

4.3 Layer 3: adding observations

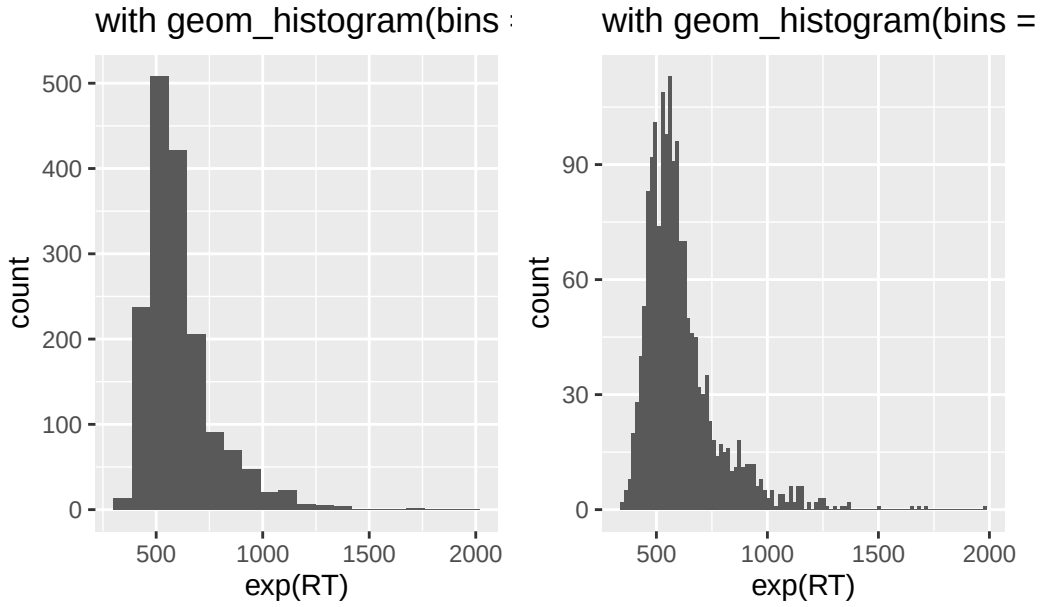
- we don't see any observations (i.e., the bars) on the plot, why not?
  - we haven't told `ggplot()` how to visualise them
- we have to define a **geom**: the *geom*etrical object that a plot uses to represent data
  - in `ggplot2`, geom functions start with `geom_`
  - we often describe plots in terms of types of geoms they use, e.g., bar charts use bar geoms (`geom_bar()`), line charts use line geoms (`geom_line()`), scatterplots use a point geom (`geom_point()`), etc.
- let's create our histogram using the geom `geom_histogram()`

```
ggplot(data = df_lexdec) +
  aes(x = exp(RT)) +
  geom_histogram()
```



> **Note**
>
> We got the following message when including `geom_point()`:
>
> `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
>
> This is just telling us about the width of our bars: each bar represents a range of possible reaction time values + `bins = 30` simply means there are 30 bars, we can change this have more or fewer bars by including e.g., `bins = 20` or `bins = 100` inside `geom_histogram()`

