Analysing quantitative data: An introduction for language researchers

Jan Vanhove
jan.vanhove@unifr.ch
https://janhove.github.io

University of Fribourg
Autumn 2025

Analysing quantitative data: An introduction for language researchers © 2025 by Jan Vanhove is licensed under Creative Commons Attribution-NonCommercial 4.0 International.
To view a copy of this license, visit https://creativecommons.org/licenses/by-nc/4.0/.
jan.vanhove@unifr.ch
https://janhove.github.io

Goals and philosophy

This script aims to provide students and researchers in the humanities and social sciences with foundational statistical knowledge. It is intended to be useful both for reading quantitative research reports and for designing and analysing your own studies. Before we dive in, I'd like to explain the guiding principles behind this script and what you can expect from it.

Current practice vs best practice At least in applied linguistics, my own research area, there is a substantial gap between how statistical analyses are actually carried out and reported, and how they ideally should be. Exactly what contributes to this gap is somewhat a matter of opinion. Here is my tuppence-worth:

- Many articles inundate readers with a flood of significance tests. Most of these are only marginally relevant to the research questions. Readers must therefore separate the relevant from the irrelevant information—a responsibility that ought to lie with the authors. I suspect that this overabundance of trivial information stems from researchers following a formulaic approach to data analysis. They often do not consider whether the calculations they perform and report actually make sense in the context of their study.

- Despite the large volume of numbers in research reports, readers often learn very little about the data themselves or the relationships between variables. The dangers of blind number crunching are illustrated in several exercises in this script.

- Outputs of statistical models are usually interpreted too eagerly in terms of the study's theories and hypotheses, without a clear understanding of what all these numbers *literally* mean. Linking results to theory is desirable, of course. But without understanding the literal meaning of the model output, there is a high risk of self-deception. Accordingly, this script includes several exercises in which readers must explain the literal meaning of numbers in model output.

- Some researchers lose sight of their primary goal—answering a research question—when applying more complex statistical methods. They may treat the application of the method itself as the goal. (Admittedly, I sometimes fall into this trap as well.)

This script will not dwell excessively on common questionable or pointless practices. However, several exercises are designed as a safeguard against them. Still, it is advisable

to familiarise yourself with these practices at some point (see Chapters 16 and 17 and the recommended reading).

Curb your enthusiasm. Many widespread questionable practices probably persist because researchers in the humanities and social sciences overestimate what statistical methods can achieve. A seemingly sophisticated statistical analysis cannot rescue poorly planned or executed data collection. Therefore, foundational knowledge of research design is often more important than purely statistical skills (see my other script, *Quantitative methodology: An introduction*). Do not expect that an elaborate statistical analysis will provide a useful answer to a poorly formulated research question.

Content. Chapter 1 explains how to properly set up the software we will use (R and RStudio) and is intended as self-study. Experience shows that students often struggle to structure datasets for easy analysis. Chapter 2 is devoted to these steps. You will also learn several useful commands to import, rearrange, and merge datasets in R.

Chapter 3 imparts the fundamentals of probability theory, which is then applied in Chapters 4, 5, and 6 on descriptive statistics, parameter estimation, and the estimation of uncertainty, respectively. Concepts such as sampling error and confidence intervals are illustrated primarily through simulations and related methods. I hope that this approach clarifies the concepts and makes the underlying assumptions more transparent than a purely mathematical explanation would. That said, this script doesn't shy away from some of the more accessible mathematical proofs. The rationale for providing these proofs is to make it clear that the procedures discussed are the result of reason and not of decree.

In Chapters 7 to 11, we examine a key tool in statistical practice: the general linear model. Most statistical procedures you will encounter in the social and humanities sciences are instances of this model or its generalisations.

While I do not deny occasional usefulness of significance tests, I consider them often overused. To prevent readers from thinking that p-values are the be-all and end-all of analysis, they aren't introduced until Chapter 12, alongside many caveats.

Chapters 18 to 20 provide recommendations for exploring more complex methods that, in my view, do not belong to the basics. Many procedures occasionally seen in social and humanities research remain entirely unmentioned. My hope is that readers who have worked through this script will be able to acquire such methods independently from other introductions.

The appendix contains explanations of common R error messages and an overview of the software versions used.

Prerequisites. I work in a Department of Multilingualism Research and Foreign Language Didactics. Students in our programmes generally have minimal experience with quantitative data analysis. This script assumes no prior knowledge in this area. It does

assume, however, that readers don't freeze up when confronted with with some mathematical notation. Most equations are implemented in software commands, which should make their meaning more transparent.

Learning how to learn. I assume a healthy dose of curiosity and initiative. I have tried to explain most of the settings in software commands. If a setting is unclear, consult the command's help page or experiment by changing it to see how the output changes.

Curiosity and initiative are essential because it is impossible to convey all relevant insights in a single script—even one of over 300 pages long. The aim is to provide you with the foundations to acquire further knowledge independently. You will find many literature recommendations throughout; approach them as a reading programme over several years. Many of the concepts discussed here are also featured on my blog, `https://janhove.github.io`.

Simulations are an extremely useful tool for learning statistics. We will analyse existing datasets and generate our own. Simulations allow you to know exactly how the data were generated and see how this affects model output. Experiment with and modify these simulations—you will learn a great deal from doing so. When learning new techniques, first apply them to simulated datasets to ensure you truly understand the output.

Script under revision. This script represents the latest iteration of a manuscript I have worked on since 2012 and is likely not the last. The previous versions were in German, but I decided to make use of the last round of revisions to translate the whole thing into English. I used ChatGPT to draft a translation for several of the chapters.

I welcome any content- or language-related feedback. Special thanks to Isabelle Udry, who pointed out numerous linguistic errors and unclear passages in one of the more recent versions, which I then promptly replaced by others.

Good luck, stay patient with yourself, and enjoy the process! :)

Contents

Goals and philosophy iii

I Basics 1

1 Self-study: Software 3

2 Working with datasets 17

3 Fundamentals of probability theory 63

4 Descriptive statistics of a univariate sample 99

II Estimates 119

5 Random samples 121

6 Estimating estimation uncertainty 131

III The general linear model 153

7 Another look at the mean 155

8 Adding a predictor 169

9 Group differences 203

10 Differences between differences 215

11 Several predictors in one model 225

IV Significance testing 253

12 The logic of significance testing 255

13 The *t*-test **281**

14 Analysis of variance **291**

15 *Calculating statistical power **309**

16 *Silly tests **315**

17 *Questionable research practices **319**

V Miscellaneous topics **323**

18 *Within-subjects experiments **325**

19 *Logistic regression **339**

20 *Recommendations for self-study and tips **355**

Appendices **359**

A Common error messages in R **359**

B Software versions **363**

Part I

Basics

Chapter 1

Self-study: Software

Our first priority is to install and configure the software we'll use throughout this course. We'll also see how this software can be used as a calculator and how we can extend its functionality.

This chapter is for self-study. If you've done all the activities in this chapter, you're ready to go. Optional sections, exercises, etc. are marked by an asterisk.

1.1 Installing and configuring R and RStudio

In order to reproduce all steps in these lecture notes, you'll need the freeware R. Other freeware for analysing quantitative data is available (e.g., Python, Julia, JASP), but R is more widely used in the humanities and the social sciences. I also happen to know R better than these alternatives.

Activity 1.1 (Installing R)**.** If you haven't installed R on your machine yet (version 4.2.0 or newer), download it from `https://www.r-project.org` and install it. ♢

I use the ♢ symbol to mark the end of an activity, exercise, remark, etc. The □ symbol is used to mark the end of a proof.

Activity 1.2 (Installing RStudio)**.** RStudio is a free interface that makes working with R considerably easier. Download the Open Source Desktop version of RStudio from `https://posit.co` and install it. ♢

Open RStudio. Your display should look like in Figure 1.1. If you only see three rather than four panes, click on `File, New file, R script`.

- In the bottom left, you see the R console. R executes any commands you enter here and also displays the output of these commands here.
- The top left pane is a text editor. Instead of entering your commands directly onto the console, you should enter them here first. This makes it easier to structure and format your commands, which in turn makes it easier to read and find errors in them. Moreover, you can save your scripts as .R files so that you can later reproduce your analyses.

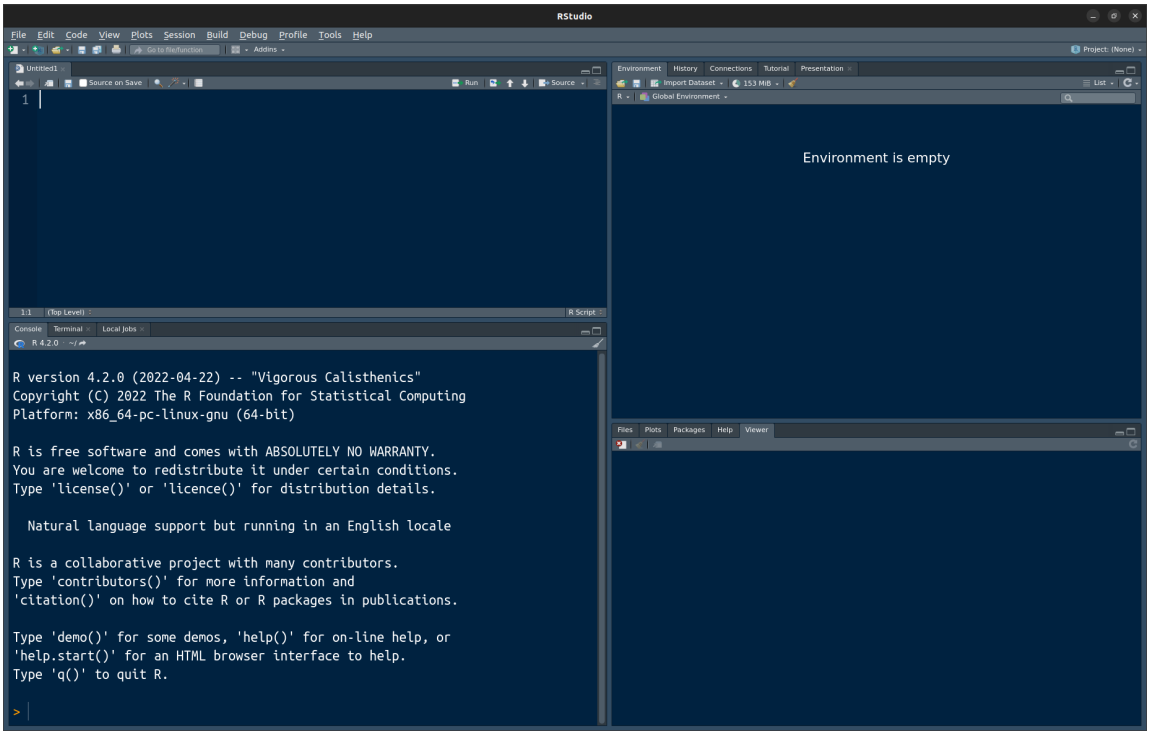


Figure 1.1: RStudio, with the R console in the bottom left, the text editor in the top left, the inventory of the working environment in the top right (currently empty), and any plots or opened help pages in the bottom right.

- In the top right panel, any objects currently in the R working environment are listed. Since we haven't read in or otherwise created any data yet, this environment is currently empty.

- If you draw a plot, you'll see it displayed in the bottom right panel. Help pages can also be displayed here, and the panel can additionally be used as a file manager (like Windows Explorer). If you delete files in the file manager, they're gone!

Activity 1.3 (Configuring RStudio)**.** In RStudio, and go to `Tools > Global options...`. In the tab `General, Basic`, make sure to untick the option 'Restore .RData into workspace at startup' and select the option 'never' for 'Save workspace to .RData on exit'. These settings help ensure that you're starting from a clean slate each time you open RStudio and prevent the results of calculations in previous sessions from messing up your analyses. Additionally, in the tab `Code, Editing`, tick the box next to 'Use native pipe operator, `|>`'. We'll encounter this pipe operator shortly. Finally, I recommend setting the default text encoding to UTF-8 under `Code, Saving`. ♢

1.2 Using R as a calculator

1.2.1 Arithmetic

R can be used as a calculator. To compute sums, products, and quotients, you can enter commands like those you find below onto the console. The results aren't part of the commands; they are displayed once you've entered the command and confirmed it by pressing ENTER. R ignores everything to the right of a hash (#), which allows us to add comments to the commands:

```
10 + 7      # addition
[1] 17
12 - 28     # subtraction
[1] -16
7 * 3.5     # multiplication
[1] 24.5
11 / 3      # usual division
[1] 3.666667
6^3         # exponentiation
[1] 216
10^(-2)     # = 1/10^2 = 1/100
[1] 0.01
4^0         # x^0 = 1; R uses the convention 0^0 = 1.
[1] 1
sqrt(16)    # square root
[1] 4
log(16, 2) # logarithm base 2 of 16, i.e., 2^4 = 16
[1] 4
log(16)     # natural logarithm of 16 (i.e., using base e =~ 2.72)
[1] 2.772589
log2(16)    # shorthand for logarithm base 2
[1] 4
log10(16)   # shorthand for logarithm base 10
[1] 1.20412
```

```
ceiling(2.3) # rounding up
[1] 3
floor(11.7)  # rounding down
[1] 11
round(2.3)   # symmetric rounding
[1] 2
round(2.7)
[1] 3
round(2.5)   # if ~.5: round to the nearest even integer
[1] 2
round(3.5)
[1] 4
abs(-3)  # absolute value
[1] 3
sign(-3) # sign
[1] -1
sign(3)
[1] 1
sign(0)
[1] 0
```

R respects the usual order of operations (e.g., multiplication before addition). To override this order, you can use round brackets or braces:

```
4 + 8 * 2
[1] 20
(4 + 8) * 2
[1] 24
{4 + 8} * 2
[1] 24
```

***Remark 1.4** (Roots)**.** The result of $\sqrt{-3}$ is `NaN` (*not a number*) as there is no real number x such that $x^2 = -3$:

```
sqrt(-3)

[1] NaN
```

In the complex numbers, however, such a number does exist. (More precisely, there exist two such numbers, namely $0 \pm \sqrt{3}\mathrm{i}$.) We won't need complex numbers in this course, so we won't explore them further.

To take arbitrary roots of non-negative number, note that $\sqrt[n]{x} = x^{1/n}$. Hence, we may compute $\sqrt[3]{216}$ as $216^{1/3}$:

```
216 ^ (1/3)

[1] 6
```

To take arbitrary roots of negative numbers (e.g., $\sqrt[3]{-8}$), we'd have to take a detour via the complex numbers, even though real solutions may exist ($(-2)^3 = -8$). ♢

Remark 1.5 (Scientific notation)**.** If we compute 3328^{27}, the result is displayed as `1.26e+95`.[1]

```
3328^27

[1] 1.255884e+95
```

This notation is known as **scientific notation** and is to be read as $1.26 \cdot 10^{95}$—that is, as 1.26 but with the decimal point moved 95 places to the right. The result of 2^{-21} is displayed as `4.8e-07`, which is to be read as $4.8 \cdot 10^{-7}$—that is, as 4.8 but with the decimal point moved seven places to the left (0.00000048).

```
2^(-21)

[1] 4.768372e-07
```

♢

Remark 1.6 (Help pages)**.** In principle, every R function has a help page. You can display this help page by typing `?function_name` at the console. It takes some practice to efficiently read a help page, so let's take a closer look at the rounding function `round()`:

```
?round # help page not shown in script
```

This help page consists of nine sections. Five of these can be found on most help pages:

1. Description: A crisp description of the function.
2. Usage: Which parameters does the function have? The function `round()` has two parameters: `x` and `digits`. If no value is specified for `digits`, the value 0 is used by default.

[1] The numbers in the running text don't always correspond exactly to the numbers in the R output since I tend to round them more aggressively.

3. Arguments: Brief specifications of the expected parameter values.
4. Details: More details about the function. For instance, the Details section on the help page for `round()` explains how and why -1.5 is rounded and why the command `round(0.15, 1)` can produce unexpected results.
5. Examples: Click on 'Run examples' to see the output generated by these examples.

The help page for the addition operator + can be displayed by typing `?"+"` at the console. ♢

1.2.2 Vectorised operations

In R, a **vector** is a specific number of numbers (or other objects) that are grouped together. Numbers and other objects can be grouped together into vector using the function `c()` (*combine*). The object so created can be given a name by means of the assignment operator `<-`; here, we'll call it `a`:

```
a <- c(-4, 2, 0.3, 2014)
```

The vector known as `a` now contains the four numbers listed. We can display the vector in its entirety by typing its name at the prompt:

```
a

[1]   -4.0    2.0    0.3 2014.0
```

Using `a[i]`, we can select the vector's i-th entry (i.e., a_i), $1 \leq i \leq n$. Note that R uses **one-based indexing**, meaning that a vector's first component is accessed using $i = 1$. This contrasts with the many other computer languages that use **zero-based indexing**, that is, they access a vector's first component using $i = 0$.

```
a[1]          # 1st component

[1] -4

a[-3]         # all components except for the 3rd one

[1]   -4    2 2014

a[c(2, 4)]    # using c() to extract several components

[1]    2 2014

a[-c(2, 4)]   # all components except for the 2nd and 4th one

[1] -4.0  0.3
```

The sum of all components of a vector a is expressed in **capital sigma notation** as

$$\sum_{i=1}^{n} a_i := a_1 + a_2 + \cdots + a_n,$$

where n is the number of components of a. It can easily be computed using `sum()`:

```
sum(a)

[1] 2012.3
```

The product of all components, i.e.,

$$\prod_{i=1}^{n} a_i := a_1 \cdot a_2 \cdot \cdots \cdot a_n,$$

can be computed using `prod()`:

```
prod(a)

[1] -4833.6
```

The length of a vector, that is, the number of components it has, can be obtained using `length()`; its maximum and minimum using `max()` and `min()`, respectively:

```
length(a)

[1] 4

max(a)

[1] 2014

min(a)

[1] -4
```

The operations from Section 1.2.1 can also be applied to vectors. In this case, they are applied componentwise:

```
2 * a

[1]   -8.0    4.0    0.6 4028.0

a - 7

[1]  -11.0   -5.0   -6.7 2007.0

sqrt(a) # warning due to square root of negative number

[1]        NaN  1.4142136  0.5477226 44.8776113

log(a, 2) # warning due to logarithm of non-positive number

[1]       NaN  1.000000 -1.736966 10.975848
```

If we define a second vector of the same length, we can run certain operations on the corresponding components:

```
b <- c(3, 8.2, -1.2, sqrt(20))
a - b

[1]   -7.000   -6.200    1.500 2009.528
```

```
a + b
[1]   -1.000   10.200   -0.900 2018.472
a / b
[1]  -1.3333333   0.2439024  -0.2500000 450.3440907
a * b
[1]  -12.000   16.400   -0.360 9006.882
a^b
[1] -6.400000e+01  2.940668e+02  4.240865e+00  5.973146e+14
```

To compute the componentwise maximum or minimum of two or more vectors, we need to use `pmax()` or `pmin()`, respectively (*p* for *parallel*):

```
pmax(a, b)
[1]    3.0    8.2    0.3 2014.0
pmin(a, b)
[1] -4.000000  2.000000 -1.200000  4.472136
```

The function `max()` and `min()` compute the overall maximum and minimum entry across all vectors:

```
max(a, b)
[1] 2014
```

Remark 1.7 (Vectors of different lengths)**.** The second vector doesn't have to have the same length as the first when running vectorised operations. But when the vectors involved have different lengths, it can be difficult to anticipate the results. By way of an example, we define a vector d that has three components and add this vector to the vector a, which has four components. (We don't name the vector `c` as there exists an R function called `c()`.)

```
d <- c(3, 8.2, -1.2)
a + d
[1]   -1.0   10.2   -0.9 2017.0
```

Since the vectors involved have different lengths, and one of the lengths isn't a multiple of the other, the fourth addition uses the fourth element of the first vector and the first element of the second vector. This is bound to cause confusion.

At least in this example, a warning pops up. However, if one of the vectors' length had been a multiple of the other's, then this warning wouldn't have appeared—despite the problem persisting!

Don't put yourself in this situation. ♢

Remark 1.8 (Sequences)**.** We've seen that we can use `c()` to define vectors. If you want to define a vector that represents a sequence, more practical alternatives are available.

```
z <- 3:12
z

 [1]  3  4  5  6  7  8  9 10 11 12

z <- 7:-1
z

[1]  7  6  5  4  3  2  1  0 -1

z <- seq(from = 1, to = 3.5, by = 0.25)
z

 [1] 1.00 1.25 1.50 1.75 2.00 2.25 2.50 2.75 3.00 3.25 3.50

z <- seq(from = 0, to = 2, length.out = 7)
z

[1] 0.0000000 0.3333333 0.6666667 1.0000000 1.3333333
[6] 1.6666667 2.0000000

z <- seq(to = 2, by = 12, length.out = 5)
z

[1] -46 -34 -22 -10   2

z <- seq_len(5)
z

[1] 1 2 3 4 5
```

Figure out what each of these commands does. ♢

1.3 *R as a programming language

We can extend the functionality of R by writing our own functions. We'll take a look at a few examples to show how this works; more examples follow in later chapters.

Example 1.9 (Scalar product)**.** Let $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{y} = (y_1, \ldots, y_n)$ both be vectors with $n \geq 1$ real numbers as their components. The so-called (standard) **scalar product** assigns to this pair of vectors a real number (i.e., a scalar, hence the name), namely

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle := \boldsymbol{x} \cdot \boldsymbol{y} := \sum_{i=1}^{n} x_i y_i = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n.$$

The standard scalar product is also known as the **dot product** due to one of the ways it is commonly represented (namely $\boldsymbol{x} \cdot \boldsymbol{y}$).

Let's write a simple R function `dot()` that takes two vectors as its arguments and outputs their dot product. To this end, we use the reserved keyword `function()` and specify that our function takes two inputs, `x` and `y`. The function computes the componentwise product of the input vectors, sums its entries, and outputs this sum:

```
dot <- function(x, y) {
  sum(x * y)
}
```

We can use the new function `dot()` as we would any other R function:

```
a <- c(4, 2, -1)
b <- c(8, -2, 2)
dot(a, b)

[1] 26
```

We can add some further features to this basic function. For instance, `dot()` should only work on numerical vectors (as opposed to on vectors of characters, or on other objects altogether) of the same length. Using *if*-clauses and the `stop()` keyword, we can check if the input to the `dot()` function fulfils these criteria.

```
dot <- function(x, y) {
  if (!is.numeric(x)) stop("First input isn't numeric.")
  if (!is.numeric(y)) stop("Second input isn't numeric.")
  if (length(x) != length(y)) stop("Inputs have different lengths.")
  sum(x * y)
}
```

If the input vectors are incompatible, `dot()` now generates an error:

```
a <- c(4, 2, -1)
b <- c(8, -2)
dot(a, b)

Error in dot(a, b):  Inputs have different lengths.
```

$\diamondsuit$

Exercise 1.10 (Euclidean norm)**.** Let $\boldsymbol{x} = (x_1, \ldots, x_n)$ be a numerical vector with $n \geq 1$ real numbers as its components. The vector's **Euclidean norm** is defined as

$$\|\boldsymbol{x}\| := \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle},$$

where $\langle \cdot, \cdot \rangle$ is the standard scalar product from the previous example. A vector's Euclidean norm tells you how far the vector is removed from the coordinate system's origin.

Write a function `norm()` that takes one numerical input vector and outputs its norm. Use the `dot()` function from the previous example when defining your function.

Hint: The norm of the vector `a` from the previous example is about 4.583. $\diamondsuit$

Example 1.11 (Decumulation)**.** Given a numerical vector $\boldsymbol{x} = (x_1, \ldots, x_n)$, we can compute the vector containing its cumulative sums, i.e.,

$$(x_1, x_1 + x_2, x_1 + x_2 + x_3, \ldots, x_1 + \cdots + x_n).$$

The built-in function `cumsum()` does just this:

```
a <- c(17, 21, -24, 8)
cumsum(a)

[1] 17 38 14 22
```

What if we want to reverse this operation, i.e., given a vector of cumulative sums $\boldsymbol{x}$, we want to reconstruct the original vector, which we'll call $\boldsymbol{y}$? Observe that $y_1 = x_1$ and $y_i = x_i - x_{i-1}$ for $i = 2, \ldots, n$. Using a ***for*-loop**, we can define a function `decum()` that reconstructs the decumulated vector like so:

```
decum <- function(x) {
  y <- vector(length = length(x))
  y[1] <- x[1]
  for (i in 2:length(x)) {
    y[i] <- x[i] - x[i - 1]
  }
  y
}
```

A quick check shows that this function does the trick:

```
decum(cumsum(a)) # same as 'a'

[1]  17  21 -24   8
```

♢

1.4 Add-on packages

There are thousands of free add-on packages that extend the functionality of R. We'll use a couple of these throughout this course.

Activity 1.12 (Installing the `tidyverse`)**.** Use the command below to install the `tidyverse` suite. This is a bundle of different add-on packages that are based on a shared philosophy and make working with datasets easier.

```
install.package("tidyverse")
```

You can find manuals, tutorials etc. of the packages included in the `tidyverse` at `https://www.tidyverse.org/`. ♢

Activity 1.13 (Installing `here`)**.** A further useful package we'll use throughout is the `here` package. Install it. ♢

Remark 1.14 (Updating packages)**.** Both R and its add-on packages are in continuous development. To update the R packages you've installed, you can run the command `update.package()`. To upgrade to a newer version of R itself, I find it easiest to just uninstall the old version and install the new one from scratch. In this case, however, you will need to reinstall any add-on packages you need. This can be quite cumbersome, which is why I only do it during semester breaks.

A word of warning, though. It occurs quite often that your R code won't work as before when you've updated your packages. For larger research projects, I recommend ticking the `Use renv with this project` option. Any packages you use in your project will then be stored along with your project. Even if you work with newer versions of these packages in other projects, the old versions will still be used for this one. This reduces the risk that your R code doesn't work a few years down the line. For more information, see `https://rstudio.github.io/renv/articles/renv.html`.

You can find an overview of the software versions used for this booklet in the appendix. Such an overview can be generated using

```
sessionInfo() # output not shown
```

♢

Remark 1.15 (Citing software and packages)**.** R and R packages are free to use and are mostly developed by other researchers. You can acknowledge their help by citing the packages you used. You can obtain a reference to R by running the command `citation()`. References to specific packages can be obtained by running the command `citation("packagename")`, e.g.,

```
citation("tidyverse") # output not shown
```

To obtain a reference to R itself, use

```
citation() # output not shown
```

♢

1.5 R projects

It's convenient to have all scripts, datasets, plots, etc., related a project in the same directory. When working in RStudio, I recommend setting up an R project for each project you're working on. Projects would include term papers, master theses, course materials, research projects, etc. You should also set up an R project for this course.

Activity 1.16 (Setting up an R project)**.** In RStudio, click on `File > New Project...`, `New Directory`, `New Project`. Navigate to somewhere on your compute where you want to create a new directory and give this directory an informative name such as `statscourse`. You will use this directory to store the data, scripts, etc. that you need for this course. It's not necessary to tick the option `Use renv with this project`.

When you're done, close RStudio. Navigate to the directory you've just created. You should find an `.Rproj` file there. Double-click it. If all is well, RStudio should fire up. Alternatively, you can use `File > Open Project...` in RStudio.

Once you've opened your project, open the `Files` tab in the bottom right corner. Use `New Folder` to create the subdirectories `data`, `figs`, `functions`, and `assignments`. The datasets we'll use should be stored in the subdirectory `data`; we'll store the plots we'll create in `figs`; `functions` will contain some custom R functions; and you can save your homework assignments in `assignments`. ♢

The datasets, custom functions, etc. that we'll use are available from this course's Github page at `https://github.com/janhove/AnalysingQuantitativeData/`.

1.6 R Markdown reports

R scripts can be compiled to reports. This is useful for ensuring that your analyses are reproducible: Only those commands that are present in the script get compiled, and not some commands that you ran but for some reason didn't include in the script. Moreover, scripts only compile if all commands can be parsed. For this reason, you should submit your homework assignments both as an script (with the `.R` extension) and as a compiled report. See the next activity for details.

Activity 1.17 (Report templates)**.** Save the `template.R` file to the `assignments` subdirectory in your R project directory. Compile it: `File > Compile Report...`. Compare the .R file with the compiled HTML report. Use this template for your homework assignments, changing dates, names, titles, etc., as necessary. ♢

Chapter 2

Working with datasets

It's often said that 80% of data analysis is getting the data in a shape in which it can be analysed. The goal of this chapter is to furnish you with the basic tools and skills to organise, transform, and query datasets in research contexts. We'll work with the `tidyverse` suite throughout.

2.1 Organising datasets

Let's say we've run a study in which speakers of German read a couple of words in Swedish and were asked to guess what these words might mean. An excerpt from the raw data might look like this, with the words having been shown to the participants in the order in which they are listed:

- Participant 1034. Woman, 51 years.
  - Word: *söka*. Translation: *Socken* (incorrect).
  - Word: *försiktig*. Translation: *vorsichtig* (correct).
  - Word: *mjölk*. Translation: *Milch* (correct).
  - Word: *behärska*. No translation provided.
  - Word: *fiende*. Translation: *finden* (incorrect).
- Participant 2384. Woman, 27 years.
  - Word: *fiende*. No translation provided.
  - Word: *behärska*. No translation provided.
  - Word: *försiktig*. Translation: *vorsichtig* (correct).
  - Word: *mjölk*. Translation: *Milch* (correct).
  - Word: *söka*. Translation: *Socke* (incorrect).
- Participant 8667. Woman, 27 years.
  - Word: *mjölk*. Translation: *Milch* (correct).

  - Word: *behärska*. No translation provided.
  - Word: *fiende*. Translation: *finden* (incorrect).
  - Word: *söka*. Translation: *suchen* (correct).
  - Word: *försiktig*. Translation: *vorsichtig* (correct).
- Participant 5901. Man, 15 years.
  - Word: *behärska*. Translation: *beherrschen* (correct).
  - Word: *mjölk*. Translation: *milch* (sic.) (correct).
  - Word: *försiktig*. Translation: *vorsichtig* (correct).
  - Word: *fiende*. Translation: *feinde* (sic.) (correct; actually *Feind*).
  - Word: *söka*. Translation: *socken* (sic.) (incorrect).

There are lots of ways in which we could represent these data in a spreadsheet. Let's look at a few rules of thumb.

2.1.1 Wide vs. long formats

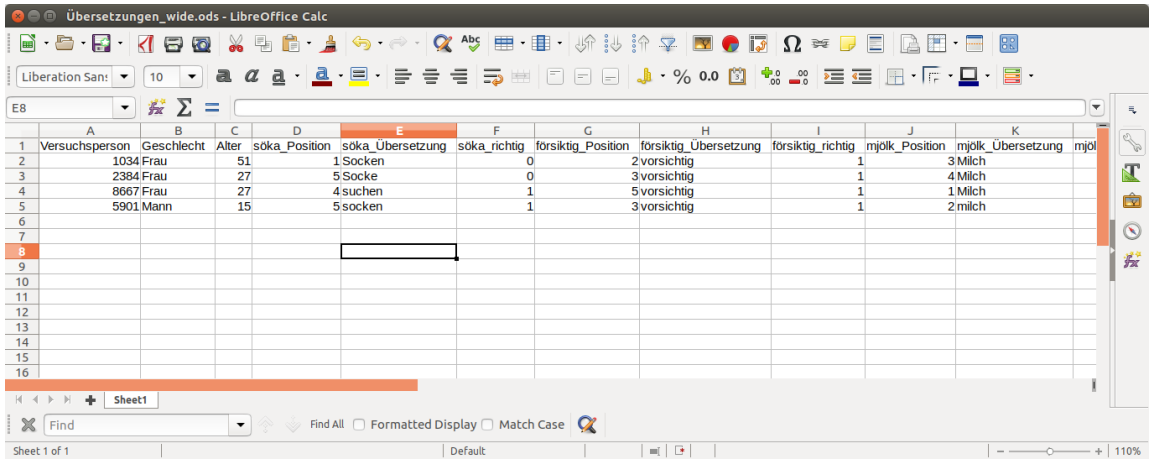
We'll concern ourselves strictly with **rectangular** datasets. These are datasets in which the information is laid out in rows and columns and in which all columns have the same length, and all rows have the same width. Examples of non-rectangular data formats are JSON and XML—see `https://json.org/example.html`.

Broadly speaking, we can organise our data in a **wide** format or in a **long** format. In a wide format, all pieces of information related to a **unit of data collection** are organised in a single row. For instance, we could think of each participant in the study as a unit of data collection, in which case we could lay out the data as in Figure 2.1. Note that the spreadsheet contains a column for each word that indicates the position in which it was presented to the participants. Alternatively, we could think of each word as a unit of data collection and organise the data like in Figure 2.2.

In a long format, all pieces of information pertaining to an **observational unit** are organised in a single row. It's difficult to precisely define what units of data collection and observational units are, and it wouldn't be too useful to have a precise definition, anyway. But in the present example, the observational units would be the individual responses, i.e., the individual translations. Figure 2.3 shows what the same data look like when organised in a long format.

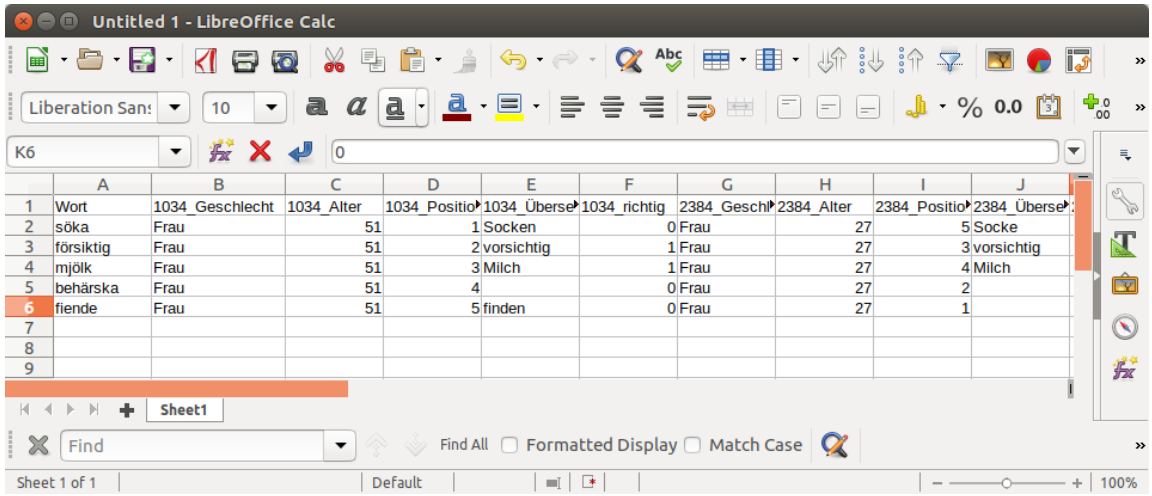
It's usually easier to work with data in a long-ish format compared to data in a wide-ish format. Moreover, when you do need your data in a wide-ish format for a particular analysis, converting a longer dataset to a wider one is typically easier than vice versa. We will, of course, illustrate how you can carry out such conversions.

Tip 2.1. When in doubt, arrange your data in a long-ish format. ♢



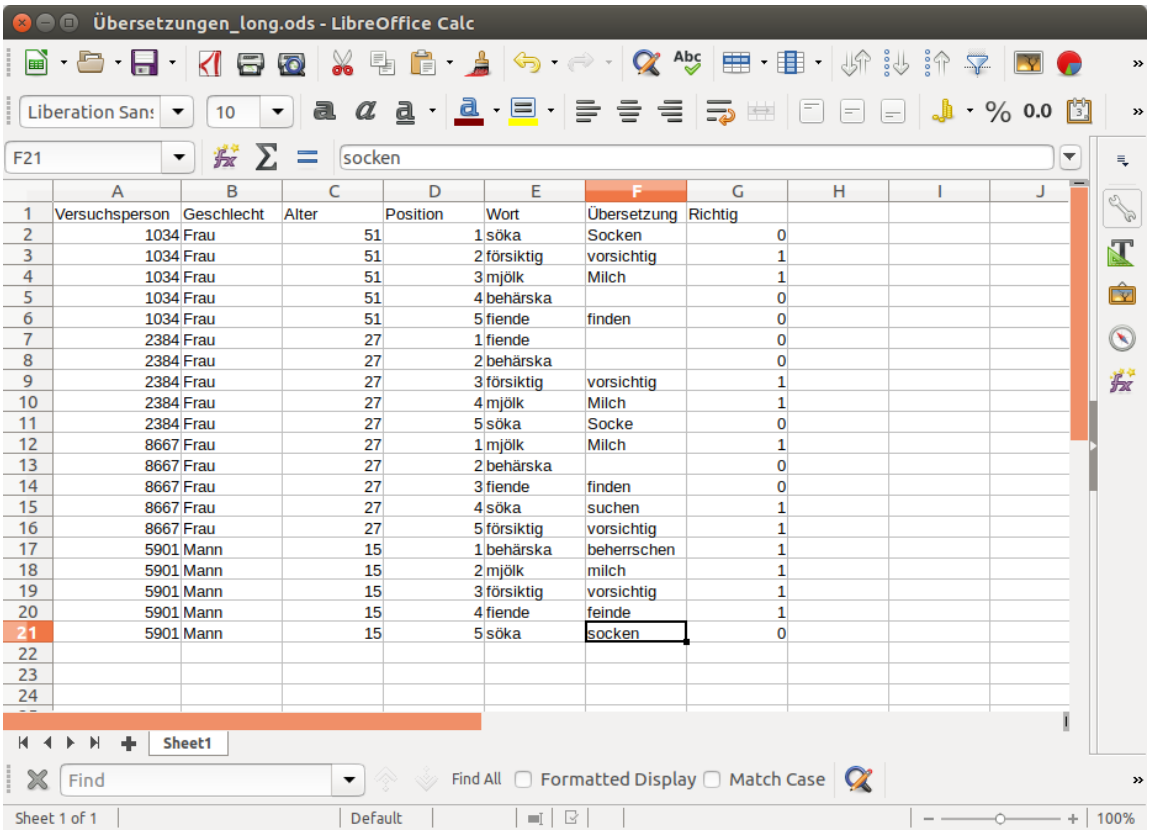
Versuchsperson	Geschlecht	Alter	söka_Position	söka_Übersetzung	söka_richtig	försiktig_Position	försiktig_Übersetzung	försiktig_richtig	mjölk_Position	mjölk_Übersetzung
1034	Frau	51	1	Socken	0	2	vorsichtig	1	3	Milch
2384	Frau	27	5	Socke	0	3	vorsichtig	1	4	Milch
8667	Frau	27	4	suchen	1	5	vorsichtig	1	1	Milch
5901	Mann	15	5	socken	1	3	vorsichtig	1	2	milch

Figure 2.1: A wide dataset with one row per participant.



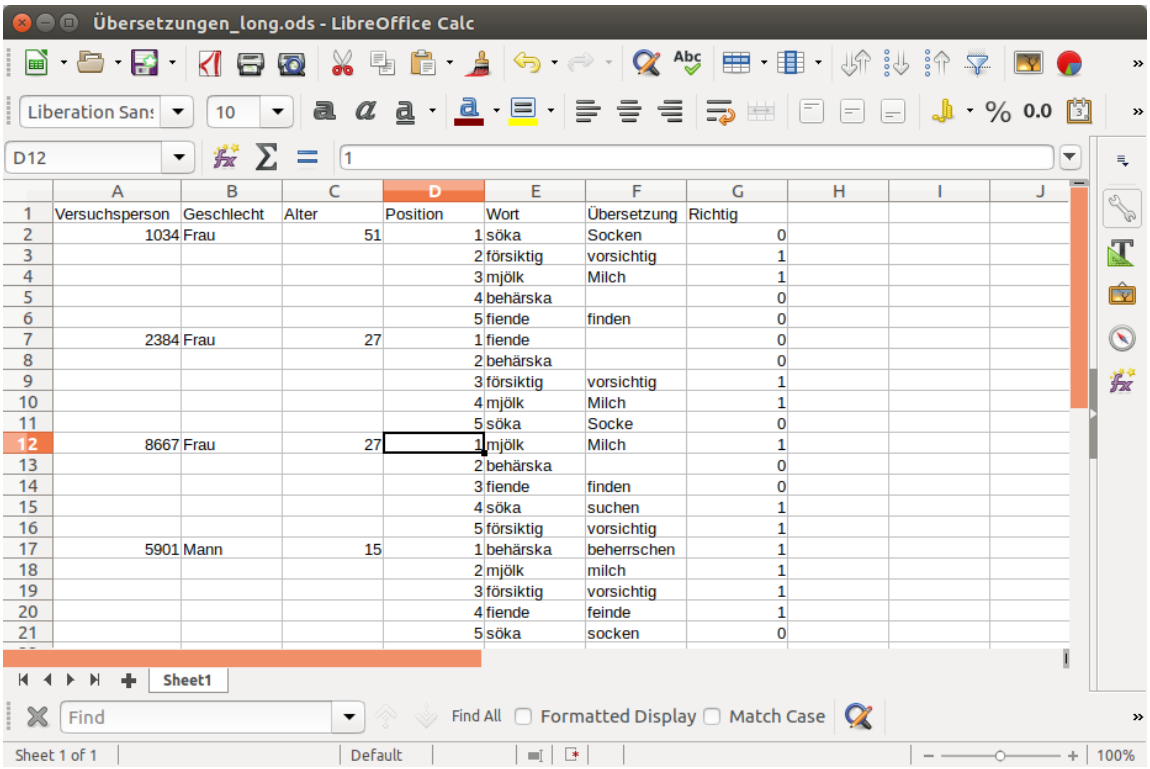
Wort	1034_Geschlecht	1034_Alter	1034_Position	1034_Übersetzung	1034_richtig	2384_Geschlecht	2384_Alter	2384_Position	2384_Übersetzung
söka	Frau	51	1	Socken	0	Frau	27	5	Socke
försiktig	Frau	51	2	vorsichtig	1	Frau	27	3	vorsichtig
mjölk	Frau	51	3	Milch	1	Frau	27	4	Milch
behärska	Frau	51	4		0	Frau	27	2	
fiende	Frau	51	5	finden	0	Frau	27	1	

Figure 2.2: A wide dataset with one row per word.



	A	B	C	D	E	F	G
1	Versuchsperson	Geschlecht	Alter	Position	Wort	Übersetzung	Richtig
2	1034	Frau	51	1	söka	Socken	0
3	1034	Frau	51	2	försiktig	vorsichtig	1
4	1034	Frau	51	3	mjölk	Milch	1
5	1034	Frau	51	4	behärska		0
6	1034	Frau	51	5	fiende	finden	0
7	2384	Frau	27	1	fiende		0
8	2384	Frau	27	2	behärska		0
9	2384	Frau	27	3	försiktig	vorsichtig	1
10	2384	Frau	27	4	mjölk	Milch	1
11	2384	Frau	27	5	söka	Socke	0
12	8667	Frau	27	1	mjölk	Milch	1
13	8667	Frau	27	2	behärska		0
14	8667	Frau	27	3	fiende	finden	0
15	8667	Frau	27	4	söka	suchen	1
16	8667	Frau	27	5	försiktig	vorsichtig	1
17	5901	Mann	15	1	behärska	beherrschen	1
18	5901	Mann	15	2	mjölk	milch	1
19	5901	Mann	15	3	försiktig	vorsichtig	1
20	5901	Mann	15	4	fiende	feinde	1
21	5901	Mann	15	5	söka	socken	0

Figure 2.3: A long dataset with one row per word per participant. Long datasets tend to be easier to manage and to analyse than wide ones.



	A	B	C	D	E	F	G
1	Versuchsperson	Geschlecht	Alter	Position	Wort	Übersetzung	Richtig
2	1034	Frau	51	1	söka	Socken	0
3				2	försiktig	vorsichtig	1
4				3	mjölk	Milch	1
5				4	behärska		0
6				5	fiende	finden	0
7	2384	Frau	27	1	fiende		0
8				2	behärska		0
9				3	försiktig	vorsichtig	1
10				4	mjölk	Milch	1
11				5	söka	Socke	0
12	8667	Frau	27	1	mjölk	Milch	1
13				2	behärska		0
14				3	fiende	finden	0
15				4	söka	suchen	1
16				5	försiktig	vorsichtig	1
17	5901	Mann	15	1	behärska	beherrschen	1
18				2	mjölk	milch	1
19				3	försiktig	vorsichtig	1
20				4	fiende	feinde	1
21				5	söka	socken	0

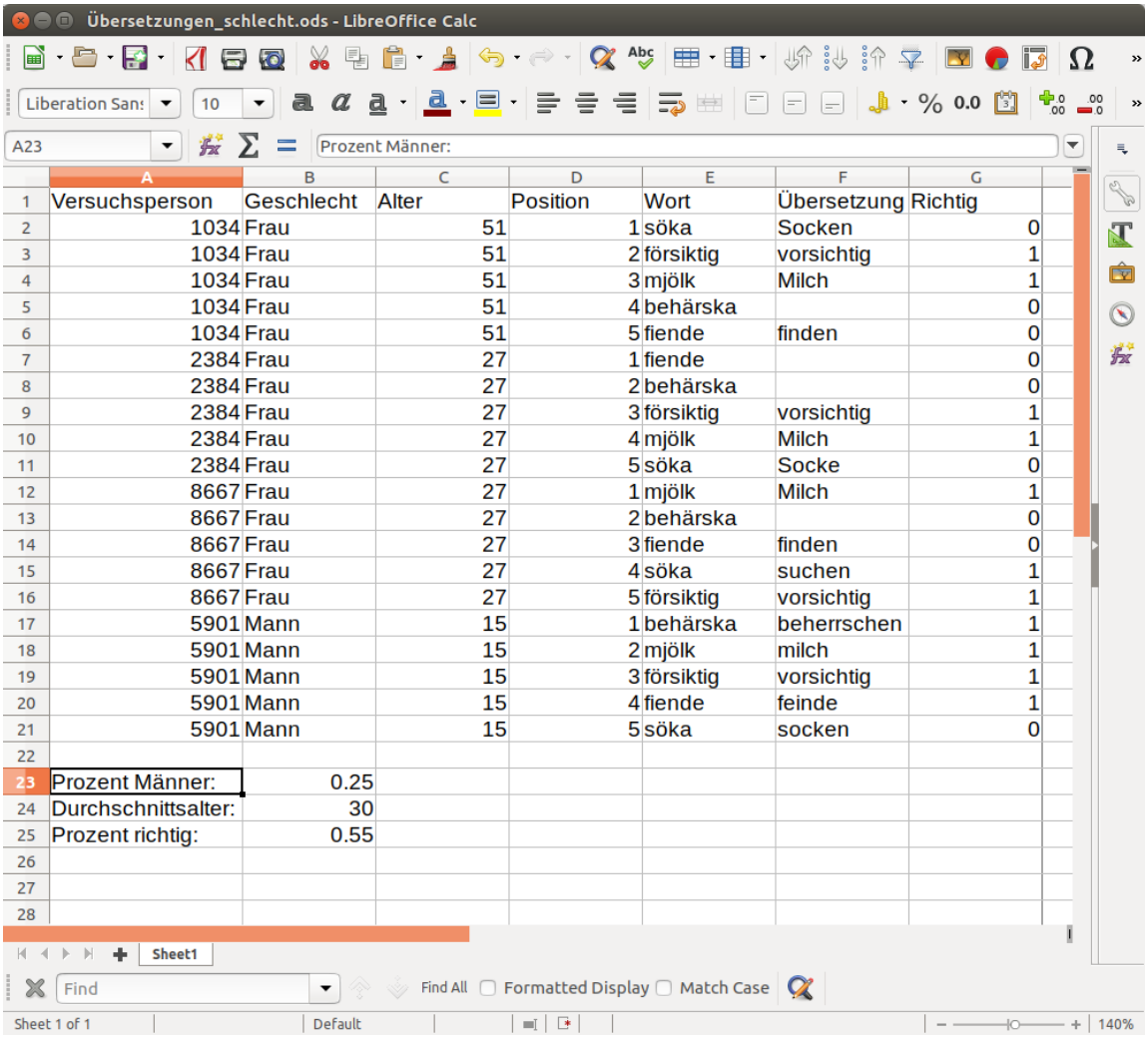
Figure 2.4: Not like this! In this dataset, several cells are left empty as their contents can be derived from other cells. But this will result in difficulties when analysing the data. Moreover, deleting some rows or changing their order would make it impossible to reconstruct the intended contents of these empty cells.

Make sure that all rows can be interpreted independently of one another. What we want to avoid is that we can't make sense of the data in some row because we need data from another row to do so, and this other row was deleted, or the order of the rows has changed, etc. For instance, in Figure 2.3, we also have a column with the `Positions`, even though we could have derived this information from the ordering of the rows. But once a row gets deleted or once the order of the rows gets permuted, we'd lose this piece of information. So don't organise the data like in Figure 2.4.

Furthermore, the dataset needs to be rectangular. Figure 2.5 shows what *not* to do: The additional rows reporting some averages don't fit in with the rest of the dataset ("Prozent Männer:" is not the ID of a `Versuchsperson`, and "30" is not a `Geschlecht`).

2.1.2 Naming variables

Make your life during the data analysis easier by using short but descriptive labels for variables or values in the spreadsheet. By doing so, you avoid that you constantly need to look up in the project's codebook what the labels mean, thereby reducing the likelihood that you'll make errors. Moreover, you save yourself some typing.



	A	B	C	D	E	F	G
1	Versuchsperson	Geschlecht	Alter	Position	Wort	Übersetzung	Richtig
2	1034	Frau	51	1	söka	Socken	0
3	1034	Frau	51	2	försiktig	vorsichtig	1
4	1034	Frau	51	3	mjölk	Milch	1
5	1034	Frau	51	4	behärska		0
6	1034	Frau	51	5	fiende	finden	0
7	2384	Frau	27	1	fiende		0
8	2384	Frau	27	2	behärska		0
9	2384	Frau	27	3	försiktig	vorsichtig	1
10	2384	Frau	27	4	mjölk	Milch	1
11	2384	Frau	27	5	söka	Socke	0
12	8667	Frau	27	1	mjölk	Milch	1
13	8667	Frau	27	2	behärska		0
14	8667	Frau	27	3	fiende	finden	0
15	8667	Frau	27	4	söka	suchen	1
16	8667	Frau	27	5	försiktig	vorsichtig	1
17	5901	Mann	15	1	behärska	beherrschen	1
18	5901	Mann	15	2	mjölk	milch	1
19	5901	Mann	15	3	försiktig	vorsichtig	1
20	5901	Mann	15	4	fiende	feinde	1
21	5901	Mann	15	5	söka	socken	0
22							
23	Prozent Männer:	0.25					
24	Durchschnittsalter:	30					
25	Prozent richtig:	0.55					
26							
27							
28							

Figure 2.5: Not like this! This dataset is not rectangular.

A few examples:

- When working with questionnaire data, don't simply label the columns with the number of the question in the questionnaire (e.g., `Q3` or `Question17`). Instead, use more descriptive labels such as `DegreeFather` or `DialectUse`.
- If you have a column labelled `Sex` that is filled with 1s and 0s, you'll constantly have to look up if the 1s refer to men or to women. Instead, either fill the column with `m(an)` and `w(oman)` values, or rename the column `Man`, so that you know that the 1s refer to men.
- Try to strike a balance between descriptive power and length. For instance, a variable named `HowOftenDoYouSpeakStandardGerman` will get on your nerves before long; `SpeakStandard` could be sufficient.

Tip 2.2 (Use a codebook)**.** Having short and descriptive variable names is not an excuse for not maintaining a codebook that spells out precisely what each variable in the dataset refers to. See `https://osf.io/d9gnh` for an example of a codebook. ♢

2.1.3 Labelling missing values

In Figure 2.3, I left some of the translation cells empty. From this, I can deduce that the word was presented to the participant but that he or she did not translate the word. However, in other situations, it could be the case that some participants were inadvertently never shown a particular word (e.g., due to a programming error) or that some of the participants' answers were irretrievably lost. We should enable ourselves to distinguish between these cases, for instance by marking the latter cases using `NA` (*not available*). Alternatively, we could explicitly label cases where the participant did not translate the word with `[no response]` or something similar.

If you want to be able to tell apart different reasons for missing data (such as data loss, programming errors, participants' refusal to answer a certain question, participants' being absent from a data collection etc.), it's probably easiest to just write `NA` in the column and add another column with comments detailing the reasons for the missingness.

Be aware that some data logging applications use -99 or -9999 to encode missingness. The problem with this is that, sometimes, -99 or -9999 don't immediately stand out as conspicuous values.

2.1.4 Using several smaller datasets

The spreadsheet above contains several repeated pieces of information. For instance, for all five translations provided by participant 1034, we indicated that they were provided by a woman aged 51. We can eliminate this source of redundancy by managing a handful of smaller datasets rather than just a single large one. More specifically, we can manage a dataset that contains all information that depends *just* on the participants, see Figure 2.6. Each participant has a unique ID (here: `Versuchsperson`), and the dataset contains contains a single row for each participant. If we need to correct an entry related to, say,

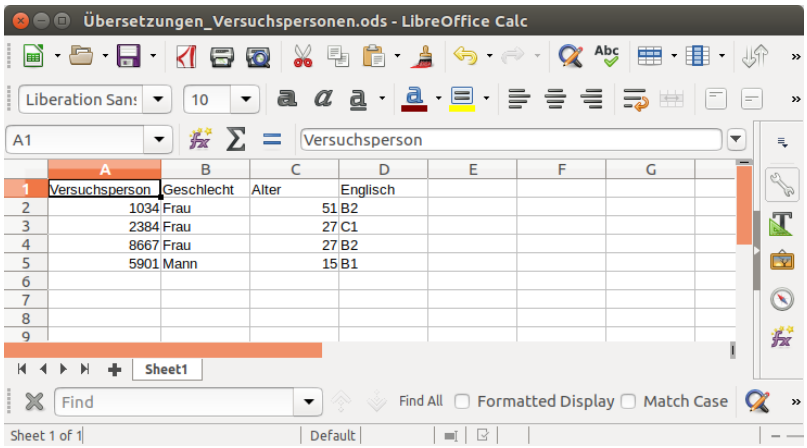


Figure 2.6: The first smaller dataset only contains information concerning the participants.

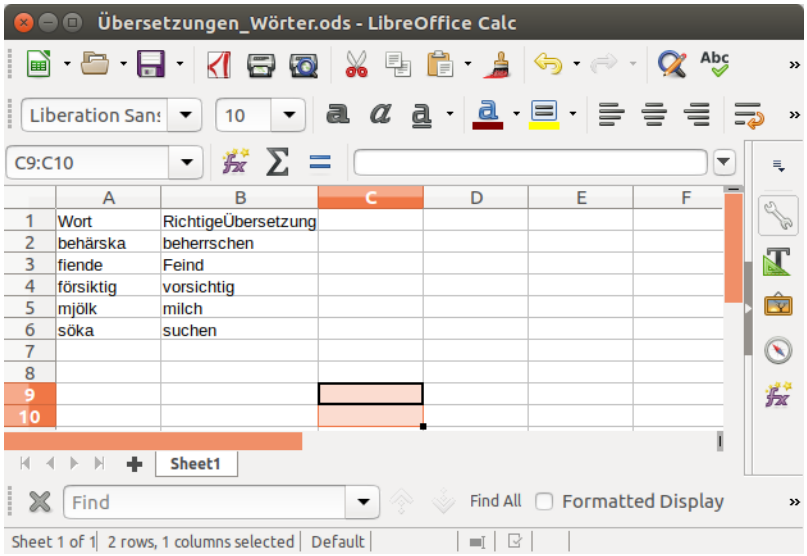
some participant's age, we just need to change it here—once—rather than five times in the larger dataset.

By the same token, we can put all information that depends *just* on the stimuli used in a separate dataset, see Figure 2.7. Here, too, each stimulus has a unique ID (here: `Wort`).

The third dataset then only contains information that depends on the *combination* of a particular participant and a particular word. As shown in Figure 2.8, each row in this dataset contains the IDs of both the participant and the stimulus that the response is related to. But any other information related to just the word or to just the participant is left out of this dataset. As we'll see shortly, we can easily add the information related to the participants or to the words to this dataset from the other two datasets.

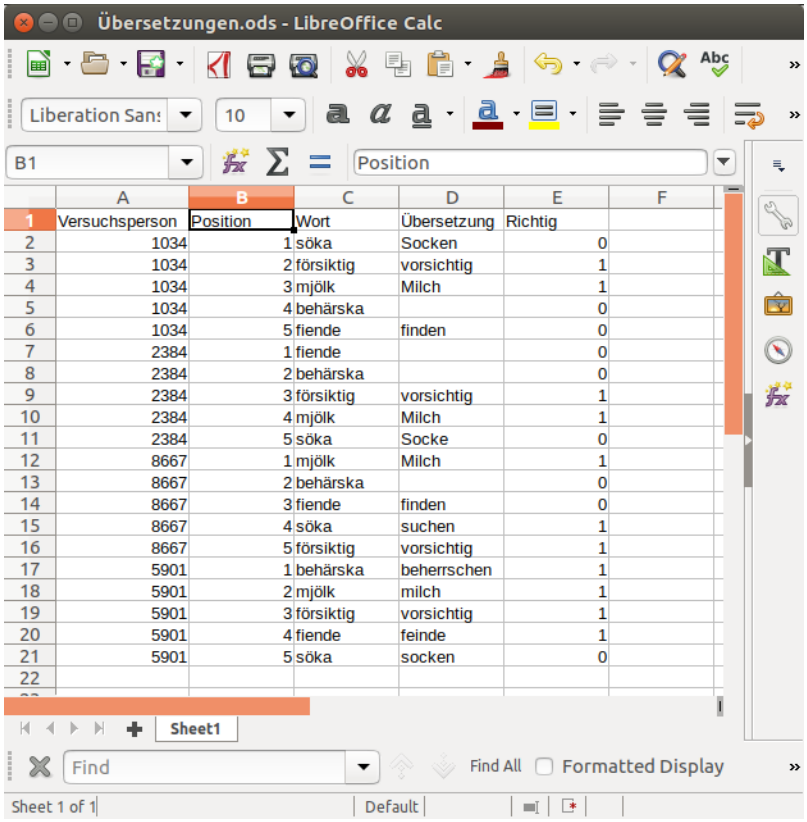
2.1.5 Miscellaneous tips

- Mind capitalisation. For some programs and computer languages (e.g., SQL), `Frau` and `frau` are the same value; for others (including R), they are not.
- Mind trailing spaces. `Mann␣` (with a trailing space) and `Mann` (without a trailing space) are different values to a computer.
- Special symbols, including the Umlaut, sometimes lead to problems, especially when collaborating with people whose localisation settings differ from yours.
- Format dates in the YYYY/MM/DD format. This way, if you sort the data alphabetically, they're already in chronological order.
- Don't use colour-coding in your spreadsheet. Or if you do use it, be aware that you'll lose this information once you import your data into your statistics program.
- Work as little as possible in the spreadsheet. After you've entered your data into a spreadsheet, all further steps in your analysis should be carried out in R (or Python,



	A	B	C	D	E	F
1	Wort	RichtigeÜbersetzung				
2	behärska	beherrschen				
3	fiende	Feind				
4	försiktig	vorsichtig				
5	mjölk	milch				
6	söka	suchen				
7						
8						
9						
10						

Figure 2.7: The second smaller dataset only contains information concerning the words.



	A	B	C	D	E	F
1	Versuchsperson	Position	Wort	Übersetzung	Richtig	
2	1034	1	söka	Socken	0	
3	1034	2	försiktig	vorsichtig	1	
4	1034	3	mjölk	Milch	1	
5	1034	4	behärska		0	
6	1034	5	fiende	finden	0	
7	2384	1	fiende		0	
8	2384	2	behärska		0	
9	2384	3	försiktig	vorsichtig	1	
10	2384	4	mjölk	Milch	1	
11	2384	5	söka	Socke	0	
12	8667	1	mjölk	Milch	1	
13	8667	2	behärska		0	
14	8667	3	fiende	finden	0	
15	8667	4	söka	suchen	1	
16	8667	5	försiktig	vorsichtig	1	
17	5901	1	behärska	beherrschen	1	
18	5901	2	mjölk	milch	1	
19	5901	3	försiktig	vorsichtig	1	
20	5901	4	fiende	feinde	1	
21	5901	5	söka	socken	0	
22						

Figure 2.8: The third dataset only contains the translations.

or Julia, or what-have-you). Don't calculate, sort, copy, paste, move, reshape, draw graphs etc. in Excel or whatever spreadsheet program you prefer. Treat your finished spreadsheet as the *immutable* source of data from which your results will be obtained, so that when in doubt, it's clear which file you should go back to.

When using R (or Python, or Julia, or whatever), use (a) script(s) to read in the original dataset and convert the data in it to a format more amenable to analysis, but whatever you do, don't overwrite the original file.

- Use data validation. When entering data in a spreadsheet program, you can use the data validation functions to minimise the chances that you enter faulty data. For instance, if you know that the possible responses to a certain question are either integers from 0 to 5 or `NA`, make a data validation rule that prevents you from entering impossible data.

  Also see the blog post *A data entry form with failsafes* (6 July 2018).

2.2 Reading in datasets

We'll work with the `here` and `tidyverse` packages you've installed in the previous chapter. Even if you've already installed these, you still need to *load* these packages in every R session in which you use them. To do so, include the following lines in your R script and execute them:

```
library(here)
library(tidyverse)
```

If you run these commands, they will output a few messages. The output of `library(here)` tells you the path relative to which the `here()` function in the `here` package will look for files etc. The messages generated by the `library(tidyverse)` command tell you which of the packages that make up the tidyverse suite are loaded by default. Some of the functions in these packages have the same name as functions that are part of packages that are loaded automatically whenever you start up R. What this means is that, if you want to use, say, the `filter()` function from the `stats` package, you need to use the notation `stats::filter()` instead of just `filter()`.

Incidentally, you don't need to install these packages every time you need them, but you do need to load them every time you need them in a new session. I recommend including all of the packages you need in a script at the top of your script. This way, everyone reading your script knows at the outset which packages they'll have to install to reproduce your analysis. Remove any `library()` commands for packages you ended up not using in the script.

With all that out of the way, let's get started. We'll focus on reading in two kinds of spreadsheets: Excel spreadsheets in the XLS(X) format, and CSV files.

2.2.1 Excel files

In order to read in XLS(X) files, we need the `readxl` package. This package is part of the `tidyverse` suite, but it does not get loaded automatically as you load the `tidyverse` suite. So we load it separately:

```
library(readxl)
```

Assuming the file `uebersetzungen.xlsx` is located in the `data` subdirectory of your R project directory, we can read it into R as follows.

```
translations <- read_excel(here("data", "uebersetzungen.xlsx"))
```

The dataset is now loaded as a so-called *tibble* named `translations`. We can display it by simply typing its name at the prompt:

```
translations

# A tibble: 20 x 5
  Versuchsperson Position Wort      Übersetzung Richtig
           <dbl>    <dbl> <chr>     <chr>         <dbl>
1           1034        1 söka      Socken            0
2           1034        2 försiktig vorsichtig        1
3           1034        3 mjölk     Milch             1
4           1034        4 behärska  <NA>              0
5           1034        5 fiende    finden            0
6           2384        1 fiende    <NA>              0
7           2384        2 behärska  <NA>              0
# i 13 more rows
```

Tip 2.3 (Type, don't copy-paste—at home)**.** You're going to take away much more from this course if you copy the code snippets by *typing* them rather than by copy-pasting them. Do this at home and not during the lecture as you'll inevitably make some typos, causing you to lose track of the lecture. ♢

Remark 2.4 (Assignment operators)**.** The symbol combination `<-` is the *assignment operator*. It creates a new object in the working memory or overwrites an existing one with the same name. This object is referred to by the name to the left of the assignment operator.

You'll often see that the equality sign (`=`) is used as the assignment operator. But it is also used to assign values to function parameters. I've tried to adhere to the *one form, one function* principle throughout and only use `<-` as the assignment operator. ♢

Remark 2.5 (Data frames and tibbles)**.** Rectangular data sets are referred to as *data frames* in R. The `tidyverse` slightly changes their functionality, mostly in order to allow for prettier displaying, and refers to them as *tibbles*. ♢

Remark 2.6. Empty cells (for instance, the `Übersetzung` value in the fourth row) are automatically interpreted as missing data (`<NA>`). The documentation of the `read_excel()`

function, which you can access by typing `?read_excel` at the R prompt, suggests that we can override this behaviour, but we can't. ♢

If, instead of printing the entire tibble at the prompt, we just want to display the first few rows, we can use `slide_head()`:

```
slice_head(translations, n = 4)

# A tibble: 4 x 5
  Versuchsperson Position Wort      Übersetzung Richtig
           <dbl>    <dbl> <chr>     <chr>         <dbl>
1           1034        1 söka      Socken            0
2           1034        2 försiktig vorsichtig        1
3           1034        3 mjölk     Milch             1
4           1034        4 behärska  <NA>              0
```

`slice_tail()` works similarly. If you want to display a random selection of rows, you can use `slice_sample()`:

```
slice_sample(translations, n = 5)

# A tibble: 5 x 5
  Versuchsperson Position Wort     Übersetzung Richtig
           <dbl>    <dbl> <chr>    <chr>         <dbl>
1           8667        3 fiende   finden            0
2           8667        2 behärska <NA>              0
3           1034        4 behärska <NA>              0
4           1034        1 söka     Socken            0
5           2384        2 behärska <NA>              0
```

Again, for details, you can check the documentation of these functions that is available at `?slice_head`.

Exercise 2.7. The file `VanDenBroek2018.xlsx` contains the data from three experiments reported by van den Broek et al. (2018). Using the `read_xlsx()` function, read in the data from the second experiment.

Hint: Inspect the Excel file and then consult the help page of `read_xlsx()` by means of `?read_xlsx`. ♢

2.2.2 CSV files

A popular format for storing spreadsheets is the CSV format. CSV is short for *comma-separated values*: Cells on the same row are separated by commas, see Figure 2.9. Sometimes, text strings are additionally surrounded by quotation marks.

To read in CSV files, we can use the `read_csv()` function, which is part of the `readr` package, which in turn is automatically loaded when the `tidyverse` suit is loaded:

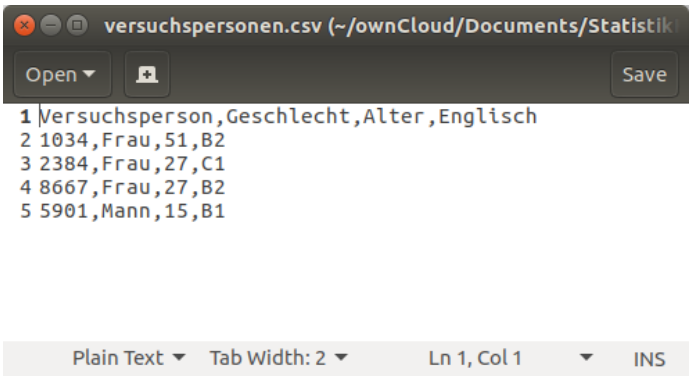


Figure 2.9: A spreadsheet stored as comma-separated values.

```
translations <- read_csv(here("data", "uebersetzungen.csv"))
translations
```

```
# A tibble: 20 x 5
  Versuchsperson Position Wort      Übersetzung Richtig
           <dbl>    <dbl> <chr>     <chr>         <dbl>
1           1034        1 söka      Socken            0
2           1034        2 försiktig vorsichtig        1
3           1034        3 mjölk     Milch             1
4           1034        4 behärska  <NA>              0
5           1034        5 fiende    finden            0
6           2384        1 fiende    <NA>              0
7           2384        2 behärska  <NA>              0
# i 13 more rows
```

Running this command produces a number of messages, which I haven't included in these lecture notes. These messages show that the `read_csv()` function correctly recognised that the columns `Wort` and `Übersetzung` contain text ('chr', character) strings, whereas `Versuchsperson`, `Position` and `Richtig` contain numbers ('dbl' for 'double', a number format).

***Remark 2.8.** With `read_csv()`, we can specify that only cells containing `NA` are marked as missing data:

```
translations2 <- read_csv(here("data", "uebersetzungen.csv"), na = "NA")
translations2
```

```
# A tibble: 20 x 5
  Versuchsperson Position Wort      Übersetzung  Richtig
           <dbl>    <dbl> <chr>     <chr>          <dbl>
1           1034        1 söka      "Socken"           0
2           1034        2 försiktig "vorsichtig"       1
3           1034        3 mjölk     "Milch"            1
```

```
4           1034        4 behärska  ""                0
5           1034        5 fiende    "finden"          0
6           2384        1 fiende    ""                0
7           2384        2 behärska  ""                0
# i 13 more rows
```

Note that the `Übersetzung` value in row 4 is just empty, not `NA`. ♢

Let's also read in the datasets containing the information pertaining to the participants and items:

```
participants <- read_csv(here("data", "versuchspersonen.csv"))
items <- read_csv(here("data", "woerter.csv"))
```

Remark 2.9 (Different CSV formats)**.** If you save an Excel spreadsheet as a CSV file on a French- or German-language computer system, the cells will be separated using semicolons rather than using commas. The reason is that commas are used as decimal separators in French and German. To read in 'CSV' files that use semicolons as cell separators, you can use the `read_csv2()` function.

Incidentally, if you use LibreOffice.org Calc instead of Excel, you can choose which cell separator gets used if you export a spreadsheet as a CSV file. ♢

Remark 2.10 (*`read_csv()` vs. `read.csv()`)**.** More seasoned R users are probably already familiar with the `read.csv()` function (with a dot rather than an underscore). The `read_csv()` function is merely the tidyverse counterpart to this function. The main difference between them is that `read_csv()` creates tibbles, whereas `read.csv()` creates dataframes. ♢

2.2.3 Other formats

Some people prefer to use tabs or spaces to separate cells rather than commas or semicolons. Consult the help page for the `read_tsv()` and `read_delim()` functions to see how you can read in data using other separators. Particularly the `delim` and `quote` arguments are relevant.

For reading in data from Google Sheets, see `googlesheets4`.

For interacting with Google Drive, see `googledrive`.

For reading in SPSS, Stata and SAS files, see `haven`.

2.3 Joining datasets

Having read in our three datasets as `translations`, `participants`, and `items`, we want to merge these datasets into one large dataset. The most useful function for this is `left_join()`, which takes the dataset passed as its `x` argument and adds to this the corresponding rows from the `y` dataset:

```
all_data <- left_join(x = translations, y = participants)
```

Note that the `left_join()` function recognises that the variable shared between both datasets is called `Versuchsperson`. Hence, the information related to participant 1034 contained in `participants` is added to each row in `translations` for which the value of `Versuchsperson` is 1034, and similarly for the other participants:

```
all_data

# A tibble: 20 x 8
  Versuchsperson Position Wort      Übersetzung Richtig
           <dbl>    <dbl> <chr>     <chr>         <dbl>
1           1034        1 söka      Socken            0
2           1034        2 försiktig vorsichtig        1
3           1034        3 mjölk     Milch             1
4           1034        4 behärska  <NA>              0
5           1034        5 fiende    finden            0
6           2384        1 fiende    <NA>              0
7           2384        2 behärska  <NA>              0
# i 13 more rows
# i 3 more variables: Geschlecht <chr>, Alter <dbl>,
#   Englisch <chr>
```

If you don't want the `left_join()` function to figure out what the shared variable is, you can specify it explicitly:

```
all_data <- left_join(x = translations, y = participants,
                      by = "Versuchsperson")
```

If the shared variable has different names in the different datasets, you can use something like

```
new <- left_join(x = left_dataset, y = right_dataset,
                 by = join_by(var_left == var_right))
```

See `?join_by` for more information.

If there are values in the shared variable that occur in the `x` dataset that don't occur in the `y` dataset, the values in the added columns will read `NA` for these rows.

Further join functions are the following, see `?join` for details:

- `right_join(x, y)`: Keep all entries in `y`. Add corresponding entries in `x` to it.
- `full_join(x, y)`: Keep all entries in both `x` and `y`. Add `NA`s if there is no corresponding entry in the other dataset.
- `inner_join(x, y)`: Only keep entries in `x` for which there is a corresponding entry in `y`. Add these corresponding entries.

- `semi_join(x, y)`: Only keep entries in `x` for which there is a corresponding entry in `y`. Don't add these corresponding entries.
- `anti_join(x, y)`: Only keep entries in `x` for which there are *no* corresponding entries in `y`.

In the current example, `left_join()`, `full_join()` and `inner_join()` would all yield the same result. But this isn't always the case.

Let's also add the information pertaining to the words:

```
all_data <- left_join(all_data, items)
```

Exercise 2.11 (Join functions)**.**

1. Use the following code to generate two tibbles called `left` and `right`, and to inspect them:

```
left <- tibble(A = c("a", "b", "c", NA),
               B = c(1, 2, NA, 4))
right <- tibble(B = c(1, 3, 4, 4),
                C = c(10, NA, 12, 7))
left
right
```

2. *Without running the following commands*, predict what their output will look like:

```
left_join(x = left, y = right)
right_join(x = left, y = right)
full_join(x = left, y = right)
inner_join(x = left, y = right)
semi_join(x = left, y = right)
semi_join(x = right, y = left) # !
anti_join(x = left, y = right)
anti_join(x = right, y = left) # !
```

3. Now run the commands above to verify your predictions.
4. Create two new tibbles using the code below:

```
left <- tibble(A = c("a", "b"),
               B = c(1, NA))
right <- tibble(B = c(1, NA, NA),
                C = c(0, 1, 2))
left
right
```

5. Consult the help page for `left_join()` and look up the `na_matches` parameter under 'Arguments'. Predict what the output of the following two commands will look like, and only then check your answer.

```
left_join(left, right)
left_join(left, right, na_matches = "never")
```

♢

2.4 Queries

2.4.1 Selecting by row number

We can select the third row of a dataset like so:

```
slice(all_data, 3)

# A tibble: 1 x 9
  Versuchsperson Position Wort  Übersetzung Richtig
           <dbl>    <dbl> <chr> <chr>         <dbl>
1           1034        3 mjölk Milch             1
# i 4 more variables: Geschlecht <chr>, Alter <dbl>,
#   Englisch <chr>, RichtigeÜbersetzung <chr>
```

Alternatively, we can write this command as follows. The symbol combination `|>` allows us to take an object (here: `all_data`) and pass it to a function as its first argument. As we'll later see, we can use `|>` to string together a host of function calls without creating illegible code.

```
all_data |>
  slice(3)

# A tibble: 1 x 9
  Versuchsperson Position Wort  Übersetzung Richtig
           <dbl>    <dbl> <chr> <chr>         <dbl>
1           1034        3 mjölk Milch             1
# i 4 more variables: Geschlecht <chr>, Alter <dbl>,
#   Englisch <chr>, RichtigeÜbersetzung <chr>
```

Use CTRL + SHIFT + M (Windows, Linux) or CMD + SHIFT + M (macOS) to insert `|>` at the current text cursor position.

We can also select multiple rows:

```
# Rows 5 and 7
all_data |>
  slice(c(5, 7))

# A tibble: 2 x 9
  Versuchsperson Position Wort     Übersetzung Richtig
           <dbl>    <dbl> <chr>    <chr>         <dbl>
1           1034        5 fiende   finden            0
2           2384        2 behärska <NA>              0
```

```
# i 4 more variables: Geschlecht <chr>, Alter <dbl>,
#   Englisch <chr>, RichtigeÜbersetzung <chr>
```

```
# Rows 5 to (and including) 7
all_data |>
  slice(5:7)
```

```
# A tibble: 3 x 9
  Versuchsperson Position Wort     Übersetzung Richtig
           <dbl>    <dbl> <chr>    <chr>         <dbl>
1           1034        5 fiende   finden            0
2           2384        1 fiende   <NA>              0
3           2384        2 behärska <NA>              0
# i 4 more variables: Geschlecht <chr>, Alter <dbl>,
#   Englisch <chr>, RichtigeÜbersetzung <chr>
```

The results of such actions can be stored as separate objects, for instance, like so:

```
rows7_12 <- all_data |>
  slice(7:12)
rows7_12
```

```
# A tibble: 6 x 9
  Versuchsperson Position Wort      Übersetzung Richtig
           <dbl>    <dbl> <chr>     <chr>         <dbl>
1           2384        2 behärska  <NA>              0
2           2384        3 försiktig vorsichtig        1
3           2384        4 mjölk     Milch             1
4           2384        5 söka      Socke             0
5           8667        1 mjölk     Milch             1
6           8667        2 behärska  <NA>              0
# i 4 more variables: Geschlecht <chr>, Alter <dbl>,
#   Englisch <chr>, RichtigeÜbersetzung <chr>
```

We can export this new object as a CSV file like so:

```
write_csv(rows7_12, here("data", "rows7_12.csv"))
```

2.4.2 Selecting rows by values

Selecting rows by their number isn't too useful. But selecting rows satisfying some set of conditions is very useful. Here are a few examples:

```
# All data corresponding to the word 'fiende'
all_data |>
  filter(Wort == "fiende")
```

```
# A tibble: 4 x 9
  Versuchsperson Position Wort   Übersetzung Richtig
```

```
           <dbl>    <dbl> <chr>  <chr>         <dbl>
1           1034        5 fiende finden            0
2           2384        1 fiende <NA>              0
3           8667        3 fiende finden            0
4           5901        4 fiende feinde            1
# i 4 more variables: Geschlecht <chr>, Alter <dbl>,
#   Englisch <chr>, RichtigeÜbersetzung <chr>

# Note: use '!=' for 'not equal to'.

# All data corresponding to participants older than 30
all_data |>
  filter(Alter > 30)

# A tibble: 5 x 9
  Versuchsperson Position Wort      Übersetzung Richtig
           <dbl>    <dbl> <chr>     <chr>         <dbl>
1           1034        1 söka      Socken            0
2           1034        2 försiktig vorsichtig        1
3           1034        3 mjölk     Milch             1
4           1034        4 behärska  <NA>              0
5           1034        5 fiende    finden            0
# i 4 more variables: Geschlecht <chr>, Alter <dbl>,
#   Englisch <chr>, RichtigeÜbersetzung <chr>

# Note: use '>=' for 'at least as old as',
#           '<=' for 'no older than',
#        and '<' for 'younger than'.
```

We can use `is.na()` to check for missing values. Note the use of `!` to negate a condition.

```
all_data |>
  filter(is.na(Übersetzung))

# A tibble: 4 x 9
  Versuchsperson Position Wort     Übersetzung Richtig
           <dbl>    <dbl> <chr>    <chr>         <dbl>
1           1034        4 behärska <NA>              0
2           2384        1 fiende   <NA>              0
3           2384        2 behärska <NA>              0
4           8667        2 behärska <NA>              0
# i 4 more variables: Geschlecht <chr>, Alter <dbl>,
#   Englisch <chr>, RichtigeÜbersetzung <chr>

all_data |>
  filter(!is.na(Übersetzung))
```

```
# A tibble: 16 x 9
  Versuchsperson Position Wort      Übersetzung Richtig
           <dbl>    <dbl> <chr>     <chr>         <dbl>
1           1034        1 söka      Socken            0
2           1034        2 försiktig vorsichtig        1
3           1034        3 mjölk     Milch             1
4           1034        5 fiende    finden            0
5           2384        3 försiktig vorsichtig        1
6           2384        4 mjölk     Milch             1
7           2384        5 söka      Socke             0
# i 9 more rows
# i 4 more variables: Geschlecht <chr>, Alter <dbl>,
#   Englisch <chr>, RichtigeÜbersetzung <chr>
```

We can string together multiple `filter()` calls:

```
# incorrect translations to first word
all_data |>
  filter(Position == 1) |>
  filter(Richtig == 0)
```

```
# A tibble: 2 x 9
  Versuchsperson Position Wort   Übersetzung Richtig
           <dbl>    <dbl> <chr>  <chr>         <dbl>
1           1034        1 söka   Socken            0
2           2384        1 fiende <NA>              0
# i 4 more variables: Geschlecht <chr>, Alter <dbl>,
#   Englisch <chr>, RichtigeÜbersetzung <chr>
```

An alternative way of writing this is as follows:

```
all_data |>
  filter(Position == 1 & Richtig == 0)
```

```
# A tibble: 2 x 9
  Versuchsperson Position Wort   Übersetzung Richtig
           <dbl>    <dbl> <chr>  <chr>         <dbl>
1           1034        1 söka   Socken            0
2           2384        1 fiende <NA>              0
# i 4 more variables: Geschlecht <chr>, Alter <dbl>,
#   Englisch <chr>, RichtigeÜbersetzung <chr>
```

'or' conditions can be created using `|`:

```
# translations to first word or incorrect translations
all_data |>
  filter(Position == 1 | Richtig == 0)
```

```
# A tibble: 11 x 9
  Versuchsperson Position Wort     Übersetzung Richtig
           <dbl>    <dbl> <chr>    <chr>         <dbl>
1           1034        1 söka     Socken            0
2           1034        4 behärska <NA>              0
3           1034        5 fiende   finden            0
4           2384        1 fiende   <NA>              0
5           2384        2 behärska <NA>              0
6           2384        5 söka     Socke             0
7           8667        1 mjölk    Milch             1
# i 4 more rows
# i 4 more variables: Geschlecht <chr>, Alter <dbl>,
#   Englisch <chr>, RichtigeÜbersetzung <chr>
```

Note that in logic, 'or' is always inclusive. Exclusive 'or' ('xor') can be obtained as follows:

```
# translations to first word or incorrect translations, but not both
all_data |>
  filter(Position == 1 | Richtig == 0) |>
  filter(!(Position == 1 & Richtig == 0))
```

```
# A tibble: 9 x 9
  Versuchsperson Position Wort     Übersetzung Richtig
           <dbl>    <dbl> <chr>    <chr>         <dbl>
1           1034        4 behärska <NA>              0
2           1034        5 fiende   finden            0
3           2384        2 behärska <NA>              0
4           2384        5 söka     Socke             0
5           8667        1 mjölk    Milch             1
6           8667        2 behärska <NA>              0
7           8667        3 fiende   finden            0
# i 2 more rows
# i 4 more variables: Geschlecht <chr>, Alter <dbl>,
#   Englisch <chr>, RichtigeÜbersetzung <chr>
```

Alternatively,

```
all_data |>
  filter(xor(Position == 1, Richtig == 0))
```

```
# A tibble: 9 x 9
  Versuchsperson Position Wort     Übersetzung Richtig
           <dbl>    <dbl> <chr>    <chr>         <dbl>
1           1034        4 behärska <NA>              0
2           1034        5 fiende   finden            0
3           2384        2 behärska <NA>              0
```

```
4           2384        5 söka     Socke             0
5           8667        1 mjölk    Milch             1
6           8667        2 behärska <NA>              0
7           8667        3 fiende   finden            0
# i 2 more rows
# i 4 more variables: Geschlecht <chr>, Alter <dbl>,
#   Englisch <chr>, RichtigeÜbersetzung <chr>
```

Exercise 2.12 (Filtering)**.**

1. Run the following commands:

```
d1 <- all_data |>
  filter(Übersetzung == "vorsichtig")
d2 <- all_data |>
  filter(Übersetzung != "vorsichtig")
```

2. Explain what both commands achieve.

3. How many rows are there in `d1`? How many in `d2`? How many in `all_data`? Explain.

4. Create a tibble `d3` that contains only those rows in `all_data` where the participants did not translate the word as *vorsichtig*. ♢

Remark 2.13 (String detection)**.** It is also possible to perform more complex string-based queries. For instance, the `stringr` package, which is loaded automatically as part of the tidyverse, contains the function `str_detect()`. As its name suggests, this function can be used to detect if a string contains a particular combination of symbols:

```
# all rows with an Übersetzung containing "ch"
all_data |>
  filter(str_detect(Übersetzung, "ch"))
```

```
# A tibble: 10 x 9
  Versuchsperson Position Wort      Übersetzung Richtig
           <dbl>    <dbl> <chr>     <chr>         <dbl>
1           1034        2 försiktig vorsichtig        1
2           1034        3 mjölk     Milch             1
3           2384        3 försiktig vorsichtig        1
4           2384        4 mjölk     Milch             1
5           8667        1 mjölk     Milch             1
6           8667        4 söka      suchen            1
7           8667        5 försiktig vorsichtig        1
# i 3 more rows
# i 4 more variables: Geschlecht <chr>, Alter <dbl>,
#   Englisch <chr>, RichtigeÜbersetzung <chr>
```

Related functions are `str_starts()` and `str_ends()` for checking if a string starts or ends with a particular combination of symbols, as well as `str_sub()` for extracting substrings based on their position in the string. As the examples below illustrate, these queries are case-sensitive:

```
# Übersetzung starts with an "s" (lowercase!)
all_data |>
  filter(str_starts(Übersetzung, "s"))

# A tibble: 2 x 9
  Versuchsperson Position Wort  Übersetzung Richtig
           <dbl>    <dbl> <chr> <chr>         <dbl>
1           8667        4 söka  suchen            1
2           5901        5 söka  socken            0
# i 4 more variables: Geschlecht <chr>, Alter <dbl>,
#   Englisch <chr>, RichtigeÜbersetzung <chr>

# Übersetzung starts with an "S" (uppercase!)
all_data |>
  filter(str_starts(Übersetzung, "S"))

# A tibble: 2 x 9
  Versuchsperson Position Wort  Übersetzung Richtig
           <dbl>    <dbl> <chr> <chr>         <dbl>
1           1034        1 söka  Socken            0
2           2384        5 söka  Socke             0
# i 4 more variables: Geschlecht <chr>, Alter <dbl>,
#   Englisch <chr>, RichtigeÜbersetzung <chr>

# Übersetzung starts with an "s" or an "S"
all_data |>
  filter(str_starts(Übersetzung, "[sS]"))

# A tibble: 4 x 9
  Versuchsperson Position Wort  Übersetzung Richtig
           <dbl>    <dbl> <chr> <chr>         <dbl>
1           1034        1 söka  Socken            0
2           2384        5 söka  Socke             0
3           8667        4 söka  suchen            1
4           5901        5 söka  socken            0
# i 4 more variables: Geschlecht <chr>, Alter <dbl>,
#   Englisch <chr>, RichtigeÜbersetzung <chr>

# Übersetzung ends with "en"
all_data |>
  filter(str_ends(Übersetzung, "en"))

# A tibble: 6 x 9
  Versuchsperson Position Wort     Übersetzung Richtig
```

```
           <dbl>    <dbl> <chr>    <chr>         <dbl>
1           1034        1 söka     Socken            0
2           1034        5 fiende   finden            0
3           8667        3 fiende   finden            0
4           8667        4 söka     suchen            1
5           5901        1 behärska beherrschen       1
6           5901        5 söka     socken            0
# i 4 more variables: Geschlecht <chr>, Alter <dbl>,
#   Englisch <chr>, RichtigeÜbersetzung <chr>
```

```
# The third symbol in Übersetzung is "c"
all_data |>
  filter(str_sub(Übersetzung, start = 3, end = 3) == "c")
```

```
# A tibble: 4 x 9
  Versuchsperson Position Wort  Übersetzung Richtig
           <dbl>    <dbl> <chr> <chr>         <dbl>
1           1034        1 söka  Socken            0
2           2384        5 söka  Socke             0
3           8667        4 söka  suchen            1
4           5901        5 söka  socken            0
# i 4 more variables: Geschlecht <chr>, Alter <dbl>,
#   Englisch <chr>, RichtigeÜbersetzung <chr>
```

More complicated queries are possible through the use of so-called regular expressions, which we'll touch on later. For more information about the `str_*()` functions, see the `stringr` documentation.

2.4.3 Selecting columns

Sometimes, a dataset is too cumbersome to handle because it contains a lot of irrelevant columns. Using `select()`, we can select those columns that are of interest. For instance,

```
all_data |>
  select(Wort, RichtigeÜbersetzung, Übersetzung) |>
  slice_head(n = 5)
```

```
# A tibble: 5 x 3
  Wort      RichtigeÜbersetzung Übersetzung
  <chr>     <chr>               <chr>
1 söka      suchen              Socken
2 försiktig vorsichtig          vorsichtig
3 mjölk     milch               Milch
4 behärska  beherrschen         <NA>
5 fiende    Feind               finden
```

There are a couple of auxiliary functions that make it easier to select columns. These are especially useful when working with large datasets. Examples are `contains()` and `starts_with()`:

```
all_data |>
  select(contains("Übersetzung"))

# A tibble: 20 x 2
  Übersetzung RichtigeÜbersetzung
  <chr>       <chr>
1 Socken      suchen
2 vorsichtig  vorsichtig
3 Milch       milch
4 <NA>        beherrschen
5 finden      Feind
6 <NA>        Feind
7 <NA>        beherrschen
# i 13 more rows

all_data |>
  select(starts_with("Richt"))

# A tibble: 20 x 2
  Richtig RichtigeÜbersetzung
    <dbl> <chr>
1       0 suchen
2       1 vorsichtig
3       1 milch
4       0 beherrschen
5       0 Feind
6       0 Feind
7       0 beherrschen
# i 13 more rows
```

See the `tidyselect` documentation for further functions.

2.4.4 Further examples

We can string together different types of commands using the pipe (`|>`):

```
# All translations for 'fiende'
all_data |>
  filter(Wort == "fiende") |>
  select(Übersetzung)

# A tibble: 4 x 1
  Übersetzung
  <chr>
1 finden
```

```
2 <NA>
3 finden
4 feinde
```

```
# All *distinct* translations for 'behärska':
all_data |>
  filter(Wort == "behärska") |>
  select(Übersetzung) |>
  distinct()
```

```
# A tibble: 2 x 1
  Übersetzung
  <chr>
1 <NA>
2 beherrschen
```

Without the pipe, these commands become difficult to read:

```
distinct(select(filter(all_data, Wort == "behärska"), Übersetzung))
```

```
# A tibble: 2 x 1
  Übersetzung
  <chr>
1 <NA>
2 beherrschen
```

2.5 Pivoting

In the course of an analysis, it is often necessary to convert a long-ish dataset to a wider one, and vice versa. This process is known as **pivoting**. To illustrate pivoting, we'll make use of a more realistic—and more complicated—dataset derived from a longitudinal project on the development of reading and writing skills in Portuguese–French and Portuguese–German bilingual children (Desgrippes et al., 2017; Pestana et al., 2017). We read in the dataset `helascot_skills.csv` as `skills`:

```
skills <- read_csv(here("data", "helascot_skills.csv"))
skills
```

```
# A tibble: 1,904 x 6
  Subject   Time LanguageTested Reading Argumentation
  <chr>    <dbl> <chr>            <dbl>         <dbl>
1 A_PLF_1      1 French           0.211             7
2 A_PLF_1      1 Portuguese       0.579             9
3 A_PLF_1      2 French           0.684            14
4 A_PLF_1      2 Portuguese       0.737            13
5 A_PLF_1      3 French           0.947            14
6 A_PLF_1      3 Portuguese       0.842            13
7 A_PLF_10     1 French           0.579             5
```

```
# i 1,897 more rows
# i 1 more variable: Narration <dbl>
```

For each participant (`Subject`) at each `Time` (T1, T2, T3) and for each `LanguageTested` (Portuguese, French, German), we have up to three scores (`Reading`, `Argumentation`, and `Narration`), arranged next to each other. But let's say we wanted to compute, for each participant, their progress on each task in each language from T1 to T2 and from T2 to T3. The way the data are laid out at present, this would at best be pretty difficult to do. But it would be easy if only the data were arranged differently, namely with all three measurements per participant and task next to each other. We can convert this dataset to the desired format in two steps.

First, we make the dataset longer by stacking the different scores underneath each other rather than next to each other. To this end, we use the function `pivot_longer()` and specify that we want to stack the values in the `Reading`, `Argumentation`, and `Narration` columns under each other, that we want to call the resulting column `Score`, and that we want to put the column names in a new column called `Skill`:

```
skills_longer <- skills |>
  pivot_longer(cols = c("Reading", "Argumentation", "Narration"),
               names_to = "Skill", values_to = "Score")
skills_longer
```

```
# A tibble: 5,712 x 5
  Subject  Time LanguageTested Skill          Score
  <chr>   <dbl> <chr>          <chr>          <dbl>
1 A_PLF_1     1 French         Reading        0.211
2 A_PLF_1     1 French         Argumentation  7
3 A_PLF_1     1 French         Narration     NA
4 A_PLF_1     1 Portuguese     Reading        0.579
5 A_PLF_1     1 Portuguese     Argumentation  9
6 A_PLF_1     1 Portuguese     Narration      6
7 A_PLF_1     2 French         Reading        0.684
# i 5,705 more rows
```

Then, we make this tibble wider by putting the three measurements per skill and language next to each other using `pivot_wider()`. We also prefix the `Time` values with a `T`.

```
skills_wider_time <- skills_longer |>
  pivot_wider(names_from = Time, values_from = Score,
              names_prefix = "T")
skills_wider_time
```

```
# A tibble: 2,100 x 6
  Subject  LanguageTested Skill             T1     T2     T3
  <chr>    <chr>          <chr>          <dbl>  <dbl>  <dbl>
1 A_PLF_1  French         Reading        0.211  0.684  0.947
```

```
2 A_PLF_1  French         Argumentation  7     14     14
3 A_PLF_1  French         Narration     NA     10      8
4 A_PLF_1  Portuguese     Reading        0.579  0.737  0.842
5 A_PLF_1  Portuguese     Argumentation  9     13     13
6 A_PLF_1  Portuguese     Narration      6      9     NA
7 A_PLF_10 French         Reading        0.579  0.474  0.316
# i 2,093 more rows
```

Using `mutate()`, we can now easily compute the differences between `T1` and `T2` and between `T2` and `T3`:

```
skills_wider_time |>
  mutate(ProgressT1_T2 = T2 - T1,
         ProgressT2_T3 = T3 - T2) |>
  select(Subject, LanguageTested, Skill, ProgressT1_T2, ProgressT2_T3)
```

```
# A tibble: 2,100 x 5
  Subject  LanguageTested Skill  ProgressT1_T2 ProgressT2_T3
  <chr>    <chr>          <chr>          <dbl>         <dbl>
1 A_PLF_1  French         Readi~         0.474         0.263
2 A_PLF_1  French         Argum~         7             0
3 A_PLF_1  French         Narra~        NA            -2
4 A_PLF_1  Portuguese     Readi~         0.158         0.105
5 A_PLF_1  Portuguese     Argum~         4             0
6 A_PLF_1  Portuguese     Narra~         3            NA
7 A_PLF_10 French         Readi~        -0.105        -0.158
# i 2,093 more rows
```

Tip 2.14 (First longer, then wider)**.** The two-step approach shown above is one that I've found generally useful. First convert the tibble to a format that is longer than needed, then pivot it to the wider format desired. ♢

Now imagine that we wanted to compute, for each participant in each skill at each data collection, the difference between the Portuguese score and the German/French score. The first step is the same, resulting in `skills_longer`. The second step is now similar, but we put the different languages next to each other:

```
skills_wider_language <- skills_longer |>
  pivot_wider(names_from = LanguageTested, values_from = Score)
skills_wider_language
```

```
# A tibble: 3,999 x 6
  Subject  Time Skill         French Portuguese German
  <chr>   <dbl> <chr>          <dbl>      <dbl>  <dbl>
1 A_PLF_1     1 Reading        0.211      0.579     NA
2 A_PLF_1     1 Argumentation  7          9         NA
3 A_PLF_1     1 Narration     NA          6         NA
4 A_PLF_1     2 Reading        0.684      0.737     NA
```

```
5 A_PLF_1     2 Argumentation 14         13         NA
6 A_PLF_1     2 Narration     10          9         NA
7 A_PLF_1     3 Reading        0.947      0.842     NA
# i 3,992 more rows
```

Incidentally, not all values for `German` are `NA`. It's just that the first couple of children were tested in French and Portuguese, not in German. We can check this like so:

```
skills_wider_language |>
  filter(!is.na(German)) |>
  slice_sample(n = 10)
```

```
# A tibble: 10 x 6
  Subject   Time Skill         French Portuguese German
  <chr>    <dbl> <chr>          <dbl>      <dbl>  <dbl>
1 E_PLD_14     2 Narration         NA         11  5
2 V_CD_17      1 Reading           NA         NA  0.158
3 L_PLD_10     1 Argumentation     NA         11  3
4 O_PLD_9      3 Narration         NA         11  8
5 O_PLD_13     1 Argumentation     NA          9 11
6 V_CD_28      1 Argumentation     NA         NA  9
7 V_CD_30      3 Narration         NA         NA  6
# i 3 more rows
```

If we needed to, we could make this dataset wider still:

```
skills_wider_time_language <- skills_longer |>
  pivot_wider(names_from = c(LanguageTested, Time), # combo of Language, Time
              values_from = Score)
skills_wider_time_language
```

```
# A tibble: 1,410 x 11
  Subject  Skill French_1 Portuguese_1 French_2 Portuguese_2
  <chr>    <chr>    <dbl>        <dbl>    <dbl>        <dbl>
1 A_PLF_1  Read~    0.211        0.579    0.684        0.737
2 A_PLF_1  Argu~    7            9       14           13
3 A_PLF_1  Narr~   NA            6       10            9
4 A_PLF_10 Read~    0.579        0.316    0.474        0.579
5 A_PLF_10 Argu~    5            6       10            7
6 A_PLF_10 Narr~   10            7        8           NA
7 A_PLF_12 Read~    0.895       NA        1            0.947
# i 1,403 more rows
# i 5 more variables: French_3 <dbl>, Portuguese_3 <dbl>,
#   German_1 <dbl>, German_2 <dbl>, German_3 <dbl>
```

We could reconvert this tibble to a long one using the code snippet below. The code becomes a bit more complex:

- The notation `French_1:German_3` selects all columns between `French_1` and `German_3` (including). Alternatively, we could have selected the relevant columns using `starts_with(c("French", "Portuguese", "German"))`.

- A so-called **regular expression** (regex) is used to split up these column names into a `Language` bit and into a `Time` bit. The split happens at the first underscore (_) encountered.

```
skills_wider_time_language |>
  pivot_longer(cols = French_1:German_3,
               names_to = c("Language", "Time"),
               names_pattern = "(.*)_(.*)",
               values_to = "Score")

# A tibble: 12,690 x 5
  Subject Skill   Language   Time   Score
  <chr>   <chr>   <chr>      <chr>  <dbl>
1 A_PLF_1 Reading French     1      0.211
2 A_PLF_1 Reading Portuguese 1      0.579
3 A_PLF_1 Reading French     2      0.684
4 A_PLF_1 Reading Portuguese 2      0.737
5 A_PLF_1 Reading French     3      0.947
6 A_PLF_1 Reading Portuguese 3      0.842
7 A_PLF_1 Reading German     1     NA
# i 12,683 more rows
```

For the present example, we don't need this code snippet since we already have the tibble `skills_longer`. But I wanted to illustrate that such conversions are possible. If you ever need to convert a similar dataset to a longer format, you now know that you can look up the details on the help page of `pivot_longer()` (`?pivot_longer`) and take it from there.

Tip 2.15 (About regex)**.** With regular expressions, it's less important to *know* them than to know *about* them. Any time a string is comprised of several pieces of information in a more or less predictable way, regular expressions can be used to extract these pieces. That said, compiling regular expressions that actually do the job is pretty difficult. But once you're aware that they exist and can be used to such ends, you can enlist the help of AI tools to construct them. ♢

Incidentally, the regex used above (`(.*)_(.*)`) matches any two groups of characters (`(.*)`) preceding and following an underscore (_).

2.6 Summaries

Using the `summarise()` function, we can easily summarise large tibbles. For instance, we can compute the average (mean) narration and argumentation scores in the `skills` tibble like so:

```
skills |>
  summarise(mean_narr = mean(Narration, na.rm = TRUE),
            mean_arg  = mean(Argumentation, na.rm = TRUE))

# A tibble: 1 x 2
  mean_narr mean_arg
      <dbl>    <dbl>
1      8.51     13.0
```

We set the `na.rm` parameter in the `mean()` call to `TRUE` since there are several missing observations in both the `Narration` and `Argumentation` variable. If we didn't set `na.rm` to `TRUE`, the result of both computations would be `NA`. By setting `na.rm` to `TRUE`, missing observations are ignored.

`summarise()` is often used in conjuction with `group_by()`, which splits up a tibble into subgroups. This way, we can straightforwardly compute summaries for different subgroups. For instance, if we wanted to compute the mean narration and argumentation scores for each time of data collection and each language tested, we could use the following code snippet:

```
skills |>
  group_by(Time, LanguageTested) |>
  summarise(mean_narr = mean(Narration, na.rm = TRUE),
            mean_arg  = mean(Argumentation, na.rm = TRUE),
            .groups = "drop")

# A tibble: 9 x 4
   Time LanguageTested mean_narr mean_arg
  <dbl> <chr>              <dbl>    <dbl>
1     1 French              7.79    11.2
2     1 German              6.33     9.46
3     1 Portuguese          8.50    11.4
4     2 French              8.37    13.2
5     2 German              7.06    12.2
6     2 Portuguese          9.16    13.3
7     3 French             10.1     16.3
# i 2 more rows
```

By setting `.groups = "drop"`, we make sure that the grouping applied to `skills` doesn't apply to the summary tibble any more. For instance, compare the outcomes of these commands:

```
skills |>
  group_by(Time, LanguageTested) |>
  summarise(mean_narr = mean(Narration, na.rm = TRUE),
            mean_arg  = mean(Argumentation, na.rm = TRUE),
            .groups = "drop") |>
  summarise(grand_mean_narr = mean(mean_narr))
```

```
# A tibble: 1 x 1
  grand_mean_narr
            <dbl>
1            8.36
```

```
skills |>
  group_by(Time, LanguageTested) |>
  summarise(mean_narr = mean(Narration, na.rm = TRUE),
            mean_arg  = mean(Argumentation, na.rm = TRUE)) |>
  summarise(grand_mean_narr = mean(mean_narr))
```

```
# A tibble: 3 x 2
   Time grand_mean_narr
  <dbl>           <dbl>
1     1            7.54
2     2            8.20
3     3            9.36
```

We can treat these summary tibbles like ordinary tibbles and apply all of the other commands to them:

```
skills |>
  group_by(Time, LanguageTested) |>
  summarise(mean_narr = mean(Narration, na.rm = TRUE),
            .groups = "drop") |>
  pivot_wider(names_from = "Time", names_prefix = "T",
              values_from = "mean_narr")
```

```
# A tibble: 3 x 4
  LanguageTested    T1    T2    T3
  <chr>          <dbl> <dbl> <dbl>
1 French          7.79  8.37 10.1
2 German          6.33  7.06  7.68
3 Portuguese      8.50  9.16 10.2
```

We'll encounter several other functions that can meaningfully be used to summarise data in the chapters to come.

Remark 2.16 (Computing proportions and counts)**.** The `mean()` function can also be used to compute proportions. Consider the following example. The `is.na()` function checks if a value is `NA` (in which case it returns `TRUE`) or not (`FALSE`). If we compute the `mean()` of a bunch of `TRUE/FALSE` values, we obtain the proportion of values that are `TRUE`. Similarly, the `sum()` of a bunch of `TRUE/FALSE` values is the number of `TRUE` values. Hence, we can quickly obtain the proportion, and number, of the missing `Narration` scores for each combination of `Time` and `LanguageTested`:

```
skills |>
  group_by(Time, LanguageTested) |>
```

```
  summarise(
    prop_narr_NA = mean(is.na(Narration)),
    nr_narr_NA = sum(is.na(Narration)),
    n = n(),
    .groups = "drop"
  )

# A tibble: 9 x 5
   Time LanguageTested prop_narr_NA nr_narr_NA     n
  <dbl> <chr>                 <dbl>      <int> <int>
1     1 French               0.297          54   182
2     1 German               0.0815         15   184
3     1 Portuguese           0.141          40   284
4     2 French               0.120          22   183
5     2 German               0.0862         15   174
6     2 Portuguese           0.129          36   280
7     3 French               0.0632         11   174
# i 2 more rows
```

♢

2.7 String manipulation

It often happens that a single cell in a dataset contains different pieces of information. So, too, it is the case in our current example. The first participant in the dataset is referred to as `A_PLF_1`:

- The `A` in this ID refers to their class.
- The `PLF` tells us that this participant resided in French-speaking Switzerland (`F`), had a Portuguese background (`P`) and took Portuguese heritage and language courses (`L`).

  Other abbreviations in the dataset are `CD`, `CF`, and `CP` for comparison groups in German-speaking Switzerland, French-speaking Switzerland, and Portugal, respectively, `PLD` for participants residing in German-speaking Switzerland with a Portuguese background that attended a Portuguese course, as well as `PND` and `PNF` for participants in German- and French-speaking Switzerland, respectively, with a Portuguese background that did not take Portuguese classes.
- The `1` uniquely identifies this participant within their class.

It could make sense to split up the information contained in this one cell into multiple cells. Thankfully, the strings in `Subject` are structured logically and consistently: the different pieces of information are separated using underscores. We can hence split these strings at the underscores and retrieve the first and second pieces like so, obviating the need for more complicated regular expressions:

```
skills_wider_language <- skills_wider_language |>
  mutate(
    Class = str_split_i(Subject, "_", 1),
    Group = str_split_i(Subject, "_", 2)
  )
# check:
skills_wider_language |>
  select(Subject, Class, Group) |>
  sample_n(10)
```

```
# A tibble: 10 x 3
  Subject  Class Group
  <chr>    <chr> <chr>
1 A_PLF_9  A     PLF
2 Q_CF_7   Q     CF
3 J_PLD_8  J     PLD
4 E_PLD_5  E     PLD
5 S_CD_4   S     CD
6 J_PLD_8  J     PLD
7 AI_CP_19 AI    CP
# i 3 more rows
```

Refer to the `stringr` documentation for further guidance.

We can now further break down the information contained in the new `Group` column, for instance as follows:

```
skills_wider_language <- skills_wider_language |>
  mutate(language_group = case_when(
    Group == "CP" ~ "Portuguese control",
    Group == "CF" ~ "French control",
    Group == "CD" ~ "German control",
    Group %in% c("PLF", "PNF") ~ "French-Portuguese",
    Group %in% c("PLD", "PND") ~ "German-Portuguese",
    .default =  "other"
  )) |>
  mutate(heritage_course = case_when(
    Group %in% c("PLF", "PLD") ~ 1,
    .default =  0
  )) |>
  mutate(has_German = case_when(
    Group %in% c("CD", "PLD", "PND") ~ 1,
    .default =  0
  )) |>
  mutate(has_French = case_when(
    Group %in% c("CF", "PLF", "PNF") ~ 1,
```

```
    .default =  0
  )) |>
  mutate(has_Portuguese = case_when(
    Group %in% c("CF", "CD") ~ 0,
    .default =  1
  ))

# check:
skills_wider_language |>
  select(Subject, language_group, heritage_course,
         has_German, has_French, has_Portuguese) |>
  sample_n(10)
```

```
# A tibble: 10 x 6
  Subject  language_group     heritage_course has_German
  <chr>    <chr>                        <dbl>      <dbl>
1 AL_CP_15 Portuguese control               0          0
2 R_CF_5   French control                   0          0
3 T_CF_20  French control                   0          0
4 P_CF_11  French control                   0          0
5 V_CD_30  German control                   0          1
6 N_CD_11  German control                   0          1
7 V_CD_31  German control                   0          1
# i 3 more rows
# i 2 more variables: has_French <dbl>,
#   has_Portuguese <dbl>
```

2.8 A full-fledged example

We had the children in the French/German/Portuguese project write short narrative and argumentative texts in each of their languages at three points in time. The quality of these texts was scored using a grid (Desgrippes et al., 2017); it is these scores that are listed in the `skills` tibble we worked with above. In addition, 3,060 of these texts were rated for their lexical richness by between two and eighteen naïve (i.e., non-instructed) raters each using a 1-to-9 scale (Vanhove et al., 2019). The individual ratings are available in the file `helascot_ratings.csv`. Furthermore, we computed a bunch of lexical metrics for each text, such as the number of tokens[1], the mean corpus frequency of the words occurring in the text, etc. These metrics are available in the file `helascot_metrics.csv`; see Vanhove (2017b) for details.