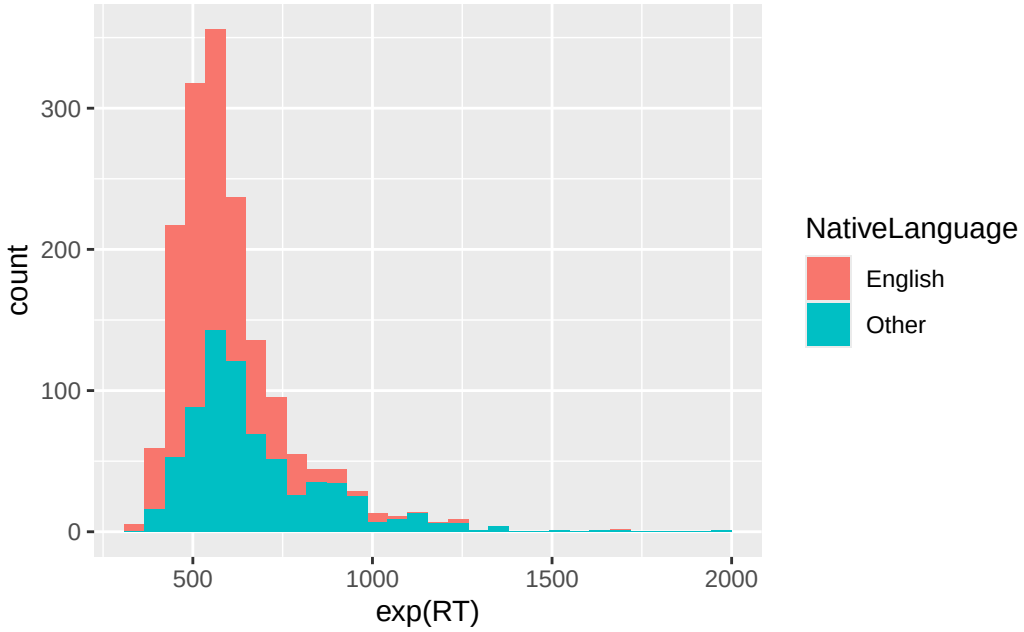
```
ggplot(
  data = df_lexdec,
  mapping = aes(x = exp(RT))
) +
  labs(title = "with geom_histogram(bins = 20)") +
  geom_histogram(bins = 20) +

  ggplot(
  data = df_lexdec,
  mapping = aes(x = exp(RT))
) +
  labs(title = "with geom_histogram(bins = 100)") +
  geom_histogram(bins = 100)
```



4.4 Adding aesthetics

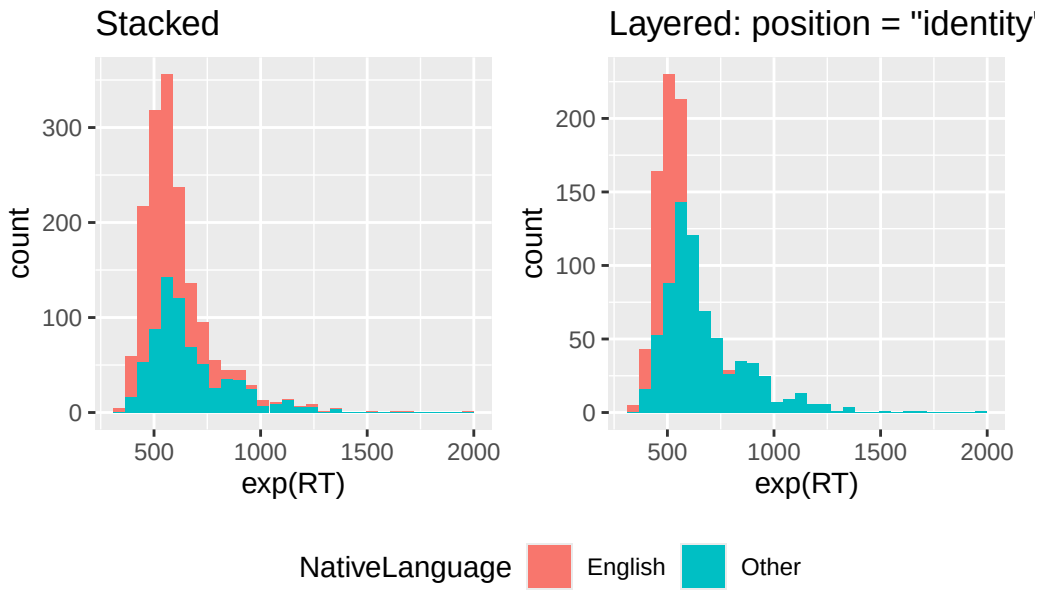
- seeing the distribution of reaction times in general is useful
  - but we usually want to compare groups
  - e.g., differences between native and non-native speakers, or between types of words
- we also have native language as a variable, how might we visualise this in our plot?

```
ggplot(
  data = df_lexdec,
  aes(x = exp(RT), fill = NativeLanguage)
) +
  geom_histogram()
```