We'll use these datasets to answer three questions:

[1]The number of tokens in a text is the total number of words it contains, counting repetitions. The number of types in a text is the number of distinct words, i.e., not counting repetitions.

- What's the relation between the average lexical richness ratings per text and the text's Guiraud index? (The Guiraud index is the ratio of the number of types in a text and the square root of the number of tokens in that text.)
- What's the relation between the average lexical richness ratings per text and the mean corpus frequency of the words occuring in the texts?
- What's the relation between the grid-based ratings and the lexical richness ratings?

In doing so, we'll need to make use of some of the tools introduced earlier.

2.8.1 Reading in the data

Let's start from a clean slate and read in the three datasets:

```
skills <- read_csv(here("data", "helascot_skills.csv"))
metrics <- read_csv(here("data", "helascot_metrics.csv"))
ratings <- read_csv(here("data", "helascot_ratings.csv"))
```

Inspect the structure of these three datasets. Note that

- `skills` contains one row per combination of `Subject`, `Time`, and `LanguageTested`;
- `metrics` contains one row per text, i.e., one row per combination of `Subject`, `Text_Language`, `Text_Type`, and `Time`;
- `ratings` contains one row per rating, i.e., one row per combination of `Rater` and `Text`.

These datasets are already pretty clean and a lot of work (partly manual, partly using R, partly using other tools) went into creating them. See the technical report as well as the R code available from `https://osf.io/vw4pc/`.

There are many ways to skin a cat. But it seems to me that the following course of action is reasonable:

1. Using `ratings`, compute the average `Rating` per text.
2. Add these average ratings to `metrics` and draw some plots.
3. Make `skills` longer and add the average ratings to it. Then draw some more plots.

2.8.2 Average rating per text

```
rating_per_text <- ratings |>
  group_by(Text, Subject, Text_Language, Text_Type, Time) |>
  summarise(mean_rating = mean(Rating),
            n_ratings = n(),
            .groups = "drop")
```

Exercise 2.17 (Filtering out bilingual raters)**.** Some of the raters consider themselves native speakers of several languages (`bi-French` etc.), others consider themselves monolingual native speakers (`mono-French` etc.):

```
table(ratings$Rater_NativeLanguage)


      bi-French       bi-German   bi-Portuguese
           1049            2241             728
    mono-French     mono-German mono-Portuguese
           6235           14149            4777
```

Compute the average text ratings based only on the ratings provided by monolingual French speakers. ♢

2.8.3 Add to `metrics`

The tibbles `metrics` and `rating_per_text` share a variable (`Text`), so the average ratings can easily be added to `metrics`:

```
metrics_ratings <- metrics |>
  left_join(rating_per_text)
```

We'll encounter some useful ways of plotting data throughout these lecture notes, but here's one way of visualising the relationship between `Guiraud` and `mean_rating`; see Figure 2.10 for the result.

```
metrics_ratings |>
  ggplot(aes(x = Guiraud, y = mean_rating)) +
  geom_point(shape = 1) +
  facet_grid(rows = vars(Time, Text_Type),
             cols = vars(Text_Language),
             scales = "free_y") +
  xlab("Guiraud value") +
  ylab("average lexical richness rating")
```

We can draw a similar plot involving `meanSUBTLEX` (average corpus frequency); the result is not shown here:

```
metrics_ratings |>
  ggplot(aes(x = meanSUBTLEX, y = mean_rating)) +
  geom_point(shape = 1) +
  facet_grid(rows = vars(Time, Text_Type),
             cols = vars(Text_Language),
             scales = "free_y") +
  xlab("mean SUBTLEX frequency") +
  ylab("average lexical richness rating")
```

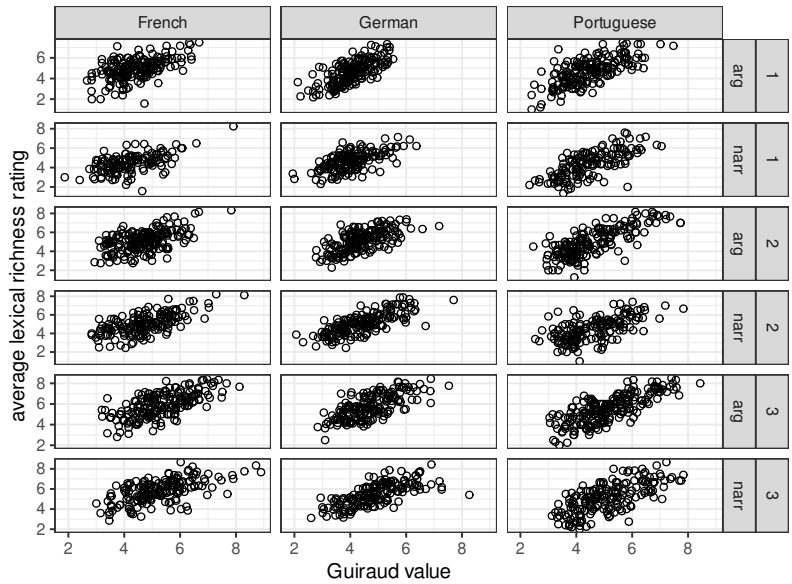


Figure 2.10: Association between the texts' Guiraud values and their average lexical richness ratings.

2.8.4 Add to `skills`

The code snippet below takes the `skills` tibble, transforms it so that each row represents one participant's grid-based score on one of the writing tasks, and then adds this information to the relevant row in the tibble containing the lexical richness ratings.

```
rating_gridscore <- skills |>
  pivot_longer(Reading:Narration, names_to = "skill",
               values_to = "grid_score") |>
  filter(skill != "Reading") |>
  mutate(Text_Type = case_when(
    skill == "Argumentation" ~ "arg",
    skill == "Narration" ~ "narr"
  )) |>
  full_join(rating_per_text,
            by = join_by(Text_Type, Subject, Time,
                         LanguageTested == Text_Language))
```

The relationship between the grid-based scores and the lexical richness ratings may be visualised as follows; the result is not shown here:

```
rating_gridscore |>
  filter(!is.na(grid_score)) |>
  filter(!is.na(mean_rating)) |>
  ggplot(aes(x = mean_rating, y = grid_score)) +
  geom_point(shape = 1) +
```

```
facet_grid(rows = vars(Time, Text_Type),
           cols = vars(LanguageTested),
           scales = "free_y") +
xlab("average lexical richness rating") +
ylab("grid-based text quality rating")
```

2.9 *Walkthrough with exercises

In the previous sections, we started from a pretty clean dataset containing the writing data. The goal of the present section is to give you some insight into how this data set was compiled starting from a more basic representation of the underlying data. This section is organised as a series of exercises that you should tackle in order. In completing these exercises, you'll convert a very wide dataset (with lots of criterion scores for up to twelve texts per child on a single row) into a dataset that contains a single composite score for a single text on each row. This process is considerably more complicated than running a couple of `pivot_*()` and `summarise()` commands due to how the data were laid out. Some of the instructions in the exercises are intentionally not spelt out in great detail: In real life, it's up to you to figure out the intermediate steps, too.

Exercise 2.18 (Reading in the data)**.** The file `writing_data.csv` contains the data reported on in Desgrippes et al. (2017). Importing it into R requires some additional work.

1. Open the file using Notepad (Windows), TextEdit (Mac) or whatever plain text editor you prefer. What character is used to separate different cells?

2. Try to read it in as a tibble named `writing`.

3. You *may* have obtained the following error message: `Error in nchar(x, "width") : invalid multibyte string, element 1`. Error messages can be notoriously opaque in R, but experience suggests that the accents and umlauts may have something to do with this error.

   To fix the error, try to read in the dataset again, but additionally override the `locale` argument of the function you used in Step 2 as follows: `locale = locale(encoding = "Windows-1252")`. This is a Windows character encoding for Western languages; consult the Wikipedia page on character encoding for an overview of common character encodings.

4. Inspect the resulting tibble. If all is well, it should contain 473 rows and 288 columns. Columns such as `Sco1R_Nb_mots` and `Sco1R_unitThemNbre` should be recognised as containing numbers (`<dbl>`, short for *double*, a format for representing numbers in computers), whereas columns such as `VPNID` and `Sco1R_unitThem` should be recognised as containing text strings (`<chr>`, short for *character*). You'll also observe that the first column wasn't named in the CSV file, so R assigned some meaningless name to it. ♢

Tip 2.19 (UTF-8)**.** When saving spreadsheets as CSV files, use the UTF-8 character encoding. In Excel, you can select the character encoding from a drop-down menu when

exporting the spreadsheet as a CSV; in LibreOffice.org Calc you'll be asked to specify how you want to save the CSV file exactly in a pop-up window. ♢

You can obtain the column names using `colnames(writing)` (not shown here). Unfortunately, I wasn't able to locate a technical report or codebook that outlines the meaning of all of these column names and what range of values they can take. That said, this project featured three waves of data collection in which children were, among other things, asked to write two kinds of text: argumentative and narrative. Most of the children were French–Portuguese or German–Portuguese bilinguals, and these were asked to write both kinds of texts in each of their languages. Other children were considered monolingual speakers of French, German or Portuguese, and these only wrote texts in their respective language. Each text was scored on a grid. Using `View(writing)`, we can observe the following:

- The second column (`VPNID`) contains the participants' identification codes. We already encountered these in Section 2.7. As you can verify for yourself, there is a single row in the dataset for each participant.
- Columns 154 (`hkngroup`), 155 (`Schulsprache`), 156 (`zweiSprachig`) and 157 (`Group`) seem to contain information regarding what type of participant we're dealing with. As we saw in Section 2.7, we can extract this information from the `VPNID` entries.
- Columns 1 (`...1`), 3 (`X.x`), 158 (`PartDef`) and 159 (`X.y`) don't seem to contain any relevant information. Perhaps they served some administrative purpose?
- All the other column names are built up according to a pattern:
  - Either the first letter is `P` or the first three letters are `Sco`. The `P` stands for 'Portuguese', the `Sco` for 'langue scolaire' (language of education).
  - After the initial letter(s), we find a number between 1 and 3. This number specifies the time of data collection (i.e., T1, T2 or T3).
  - After the number and before the first underscore, we find either the letter `A` or the letter `R`. From context, `A` means that we're dealing with an argumentative text, whereas `R` means that we're dealing with a narrative text (perhaps `R` is short for 'to relate'?).
  - Whatever comes after the first underscore specifies either a criterion on the rating grid or some further information about the text production.

The goal of the next few exercises is to reshape this dataset so that each row represents a single text.

Exercise 2.20 (Converting the dataset to a longer format)**.** Convert the `writing` tibble to a tibble named `writing_long` by following these steps:

1. Retain only the columns `VPNID` as well as all columns containing solely numerical information whose names start with either `Sco` or `P`.

Hint: Check out the examples on the help page for the `where()` function in the `tidyselect` package. This package is part of the tidyverse.

Hint: The resulting tibble contains 473 rows and 250 columns.

2. Convert the resulting tibble to a longer one containing just three columns: `VPNID`, `criterion` (containing the former column names) and `score` (containing the values in those former columns).

   Hint: The resulting tibble contains 117,777 rows. ♢

We continue with the `writing_long` tibble from the previous exercise. The `criterion` column contains a subset of the column names that were shown above. We want to parse these former column names and extract the following pieces of information from them: Language, time of data collection, text type, and criterion proper. For example, given the string `"Sco2R_some_criterion"`, we want to extract `"Sco"` as the language, `2` as the time of data collection, `"R"` as the text type, and `"some_criterion"` as the criterion name. Similarly, given the string `"P3A_OtherCriterion"`, we want to extract `"P"` as the language, `3` as the time of data collection, `"A"` as the text type, and `"OtherCriterion"` as the criterion name.

If you feed the description in the previous paragraph to some AI tool like ChatGPT, it will probably be able to come up with a suggestion as to how to go about this. After some prodding, it suggested the following solution to me:

```
my_string <- "Sco2R_some_criterion"
my_regex <- "([^\\d]+)(\\d)([^_]*)_(.*)$"
str_match(my_string, my_regex)
```

```
     [,1]                   [,2]  [,3] [,4]
[1,] "Sco2R_some_criterion" "Sco" "2"  "R"
     [,5]
[1,] "some_criterion"
```

```
my_string <- "P3A_OtherCriterion"
str_match(my_string, my_regex)
```

```
     [,1]                 [,2] [,3] [,4] [,5]
[1,] "P3A_OtherCriterion" "P"  "3"  "A"  "OtherCriterion"
```

Though it's not too important for the remainder of this walkthrough, the regex above is constructed in the following way:

- `([^\d]+)`: This group matches any group of one or more (`+`) non-numerical (`^` for 'non', `\d` for digits) characters.
- `(\d)` matches any digit.
- `([^_]*)` matches any group of zero or more (`*`) characters that aren't underscores.
- `(.*)` matches any group of zero or more characters without restrictions.

Evidently, it will be important to verify that the parsing was done correctly.

We can use `str_match()` to parse any number of strings using a regex. For instance, the command below parses two strings and outputs the result as a matrix:

```
my_strings <- c("Sco2R_some_criterion", "P3A_OtherCriterion")
my_regex <- "([^\\d]+)(\\d)([^_]*)_(.*)$"
str_match(my_strings, my_regex)
```

```
     [,1]                   [,2]  [,3] [,4]
[1,] "Sco2R_some_criterion" "Sco" "2"  "R"
[2,] "P3A_OtherCriterion"   "P"   "3"  "A"
     [,5]
[1,] "some_criterion"
[2,] "OtherCriterion"
```

If we want to just extract, say, the language information from these strings, we can select the second column of this matrix like so:

```
str_match(my_strings, my_regex)[, 2]
```

```
[1] "Sco" "P"
```

Similarly, we can extract the criterion name like so:

```
str_match(my_strings, my_regex)[, 5]
```

```
[1] "some_criterion" "OtherCriterion"
```

We can define a helper function `extract_info()` that bundles these steps:

```
extract_info <- function(my_strings, i,
                         my_regex = "([^\\d]+)(\\d)([^_]*)_(.*)$") {
  str_match(my_strings, my_regex)[, i]
}
extract_info(my_strings, 4)
```

```
[1] "R" "A"
```

```
extract_info(my_strings, 2)
```

```
[1] "Sco" "P"
```

Exercise 2.21 (Parsing the column names)**.** Using the `extract_info()` function just defined, add columns called `Language`, `Time`, `TextType` and `Criterion` (uppercase 'C') to the `writing_long` tibble. Evidently, these columns should contain the appropriate pieces of information. Then drop the `criterion` (lowercase 'c') column from the tibble. ♢

Exercise 2.22 (Splitting up the dataset)**.** Split the `writing_long` tibble up into two tibbles: `argumentative`, which contains only the data relating to argumentative texts, and `narrative`, which contains only the data relating to narrative texts.

If all went well, the `argumentative` tibble should contain 59,125 rows and 6 columns, and the `narrative` tibble 58,179 rows and 6 columns. ♢

While I wasn't able to retrieve a codebook for this dataset, Desgrippes et al. (2017) describe the grid according to which the texts were scored. I put the relevant information in the Excel file `scoring_criteria.xlsx`.

Exercise 2.23 (Obtaining the relevant scores for each argumentative text)**.** We now want to reorganise the data in such a way that we can easily calculate the total score assigned to each text. Since the scoring criteria were different for argumentative and narrative texts, we do this separately for both text types. Here, we'll only go through the steps for argumentative texts, but as an additional exercise, you could follow the same steps (*mutatis mutandis*) to calculate the total scores for the narrative texts. Importantly, we need to check whether the scores obtained actually make sense.

1. Inspect the `scoring_criteria.xlsx` file in your spreadsheet program. In particular, observe that it contains two sheets.

2. Read in the sheet containing the criteria for scoring the argumentatitive texts into R as `criteria_arg`.

3. In the tibble `argumentative`, only retain the rows where the value in the `Criterion` column is one of the scoring criteria for the argumentative texts (i.e., those contained in `criteria_arg`). (See the hint below.)

4. Drop the rows containing missing `score` values from `argumentative`.

5. Convert the resulting tibble to a wide format in which each row represents a single argumentative *text* and contains the scores on all relevant criteria for this text. Note that each combination of `VPNID`, `Language` and `Time` present in the dataset represents one text.

6. In principle, the current tibble should not contain any missing data. Moreover, the criterion scores should not exceed the `MaxScore` values that are listed in the file `scoring_criteria.xlsx`. Check if this is the case.

   Hint: It isn't.

   Inspect the rows in the dataset that contain abnormalities and figure out what went wrong.

   Hint: First use `summary(argumentative)` to identify the columns containing aberrant entries. Then inspect the rows containing such entries. Look up the corresponding information in `writing_data.csv`.

   Write a brief paragraph detailing what the issues are. Fix the issues *without* editing the spreadsheet. This may require you to go all the way back to the start of this section and make some modifications there.

7. Once all issues have been taken care of, reconvert the tibble to a longer format, i.e., with one row per criterion per text. Then, for each text, compute the variable `total_score_arg` as the sum of all criterion scores.

8. Using `write_csv()`, save the tibble containing the variables `VPNID`, `Language`, `Time` and `total_score_arg` to a CSV file in the `data` subdirectory.

Hint for step 3: One option is to use one of the `*_join()` functions. Another one is to combine `filter()` with the `%in%` operator. The following example shows how the `%in%` operator works.

```
x <- c("A", "B", "C") # define vectors x, y
y <- c("E", "C", "D", "A")
y %in% x # which of the values in y occur in x?

[1] FALSE  TRUE FALSE  TRUE

x %in% y # which of the values in x occur in y?

[1]  TRUE FALSE  TRUE
```

♢

Exercise 2.24 (Comparing the calculated scores with the scores used earlier)**.** We now have two sets of argumentative writing scores: the ones that you calculated in the previous exercise, and the ones we used earlier. Let's compare them.

1. Read in the scores you calculated from the CSV file you created in the previous exercise.

2. Read in the `helascot_skills.csv` file we used earlier.

3. Combine both files in such a way that, for each text, you have both the value of `total_score_arg` that you computed as well as the `Argumentation` value provided in the dataset `helascot_skills.csv` side-by-side. Make sure that the resulting tibble contains the scores of all texts, even in the case that a score is present for a text in one file but not in the other.

   Hint: You can't directly use one of the `*_join()` functions. You'll need to do some thinking and some minor data wrangling first.

4. Are there any texts for which an `Argumentation` score exists but no `total_score_arg` was computed? What is (are) the reason(s) for this (these) discrepancy(ies)?

5. Are there any texts for which a `total_scores_arg` value was computed but no `Argumentation` score exists? If so, output the relevant rows.

6. For each text, compute the difference between the `Argumentation` socre and the `total_score_arg` score. For how many texts is the difference not equal to 0? Can you think of any reason(s) for any such differences? ♢

Exercise 2.25 (Summarising the results)**.**

1. For each combination of language (i.e., French, German, Portuguese), group (i.e., bilingual vs. comparison group) and time of data collection, compute the number of texts for which you calculated a `total_score_arg` value as well as the mean `total_score_arg` value.

2. For each combination of language and group, compute the number of children who have `total_score_arg` values for *all* three data collections. How many children in each cell have `total_score_arg` scores for the first and second data collection but not for the third? How many for the first and third, but not for the second? For the second and third, but not for the first? How many children in each cell only have a `total_score_arg` score at one data collection? ♢

2.10 *Further reading

The Bible for working with tibbles in R is *R for Data Science* (Wickham et al., 2023), which is freely available from `https://r4ds.hadley.nz/`. Broman & Woo (2017) offer further valuable guidance for organising spreadsheets.

Chapter 3

Fundamentals of probability theory

If you want to analyse quantitative data, you need to know a thing or two about how probabilities work. This chapter introduces some concepts in probability theory that I think you need to know. Incidentally, I've chosen to include a few proofs of mathematical statements—not because you should be able to do such proofs yourself, but because I want to show that these statements are fairly straightforward consequences of the definitions rather than inscrutable edicts. For the statements that are more difficult to prove using high-school mathematics, illustrations rather than proofs are included.

3.1 Probability spaces

3.1.1 Events and probabilities

Jass is a family of Swiss card games that are played with a deck of 36 cards ranging from the 6 to the Ace in four suits (hearts, spades, diamonds, and clubs).[1] We consider two simple **random experiments**:

1. In the first experiment, we draw one random card from the deck.
2. In the second one, we draw two random cards from the deck, one after the other. We make note of the order in which we draw the cards.

The first experiment has 36 possible outcomes. The second one has $36 \cdot 35 = 1260$ possible outcomes as the first card drawn is one of 36 possible ones, whereas the second card drawn is one of the remaining 35.

We can formulate yes–no questions about the outcomes of both experiments. For the first experiment, some possible yes–no questions are the following:

- Is the card drawn the 7♣?
- Is the card drawn contained in the following list: $7\heartsuit, Q\diamondsuit, A\spadesuit$? (J = jack, Q = queen, K = king, A = ace.)
- Does the card drawn represent an even number?

[1] More arcane ranks and suits are used in some parts of Switzerland; we'll stick to an international deck.

- Does the card represent an even number or is it black (♠, ♣)?

In probability theory, and throughout these lecture notes, 'or' is always intended inclusively. That is, we'd also have to answer 'yes' to the fourth question if the card drawn were, say, the 8♣. For the exclusive 'or', we'll use such expressions as 'A or B, but not both'.

For the second experiment, some possible yes–no questions are the following:

- Is the first card drawn the 7♣ and the second the 8♡?
- Is one of the two cards drawn black?
- Is the rank of the first card lower than the rank of the second one?
- Have we drawn two Aces?

If we answer 'yes' to such a yes–no question, we say that the corresponding event occurred. In probability theory, outcomes and events are defined in set-theoretical terms.

Definition 3.1 (Outcomes and events)**.** An **outcome** is a possible result of a random experiment. (Here we don't further define what a random experiment is.)

The set of all outcomes of a random experiment is called the **sample space** of this random experiment. This sample space is often written as Ω.

An **event** is a subset of the sample space. In other words, an event is a set of outcomes. This set doesn't have to contain all outcomes, nor does it have to contain any outcome at all. The event that does not contain any outcomes is written as $\emptyset$ or as $\{\}$.

An event of the form $\{\omega\}$, with $\omega \in \Omega$ being an outcome, is known as an **elementary event** or as an **atomic event**.[2] $\diamondsuit$

In our first experiment, the sample space Ω_1 consists of 36 elements:

$$\begin{aligned}\Omega_1 = \{&6\clubsuit, 7\clubsuit, 8\clubsuit, 9\clubsuit, 10\clubsuit, J\clubsuit, Q\clubsuit, K\clubsuit, A\clubsuit,\\ &6\diamondsuit, 7\diamondsuit, 8\diamondsuit, 9\diamondsuit, 10\diamondsuit, J\diamondsuit, Q\diamondsuit, K\diamondsuit, A\diamondsuit,\\ &6\heartsuit, 7\heartsuit, 8\heartsuit, 9\heartsuit, 10\heartsuit, J\heartsuit, Q\heartsuit, K\heartsuit, A\heartsuit,\\ &6\spadesuit, 7\spadesuit, 8\spadesuit, 9\spadesuit, 10\spadesuit, J\spadesuit, Q\spadesuit, K\spadesuit, A\spadesuit\}.\end{aligned}$$

The yes–no question *Does the card represent an even number?* corresponds to the event

$$A := \{6\clubsuit, 8\clubsuit, 10\clubsuit, 6\diamondsuit, 8\diamondsuit, 10\diamondsuit, 6\spadesuit, 8\spadesuit, 10\spadesuit, 6\heartsuit, 8\heartsuit, 10\heartsuit\},$$

which contains 12 elements. The yes–no question *Is the card black?* corresponds to the event

$$B := \{6\clubsuit, 7\clubsuit, \ldots, A\clubsuit, 6\spadesuit, 7\spadesuit, \ldots, A\spadesuit\},$$

[2]The notation $x \in S$ means that x is an element of the set S; $x \notin S$ means that x is not an element of S. If S, T are two sets, then $S \subset T$ means that each element of S is also an element of T, and we say that S is a subset of T. This includes the possibility that S, T coindice.

which contains 18 elements. The yes–no question *Does the card represent an even number or is it black?* corresponds to the set-theoretical **union** of A and B; Figure 3.1 depicts the set-theoretical operations we use here:

$$\begin{aligned} A \cup B &= A \text{ or } B \\ &= \{\omega \in \Omega_1 : \omega \in A \text{ or } \omega \in B\} \\ &= \{6\clubsuit, 7\clubsuit, \dots, A\clubsuit, 6\spadesuit, 7\spadesuit, \dots, A\spadesuit, 6\diamondsuit, 8\diamondsuit, 10\diamondsuit, 6\heartsuit, 8\heartsuit, 10\heartsuit\}. \end{aligned}$$

The event $A \cup B$ contains 24 outcomes. By contrast, the yes–no question *Does the card represent an even number* and *is it black?* corresponds to the set-theoretical **intersection** of A and B:

$$\begin{aligned} A \cap B &= A \text{ and } B \\ &= \{\omega \in \Omega_1 : \omega \in A \text{ and } \omega \in B\} \\ &= \{6\clubsuit, 8\clubsuit, 10\clubsuit, 6\spadesuit, 8\spadesuit, 10\spadesuit\}. \end{aligned}$$

The yes–no question *Does the card* not *show an even number?* corresponds to the set-theoretical **complement** of A:

$$\begin{aligned} A^c &= \text{not } A \\ &= \{\omega \in \Omega_1 : \omega \notin A\}. \end{aligned}$$

We answer 'yes' to such yes–no questions if and only if the observed outcome ω is contained in the event E corresponding to this question, that is, if and only if $\omega \in E$.

In the second random experiment, the sample space Ω_2 consists of the 1,260 tuples that consist of two different cards. The yes–no question *Have we drawn two Aces* then corresponds to the event

$$\begin{aligned} C = \{&(A\clubsuit, A\diamondsuit), (A\clubsuit, A\heartsuit), (A\clubsuit, A\spadesuit), (A\diamondsuit, A\clubsuit), (A\diamondsuit, A\heartsuit), (A\diamondsuit, A\spadesuit), \\ &(A\heartsuit, A\clubsuit), (A\heartsuit, A\diamondsuit), (A\heartsuit, A\spadesuit), (A\spadesuit, A\clubsuit), (A\spadesuit, A\diamondsuit), (A\spadesuit, A\heartsuit)\}. \end{aligned}$$

Definition 3.2 (Countability)**.** A set is **countable** if its elements can be enumerated in a finite or infinite list. A set is **uncountable** if it is not countable. $\diamondsuit$

Example 3.3. Sets with a finite number of elements are countable.

The set of natural numbers

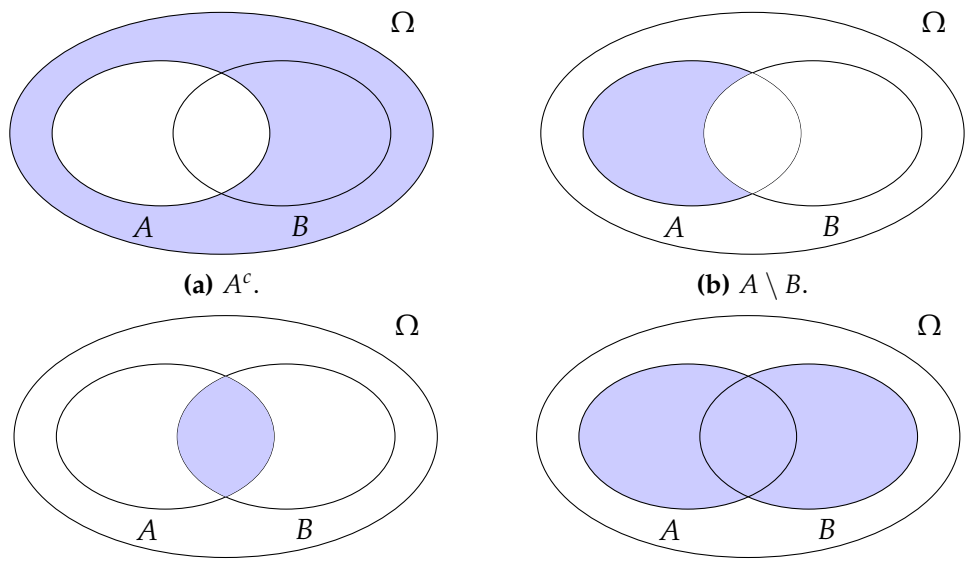
$$\mathbb{N} = \{0, 1, 2, \dots\}$$

is infinite but can be enumerated in an infinite list $(0, 1, 2, 3, \dots)$. Hence, the natural numbers are countable.

The integers

$$\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$$

can similarly be enumerated $(0, -1, 1, -2, 2, \dots)$ and are hence countable.



(a) A^c. **(b)** $A \setminus B$. **(c)** $A \cap B$. **(d)** $A \cup B$.

Figure 3.1: The complement of A ('not A', A^c); the relative complement of B in A ('A, but not B', $A \setminus B$); the intersection of A and B ('A and B', $A \cap B$); and the union of A and B ('A or B', $A \cup B$).

Even the rational numbers

$$\mathbb{Q} = \{a/b : a \in \mathbb{Z}, b \in \mathbb{N}, b \neq 0\}$$

can be enumerated:

$$\begin{aligned}\mathbb{Q} = \{&0/1, 1/1, -1/1, 1/2, -1/2, 2/1, -2/1, 1/3, -1/3, 3/1, -3/1,\\ &1/4, -1/4, 2/3, -2/3, 4/1, -4/1, 3/2, -3/2, \dots\}.\end{aligned}$$

The real numbers $\mathbb{R}$, however, can provably not be enumerated and are hence uncountable. $\diamondsuit$

Definition 3.4 (Discrete probability space)**.** A **discrete probability space** consists of a countable sample space Ω and a mapping (function) $\mathbb{P}$ that assigns to each event a number. This mapping specifies the **probability** with which each event occurs. To that end, it needs to fulfil the following criteria (axioms):

1. For each event E, we have $\mathbb{P}(E) \geq 0$. That is, probabilities are non-negative.
2. $\mathbb{P}(\Omega) = 1$. That is, the probability of observing *some* outcome is 1.
3. Let $A_1, A_2, A_3, \dots$ be an enumeration of events. If these events are pairwise disjunct, that is, if no outcome occurs in more than one of these events, then, the probability that *any* of the listed events occurs is the sum of the probabilities of the

individual events. In symbols: If for each $i \neq j$, we have $A_i \cap A_j = \emptyset$, then

$$\mathbb{P}\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} \mathbb{P}(A_i). \qquad \diamondsuit$$

Lemma 3.5 (Calculating with probabilities)**.** From the criteria that the mapping $\mathbb{P}$ has to satisfy, we can derive a few important rules for calculating with probabilities; refer to Figure 3.1 for the set-theoretical operations referred to.

1. For each event A, we have $A \cup A^c = \Omega$ and $A \cap A^c = \emptyset$. Hence,

   $$1 = \mathbb{P}(\Omega) = \mathbb{P}(A \cup A^c) = \mathbb{P}(A) + \mathbb{P}(A^c).$$

   So

   $$\mathbb{P}(A^c) = 1 - \mathbb{P}(A).$$

   Since $\Omega^c = \emptyset$, we have in particular that $\mathbb{P}(\emptyset) = 0$.

2. For two events A, B, we denote by $A \setminus B$ the event consisting of the outcomes occurring in A but not in B:

   $$A \setminus B = \{\omega \in \Omega : \omega \in A, \omega \notin B\}.$$

   We have

   $$A \cup B = (A \setminus B) \cup (B \setminus A) \cup (A \cap B),$$

   where $A \setminus B, B \setminus A, A \cap B$ are pairwise disjunct. Hence,

   $$\mathbb{P}(A \cup B) = \mathbb{P}((A \setminus B) \cup (B \setminus A) \cup (A \cap B)) = \mathbb{P}(A \setminus B) + \mathbb{P}(B \setminus A) + \mathbb{P}(A \cap B).$$

   Note that $A = (A \setminus B) \cup (A \cap B)$ and $B = (B \setminus A) \cup (A \cap B)$. Consequently,

   $$\begin{aligned}\mathbb{P}(A \cup B) &= \mathbb{P}(A \setminus B) + \mathbb{P}(B \setminus A) + \mathbb{P}(A \cap B)\\ &= \mathbb{P}(A) - \mathbb{P}(A \cap B) + \mathbb{P}(B) - \mathbb{P}(A \cap B) + \mathbb{P}(A \cap B)\\ &= \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(A \cap B)\\ &\leq \mathbb{P}(A) + \mathbb{P}(B).\end{aligned}$$

3. In discrete probability spaces, each event can be written as the union of elementary events. These elementary events are pairwise disjunct. As a result, the probability of each event is determined entirely by the probabilities of the elementary events. $\diamondsuit$

Example 3.6 (Discrete uniform distribution on Ω_1)**.** Consider the first random experiment. Assume that each card has the same probability of being drawn, that is, $\mathbb{P}(\{\omega\}) = 1/36$ for each $\omega \in \Omega_1$. In this case, we say that $\mathbb{P}$ is a **uniform distribution** on Ω_1.

Event A ('The card show an even number') consists of 12 outcomes and hence occurs with probability $12 \cdot 1/36 = 1/3$. Event B ('The card is black') consists of 18 outcomes, so $\mathbb{P}(B) = 18/36 = 1/2$. The intersection of both events, $A \cap B$, consists of 6 outcomes,

hence $\mathbb{P}(A \cap B) = 6/36 = 1/6$. Consequently, $\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(A \cap B) = 2/3$. ♢

Example 3.7 (Zipf distibution)**.** Consider a language with a finite vocabulary Ω. We denote the words in the language as $\omega_1, \omega_2, \ldots, \omega_n$ in decreasing order of the frequency with which they occur. We assume that these frequencies of occurrence are pairwise different. According to the Zipf model, the relative frequency of occurrence of a word in this language is approximately inversely proportional to its frequency rank, that is,

$$\mathbb{P}(\{\omega_k\}) = C_n \frac{1}{k}$$

for some constant C_n that depends on n, for all $k = 1, \ldots, n$. We can compute the constant C_n as follows. According to the axioms of probability, we require

$$1 = \mathbb{P}(\Omega) = \mathbb{P}(\bigcup_{k=1}^{n} \{\omega_k\}) = \sum_{k=1}^{n} C_n \frac{1}{k} = C_n \sum_{k=1}^{n} \frac{1}{k}.$$

So

$$C_n = \frac{1}{\sum_{k=1}^{n} 1/k}.$$

For $n = 10$, the Zipf model predicts the following relative frequencies:

```
n <- 10
words <- 1:n
Hn <- sum(1/words)
tibble(Word = words,
       rel.Frequency  = 1/Hn * 1/words)
```

```
# A tibble: 10 x 2
   Word rel.Frequency
  <int>         <dbl>
1     1        0.341
2     2        0.171
3     3        0.114
4     4        0.0854
5     5        0.0683
6     6        0.0569
7     7        0.0488
# i 3 more rows
```

Incidentally, we need to assume a finite vocabulary since $\sum_{k=1}^{\infty} \frac{1}{k}$ diverges to infinity. The Zipf model can be generalised to deal with infinite vocabularies, though. ♢

Example 3.8 (An infinite discrete probability space)**.** Imagine a random number generator that outputs each number $k = 1, 2, 3, \ldots$ with probability $1/2^k$. The sample space consists of all positive natural numbers $k \geq 1$. Since

$$\sum_{k=1}^{\infty} \mathbb{P}(\{k\}) = \sum_{k=1}^{\infty} \frac{1}{2^k} = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots = 1,$$

this is a permissible discrete probability space. The probability of observing an even number is

$$\sum_{k=1}^{\infty} \mathbb{P}(\{2k\}) = \sum_{k=1}^{\infty} \frac{1}{2^{2k}} = \sum_{k=1}^{\infty} \frac{1}{4^k} = \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \cdots = \frac{1}{3}. \qquad \diamondsuit$$

Discrete probability spaces are fairly easy to describe as they are characterised completely by the elementary events and the probabilities assigned to these. For many applications, though, discrete probability spaces aren't suitable since it is more sensible to conceive of the sample space as the set of real numbers ($\mathbb{R}$), which is uncountable. When we're dealing with uncountable sample spaces, though, it turns out that we can no longer allow all sets of outcomes to be events that we can assign a probability to. We'd have to go into the weeds of the mathematical field of measure theory to explain the reasons why, which we won't do; interested readers can use the search terms 'measurable set' and 'σ-algebra'.

In practical terms, fortunately, this problem isn't too serious: Sets of outcomes that don't represent permissible events are pretty difficult to construct, and they aren't relevant to us. For this reason, the following definition doesn't specify the criteria that needs to be fulfilled by the family of events to which we want to assign probabilities.

Definition 3.9 (General probability spaces)**.** A (general) **probability space** consists of (1) a (countable or uncountable) sample space Ω, (2) a family of events to which probabilities are assigned, and (3) a mapping $\mathbb{P}$ that assigns probabilities to these events. The mapping $\mathbb{P}$ has the same properties as in Definition 3.4. $\diamondsuit$

Example 3.10 (Continuous uniform distribution)**.** Consider a fortune wheel, the boundary of which is labelled with numbers between 0 and 360 (exclusive) as in Figure 3.2. Each time the arrow is turned, it points to a random spot along the fortune wheel's circumference. We assume that we can read off the number the wheel lands on with arbitrary precision.

In this example, the sample space consists of all real numbers in the interval $[0, 360)$. This set is uncountable. Nonetheless, we can assign probabilities to all events to which we reasonably want to assign probabilities. For instance, the probability that the arrow lands somewhere between the numbers a and b, $0 \leq a \leq b < 360$, is proportional to the length of the interval $[a, b)$, that is, to $b - a$. Since $\mathbb{P}([0, 360)) = 1$ is required, it follows that

$$\mathbb{P}([a, b)) = \frac{b - a}{360}.$$

Thus, for instance, the probability that the arrow lands between 45 and 93 is $\frac{93-45}{360} \approx 0.133$.

Somewhat counterintuitively, perhaps, the probability that an arrow lands *exactly* on any specific value $a \in [0, 360)$ is exactly 0. This is true for all $a \in [0, 360)$. The reason is that a single point has length 0. The third property of $\mathbb{P}$ now implies that the probability that the arrow lands on *any* point occurring in an infinite list is also exactly 0. Thus, for

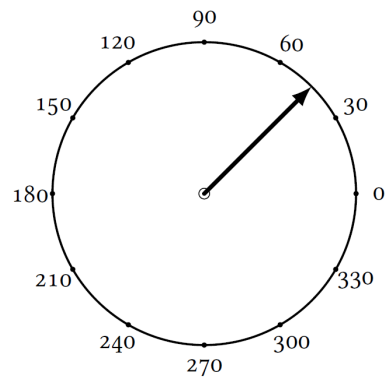


Figure 3.2: A fortune wheel.

instance,

$$\mathbb{P}(\{1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \dots\}) = 0.$$

This does not clash with the requirement that $\mathbb{P}([0, 360)) = 1$: The interval $[0, 360)$ cannot be written as an infinite list.

A consequence of the fact that, in this example, individual points have probability 0, is that we may add or remove bounds to or from intervals without any consequence:

$$\mathbb{P}([a, b]) = \mathbb{P}((a, b)) = \mathbb{P}([a, b)) = \mathbb{P}((a, b]). \qquad \diamondsuit$$

3.1.2 Independence

Definition 3.11 (Independence (1))**.** We say that two events A, B are **independent** if

$$\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B).$$

Less formally, we can also say that A, B are independent *of each other* or that A is independent of B and vice versa. $\diamondsuit$

***Example 3.12.** The event $\emptyset$ is independent of all other events. To see this, let B be an arbitrary event. Since $\emptyset \cap B = \emptyset$, it follows that

$$\mathbb{P}(\emptyset \cap B) = \mathbb{P}(\emptyset) = 0 = 0 \cdot \mathbb{P}(B) = \mathbb{P}(\emptyset)\mathbb{P}(B).$$

The event Ω is also independent of all other events: Since for any event B, we have $\Omega \cap B = B$, it follows that

$$\mathbb{P}(\Omega \cap B) = \mathbb{P}(B) = 1 \cdot \mathbb{P}(B) = \mathbb{P}(\Omega)\mathbb{P}(B). \qquad \diamondsuit$$

Example 3.13. The events A (*The card shows an even number.*) and B (*The card is of a black suit*) from the first random experiment are independent since

$$\mathbb{P}(A \cap B) = \frac{6}{36} = \frac{12}{36} \cdot \frac{18}{36} = \mathbb{P}(A)\mathbb{P}(B).$$

The events $A \cup B$ and $A \cap B$, by contrast, are not independent since

$$\mathbb{P}((A \cup B) \cap (A \cap B)) = \mathbb{P}(A \cap B) = \frac{1}{6} \neq \frac{2}{3} \cdot \frac{1}{6} = \mathbb{P}(A \cup B)\mathbb{P}(A \cap B). \qquad \diamondsuit$$

Exercise 3.14. Consider the second random experiment. Are the events *The first card is an Ace* and *The second card is an Ace* independent? Justify your answer based on Definition 3.11. $\diamondsuit$

Exercise 3.15. Let F, G be some events with $\mathbb{P}(F) = 0.6, \mathbb{P}(G) = 0.2$ and $\mathbb{P}(F \cup G) = 0.72$. Are F, G independent? Justify your answer based on Definition 3.11. $\diamondsuit$

Definition 3.16 (Independence (2))**.** We say that n events $A_1, \ldots, A_n$ are **pairwise independent** if A_i, A_j are independent for all $1 \leq i < j \leq n$.

We say that n events $A_1, \ldots, A_n$ are **independent** if, for each subset $I \subset \{1, \ldots, n\}$, we have

$$\mathbb{P}\left(\cap_{i \in I} A_i\right) = \prod_{i \in I} \mathbb{P}(A_i).$$

In words: For *every* choice of at most n events from among the list, the probability that all chosen events occur simultaneously equals the product of the probabilities with which they occur individually. Independence implies pairwise independence. To see this, consider all subsets I with two different events. $\diamondsuit$

***Example 3.17.** We define the probability space consisting of the sample space

$$\Omega := \{(0,0,0), (0,1,1), (1,0,1), (1,1,0)\}$$

and a discrete uniform distribution, i.e., $\mathbb{P}(\{\omega\}) = 1/4$ for all $\omega \in \Omega$. We consider the events $A :=$ *The first number is 1*, $B :=$ *The second number is 1*, and $C :=$ *The third number is 1*. As you can verify, A, B, C are pairwise independent. Nonetheless,

$$\mathbb{P}(A \cap B \cap C) = 0 \neq \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \mathbb{P}(A)\mathbb{P}(B)\mathbb{P}(C).$$

Hence, A, B, C are not independent. $\diamondsuit$

We only need the next definition so that we can make sense of the hypotheses for the Central Limit Theorem, which we'll encounter at the end of this chapter.

Definition 3.18 (Independence (3))**.** A infinite family of events is **independent** if each of its finite subfamilies is independent as per Definition 3.16. $\diamondsuit$

3.1.3 Conditional probabilities

Definition 3.19 (Conditional probability)**.** Let A, B be events such that $\mathbb{P}(B) > 0$. We call

$$\mathbb{P}(A|B) := \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}$$

the **conditional probability of** A **given** B. The conditional probability of A given B expresses the probability that A occurs if you already know that B has occurred. $\diamondsuit$

Remark 3.20. If A, B are independent with $\mathbb{P}(B) > 0$, then

$$\begin{aligned}\mathbb{P}(A|B) &= \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)} && \text{[Definition 3.19]}\\ &= \frac{\mathbb{P}(A)\mathbb{P}(B)}{\mathbb{P}(B)} && \text{[Definition 3.11]}\\ &= \mathbb{P}(A).\end{aligned}$$

This meshes with the intuition that if A, B are independent, then knowing whether B has occurred or not doesn't tell you anything new about whether A will occur. ♢

Example 3.21. In the fortune wheel example, we may wonder about the probability that the arrow lands somewhere between 90 and 270 ('A') assuming that it landed between 0 and 120 ('B'). We have $\mathbb{P}(B) = (120 - 0)/360 = 1/3$ and $\mathbb{P}(A \cap B) = (120 - 90)/(360) = 1/12$. Hence

$$\mathbb{P}(A|B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)} = \frac{1/12}{1/3} = \frac{1}{4}.$$

Since $\mathbb{P}(A) = (270 - 90)/360 = 1/2$, $\mathbb{P}(A|B) \neq \mathbb{P}(A)$. So A, B are not independent. ♢

The following theorem is useful for partioning a complex event into simpler events.

Theorem 3.22 (Total probability)**.** Let $E_1, E_2, \ldots$ be pairwise disjunct events that together comprise Ω, i.e., $E_1 \cup E_2 \cup \cdots = \Omega$. Assume that $\mathbb{P}(E_k) > 0$ for all $k = 1, 2, \ldots$. Then, for each event A,

$$\mathbb{P}(A) = \sum_{k=1}^{\infty} \mathbb{P}(A|E_k)\mathbb{P}(E_k).$$ ♢

Proof. We have

$$\begin{aligned}\mathbb{P}(A) &= \mathbb{P}(A \cap \Omega)\\ &= \mathbb{P}(A \cap \bigcup_{k=1}^{\infty} E_k)\\ &= \mathbb{P}\left(\bigcup_{k=1}^{\infty} (A \cap E_k)\right)\\ &= \sum_{k=1}^{\infty} \mathbb{P}(A \cap E_k)\\ &= \sum_{k=1}^{\infty} \mathbb{P}(A|E_k)\mathbb{P}(E_k).\end{aligned}$$ □

The above theorem, with the same proof, is also true if the list of pairwise disjunct events is finite.

***Example 3.23.** Let's consider once more the second random experiment. The discrete uniform distribution on Ω_2 assigns probability $1/1260$ to each elementary event

$\{(\omega_1, \omega_2)\}, (\omega_1, \omega_2) \in \Omega_2$. The probability that the first card drawn is the 6♣ and the second is not a 6 is

$$\frac{1}{36} \cdot \frac{36-4}{35} \approx 0.0254.$$

Consequently, the probability that the first card is a 6 and the second one isn't is

$$4\left(\frac{1}{36} \cdot \frac{36-4}{35}\right) \approx 0.1016.$$

Similarly, we may compute the probability that the first card is some 7 and that the second's rank is greater than 7:

$$4\left(\frac{1}{36} \cdot \frac{36-8}{35}\right) \approx 0.0889.$$

The events *The first card is a 6*, *The first card is a 7*, ..., are pairwise disjunct and together comprise the entire sample space. Hence, we may use the total probability theorem to compute the probability that the first card's rank is lower than the second card's rank:

$$\sum_{k=1}^{9} 4\left(\frac{1}{36} \cdot \frac{36-4k}{35}\right) = \frac{4}{36 \cdot 35} \sum_{k=1}^{9} (36-4k) = \frac{4}{36 \cdot 35}(9 \cdot 36 - (4 + 8 + \cdots + 36)).$$

We can calculate the result in R:

```
4/(36*35) * (9*36 - sum(4*seq(1, 9)))

[1] 0.4571429
```

That is, about 46%.

There's a more elegant way to arrive at the same solution. The probability that both cards have the same rank is 3/35. So the probability that the card differ in rank is 32/35. It is then clear that the probability that the first card is of lower rank than the second equals the probability that the first card is of greater rank that the second. Hence, the probability that the first card is of lower rank than the second rank is $(32/35)/2 \approx 0.4571$. ♢

Exercise 3.24 (M&Ms)**.** M&Ms come in six different colours; Table 3.1 on the following page lists their relative frequencies. When answering the following questions, assume that the M&Ms are drawn independently from an infinite population of M&Ms.

1. What's the probability that a randomly drawn M&M is red or orange?
2. What's the probability that, when you randomly draw two M&Ms, both are red or orange? (That is, that both are red, both are orange, or one of them is red and the other is orange.)
3. What's the probability that, when you randomly draw two M&Ms, one is red and the other orange?
4. What's the probability that, when you randomly draw 5 M&Ms, all of them are blue?

Table 3.1: Relative frequencies of occurrence of M&Ms by colour.

Colour	Relative frequency
Blue	23%
Orange	23%
Yellow	15%
Green	15%
Brown	12%
Red	12%

5. What's the probability that, when you randomly draw 5 M&Ms, none of them is blue? ♢

Both in scientific and in societal discussions, the conditional probabilities $\mathbb{P}(A|B)$ and $\mathbb{P}(B|A)$ are, unfortunately, often conflated. The classic example below and the theorem following it tell you how $\mathbb{P}(A|B)$ and $\mathbb{P}(B|A)$ can be converted into one another.

Example 3.25 (Medical screening)**.** Imagine that all newly born infants are subjected to a medical test that screens for a rare genetic disease. This genetic disease is thought to affect about 0.01% of newborns. The test labels newborns affected with the disease as affected with probability 97%. However, 1% of newborns that are not affected by the disease will also be labelled as affected with the disease. If the test labels a newborn as affected by the disease, what's the probability that the newborn actually is affected by the disease?

To answer this question, we can consider a large number of newborns, for instance, one million of them. Of these, about $0.0001 \cdot 10^6 = 100$ are expected to be affected by the disease; the remaining 999,900 aren't affected. Of these 100 affected children, 97 are expected to also be labelled as affected; the disease won't be flagged immediately in the remaining three ones. Of the 999,900 non-affected children, $0.01 \cdot 999900 = 9999$ are expected to be falsely labelled as affected. If we randomly pick a child from the $97 + 9999 = 10096$ children that are expected to be labelled as affected, the probability that this child does indeed carry the disease is only $97/10096 \approx 0.0096$, that is, not even 1%. ♢

The procedure from the previous example can be applied generally, as the famous Bayes' theorem shows.

Theorem 3.26 (Bayes)**.** Let A, B be events with $\mathbb{P}(A), \mathbb{P}(B) > 0$. Then

$$\mathbb{P}(B|A) = \frac{\mathbb{P}(A|B)\mathbb{P}(B)}{\mathbb{P}(A|B)\mathbb{P}(B) + \mathbb{P}(A|B^c)(1 - \mathbb{P}(B))}. \qquad \diamondsuit$$

Proof. We have

$$\begin{aligned}
\mathbb{P}(B|A) &= \frac{\mathbb{P}(B \cap A)}{\mathbb{P}(A)} && \text{[Definition 3.19]} \\
&= \frac{\mathbb{P}(A|B)\mathbb{P}(B)}{\mathbb{P}(A)} && [\mathbb{P}(A|B) = \mathbb{P}(A \cap B)/\mathbb{P}(B)] \\
&= \frac{\mathbb{P}(A|B)\mathbb{P}(B)}{\mathbb{P}(A|B)\mathbb{P}(B) + \mathbb{P}(A|B^c)(1 - \mathbb{P}(B))}. && \text{[total probability]}
\end{aligned}$$

□

In terms of Example 3.25, B would be *The child is affected* and A *The child is labelled as affected*. Then $\mathbb{P}(A|B) = 0.97, \mathbb{P}(B) = 0.0001, \mathbb{P}(A|B^c) = 0.01$. So, by Bayes' theorem,

$$\mathbb{P}(B|A) = \frac{0.97 \cdot 0.0001}{0.97 \cdot 0.0001 + 0.01 \cdot (1 - 0.0001)} \approx 0.0096.$$

3.2 Random variables

When analysing quantitative data, we often model these as observations of random variables.

Definition 3.27 (Random variables)**.** Let Ω be the sample space of a probability space. A **random variable** X maps each outcome $\omega \in \Omega$ to a real number.

If the set of possible values of X is countable, we call X a **discrete random variable**. ♢

Example 3.28. In *Jass*, we can consider the random variable X that maps each card to its point value in the *Obenabe* mode of play, that is,

$$X(\omega) := \begin{cases} 11, & \text{if } \omega \text{ is an Ace,} \\ 4, & \text{if } \omega \text{ is a King,} \\ 3, & \text{if } \omega \text{ is a Queen,} \\ 2, & \text{if } \omega \text{ is a Jack,} \\ 10, & \text{if } \omega \text{ is a 10,} \\ 8, & \text{if } \omega \text{ is an 8,} \\ 0, & \text{else.} \end{cases}$$

This random variable is one of arbitrarily many you can define on Ω_1.

If we draw two cards, you could define a random variable that represents, for instance, the sum of the point values of both cards. ♢

We can also consider the values that a random variable can take on as outcomes on their own. This way, we obtain a new sample space

$$\widetilde{\Omega} := \{X(\omega) : \omega \in \Omega\}.$$

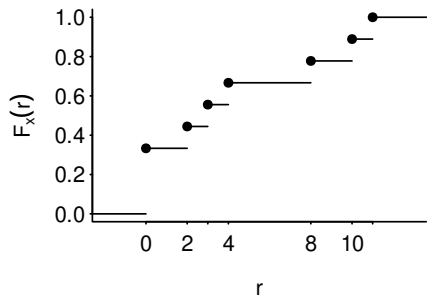


Figure 3.3: Distribution function of the point value of *Jass* cards in the *Obenabe* mode of play.

For the random variable X, we'd have

$$\widetilde{\Omega}_1 = \{0, 2, 3, 4, 8, 10, 11\}.$$

The probability that $X = 11$ is $4/36 = 1/9$. We write $\mathbb{P}(X = 11) = 1/9$. Similarly, $\mathbb{P}(X = 0) = 1/3$. We can also discuss events, for instance $\mathbb{P}(X \geq 10) = 2/9$.

Let's take a closer look at the most important representations and properties of random variables.

3.2.1 The distribution function and the quantile function

Random variables can be characterised by their **distribution function**.

Definition 3.29 (Distribution function)**.** Let X be a random variable. The distribution function F_X of X is defined by

$$F_X(r) := \mathbb{P}(X \leq r)$$

for all real numbers r. ♢

By way of example, consider random variable defined in Example 3.28. We can construct the following table with cumulative probabilities:

Value	0	2	3	4	8	10	11
Probability	0.333	0.111	0.111	0.111	0.111	0.111	0.111
Cumulative probability	0.333	0.444	0.555	0.666	0.777	0.888	1

Thus, for instance, $F_X(-1) = 0, F_X(2.5) = 0.444$ and $F_X(12) = 1$. Figure 3.3 shows the distribution function of X.