- we see the red bars and the blue bars, but is the blue histogram layered on top of the red?
  - or are the red bars stacked in top of the blue bars?
- it's the latter
  - let's make is so that the blue histogram is layered on top of the red

```
ggplot(
  data = df_lexdec,
  aes(x = exp(RT), fill = NativeLanguage)
) +
  labs(title = "Stacked") +
  geom_histogram() +

ggplot(
  data = df_lexdec,
  aes(x = exp(RT), fill = NativeLanguage)
) +
  labs(title = "Layered: position = \"identity\"") +
  geom_histogram(position = "identity") +


  plot_layout(guides = "collect") & theme(legend.position = 'bottom')
```

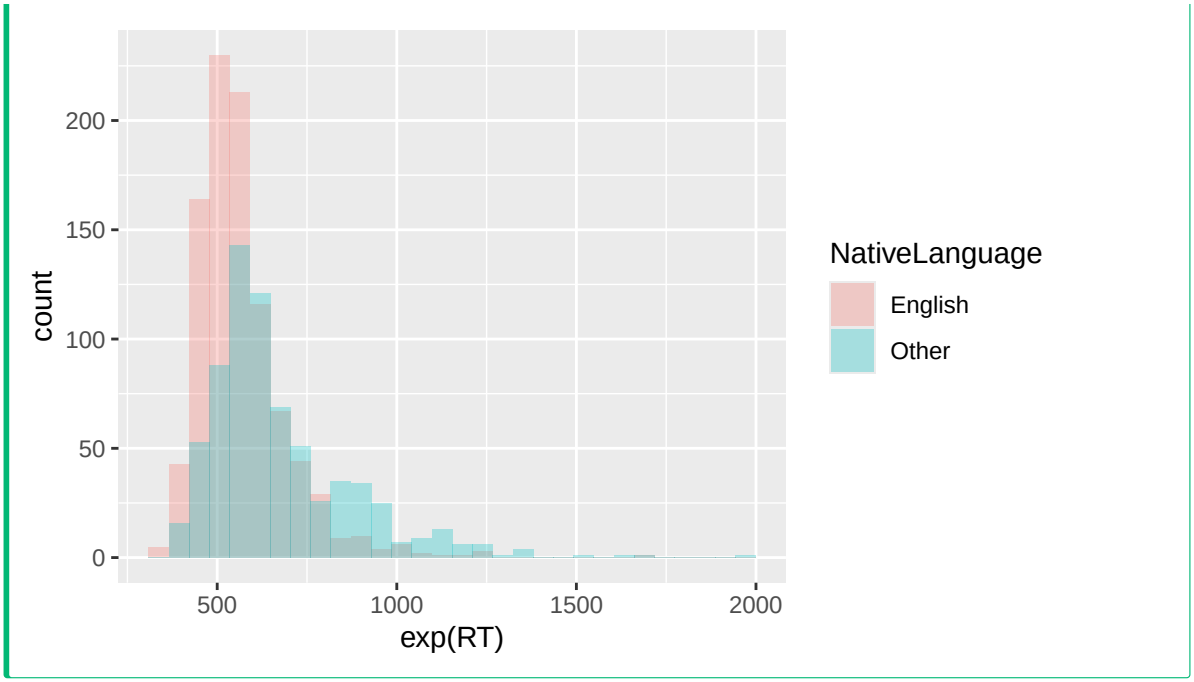


4.5 Global and local aesthetics

- in our final plot, the colour of the histograms is slightly transparent
  - we can control this by adding the argument `alpha = 0.3` to `geom_histogram()`
  - alpha takes any other value between 0 and 1

Aufgabe 4.2: Histogram transparency

Example 4.2.
Play around with the transparency of the histogram geom. Choose the alpha-value you prefer. The output should look something like this:



4.6 Customising our plot

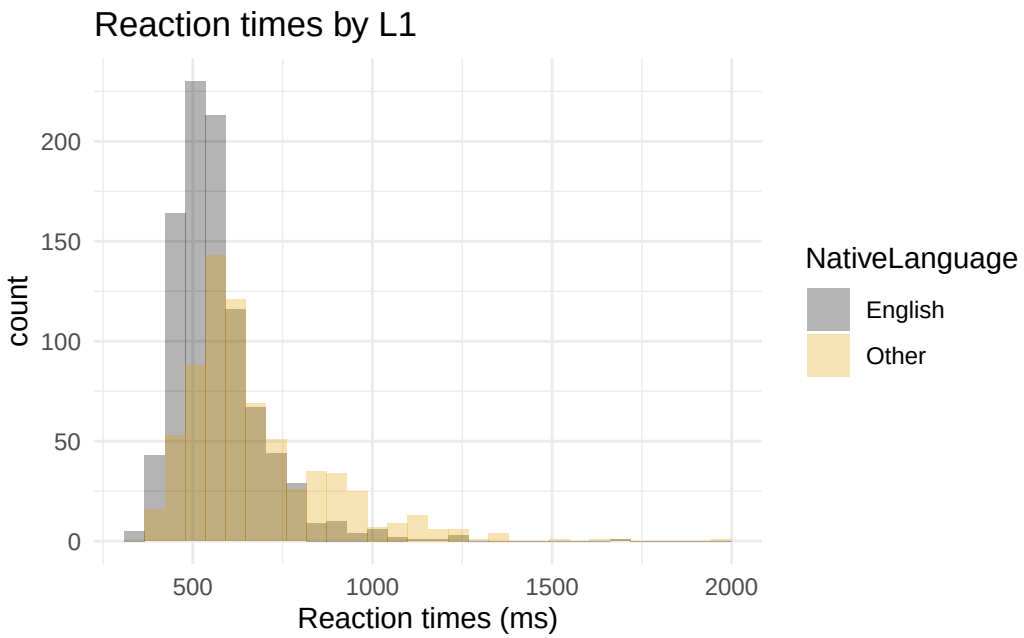
- we can improve our axis and legend labels, and also add titles using the `labs()` function
- let's also use the function `scale_fill_colorblind()` from the `ggthemes` package
  - this creates colourblind-safe colours
- we'll also use the `theme_minimal()` function from `ggplot2`; what does this do?
- try to add the following to your plot
  - change the labels accordingly
  - and add meaningful comments to the code using `#`

```
labs(title = "Plot title",
     x = "x-axis label",
     y = "y-axis label") +
  scale_fill_colourblind() +
  theme_minimal()
```

4.7 Commenting

- the code and plot should look something like this:

```
# histogram of reaction times by native language
ggplot(data = df_lexdec) +
  aes(x = exp(RT), fill = NativeLanguage) + # set aesthetics
  labs(title = "Reaction times by L1",
     x = "Reaction times (ms)") +
  geom_histogram(position = "identity", alpha = 0.3) +
  scale_fill_colorblind() + # make fill colorblind friendly
  theme_minimal() # set plot theme
```



4.8 Saving plots

- we can store plots in our Environment, just like we can store numbers and data as objects
  - you can name objects anything you want
  - but it's wise to make the name meaningful (e.g., *not* `fig1` or `xyz`)
- let's name this plot `fig_lexdec_rt`, for "figure lexical decision task reaction times"

Aufgabe 4.3: `ggplot2` review

Example 4.3.