If we know the distribution function of a random variable, we can compute the probability that the random variable lies in some interval:

$$\mathbb{P}(X \in (-\infty, b]) = \mathbb{P}(X \leq b) = F_X(b),$$
$$\mathbb{P}(X \notin (-\infty, a]) = \mathbb{P}(X > a) = 1 - F_X(a).$$

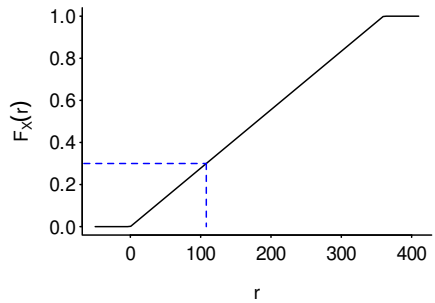


Figure 3.4: Distribution function of a random variable with a continuous uniform distribution on $[0, 360)$.

So

$$\mathbb{P}(X \in (a, b]) = F_X(b) - F_X(a).$$

We need to be somewhat more cautious when making probability claims about intervals of the form $[a, b]$, $[a, b)$ and (a, b). The reason is that the end points of such intervals may have some positive probability. We therefore define a further function:

$$F_X(r-) := \mathbb{P}(X \in (-\infty, r)) = F_X(r) - \mathbb{P}(X = r).$$

Consequently,

$$\begin{aligned}
\mathbb{P}(X \in [a, b]) &= F_X(b) - F_X(a-) = F_X(b) - F_X(a) + \mathbb{P}(X = a), \\
\mathbb{P}(X \in [a, b)) &= F_X(b-) - F_X(a-) = F_X(b) - F_X(a) - \mathbb{P}(X = b) + \mathbb{P}(X = a), \\
\mathbb{P}(X \in (a, b)) &= F_X(b-) - F_X(a) = F_X(b) - F(X)(a) - \mathbb{P}(X = b).
\end{aligned}$$

Whereas the distribution function F_X tells us the probability that a random value assumes a value no larger than some number r, the **quantile function** F_X^{-1} tells us the inverse: Given some number $p \in (0, 1)$, what is the lowest value q such that $F_X(q) \geq p$?

Definition 3.30 (Quantile function)**.** Let X be a random variable. Its quantile function F_X^{-1} is defined as

$$F_X^{-1}(p) := \min\{q \in \mathbb{R} : F_X(q) \geq p\}. \qquad \diamondsuit$$

From Figure 3.3, we may, for instance, glean that $F_X^{-1}(0.4) = 2$ and $F_X^{-1}(7/9) = 8$.

Example 3.31 (Continuous uniform distribution)**.** Consider the fortune wheel from Example 3.10. We define the random variable X that quite simply tells us the value the arrow points to. The distribution function of this random variable, shown in Figure 3.4, is **continuous** (without jumps). The 0.3 quantile of this variable's distribution is $F_X^{-1}(0.3) = 108$. $\diamondsuit$

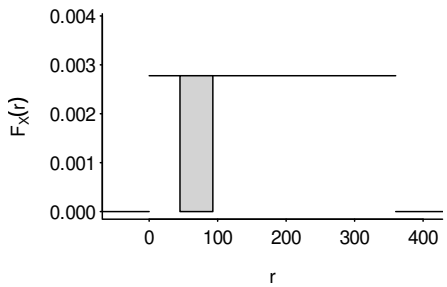


Figure 3.5: Probability density function of a variable following a continuous uniform distribution from 0 to 360. The probability of observing a value between 45 and 93 corresponds to the area underneath the density over this interval.

3.2.2 Probability density functions

The distribution of some random variables X can be represented by means of a **probability density function** f_X. Let us get the formal definition out of the way before discussing what the underlying idea is.

Definition 3.32 (Probability density function)**.** Let X be a random variable and F_X its distribution function. If there exists a non-negative function f_X such that

$$F_X(b) - F_X(a) = \int_a^b f_X(t)\,\mathrm{d}t$$

for all real numbers a, b, then f_X is a probability density function of X. $\diamondsuit$

Figure 3.5 shows what the idea is. We want to represent the distribution of X in such a way that the probability that X falls in a certain interval corresponds to the area underneath the probability density function over this interval.

For a random variable X with a continuous uniform distribution on $[0, 360)$, one possible probability density function is

$$f_X(x) = \begin{cases} \frac{1}{360}, & \text{if } x \in [0, 360), \\ 0, & \text{else.} \end{cases}$$

As you may recall from school, the area underneath a curve over an interval is given by the integral of the function over this interval. Indeed, we have

$$\mathbb{P}(X \in [45, 93]) = \int_{45}^{93} f_X(t)\,\mathrm{d}t = \int_{45}^{93} \frac{1}{360}\,\mathrm{d}t = \frac{t}{360}\bigg|_{45}^{93} = \frac{93-45}{360}.$$

Remark 3.33. Not all random variables have a probability density function. If a random variable has a probability density function, then its distribution function is continuous. That said, it is possible for a random variable to have a continuous distribution function

but no probability density function. Such random variables are difficult to construct, though. ♢

***Remark 3.34.** If a random variable has *a* probability density function, then it has an infinite number of them: Given a suitable probability density function f_X, we may define another one by changing the value of f_X at one single arbitrary point. This doesn't affect the integral. That said, if the distribution function F_X is not only continuous but also differentiable, then the derivative of F_X is a probability density function of X and is considered the canonical one. ♢

Definition 3.35. If a random variable has a probability density function, we call it **(absolutely) continuous**. ♢

We'll have a look at some examples of continuous probability distributions in Section 3.4.

***Remark 3.36.** An absolutely continuous random variable cannot be discrete and vice versa: If X has a probability density function f_X, then

$$\mathbb{P}(X = r) = \int_r^r f_X(t)\,\mathrm{d}t = 0$$

for all real numbers r. If X is discrete, however, there must be some r such that $\mathbb{P}(X = r) > 0$.

However, random variables can be neither absolutely continuous nor discrete. One example would the amount of precipitation on a given day in a given area. It's possible that there's a 70% chance of no precipitation ($\mathbb{P}(X = 0) = 0.7$), but that, if there is some precipitation, then the amount of precipitation is absolutely continuous. ♢

3.2.3 The expected value and the variance

Before we have a closer look at some classic probability distributions, let's introduce the two most commonly used numeric properties of random variables (or of their probability distributions). The first property is the **expected value** (also called **expectation** or **mean**), which expresses which value the random variable takes on on average. The expected value generalises the arithmetic mean, which can straightforwardly be computed if you have a finite number of values, to sample spaces of arbitrary size.

If X is a discrete random variable on a sample space Ω, then its expected value $\mathbb{E}(X)$ can be computed as

$$\mathbb{E}(X) = \sum_{x \in \Omega} x\mathbb{P}(X = x),$$

if this sum is a well-defined real number.[3] In the *Obenabe* example (Example 3.28), we hence have

$$\mathbb{E}(X) = 0 \cdot \frac{1}{3} + 2 \cdot \frac{1}{9} + 3 \cdot \frac{1}{9} + 4 \cdot \frac{1}{9} + 8 \cdot \frac{1}{9} + 10 \cdot \frac{1}{9} + 11 \cdot \frac{1}{9} = 4.22.$$

[3] It is possible for the sum to depend on the order in which we enumerate Ω, in which case it isn't well-defined. It's also possible for this sum to diverge to $-\infty$ or ∞, in which case it isn't a real number. We won't deal with such pathological distributions, though.

As a further example consider the random variable G generated by the random number generator from Example 3.8. Its expectation is

$$\mathbb{E}(G) = \sum_{k=1}^{\infty} k\mathbb{P}(G = k) = \sum_{k=1}^{\infty} \frac{k}{2^k} = 2.$$

For this course, you don't have to be able to compute expected values for distributions with infinite sample spaces. The idea here is merely to show that the arithmetic mean can be generalised to cases with an infinite number of values.

Incidentally, and more generally, it holds that

$$\mathbb{E}(g(X)) = \sum_{x \in X(\Omega)} g(x)\mathbb{P}(X = x)$$

for each function g, if this sum is a well-defined real number.

If the random variable X has a probability density function f_X, then its expectation can be computed as

$$\mathbb{E}(X) = \int_{-\infty}^{\infty} x f_X(x)\,\mathrm{d}x,$$

if this integral is a well-defined real number. In the fortune wheel example, for instance,

$$\begin{aligned}
\mathbb{E}(W) &= \int_{-\infty}^{\infty} x f_X(x)\,\mathrm{d}x \\
&= \int_0^{360} \frac{x}{360}\,\mathrm{d}x \\
&= \frac{1}{360}\left[\frac{1}{2}x^2\right]\Big|_0^{360} \\
&= \frac{360^2}{2 \cdot 360} \\
&= 180.
\end{aligned}$$

More generally,

$$\mathbb{E}(g(X)) = \int_{-\infty}^{\infty} g(x) f_X(x)\,\mathrm{d}x,$$

for (essentially) each function g, if this integral is a well-defined real number.[4]

If a probability distribution is neither absolutely continuous nor discrete, one can compute the expectations of its continuous and its discrete parts separately and then combine them using a suitable weighting. We'll only discuss discrete and absolutely continuous distributions, though.

Lemma 3.37 (Properties of the expected value)**.** Let X, Y be random variables. If X is constant, that is, $X \equiv c$, then $\mathbb{E}(X) = c$.

[4]Regarding 'essentially': It is possible to construct so-called non-measurable functions g that cannot be integrated.

The expected value is **linear**. This means that $\mathbb{E}(aX + bY) = a\mathbb{E}(X) + b\mathbb{E}(Y)$ for constants a, b, if $\mathbb{E}(X)$ and $\mathbb{E}(Y)$ exist in the first place. This property follows from the linearity of finite and infinite sums as well as from the linearity of integrals. $\diamondsuit$

The expected value expresses which value the random variable takes on on average. But it would be useful to also have some numerical measure that expresses how large the discrepancies between individual instantiations of this random variable and its expected value are expected to be. At first blush, it would seem to make sense to compute the expected value of these discrepancies, that is,

$$\mathbb{E}(X - \mathbb{E}(X)).$$

By linearity of the expected value, however,

$$\mathbb{E}(X - \mathbb{E}(X)) = \mathbb{E}(X) - \mathbb{E}(\mathbb{E}(X)) = \mathbb{E}(X) - \mathbb{E}(X) = 0$$

for each random variable X that has an expected value. So this measure doesn't carry any information. A more useful measure is the expected value of the absolute discrepancies, that is,

$$\mathbb{E}(|X - \mathbb{E}(X)|).$$

While this measure is sometimes used, it's cumbersome to use. Instead, one usually works with the expected value of the squared discrepancies, that is,

$$\mathbb{E}((X - \mathbb{E}(X))^2).$$

If X is a random variable such that this measure exists, we call $\mathrm{Var}(X) := \mathbb{E}((X - \mathbb{E}(X))^2)$ the **variance** of X.[5]

Lemma 3.38. If X is a random variable such that $\mathrm{Var}(X)$ exists, then

$$\mathrm{Var}(X) = \mathbb{E}(X^2) - \mathbb{E}(X)^2. \qquad \diamondsuit$$

Proof. We expand the expression in the definition and then apply properties of the expected value:

$$\begin{aligned}
\mathrm{Var}(X) &= \mathbb{E}((X - \mathbb{E}(X))^2) && \text{[Definition]} \\
&= \mathbb{E}(X^2 - 2X\mathbb{E}(X) + \mathbb{E}(X)^2) && [(a-b)^2 = a^2 - 2ab + b^2] \\
&= \mathbb{E}(X^2) - 2\mathbb{E}(X(\mathbb{E}(X))) + \mathbb{E}(X)^2 && \text{[Linearity of } \mathbb{E}] \\
&= \mathbb{E}(X^2) - 2\mathbb{E}(X)^2 + \mathbb{E}(X)^2 && [\mathbb{E}(X) \text{ is constant}] \\
&= \mathbb{E}(X^2) - \mathbb{E}(X)^2.
\end{aligned}$$

$\square$

[5] Not all random variables have a variance. This includes all random variable that do not have an expected value.

If a random variable is expressed in some unit (e.g., seconds), then its variance is expressed in the square of this unit (e.g., squared seconds). By taking the root of the variance, we get rid of these squared units. The numerical property so obtained is called the random variable's **standard deviation**: $\sqrt{\mathrm{Var}(X)} =: \mathrm{Std}(X)$.

Example 3.39. Lemma 3.38 allows us to compute variances quite straightforwardly. In the *Obenabe* example, we have

$$\mathbb{E}(X^2) = 0^2 \cdot \frac{1}{3} + 2^2 \cdot \frac{1}{9} + 3^2 \cdot \frac{1}{9} + 4^2 \cdot \frac{1}{9} + 8^2 \cdot \frac{1}{9} + 10^2 \cdot \frac{1}{9} + 11^2 \cdot \frac{1}{9} = 34.89.$$

Hence, the variance of the random variable in the *Obenabe* example is

$$\mathbb{E}(X^2) - \mathbb{E}(X)^2 = 34.89 - 4.22^2 = 17.08.$$

So its standard deviation is $\sqrt{17.08} \approx 4.13$.

For the random number generator in Example 3.8, we have

$$\mathbb{E}(G^2) = \sum_{k=1}^{\infty} \frac{k^2}{2^k} = 6.$$

(This computation is not obvious; for the purposes of this course, we may consider the result as given.) Hence, the variance of G is

$$\mathrm{Var}(G) = \mathbb{E}(G^2) - \mathbb{E}(G)^2 = 6 - 2^2 = 2.$$

So its standard deviation is $\sqrt{2}$.

In the fortune wheel example, we have

$$\mathbb{E}(W^2) = \int_0^{360} \frac{t^2}{360}\,\mathrm{d}t = \frac{360^3}{3 \cdot 360} = 43200.$$

So the variance of W is

$$\mathbb{E}(W^2) - \mathbb{E}(W)^2 = 43200 - 180^2 = 10800.$$

So its standard deviation is $\sqrt{10800} \approx 103.9$. ♢

Lemma 3.40 (Properties of the variance)**.** Let X be a random variable such that $\mathrm{Var}(X)$ exists and let a, b be constants. Then

$$\mathrm{Var}(aX + b) = a^2\mathrm{Var}(X).$$ ♢

Proof. We have

$$\begin{aligned}
\mathrm{Var}(aX + b) &= \mathbb{E}\left((aX + b - \mathbb{E}\,(aX + b))^2\right) \\
&= \mathbb{E}\left((aX + b - a\mathbb{E}\,(X) - b)^2\right) \\
&= \mathbb{E}\left(a^2\,(X - \mathbb{E}(X))^2\right) \\
&= a^2\mathbb{E}\left((X - \mathbb{E}(X))^2\right) \\
&= a^2\mathrm{Var}(X).
\end{aligned}$$ □

In general, though, it is *not* the case that $\mathrm{Var}(X+Y) = \mathrm{Var}(X) + \mathrm{Var}(Y)$. If X, Y are independent (with independence of random variables being defined in the next section), however, it *is* true that $\mathrm{Var}(X+Y) = \mathrm{Var}(X) + \mathrm{Var}(Y)$.

Example 3.41. Based on the random variable G from Example 3.8, we define the random variable $\widetilde{G} := 3G + 4$. By the properties of the expected value and the variance, we obtain

$$\mathbb{E}(\widetilde{G}) = \mathbb{E}(3G+4) = 3\mathbb{E}(G) + 4 = 10$$

and

$$\mathrm{Var}(\widetilde{G}) = \mathrm{Var}(3G+4) = 3^2\mathrm{Var}(G) = 18. \qquad \diamondsuit$$

Example 3.42 (Variance of a sum $\neq$ sum of the variances)**.** Based on the random variable G from Example 3.8, we define the random variable $\overline{G} := -G$. Then $G + \overline{G} = G - G = 0$. So

$$\mathrm{Var}(G + \overline{G}) = 0 \neq 4 = \mathrm{Var}(G) + \mathrm{Var}(\overline{G}).$$

It's intuitively clear that $G, \overline{G}$ are not independent. This intuition is formalised in the next section. $\diamondsuit$

3.2.4 Independence

Definition 3.43 (Independence of random variables)**.** Let X, Y be random variables. We say that X, Y are **independent** if, for any set of numbers E_1, E_2 about which it's possible to make probabilistic claims (cf. the comment on page 69), it is the case that

$$\mathbb{P}(X \in E_1, Y \in E_2) = \mathbb{P}(X \in E_1)\mathbb{P}(Y \in E_2).$$

This definition for the independence of two random variables can be generalised to the definition of multiple and of an infinite number of random variables in the same way that the definition of independence of two events was generalised. $\diamondsuit$

Conceptually, independence of X, Y means that knowing the value that X takes on doesn't provide any information as to the value of Y, and vice versa. If random variables are independent, the sum of their variances can easily be computed, as per the next lemma, which is presented without proof.

Lemma 3.44. Let X, Y be independent random variables with existing variance. Then

$$\mathrm{Var}(X+Y) = \mathrm{Var}(X) + \mathrm{Var}(Y). \qquad \diamondsuit$$

3.3 Examples of discrete probability distributions

3.3.1 The discrete uniform distribution

Let n be a natural number and $\Omega := \{1, \ldots, n\}$. Let X be the random variable defined by $X(k) = k$ for each $k \in \Omega$. If, for each $k \in \Omega$, $\mathbb{P}(X = k) = 1/n$, then X has a **discrete uniform distribution** on Ω. We write $X \sim \mathrm{Unif}(\Omega)$.

The classic example for such a discrete uniform distribution is a throw with a fair six-sided dice. In this example, we have $\mathbb{P}(X = k) = 1/6$ for $k \in \{1,2,3,4,5,6\}$ and $\mathbb{P}(X = k) = 0$ for all $k \notin \{1,\dots,6\}$.

The expected value of a random variable X with a discrete uniform distribution over the sample space $\{1,2,\dots,n\}$ is

$$\mathbb{E}(X) = \frac{1}{n}(1+2+\cdots+n) = \frac{n(1+n)}{2n} = \frac{1+n}{2},$$

which should make sense intuitively. Its variance is

$$\mathrm{Var}(X) = \frac{n^2-1}{12},$$

which we won't derive ourselves.

More generally, let X be a random variable with a discrete uniform distribution over

$$\Omega := \{a, a+1, a+2, \dots, b-2, b-1, b\}.$$

Then

$$\mathbb{E}(X) = \frac{a+b}{2}, \mathrm{Var}(X) = \frac{(b-a+1)^2-1}{12}.$$

***Exercise 3.45.** Let X be a random variable with a discrete uniform distribution over

$$\{1,2,\dots,n-2,n-1\}.$$

Define $Y := X/2 + 1/2$. Then Y has a discrete uniform distribution over

$$\{1, 1.5, 2, \dots, n/2\}.$$

Use the properties of the expected value and the variance to determine $\mathbb{E}(Y), \mathrm{Var}(Y)$. ♢

Remark 3.46 (Generating data from a discrete uniform distribution)**.** In the upcoming sections and chapters, we'll try to get a handle on concepts by means of simulations. These will require us to generate data from certain distributions. The following snippet shows how we can generate `n_obs` independent data points from a discrete uniform distribution on $\{1,\dots,n\}$.

```
# Define sample space
n <- 6
Omega <- 1:n
# Generate n_obs observations discrete unif. dist. on sample space
n_obs <- 20
observations <- sample(Omega, n_obs, replace = TRUE)
observations
```

```
 [1] 4 2 2 1 3 2 2 1 3 3 1 6 4 4 2 5 3 3 1 3
```

♢

3.3.2 The Bernoulli distribution

The **Bernoulli distribution** models the outcome X of a random experiment with two possible outcomes, which are labelled 0 ('failure') and 1 ('success'). It has a parameter $p \in [0,1]$, where $\mathbb{P}(X = 1) = p$ and $\mathbb{P}(X = 0) = 1 - p$. If a random variable X follows a Bernoulli distribution with parameter p, we write $X \sim \text{Bernoulli}(p)$.

The expected value of a Bernoulli(p)-distributed random variable X is easily computed:

$$\mathbb{E}(X) = 0 \cdot (1 - p) + 1 \cdot p = p.$$

Similarly,

$$\mathbb{E}(X^2) = 0^2 \cdot (1 - p) + 1^2 \cdot p = p.$$

Hence,

$$\text{Var}(X) = p - p^2 = p(1 - p).$$

Example 3.47. We draw a random card from a deck of 36 *Jass* cards. The probability that we've drawn an Ace is $p = 4/36 = 1/9$. We can thus model the random experiment 'Drawing an Ace' as a Bernoulli experiment with parameter $p = 1/9$, where we label the outcome of the experiment as 1 if we draw an Ace and 0 otherwise. $\diamondsuit$

3.3.3 The binomial distribution

Let $X_1, \ldots, X_n$ be independent random variables that all follow a Bernoulli(p) distribution. Then $X := X_1 + \cdots + X_n$ is said to follow a **binomial distribution** with parameters n and p. We write $X \sim \text{Binomial}(n, p)$. In words, the binomial distribution models the number of n independent and identical Bernoulli experiments that have successful outcomes.

By the linearity of the expected value,

$$\mathbb{E}(X) = \mathbb{E}(X_1) + \cdots + \mathbb{E}(X_n) = np.$$

By the independence of $X_1, \ldots, X_n$,

$$\text{Var}(X) = \text{Var}(X_1) + \cdots + \text{Var}(X_n) = np(1 - p).$$

The probability that precisely the first k of n independent and identical Bernoulli experiments produce a success is

$$\mathbb{P}(X_1 = 1)\mathbb{P}(X_2 = 1) \cdots \mathbb{P}(X_k = 1)\mathbb{P}(X_{k+1} = 0) \cdots \mathbb{P}(X_n = 0) = p^k(1 - p)^{n-k}.$$

There are $n \cdot (n - 1) \cdot \cdots \cdot 1 = n!$ possibilities to reorder these Bernoulli experiments. Since neither the outcomes of the first k of these experiments nor the outcomes of the last $n - k$ ones can be distinguished from one another, there are

$$\frac{n!}{k!(n - k)!} =: \binom{n}{k}$$

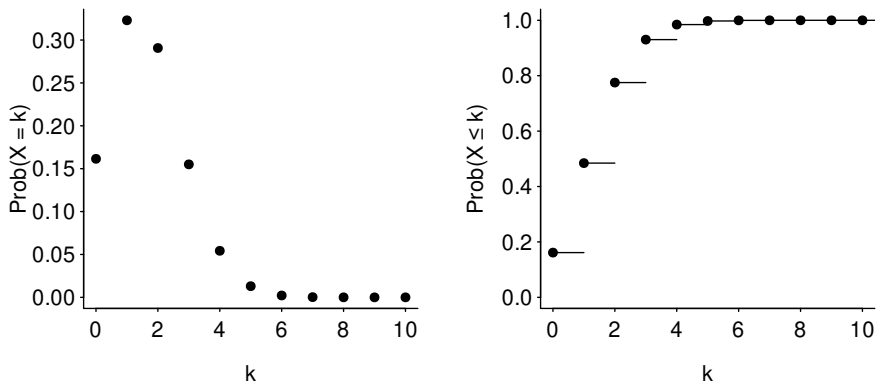


Figure 3.6: Left: Probability of the number of successes in a Binomial(10, 1/6) distribution. Right: Distribution function of the same distribution.

different possible orders in which these outcomes can be observed. Hence, the probability that exactly k of n independent and identical Bernoulli experiments produce a success is

$$\mathbb{P}(X = k) = \binom{n}{k} p^k (1-p)^{n-k}.$$

The number $\binom{n}{k}$ is called the **binomial coefficient** of n and k. We also say 'n choose k'. For $k < 0$ or $k > n$, we define $\binom{n}{k} = 0$. Consequently, the distribution function of a Binomial(n, p) distribution is

$$F(r) = \sum_{k=0}^{\lfloor r \rfloor} \binom{n}{k} p^k (1-p)^{n-k}.$$

The binomial coefficient $\binom{n}{k}$ can be calculated using the R command `choose(n, k)`. R also features some functions that can be used to retrieve information about the binomial distribution and that can generate data from a binomial distribution. As an example, imagine that we throw a 6-sided dice ten times and count how often we've thrown a 6. This outcome can be modelled by means of a Binomial(10, 1/6) distribution. The function `dbinom()` allows us compute $\mathbb{P}(X = k)$. For instance, the probability that we throw exactly five sixes is about 1.3%:

```
dbinom(5, 10, 1/6)

[1] 0.01302381
```

Figure 3.6 shows $\mathbb{P}(X = k)$ for $X \sim \text{Binomial}(10, 1/6)$ and $k = 0, \ldots, 10$.

The function `pbinom()` can be used to retrieve $\mathbb{P}(X \leq k)$. For instance, the probability that we throw no more than two sixes is about 78%:

```
pbinom(2, 10, 1/6)

[1] 0.7752268
```

The probability that we throw more than two sixes is hence about 22%. Similarly, the probability that we throw at least two sixes is about 52%:

```
1 - pbinom(1, 10, 1/6)

[1] 0.5154833

pbinom(1, 10, 1/6, lower.tail = FALSE)

[1] 0.5154833
```

The `qbinom()` function retrieves the quantiles of the Binomial(n, p) distribution:

```
qbinom(0.70, 10, 1/6)

[1] 2

qbinom(0.75, 10, 1/6)

[1] 2

qbinom(0.80, 10, 1/6)

[1] 3
```

In other words, if lots of people were to throw a dice ten times each, at least 70% of them would throw at most two sixes. In fact, at least 75% of them would throw at most two sixes. At least 80% of them would throw at most three sixes. In fact, about 77.5% of them would throw at most two sixes, as we can compute using `pbinom()`:

```
k <- 0:10
tibble(k = k,
       "Prob(X <= k)" = pbinom(k, 10, 1/6))

# A tibble: 11 x 2
      k `Prob(X <= k)`
  <int>          <dbl>
1     0          0.162
2     1          0.485
3     2          0.775
4     3          0.930
5     4          0.985
6     5          0.998
7     6          1.000
# i 4 more rows
```

The function `rbinom()` can be used to generate independent observations from a binomial distribution. Let's say that we wanted to simulate the number of sixes produced by each of twenty people throwing the dice 10 times:

```
rbinom(20, 10, 1/6)

 [1] 2 2 1 1 1 0 5 2 1 0 2 3 2 3 2 3 1 3 0 1
```

The Bernoulli(p) distribution is the same as the Binomial(1, p) distribution.

3.4 Examples of continuous probability distributions

3.4.1 The continuous uniform distribution

We've already encountered the continuous uniform distribution in the fortune wheel example. More generally, we denote as $\text{Unif}([a, b])$ the continuous uniform distribution on the interval $[a, b], a < b$. The endpoints can be included or left out at will; the distribution doesn't change as a a result. For $X \sim \text{Unif}([a, b])$, we have

$$\mathbb{E}(X) = \frac{a+b}{2}$$

and

$$\text{Var}(X) = \frac{(b-a)^2}{12}.$$

A density function for a continuous uniform distribution on $[a, b]$ is

$$f_U(x) = \begin{cases} \frac{1}{b-a}, & \text{if } x \in [a, b], \\ 0, & \text{otherwise.} \end{cases}$$

The distribution function is

$$F_U(r) = \begin{cases} 0, & \text{if } r < a, \\ \frac{r-a}{b-a}, & \text{if } r \in [a, b], \\ 1, & \text{if } r > b. \end{cases}$$

We can evaluate a valid density function at a given point using `dunif()`. For instance, for a continuous uniform distribution on $[-\pi, \pi]$, we find

```
dunif(2, -pi, pi) # = 1/(2*pi)

[1] 0.1591549

dunif(4, -pi, pi)

[1] 0
```

The distribution function is implemented in `punif()`:

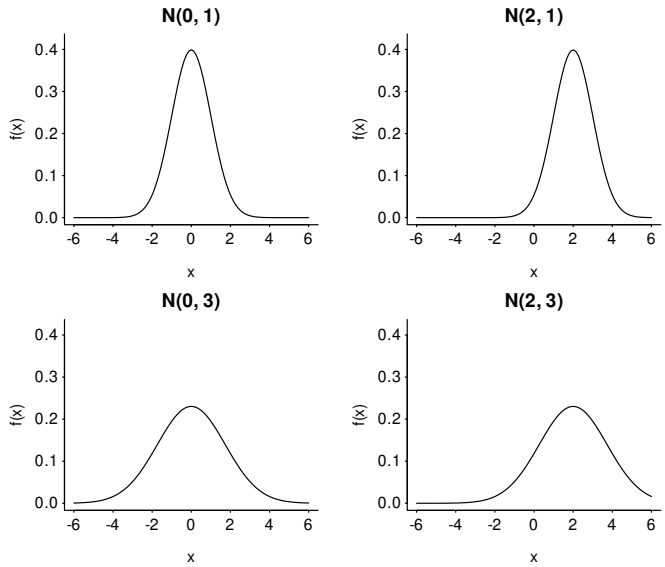


Figure 3.7: Density functions of four normal distributions.

```
punif(2, -pi, pi)

[1] 0.8183099
```

That is, if $X \sim \text{Unif}([-\pi, \pi])$, then $\mathbb{P}(X \leq 2) \approx 82\%$. Using `qunif()`, we can obtain the quantiles of this distibution; using `runif()`, we can generate independent observations from it:

```
runif(10, -pi, pi)

 [1] -0.9495818 -0.8845674 -1.9287873 -2.7450394  0.1155496
 [6]  2.3338782  2.8062412 -1.0700206  2.5539018  0.3239638
```

3.4.2 The normal distribution

The **normal distribution** occupies a central space in the theory and practice of probability theory, statistics, and data analysis. The normal distribution has two parameters: its expected value μ, and its variance σ^2. If a random variable X follows a normal distribution with parameters μ, σ^2, we can write $X \sim \text{Normal}(\mu, \sigma^2)$ or $X \sim \mathcal{N}(\mu, \sigma^2)$. Figure 3.7 shows density functions for normal distributions with four different parameter combinations. The plot in the top left shows the **standard normal distribution**, that is, the normal distribution with mean 0 and variance 1.

The canonical density function f of a $\mathcal{N}(\mu, \sigma^2)$ distribution is

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{\frac{-(x-\mu)^2}{2\sigma}\right\},$$

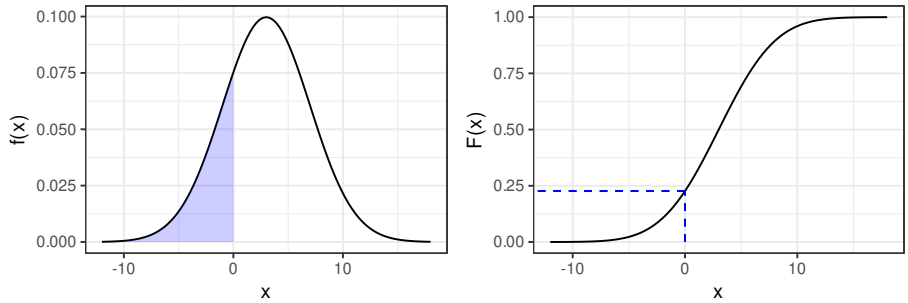


Figure 3.8: The $\mathcal{N}(3, 16)$ distribution. Left: The probability of observing a value lower than 0 corresponds to the area under the curve up till 0. Right: We can compute this probability using the distribution function.

where $\exp\{\cdot\}$ is the exponential function. This formula isn't too important for us. It suffices to appreciate that μ determines the central tendency of the normal distribution, whereas σ^2 determines how tall and how wide it is. The distribution function of a normal distribution cannot be represented analytically.

The R functions `dnorm()`, `pnorm()`, and `rnorm()` can be used to evalute the density and distribution functions, and to generate independent observations from normal distributions. Importantly, R parametrises normal distributions using their standard deviation rather than using their variance. For instance, consider $X \sim \mathcal{N}(3, 16)$. Then we can compute $\mathbb{P}(X \leq 0)$ as follows; also see Figure 3.8:

```
pnorm(0, mean = 3, sd = sqrt(16))

[1] 0.2266274
```

Exercise 3.48 (IQ)**.** The distribution of IQ scores in a population can be modelled as a $\mathcal{N}(100, 15^2)$ distribution. Use the `pnorm()` und `qnorm()` functions in order to answer the questions below. For the last three questions, you may want to also use the binomial distribution, but you don't have to.

1. What's the probability that a randomly picked person has an IQ lower than 90?
2. What's the probability that a randomly picked person has an IQ higher than 85?
3. What's the probability that a randomly picked person has an IQ between 110 and 120?
4. What's the probability that a randomly picked person has an IQ that lies at least a standard deviation away from the mean?
5. For which IQ score is it the case that 25% of the population has a lower IQ score and 75% of the population has a higher IQ score?
6. A person is 'of average intelligence' if their IQ score belongs to the middle 45% of the population. Between which two IQ scores does average intelligence fall?

7. You randomly pick two people and obtain their IQ scores. What's the probability that none of them has an IQ higher than 105?
8. You randomly pick three people and obtain their IQ scores. What's the probability that exactly one of them has an IQ below 90?
9. You randomly pick three people and obtain their IQ scores. What's the probability that at least one of them has an IQ below 90? ♢

***Remark 3.49.** If X has a continuous distribution function, then the p-th quantile $q := F_X^{-1}(p)$ satisfies $F_X(q) = p, p \in (0,1)$. If X does not have a continuous distribution function, then it is possible that $F_X(q) > p$. ♢

3.5 The sampling distribution of the sample mean

Let $X_1, \ldots, X_n$ be independent and identically distributed random variables with an existing expected value and variance. We define the arithmetic mean of these random variables as

$$\overline{X} := \frac{X_1 + \cdots + X_n}{n}.$$

Can we make any sensible claims about $\overline{X}$?

The answer is 'yes'. First, due to the linearity of the expected value, we have

$$\begin{aligned}
\mathbb{E}(\overline{X}) &= \mathbb{E}\left(\frac{X_1 + \cdots + X_n}{n}\right) \\
&= \frac{1}{n}\mathbb{E}(X_1 + \cdots + X_n) \\
&= \frac{1}{n}(\mathbb{E}(X_1) + \cdots + \mathbb{E}(X_n)) \\
&= \frac{n}{n}\mathbb{E}(X_1) \\
&= \mathbb{E}(X_1).
\end{aligned}$$

In words, the mean of n identical random variables has the same expectation as the individual random variables. Independence is not required for this claim.

Second, thanks to the independence of the random variables, we have

$$\begin{aligned}
\mathrm{Var}(\overline{X}) &= \mathrm{Var}\left(\frac{X_1 + \cdots + X_n}{n}\right) \\
&= \frac{1}{n^2}\mathrm{Var}(X_1 + \cdots + X_n) \\
&= \frac{1}{n^2}(\mathrm{Var}(X_1) + \cdots + \mathrm{Var}(X_n)) \\
&= \frac{n}{n^2}\mathrm{Var}(X_1) \\
&= \frac{\mathrm{Var}(X_1)}{n}.
\end{aligned}$$

Hence, the standard deviation of the mean is

$$\mathrm{Std}(\overline{X}) = \sqrt{\mathrm{Var}(\overline{X})} = \frac{\mathrm{Std}(X_1)}{\sqrt{n}}.$$

Hence, the larger the sample size n, the less individual sample means deviate from their expected value.

Finally, the celebrated **Central Limit Theorem** (CLT) allows us to say something about the shape of the distribution of the sample mean.

Theorem 3.50 (Central Limit Theorem)**.** Let $X_1, \ldots, X_n$ be independent observations drawn at random from a distribution with expectation μ and finite variance $\sigma^2 < \infty$. Then the distribution of the statistic

$$\frac{\overline{X} - \mu}{\sigma/\sqrt{n}}, \tag{3.1}$$

where $\overline{X} := (X_1 + \cdots + X_n)/n$, converges to a standard normal distribution (i.e., to $\mathcal{N}(0,1)$). ♢

We won't prove this theorem, but the examples will illustrate what this theorem says and doesn't say. Incidentally, the theorem is about a limit (that is, about an asymptotic finding) and it is central (that is, of key importance) to probability theory, hence the name. It's not a theorem about 'central limits', which don't exist.

Remark 3.51. Let a, b be real numbers. If $Z \sim \mathcal{N}(0,1)$, then $a + bZ \sim \mathcal{N}(a, b^2)$. Hence, if the statistic in (3.1) is approximately $\mathcal{N}(0,1)$ for a given sample size n, then

$$\overline{X} = \mu + \frac{\sigma}{\sqrt{n}}\left(\frac{\overline{X} - \mu}{\sigma/\sqrt{n}}\right) \sim \mathcal{N}(\mu, \sigma^2/n)$$

approximately, allowing us to use the normal distribution as an approximation to the distribution of the sample mean. ♢

Example 3.52. Consider the random number generator from Example 3.8. Let us generate 5 independent observations $G_1, \ldots, G_5$ using this random number generator. In the previous sections, we found out that $\mathbb{E}(G_1) = \mathrm{Var}(G_1) = 2$. Hence, the mean $\overline{G} = (G_1 + \cdots + G_5)/5$ also has expectation $\mathbb{E}(\overline{G}) = \mathbb{E}(G_1) = 2$ and standard deviation

$$\mathrm{Std}(\overline{G}) = \sqrt{\frac{2}{5}} \approx 0.632.$$

The CLT suggests that we could approximate the distribution of $\overline{G}$ as $\mathcal{N}(2, 2/5)$.

Let's see how good this approximation actually is. The file `rng_2k.R` in the `functions` subdirectory contains a function (`rrng_2k()`) for actually generating random numbers from this distribution.[6] Using `source()`, we read it in. Then, we simulate the scenario above 50,000 times, ending up with 50,000 sample means of five observations each. This is easily done using `replicate()`.

[6]This distribution is a special case of the **geometric distribution**, which wasn't discussed in this script. The functions in `rng_2k.R` are simple wrappers around the functions documented on `?Geometric`.

```
source(here("functions", "rng_2k.R"))
M <- 50000
n <- 5
means <- replicate(M, {
  rrng_2k(n) |> mean()
})
```

The mean of these 50,000 sample means corresponds closely to the value of $\mathbb{E}(\overline{G})$ we computed. Similarly, the standard deviation of the 50,000 sample means corresponds closely to the theoretical value of $\text{Std}(\overline{G})$. Indeed, any difference between these numbers and their theoretical value is due solely to our having simulated 'only' 50,000 samples.

```
mean(means)

[1] 2.001996

sqrt(mean((means - mean(means))^2)) # standard deviation of sample means

[1] 0.6330472
```

Let's take a look at the shape of the distribution of the sample means. The left panel in Figure 3.9 shows, in black, the **empirical cumulative distribution function** of these sample means. In essence, this is the distribution function you'd obtain if you treated the 50,000 observations of $\overline{G}$ as their own finite distribution. The blue line shows the distribution function of the $\mathcal{N}(2, 2/5)$ distribution.

The right panel in the same figure shows a **histogram** of the same 50,000 observations of $\overline{G}$, which provides an estimate of the density. The blue line shows the density function of the $\mathcal{N}(2, 2/5)$ distribution.

```
par(mfrow = c(1, 2)) # plots side by side
plot(ecdf(means), xlab = "Sample mean", ylab = "Cumulative frequency",
     main = paste0("n = ", n))
curve(pnorm(x, mean = 2, sd = sqrt(2/n)),
      add = TRUE, col = "steelblue1", lwd = 2)

hist(means, xlab = "Sample mean", ylab = "Density",
     xlim = c(-0.5, max(means)), freq = FALSE,
     col = "grey", breaks = 30, main = paste0("n = ", n))
curve(dnorm(x, mean = 2, sd = sqrt(2/n)),
      add = TRUE, col = "steelblue", lwd = 2)
par(mfrow = c(1, 1)) # normal plotting from here on
```

For $n = 5$, the CLT-based normal approximation is pretty poor. Indeed, if we compute $\mathbb{P}(\overline{G} < 1)$ under the assumption that $\overline{G} \sim \mathcal{N}(2, 2/5)$, we obtain an estimate of about 5.7%.

```
pnorm(1, mean = 2, sd = sqrt(2/n))
```

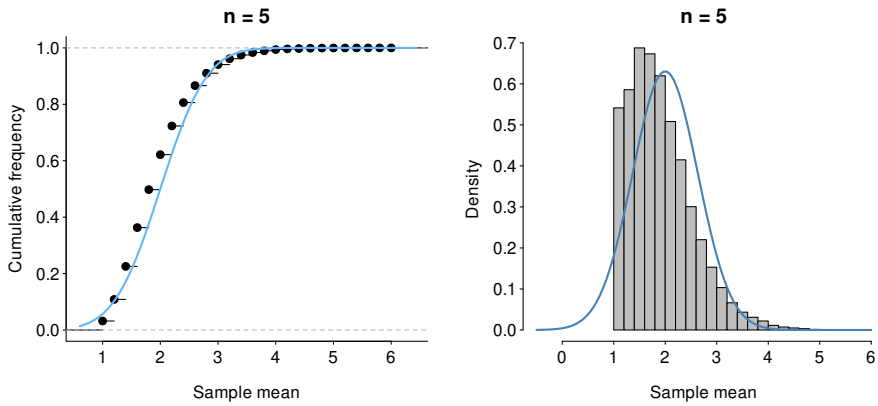


Figure 3.9: The distribution of $\overline{G}$ in 50,000 simulations for $n = 5$. The blue curves show the normal approximation.

```
[1] 0.05692315
```

But since the random number generator only generates values of 1 or larger, the actual probability is $\mathbb{P}(\overline{G} < 1) = 0$. Overestimates also occur, e.g., for $\mathbb{P}(2 < \overline{G} < 2.5)$.

```
pnorm(2.5, mean = 2, sd = sqrt(2/n)) - pnorm(2, mean = 2, sd = sqrt(2/n))
```

```
[1] 0.2854023
```

A better estimate is obtained by querying the simulated means:

```
mean(means < 2.5 & means > 2)
```

```
[1] 0.18462
```

What the CLT tells us, however, is that the normal approximation will become arbitrarily good if we increase n. To see this, increase the sample size to $n = 10, 30, 150$, redraw the plots, and run the computations above again. As you'll observe, the approximation becomes ever better, without ever being perfect. ♢

Example 3.53. We can do the same computations and run a similar simulation for the fortune wheel example. Let us generate two independent observations from the fortune wheel, W_1, W_2, and compute their mean $\overline{W} = (W_1 + W_2)/2$. Since $\mathbb{E}(W_1) = 180$, we have $\mathbb{E}(\overline{W}) = 180$ as well. Further, since $\mathrm{Var}(W_1) = 10800$,

$$\mathrm{Std}(\overline{W}) = \sqrt{\frac{10800}{2}} \approx 73.48.$$

The simulation below ends up with similar values.

```
n <- 2
M <- 50000
means <- replicate(M, {
  runif(n, 0, 360) |> mean()
```

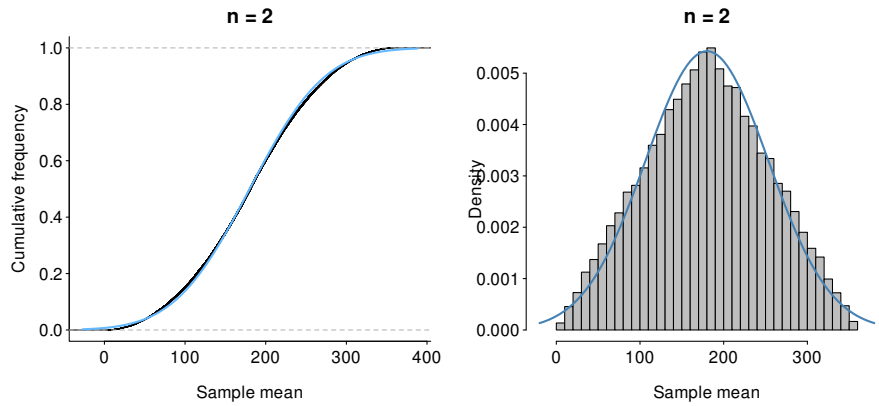


Figure 3.10: The distribution of $\overline{W}$ in 50,000 simulations for $n = 2$.

```
})
mean(means)

[1] 179.8841

sqrt(mean((means - mean(means))^2)) # standard deviation

[1] 73.06361
```

Figure 3.10 shows the empirical cumulative distribution function and a histogram of these 50,000 sample means, with the normal approximation in blue. Compared to the previous example, the normal approximation already looks much more useful even for $n = 2$.

```
par(mfrow = c(1, 2))
plot(ecdf(means), xlab = "Sample mean", ylab = "Cumulative frequency",
     main = paste0("n = ", n))
curve(pnorm(x, mean = 180, sd = sqrt(10800/n)),
      add = TRUE, col = "steelblue1", lwd = 2)

hist(means, xlab = "Sample mean", ylab = "Density", freq = FALSE,
     col = "grey", breaks = 30, main = paste0("n = ", n),
     xlim = c(-20, 380))
curve(dnorm(x, mean = 180, sd = sqrt(10800/n)),
      add = TRUE, col = "steelblue", lwd = 2)
par(mfrow = c(1, 1))
```

That said, it is still an imperfect approximation. To appreciate this, let's estimate $\mathbb{P}(0 < \overline{W} < 50)$ using the normal approximation:

```
pnorm(50, mean = 180, sd = sqrt(10800/n)) -
  pnorm(0, mean = 180, sd = sqrt(10800/n))
```

```
[1] 0.03128766
```

That is, about 3.1%. The simulation, however, shows that the actual probability is closer to 4%.[7]

```
mean(means < 50 & means > 0)

[1] 0.03818
```

Again increase the sample size—say, to $n = 3, 5, 10$. You'll observe that the normal approximation is excellent even for lower values of n. ♢

Comparing these two examples, we find that the CLT-based normal approximation is more useful more quickly for a continuous, bounded, symmetric distribution such as the continuous uniform distribution than it is for the discrete, unbounded, skewed distribution generated by our random number generator. It's important to appreciate that there is no magic sample size where the CLT 'kicks in'. The CLT offers an *approximation*. Whether this approximation is sufficiently good, depends on what you consider sufficiently good, on the sample size, and on the shape of the distribution from which the observations stem.[8]

***Activity 3.54.** Finally, let's turn to the *Obenabe* example. The random variable V that represents the point value of cards in the *Obenabe* mode of play takes on the values 10, 4 3, 2, 10, and 8, each with probability 1/9, and the value 0 with probability 1/3. The function `jass_one_run()` generates `n` independent observations from this distribution and computes their mean $\overline{V}$.

```
jass_one_run <- function(n) {
  values <- c(11,    4,   3,   2,  10,   8,   0)
  probs  <- c(1/9, 1/9, 1/9, 1/9, 1/9, 1/9, 1/3)
  sample(values, n, replace = TRUE, prob = probs) |> mean()
}
```

Again using `replicate()`, we can quickly generate 50,000 observations of $\overline{V}$, for instance for $n = 2$:

```
n <- 2
M <- 50000
means <- replicate(M, {
  jass_one_run(n = n)
})
```

The empirical cumulative distribution function and a histogram can be drawn like so.

```
par(mfrow = c(1, 2))
plot(ecdf(means), xlab = "Sample mean", ylab = "Cumulative frequency",
```

[7] The true actual probability in this particular example is $\int_0^{50} x/180^2 \, dx = 25/648 \approx 3.86\%$.

[8] The facts that $\mathbb{E}(\overline{X}) = \mathbb{E}(X_1)$ and $\mathrm{Var}(\overline{X}) = \mathrm{Var}(X_1)/n$ do *not* depend on n or on the shape of the distribution from which the sample is drawn, though.

```
     main = paste0("n = ", n))
curve(pnorm(x, mean = 4.22, sd = 4.13/sqrt(n)),
      add = TRUE, col = "steelblue1", lwd = 2)

hist(means, xlab = "Sample mean", ylab = "Density",
     freq = FALSE,
     col = "grey", breaks = 30, main = paste0("n = ", n),
     xlim = c(0, 11))
curve(dnorm(x, mean = 4.22, sd = 4.13/sqrt(n)),
      add = TRUE, col = "steelblue1", lwd = 2)
par(mfrow = c(1, 1))
```

Run the code snippets above for increasing values of n (e.g., $n = 2, 3, 5, 10, 25$) and observe the shape of the distribution of $\overline{V}$ compared to its normal approximation. $\diamondsuit$

If you know the distribution from which the samples are drawn, generating thousands of samples and computing their means is the more reliable method for making claims about the sampling distribution of the sample mean. The added value of the CLT is that we only need to know the expected value and the variance of the distribution from which the samples are drawn in order to make approximate statements about the sampling distribution of the sample mean.

Chapter 4

Descriptive statistics of a univariate sample

The aim of **descriptive statistics** is to describe and summarise characteristics of some pieces of information that were collected or observed. We'll call such a collection of pieces of information a **sample**, even though this term will make more sense in the next chapters. In the present chapter, we focus on the case where we've obtained n observations $(x_1, \ldots, x_n)$ of some numeric property; we want to show both ourselves and our audience what the key features of these observations are. From the next chapter onwards, our focus will lie on **inferential statistics**. In inferential statistics, we treat the n observations as a sample drawn from some distribution or generated according to some mechanism and try to use them to learn something about this distribution or this mechanism. Don't let the fact that only one chapter is devoted to descriptive statistics compared to several that are devoted to inferential statistics fool you into thinking that descriptive statistics is less important than inferential statistics, though: A solid description of the data will often be more insightful than the inferential analysis itself, and it will usually be easier to understand, too. In the chapters on inferential statistics, the necessity for a descriptive analysis will therefore often be stressed.

Throughout this chapter, we'll use the small dataset that I collected for my Bachelor's thesis. I gave 23 learners of Swedish as a foreign language four reading tasks: one in Swedish, one in Danish, and one in each of the two written standards of Norwegian (*bokmål*, *nynorsk*). Sadly, the different reading tasks can't directly be compared with each other (I was young!), and I couldn't retrieve the *nynorsk* data any more (it was a long time ago...). The dataset also contains some further pieces of information regarding the participants' language skills. Our goal is to communicate the key properties of the participants' scores on the *bokmål* test, which are stored in the column called `Norwegian` in the `jv_bachpap.csv` file. I assume that you've loaded the `tidyverse` and that the dataset is stored in the `data` subdirectory of your project directory.

```
d <- read_csv(here("data", "jv_bachpap.csv"))
d |>
  slice_head(n = 3)
```

```
# A tibble: 3 x 9
  LvlFrench LvlEnglish LvlGerman LvlSpanish NoLanguages
```

```
      <dbl>      <dbl>     <dbl>      <dbl>       <dbl>
1         5          5         5          2           5
2         5          4         3          0           3
3         5          5         0          0           2
# i 4 more variables: Swedish <dbl>, Danish <dbl>,
#   Norwegian <dbl>, Participant <chr>
```

Of course, we could just report the 23 values in the `Norwegian` column and call it a day. Note the use of the dollar sign to access a column in a data frame/tibble.

```
d$Norwegian
```

```
 [1] 13  9 11  7  6  9  3  4  6  8  6  5  3  2  6  4  5  5
[19] 10  4  3  2  5
```

But little insight can be gleaned from this. Some progress can be made by considering the empirical distribution that the sample gives rise to (as defined next) and applying the techniques for visualising and summarising a distribution that we've covered in the previous chapter to this empirical distribution.

Definition 4.1 (Empirical distribution)**.** Let $\boldsymbol{x} = (x_1, \ldots, x_n)$ be a real-valued vector. Its **empirical distribution** is the distribution of a random variable Y on the (finite) sample space

$$\Omega := \{x_1, \ldots, x_n\}$$

with

$$\mathbb{P}(Y = y) = \frac{1}{n}\#\{i : x_i = y\}$$

for all real numbers y. That is, the probability of observing $Y = y$ corresponds exactly to the proportion of x_i values for which $x_i = y$.

The corresponding **empirical (cumulative) distribution function** is defined by

$$\widehat{F}(r) := \frac{1}{n}\#\{i : x_i \leq r\}$$

for all real numbers r. ♢

For instance, the empirical distribution of the Norwegian data assigns a probability of about 13% to the event that the score equals 4; the probability of obtaining a score no more than 4 is about 35% according to this empirical distribution.

```
mean(d$Norwegian == 4)
```

```
[1] 0.1304348
```

```
mean(d$Norwegian <= 4)
```

```
[1] 0.3478261
```

The empirical distribution can be visualised by means of its distribution function. This can be drawn using the `ecdf()` command, which we've already encountered in the simulations in the previous chapter; see Figure 4.1. While distribution functions are popular

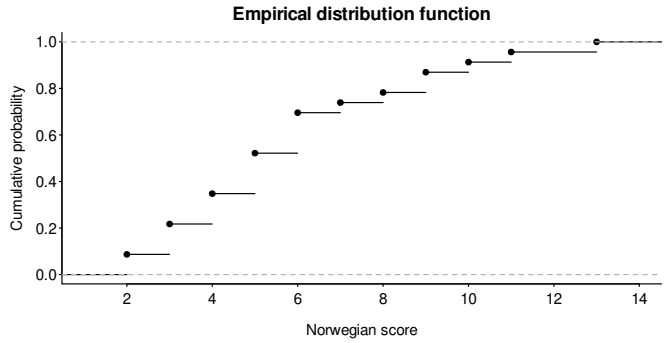


Figure 4.1: The empirical (cumulative) distribution function

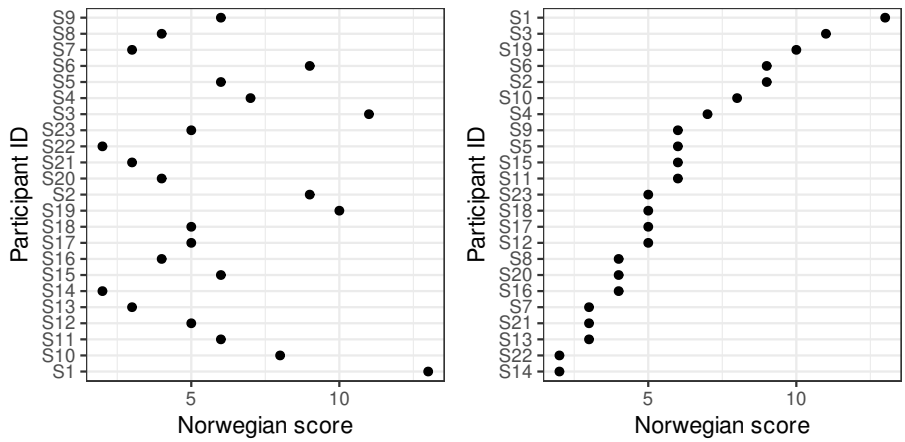


Figure 4.2: A (Cleveland) dot plot of the participants' Norwegian scores sorted by their IDs (left) and one sorted by their scores (right). There are no values lying suspiciously far from the bulk of the data.

among statisticians, researchers and laypeople don't seem to be overly enamoured with them, possibly partly because distribution functions can be difficult to compare between datasets.[1] So let's take a look at some visualisations that may be more useful.