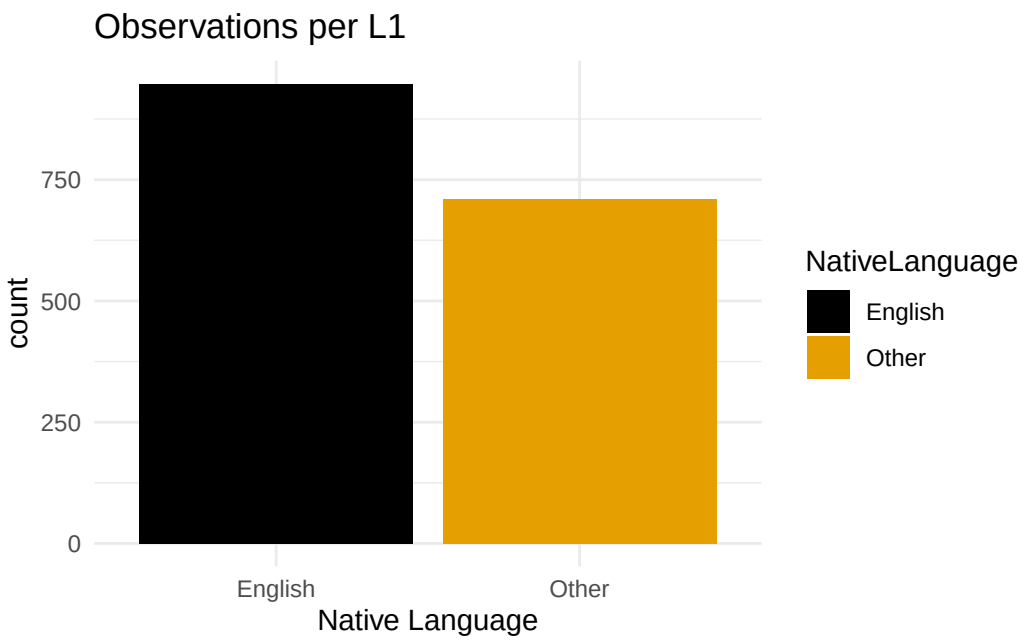
1. Save our final plot as an object called `fig_lexdec_rt`

4.9 Barplots

- copy the code for your histogram
- make the following changes to render our barplot
  - remove the name assignment (`fig_lexdec_rt`)
  - on the x-axis we want `NativeLanguage`
  - replace `geom_histogram()` with `geom_bar()`
    - remove the arguments for the histogram (not position or alpha)
  - change the labels accordingly
  - save the plot as an object with some meaningful name (e.g., `fig_lexdec_l1`)

4.10

- the plot should look something like this:

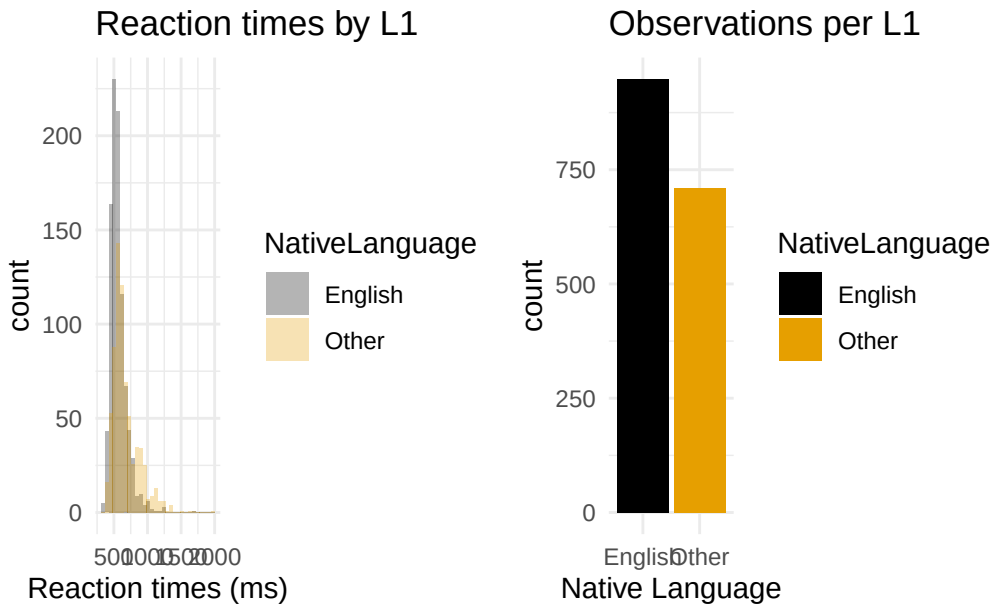


4.11 Combining plots

- one reason to save your plot as an object is so that we can call on it later
  - i.e., you can produce the plot at one point in your document, but decide to only print it in the rendered report lower down
- another reason is so that we can combine multiple plots
  - this can be done with a variety of packages
  - but let's try it with the `patchwork` package
  - use `+` to connect two plots side-by-side
  - or `/` to present them one on top of the other