4.1 Visualisations

4.1.1 Dotplots

A first such visualisation is the **(Cleveland) dot plot**; see Figure 4.2. The observed values are shown as dots on separate lines along the x-axis; each line has a label that's shown along the y-axis.

To draw the graph on the left, you can use the commands below. The comments following the # symbol explain how the graph is built up. I recommend that you use such

[1]Consider two numeric vectors $\boldsymbol{x} = (x_1, \ldots, x_m)$ and $\boldsymbol{y} = (y_1, \ldots, y_n)$. If the empirical distribution function curve of $\boldsymbol{x}$ lies *above* the one of $\boldsymbol{y}$, then the values in $\boldsymbol{x}$ tend to be *lower* than those in $\boldsymbol{y}$.

comments liberally at the start of your R career. This way, you'll still be able to understand your code months later. As you get more proficient in R, your comments can become more high-level.

```
ggplot(data = d,                # dataset containing the variables
       # aes() = aesthetics = how to plot which variable
       aes(x = Norwegian,       # variable along x-axis
           y = Participant)) +  # variable along y-axis
  geom_point() +                # show data as points
  xlab("Norwegian score") +     # esp. for papers/presentations:
  ylab("Participant ID")        #     label your axes
```

Make sure that you've loaded the `tidyverse` and `here` packages: Even if you've already installed them, you need to load them again using `library()` in each session in which you need them. Also mind capitalisation, brackets, commas, and the plus signs. The latter are used to add new layers to a graph produced using `ggplot()`. The first four lines of the code snippet above (`ggplot(...)`) merely draw the canvas on which the graph is constructed. The command after the first plus sign (`geom_point(...)`) draws the data as points on this canvas. The commands `xlab(...)` and `ylab(...)` add labels to the axes. Note that these different layers are strung together using plus signs rather than using the pipe (`|>`).

To draw the graph on the right, replace the `y = Participant` on the fourth line by `y = reorder(Participant, Norwegian)`. (Don't forget the bracket!)

When you run the commands above, you'll notice that the plots generated have a grey background. I don't particularly like this default setting, which is why I override it using the following line. You only need to run it once per session, and then all subsequent plots will be plotted in black on a white background.

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

Remark 4.2 (Coding style)**.** You could also use the code snippet below to generate a dotplot as R mostly ignores empty spaces and line breaks.

```
ggplot(data=d,aes(x=
Norwegian,y=Participant))+geom_point(
)+xlab("Ergebnis Norwegisch")+ylab("ID Versuchsperson")
```

But the original code snippet is much clearer since the code's structure (including indentations) reflect the logical structure of the command. Further, the use of empty spaces makes the code snippet easier on the eye.

Try to adhere to a clear and consistent coding style, even as you start to dabble in R. You could adopt my style or you could follow a style guide (e.g., `https://style.tidyverse.org/`). ♢

Remark 4.3 (Saving plots)**.** Once you've drawn a plot using `ggplot()`, you can store it using `ggsave()`. Consult this function's help page for guidance (`?ggsave`).

There's also a more general way to save plots, including plots that weren't drawn using `ggplot()`. To save the plot on the left in Figure 4.2, you can put the `ggplot()` commands between the commands `pdf()` and `dev.off()`, like so:

```
pdf(here("figs", "dotchart.pdf"),
    width = 5, height = 4) # width and height in inches
ggplot(data = d,
       aes(x = Norwegian,
           y = Participant)) +
  geom_point() +
  xlab("Norwegian score") +
  ylab("Participant ID")
dev.off()
```

This saves the plot as a PDF file named `dotchart.pdf` in the subdirectory `figs` of your project directory. You can replace `pdf()` by `svg()`, `png()`, `tiff()` or `bmp()` to export your graph to a different format.

Alternatively, you can use the `Export` menu in the `Plots` tab in the bottom right pane in RStudio. However, I encourage you to embrace the use of `pdf()` and its cousins. By using these functions, you document the settings you've used to export the graph. This is a considerable time-saver if you then ever have to redraw the graph with somewhat different settings, for instance in order to satisfy publisher guidelines. ♢

In the present example, the dot plot is particularly useful because of something it does not show: There aren't any data points, or clusters of data points, that lie far from the bulk of the data. Such data points would be called **outliers** and can considerably affect the outcome of an analysis. As the analyst, you need to be aware of their existence. Outliers can (not 'have to') be due to technical errors and it therefore pays to double-check them. Figure 4.3 shows a made-up example the average number of morphological errors per page per learner for a group of learners. One data point lies so far removed from the others that the analyst ought to double-check if it was entered correctly. Don't just remove data points because they are outliers, though!

4.1.2 Histograms

A commonly used and useful plot is the **histogram**. To draw a histogram, we define a number of intervals called bins that jointly cover the range of the observed values. We then count the number of observations that fall within each bin, and visualise this count as rectangles above the respective bins; see Figure 4.4 for an example. As in this example, the vast majority of histograms you'll encounter will use bins of constant width. When drawing a histogram, you need to set the width of the bins or, alternatively, their number, yourself. Finding a suitable bin width is a matter of trial and error, as the following examples illustrate.

```
ggplot(data = d,
       aes(x = Norwegian)) +
  # default settings (always 30 bins)
```

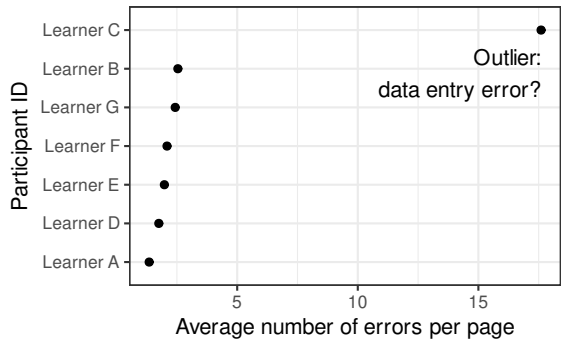


Figure 4.3: An example of an outlier. Here, the analyst ought to check if the outlying data point was entered correctly and isn't just a typo (17.6 instead of 1.76).

```
  geom_histogram()

ggplot(data = d,
       aes(x = Norwegian)) +
  # define number of bins
  geom_histogram(bins = 10)

ggplot(data = d,
       aes(x = Norwegian)) +
  # define bin width
  geom_histogram(binwidth = 3)

ggplot(data = d,
       aes(x = Norwegian)) +
  # define breaks between bins, e.g., 0, 4, 8, 12, 16:
  # seq(from = 0, to = 16, by = 4).
  # You can also override the colours yourself.
  geom_histogram(breaks = seq(from = 0, to = 16, by = 4),
                 fill = "lightgreen",
                 colour = "darkgreen") +
  # Label your axes
  xlab("Ergebnisse cloze-Test Norwegisch") +
  ylab("Anzahl Studierende")
```

Compared to a Cleveland dot plot, a major advantage of the histogram is that it can summarise a large number of observations in a single graph.

Exercise 4.4. Take a closer look at the middle histogram in Figure 4.4. This histogram uses eight bins. These bins are half-open intervals, that is, intervals that include only one

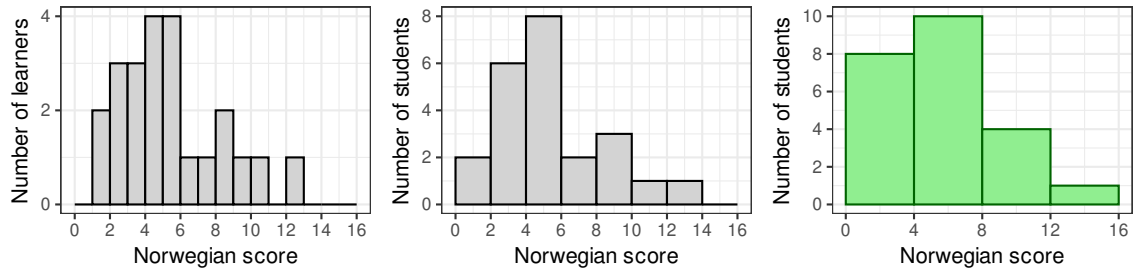


Figure 4.4: Three histograms for the Norwegian scores. In the left plot, the breakpoints between the bins are 0, 1, 2, . . . , 15, 16. In the middle plot, the breakpoints are 0, 2, . . . , 16. In the right plot, they are 0, 4, 8, 12, 16; the histogram is coloured in for good measure. In my opinion, both the plots on the left and the one in the middle are decent choices; the one on the right is a bit too coarse. There aren't any hard and fast rules for choosing the number of bins or their width.

of the endpoints. But are these intervals of the form

$$(0, 2], (2, 4], \ldots, (14, 16],$$

in which the left endpoint isn't included in the interval, or of the form

$$[0, 2), [2, 4), \ldots, [14, 16),$$

in which the right endpoint isn't included in the interval? (Hint: Inspect Figure 4.2.)

Now draw the histogram again, but this time in such a way that the other endpoint belongs to the interval. To this end, you can set the `closed` parameter in `geom_histogram()`. Look up how this parameter works on the function's help page. ♢

In the histograms above, the value along the y-axis show the number of observations in the bin in question. But if you want to compare different histograms (for instance, showing different groups) with each other, it could make sense to convert these counts into a relative frequency or something similar. This doesn't change the shape of the histogram. A popular choice is to choose these relative numbers in such a way that the area of the entire histogram (the sum of the rectangle heights times their width) equals 1. The histogram then shows a density function of sorts (see Section 3.2.2), though empirical distributions are discrete and not absolutely continuous.

Figure 4.5 shows some examples. The bin width in the plot on the right is 4. The rectangle heights are roughly 0.09, 0.105, 0.045, and nearly 0.015. Indeed, $(4 \cdot 0.09) + (4 \cdot 0.105) + (4 \cdot 0.045) + (4 \cdot 0.015) \approx 1$.

To use such relative frequencies ('densities'), set the `y` parameter in the `aes()` call to `after_stat(density)`:

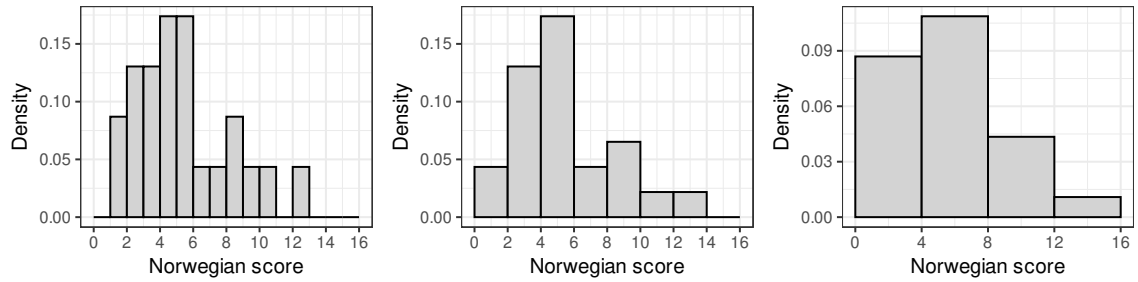


Figure 4.5: Three histograms of the Norwegian scores using densities rather than counts.

```
ggplot(data = d,
       aes(x = Norwegian,
           y = after_stat(density))) +
  geom_histogram(bins = 10)
```

Exercise 4.5 (Densities)**.** Imagine that we were to express the `Norwegian` scores shown in Figure 4.5 as percentages. A score of 10 corresponds to a percentage of 50, one of 14 to a percentage of 70, etc. That is, multiply the Norwegian score by 5 to obtain the percentage. We then draw a histogram using four bins spanning the percentage range of 0 to 80.

Compute the densities for each bin without drawing the histogram.

What would the densities be if we had expressed the same scores as proportions rather than as percentages? ♢

4.2 Numerical descriptions

4.2.1 *Quantiles

In the previous chapter, we encountered the quantile *function* (Section 3.2.1). We can define the notion of a quantile more generally, though:

Definition 4.6 (Quantiles)**.** Let X be a random variable and let $\gamma \in (0,1)$. A number q is a **γ-quantile** of the distribution of X if

$$\mathbb{P}(X \leq q) \geq \gamma$$

and

$$\mathbb{P}(X \geq q) \geq 1 - \gamma.$$

For $p \in (0,100)$, we say that a number q is a **p-percentile** of the distribution of X if q is a $p/100$-quantile of the distribution of X.

A number q is a **first/second/third quartile** if it is a 0.25-/0.50-/0.75-quantile. A second quartile (i.e., a 0.50-quantile) is also called a **median**. ♢

For the Norwegian data, we see that 21.7% of the participants had a score no higher than 3, and that 91.3% had a score no lower than 3.

```
mean(d$Norwegian <= 3)

[1] 0.2173913

mean(d$Norwegian >= 3)

[1] 0.9130435
```

Since $0.217 \geq 0.2$ and $0.913 \geq 1 - 0.2$, 3 is a 0.2-quantile of the empirical distribution of the Norwegian scores. But note that 3 is also a 0.1-quantile of this distribution! Further, as you can verify using the definition, 3.4 is also both a 0.1- and a 0.2-quantile of this distribution. By contrast, neither 3 nor 3.4 are 0.05- or 0.25-quantiles, for $0.217 < 0.25$ and $0.913 < 1 - 0.05$.

As these examples show, quantiles defined in this sense aren't necessarily unique. Consequently, the built-in `quantile()` function, which can be used to obtain quantiles of an empirical distribution, accommodates different algorithms for choosing a single number from among the γ-quantiles. See the `type` parameter on the help page of the `quantile()` function.

```
# different choices of type lead to different quantiles output
quantile(d$Norwegian, probs = 0.2, type = 1)

20%
  3

quantile(d$Norwegian, probs = 0.2, type = 5)

20%
3.1

quantile(d$Norwegian, probs = 0.2, type = 7)

20%
3.4

quantile(d$Norwegian, probs = 0.2, type = 9)

  20%
3.025
```

4.2.2 Averages

Averages (or a **measures of central tendency**) represent attempts to express numerically what a typical value of a distribution or of a sample is. Since 'typical' is a pretty vague term, different kinds of average exist. We'll take a look at some of the most popular ones, but the list is far from exhaustive.

The arithmetic mean

When talking about the 'average', people typically have the **(arithmetic) mean** in mind. The mean of a sample is the expected value of the empirical distribution defined by this sample. In practical terms, the sample mean $\overline{x}$ of a sample $\boldsymbol{x} = (x_1, \ldots, x_n)$ is computed as

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i.$$

For the Norwegian data,

```
sum(d$Norwegian) / length(d$Norwegian)

[1] 5.913043
```

Or more simply:

```
mean(d$Norwegian)

[1] 5.913043
```

The mean inherits from the expected value introduced in the previous chapter the linearity property.

A second useful property is the following one. Let $\boldsymbol{x} = (x_1, \ldots, x_m)$ and $\boldsymbol{y} = (y_1, \ldots, y_n)$ be samples. We concatenate these samples: $\boldsymbol{z} = (x_1, \ldots, x_m, y_1, \ldots, y_n)$. Then the mean of $\boldsymbol{z}$ is the **weighted mean** of $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$\overline{z} = \frac{m\overline{x} + n\overline{y}}{m + n}.$$

A third useful property of the sample mean concerns the Central Limit Theorem (Section 3.5). This theorem allows us to make approximate probabilistic claims about where the mean of a random sample from a distribution will end up based solely on the expected value and the variance of this distribution.

The main drawback of the mean is that it is highly sensitive to outliers. Indeed, a single outlying value can cause the mean to lie far from the bulk of the data, in which case the mean fails to be a good measure of central tendency. To see this, consider the data in Figure 4.3. The mean of the reported values is about 4.2—even though six of the seven values are no larger than 2.6 and the remaining value is considerably higher than 4.2. If we remove the outlying value, the mean of the remaining six values is about 2.0, which appropriately captures the central tendency of these six values. The problem, however, is that it is not always clear whether it is defensible to remove the outlying values before computing the mean.

The median

Another commonly used kind of average is the **median**. As mentioned in Definition 4.6 on page 106, *a* median is any value q such that at least half of the observed values are no greater than it and at least half of the observed values are no lower than it. If

the sample consists of an even number of observations, it is possible for there to be no unique median. In this case, the mean of all the medians is computed and referred to as *the* median.

In practical terms, to compute the median, sort the observations by their value. If there are an uneven number of observations (i.e., $n = 2m + 1$ for some natural number m), take the middle value (i.e., the $m + 1$-th one) as the median. Else, if there are an even number of observations (i.e., $n = 2m$), take the mean of the two middle values (i.e., the m-th and the $m + 1$-th) as the median.[2] For the Norwegian data, $n = 23 = 2 \cdot 11 + 1$, so we sort the observations and read out the 12th one:

```
sort(d$Norwegian)[12]

[1] 5
```

Of course, we can use the built-in `median()` function:

```
median(d$Norwegian)

[1] 5
```

The median is highly robust to outlying values. For the data in Figure 4.3, the median with the outlying value is 2.09; the median without the outlier is 2.035. Both of these values are good indicators of where the bulk of the observations lie in this example.

However, the median doesn't have any of the nice properties that the mean has. First, the median is not linear. For instance, the medians of $\boldsymbol{x} = (-1, 0, 1)$ and $\boldsymbol{y} = (0, -1, 1)$ are both 0. But the median of $\boldsymbol{x} + \boldsymbol{y} = (-1, -1, 2)$ is -1.

Second, the median of a concatenated vector $(\boldsymbol{x}, \boldsymbol{y})$ can't be computed from the medians of $\boldsymbol{x}$ and $\boldsymbol{y}$. For instance, the median of $\boldsymbol{x} = (0, 1)$ is 0.5; that of $\boldsymbol{y} = (0, 0)$ is 0. But the median of $(0, 1, 0, 0)$ is again 0, not 0.25 or something similar.

Third, the Central Limit Theorem does not apply to the sample median.[3]

The absence of these nice properties makes the median more difficult to handle mathematically than the mean.

Large differences between the mean and the median are often due to outliers or to asymmetric distributions. In either case, don't compute averages without having visualised the data first. Even though the computations may be technically correct, they may not make much sense.

The mode

The mode is encountered less frequently than the mean and the median. It defines the most typical values quite straightforwardly as those that occur most often. There is no

[2] More efficient computer algorithms exist that do not rely on sorting.

[3] Some CLT-like statement for the sample median does, in fact, exist. But it applies only to distributions whose distribution function F is differentiable at $1/2$ and it requires knowledge of this derivative $F'(1/2)$.

built-in mode function in R, but we can tabulate the frequencies of occurrence of the different Norwegian scores straigthforwardly:

```
table(d$Norwegian)


 2  3  4  5  6  7  8  9 10 11 13
 2  3  3  4  4  1  1  2  1  1  1

sort(table(d$Norwegian))


 7  8 10 11 13  2  9  3  4  5  6
 1  1  1  1  1  2  2  3  3  4  4
```

We see that the values 5 and 6 both occur four times, whereas all other values occur less frequently. Hence, 5 and 6 are the modes of the Norwegian scores.

If you're dealing with fine-grained data, each value will likely only occur once or twice in the data. For such data, it doesn't make much sense to compute the mode as defined here.

Exercise 4.7. The file `stocker2017.csv` contains part of the data from an on-line study by Stocker (2017). She asked 160 participants to rate the credibility of claims uttered by talkers with different accents (English, French, German, Italian) on a scale from 0 to 100 by means of a slider. These responses are contained in the `score` column.

1. Read in this file in R.
2. Compute the mean and the median of the `score` data. Are they similar?
3. Visualise the distribution of the `score` data in a histogram with 10 bins. Describe the form of this histogram.
4. Visualise the distribution of the `score` data in a histogram with 100 bins. Describe the form of this histogram. Are the mean and the median good measures of the central tendency of these data?
5. Which value is the third most frequent? Why is that, do you think?
6. What do the fourth, fifth, sixth etc. most frequent values have in common? ♢

***Other averages**

Several other kinds of average exist. Some of these, such as the **harmonic mean** and the **geometric mean**, are useful in situations where neither the ordinary (arithmetic) mean nor the median gives a meaningful answer—for example, when combining rates, ratios, or growth factors. These two averages are introduced in the following exercises.

Exercise 4.8 (Harmonic mean)**.** Say you travelled three kilometres. You completed the first kilometre at 5 km/h, the second at 10 km/h and the third at 2 km/h. As you can

verify, you needed 48 minutes to complete these three kilometres. Hence, your average speed was 3.75 km/h. You can compute this average speed using the harmonic mean.

Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ be a vector with strictly positive entries. The harmonic mean of $\boldsymbol{x}$ is defined as

$$H(\boldsymbol{x}) = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}.$$

Write your own R function called `harmonic_mean()` that takes in a vector of arbitrary length containing strictly positive entries and outputs their harmonic mean.

Hint: If your function works correctly, the following command should output 3.75:

```
harmonic_mean(c(5, 10, 2))
```

$\diamondsuit$

Exercise 4.9 (Geometric mean)**.** Let's say you buy CHF 1,000 worth of shares of some company. At the end of the first year, the shares are worth CHF 1,100, for an increase of 10%. At the end of the second year, the shares' worth has dropped to CHF 1,045—a 5%-drop relative to the previous year. One year later, the shares are worth CHF 1,139—a 9%-increase relative to the previous year. Over the course of these three years, the shares' annualised return is about 4.44%. This means that, starting with a CHF 1,000 investment, you'd obtain CHF 1,139 after three years if the yearly growth rate was constant at 4.44% ($1000 \cdot 1.0444 \cdot 1.0444 \cdot 1.0444 \approx 1139$). This differs from the average of the yearly returns (about 3.5%). The annualised return can be computed as the geometric mean of the ratios of the closing price to the opening price for each year.

Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ be a vector with strictly positive entries. The geometric mean of $\boldsymbol{x}$ is defined as

$$G(\boldsymbol{x}) = \sqrt[n]{x_1 x_2 \ldots x_n} = (x_1 x_2 \ldots x_n)^{1/n},$$

that is, as the n-th root of the product of the entries.

Write your own R function called `geometric_mean()` that takes in a vector of arbitrary length containing strictly positive entries and outputs their geometric mean.

Hint: If your function works correctly, the following command should output 1.0444.

```
geometric_mean(c(1.1, 0.95, 1.09))
```

$\diamondsuit$

Then, there is a whole class of averages that are designed to reduce the influence of extreme values on the result while still retaining some more sensitivity to differences in the data than the median. Two examples are the **trimmed mean** and the **winsorised mean**. Admittedly, you'll rarely explicitly run across these in the language sciences.

Definition 4.10 (Trimmed mean)**.** Let $\boldsymbol{x} = (x_1, \ldots, x_n)$ be a numeric vector. For $\alpha \in (0, 0.5)$, the trimmed mean with trimming factor α is defined as

$$\overline{x}_\alpha := \frac{1}{n-2k} \sum_{i=k+1}^{n-k} x_{(i)},$$

where $k = \lfloor \alpha n \rfloor$ (i.e., αn rounded down to the nearest integer) and $(x_{(1)}, \ldots, x_{(n)})$ is the vector with the same entries as $\boldsymbol{x}$ but sorted by ascending value.

For $\alpha = 0$, the trimmed mean is defined as the arithmetic mean; for $\alpha = 0.5$, it is defined as the median. ♢

For instance, the Norwegian data have $n = 23$. If we set $\alpha = 0.1$, then $k = \lfloor 0.1 \cdot 23 \rfloor = \lfloor 2.3 \rfloor = 2$. To compute the trimmed mean with trimming factor 0.2, then, we compute mean of the Norwegian data but ignoring the two smallest and the two largest observations:

```
(sort(d$Norwegian)[-c(1, 2, 22, 23)] |> sum()) / (23 - 2*2)

[1] 5.684211

mean(d$Norwegian, trim = 0.1)

[1] 5.684211
```

Trimmed means, then, are a compromise between the mean and the median. Winsorised means serve a similar purpose.

Definition 4.11 (Winsorised mean)**.** Let $\boldsymbol{x} = (x_1, \ldots, x_n)$ be a numeric vector, and let $\alpha \in (0, 0.5)$ be a trimming factor. Define $k = \lfloor \alpha n \rfloor$. Let $(x_{(1)}, \ldots, x_{(n)}$ be the vector with the same entries as $\boldsymbol{x}$ but sorted by ascending value. In this vector, replace the values $x_{(1)}, \ldots, x_{(k)}$ by the value of $x_{(k+1)}$, and replace the values $x_{(n-k+1)}, \ldots, x_{(n)}$ by the value of $x_{(n-k)}$. The arithmetic mean of the vector so created is the winsorised mean of $\boldsymbol{x}$ using trimming factor α. ♢

If we again use a trimming factor of $\alpha = 0.1$ for the Norwegian data, we would compute winsorised mean like so:

```
n <- length(d$Norwegian) # i.e., 23
k <- floor(0.1 * n) # i.e., 2
norwegian_sorted <- sort(d$Norwegian)
norwegian_sorted[1:k] <- norwegian_sorted[k+1]
norwegian_sorted[(n-k+1):n] <- norwegian_sorted[n-k]
mean(norwegian_sorted)

[1] 5.826087
```

4.2.3 Measures of spread

Measures of spread seek to quantify how strongly the observations tend to deviate from their central tendency.

The range

A very simple measure of spread is the **range**, defined as the difference between the largest and the lowest observed value.

```
min(d$Norwegian)

[1] 2

max(d$Norwegian)

[1] 13

range(d$Norwegian) # interval of min, max

[1]  2 13

range(d$Norwegian) |> diff() # length of interval

[1] 11
```

The range, or the interval it spans, are sometimes reported in summary tables, but it is rarely used as the measure of spread of choice. This is for good reason: It is highly sensitive to outliers. For a critique on the use of the range in the field of second language acquisition, see Vanhove (2020b).

The variance

We've defined the variance in the previous chapter. We can apply this definition to the empirical distibution defined by the sample. This quantity is called the **uncorrected (sample) variance**, sometimes written as s_n^2 (n for 'naïve'), and can be computed as follows:

$$s_n^2(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2.$$

For the Norwegian scores, the variance of the empirical distribution is about 8.17:

```
mean((d$Norwegian - mean(d$Norwegian))^2) # cf. definition

[1] 8.166352

mean(d$Norwegian^2) - mean(d$Norwegian)^2 # cf. lemma

[1] 8.166352
```

For reasons that will become clearer in the next chapter, though, the **(corrected) (sample) variance** is typically used instead. It is defined as

$$s^2(\boldsymbol{x}) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2,$$

that is, with $n-1$ instead of n in the denominator. For large n, s_n^2 and s^2 hardly differ, though.

The built-in `var()` function computes the corrected variance:

```
sum((d$Norwegian - mean(d$Norwegian))^2) / (n - 1)
[1] 8.537549
var(d$Norwegian)
[1] 8.537549
```

The standard deviation

Similarly, we can apply the definition of the standard deviation provided in the previous chapter to the empirical distribution of $\boldsymbol{x}$. This results in the **uncorrected (sample) standard deviation**, and we have

$$s_n(\boldsymbol{x}) = \sqrt{s_n^2(\boldsymbol{x})}.$$

Typically, though, the **(corrected) (sample) standard deviation** is used instead, which is defined as

$$s(\boldsymbol{x}) = \sqrt{s^2(\boldsymbol{x})}.$$

The built-in `sd()` function computes the corrected standard deviation:

```
sqrt(sum((d$Norwegian - mean(d$Norwegian))^2) / (n - 1))
[1] 2.921909
sd(d$Norwegian)
[1] 2.921909
```

Both the variance and the standard deviation are highly sensitive to outliers. For instance, for the numbers shown in Figure 4.3 on page 104, the variance with the outlier is about 34.8; without, it is only 0.19.

Remark 4.12 (Don't summarise data purely numerically!)**.** Imagine reading a study in which 39 respondents answered a question on a 6-point scale from 0 to 5. The study reports that their mean response was 2.43.

Perhaps you picture in your mind's eye that most respondents ticked '2' or '3' as their response. But many different data patterns can produce the same mean of 2.43 (see Figure 4.6), and you'd have to draw different conclusions depending on the data pattern. Perhaps the respondents were unanimous in picking response options in the middle of the scale—possibly suggesting widespread indifference. Or perhaps the question touched on a hugely controversial topic, causing respondents to pick the extreme response options. Alternatively, a whole range of opinions and strength of conviction may be represented among the 39 respondents.

If, in addition to the mean, a standard deviation is reported, the number of possible data patterns compatible with the numerical summary is reduced. But nonetheless, a variety

Figure 4.6: Three different distributions with 39 observations. All three have the same mean after rounding.

of different data patterns can give rise to the same mean and the same standard deviation (see Figure 4.7).[4]

The upshot of this is clear: Draw graphs of your data so that both you and your audienne knows what they actually look like. Averages and measures of spread don't tell the whole story. Don't blindly run computations. ♢

***The mean and median absolute deviation**

The standard deviation isn't *quite* the average deviation of the observations from their mean. The **mean absolute deviation around the mean**, however, does exactly what it says on the tin: On average, the Norwegian data lie about 2.26 points removed from the mean.

```
(d$Norwegian - mean(d$Norwegian)) |>
  abs() |> # absolute differences between obs. and mean
  mean()

[1] 2.257089
```

Similarly, you can compute, say, the **mean absolute deviation around the median**, or around any other measure of central tendency that you're interested in.

```
(d$Norwegian - median(d$Norwegian)) |>
  abs() |> # absolute differences between obs. and median
  mean()

[1] 2.217391
```

By the same token, the **median absolute deviation** around a central tendency measure can be computed, e.g., around the median:

[4] I generated these distributions using the R code provided by Richard Morey on `https://bayesfactor.blogspot.com/2016/03/how-to-check-likert-scale-summaries-for.html`.

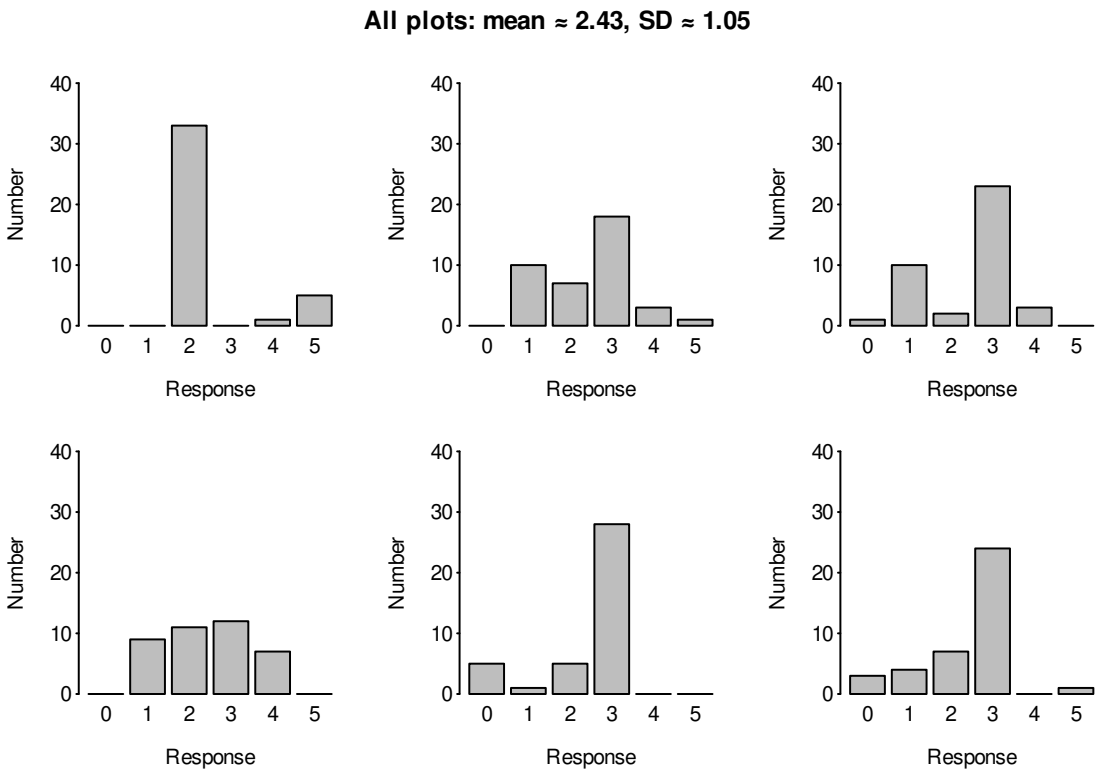


Figure 4.7: Six different distributions of 39 observations. All six have the same mean and the same standard deviation after rounding.

```
(d$Norwegian - median(d$Norwegian)) |>
  abs() |>
  median()

[1] 2
```

R's `mad()` function by default computes the median absolute deviation around the median but multiplies the result by 1.4826. The reason for this seemingly odd choice is that the so-adjusted median absolute deviation around the median approximately agrees with the standard deviation for large samples of normally distributed data. While there are some practical use cases for this, this adjustment makes the result more difficult to interpret for non-statisticians.

Of the measures of spread introduced above, the median absolute deviation around the median is the least sensitive to outliers. For the data in Figure 4.3, it is 0.34 both with and without the outlier. The other mean and median absolute deviations change an order of magnitude depending on whether the outlier is in- or excluded.

4.3 *Further reading

Huff (1954) (*How to lie with statistics*) is a short and entertaining booklet that's still worth reading. Among other things, it discusses how different kinds of averages are used manipulatively in everyday life.

Healy (2019) is an excellent resource for learning how to think about graphs and then draw them using `ggplot2`. This book is also freely available from `https://socviz.co/`. I also wrote a tutorial myself that focuses exclusively on drawing descriptive graphs. You can find it on `https://github.com/janhove/DatasetsAndGraphs`.

Part II

Estimates

Chapter 5

Random samples

When you've specified a probability distribution, you can make probabilistic claims about random variables that are distributed according to this distribution. **Inferential statistics**, though, is concerned with the inverse problem: Having observed some random variables (i.e., data points), what conclusions can we draw about the probability distribution they follow?

This probability distribution could correspond to some population that we want to learn something about. For instance, the population in question could be all Swiss citizens with the vote, and we'd like to figure out what percentage of them will vote in favour of the next federal referendum. Often, though, it makes more sense to think of the probability distribution not as a literal population, but as an abstract data-generating mechanism: a theoretical model that connects the data we see to the real-world process we want to study. For instance, an astronomer could make a series of imperfect measurements about a celestial body's trajectory. The theoretical data-generating mechanism would then spell out how the trajectory and the imperfect measurements are presumed to be related to one another. The imperfect measurements can then be used to estimate the specifics of the trajectory.

Typically, the data collected are only a small part of the data that could have been collected—be it because the population of interest is much larger or because the abstract data-generating mechanism could in principle generate an infinite amount of data. Such limited collections of data are known as **samples**, and they necessarily only give you an incomplete picture about the distribution from which they stem. The goal of inferential statistics, then, is to use samples to infer properties about the parent distributions. Consequently, it's not so much, say, the central tendency and the spread in the sample that's of interest, but rather the central tendency and the spread of the parent distribution.

In the following, we assume that we have access to a (simple) random sample and that the parent distribution is either a data-generating mechanism capable of producing an infinite amount of data or a population that can for all intents and purposes be considered infinitely large.

Definition 5.1. Let P be a probability distribution. Then a **(simple) random sample** from P of size n consists of independent random variables $X_1, \ldots, X_n$ that are all distributed according to P. ♢

This chapter focuses on the estimation of the central tendency of the parent distribution P (in particular its expected value) as well as its spread (in particular its variance and standard deviation) by means of a random sample. In practice, though, you rarely come across truly random samples; we'll discuss this further at the end of this chapter.

5.1 Sampling error

Random samples don't perfectly reflect every aspect of their parent distribution. To better appreciate this, we'll draw some random samples from distributions whose properties we know and see how different they look both from each other and from their parent distributions.

Activity 5.2 (Normal distribution)**.** The commands below draw nine random samples of size 20 from a normal distribution with mean 3 and standard deviation 7 and plot these in histograms.

```
size <- 20
mu <- 3
sigma <- 7

par(mfrow = c(3, 3)) # multiple graphs in one plot
for (i in 1:9) {
  x <- rnorm(n = size, mean = mu, sd = sigma)
  hist(x, main = paste0("Sample ", i), freq = FALSE)
}
par(mfrow = c(1, 1)) # back to default plotting
```

Run these commands a couple of times, also varying the sample sizes. Pay attention to the differences between the histograms as well as to how dissimilar they are to the normal distribution's bell-shaped density function. ♢

Exercise 5.3 (Uniform distribution)**.** Adapt the code from Activity 5.2 in such a way that it draws and plots random samples from a continuous uniform distribution on $[-5, 5]$. Run the code a couple of times, varying the sample size. Again pay attention to the differences between the histograms as well as to how dissimilar these are to the uniform distribution's flat density function. ♢

In sum, random samples reflect the distributions from which they are drawn imperfectly. This fact is referred to as **sampling error**. Consequently, any inferences we draw about the parent distribution based on a random sample are estimates. Inferential statistics is concerned both with coming up with good estimation techniques as well as with quantifying the inherent uncertainty about the resulting estimates.

5.2 Estimating the expected value

In the following, we assume that the distribution P that we've drawn a random sample from has an expected value (mean) and finite variance. As we've seen in Chapter 4, given a random sample $\boldsymbol{X} = (X_1, \ldots, X_n)$ from P, we can define the empirical distribution $\widehat{P}$ of $\boldsymbol{X}$. A natural way to estimate properties of P is to calculate the corresponding properties of $\widehat{P}$. This raises the question of how good such so-called *plug-in* estimators are.

Definition 5.4 (Estimand, estimate, estimator)**.** An **estimand** is a numerical property of a distribution that we want to estimate using data.

An **estimate** is a number that is based on data and prior knowledge and that is intended as an approximation of an estimand.

An **estimator** is an algorithm that specifies how data and prior knowledge are to be combined in order to produce an estimate. ♢

For instance, the (unknown) expected value of a distribution P could be one's estimand of interest. A popular estimator would then be the instruction 'sum all observations in the sample and divide the sum by the number of observations'. Since this is precisely how you'd compute the expected value of $\widehat{P}$, this estimator is the plug-in estimator of the expected value. The resulting number is the corresponding estimate.

We can't make general claims about specific estimates. But we can meaningfully compare different estimators on a number of quality criteria. The most important of these are the estimator's bias, consistency, and variance.

Definition 5.5 (Bias)**.** Consider an estimator g and estimand θ of a distribution P. The **bias** of g as an estimator of θ is defined to be

$$\mathbb{E}(g(X_1, \ldots, X_n)) - \theta,$$

where $X_1, \ldots, X_n$ are independently distributed according to P.

If $\mathbb{E}(g(X_1, \ldots, X_n)) = \theta$ for each possible value of θ, then g is known as an **unbiased** estimator of θ. ♢

In other words, an unbiased estimator yields the correct answer on average.

To give a precise definition of consistency, we'd have to introduce some further mathematical concepts. For our purposes, though, the following more conceptual definition suffices.

Definition 5.6 (Consistency)**.** An estimator g is a **consistent** estimator of θ if

$$\mathbb{E}\left(|g(X_1, \ldots, X_n) - \theta|\right)$$

becomes arbitrarily small for large n, for all possible values of θ. ♢

That is, the differences between the estimates produced by a consistent estimator and the true parameter value are expected to become negligible as the sample size increases.

Definition 5.7 (Variance of an estimator)**.** The variance of an estimator g is defined as

$$\mathrm{Var}(g(X_1, \ldots, X_n)). \qquad \diamondsuit$$

That is, an estimator's variance expresses how much the resulting estimates vary from sample to sample.

Ideally, estimators are unbiased and consistent, and they should have low variance. However, not all estimands permit such ideal estimators. Further, as you'll see in Exercise 5.12, you can often decrease the variance by sacrificing unbiasedness. This phenomenon is known as the **bias–variance trade-off**.

***Example 5.8** (Unbiased but useless)**.** Unbiased estimators can be inconsistent, rendering them pretty useless for practical purposes. By way of an example, consider $P = \mathrm{Binomial}(n, p)$ and define $\theta := (1-p)^n$ (Dümbgen, 2016, p. 49). We observe $X \sim P$. One can show that there exists only one unbiased estimator $g(X)$ of θ, namely

$$g(X) := \begin{cases} 1, & \text{if } X = 0, \\ 0, & \text{otherwise.} \end{cases}$$

But depending on p, θ can take on any value in the interval $[0, 1]$. The value of $g(X)$, by contrast, is always either 0 or 1. As a result, g is not a consistent estimator of θ. $\diamondsuit$

The **sample mean** is the expected value of the sample's empirical distribution. It serves as an estimate of the parent distribution's expected value μ:

$$\widehat{\mu} := \overline{X} := \frac{1}{n}(X_1 + \cdots + X_n).$$

In Section 3.5, we showed that

$$\mathbb{E}(\overline{X}) = \mu.$$

Consequently, the sample mean is an unbiased estimator of the parent distribution's expected value. Further, we know that the variance of $\overline{X}$ equals $\frac{\sigma^2}{n}$, where σ^2 is the parent distribution's variance. Hence, the variance of $\overline{X}$ becomes arbitrarily small for large n. This combined with the unbiasedness of the sample mean implies that the sample mean is a consistent estimator of μ. Finally, the Gauss–Markov theorem tells us that, among all linear unbiased estimators of the population's expected value, the sample mean is the one with the lowest variance.[1]

Theorem 5.9 (Gauss–Markov)**.** Let $X_1, \ldots, X_n$ be a simple random sample from a distribution P. Then each linear unbiased estimator $g(X_1, \ldots, X_n)$ of μ satisfies

$$\mathrm{Var}(\overline{X}) \leq \mathrm{Var}(g(X_1, \ldots, X_n)). \qquad \diamondsuit$$

It is, however, sometimes possible to come up with linear unbiased estimators with lower variance if the sample is not a simple random sample.

[1] Given a data vector $\boldsymbol{X} = (X_1, \ldots, X_n)$, a linear estimator $g(\boldsymbol{X})$ can be written as $g(\boldsymbol{X}) = w_1 X_1 + \cdots + w_n X_n$ for certain coefficients $w_1, \ldots, w_n$. For the sample mean, $w_1 = \cdots = w_n = 1/n$.

Example 5.10. Let $X_1, \ldots, X_n$ be independent Bernoulli(p)-distributed random variables. Then $\frac{1}{n}(X_1 + \cdots + X_n)$ is an unbiased and consistent estimator of p. Among all linear unbiased estimators of p, it has the lowest variance. $\diamondsuit$

Example 5.11 (Weighted mean)**.** We assume that P_1, P_2 are distributions with the same unknown mean μ, but different and known variances: $\mathrm{Var}(P_1) = 1, \mathrm{Var}(P_2) = 4$. We now observe independent random variables $X_1 \sim P_1, X_2 \sim P_2$ and want to use these to estimate μ.

The first option is to estimate μ as the mean of X_1, X_2. This estimator is unbiased as

$$\mathbb{E}\left(\frac{X_1 + X_2}{2}\right) = \frac{1}{2}(\mathbb{E}(X_1) + \mathbb{E}(X_2)) = \frac{1}{2}(\mu + \mu) = \mu.$$

Since X_1, X_2 are independent, the variance of this estimator is easily calculated:

$$\mathrm{Var}\left(\frac{X_1 + X_2}{2}\right) = \frac{1}{4}(\mathrm{Var}(X_1) + \mathrm{Var}(X_2)) = \frac{5}{4}.$$

A second option is to just ignore X_2 and use the value of X_1 as the estimate of μ. This estimator, too, is unbiased: $\mathbb{E}(X_1) = \mu$. Furthermore, its variance is lower than that of the first estimator: $\mathrm{Var}(X_1) = 1 < 5/4$.

But it would be even better to weight the observations X_1, X_2 by the inverses of their variance:

$$g(X_1, X_2) := \frac{1}{1 + \frac{1}{4}}\left(X_1 + \frac{1}{4}X_2\right) = \frac{4}{5}\left(X_1 + \frac{1}{4}X_2\right).$$

This estimator is also unbiased:

$$\mathbb{E}(g(X_1, X_2)) = \frac{4}{5}\left(\mathbb{E}(X_1) + \frac{1}{4}\mathbb{E}(X_2)\right) = \mu.$$

However, its variance is lower:

$$\mathrm{Var}(g(X_1, X_2)) = \frac{4^2}{5^2}\left(\mathrm{Var}(X_1) + \frac{1}{4^2}\mathrm{Var}(X_2)\right) = \frac{4}{5}.$$

The Gauss–Markov theorem does not apply here since X_1, X_2 do not represent a simple random sample: The observations stem from different distributions.

This example corresponds to the realistic scenario in which two random samples are drawn from the same distribution with mean μ and variance σ^2, but we only know their sample means X_1, X_2 and their sample sizes n_1, n_2. In this case, X_1, X_2 are distributed with the same mean μ but different variances σ^2/n_1 and σ^2/n_2. The expected value of the parent distribution is then best estimated using the weighted mean. $\diamondsuit$

***Exercise 5.12** (Comparison of estimators of μ)**.** Let $X_1, \ldots, X_n$ be a simple random sample from a distribution P with expected value μ and with finite variance. Decide for each of the estimators below if they estimate μ in an unbiased way, if they are consistent, and

if their variance is smaller or larger than that of the sample mean $\overline{X}$. You may assume in the following that $\mathbb{P}(X_1 = X_2) = 0$; you may also use the fact that $1/n$ becomes arbitrarily small for large n.

$$
\begin{aligned}
g_1(X_1, \ldots, X_n) &:= X_1; \\
g_2(X_1, \ldots, X_n) &:= \overline{X} + \frac{1}{n}; \\
g_3(X_1, \ldots, X_n) &:= \frac{n-1}{n}\overline{X}; \\
g_4(X_1, \ldots, X_n) &:= \begin{cases} \overline{X} - 1, & \text{if } X_1 < X_2, \\ \overline{X}, & \text{if } X_1 = X_2, \\ \overline{X} + 1, & \text{if } X_1 > X_2. \end{cases}
\end{aligned}
$$

Which of the estimators above could potentially be practically useful?

In addition, define an estimator g_5 that estimates μ in an unbiased and consistent way, but that has a larger variance than the sample mean. ♢

5.3 Estimating the variance and the standard deviation

The previous section showed that the plug-in estimator of the expected value has excellent properties. We'll now check if the plug-in estimator of the variance excels similarly. To that end, we consider a random sample $X_1, \ldots, X_n$ from a continuous uniform distribution on $[-5, 5]$. The variance of this distribution is

$$\sigma^2 = \frac{(5 - (-5))^2}{12} \approx 8.33.$$

The function `var_one_run()` defined below generates a single random sample of size `n` from this distribution and computes the plug-in estimate of the variance. The `sim_var()` function runs `var_one_run()` a couple of thousand times and outputs the mean of the variance estimates.

```
var_one_run <- function(n, min = -5, max = 5) {
  x <- runif(n, min = min, max = max)
  mean(x^2) - mean(x)^2
}
sim_var <- function(n_sample, min = -5, max = 5, n_runs = 50000) {
  sample_vars <- replicate(n_runs,
                           var_one_run(n = n_sample, min = min, max = max))
  mean(sample_vars)
}
```

For $n = 25$, we obtain the following mean variance estimate:

```
sim_var(n_sample = 25)

[1] 7.992254
```

Since this number is based on random sampling, you'll obtain a slightly different result.

Activity 5.13 (Varying n)**.** Run the same simulation for $n = 16, 9, 4, 1$. You don't have to make any changes to `var_one_run()` or `sim_var()`! Compare the mean variance estimate you obtain for each n to the estimand's true value of $\sigma^2 \approx 8.33$. ♢

As this simulation illustrates, the plug-in estimator of the variance is biased: On average, it produces estimates that are lower than the estimand's true value. This bias is more pronounced in smaller samples. As the next lemma shows, for a sample of size n, the bias of the ('naive') plug-in estimator s_n^2 has a negative bias of σ^2/n. This naive estimator is, however, consistent.

***Lemma 5.14** (Bias of the naive variance estimator)**.** Let $\boldsymbol{X} = (X_1, \ldots, X_n)$ be simple random sample from a distribution P with expected value μ and variance $\sigma^2 < \infty$. Denote by $\overline{X}$ the sample mean of $\boldsymbol{X}$. Then

$$s_n^2(\boldsymbol{X}) = \frac{1}{n}\sum_{i=1}^{n}\left(X_i - \overline{X}\right)^2$$

is a biased estimator of σ^2. Specifically,

$$\mathbb{E}\left(s_n^2(\boldsymbol{X})\right) = \frac{n-1}{n}\sigma^2. \qquad ♢$$

Proof. We have

$$\begin{aligned}
\frac{1}{n}\sum_{i=1}^{n}\left(X_i - \overline{X}\right)^2 &= \frac{1}{n}\sum_{i=1}^{n}\left((X_i - \mu) - (\overline{X} - \mu)\right)^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}(X_i - \mu)^2 - \frac{2(\overline{X} - \mu)}{n}\sum_{i=1}^{n}(X_i - \mu) + \frac{1}{n}\sum_{i=1}^{n}(\overline{X} - \mu)^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}(X_i - \mu)^2 - 2(\overline{X} - \mu)^2 + (\overline{X} - \mu)^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}(X_i - \mu)^2 - (\overline{X} - \mu)^2.
\end{aligned}$$

These identities can be verified by applying the binomial formula $(a-b)^2 = a^2 - 2ab + b^2$ and using $\overline{X} = (\sum_{i=1}^{n} X_i)/n$.

Taking the expected values on both sides, we obtain

$$\begin{aligned}
\mathbb{E}\left(\frac{1}{n}\sum_{i=1}^{n}\left(X_i - \overline{X}\right)^2\right) &= \mathbb{E}\left(\frac{1}{n}\sum_{i=1}^{n}(X_i - \mu)^2 - (\overline{X} - \mu)^2\right) \\
&= \frac{1}{n}\sum_{i=1}^{n}\underbrace{\mathbb{E}(X_i - \mu)^2}_{=\mathrm{Var}(X_i)=\sigma^2} - \underbrace{\mathbb{E}\left((\overline{X} - \mu)^2\right)}_{=\mathrm{Var}(\overline{X})=\sigma^2/n} \\
&= \sigma^2 - \sigma^2/n \\
&= \frac{n-1}{n}\sigma^2. \qquad \square
\end{aligned}$$

The lemma suggests a straightforward way to correct the naive variance estimator so that it becomes unbiased: Multiply its estimates by the factor $n/(n-1)$. This results in the sample variance that was already introduced in the previous chapter:

$$s^2(\boldsymbol{X}) = \frac{n}{n-1} s_n^2(\boldsymbol{X}) = \frac{n}{n-1}\left(\frac{1}{n}\sum_{i=1}^{n}(X_i - \overline{X})^2\right) = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2.$$

The estimator s^2 estimates σ^2 in an unbiased and consistent way; we won't prove consistency, however. It can be computed using `var()`.

While the central limit theorem allows us to make approximate claims about the distribution of the sample mean, the corresponding theorem about the large-sample distribution of the sample variance is less practically useful. For samples drawn from a normal distribution, however, precise claims can be made. These are introduced in the following optional paragraphs.

***Remark 5.15.** Let $Z_1, \ldots, Z_n$ be independent random variables following a standard normal distribution. Then the distribution of $Z_1^2 + \cdots + Z_n^2$ is called the **chi-squared distribution** with n degrees of freedom. We write χ_n^2. Figure 5.1 shows density functions of a few χ^2-distributions.

Now, if $\boldsymbol{X} = (X_1, \ldots, X_n)$ is a simple random sample drawn from a normal distribution with mean μ and variance σ^2, then it can be shown that the quantity

$$(n-1)\frac{s^2(\boldsymbol{X})}{\sigma^2}$$

follows a χ_{n-1}^2-distribution. We can use this fact to make probabilistic claims about the sample variance of normally distributed data. For instance, if we draw a random sample of size $n = 10$ from a $\mathcal{N}(3, 12)$-distribution, and we want to know the probability that the sample variance will be no greater than 14, then we proceed as follows:

$$\begin{aligned}
\mathbb{P}\left(s^2(\boldsymbol{X}) \leq 14\right) &= \mathbb{P}\left(\frac{10-1}{12}s^2(\boldsymbol{X}) \leq \frac{(10-1)14}{12}\right) \\
&= \mathbb{P}\left(\frac{n-1}{\sigma^2}s(\boldsymbol{X})^2 \leq 10.5\right) \\
&= \mathbb{P}(H^2 \leq 10.5),
\end{aligned}$$

where $H^2 \sim \chi_9^2$. We use R to compute the exact value:

```
pchisq(10.5, df = 9)

[1] 0.6884577
```

That is, about 69%. ♢

For the same reason why the sample variance is used instead of the naive plug-in estimator when estimating the variance of a distribution, the sample standard deviation is

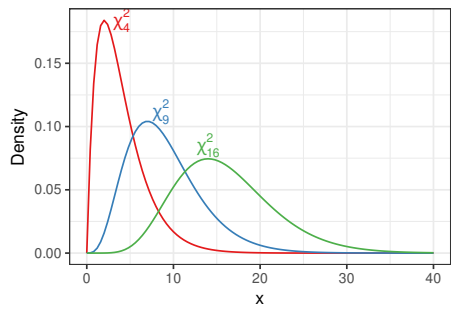


Figure 5.1: Probability density functions of the χ^2-distributions with 4, 9, and 16 degrees of freedom.

used instead of the naive plug-in estimator when estimating the standard deviation of a distribution:

$$s(\boldsymbol{X}) = \sqrt{s^2(\boldsymbol{X})} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2}.$$

The R function `sd()` implements this formula. The sample standard deviation is a consistent estimator of the distribution's standard deviation. However, it is *not* an unbiased estimator. Even though the sample variance estimates the distribution variance in an unbiased way, the sample standard deviation still underestimates the distribution standard deviation slightly. Correcting for this bias is difficult at best and usually impossible, but it is of little practical importance.[2]

***Exercise 5.16.** Let $\boldsymbol{X} = (X_1, \ldots, X_{10})$ be a simple random sample from a $\mathcal{N}(-4, 8)$-distribution. Compute the probability that $s(\boldsymbol{X})$ is at least 3.5. To this end, you can proceed as in Remark 5.15. ♢

5.4 Non-random samples

Random samples have two major advantages: they yield unbiased estimates of the population mean and variance, and they allow us to apply mathematical results such as the central limit theorem.