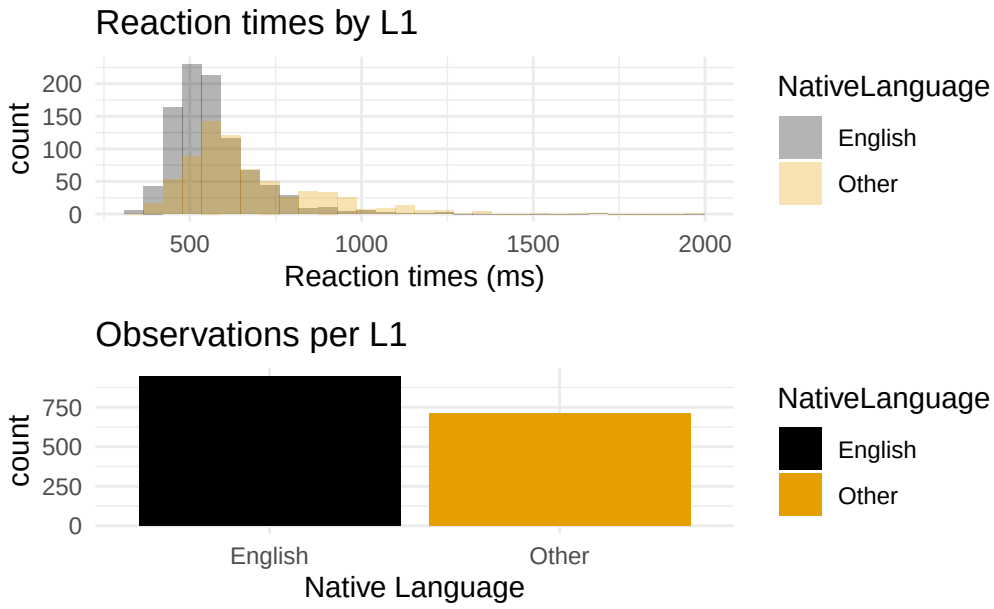
4.11.1 Combining plots with +

```
fig_lexdec_rt + fig_lexdec_l1
```



4.11.2 Combining plots with /

```
fig_lexdec_rt / fig_lexdec_l1
```



5 Deciding on a geom

- why do we use histogram for reaction times, and a barplot for native language?
- what types of variables are these?
  - reaction time is continuous
  - native language is categorical
- we use histograms to visualise distributions of *continuous* variables
- we use barplots to visualise distributions of *cateogrical* variables
- knowing what we want to visualise (e.g., distributions) and what type of variable we have (i.e., continuous, categorical) helps us decide which type of plot to produce
- often, trying to draw your plot on paper before you start in R is a good idea (I often do this, too)

6 Exercises

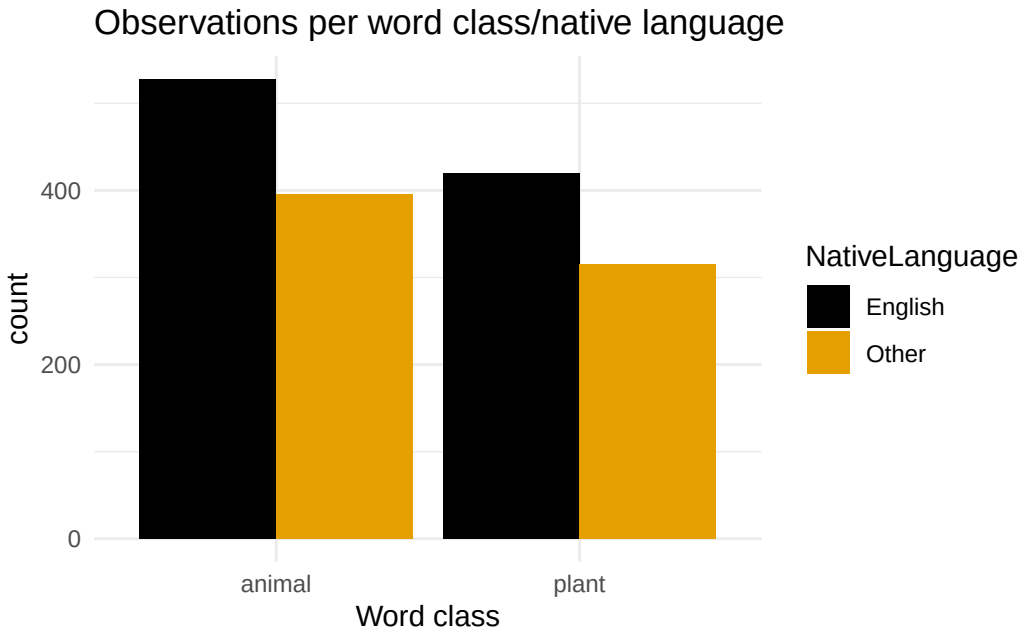
These exercises should be also be included in your script if you upload it to Moodle. Working through the class materials will prepare you for these tasks.