In practice, however, drawing a genuine random sample from any reasonably interesting population is extremely difficult. Take, for example, the task of assessing the English proficiency of adults of Kosovar origin in the canton of St. Gallen. To obtain a random sample, we would first need a complete list of all such adults in the canton. From that list, we would randomly select a group and then persuade all of them to take part in the study. But as soon as someone refuses, the sample is no longer random with respect to the original population. What we end up with is a sample of those who are

[2]Just because $g(X_1, \ldots, X_n)$ estimates the estimand θ in an unbiased way, doesn't mean that $h(g(X_1, \ldots, X_n))$ estimates the estimand $h(\theta)$ in an unbiased way. For instance, $\overline{X}$ is an unbiased estimator of μ, but $(\overline{X})^2$ overestimates μ^2. To convince yourself of this, consider a distribution with $\mathbb{P}(X = -1) = \mathbb{P}(X = 1)$.

willing to participate, not of all adults of Kosovar origin in St. Gallen. Our estimates therefore apply primarily to this new population, not to the one we initially set out to study. Furthermore, the sample would not consist of independent observations, since each individual can only be selected once.

This example highlights an important consequence: while a random sample provides an unbiased estimate of the mean of the true population of interest, nonresponse can distort results. In our case, refusals may come disproportionately from people who consider their English weak or who find linguistic research irrelevant. Those who remain are likely to have stronger English skills or a greater interest in languages. As a result, the sample mean will tend to overestimate the average ability in the target population.

In short, social science researchers rarely work with true random samples. Instead, they rely on non-random samples, which means that far more care is needed when interpreting results and generalising them to a broader population.

- Social media surveys tend to reach people who share similar views. Even the most reputable polling institutes face the same problem: In the U.S., for instance, fewer than 10% of people contacted in telephone surveys actually respond, according to an article published by the Pew Research Center.
- A person who completes a long questionnaire on their multilingual behaviour without compensation is likely to value multilingualism more than someone who closes the browser after three questions—or never receives the questionnaire at all, when snowball sampling is used.
- Findings from a sample of highly educated, relatively affluent individuals (such as most university students) may not generalise to populations with lower levels of education or socioeconomic status. This is especially problematic when these factors are directly relevant to the research question. If we are willing to assume that they have only a minimal effect, generalisation may be defensible—but whether that assumption is justified is a matter of subject-matter knowledge, not statistics.

So should you despair and toss these lecture notes into the camp fire halfway through the semester?

I'd wait till the end of the semester, if I were you.

On the one hand, the ideal case of the random sample provides a useful model for thinking about data. When the data-generating process in a particular study does not fit that ideal, the model can often be refined to better capture the reality at hand. On the other hand, many of the insights still transfer well to common research settings—for instance, experiments in which participants are not a random sample of any population but in which they *are* randomly assigned to conditions within the study.

Chapter 6

Estimating estimation uncertainty

An unavoidable fact when working with samples is that we can only *estimate* properties of distributions. The question arises as to how accurate these estimates actually are. Unfortunately, in most cases we do not know this precisely either, which is why this inaccuracy *also* has to be estimated on the basis of the sample.

The aim of this chapter is to illustrate, by means of an example, how one can assess the inaccuracy of a parameter estimate using a sample. To this end, this chapter introduces the **bootstrap**—a flexible, mechanistic procedure for estimating such inaccuracy. Then, it shows how the central limit theorem (cf. Section 3.5) can be used for the same purpose.

In what follows, we work with a dataset from the study by DeKeyser et al. (2010). They investigated how the age at which migrants began to learn a second language (*age of acquisition*, AOA) is related to their performance on a grammar task (*grammaticality judgement task*, GJT). The participants were Russian migrants in Israel and in North America. The grammar task consisted of 204 true/false items. In the next chapters, we will consider the relationship between AOA and GJT; here, we use the dataset from DeKeyser et al. (2010) to show how one can quantify the inaccuracy of sample estimates.

Activity 6.1. The dataset `dekeyser2010.csv` contains the AOA and GJT data of the Russian immigrants in North America that participated in the study by DeKeyser et al. (2010). Read in this dataset into R. Then reproduce the graph in Figure 6.1 and compute the sample mean of the GJT values. ♢

6.1 Sample means vary

Let's assume that the GJT data in the entire population were distributed exactly as in the dataset of DeKeyser et al. (2010, cf. Figure 6.1). This is merely an assumption for pedagogical purposes: as researchers we do not have access to the entire population, i.e., we actually do not know what this population distribution looks like. Instead, we must make do with samples. But let's temporarily assume that the data in the population were distributed exactly as in this study.

As discussed in Section 3.5, the means of random samples of the same size follow a certain distribution, the mean of which equals the mean of the parent distribution ($\mu_{\overline{X}} =$

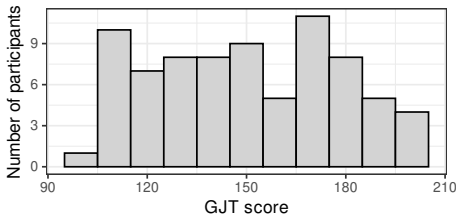


Figure 6.1: Histogram of the GJT data from the North America study by DeKeyser et al. (2010).

μ). Figure 6.2 shows, by way of example, five samples of size 20 from this GJT population, as well as the distribution of the means of 20,000 samples of 20 observations each from the population. GJT values that were already drawn could be drawn again, that is, sampling with replacement was used. The standard deviation of the sampling distribution of the mean is 6.07 points. Further, 2.5% of the sample means are smaller than 138.95, and 2.5% are greater than 162.60. Thus, 95% of the 20,000 sample means lie within an interval of $162.60 - 138.95 = 23.65$ points. The standard deviation of the sampling distribution or the width of such an interval would be sensible measures of the accuracy with which one can estimate a population parameter (here: the mean) from a sample: if the standard deviation is small, or if the interval is narrow, then the sample-based parameter estimates (here: the sample means) lie closer both to one another and to the population parameter.

The problem we're faced with is that we can only generate the sampling distribution if we have access to the entire population. If we only have a sample at our disposal, we need to estimate the standard deviation or the size of an interval expressing the spread of this sampling distribution.

6.2 The bootstrap

Enter the plug-in principle. Figure 6.2 illustrates that each individual sample imperfectly reflects the population. But in practice, this imperfect reflection may be the only thing we have at our disposal.[1] In order to estimate the standard deviation or gauge the form of the sampling distribution of the mean, we can apply the plug-in principle and treat the sample as a stand-in for the population.[2]

Example 6.2 (Red sample)**.** Figure 6.3 on page 134 illustrates the procedure. We've drawn the first (red) sample from Figure 6.2. Applying the plug-in principle, we pretend that the GJT population is distributed exactly as in this sample. To generate the sampling distribution of the mean under this assumption, we draw random samples of size 20 from the red sample using sampling with replacement. These new samples are called **bootstrap replicates**. Figure 6.3 shows three such bootstrap replicates. We compute the

[1] We could in principle make further assumptions about the data that cannot be inferred from the data themselves.

[2] This section was inspired by Hesterberg (2015).

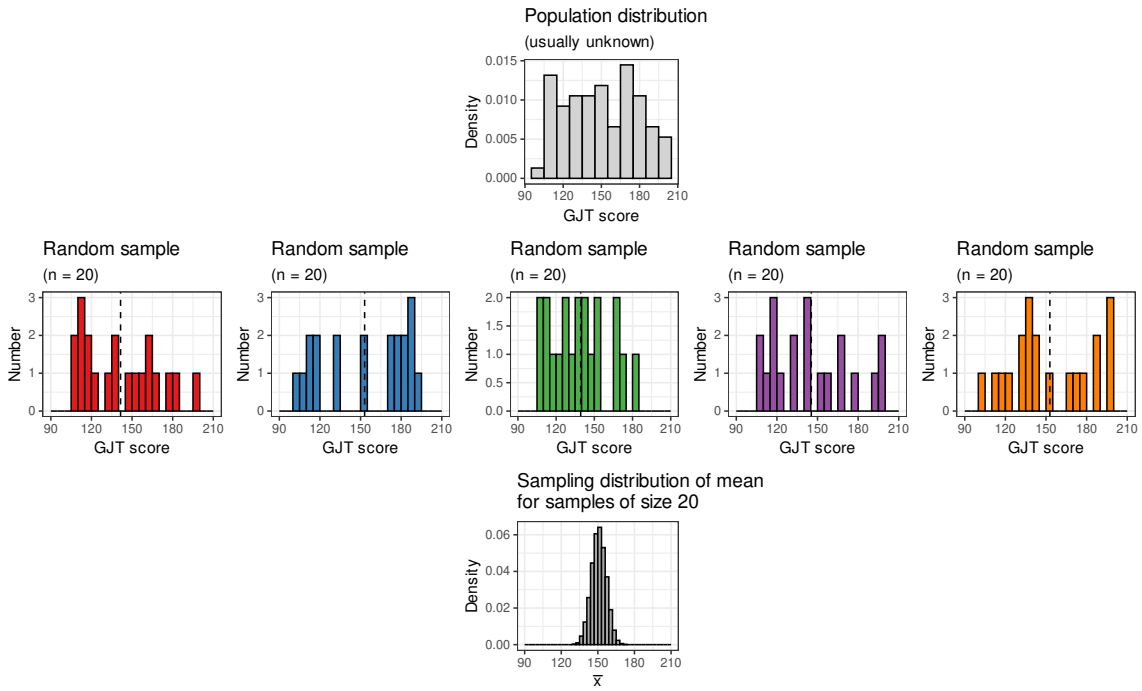


Figure 6.2: When we draw a large number of random samples of the size size from the population and compute their sample means (vertical line), we obtain the sampling distribution of the sample mean. In this case, the latter distribution looks approximately normal, but this doesn't have to be the case. For illustrative purposes, five random samples are shown as well.

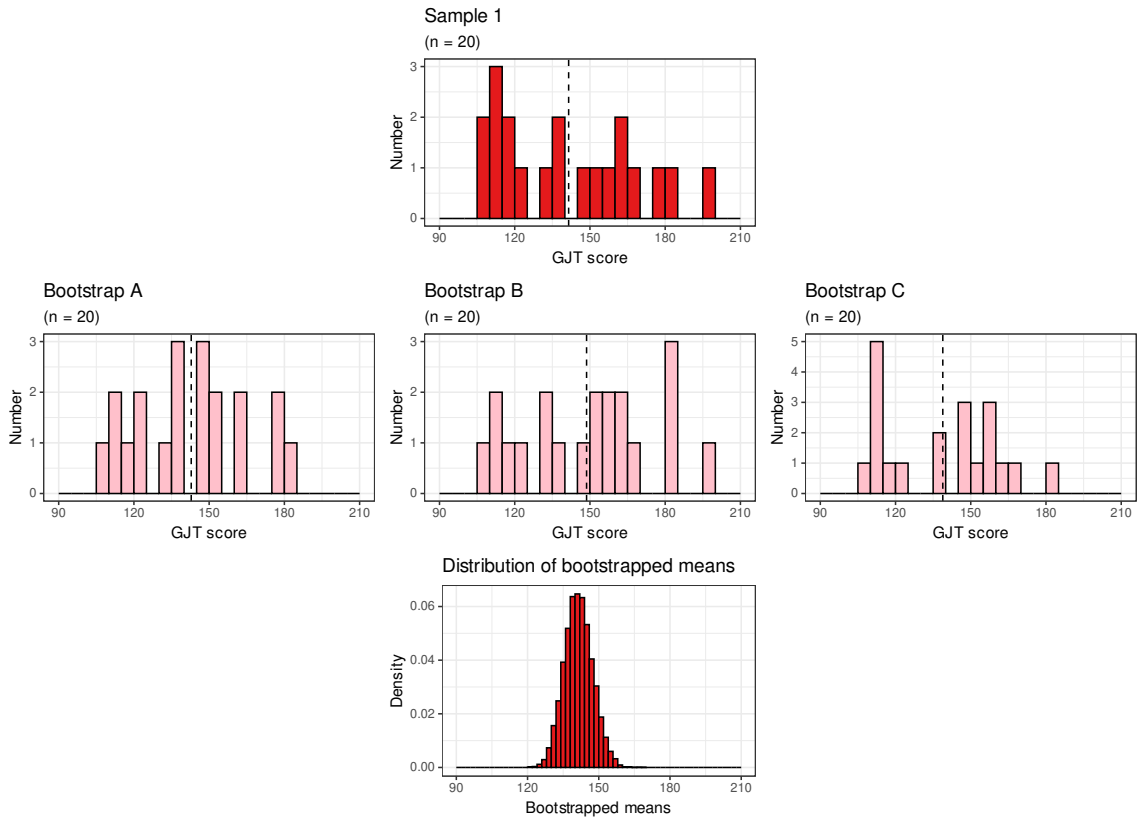


Figure 6.3: The first sample from Figure 6.2 serves as a stand-in for the GJT population. By way of example, three bootstrap replicates of size 20 are shown. The distribution of the means of 20,000 such bootstrap replicates are shown in the bottom plot. This distribution looks approximately normal, but this doesn't have to be the case.

sample mean of each bootstrap replicate; the distribution of 20,000 such means is shown in the bottom plot.

The expected value of the mean of the bootstrapped means equals the mean of the sample (141.5); any differences between the observed mean of the bootstrapped means and the sample means will be small. The standard deviation of the bootstrapped means is about 6.0. 2.5% of the bootstrapped means are smaller than 130.10, whereas 2.5% are larger than 153.55. The middle 95% of the bootstrapped means, then, is covered by an interval of 23.45 points. ♢

Example 6.3 (Blue sample)**.** Figure 6.4 on the facing page illustrates the same procedure but applied to the second (blue) sample from Figure 6.2. The mean of the bootstrapped means is very close to the mean of the sample (153). The standard deviation of the bootstrapped means is about 7.05. 2.5% of the bootstrapped means are smaller than 139.05, whereas 2.5% are larger than 166.60. The middle 95% of the bootstrapped means, then, is covered by an interval of 27.55 points. ♢

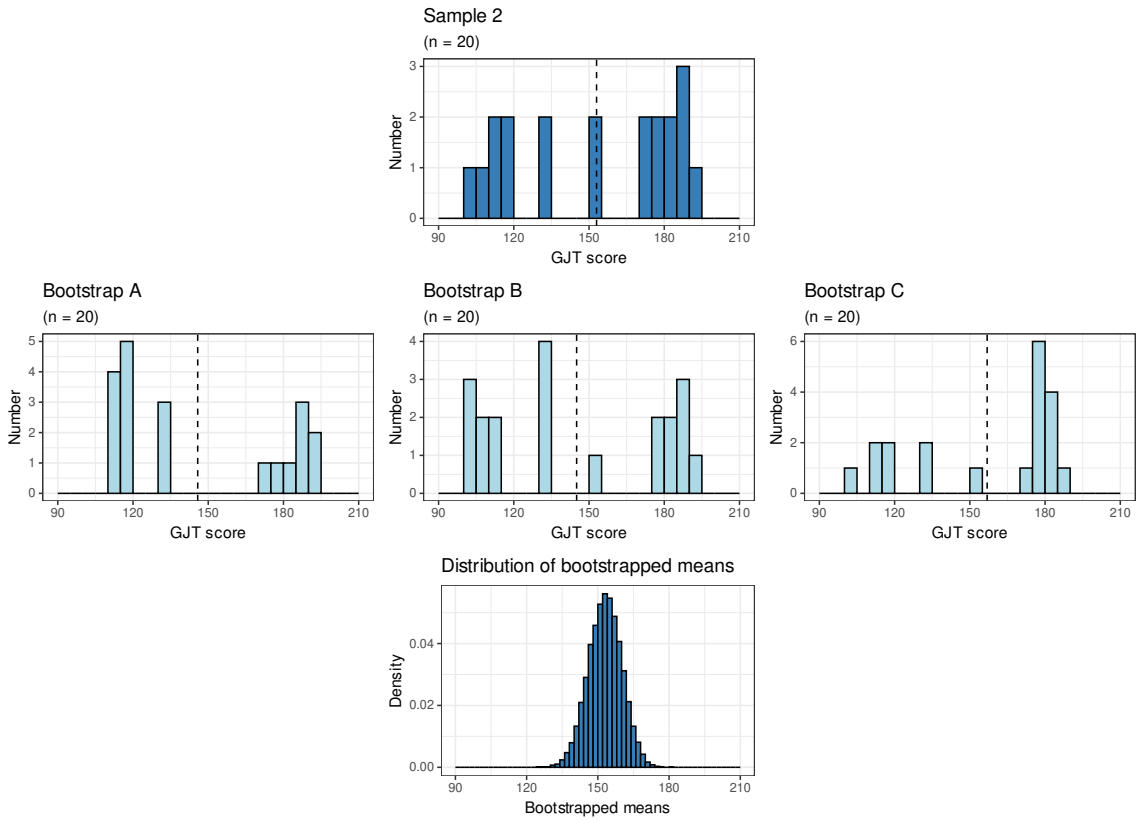


Figure 6.4: The second sample from Figure 6.2 serves as a stand-in for the GJT population. By way of example, three bootstrap replicates of size 20 are shown. The distribution of the means of 20,000 such bootstrap replicates are shown in the bottom plot. This distribution looks approximately normal, but this doesn't have to be the case.

The bootstrap is a technique for quantifying the uncertainty or imprecision of parameter estimates (Efron, 1979; Efron & Tibshirani, 1993). To calculate the actual uncertainty of a parameter estimate, one could draw a large number of samples from the same population and observe how the estimates vary between samples:

- Define the population,
  → draw samples,
  → generate the distribution of estimates across samples,
  → compute the variability of the estimates.

In the absence of a large number of samples from the same population, one relies on the plug-in principle: the single observed sample stands in for the population, and one examines how well samples of the same size drawn from this sample can estimate the parameter of interest:

- Define the sample,
  → draw bootstrap samples,
  → generate the distribution of estimates across bootstrap samples,
  → estimate the variability of the estimates.

So, crucially, the bootstrap provides an *estimate* of the uncertainty of a parameter estimate. This becomes clear when examining Figure 6.5 on the next page and Table 6.1. Figure 6.5 shows the distribution of the bootstrapped means for the five samples, while Table 6.1 summarises their standard deviations, their 2.5th and 97.5th percentiles, and the difference between these percentiles. The standard deviations and the widths of the intervals between the 2.5th and 97.5th percentiles do not exactly match the corresponding actual, but unknown, values in any of the five examples. But, on average, they are fairly similar.

Table 6.1: Standard deviation, percentiles, and the difference between the percentiles for the actual sampling distribution of the sample mean and for the five distributions of bootstrapped means. The percentiles and the differences between them aren't entirely consistent due to rounding.

Sampling distribution	SD	2.5th percentile	97.5th percentile	Difference
True (unknown)	6.1	139	163	24
Based on bootstrap sample 1	6.0	130	154	23
Based on bootstrap sample 2	7.0	139	167	28
Based on bootstrap sample 3	4.9	130	149	19
Based on bootstrap sample 4	6.5	133	158	25
Based on bootstrap sample 5	6.6	140	166	26

Hence, when no additional information is available (for example, prior studies or subject-matter reasoning), the bootstrap can provide useful, albeit imperfect, information about the uncertainty of a parameter estimate.

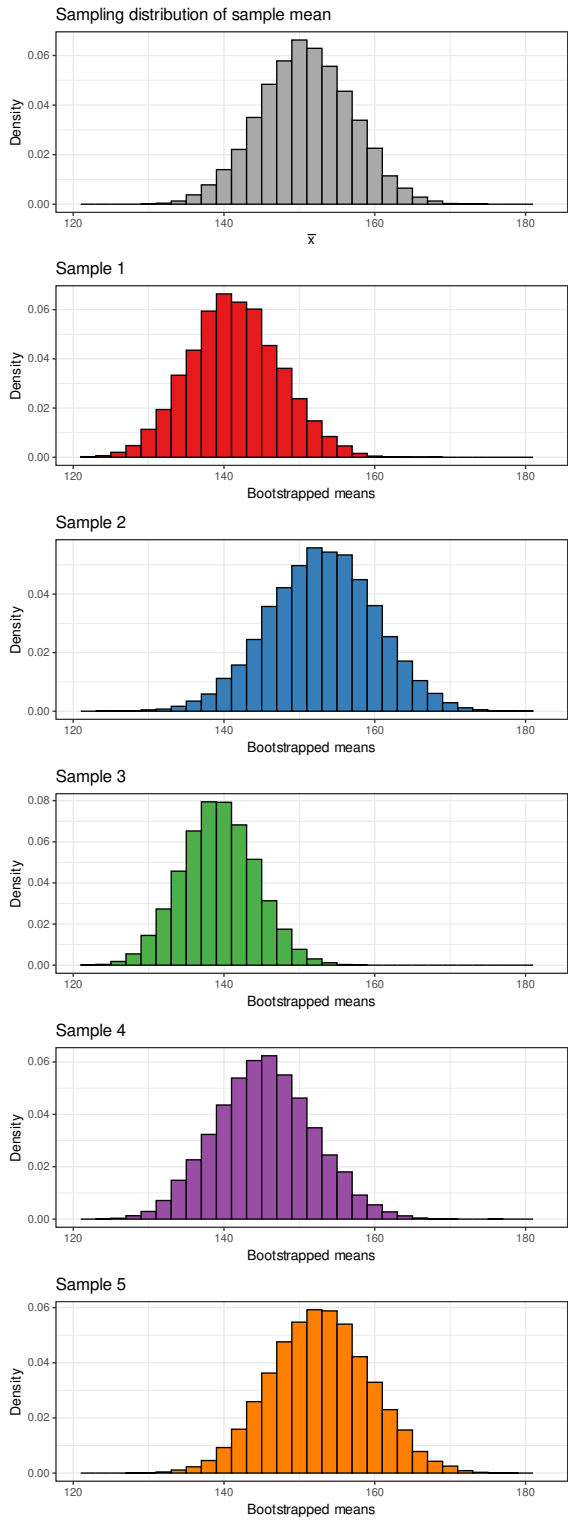


Figure 6.5: The distribution of bootstrapped means on the basis of five samples.

Remark 6.4 (Advantages of the bootstrap)**.** These are the main advantages of the bootstrap:

- The bootstrap is pedagogically valuable (I hope). The mathematical requirements are modest, which allows us to discuss important concepts independently of their usual formal treatment.

- The bootstrap is flexible. Here, we have focused on the uncertainty of a sample mean. This can also be expressed using a relatively simple analytical method (see below). However, the bootstrap can also be used to quantify the uncertainty of many other estimators, such as a trimmed or winsorised mean, a median, a standard deviation, a specific quantile, etc. Examples appear in the exercises. Moreover, the bootstrap can also be applied to more complex models (e.g., when examining relationships between multiple variables).

- In some cases, the bootstrap assumptions are more plausible than those of other common methods. This point will become clearer later in this chapter. Here, it is worth noting that in the examples above we never assumed that the sampling distribution of the mean was normally distributed. In these examples, the distributions of the bootstrapped means *happened* to be approximately normal—but we did not assume this a priori. In particular, we did not assume that the population from which the samples were drawn is normally distributed. ♢

Remark 6.5 (Disadvantages of the bootstrap)**.** Naturally, the bootstrap is not a silver bullet.

- "Bootstrapping does not overcome the weakness of small samples as a basis for inference." (Hesterberg, 2015, p. 379) On the one hand, the *actual* uncertainty of a parameter estimate is, of course, larger for a small sample than for a large one (see also the central limit theorem). On the other hand, our *estimate* of this uncertainty is also less accurate for smaller samples. This, however, is not so much a drawback of the bootstrap itself, but of small samples in general: other methods do not provide a better solution here.[3]

- The bootstrap implementation illustrated above tends to slightly underestimate, rather than overestimate, the uncertainty of a parameter estimate. This effect is stronger for small samples. The reason is that the variability of a sample typically underestimates the variability in the population; this is why the sample variance is calculated slightly differently from the population variance. In the bootstrap, however, the sample stands in for the population. To the extent that the sample underestimates population variability, the bootstrap underestimates the uncertainty of the parameter estimate. There are several ways to correct this bias (see Efron & Tibshirani, 1993), but these aren't so pedagogically useful.

- Because the bootstrap is so flexible, it is difficult to write a general, user-friendly function for it. In my view, it is best to program the bootstrap yourself. ♢

[3] Unless they make stricter assumptions or incorporate information not derivable from the data proper.

The examples above served purely a pedagogical purpose: if we have a sample of 76 participants, it is hardly sensible to draw smaller samples from it. Example 6.7 below shows how to estimate the uncertainty of DeKeyser et al.'s original sample mean. Incidentally, we find ourselves in the somewhat odd—but fairly common—situation that we do not really know precisely which population our statements can generalise to. Two further exercise illustrate the flexibility of the bootstrap.

Definition 6.6 (Standard error)**.** An estimate of the standard deviation of a sampling distribution is known as a **standard error**. ♢

Example 6.7. The function `btstrp_mean_one_run()` generates a single bootstrap replicate and calculates its mean. The function `replicate()` then executes it `n_bootstraps` times. It is assumed that your dataset is named `d`. If this is not the case, you need to replace `d` everywhere with the correct object name, or rename the dataset.

```
btstrp_mean_one_run <- function(x) {
  btstrp_sample <- sample(x, replace = TRUE)
  mean(btstrp_sample)
}
n_bootstraps <- 20000
bootstraps <- replicate(n_bootstraps, btstrp_mean_one_run(d$GJT))
```

Hesterberg (2015) recommends generating 20,000 bootstrap replicates so that the result is minimally affected by randomness in the bootstrap itself. To get a rough idea, 1,000 replicates would suffice, but in principle this computation should not take too long. A quick histogram (without `ggplot2`) illustrates the central limit theorem at work.

```
hist(bootstraps)
```

The standard deviation of the bootstrapped means serves as the standard error:

```
# standard error of mean
sd(bootstraps)

[1] 3.116992
```

I would report the estimate of the mean and its uncertainty as 150.8 ± 3.1 or even 151 ± 3. The exact values 150.7763 ± 3.1170 would include too many digits, giving a misleading sense of precision. In Vanhove (2021b), I provide some guidelines for rounding estimates.

Approximately 95% of the bootstrapped means lie between 145 and 157. This interval represents a **confidence interval**, which we will discuss in more detail later.

```
quantile(bootstraps, probs = c(0.025, 0.975))

    2.5%    97.5%
144.6184 156.8553
```

Since the distribution of the bootstrapped means appears roughly normal, we can also calculate these quantiles using the properties of the normal distribution. The 2.5th per-

centile of any normal distribution lies about 1.96 standard deviations below the mean:

```
qnorm(0.025)

[1] -1.959964
```

And the 97.5th percentile lies the same distance above the mean:

```
qnorm(0.975)

[1] 1.959964
```

The following calculation therefore yields essentially the same solution:

```
mean(d$GJT) + qnorm(c(0.025, 0.975)) * sd(bootstraps)

[1] 144.6671 156.8855
```

Of course, this only holds if the distribution of the bootstrapped means is approximately normal; the percentile method (that is, the approach using `quantile()` is more generally applicable. ♢

Exercise 6.8. The standard error and the interval width obtained in Example 6.7 are smaller than those reported in Table 6.1 on page 136. Why? ♢

Exercise 6.9 (Trimmed mean)**.** Definition 4.10 on page 112 introduced trimmed means. Here, we want to compute the 20% trimmed mean of the GJT data:

```
mean(d$GJT, trim = 0.2)

[1] 150.6739
```

Compute a standard error of the 20% trimmed mean of the GJT data using the bootstrap. To do so, you only need to change one line in the code from Example 6.7. Also plot a histogram of the bootstrap estimates. ♢

Exercise 6.10 (Uncertainty of the standard deviation)**.** The bootstrap is not only useful for quantifying the uncertainty in the estimation of a mean. Compute the standard deviation of the GJT data and use the bootstrap to assess the uncertainty of this estimate. Also draw the corresponding histogram. ♢

Exercise 6.11 (Median)**.** Compute the median of the GJT data and use the bootstrap to quantify the uncertainty of this estimate. What do you notice compared to the previous exercises? If nothing stands out at first, try increasing the number of *bins* in the histogram: `hist(bootstraps, breaks = 100)`. How do you explain your findings? ♢

6.3 The plug-in principle and the central limit theorem

In applied linguistics, the bootstrap isn't often used to quantify the uncertainty of a sample mean. Instead, researchers typically rely on the central limit theorem (see Section 3.5 on page 91). To recall: the central limit theorem states that the distribution of sample

means ($\overline{X}$) tends towards a normal distribution when sample sizes are sufficiently large. The mean of the sampling distribution equals the population mean ($\mu_{\overline{X}} = \mu$); its standard deviation is

$$\sigma_{\overline{X}} = \frac{\sigma}{\sqrt{n}}.$$

If the population standard deviation σ is known, we can compute $\sigma_{\overline{X}}$ directly. If the population standard deviation is unknown, we can once again apply the plug-in principle: the sample standard deviation s is the best available estimate of the population standard deviation σ, so we substitute it into the formula. This gives us an estimate of $\sigma_{\overline{X}}$, i.e., a standard error (SE):

$$\sigma_{\overline{X}} \approx \text{SE} = \frac{s}{\sqrt{n}}.$$

The sample standard deviation of the GJT scores is about 27.32. So the standard error is $\frac{27.32}{\sqrt{76}} = 3.13$.

The central limit theorem also allows us to construct a 95% confidence interval:

```
mean(d$GJT) + qnorm(c(0.025, 0.975)) * sd(d$GJT)/sqrt(76)

[1] 144.6347 156.9180
```

This confidence interval is very similar to the one obtained via bootstrapping, which is of course reassuring. The difference is that this time we have *assumed* that the means of samples of size 76 drawn from the population are normally distributed. This assumption was not required when using the bootstrap. For strongly skewed or otherwise wonky distributions, it is quite possible that the central limit theorem has not yet "kicked in" with samples of 76 observations. In such cases, the bootstrap is likely to be the more appropriate approach. That said, for such distributions, one should pause to consider whether the mean is really of interest; see *Before worrying about model assumptions, think about model relevance* (11 April 2019). It should also be noted that the central limit theorem applies only to the sample mean, not to other parameter estimates. For some parameters, alternative formulae exist, but the bootstrap is considerably more flexible.

6.4 The t distributions

The sample standard deviation (s) is merely an estimate of the population standard deviation (σ). If s underestimates σ, the uncertainty in the parameter estimate will in turn be underestimated; if s overestimates σ, the uncertainty will be overestimated. This would not be problematic if the under- and overestimations cancelled out on average. However, as noted on page 129, s tends to underestimate σ, particularly in small samples. Consequently, the sampling distribution of the mean is typically wider than a normal distribution with standard deviation $s/\sqrt{n}$. In most cases this bias in s cannot be corrected. An important exception is when the sample comes from a normally distributed population; this exception is dealt with in Student's theorem.

Theorem 6.12 (Gosset (Student))**.** Let $\boldsymbol{X} = (X_1, \ldots, X_n)$ be a random sample of independent observations from a $\mathcal{N}(\mu, \sigma^2)$ distribution. Let $\overline{X}$ and $s(\boldsymbol{X})$ denote the sample mean

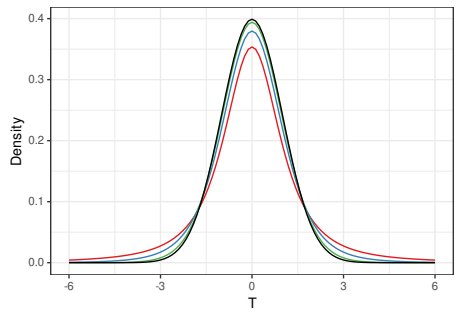


Figure 6.6: Densities of the Student's t distributions with two (red), five (blue), and twenty (green) degrees of freedom. The black curve shows the density function of the standard normal distribution.

and sample standard deviation, respectively. Then the quantity

$$T := \frac{\overline{X} - \mu}{s(\boldsymbol{X})/\sqrt{n}}$$

has the same distribution as

$$\frac{Z_0}{\sqrt{\frac{1}{n-1}\left(Z_1^2 + \cdots + Z_{n-1}^2\right)}},$$

where $Z_0, \ldots, Z_{n-1} \sim \mathcal{N}(0,1)$ are independent. This distribution is called the **Student's t distribution** with $n-1$ degrees of freedom (equal to the number of random variables appearing in the denominator). It is denoted by $T \sim t_{n-1}$ or $T \sim t(n-1)$. $\diamondsuit$

For comparison: if σ is known, then

$$\frac{\overline{X} - \mu}{\sigma/\sqrt{n}} \sim \mathcal{N}(0,1)$$

or, equivalently,

$$\overline{X} \sim \mathcal{N}(\mu, \sigma^2/n).$$

Figure 6.6 shows t distributions with two, five, and twenty degrees of freedom. As the number of degrees of freedom increases, the t distribution becomes increasingly similar to the standard normal distribution. This reflects the fact that larger samples tend to underestimate σ less than smaller samples. Incidentally, "Student" was the pseudonym of William S. Gosset, whose employer, Guinness, forbade him from publishing under his real name for fear of revealing ideas to competitors.

To construct a 95% confidence interval around a sample mean using the t distribution, one first uses `qt()` to find the 2.5th and 97.5th percentiles of the t distribution with $n-1$ degrees of freedom (here: $76 - 1 = 75$). These values are then multiplied by the standard error. This procedure is entirely analogous to constructing a confidence interval based on the central limit theorem; the only difference is that a t distribution is used instead of a normal distribution.

```
qt(0.025, df = 75)
[1] -1.992102
qt(0.975, df = 75)
[1] 1.992102
mean(d$GJT) + qt(c(0.025, 0.975), df = 75) * sd(d$GJT) / sqrt(76)
[1] 144.5340 157.0187
```

In this example, all calculation methods yield highly similar results. However, this is not necessarily the case for small samples or when the sample suggests that the population is highly skewed. Of the three methods discussed, the t-method makes the most assumptions: it not only assumes that one can make meaningful statements about the uncertainty from the sample and that the population has some distribution for which the central limit theorem applies at this sample size, but also that the population itself is normally distributed. When all these assumptions hold, this method is also the most accurate. The bootstrap, by contrast, is the Swiss Army knife of estimation methods: it can be applied in many situations, but depending on the circumstances, there may be specialised methods that perform better. [4]

6.5 Confidence intervals

Over the course of this chapter, we have constructed a few confidence intervals. It is now high time to explain what these actually are. Unfortunately, their definition is somewhat challenging.

Definition 6.13 (Confidence bounds and intervals)**.** Let $\boldsymbol{X} = (X_1, \ldots, X_n)$ be a sample from a population, and let θ be a parameter of interest (estimand). A mapping (function) $a(\alpha, \boldsymbol{X})$ provides a **lower** $100(1-\alpha)$**% confidence bound** if

$$\mathbb{P}(a(\alpha, \boldsymbol{X}) \leq \theta) \geq 1 - \alpha \tag{6.1}$$

for all $\alpha \in (0, 1)$. A mapping $b(\alpha, \boldsymbol{X})$ provides an **upper** $100(1-\alpha)$**% confidence bound** if

$$\mathbb{P}(\theta \leq b(\alpha, \boldsymbol{X})) \geq 1 - \alpha.$$

for all $\alpha \in (0, 1)$. Two mappings $\tilde{a}(\alpha, \boldsymbol{X})$ and $\tilde{b}(\alpha, \boldsymbol{X})$ provide a $100(1-\alpha)$**% confidence interval** if

$$\mathbb{P}(\tilde{a}(\alpha, \boldsymbol{X}) \leq \theta \leq \tilde{b}(\alpha, \boldsymbol{X})) \geq 1 - \alpha$$

for all $\alpha \in (0, 1)$. ♢

The idea is this. We want to estimate a parameter θ from a random sample. Rather than providing only a point estimate, we aim to give a range of possible values for θ that are compatible with the observed data. To do this, we need a procedure that guarantees

[4] In fact, the predecessor of the bootstrap was called the *jackknife*.

that the lower or upper confidence bound is greater or smaller than θ with probability at most α, where α is the allowable error probability. For a confidence interval, we need a procedure that constructs an interval containing the parameter θ with probability at least $1-\alpha$.

Example 6.14 (Exact confidence bounds for the mean of a normal distribution with known variance)**.** Let $X_1, \ldots, X_n$ be a simple random sample from a $\mathcal{N}(\mu, \sigma^2)$ distribution, where σ^2 is known. The sample mean of normally distributed variables is again normally distributed, so we know that

$$\overline{X} \sim \mathcal{N}(\mu, \sigma^2/n)$$

or, equivalently,

$$\frac{\overline{X}-\mu}{\sigma/\sqrt{n}} \sim \mathcal{N}(0,1).$$

Thus, with $q_{1-\alpha}$ the $(1-\alpha)$-quantile of the standard normal distribution $\mathcal{N}(0,1)$,

$$\begin{aligned}
1-\alpha &= \mathbb{P}\left(\frac{\overline{X}-\mu}{\sigma/\sqrt{n}} \leq q_{1-\alpha}\right) && \text{[quantile of continuous distribution]} \\
&= \mathbb{P}\left(\overline{X} - q_{1-\alpha}\frac{\sigma}{\sqrt{n}} \leq \mu\right) && \text{[rearranging]} \\
&= \mathbb{P}\left(\overline{X} + q_{\alpha}\frac{\sigma}{\sqrt{n}} \leq \mu\right),
\end{aligned}$$

since $-q_{1-\alpha} = q_\alpha$. Accordingly, we define

$$a(\alpha, X_1, \ldots, X_n) := \overline{X} + q_\alpha \frac{\sigma}{\sqrt{n}},$$

where q_α is the α-quantile of the standard normal distribution. Then $a(\alpha, X_1, \ldots, X_n)$ is a lower $100(1-\alpha)\%$ confidence bound for μ. Let's illustrate this with a simulation, here with $n = 5$, $\mu = -34$, $\sigma^2 = 9^2$, and $\alpha = 0.13$:

```
known_sigma_one_run <- function(n, mu, sigma, alpha) {
  # draw sample
  x <- rnorm(n, mu, sigma)
  # lower confidence bound
  mean(x) + qnorm(alpha) * sigma / sqrt(n)
}
lwr <- replicate(
  20000, known_sigma_one_run(n = 5, mu = -34, sigma = 9, alpha = 0.13))
mean(lwr <= -34)
```

```
[1] 0.86915
```

87% of the lower confidence bounds, then, lie below the true parameter value of μ. Indeed, the inequality in (6.1) holds with equality in this special setting.

Let's generate a concrete sample from $\mathcal{N}(12, 17^2)$ and use it to construct an 80% lower confidence bound for μ:

```
mu <- 12
sigma <- 17
n <- 7
alpha <- 0.2
x <- rnorm(n, mu, sigma)
x # sample
```

```
[1] 28.693480 19.968156 10.164487  8.381071 31.687674
[6] 33.970032 21.089001
```

```
mean(x) + qnorm(alpha) * sigma / sqrt(n)
```

```
[1] 16.58566
```

A priori, the probability that a random sample produces a 'correct' lower $100(1-\alpha)\%$ confidence bound (that is, a bound that is below θ) is at least $1-\alpha$. However, it can of course happen, as in this case, that the lower confidence bound is greater than θ. This example also shows that the probability guarantees refer to the construction procedure, not to individual bounds: it would be nonsensical to say that

$$\mathbb{P}(16.58 \leq \mu) \geq 0.80,$$

because the expression '$16.58 \leq \mu$' does not involve any randomness: it is simply false.[5]

An upper $100(1-\alpha)\%$ confidence bound is given by

$$b(\alpha, X_1, \ldots, X_n) := \overline{X} + q_{1-\alpha}\frac{\sigma}{\sqrt{n}}.$$

So a $100(1-\alpha)\%$ confidence interval is given by

$$[a(\alpha/2, X_1, \ldots, X_n), b(\alpha/2, X_1, \ldots, X_n)].$$

That is, we use two $100(1-\alpha/2)\%$ confidence bounds.

Other construction methods are possible; these yield different specific values, but provide the same probability guarantees. The method described here is the most common. ♢

Remark 6.15 (Quality criteria for confidence intervals)**.** For any given estimation problem, there is not always a unique procedure to construct confidence intervals. The most important criterion when choosing a method is that the probability guarantees (as in 6.1) are respected. If these guarantees are satisfied exactly, the intervals are called **exact** confidence bounds. In practice, however, they are often only approximately satisfied. If its assumptions are satisfied, then the method from Example 6.14 is exact, whereas the bootstrap intervals discussed earlier are approximate.

If two procedures satisfy this criterion equally well, one generally prefers the method that tends to produce narrower intervals, or for which the interval width converges to zero

[5] Another example: before rolling a six-sided die, the probability of rolling at least a 4 is 50%. But if we observe a 2, it no longer makes sense to claim that the probability of 2 being at least as large as 4 is 50%.

as the sample size increases to infinity. Indeed, it is possible to devise procedures that satisfy the necessary probability guarantees but either produce entirely trivial confidence bounds or fail to shrink as the sample size increases to infinity. We will not consider such procedures here.

Another important quality criterion for confidence bounds and intervals is their **robustness**: if a procedure satisfies the probability guarantees exactly only under rather specific assumptions about the data, it is desirable in practice that the method still approximately maintains the guarantees under less restrictive conditions. See Exercise 6.21. ♢

Remark 6.16 (Choice of bounds)**.** A lower confidence bound is appropriate when one is interested in whether θ is at least a certain value (with a specified probability of error). An upper confidence bound is appropriate when one is interested in whether θ is at most a certain value. A confidence interval is appropriate when one wishes to answer both questions simultaneously. For the questions that concern us, confidence intervals are typically suitable.

By default, one typically sets $\alpha = 0.05$, i.e., one calculates 95% confidence bounds or 95% confidence intervals. This, however, is merely a convention. ♢

The concepts of confidence bounds and confidence intervals are more subtle than they might appear—even for experienced researchers (Hoekstra et al., 2014). A 95% confidence interval is often interpreted as the range within which the population parameter (here: μ) lies with 95% probability. However, this interpretation is incorrect, as Example 6.14 demonstrates (see also Morey et al., 2016). Nonetheless, Ehrenberg (1982) offers the following perspective on interpreting confidence intervals:

> "[This] rough-and-ready interpretation of confidence limits ... will be close to the truth. The choice is between making a statement which is true but so complex that it is almost unactionable, and making one which is much simpler but not quite correct. Fortunately, the effective content of the two kinds of statement is generally similar." (p. 125)

Instead of confidence intervals, Morey et al. (2016) recommend the use of **credibility intervals**. These are grounded in Bayesian statistics and are rare in our research literature, which is why they are not discussed here. Albers et al. (2018) note that confidence and credibility intervals are usually very similar; however, Nalborczyk et al. (2019) question this conclusion.

Despite this, I consider the following points particularly important:

- Confidence intervals emphasise that estimates are inherently uncertain.
- With large samples, or with populations that exhibit little variation, confidence intervals tend to be narrower.
- By chance alone, a sample may underestimate population variation, and thus the confidence interval may also underestimate the uncertainty of the estimate.

- More accurate assessments of uncertainty can be obtained with larger samples, more sophisticated study designs, or by making additional reasonable assumptions about the data.

In the remainder of this chapter, we will examine the construction of confidence intervals for more realistic scenarios.

Example 6.17 (Exact confidence bounds for the mean of a normal distribution with unknown variance)**.** If we have a random sample of independent observations $X_1, \ldots, X_n \sim \mathcal{N}(\mu, \sigma^2)$, where both μ and σ^2 are unknown, the method from Example 6.14 can no longer be used. However, using Theorem 6.12, we can exploit the fact that

$$\frac{\overline{X} - \mu}{S/\sqrt{n}} \sim t(n-1).$$

A derivation analogous to the one in Example 6.14 yields a lower confidence bound of

$$a(\alpha, X_1, \ldots, X_n) := \overline{X} + q_{n-1;\alpha} \frac{S}{\sqrt{n}},$$

where $q_{n-1;\alpha}$ is the α-quantile of the $t(n-1)$ distribution. Upper confidence bounds and confidence intervals can be constructed analogously.

Using the function `t.test()`, you can quickly compute these confidence bounds:

```
# 80% confidence interval for the GJT mean
t.test(d$GJT, conf.level = 0.80)$conf.int

[1] 146.7248 154.8278
attr(,"conf.level")
[1] 0.8

# lower 95% confidence bound for the GJT mean
t.test(d$GJT, conf.level = 0.95, alternative = "greater")$conf.int

[1] 145.5576      Inf
attr(,"conf.level")
[1] 0.95
```

Use `alternative = "less"` for upper confidence bounds. ♢

Remark 6.18. Identical random experiments can yield quite different confidence intervals. With the code below, you can demonstrate this yourself. The same random experiment is carried out `n_sim` times: each time, `n_obs` independent observations are generated from a $\mathcal{N}(\texttt{popmean}, \texttt{stdev}^2)$ distribution. On the basis of this sample, a 95% confidence interval is constructed using `t.test()`. The confidence intervals, sorted by their width, are displayed in Figure 6.7. About 5% of the intervals shown don't contain the value of `popmean`, and the width of the intervals varies appreciably between samples.

```
n_sim  <- 100
n_obs  <- 23
```

```
stdev  <- 2.5
popmean <- -7

cis <- matrix(nrow = n_sim, ncol = 2)

for (i in 1:n_sim) {
  x <- rnorm(n_obs, popmean, stdev)
  cis[i, ] <- t.test(x)$conf.int
}

in_interval <- cis[, 1] <= popmean & popmean <= cis[, 2]

results <- tibble(min = cis[, 1],
                  max = cis[, 2],
                  in_interval,
                  sim = 1:n_sim) |>
  mutate(width = max - min)

ggplot(results,
       aes(xmin = min, xmax = max,
           y = reorder(sim, width),
           colour = in_interval)) +
  geom_errorbarh() +
  geom_vline(xintercept = popmean,
             linetype = "dashed") +
  scale_y_discrete(breaks = NULL) +
  scale_colour_manual("True mean in interval?",
                      limits = c(FALSE, TRUE),
                      labels = c("no", "yes"),
                      values = c("red", "black")) +
  theme(legend.position = "bottom") +
  labs(y = element_blank(),
       title = "100 95% confidence intervals for the same mean")
```

◇

Exercise 6.19. Assume you have two random samples with the same sample standard deviation: $s_1 = s_2$. Sample 1 consists of 16 observations; sample 2 of only four. Based on both samples, 95% confidence intervals for the mean are computed using t-distributions. What are the *two* reasons that the 95% confidence inteval based on sample 1 will be narrower than the one based on sample 2. ◇

Remark 6.20 (Approximate confidence bounds for the mean)**.** If the random sample was not drawn from a normal distribution, the confidence bounds and confidence intervals from Example 6.17 can still be used as approximations. Another option is to use bootstrap-based intervals. ◇

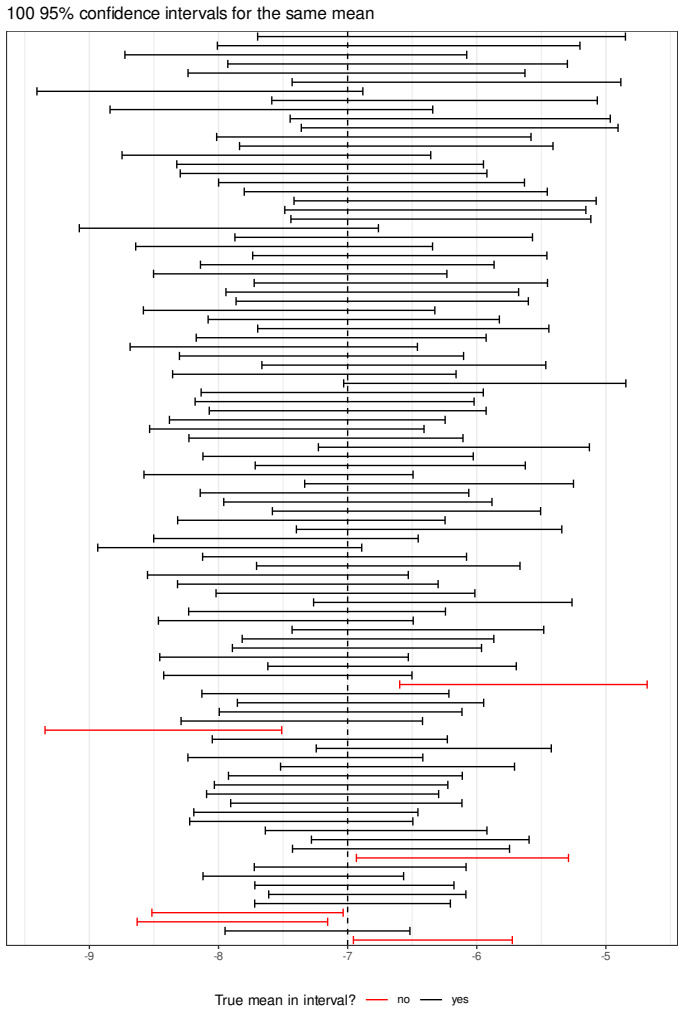


Figure 6.7: Confidence intervals based on 100 samples of 23 observations from a $\mathcal{N}(-7, 2.5^2)$-distribution. The confidence intervals are sorted by their width.

Exercise 6.21 (Robustness of the t method)**.** We wish to investigate if the confidence interval obtained using the method discussed in Example 6.17 is useful if the data stem from a uniform distribution instead of a normal distribution. To this end, we first define a function that generates a random sample from a uniform distribution, constructs a t-based confidence interval for the mean, and outputs if the mean of the uniform distribution is contained in this interval:

```
uniform_one_run <- function(n, min, max, conf_level) {
  x <- runif(n, min, max)
  ci <- t.test(x, conf.level = conf_level)$conf.int
  popmean <- (min + max)/2
  ci[1] <= popmean & popmean <= ci[2]
}
```

We run this function 20,000 times with some parameter settings. Then, we calculate how often the confidence interval contained the true population mean.:

```
simulation <- replicate(20000, uniform_one_run(9, -12, 2, 0.7))
mean(simulation)

[1] 0.70735
```

In this case, about 70.7% of the 70% confidence intervals contain the true mean.

1. Run this simulation again with smaller and larger sample sizes. Also vary the value of `conf_level`. What do you conclude?

2. Uniform distributions, like normal distributions, are symmetric around their mean. χ^2 distributions (see Remark 5.15), by contrast, are right-skewed: There is more probability mass below the mean than above it, especially if the number of degrees of freedom is low. We want to check if the t method still works for such right-skewed distributions. To this end, adapt the code above so that it generates samples from a χ^2 distribution with `df` degrees of freedom (`rchisq(n, df)`); you should also adapt the code elsewhere so that it makes sense. (The mean of a χ^2 distribution equals its number of degrees of freedom.) Play around with the values for `n`, `df`, and `conf_level`, and draw a conclusion. ♢

***Example 6.22** (Exact confidence bounds for a binomial parameter)**.** We conclude this chapter with a classic example. Let $X \sim \text{Binomial}(n, p)$. An unbiased estimator of p is

$$\widehat{p} := \frac{X}{n}.$$

We now wish to construct exact lower and upper confidence bounds for p. To this end, we consider the distribution function $F_{n,p}$ of the Binomial(n, p) distribution. A useful result, which we won't prove, states that for all $\alpha \in (0, 1)$,

$$\mathbb{P}(F_{n,p}(X) \leq \alpha) = 1 - \mathbb{P}(F_{n,p}(X) > \alpha) \leq \alpha.$$

(This holds not only for binomial distributions but for all random variables X with distribution function F.) Consequently,

$$\mathbb{P}(F_{n,p}(X) > \alpha) \geq 1 - \alpha. \tag{$*$}$$

We now regard the expression $F_{n,p}(r)$ not as a function of r but rather as a function of p. For r we substitute the observed number of successes X. The function thus obtained is strictly decreasing in p: for a fixed number of successes, the probability that a binomially distributed random variable generates at most r successes is greater for small p than for large p. As the upper $100(1-\alpha)\%$ confidence bound, we now choose the smallest possible b such that $F_{n,b}(X) > \alpha$. This number b represents a valid upper $100(1-\alpha)\%$ confidence bound, for

$$\begin{aligned}
\mathbb{P}(p \leq b) &= \mathbb{P}(F_{n,p}(X) \geq F_{n,b}(X)) && \text{[strictly decreasing]} \\
&\geq \mathbb{P}(F_{n,p}(X) > \alpha) && \text{[choice of } b\text{]} \\
&\geq 1 - \alpha. && [(*)]
\end{aligned}$$

Let us take a concrete examples with 23 trials and seven successes. Then $\widehat{p} = 7/23 \approx 0.304$. The red curve in Figure 6.8 shows how $F_{23,p}(7)$ varies with p. The upper 90% confidence bound for p (i.e., with $\alpha = 0.1$) is about 0.46.