1. Reproduce our histogram as a *density plot* by replacing `geom_histogram()` with `geom_density()`
   - what does this type of plot show?

2. Produce a barplot that shows the number of observations per word class (hint: you'll need the variable `Class` from our dataset).
3. Print your density plot and class barplot one on top of the other using the `patchwork` package

6.1

4. Reproduce the following plots as exactly as you can (hint: you will need the `position = "dodge"` argument):



Heutige Ziele

Today we learned...

- what data frames are
- the difference between categorical and continuous data
- how to produce plots with `ggplot`
- choose the right plot for our data

Session Info

Hergestellt mit R version 4.4.0 (2024-04-24) (Puppy Cup) und RStudioversion 2023.3.0.386 (Cherry Blossom).

```
sessionInfo()
```

```
R version 4.4.0 (2024-04-24)
Platform: aarch64-apple-darwin20
Running under: macOS Ventura 13.2.1

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.4-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 datasets  utils     methods   base

other attached packages:
 [1] magick_2.8.3     kableExtra_1.4.0 knitr_1.46       patchwork_1.2.0
 [5] ggthemes_5.1.0   languageR_1.5.0  lubridate_1.9.3  forcats_1.0.0
 [9] stringr_1.5.1    dplyr_1.1.4      purrr_1.0.2      readr_2.1.5
[13] tidyr_1.3.1      tibble_3.2.1     ggplot2_3.5.1    tidyverse_2.0.0

loaded via a namespace (and not attached):
 [1] utf8_1.2.4        generics_0.1.3    renv_1.0.7        xml2_1.3.6
 [5] stringi_1.8.3     hms_1.1.3         digest_0.6.35     magrittr_2.0.3
 [9] evaluate_0.23     grid_4.4.0        timechange_0.3.0  fastmap_1.1.1
[13] rprojroot_2.0.4   jsonlite_1.8.8    tinytex_0.50      fansi_1.0.6
[17] viridisLite_0.4.2 scales_1.3.0      cli_3.6.2         rlang_1.1.3
[21] munsell_0.5.1     withr_3.0.0       yaml_2.3.8        tools_4.4.0
[25] tzdb_0.4.0        colorspace_2.1-0  here_1.0.1        pacman_0.5.1
[29] vctrs_0.6.5       R6_2.5.1          lifecycle_1.0.4   pkgconfig_2.0.3
[33] pillar_1.9.0      gtable_0.3.5      Rcpp_1.0.12       glue_1.7.0
[37] systemfonts_1.0.6 xfun_0.43         tidyselect_1.2.1  rstudioapi_0.16.0
[41] farver_2.1.1      htmltools_0.5.8.1 labeling_0.4.3    rmarkdown_2.26
```

```
[45] svglite_2.1.3     compiler_4.4.0
```

Literaturverzeichnis

Baayen, R. H. (2008). *Analyzing Linguistic Data: A Practical Introduction to Statistics using R*.

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* (2nd ed.).