For the lower confidence bound, we consider successes as failures and vice versa. That is, instead of X, we consider the random variable $Y := n - X$. This random variable has a Binomial$(n, 1-p)$ distribution. Using the same method as before, we compute an upper confidence bound $\tilde{b}$ for $1-p$. The lower confidence bound for p is then simply $a := 1 - \tilde{b}$. The blue curve in Figure 6.8 shows how $F_{23,p}(23-7)$ varies with p. The upper 90% confidence bound for $1-p$ is about 0.82. Accordingly, the lower 90% confidence bound for p is about 0.18. Hence, an 80% confidence interval for p is $[0.18, 0.46]$.

But it's considerably easier to use the `binom.test()` function:

```
binom.test(7, 23, conf.level = 0.80)$conf.int

[1] 0.1781578 0.4585561
attr(,"conf.level")
[1] 0.8
```

♢

***Remark 6.23** (Alternative confidence bounds for a binomial parameter)**.** The method for constructing confidence bounds for a binomial parameter introduced in Example 6.22 was first published by Clopper & Pearson (1934). It is exact in the sense of Definition 6.13. But it also conservative: The true coverage of Clopper–Pearson intervals is higher than $1-\alpha$, sometimes considerably so. Agresti (2002, Section 1.4) discusses a handful of alternative methods for constructing confidence bounds for a binomial parameter. Some of these are approximate, while others are exact but still tend to produce narrower confidence intervals than the Clopper–Pearson method. ♢

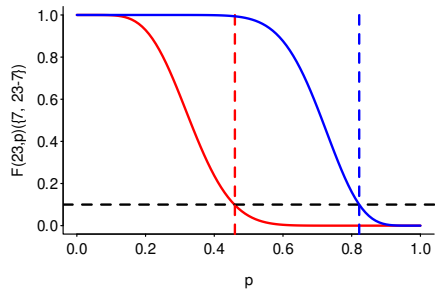


Figure 6.8: Constructing a 80% confidence interval for the binomial parameter p when $n = 23$ and $X = 7$. The red curve shows the quantity $F_{23,p}(7)$ as a function of p. The blue curve shows the quantity $F_{23,p}(23 - 7)$ as a function of p. The dashed lines show the upper 90% confidence bounds for p (red) and $1 - p$ (blue). The lower confidence bound for p can be derived from the upper confidence bound for $1 - p$.

***Remark 6.24** (Exact confidence bounds for quantiles)**.** There exists a general method for constructing exact confidence bounds for quantiles, including for the median. See Dümbgen (2016, Section 3.3). This method is implemented in the functions made available in `functions/quantile_ci.R` but not elaborated on here.

```
# Load functions
source(here("functions", "quantile_ci.R"))

# Use alternative = "less" for upper bound
# and alternative = "greater" for lower bound.
median_ci(d$GJT, conf_level = 0.95, alternative = "two.sided")

[1] 141 162

# For arbitrary quantiles, e.g., 0.75 quantile.
quantile_ci(d$GJT, gamma = 0.75, conf_level = 0.9, alternative = "two.sided")

[1] 168 182
```

◇

Part III

The general linear model

Chapter 7

Another look at the mean

This part of the lecture notes is devoted to the workhorse of quantitative data analysis: the **general linear model**.[1] The general linear model is a tool with which we can express how one or several **predictors** (or, indeed, none) are related to a single **outcome**. (It is common to speak instead of independent and dependent variables, but I find the terms predictor and outcome clearer and more intuitive.) Commonly used techniques such as t-tests, analysis of variance (ANOVA) as well as regression models are all instantiations of the general linear model or are otherwise closely connected to it. So a firm grasp of its underpinnings will stand you in good stead when learning how to analyse quantitative data. Moreover, if you want to master more advanced techniques such as generalised linear models (e.g., logistic regression) or mixed-effects modelling, you need to be comfortable with the general linear model first.

In this chapter, we introduce several key concepts of the general linear model by estimating the mean of a population in a different manner from what we have done previously. In the chapters that follow, the models become gradually more complex, but the fundamental principles from this chapter will still apply.

Let us, for the moment, set aside everything we have said about averages. We are given data (in this case, the GJT scores from DeKeyser et al. (2010), see Chapter 6) and we need to describe these data in a meaningful way. Listing all the datapoints would not be very informative. A better approach is to separate the data into two parts: a systematic component, capturing what the datapoints have in common, and an unsystematic component, capturing the individual discrepancies between these commonalities and the actual values:

$$\text{value of an observation} = \text{systematic component} + \text{discrepancy}.$$

To keep the notation concise, this equation is usually written as

$$y_i = \beta_0 + \varepsilon_i, \tag{7.1}$$

[1] Not to be confused with the **general*ised* linear model**. This is an extension of the general linear model—I didn't invent these names.... We'll touch briefly on one popular instantiation of the generalised linear model, namely logistic regression, in the optional Chapter 19.

for $i = 1, \ldots, n$. Here, y_i is the i-th observation in the dataset, β_0 represents what is common to all values in the population, and ε_i expresses how far the i-th observation deviates from this population value. The ε_i values are called the **errors**; we assume that $\mathbb{E}(\varepsilon_i) = 0$. We write β_0 rather than simply β, because later we'll use several βs to represent the commonality among the y values.

Typically, we are more interested in the βs than in the εs. Since we don't have access to the entire population, we must content ourselves with an estimate of β_0. In Equation 7.1, β_0 is a parameter with a specific, though usually unknown, value; for estimates of this parameter we use the notation $\widehat{\beta}_0$. Because $\widehat{\beta}_0$ is only an estimate, the errors are also only estimated:

$$y_i = \widehat{\beta}_0 + \widehat{\varepsilon}_i,$$

which is equivalent to

$$\widehat{\varepsilon}_i = y_i - \widehat{\beta}_0,$$

for $i = 1, \ldots, n$. Estimated errors are called **residuals**.

But how can we obtain $\widehat{\beta}_0$ (or, in other words, estimate β_0)? In principle, the equation holds for any value of $\widehat{\beta}_0$, since we can simply choose the $\widehat{\varepsilon}$ values to accommodate our choice of $\widehat{\beta}_0$. The first two GJT scores in the dataset are 151 and 182. If we were to select an arbitrary value for β_0, say 1823, we could set $\widehat{\varepsilon}_1 = -1672$ and $\widehat{\varepsilon}_2 = -1641$, and the equation would be satisfied:

$$y_1 = 151 = 1823 - 1672,$$

$$y_2 = 182 = 1823 - 1641.$$

Alternatively, if we were to choose $\beta_0 = 14$, we could set $\widehat{\varepsilon}_1 = 137$ and $\widehat{\varepsilon}_2 = 168$, and again the equation would hold:

$$y_1 = 151 = 14 + 137,$$

$$y_2 = 182 = 14 + 168.$$

So we need a principled method for estimating β_0.

7.1 Optimisation criteria

But what is the optimal way to determine $\widehat{\beta}_0$? The unsurprising answer is: it depends on what we mean by 'optimal'. One sensible definition of 'optimal' is to estimate β_0 in such a way that the sum of the absolute residuals ($\sum_{i=1}^{n} |\widehat{\varepsilon}_i|$) is as small as possible. If we choose 135 as the value of $\widehat{\beta}_0$, the sum of the absolute residuals is 1993. Here we assume that the dataset `dekeyser2010.csv` has the object name `d`.

```
sum(abs(d$GJT - 135))

[1] 1993
```

If instead we choose 148 as the value, the sum of the absolute residuals is 1799.

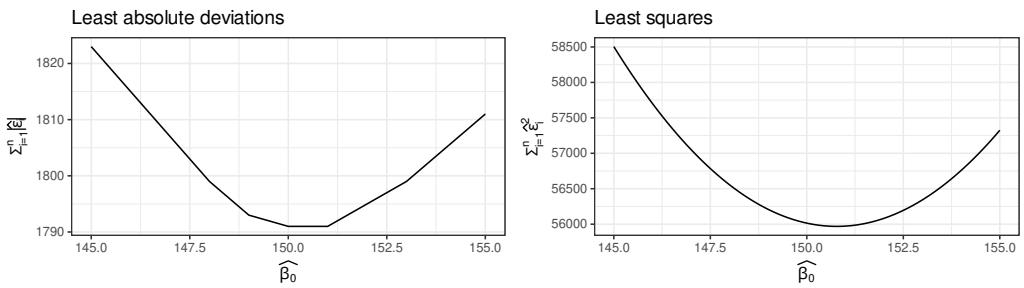


Figure 7.1: Left: If we estimate the parameter so as to minimise the sum of the absolute deviations, we obtain the sample median. Right: If we estimate the parameter so as to minimise the sum of the squared deviations, we obtain the sample mean.

```
sum(abs(d$GJT - 148))

[1] 1799
```

If we define 'optimal' in this way, then 148 is the better estimate of β_0. We could repeat this exercise for a wide range of candidate values and then select the one that minimises the sum. This approach is called the **method of least absolute deviations**. As Figure 7.1 (left) shows, β_0 estimates between 150 and 151 are optimal in this sense. Not coincidentally, all values in the interval $[150, 151]$ are medians of the GJT scores: when β_0 is estimated using the method of least absolute deviations, the result is a median of the sample. For a proof, see Schwertman et al. (1990).

Another sensible definition of 'optimal' is that β_0 needs to be estimated in such a way that the sum of the squared residuals, i.e.,

$$\sum_{i=1}^{n} \widehat{\varepsilon}_i^2 = \widehat{\varepsilon}_1^2 + \cdots + \widehat{\varepsilon}_n^2,$$

is as small as possible. This is the **method of least squares**. As shown in the right graph of Figure 7.1, the optimal $\widehat{\beta}_0$ value when using least squares is about 150.8 for the GJT data. The mean of the GJT data is about 150.8, too, and of course, this isn't a coincidence, either: when we estimate β_0 using least squares, we obtain the sample mean!

Theorem 7.1. The sample mean minimises the sum of squared residuals. ♢

This result can be proved using secondary school mathematics. In fact, here are two different proofs—one using derivatives and another using elementary algebra.

Proof using derivatives. Consider the expression

$$\sum_{i=1}^{n} (y_i - \widehat{\beta}_0)^2$$

as a function of $\widehat{\beta}_0$. This function is the sum of n quadratic functions, so it is itself a quadratic function. We're interested in the value of $\widehat{\beta}$ that minimises this function. As you may recall from secondary school, we can find this value by differentiating the function, setting the derivative to zero, and solving the resulting expression for $\widehat{\beta}_0$. So let's do this.

First compute the derivative of $\sum_{i=1}^n (y_i - \widehat{\beta}_0)^2$ with respect to $\widehat{\beta}_0$:

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}\widehat{\beta}_0} \sum_{i=1}^n (y_i - \widehat{\beta}_0)^2 &= \frac{\mathrm{d}}{\mathrm{d}\widehat{\beta}_0} \sum_{i=1}^n \left(y_i^2 - 2y_i\widehat{\beta}_0 + \widehat{\beta}_0^2 \right) \\
&= \sum_{i=1}^n \left(\frac{\mathrm{d}}{\mathrm{d}\widehat{\beta}_0} y_i^2 - \frac{\mathrm{d}}{\mathrm{d}\widehat{\beta}_0} 2y_i\widehat{\beta}_0 + \frac{\mathrm{d}}{\mathrm{d}\widehat{\beta}_0} \widehat{\beta}_0^2 \right) \\
&= \sum_{i=1}^n \left(0 - 2y_i + 2\widehat{\beta}_0 \right) \\
&= -\left(2\sum_{i=1}^n y_i \right) + 2n\widehat{\beta}_0.
\end{aligned}$$

These equalities rely on the identity $(a-b)^2 = a^2 - 2ab + b^2$ and on basic derivatives. The above is equal to zero if and only if $2\sum_{i=1}^n y_i = 2n\widehat{\beta}_0$. Dividing both sides by $2n$, we obtain

$$\widehat{\beta}_0 = \frac{1}{n} \sum_{i=1}^n y_i,$$

that is, the arithmetic mean of the y_i values. □

Proof using algebra. An alternative proof that doesn't rely on taking derivatives is this. Write $\overline{y} := \left(\sum_{i=1}^n y_i\right)/n$, i.e., the sample mean. Note that

$$\begin{aligned}
\sum_{i=1}^n (y_i - \widehat{\beta}_0)^2 &= \sum_{i=1}^n \left((y_i - \overline{y}) + (\overline{y} - \widehat{\beta}_0) \right)^2 \\
&= \sum_{i=1}^n \left((y_i - \overline{y})^2 + 2(y_i - \overline{y})(\overline{y} - \widehat{\beta}_0) + (\overline{y} - \widehat{\beta}_0)^2 \right) \\
&= \sum_{i=1}^n (y_i - \overline{y})^2 + 2(\overline{y} - \widehat{\beta}_0) \underbrace{\left(\sum_{i=1}^n y_i - n\overline{y} \right)}_{=n\overline{y} - n\overline{y} = 0} + \underbrace{n(\overline{y} - \widehat{\beta}_0)^2}_{\geq 0} \\
&\geq \sum_{i=1}^n (y_i - \overline{y})^2,
\end{aligned}$$

with equality if and only if $\widehat{\beta}_0 = \overline{y}$. □

Both computationally and mathematically, the method of least squares is the easiest to work with. Unless mentioned otherwise, the parameter estimates in the general linear model are obtained using this method ('**ordinary least squares**', or OLS). If you want to

use the method of least absolute deviations, median (or quantile) regression models are available. Further, in several applications in machine learning, other optimality criteria are quite common.

7.2 General linear models in R

We can use the `lm()` function to build linear models whose parameters are estimated using least squares. This function takes a formula in which the outcome is listed in front of the tilde and the predictors are listed after it. In the present case, we don't have any predictors, so we just write a 1 instead:

```
mod.lm <- lm(GJT ~ 1, data = d)
```

The estimated parameters can be printed by typing the name of the model at the prompt:

```
mod.lm


Call:
lm(formula = GJT ~ 1, data = d)

Coefficients:
(Intercept)
      150.8
```

A more compact overview can be obtained using `coef()`:

```
coef(mod.lm)

(Intercept)
   150.7763
```

Don't be confused by differences in the formatting (150.8 vs 150.7763). This is merely a matter of rounding the output; the underlying representation of these estimates is the same.

We can obtain the residuals, that is, the $\widehat{\varepsilon}_i$ values as follows:

```
resid(mod.lm) # output not shown
```

If we subtract the residuals from the observed values, we obtain the (so-called) **predicted values**. (A term I'm not entirely chuffed with.) Since

$$y_i - \widehat{\varepsilon}_i = y_i - (y_i - \widehat{\beta}_0) = \widehat{\beta}_0,$$

this results in 76 repetitions of $\widehat{\beta}_0$:

```
# output not shown:
d$GJT - resid(mod.lm)
# alternatively (output not shown):
```

```
predict(mod.lm)
```

7.3 Quantifying uncertainty in a general linear model

What we've done so far is *estimate* the β_0 parameter that we're interested in. While we now have a principled approach for obtaining this estimate, the resulting value will depend on the data that we've collected. Usually, the estimate is based only on a sample from some larger population. As a result, our estimate of β_0 won't exactly coincide with the true but unknown value of β_0, that is, there is some inherent uncertainty about our estimate.

7.3.1 The bootstrap

Just as in the previous chapter, we can use the bootstrap to quantify the uncertainty in the estimate of β_0. Since in this case $\widehat{\beta}_0$ is equal to the sample mean, the result is, of course, identical to what we obtained before. But this gives me the opportunity to show how the bootstrap can also be applied to more complex linear models.

The logic is again that we treat the sample as a stand-in for the population. However, instead of drawing bootstrap samples from the data itself, this time we draw bootstrap samples from the residuals ($\widehat{\varepsilon}$). We then combine these with $\widehat{\beta}_0$ to generate the bootstrap replicates. If we are only interested in the mean, this method offers no additional benefit, as it is mathematically equivalent to the earlier bootstrap approach. But it is pedagogically more useful (and sometimes statistically preferable), especially once we have several βs. Concretely:

1. Compute $\widehat{\beta}_0$ and obtain in addition a vector $\hat{\boldsymbol{\varepsilon}} = (\widehat{\varepsilon}_1, \ldots, \widehat{\varepsilon}_n)$.
2. Draw a bootstrap sample from $\hat{\boldsymbol{\varepsilon}}$ by means of sampling with replacement. Call this bootstrap sample $\hat{\boldsymbol{\varepsilon}}^*$. This vector also has n values, with some $\widehat{\varepsilon}_i$ possibly absent and others appearing more than once.
3. Combine $\widehat{\beta}_0$ and $\hat{\boldsymbol{\varepsilon}}^*$ to create a a new vector $\boldsymbol{y}^*$, with $y_i^* = \widehat{\beta}_0 + \widehat{\varepsilon}_i^*$ for $i = 1, \ldots, n$.
4. On the basis of $\boldsymbol{y}^*$, re-estimate the parameter of interest ($\widehat{\beta}_0^*$).
5. Repeat steps 2–4 several thousand times to obtain the distribution of the bootstrapped β_0 estimates.

The R code below implements these steps.

```
n_bootstrap <- 20000
bs_b0 <- vector(length = n_bootstrap)
residuals <- resid(mod.lm)
predictions <- predict(mod.lm)

for (i in 1:n_bootstrap) {
  bs_residual <- sample(residuals, replace = TRUE)
```

```
  bs_outcome <- predictions + bs_residual
  bs_mod <- lm(bs_outcome ~ 1)
  bs_b0[i] <- coef(bs_mod)[1]
}
```

We can then again visualise the distribution of the bootstrapped parameter estimates, and compute their standard deviation and the 2.5 and 97.5 percent quantiles. The results are, of course, identical to those in Chapter 6.

```
hist(bs_b0)
sd(bs_b0)
quantile(bs_b0, probs = c(0.025, 0.975))
```

7.3.2 Another kind of bootstrap

In the bootstrap approach just discussed, we assumed that the errors in the population are distributed in exactly the same way as the residuals in the sample. One drawback of this assumption is that it may underestimate the granularity of the residuals in the population. For example, in the `mod.lm` model there is a residual of -14.78 and one of -12.78, but none of -13.78. A residual of -14.78 corresponds to an observation of $150.78 - 14.78 = 136$; a residual of -13.78 would correspond to an observation of $150.78 - 13.78 = 137$. According to our assumption, then, there would be no participants in the population with a GJT score of 137.

This is, of course, a rather odd assumption; in practice, however, it usually has little effect on inference. An alternative is to assume instead that the errors are normally distributed. Normal distributions are infinitely fine-grained, so this assumption is, as it were, the opposite extreme of the earlier one. We assume that the expectation of the errors is 0, so we only need to determine the standard deviation of the normally distributed residuals in the population. This is estimated by the standard deviation of the residuals in the sample. Here we can use two functions:

```
sd(resid(mod.lm))

[1] 27.31769

sigma(mod.lm)

[1] 27.31769
```

In this example (a linear model without predictors) the two values are identical, but once predictors are included, `sigma()` provides a better estimate of the standard deviation of the residuals. It is computed as

$$\widehat{\sigma}_\varepsilon = \sqrt{\frac{1}{n-p}\sum_{i=1}^{n}\widehat{\varepsilon}_i^2}, \tag{7.2}$$

where p is the number of estimated βs. In this case $p = 1$, so the equation reduces to the sample standard deviation of the residuals:

```
sqrt(sum(resid(mod.lm)^2)/(length(resid(mod.lm)) - length(coef(mod.lm))))
```

```
[1] 27.31769
```

We divide by $n - p$ for the same reason that one divides by $n - 1$ when calculating the standard deviation of a sample: to counteract systematic underestimation.

Instead of generating bootstrap residuals using sampling with replacement, we now generate them randomly from a normal distribution with $\mu = 0$ and $\sigma = \widehat{\sigma}_\varepsilon$:

```
n_bootstrap <- 20000
bs_b0 <- vector(length = n_bootstrap)
sd_residuals <- sigma(mod.lm)
predictions <- predict(mod.lm)

for (i in 1:n_bootstrap) {
  bs_residual <- rnorm(n = length(predictions), mean = 0, sd = sd_residuals)
  bs_outcome <- predictions + bs_residual
  bs_mod <- lm(bs_outcome ~ 1)
  bs_b0[i] <- coef(bs_mod)[1]
}
```

```
hist(bs_b0) # histogram, not shown
sd(bs_b0)   # standard error
# 95% confidence interval
quantile(bs_b0, probs = c(0.025, 0.975))
```

This kind of bootstrap—in which we assume that the residuals follow a particular distribution and estimate the relevant parameters of that distribution from the sample—is called a **parametric bootstrap**. The bootstrap from the previous section—where bootstrap samples of model residuals are combined with model predictions and the relevant parameters are then re-estimated from these new values—is called a **semiparametric bootstrap**. The bootstrap from the previous chapter—where bootstrap samples are drawn from the original dataset—is called a **nonparametric bootstrap**.

Neither the assumption that the errors are distributed exactly as the residuals in sample nor the assumption that they are normally distributed and infinitely fine-grained can actually be correct here, since in this study only integers between 0 and 204 could occur. The fact that we nevertheless obtain the same results under different assumptions already suggests that with this sample size and this kind of data distribution, the estimation of the uncertainty of a mean is not substantially affected by assumptions about the exact distribution of the errors in the population.

7.3.3 Using t distributions

If we're prepared to assume that the residuals are normally distributed, then the standard error and a confidence interval can also be derived analytically. The formulas required

for this are not shown here, as they offer little additional pedagogical value. In R, one can use the `summary()` function to obtain the standard error (`Std. Error`):

```
summary(mod.lm)


Call:
lm(formula = GJT ~ 1, data = d)

Residuals:
    Min      1Q  Median      3Q     Max 
-46.776 -23.026  -0.276  23.224  47.224 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  150.776      3.134   48.12   <2e-16

Residual standard error: 27.32 on 75 degrees of freedom
```

Here, β_0 is labelled `(Intercept)`. What `t value` and `Pr(>|t|)` mean will only be discussed in a later chapter.

The 95% confidence interval can be calculated with `confint()`:

```
confint(mod.lm)

              2.5 %   97.5 %
(Intercept) 144.534 157.0187
```

Of course, other intervals may also be calculated, for example an 80% confidence interval:

```
confint(mod.lm, level = 0.80)

                10 %     90 %
(Intercept) 146.7248 154.8278
```

Exercise 7.2. Why does the `summary()` function show the median but not the mean of the residuals? ♢

7.4 *Maximum likelihood estimation

With the method of least squares (and its variants), parameter estimates are obtained by minimising the resulting estimated residuals. For certain more complex models (for example, the generalised linear model), this estimation method cannot be applied, and instead one uses the so-called **maximum likelihood method** (or a variant thereof). The principle of maximum likelihood estimation will now be illustrated using the GJT data. You may safely skip this section for now, until you encounter the maximum likelihood method later in your journey through statistics.

Suppose that the observations $y_1, \ldots, y_n$ are independent and normally distributed with mean μ and variance σ^2. The probability density $p(y_i)$ of such an observation is given by

$$p(y_i) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mu)^2}{2\sigma^2}\right).$$

Because the observations are independent, the joint probability density $p(y_1, \ldots, y_n)$ of the random vector $(y_1, \ldots, y_n)$ is equal to the product of the individual densities. (This was not shown in these lecture notes, but it follows from Definition 3.16 on page 71.) Hence,

$$p(y_1, \ldots, y_n) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mu)^2}{2\sigma^2}\right). \tag{7.3}$$

If we treat this expression as a function of μ, we call it the **likelihood function**. A sensible way to estimate μ on the basis of the data is to choose the value $\widehat{\mu}$ that maximises the likelihood function. The product makes this somewhat cumbersome, but we can take the logarithm and rewrite the product as a sum, since $\log ab = \log a + \log b$. This yields the **log-likelihood function**. As the logarithm is strictly increasing, the value that maximises the log-likelihood function also maximises the likelihood function. Accordingly, we determine $\widehat{\mu}$ as follows:

$$\begin{aligned}
\widehat{\mu} &= \arg\max_{\mu} \left(\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mu)^2}{2\sigma^2}\right)\right) && \text{[maximise likelihood function]} \\
&= \arg\max_{\mu} \left(\prod_{i=1}^{n} \exp\left(-\frac{(y_i - \mu)^2}{2\sigma^2}\right)\right) && \text{[constants are irrelevant]} \\
&= \arg\max_{\mu} \log\left(\prod_{i=1}^{n} \exp\left(-\frac{(y_i - \mu)^2}{2\sigma^2}\right)\right) && \text{[logarithm]} \\
&= \arg\max_{\mu} \sum_{i=1}^{n} \log\left(\exp\left(-\frac{(y_i - \mu)^2}{2\sigma^2}\right)\right) && \text{[rewrite product as sum]} \\
&= \arg\max_{\mu} -\sum_{i=1}^{n} \frac{(y_i - \mu)^2}{2\sigma^2} && [\log(\exp x) = x] \\
&= \arg\max_{\mu} -\sum_{i=1}^{n} (y_i - \mu)^2 && \text{[constants are irrelevant]} \\
&= \arg\min_{\mu} \sum_{i=1}^{n} (y_i - \mu)^2. && \text{[maximising } -x = \text{minimising } x]
\end{aligned}$$

Thus, we arrive at precisely the same optimisation problem as with the method of least squares! The solution is therefore also the same:

$$\widehat{\mu} = \frac{1}{n} \sum_{i=1}^{n} y_i.$$

More generally, in the classical linear model, we obtain the same β estimates, regardless of whether we use the least squares method (without making any assumptions about

the distribution of the data) or the maximum likelihood method (under the assumption of independent and identically normally distributed data). For certain more complex models, least squares cannot be applied, but maximum likelihood can.

7.5 Assumptions and relevance

Model assumptions enter into the data analysis at some point, be it when estimating the model parameters (using maximum likelihood) or when estimating the uncertainty about these estimates. In this chapter, three assumptions about the errors ε_i were made: The ε_i values are independent, they are drawn from the same distribution with expectation 0, and—when using t distributions or maximum likelihood estimation—this distribution is a normal one. The first two of these assumptions are jointly referred to as the **i.i.d. assumption**, where i.i.d. is the abbreviation of *independently and identically distributed*. The i.i.d. assumption is equivalent to assuming that $\varepsilon_1, \ldots, \varepsilon_n$ constitute a random sample from some distribution with expectation 0.

The independence assumption entails that if we know the value of one error, then this doesn't provide us with any more information about any of the remaining errors. (The formal definition of independence in Definition 3.43 on page 83 boils down to this.) To make this more concrete, consider the following example. Let's say you wanted to estimate the average length of the [u] vowel in the Berne vernacular. To this end, you have 25 informants from Berne read out 50 words containing [u]. It is conceivable that some informants tend to produce longer [u] sounds than others. That is, once you obtain a positive error (corresponding to longer than average vowel length) for a single production from a single speaker, chances are the other productions from this speaker will also have positive errors. That is, once you've learnt something about the value of one specific error, you can make more informed guesses about the values of some other error, too. (Recall that 'error' merely refers to the discrepancy between the observation y_i and the shared part $\mu = \beta_0$.) Similarly, it is conceivable that [u] sounds tend to be longer in some words or phonological contexts than others. If so, the error for a single word produced by a single speaker may give you some clues as to the error for other productions of the same word. In sum, the $25 \cdot 50 = 1250$ data points that this study would produce would violate the independence assumption if we were to analyse them with the tools we're going to discuss.

The independence assumption is essential for inference. If it is violated, the uncertainty about the parameter estimates may be massively underestimated using the methods covered in these lecture notes. Deciding whether this assumption is justified typically requires knowledge about how the data were collected. Problems with dependencies between errors can be tackled in a number of ways, with one popular approach since 2008 or so being the use of mixed-effects models.

The assumption that the $\varepsilon_1, \ldots, \varepsilon_n$ values have the same distribution isn't too relevant just yet, but we will come back to it in the next chapter. (That said, we already discussed one of its consequences in Example 5.11 on page 125.)

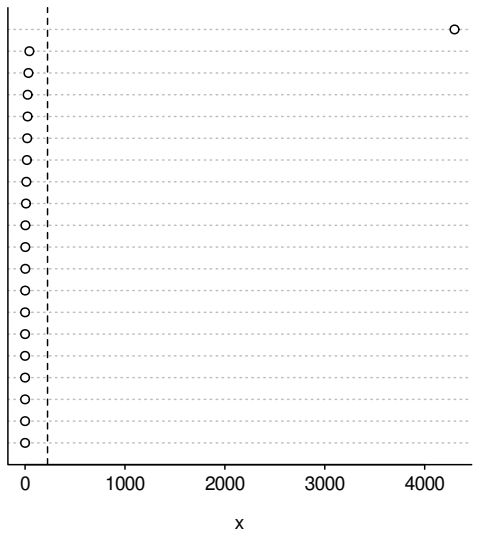


Figure 7.2: You *could* fit these data in a general linear model. But the mean (dashed vertical line) just isn't an informative measure of these data's central tendency. So why bother?

The normality assumption is the least important of the three. In fact, in our running example, we *know* that this assumption is violated: Normal distributions theoretically range from $-\infty$ to $+\infty$, but our data are known to be constrained to the interval $[0, 204]$. Moreover, normal data are infinitely fine-grained, whereas our data consists of integers. That said, we observed that we obtained pretty much the same uncertainty measures when assuming normality as when using the semiparametric bootstrap, which does not assume normality. This is quite common and can be attributed to the workings of the central limit theorem.

In my view, you shouldn't worry too much about the normality and equality of distributions assumptions. What you should ask yourself instead, however, is whether the model you want to fit and the statistics you want to compute are at all relevant in the context of your study and in the light of your data. Consider the data in Figure 7.2. It seems unlikely that the errors were drawn from a normal distribution. But rather than fit these data in a general linear model and then worry about the normality violation, you ought to ask yourself whether whatever research question you have can sensibly be answered in terms of the mean of these data. For this example, this seems unlikely, and a more pressing question presents itself: Where does the outlier come from? If, by contrast, you do think that the mean *is* a sensible statistic to compute in light of your research question and your data, then normality violations are unlikely to affect your results—but you can always verify those results using the semiparametric bootstrap. Drawing plots of your data, both for yourself and for your readership, is essential in ensuring that your numeric analyses are relevant to the research questions at hand. Also see Vanhove (2021b).

7.6 Summary

- Data points can be understood as a combination of a common component shared across all observations in the population and an individual deviation specific to each observation.
- In most cases, it is the common component that is of primary interest; this is estimated on the basis of the deviations and an appropriate optimisation criterion.
- The most widely used optimisation criterion is the method of least squares, which in the 'univariate' case (when working with a single variable) yields the mean. However, other methods exist. Under the assumption of independently and identically normally distributed data, the maximum likelihood method leads to the same solution as least squares.
- The uncertainty of parameter estimates can be quantified either by means of the bootstrap or by making further assumptions.
- Rather than worry too much about normality, ask yourself whether what you want to compute is actually relevant in the light of your research questions and your data. If it is and you still have nagging doubts, use the semiparametric bootstrap (or some similar procedure) to construct confidence bounds or to verify the results obtained using t distributions.

Chapter 8

Adding a predictor

Having gained some understanding of how parameters and the uncertainty about those parameters can be estimated in the general linear model, we are now ready to consider a more interesting flavour of this model. What DeKeyser et al. (2010) wanted to find out wasn't so much the mean GJT value as the relationship between AOA and GJT. In this chapter, we'll discuss two approaches to characterise this relationship.

It's always a good idea to draw a graph first. When you're interested in the relationship between two fairly fine-grained numeric variables, a **scatterplot** is a reasonable choice. Since it's impossible for GJT to influence AOA but quite likely that AOA influences GJT, we put the AOA values along the x axis and the GJT values along the y axis (Figure 8.1).

```
ggplot(data = d,
       aes(x = AOA,
           y = GJT)) +
  geom_point(shape = 1) + # shape = 1 draws empty circles
  xlab("Age of acquisition (years)") +
  ylab("Grammaticality judgement score")
```

The scatterplot reveals a general tendency for the GJT to be lower for ever higher AOA values. Moreover, it seems as though this decrease is roughly linear. By way of comparison, Figure 8.2 shows four examples of nonlinear relationships. Moreover, the scatterplot doesn't reveal any implausible data points that could be due to errors during data entry and the like: There are no 207-year-olds or GJT scores beyond the permissible range of $[0, 204]$.

Tip 8.1 (Plot, plot, plot!)**.** Take out time to learn to draw plots, both to learn more about your data and to communicate your findings in talks and papers. These lecture notes feature some basic examples, but for a better introduction, I recommend Kieran Healy's *Data visualization: A practical introduction*, which is available for free at `https://socviz.co/` (Healy, 2019). You may also find my twin tutorials *Working with datasets and visualising data in R* useful (`https://github.com/janhove/DatasetsAndGraphs`). ♢

In what follows, we will address two related questions:

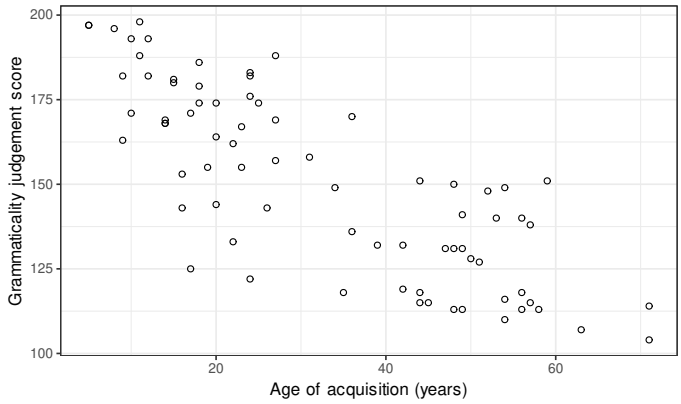


Figure 8.1: Scatterplot of the AOA-GJT relationship.

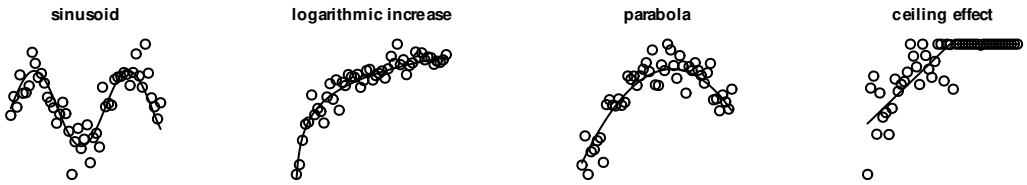


Figure 8.2: Examples of nonlinear relationships.

1. *What* is the relationship between the two variables? In other words, if we know the value of one variable, *how* can we estimate the value of the other? To answer this question, **regression analysis** is commonly used.
2. *How perfect* is the relationship between the GJT and AOA variables? What exactly 'perfect' means in this context will become clear later. To answer this question, **correlation analysis** is often used.

These two questions are often confused with one another, which can lead to misunderstandings (see Vanhove, 2013). Two examples should make the distinction clear.

Example 8.2 (Temperature)**.** If we know the temperature in degrees Celsius, we can 'perfectly' estimate the temperature in degrees Fahrenheit. The correlation is therefore extremely strong (second question). However, this alone does not tell us how to calculate the temperature in degrees Fahrenheit from the temperature in degrees Celsius. A regression analysis would reveal that the following formula is required:

$$\text{degrees Fahrenheit} = 32 + \frac{9}{5} \cdot \text{degrees Celsius}. \tag{8.1}$$

♢

Example 8.3 (Height and weight)**.** If we know a person's height, we can estimate their weight much better than if we did not know their height. The estimate, however, is not perfect, since people of the same height differ in weight. The correlation is therefore positive but not as high as in the previous example. To know how best to estimate weight on the basis of height (e.g. weight in kg $= 0.6$ kg/cm $\cdot$ height in cm $- 35$ kg for women between 145 and 185 cm), we need regression analysis. ♢

In my view, the first question is usually (though not always) the more informative one. In our case, the aim would be to find an equation like Equation 8.1 that links differences in GJT scores to differences in AOA. Nevertheless, since correlation analyses abound in the research literature, the second part of this chapter will be devoted to the second question.

Remark 8.4 (Nonlinear relationships)**.** Correlation and regression analysis can be useful for investigating linear relationships. If the relationship between the variables is not *approximately* straight, then the calculation can still be carried out. However, while the results may be correct mathematically, they risk being irrelevant substantively. When working with quantitative research data, the issue of relevance should be uppermost in your mind.

Sometimes, data can be transformed in a meaningful way so that the relationship becomes linear (for examples, see Baayen, 2008 and Gelman & Hill, 2007). If this is not possible and a graphical representation is insufficient, then more complex methods, such as the generalised additive model, may be appropriate. For accessible introductions, see `https://m-clark.github.io/generalized-additive-models/`, Wieling (2018), and Baayen & Linke (2020). ♢

Tip 8.5 (Questions and tools)**.** Correlation analysis, regression analysis, and other analyses, models, and tests are merely tools. Depending on the research question, these tools

may be useful or useless. Rather than deciding in advance to run a correlation analysis or a t-test (perhaps because this is common practice in a particular research field), or what-have-you, it is better to formulate the question without distracting technical jargon—such as *correlation*, *significant*, *interaction*—and then think about which tool will be most useful in answering it. The aim of data collection and analysis is to answer a question—not to use some fancy tool. ♢

8.1 Simple linear regression

We will apply the techniques learnt in the previous chapter in order to tackle the first question: How does the GJT variable relate to the AOA variable? That is, once we know the value of someone's AOA, what should be our best guess for that person's GJT score?

We proceed as in Chapter 7: we partition the outcome values into a part common to all outcome values and some discrepancy from the general trend. This time, however, we include some information about the link between AOA and GJT in the systematic term:

$$\text{observed value} = \text{systematic component (incl. link with AOA)} + \text{discrepancy}.$$

Specifically, we will model this systematic part in terms of a straight line that is a function of AOA. Straight lines are parametrised by an intercept (we'll write β_0) and a slope (β_1):

$$f(x) = \beta_0 + \beta_1 x.$$

The intercept tells us the $f(x)$ value for $x = 0$; the slope tells us by how much $f(x)$ changes when x is increased by one unit; see Figure 8.3. From this we obtain the following **simple linear regression** equation:

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \tag{8.2}$$

for $i = 1, \ldots, n$. In our running example, y_i still refers to the i-th outcome, i.e., the i-th GJT score, and ε_i still refers to the i-th error. What's new is the predictor, x (with values $x_1, x_2, \ldots, x_n$); in our example, these are the AOA values. The parameters β_0 and β_1 represent the intercept and the slope of the straight line that models the relationship between AOA and GJT.

Remark 8.6 (The Greek letter fallacy)**.** Both the model in Equation 7.1 on page 155 and the one in Equation 8.2 contain the term β_0. But these two β_0s refer to different things (mean vs intercept). The meaning of a model parameter hinges crucially on the other parameters as well as the variables included in the model, so don't equate parameters in different models just because they have the same name. The next chapters in particular will drive this point home. The same goes, incidentally, for the ε_i term. ♢

Incidentally, it's called 'simple linear regression' because we're modelling the outcome in terms of a single predictor (hence simple, as opposed to multiple), and the outcome is modelled as a weighted sum of the predictor values (i.e., as a 'linear combination'). The 'regression' bit stems from its origins in studying the phenomenon of 'regression to the mean'; see Senn (2011) for a readable and short explanation.

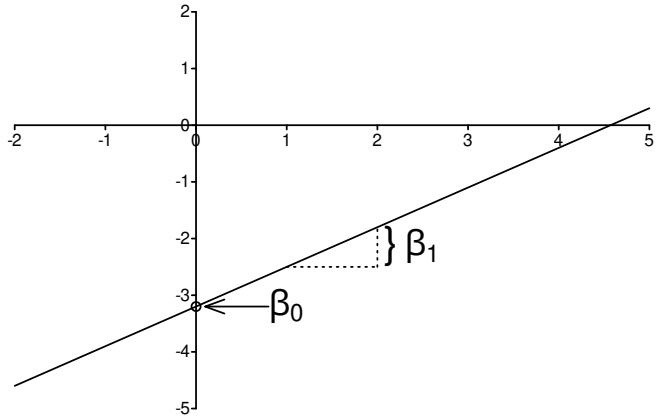


Figure 8.3: Intercept (β_0) and slope (β_1) of a straight line.

8.1.1 Estimating the parameters

If we just pick our estimates for the β_0, β_1 parameters by hand, we could come up with any number of estimates that work sort of okay, just like in Chapter 7. We need a more principled approach.

The estimation approaches we discussed in Chapter 7 still work. We will content ourselves with the least squares method as this is the one you'll encounter in practice: we choose $\widehat{\beta}_0, \widehat{\beta}_1$ so as to minimise the sum of the residuals. (Equivalently, we could choose them so as to maximise the likelihood of the data assuming normally distributed errors.) In principle, we could work through a number of suggestions for $\widehat{\beta}_0$ and $\widehat{\beta}_1$ and then choose the pair of estimates that minimises $\sum_{i=1}^{n} \widehat{\varepsilon}_i^2$. Figure 8.4 shows that estimates of $\widehat{\beta}_0 \approx 190$ and $\widehat{\beta}_1 \approx -1.2$ are optimal in this sense.

We could also derive these estimates mathematically. Conceptually, the maths are fairly easy: we want to solve the same problem as in the previous lecture, but for two parameter estimates simultaneously rather than for just one. But to spell this out precisely, we would need to introduce some matrix algebra. Let's instead skip straight to computing the OLS estimates in R:

```
aoa.lm <- lm(GJT ~ AOA, data = d)
coef(aoa.lm)
```

```
(Intercept)         AOA
 190.408634   -1.217977
```

Having estimated these parameters, we can add the estimated regression line to the scatterplot (Figure 8.5).

```
ggplot(data = d,
       aes(x = AOA,
           y = GJT)) +
  geom_point(shape = 1) +
```

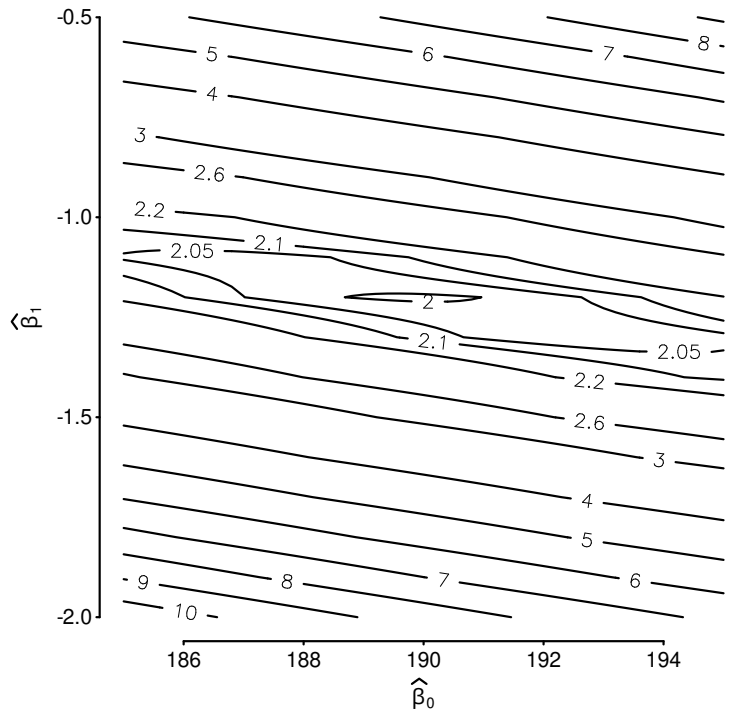


Figure 8.4: The sum of the squared residuals (divided by 10,000) of the GJT data for different combinations of parameter estimates. You can read this plot like a topographic map: the lines are contour lines. The sum is minimised for an intercept estimate near 190 and a slope estimate near -1.2.

```
  geom_abline(intercept = coef(aoa.lm)[1],
              slope = coef(aoa.lm)[2])
```

We could also directly use the `geom_smooth()` function:

```
# not shown
ggplot(data = d,
       aes(x = AOA,
           y = GJT)) +
  geom_point(shape = 1) +
  geom_smooth(method = "lm", se = FALSE)
```

I set the `se` parameter in the `geom_smooth()` layer to `FALSE`. Setting it to `TRUE` would plot a pointwise 95% confidence band—which is a concept we haven't yet discussed.

***Remark 8.7** (Other optimisation criteria)**.** Apart from the least squares method, other methods exist for estimating the parameters in the general linear model. In Chapter 7,

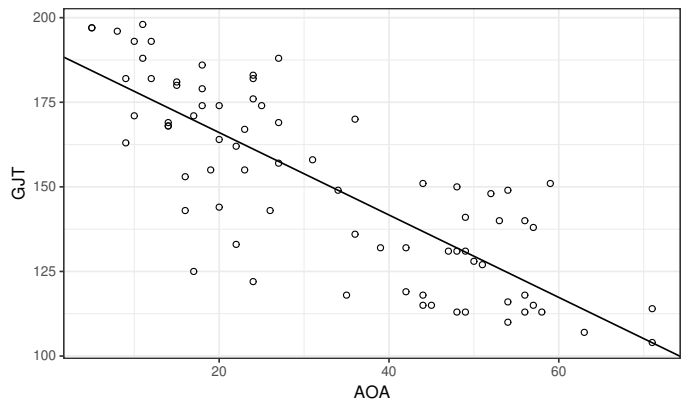


Figure 8.5: Scatterplot with the estimated regression line.

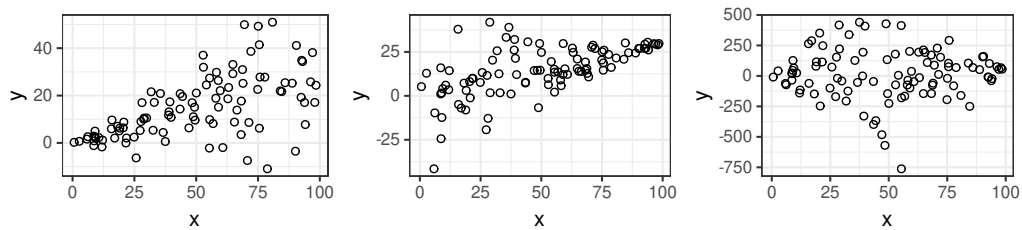


Figure 8.6: In all scatterplots, the spread of the data varies considerably with the x values.

the method of least absolute deviations was already mentioned, which leads to **median regression** (a form of **quantile regression**). Some methods seek to minimise some combination of the sum of the squared deviations and the size of the β estimates. Depending on the precise criterion, these techniques are known as **lasso regularisation**, **ridge regularisation** or the **elastic net**. The basic idea behind these techniques is to deliberately introduce some bias in the estimation of the β parameters but by doing so reduce the variability of the estimates across samples. ♢

8.1.2 Quantifying uncertainty

Our estimates for β_0, β_1 are based on a sample and are hence inherently uncertain. We can estimate the degree of this uncertainty if we are willing to make some assumptions about the errors. As in Chapter 7, we assume that the errors are independently and identically distributed (i.i.d.). We already covered the 'independence' part of this assumption in Chapter 7. The 'identical' bit can be illustrated by taking a look at what some crass violations of it look like: Figure 8.6 shows three examples where the distribution of the errors, more specifically the variance of this distribution, changes with x. If we're willing to make the i.i.d. assumption, we can again obtain uncertainty measures by means of bootstrapping or by assuming normality.

The semiparametric bootstrap

The logic is identical to that in Section 7.3.1 The only difference is that we now estimate two parameters. Instead of generating a vector with bootstrapped estimates, we need to write pairs of estimates in the rows of a two-column matrix.

```
# Step 1
d$Prediction <- predict(aoa.lm)
d$Residual <- resid(aoa.lm)

n_bootstrap <- 20000
# preallocate 20000-by-2 matrix
bs_b <- matrix(nrow = n_bootstrap, ncol = 2)

for (i in 1:n_bootstrap) {
  # Step 2
  bs_resid <- sample(d$Residual, replace = TRUE)
  # Step 3
  d$bs_outcome <- d$Prediction + bs_resid
  # Step 4
  bs_mod <- lm(bs_outcome ~ AOA, data = d)
  # store both estimates in i-th row
  bs_b[i, ] <- coef(bs_mod)
}
```

Let's inspect the first couple of rows of this matrix; if you run this code yourself, you'll obtain different results due to the randomness in step 2:

```
head(bs_b)

         [,1]      [,2]
[1,] 188.1361 -1.240744
[2,] 186.4667 -1.108806
[3,] 194.8941 -1.338165
[4,] 186.6490 -1.171242
[5,] 191.7768 -1.291909
[6,] 191.1294 -1.223668
```

The first column contains the bootstrapped $\widehat{\beta}_0^*$ values; the second column the bootstrapped $\widehat{\beta}_1^*$ values. If you draw a histogram of these bootstrap estimates, you'll notice that both distributions look pretty normal. This is a generalisation of the central limit theorem at play.

```
# not shown
hist(bs_b[, 1])
hist(bs_b[, 2])
```

The standard deviations of these distributions serve as standard errors for $\widehat{\beta}_0, \widehat{\beta}_1$:

```
# apply(x, 2, function) applies function to x by columns
apply(bs_b, 2, sd)
```

```
[1] 3.8214341 0.1034716
```

```
# alternatively:
sd(bs_b[, 1])
sd(bs_b[, 2])
```

That is, we estimate β_0 to be 190.4 ± 3.8 and β_1 to be -1.2 ± 0.1.

Confidence intervals can be obtained like before using the percentile method or by referring to a normal distribution. So for 95% confidence intervals:

```
# percentile method
apply(bs_b, 2, quantile, probs = c(0.025, 0.975))
```

```
          [,1]      [,2]
2.5%  182.8183 -1.421631
97.5% 197.8465 -1.014310
```

```
# normal approximation
mean(bs_b[, 1]) + qnorm(c(0.025, 0.975), sd = sd(bs_b[, 1]))
```

```
[1] 182.8857 197.8655
```

```
mean(bs_b[, 2]) + qnorm(c(0.025, 0.975), sd = sd(bs_b[, 2]))
```

```
[1] -1.420211 -1.014610
```

Because the distributions of the bootstrapped estimates are both pretty much normal, both methods yield essentially the same results.

Note, incidentally, how we made use of the i.i.d. assumption in this bootstrap. In step 2, we resampled the residuals with replacement, but without any further constraints. That is, all draws were independent and all were sampled with from the same distribution, namely the empirical distribution of the residuals.

Alternatives that do not assume equality of error distributions are easy to imagine. We could, for instance, have specified that we only resample the residuals for participants with AOA values over 40 from participants with AOA values over 40, and similarly for participants with lower AOA values. This would correspond to the assumption that the errors for participants with high vs low AOA values could have been sampled from separate distributions. The code to run this bootstrap variation would only be slightly more complicated.

The parametric bootstrap

Alternatively, we could assume that the errors are drawn from a normal distribution, i.e.,

$$\begin{aligned} y_i &= \beta_0 + \beta_1 x_i + \varepsilon_i, \\ \varepsilon_i &\overset{\text{i.i.d.}}{\sim} \text{Normal}(0, \sigma_\varepsilon^2), \end{aligned} \tag{8.3}$$

for $i = 1, 2, \ldots, n$. This notation encapsulates the equality of error distributions assumption: the normal distribution has the same parameters for all $i = 1, 2, \ldots, n$.

The R code to run the bootstrap doesn't contain anything new:

```
# Step 1: Predictions already added to data frame
sigma_aoa.lm <- sigma(aoa.lm)

n_bootstrap <- 20000
bs_b <- matrix(nrow = n_bootstrap, ncol = 2)

for (i in 1:n_bootstrap) {
  # Step 2
  bs_resid <- rnorm(n = nrow(d), sd = sigma_aoa.lm)
  # Step 3
  d$bs_outcome <- d$Prediction + bs_resid
  # Step 4
  bs_mod <- lm(bs_outcome ~ AOA, data = d)
  bs_b[i, ] <- coef(bs_mod)
}
```

Histograms show that the bootstrapped β estimates are normally distributed:

```
# not shown
hist(bs_b[, 1])
hist(bs_b[, 2])
```

The confidence intervals and estimated standard errors we obtain are essentially the same as before.

```
apply(bs_b, 2, quantile, probs = c(0.025, 0.975))

          [,1]      [,2]
2.5%  182.6498 -1.425972
97.5% 198.0572 -1.010111

apply(bs_b, 2, sd)

[1] 3.9278766 0.1060721
```

Now's a good time to visualise the uncertainty about the location of the regression line. We have 20,000 pairs of bootstrapped estimates of β_0, β_1. To gauge the uncertainty about the location of the regression line, we can draw a handful of the straight lines implied

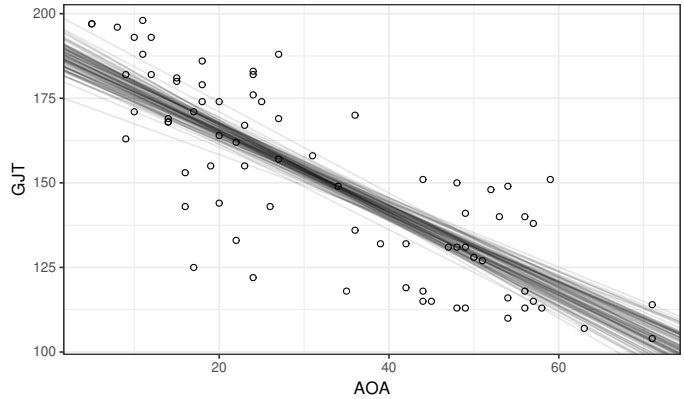


Figure 8.7: Scatterplot with 100 bootstrapped regression lines.

by these estimates. In the code below, the first 100 pairs of estimates are used; Figure 8.7 shows the result.

```
plot_bs <- ggplot(data = d,
                  aes(x = AOA,
                      y = GJT)) +
  geom_point(shape = 1)

for (i in 1:100) {
  plot_bs <- plot_bs +
    geom_abline(intercept = bs_b[i, 1],
                slope = bs_b[i, 2],
                # use alpha to make the lines a bit transparent
                alpha = 1/10)
}

plot_bs
```

In practice, people don't draw these bootstrapped regression lines. Instead, they colour in the region in which most of these lines fall. This region is known as a (pointwise) **confidence band**. ('Pointwise' as opposed to 'simultaneous', see Remark 8.8 on page 181.) We won't cover simultaneous confidence bands in these lectures.) Let's see how we can draw such confidence bands ourselves as this'll give us some further insights into the general linear model. First, we generate a sequence of predictor values for which we want to plot the confidence band. Here, we just take all integers between the minimum and maximum AOA values in our sample:

```
# Step 1
new_aoa <- seq(from = min(d$AOA), to = max(d$AOA), by = 1)
# that is, 5, 6, 7, ..., 69, 70, 71
```

There are 67 values in `new_aoa`. We have already generated 20,000 pairs of bootstrapped parameter estimates. For each of these pairs, we can compute the location of the bootstrapped regression line at each of the newly created predictor values. For instance, we can evaluate the regression line for the 37th bootstrap run like so:

```
bs_b[37, 1] + bs_b[37, 2] * new_aoa
```

```
 [1] 180.9399 179.8455 178.7512 177.6569 176.5625 175.4682
 [7] 174.3739 173.2796 172.1852 171.0909 169.9966 168.9022
[13] 167.8079 166.7136 165.6192 164.5249 163.4306 162.3362
[19] 161.2419 160.1476 159.0532 157.9589 156.8646 155.7702
[25] 154.6759 153.5816 152.4873 151.3929 150.2986 149.2043
[31] 148.1099 147.0156 145.9213 144.8269 143.7326 142.6383
[37] 141.5439 140.4496 139.3553 138.2609 137.1666 136.0723
[43] 134.9779 133.8836 132.7893 131.6950 130.6006 129.5063
[49] 128.4120 127.3176 126.2233 125.1290 124.0346 122.9403
[55] 121.8460 120.7516 119.6573 118.5630 117.4686 116.3743
[61] 115.2800 114.1856 113.0913 111.9970 110.9027 109.8083
[67] 108.7140
```

This yields 67 values—one for each `new_aoa` value. We now carry out this computation not only for the 37th bootstrap run but for all 20,000 runs. We store the results into a 20,000-by-67 matrix:

```
# Step 2
bs_y_hat <- matrix(nrow = n_bootstrap,
                   ncol = length(new_aoa))
for (i in 1:n_bootstrap) {
  bs_y_hat[i, ] <- bs_b[i, 1] + bs_b[i, 2]*new_aoa
}
```

Each row of this matrix contains the location of a different bootstrapped regression line corresponding to all 67 different values of `new_aoa`. We look up the 2.5th and 97.5th percentile of the generated values at each `new_aoa` value; at each point, 95% of the bootstrapped regression lines lie between these percentiles:

```
# Step 3
lo_95 <- apply(bs_y_hat, 2, quantile, probs = 0.025)
hi_95 <- apply(bs_y_hat, 2, quantile, probs = 0.975)
```

Finally, we combine the predictor values and these percentile values into a tibble and plot them.[1] See Figure 8.8.

```
# Step 4
confidence_band_tbl <- tibble(new_aoa, lo_95, hi_95)
```

[1] Tibbles are the counterpart of standard data frames when using the tidyverse package. But you can use `data.frame()` instead of `tibble()` if you prefer.

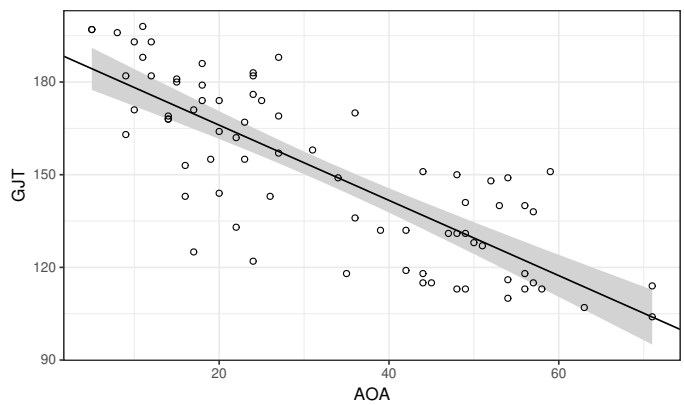


Figure 8.8: Scatterplot with a parametric bootstrap-based 95% confidence band for the regression line.

```
ggplot(data = confidence_band_tbl,
       aes(x = new_aoa)) +
  geom_ribbon(aes(ymin = lo_95,
                  ymax = hi_95),
              fill = "lightgrey") +
  geom_point(data = d,
             aes(x = AOA, y = GJT),
             shape = 1) +
  geom_abline(intercept = coef(aoa.lm)[[1]],
              slope = coef(aoa.lm)[[2]]) +
  xlab("AOA") +
  ylab("GJT")
```

***Remark 8.8** (Pointwise vs simultaneous confidence bands)**.** The confidence bands discussed here are so-called **pointwise confidence bands**. A well-calibrated pointwise $100(1-\alpha)\%$ confidence band provides, for each x value, a $100(1-\alpha)\%$ confidence interval for the corresponding value $\beta_0 + \beta_1 x$.

Occasionally, one also encounters **simultaneous confidence bands**. A simultaneous $100(1-\alpha)\%$ confidence band should, for a set of x values $x_1, x_2, \ldots, x_n$, guarantee that with probability at least $1-\alpha$, the band covers *all* of the corresponding $\beta_0 + \beta_1 x_i$, $i = 1, \ldots, n$. This is a much stricter condition than for pointwise confidence bands! Accordingly, simultaneous confidence bands are wider than pointwise ones.

In these lecture notes, we only consider pointwise confidence bands. When confidence bands are reported in the literature without further specification, you can safely assume that they are pointwise ones. ♢

t distributions

As long as we're assuming that the errors are i.i.d. draws from a normal distribution, we might as well estimate the standard errors and compute the confidence intervals for the parameter estimates directly using *t* distributions.

```
summary(aoa.lm)


Call:
lm(formula = GJT ~ AOA, data = d)

Residuals:
    Min      1Q  Median      3Q     Max 
-44.703  -9.542  -0.260  13.021  32.452 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 190.4086     3.9040   48.77   <2e-16 
AOA          -1.2180     0.1051  -11.58   <2e-16 

Residual standard error: 16.4 on 74 degrees of freedom
Multiple R-squared:  0.6446, Adjusted R-squared:  0.6398 
F-statistic: 134.2 on 1 and 74 DF,  p-value: < 2.2e-16
```

```
confint(aoa.lm)
```

```
                2.5 %     97.5 %
(Intercept) 182.62969 198.187578
AOA          -1.42747  -1.008484
```

We can also directly obtain a confidence interval for the location of the regression line at some prespecified predictor values using the `predict()` function. The following command generates a 95% confidence interval for the location of the regression line at each of the 67 values in `new_aoa`.

```
conf_band_t <- predict(aoa.lm, newdata = tibble(AOA = new_aoa),
                       interval = "confidence")
head(conf_band_t) # 1st row: AOA = 5, 2nd: AOA = 6, etc.
```

```
       fit      lwr      upr
1 184.3188 177.4392 191.1983
2 183.1008 176.3959 189.8056
3 181.8828 175.3506 188.4150
4 180.6648 174.3031 187.0266
5 179.4468 173.2532 185.6405
6 178.2289 172.2007 184.2571
```

We can plot the result of these computations, the result of which is hardly discernible from the one obtained previously:

```
# not shown
conf_band_t <- as.tibble(conf_band_t)
conf_band_t$AOA <- new_aoa
ggplot(data = conf_band_t,
       aes(x = AOA)) +
  geom_ribbon(aes(ymin = lwr,
                  ymax = upr),
              fill = "lightgrey") +
  geom_point(data = d,
             aes(x = AOA, y = GJT),
             shape = 1) +
  geom_line(aes(y = fit)) +
  xlab("AOA") +
  ylab("GJT")
```

Once we've understood what we're really doing when drawing a confidence band, we can do so without all the rigmarole:

```
# not shown
ggplot(data = d,
       aes(x = AOA, y = GJT)) +
  geom_point(shape = 1) +
  geom_smooth(method = "lm")
```

***Exercise 8.9.** Use the semiparametric bootstrap to draw a 95% confidence band for the AOA–GJT regression model. ♢

8.1.3 Interpreting regression lines

Equation 8.3 on page 178 is useful for getting a conceptual grasp on the regression line. According to this equation, we assume that the errors are drawn i.i.d. from a normal distribution with mean 0 and variance σ_ε^2. For a fixed x value, the expected distribution of the y values, then, is a normal distribution centred on $\beta_0 + \beta_1 x$ and with variance σ_ε^2 Plugging in our estimates for $\beta_0, \beta_1, \sigma_\varepsilon^2$, we obtain the estimated expected distribution of y for a given x value. The estimated regression line, then, shows the estimated **conditional means** of y for the different values of x. Figure 8.9 illustrates this concept graphically. The cross-section of the 95% confidence band at a given x value is the 95% confidence interval of the corresponding conditional mean.

Let's look at a couple of examples of how we can interpret our model.

- According to the `aoa.lm` model, the estimated β parameters are 190.4 and -1.22. So according to this model, if we were to sample lots of participants with an AOA of 15, the best guess for their mean GJT score would be $190.4 - 1.22 \cdot 15 = 172.1$. The same answer can be obtained using `predict()`:

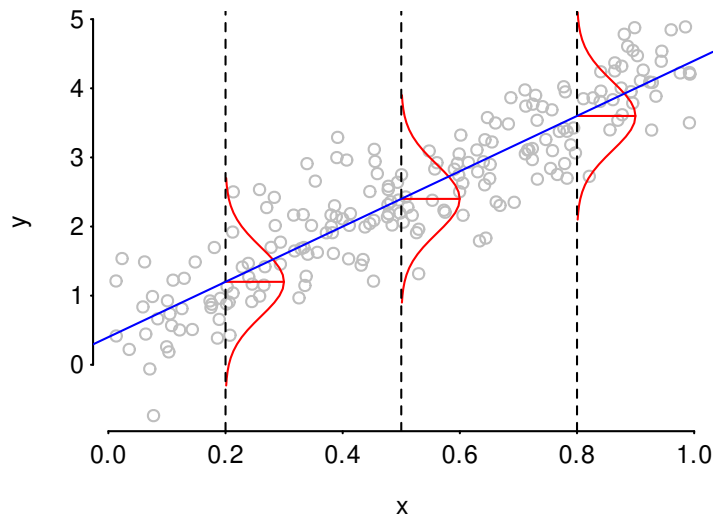


Figure 8.9: If we assume that the errors are i.i.d., then the regression line connects the estimated conditional means for the distribution of y at different x values. In this illustration, the errors are all assumed to be normally distributed, but this is not a necessary assumption for making sense of the regression line. If the errors aren't normally distributed, however, it's possible that the conditional means don't capture what's interesting about how x and y relate to each other.

```
predict(aoa.lm, newdata = tibble(AOA = 15))

      1
172.139
```

Note, however, that there are two participants in the data set with an AOA of 15, and their mean GJT score isn't 172.1:

```
d |> filter(AOA == 15)

# A tibble: 2 x 6
  Participant   AOA   GJT Prediction Residual bs_outcome
  <chr>       <dbl> <dbl>      <dbl>    <dbl>      <dbl>
1 P49            15   180       172.     7.86       141.
2 P59            15   181       172.     8.86       166.
```

The extent to which our model-based estimate of the conditional GJT mean for an AOA value of 15 is more accurate than the mean of these two observations depends on the accuracy of our modelling assumptions—especially the assumption that the AOA–GJT relationship is roughly linear. This doesn't seem too much of a stretch, and, conceptually, this assumption allows us to estimate the conditional mean for a given AOA value more accurately by leveraging the information about the relationship between AOA and GJT that we can extract from the other data points.

- There are no participants with an AOA of 21 in our sample. But according to our model, if we were to sample lots of participants with this AOA, their mean GJT would be about 165:

```
predict(aoa.lm, newdata = tibble(AOA = 21))

       1
164.8311
```

This is an example of **interpolation**, as we have both participants with lower and with higher AOA values.

- There are no participants with an AOA of 82 in our sample. But according to our model, if we were to sample lots of participants with this AOA, their mean GJT would be about 91:

```
predict(aoa.lm, newdata = tibble(AOA = 82))

       1
90.53455
```

This is an example of **extrapolation**, since the maximum AOA in our sample is 71.

- The mean AOA in our sample is about 32.5. The estimated GJT mean for this AOA corresponds exactly to the GJT mean for the entire sample:

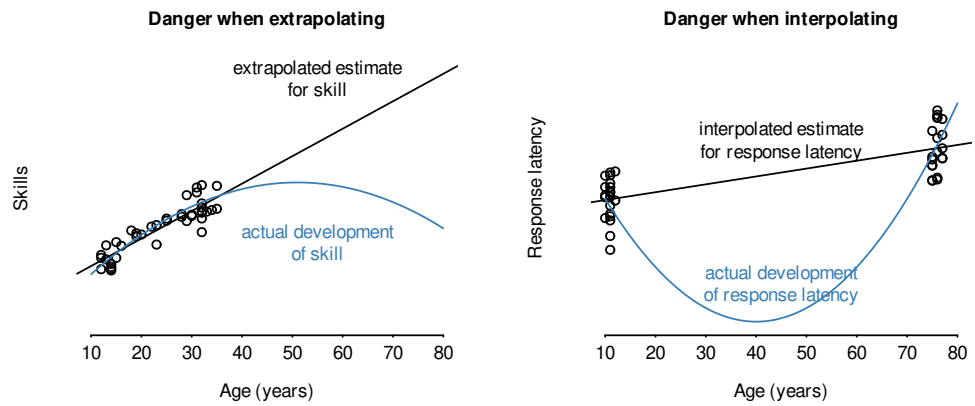


Figure 8.10: Dangers when extra- or interpolating.

```
mean(d$GJT)

[1] 150.7763

predict(aoa.lm, newdata = tibble(AOA = mean(d$AOA)))

       1
150.7763
```

This is not a coincidence but a general phenomenon.

Proposition 8.10. In a simple linear model, the estimated conditional mean for the mean predictor value equals the sample mean of the outcome. ♢

Proof. According to the regression model, $y_i = \widehat{\beta}_0 + \widehat{\beta}_1 x_i + \widehat{\varepsilon}_i$ for $i = 1, 2, \ldots, n$. Hence we have

$$\frac{1}{n}\sum_{i=1}^{n} y_i = \frac{1}{n}\sum_{i=1}^{n}(\widehat{\beta}_0 + \widehat{\beta}_1 x_i + \widehat{\varepsilon}_i) = \frac{1}{n}n\widehat{\beta}_0 + \frac{1}{n}\widehat{\beta}_1 \sum_{i=1}^{n} x_i + \frac{1}{n}\sum_{i=1}^{n}\widehat{\varepsilon}_i.$$

It can be shown that the mean of the residuals is always zero when using OLS. So this simplifies to

$$\frac{1}{n}\sum_{i=1}^{n} y_i = \widehat{\beta}_0 + \widehat{\beta}_1 \left(\frac{1}{n}\sum_{i=1}^{n} x_i\right). \qquad \square$$

Use your common sense when inter- and extrapolating. If we have a sample of participants aged 8–26, we'd be on thin ice extrapolating to participants aged 5 or 45; see Figure 8.10, left. The plot on the right illustrates the dangers with interpolation. That said, even with densely sampled data it is still *possible* that a seemingly linear relationship is strongly nonlinear; see Figure 8.11. At the end of the day, statistical inference is about combining data with assumptions.

***Remark 8.11** (Confidence intervals around conditional means)**.** Confidence intervals around conditional means can be calculated using the bootstrap methods described

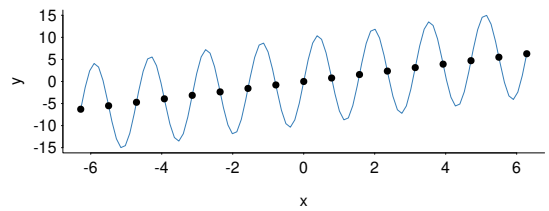


Figure 8.11: If we only observed the black dots, we'd probably think the relationship was linear. How much we'd be wrong when interpolating between the measuring points depends on the amplitude of the wave.

above by constructing the confidence band for a single value of x. Alternatively, one can use `predict()`, in which case the confidence interval is constructed on the basis of the appropriate t distribution:

```
predict(aoa.lm, newdata = tibble(AOA = 35),
        interval = "confidence", level = 0.80)

       fit      lwr      upr
1 147.7795 145.3246 150.2343
```

$\diamondsuit$

Remark 8.12 (Making the intercept more interpretable)**.** The intercept $\widehat{\beta}_0$ represents the estimated conditional mean for participants with a predictor value of 0. However, zero lies outside the range of AOA values in our sample. A common trick to render the estimated intercept more informative about the data is to **centre** the predictor at some sensible value, usually the mean or the median. Centring just means subtracting this sensible value from all the predictor values before fitting the model:

```
# centre AOA
d$c.AOA <- d$AOA - mean(d$AOA)

# fit model again
aoa.lm <- lm(GJT ~ c.AOA, data = d)

# model coefficients
summary(aoa.lm)$coefficients

              Estimate Std. Error   t value     Pr(>|t|)
(Intercept) 150.776316  1.8807316  80.16897 1.120538e-73
c.AOA        -1.217977  0.1051385 -11.58450 2.728150e-18
```

The advantage of centring is that the estimated intercept now represents the conditional mean of the outcome for the predictor value around which was centred (including the estimated standard error for this mean). If we centre around the mean predictor value, then we've already seen that this yields the sample mean, see Proposition 8.10.

Note that in order to compute the conditional mean for an AOA value of 35, we have to subtract the AOA mean from this predictor value, too, when we centred the predictor for our model:

```
predict(aoa.lm, newdata = tibble(c.AOA = 35 - mean(d$AOA)))

       1
147.7795
```

So not (!):

```
predict(aoa.lm, newdata = tibble(c.AOA = 35))

       1
108.1471
```

♢

8.1.4 Assumptions and relevance

Three of the assumptions we've made throughout this section—linear relationship, equality of error distributions, and normality of errors— almost never hold literally. Instead of worrying whether these assumptions literally hold (they don't), ask yourself whether your model is relevant to the questions you want to answer. We have already discussed this for the normality assumption in Section 7.5, so let's go over the linearity and equality of error distributions assumptions.

The simple linear model seeks to characterise the linear relationship between the predictor and the outcome. If the relationship between predictor and outcome isn't approximately linear, the model's output will still be literally correct in many situations—but it may not help you find out what you want to know. In such cases, purely graphical data analyses, data transformations or models capable of capturing nonlinearities (e.g., generalised additive models) may be of interest.

As for the assumption that the errors all have the same distribution, recall the Gauss–Markov theorem (Theorem 5.9 on page 124), which guarantees that the mean of a simple random sample has the lowest variance among all unbiased estimators of a distribution's expected value. The version of the theorem presented on page 124 is a special case of the original theorem, which states that OLS estimation has the lowest variance among all unbiased estimation methods, provided that the errors are i.i.d. If the errors don't all have the same distribution, then OLS estimation is still unbiased. But there may be other unbiased estimation methods that have lower variance. Further, since the general linear model assumes that all errors stem from the same distribution, it won't be able to pick up on non-identical error structures. If you think such differences are relevant to your research project, you may want to resort to models that not only capture conditional means but changes in the error distribution as well (e.g., generalised least-squares models).

As always, plotting the data prevents you and your readership from flying blind. If, on the other hand, you fear that you get too paranoid about assumptions everytime you plot your data, you may want to take a look at the methods discussed in Vanhove (2018).

The fourth assumption—independence of errors—is, however, absolutely vital for correct inference, as stressed in Section 7.5.

Exercise 8.13. Let's fit another model to DeKeyser et al.'s data:

```
gjt.lm <- lm(AOA ~ GJT, data = d)
summary(gjt.lm)$coefficients
```

```
               Estimate Std. Error   t value     Pr(>|t|)
(Intercept) 112.3328045 6.99861763  16.05071 8.254024e-26
GJT          -0.5292166 0.04568318 -11.58450 2.728150e-18
```

1. What do the parameter estimates for `(Intercept)` and `GJT` mean—literally?
2. Which of the two models (`aoa.lm` or `gjt.lm`) is the most relevant in the context of DeKeyser et al.'s study? Why? ♢

Exercise 8.14. The dataset `vanhove2014_cognates.csv` contains a summary of some data I collected for my PhD thesis (Vanhove, 2014). 163 speakers of German were asked to translate 45 written and 45 spoken Swedish words into German. The columns `CorrectSpoken` and `CorrectWritten` contain the number of correct translations per participants in both modalities. The dataset `vanhove2014_background.csv` contains some background information about the participants, including their performance on a handful cognitive tests.

Let's first combine these datasets.

```
cognates <- read_csv(here("data", "vanhove2014_cognates.csv"))
background <- read_csv(here("data", "vanhove2014_background.csv"))
all_data <- cognates |>
  left_join(background, by = "Subject")
```

Now try to answer the following questions.

1. The column `DS.Span` contains the participants' score on a working memory task. How is their performance on this task associated with `CorrectSpoken`?
2. How is their performance on an English proficiency test (`English.Overall`) associated with `CorrectWritten`?
3. How does their performance on the Swedish translation tasks vary with `Age` in both modalities? ♢

8.1.5 Summary

- The procedures introduced in Chapter 7 can be extended quite easily in order to model the relationship between two numeric variables. In the next chapters, we'll broaden the scope even more.
- The intercept represents the estimated outcome value when the predictor variable is fixed at 0. Depending on your data and research question, this may not be relevant

information. But you can translate your predictor data to make the intercept more meaningful.

- The slope is the estimated difference between the outcome distributions of observations that differ by one unit in the predictor variable.

8.2 Correlation analysis

Let us now turn to the problem of quantifying how 'perfect' the relation between two numeric variables is. To describe numerically how strongly two random variables X, Y are related, we need a measure whose absolute value is large when small differences in X correspond to small differences in Y and large differences in X correspond to large differences in Y, and whose absolute value is small when large differences in one variable correspond to only small differences in the other. Such a measure is the **covariance**.

Definition 8.15 (Population covariance)**.** Let X and Y be two random variables with finite variance on a probability space Ω. Then the covariance of X and Y is defined as

$$\begin{aligned}\mathrm{Cov}(X,Y) &:= \mathbb{E}\left((X-\mathbb{E}(X))\,(Y-\mathbb{E}(Y))\right)\\ &= \mathbb{E}\left(XY - X\mathbb{E}(Y) - \mathbb{E}(X)Y + \mathbb{E}(X)\mathbb{E}(Y)\right)\\ &= \mathbb{E}(XY) - \mathbb{E}(X)\mathbb{E}(Y) - \mathbb{E}(X)\mathbb{E}(Y) + \mathbb{E}(X)\mathbb{E}(Y)\\ &= \mathbb{E}(XY) - \mathbb{E}(X)\mathbb{E}(Y).\end{aligned}$$

$\diamondsuit$

Example 8.16. We revisit the Jass example from Chapter 3. In Example 3.28, the variable X was defined by the point value of the cards in the *Obenabe* game:

$$X(\omega) := \begin{cases} 11, & \text{if } \omega \text{ is an Ace,}\\ 4, & \text{if } \omega \text{ is a King,}\\ 3, & \text{if } \omega \text{ is a Queen,}\\ 2, & \text{if } \omega \text{ is a Jack,}\\ 10, & \text{if } \omega \text{ is a 10,}\\ 8, & \text{if } \omega \text{ is an 8,}\\ 0, & \text{otherwise.}\end{cases}$$

We now define an additional variable Y by the points of the cards in the *Undenufe* game:

$$Y(\omega) := \begin{cases} 11, & \text{if } \omega \text{ is a 6,}\\ 4, & \text{if } \omega \text{ is a King,}\\ 3, & \text{if } \omega \text{ is a Queen,}\\ 2, & \text{if } \omega \text{ is a Jack,}\\ 10, & \text{if } \omega \text{ is a 10,}\\ 8, & \text{if } \omega \text{ is an 8,}\\ 0, & \text{otherwise.}\end{cases}$$

As you can verify, $\mathbb{E}(Y) = \mathbb{E}(X) = 4.22$. To calculate the covariance, we take a deep breath:

$$\begin{aligned}
\mathrm{Cov}(X,Y) &= \mathbb{E}\left((X-\mathbb{E}(X))\,(Y-\mathbb{E}(Y))\right) \\
&= \frac{1}{36}(X(6\clubsuit)-\mathbb{E}(X))(Y(6\clubsuit)-\mathbb{E}(Y))+ \\
&\quad \frac{1}{36}(X(6\diamondsuit)-\mathbb{E}(X))(Y(6\diamondsuit)-\mathbb{E}(Y))+ \\
&\quad \cdots + \\
&\quad \frac{1}{36}(X(A\spadesuit)-\mathbb{E}(X))(Y(A\spadesuit)-\mathbb{E}(Y)) \\
&= \frac{1}{9}(0-4.22)(11-4.22)+\frac{2}{9}(0-4.22)(0-4.22)+ \\
&\quad \frac{1}{9}(8-4.22)(8-4.22)+\frac{1}{9}(10-4.22)(10-4.22)+ \\
&\quad \frac{1}{9}(2-4.22)(2-4.22)+\frac{1}{9}(3-4.22)(3-4.22)+ \\
&\quad \frac{1}{9}(4-4.22)(4-4.22)+\frac{1}{9}(11-4.22)(0-4.22) \\
&\approx 0.55.
\end{aligned}$$

♢

Lemma 8.17 (Properties of covariance)**.** Let X, Y, Z be three random variables with finite variance on a common probability space, and let a, b be constants. Then:

1. $\mathrm{Cov}(X,X) = \mathrm{Var}(X)$.
2. $\mathrm{Cov}(X,Y) = \mathrm{Cov}(Y,X)$.
3. $\mathrm{Cov}(X+Y,Z) = \mathrm{Cov}(X,Z)+\mathrm{Cov}(Y,Z)$.
4. $\mathrm{Cov}(aX,bY) = ab\mathrm{Cov}(X,Y)$.
5. If X and Y are independent, then $\mathrm{Cov}(X,Y) = 0$.

The first four of these properties may be verified using Definition 8.15; the last follows from the fact that $\mathbb{E}(XY) = \mathbb{E}(X)\mathbb{E}(Y)$ if X, Y are independent.

Using these properties, we can derive a general formula for the variance of the sum of two random variables:

$$\begin{aligned}
\mathrm{Var}(X+Y) &= \mathrm{Cov}(X+Y,X+Y) && [(1)] \\
&= \mathrm{Cov}(X,X+Y)+\mathrm{Cov}(Y,X+Y) && [(3)] \\
&= \mathrm{Cov}(X+Y,X)+\mathrm{Cov}(X+Y,Y) && [(2)] \\
&= \mathrm{Cov}(X,X)+\mathrm{Cov}(Y,X)+\mathrm{Cov}(X,Y)+\mathrm{Cov}(Y,Y) && [(3)] \\
&= \mathrm{Var}(X)+2\mathrm{Cov}(X,Y)+\mathrm{Var}(Y). && [(1,2)]
\end{aligned}$$

In particular, if $\mathrm{Cov}(X,Y) = 0$, then $\mathrm{Var}(X+Y) = \mathrm{Var}(X)+\mathrm{Var}(Y)$. This proves Lemma 3.40 on page 82. ♢

Example 8.18 (Zero covariance, but dependent)**.** If two random variables are independent, then $\text{Cov}(X, Y) = 0$. The converse, however, is not true. To see this, consider the sample space

$$\Omega := \{(0,0), (1,-1), (1,1)\}$$

with a uniform probability distribution. Let the random variable X map (ω_1, ω_2) to ω_1, and let Y map (ω_1, ω_2) to ω_2. Then $\mathbb{E}(X) = 2/3$ and $\mathbb{E}(Y) = 0$. Therefore,

$$\text{Cov}(X, Y) = \frac{1}{3}(0 - 2/3)(0 - 0) + \frac{1}{3}(1 - 2/3)(-1 - 0) + \frac{1}{3}(1 - 2/3)(1 - 0) = 0.$$

However,

$$\mathbb{P}(X = 1, Y = 1) = \frac{1}{3} \neq \frac{2}{3} \cdot \frac{1}{3} = \mathbb{P}(X = 1)\mathbb{P}(Y = 1),$$

so X and Y are dependent. ♢

The covariance between two random variables is usually estimated using the **sample covariance**.

Definition 8.19 (Sample covariance)**.** Let $(X_1, Y_1), \ldots, (X_n, Y_n)$ be independent and identically distributed pairs of observations. Then the sample covariance is defined as

$$\widehat{\text{Cov}}_{XY} := \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \overline{X})(Y_i - \overline{Y}), \tag{8.4}$$

where $\overline{X}$ and $\overline{Y}$ are the sample means of $\boldsymbol{X}$ and $\boldsymbol{Y}$. ♢

The sum of the products, $\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})$, is divided by $n - 1$ rather than n for the same reason as in the calculation of the sample variance. A more intuitive explanation is that you cannot really speak about the relationship between two variables if you only have one observation per variable. A scatterplot would then consist of a single point. When $n = 1$, $n - 1 = 0$, and the formula yields no answer because division by zero is undefined.

In R:

```
# more complicated:
sum((d$AOA - mean(d$AOA)) * (d$GJT - mean(d$GJT))) / (nrow(d) - 1)

[1] -394.9311

# simpler:
cov(d$AOA, d$GJT)

[1] -394.9311
```

If the covariance is positive, there is a positive linear relationship between the two variables (larger x tends to correspond to larger y); if the covariance is negative, there is a negative linear relationship (larger x tends to correspond to smaller y). Aside from these two rules of thumb, the covariance measure is difficult to interpret, which is why

you rarely encounter it in the research literature. Nonetheless, covariance is an important concept in the mathematics behind more complex procedures, so it is worth at least being aware of it.

Because covariance is not particularly easy to interpret, Pearson's **product-moment correlation coefficient** (or simply Pearson's correlation) is usually preferred. This number indicates how well the relationship can be described by a straight line. It is calculated similarly to the covariance, but the variables are expressed in standard deviations from their sample means, producing a value between -1 and 1:

$$\rho_{XY} := \frac{\mathrm{Cov}(X, Y)}{\sigma_X \sigma_Y}.$$

When estimating the correlation from a sample $(\boldsymbol{X}, \boldsymbol{Y})$, the formula is adapted accordingly:

$$r(\boldsymbol{X}, \boldsymbol{Y}) := \widehat{\rho}_{\boldsymbol{XY}} := \frac{\widehat{\mathrm{Cov}}(\boldsymbol{X}, \boldsymbol{Y})}{s(\boldsymbol{X})s(\boldsymbol{Y})} = \frac{1}{n-1} \sum_{i=1}^{n} \frac{X_i - \overline{X}}{s(\boldsymbol{X})} \frac{Y_i - \overline{Y}}{s(\boldsymbol{Y})}.$$

```
# more complicated:
cov(d$AOA, d$GJT) / (sd(d$AOA) * sd(d$GJT))

[1] -0.8028533

# simpler:
cor(d$AOA, d$GJT)

[1] -0.8028533
```

As soon as any value is missing, the `cor()` function returns 'NA' (not available). One way to handle this is to ignore observations with one or two missing values:

```
cor(d$AOA, d$GJT, use = "pairwise.complete.obs")

[1] -0.8028533
```

If $r = 1$, all data points lie perfectly on a straight, ascending line. This almost always signals a tautology. For instance, heights measured in centimetres and in inches are perfectly correlated, but this relationship is not impressive—just extremely mundane. If $r = -1$, all data points lie on a straight, descending line, which typically indicates that the two variables are perfectly complementary. For example, the number of correct responses in a test often correlates at $r = -1$ with the number of incorrect responses—again, not particularly remarkable. If $r = 0$, the line is perfectly vertical, i.e., there is no linear relationship between the two variables at all.

The larger the absolute value of r, the closer the data points lie to the straight line. In other words, the larger the absolute value of r, the more precisely one can predict y from x using a linear equation (and vice versa) than if x were unknown. The correlation between $\boldsymbol{X}$ and $\boldsymbol{Y}$ is the same as that between $\boldsymbol{Y}$ and $\boldsymbol{X}$, so it makes no difference whether you type `cor(dat$AOA, dat$GJT)` or `cor(dat$GJT, dat$AOA)`.

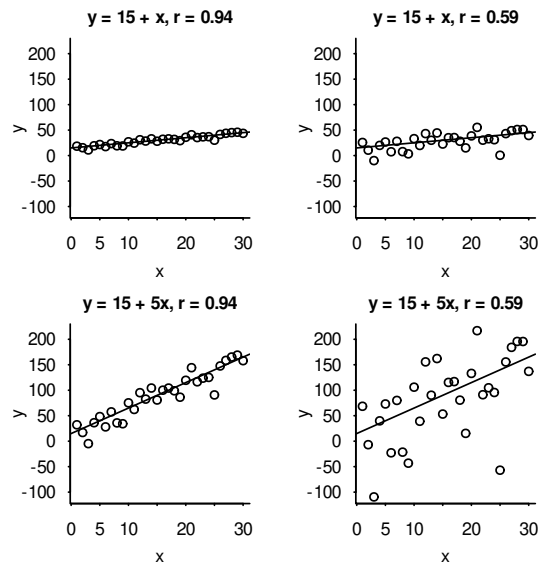


Figure 8.12: Correlation coefficients tell you little about the shape of a relationship.

Example 8.20 (Form vs strength of a relationship)**.** Figure 8.12 shows four relationships to better illustrate the meaning of Pearson's r.

- *Top left:* There is little scatter along the y axis. The variation that exists is mostly captured by a straight line. Hence, r is very high.
- *Top right:* There is now more scatter along the y axis; this is less well captured by a straight line, resulting in a lower correlation coefficient. The line has the same slope as in the left plot, but the correlation coefficient differs.
- *Bottom left:* There is substantial scatter along the y axis, but it is still largely captured by a straight line. r is therefore again very high. While the correlation coefficient is similar to that of the plot above, the slope of the line differs.
- *Bottom right:* The same line captures the scatter along the y axis less well, so while the slope of the line is the same, the correlation coefficient is lower. ♢

Remark 8.21 (Data patterns behind correlation coefficients)**.** As we have just seen, Pearson's r expresses the proportion of variation in the points of a scatterplot that is captured by a *straight line*. It doesn't answer the question of what the line looks like (other than if it increases or decreases); see the four examples above. You'd need regression analysis for this.

It is also possible for there to be a very strong but nonlinear relationship between two variables that is not reflected in Pearson's r (Figure 8.13, left). Conversely, r can give the impression of a fairly strong linear relationship even when such a relationship barely exists for most of the points (centre), or even when the relationship actually runs in the opposite direction. For example, in the right-hand plot there are two groups where the

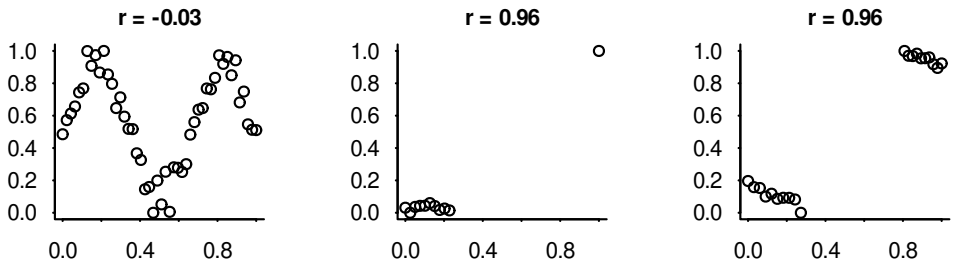


Figure 8.13: A correlation coefficient near 0 does not necessarily mean that there is no relationship between the variables, and a coefficient near 1 does not necessarily mean that the pattern in the data is best described by a strong positive relationship.

association is negative. However, the coefficient is positive if the two groups are considered together. The issue here is not that r has been miscalculated, but that computing r in this situation is largely meaningless. Before you worry about correctness, first worry about relevance.

With the `plot_r()` function from the `cannonball` package, you can draw scatterplots that all look different but share the same correlation coefficient. Instructions for installing the package can be found at `https://github.com/janhove/cannonball`. The blog post *What data patterns can lie behind a correlation coefficient?* (21 November 2016) describes the `plot_r()` function. Figure 8.14 shows sixteen relationships, each with 50 observations, all exhibiting a correlation of $r = -0.72$:

```
library(cannonball)
plot_r(n = 50, r = -0.72)
```

Play around with the `plot_r()` function to better understand what a correlation coefficient does *not* tell you, and the influence of nonlinear relationships and outliers:

```
plot_r(n = 15, r = 0.9)
plot_r(n = 80, r = 0.0)
plot_r(n = 30, r = 0.4)
```

You can access the function documentation with `?plot_r` as usual. ♢

To summarise, a single correlation coefficient can correspond to a wide variety of relationships. Always inspect your data graphically using scatterplots before calculating correlation coefficients. Include these scatterplots in your papers, assignments, and presentations. In my opinion, a correlation coefficient without a scatterplot is essentially meaningless.

Besides Pearson's correlation coefficient, you will occasionally encounter other measures of association between two variables in the research literature.

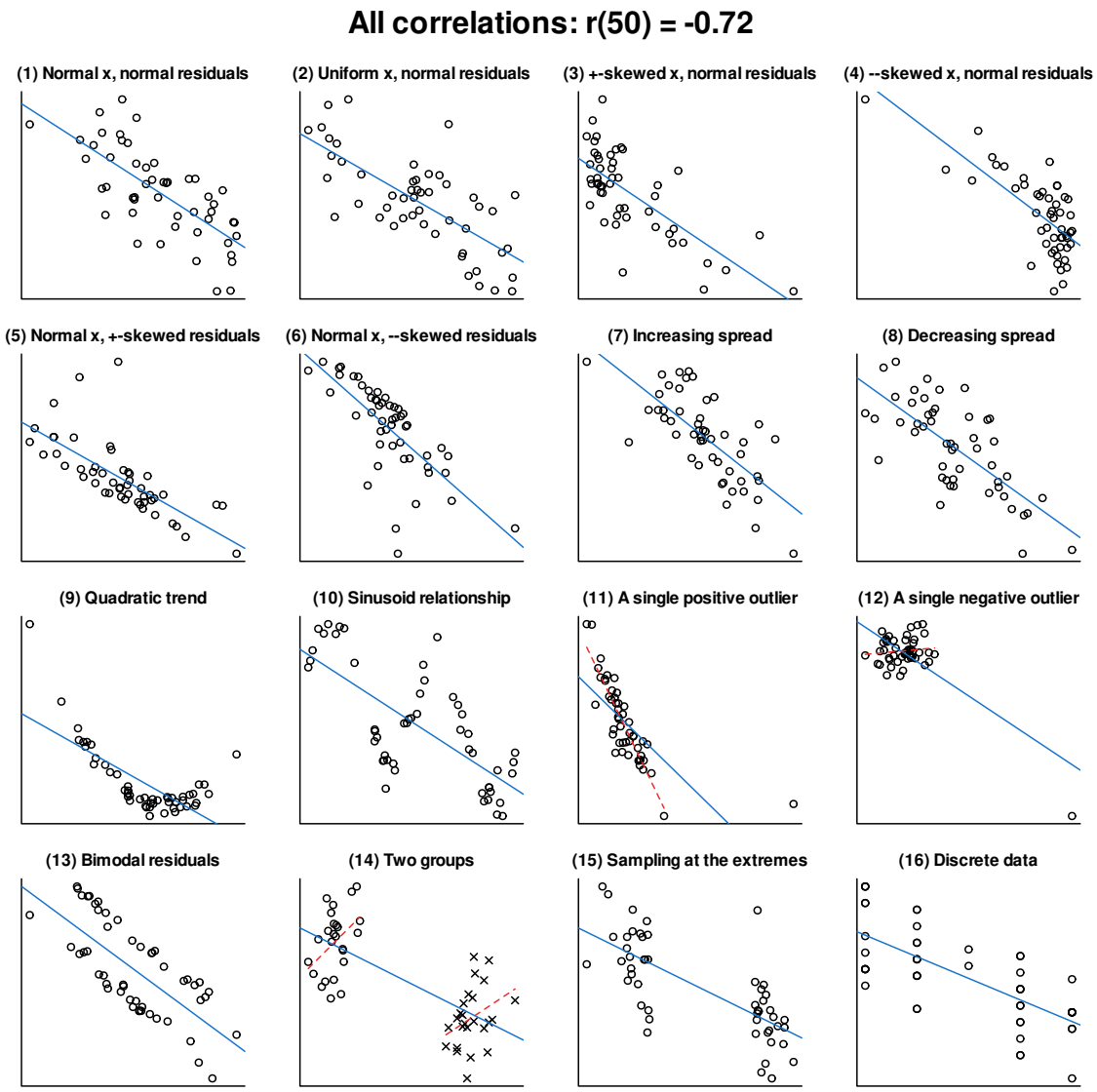


Figure 8.14: All sixteen relationships display a correlation of -0.72, yet they look quite different.

Remark 8.22 (Spearman's ρ)**.** To calculate Spearman's ρ, you first express the observations as ranks, that is, you order the data from smallest to largest and note the position of each data point. You then simply compute the Pearson correlation for the ranks instead of the raw values:

```
cor(rank(d$AOA), rank(d$GJT))

[1] -0.7887659

# simpler:
cor(d$AOA, d$GJT, method = "spearman")

[1] -0.7887659
```

Spearman's ρ can be useful when the relationship between two variables is monotonic but not linear (monotonic meaning generally increasing or generally decreasing, rather than first increasing and then decreasing) or when an outlier distorts the overall picture but cannot be removed from the dataset for some reason.

If Spearman's $\rho = 1$, the relationship is perfectly monotonically increasing (higher values of x always correspond to higher values of y); if $\rho = -1$, the relationship is perfectly monotonically decreasing; and if $\rho = 0$, there is no monotonic association in the data. Note that ρ answers a different question from r: *How perfect is the monotonic relationship?* versus *How perfect is the linear relationship?* ♢

***Remark 8.23** (Kendall's τ)**.** Kendall's τ is used fairly rarely. The calculation is conceptually straightforward (see Noether, 1981), but can look complicated in R code, so I will summarise it here in words:

1. Compare each x value with every other x value and note whether the first is larger or smaller than the second. For example, if the x values are 5, 3, 8, and 7, the comparisons are:
   - 5 vs 3: larger,
   - 5 vs 8: smaller,
   - 5 vs 7: smaller,
   - 3 vs 8: smaller,
   - 3 vs 7: smaller,
   - 8 vs 7: larger.
2. Compare each y value with every other y value. For y values 8, -2, -4, -3, we obtain: larger, larger, larger, larger, larger, smaller.
3. Count how many comparisons go in the same direction:
   - 1st comparison: larger–larger: concordant,
   - 2nd comparison: smaller–larger: discordant,

- 3rd comparison: smaller–larger: discordant,
- 4th comparison: smaller–larger: discordant,
- 5th comparison: smaller–larger: discordant,
- 6th comparison: larger–smaller: discordant.

So there is 1 concordant comparison and 5 discordant comparisons.

4. Estimate Kendall's τ as follows:

$$\hat{\tau} = \frac{\text{number of concordant} - \text{number of discordant}}{\text{total number of comparisons}}.$$

The hat indicates that this is a sample-based estimate. Thus:

$$\hat{\tau} = \frac{1-5}{6} = -0.67.$$

If some x or y values are tied, the calculation is adjusted to account for these ties.

```
# For our small example
x <- c(5, 3, 8, 7)
y <- c(8, -2, -4, -3)
cor(x, y, method = "kendall")

[1] -0.6666667

# For the AOA-GJT data
cor(d$AOA, d$GJT, method = "kendall")

[1] -0.6035606
```

Kendall's $\hat{\tau}$ estimates the difference between the proportion of concordant comparisons and the proportion of discordant comparisons. I personally find this interpretation a bit awkward, but there is a simpler way: Take any two (x, y) pairs—let's call them (x_1, y_1) and (x_2, y_2). If x_2 is larger than x_1, then it is $\frac{1+\hat{\tau}}{1-\hat{\tau}}$ times more likely that y_2 is also larger than y_1 than that it is smaller. For the AOA–GJT data: if one person has a higher AOA than another, then it is $\frac{1+(-0.60)}{1-(-0.60)} = 0.25$ times more likely that they also have a higher GJT than a lower one. In other words, it is four times more likely that they have a lower GJT than a higher one. ♢

In practice, the use of Spearman's ρ and Kendall's τ is rather limited. Rather than automatically turning to ρ or τ when a relationship is nonlinear or when an outlier is suspected, I would argue that it is usually better to consider whether (a) you are genuinely interested the strength of the relationship rather than its form, (b) one or both variables can be meaningfully transformed to yield a more linear relationship, or (c) the suspected outlier is actually a legitimate data point at all.

Remark 8.24 ('Strong' and 'weak' correlations)**.** Correlation coefficients are often—without consideration of the research question or context—classified as small or weak,

medium, or large or strong. I find this rather unhelpful, which is why I do not reproduce such classifications here. Personally, I believe correlation coefficients are overused. I explain why in these blog posts:

- *Why I don't like standardised effect sizes* (5 February 2015)
- *More on why I don't like standardised effect sizes* (16 March 2015)
- *Abandoning standardised effect sizes and opening up other roads to power* (14 July 2017)

See also Baguley (2009). ♢

Since correlation coefficients are calculated from samples, they are also affected by sampling error: different samples drawn from the same population will yield correlation coefficients that differ to some extent. The uncertainty or variability of a correlation coefficient based on a sample can be expressed as a confidence interval. Here, we discuss both a bootstrap approach and a method based on t distributions.

Remark 8.25 (Confidence interval using the bootstrap)**.** The procedure is analogous to the bootstrap described in Chapter 6: new bootstrap samples are drawn from the original sample, and for each bootstrap sample the statistic of interest (here: the correlation between AOA and GJT) is calculated. The spread of the estimates across the bootstrap samples gives us an indication of the variability of the correlation coefficient in samples of this size.

```
n_bootstraps <- 20000
bootstraps <- vector(length = n_bootstraps)

for (i in 1:n_bootstraps) {
  # sampling with replacement from the observed sample
  bootstrap_sample <- d[sample(1:nrow(d), replace = TRUE), ]

  # calculate and store correlation in the bootstrap sample
  bootstraps[[i]] <- cor(bootstrap_sample$GJT, bootstrap_sample$AOA)
}
```

```
# histogram of bootstrap estimates (not shown)
hist(bootstraps, breaks = 50)
```

As the histogram is not normally distributed, we will forego calculating a standard error. However, using the percentiles of the distribution, it is certainly possible to construct a confidence interval. Here, we calculate a 90% confidence interval:

```
quantile(bootstraps, probs = c(0.05, 0.95))
```

```
        5%        95%
-0.8648265 -0.7291892
```

♢

Remark 8.26 (Confidence interval using t distributions)**.** I will not reproduce the formula for constructing a confidence interval for a correlation coefficient using a t distribution, as it is both intimidating and conceptually adds little. It is based on the assumption that the population from which the two variables were drawn is bivariately normal. In essence, this means that the relationship between both variables is linear and both variables are normally distributed.

If these assumptions are plausible, a 90% confidence interval can be calculated using the `cor.test()` function:

```
cor.test(d$AOA, d$GJT, conf.level = 0.9)$conf.int

[1] -0.8614924 -0.7230816
attr(,"conf.level")
[1] 0.9
```

Using this method, we obtain a 90% confidence interval of approximately $[-0.86, -0.72]$. This differs only slightly from the interval we computed via the bootstrap. However, with the bootstrap we did not assume that the population is bivariately normal. In particular, for smaller samples, the results of the bootstrap and t-based methods can diverge substantially. If the assumptions hold, the t-method is undoubtedly superior, but these assumptions are difficult to verify in small samples.

The performance of the bootstrap can be improved somewhat by alternative interval constructions (see DiCiccio & Efron, 1996), though these methods are more complex and less intuitive. For our purposes, it suffices to reiterate the warning from Hesterberg (2015): "Bootstrapping does not overcome the weakness of small samples as a basis for inference." (p. 379) ♢

Exercise 8.27. There is now a substantial body of literature examining whether bilingualism confers cognitive advantages. One cognitive **construct** often mentioned in this context is *cognitive control*. This construct can only be measured indirectly, namely through cognitive tests: a person's performance on a test is not their cognitive control per se, but merely an imperfect **indicator** of it. If different indicators of cognitive control are strongly correlated, it is more likely that findings based on one indicator will generalise to other indicators.

The file `poarch2018.csv` contains data from two cognitive tests that are assumed to be indicators of cognitive control: the Flanker task (Eriksen & Eriksen, 1974) and the Simon task (Simon, 1969). In both tasks, participants must sometimes ignore irrelevant information. The data come from a small study by Poarch et al. (2019), in which participants completed both tasks. The results represent how much faster participants responded when the irrelevant information was 'congruent' with the relevant information compared to when it was 'incongruent'. Values are expressed in stimuli per second; for example, a value of 0.5 means the participant completed 5 more stimuli per 10 seconds in the congruent condition than in the incongruent condition.

1. Plot the relationship between the variables `Flanker` and `Simon`.

2. Calculate the Pearson correlation coefficient, if you consider it appropriate.
3. If applicable, calculate a 90% confidence interval, both using the bootstrap and the t distribution.
4. Briefly Summarise your findings. ♢

***Remark 8.28** (Reconstructing confidence intervals)**.** When a confidence interval around a correlation coefficient is constructed using the t distribution method, the result depends on just three factors:

- the desired confidence level (50%, 80%, 87%, etc.);
- the correlation coefficient itself;
- the number of observed pairs.

If you know the correlation coefficient and the sample size, you can calculate the t distribution-based confidence interval yourself. I find this useful when a study does not report a confidence interval for r. The function `r.test()` from the `psych` package makes this easy, although it only calculates 95% confidence intervals. You need to install the `psych` package first.

```
psych::r.test(r12 = -0.80, n = 76)

Correlation tests
Call:psych::r.test(n = 76, r12 = -0.8)
Test of significance of a  correlation
 t value -11.47    with probability < 0
 and confidence interval  -0.87 -0.7
```

By the way, when you use the notation `psych::r.test()`, you don't need to load the `psych` package with `library(psych)`. This is useful if you only need a single function from a package.

The 95% confidence interval for a correlation coefficient of $r = -0.80$ in a sample of 76 data points is therefore $[-0.87, -0.70]$.

If you prefer to calculate 80% or 90% confidence intervals for correlation coefficients, you can use the function below. It generates a dataset with the desired characteristics using the `plot_r()` function from the `cannonball` package and then calculates the confidence interval around the correlation coefficient in that dataset.

```
ci_r <- function(r, n, conf_level = 0.90) {
  dat <- cannonball::plot_r(r = r, n = n, showdata = 1, plot = FALSE)
  ci <- cor.test(dat$x, dat$y, conf.level = conf_level)$conf.int[1:2]
  ci
}

# 95% confidence interval
ci_r(r = -0.80, n = 76, conf_level = 0.95)
```

```
[1] -0.8687618 -0.7009755

# 80% confidence interval
ci_r(r = -0.80, n = 76, conf_level = 0.80)

[1] -0.8478924 -0.7391568

# 50% confidence interval
ci_r(r = -0.80, n = 76, conf_level = 0.50)

[1] -0.8266792 -0.7697318
```

Bootstrap-based confidence intervals can only be reconstructed from the data itself. ♢

Chapter 9

Group differences

The general linear model can be used to model differences between two or more groups with respect to some continuous outcome variable. We'll first take a look at how differences between two groups can be modelled. Then we'll deal with differences between three or more groups.

9.1 Differences between two groups

Our first example in this chapter stems from the field of social psychology.[1] Caruso et al. (2013) reported that American participants agree more strongly with statements that justify the US social system if they are reminded of money. Their participants responded to eight statements such as *Everyone has a fair shot at wealth and happiness* on a 7-point Likert scale. The eight responses per participant were averaged and submitted to analysis. For half of the participants, a faded but still visible dollar note was shown on the computer screen as they filled out the questionnaire; for the other half, this image was completely blurred. The authors reported that the mean system justification scores tended to be higher when a dollar note was visible than when it was blurred. Klein et al. (2014) attempted to replicate this finding in a 36 new samples. We'll work with the replication data from one such new sample:

```
d <- read_csv(here("data", "Klein2014_money_abington.csv"))
```

A sensible default choice when plotting a group comparison on a continuous outcome is to use **boxplots**. Figure 9.1 shows an example of a boxplot of some set of observations $\boldsymbol{X}$. The median score is highlighted by a thick line. The 25th and 75th percentiles are highlighted by thinner lines and together form a box. The difference between the 75th and 25th percentile (also known as the third and the first quartile) is called the **interquartile range** (IQR). The data points that don't fall in the box, that is, the data points that don't make up the middle 50% of the data, are represented by a line protruding from the upper part of the box and by a line protruding from the lower part of the box.

[1] Studies with group comparisons in language-related fields either seem to be more complex than *just* a group comparison or don't share their data. But if you know of a suitable dataset from the language sciences, please let me know!

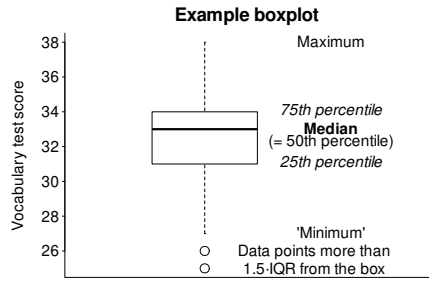


Figure 9.1: Example of a boxplot.

However, data points whose distance to the box exceeds 1.5 times the inter-quartile range are plotted separately. If such data points exist, the lines protruding from the box only extend up to the lowest / highest data point whose distance to the box is lower than 1.5 times the IQR.

Figure 9.2 shows side-by-side boxplots of the Klein et al. data. But boxplots sometimes deceive Boxplots sometimes deceive, and for this reason, I prefer to add the individual data points to the plots, too. Doing so helps both us and our readers gauge what the data actually look like; see Figure 9.3; also see Weissgerber et al. (2015).

```
# standard boxplots
p_boxplot <- ggplot(data = d,
                    aes(x = MoneyGroup,
                        y = Sysjust)) +
  geom_boxplot() +
  xlab("condition") +
  ylab("system justification score")
p_boxplot
```

```
# boxplots with individual data points added
p_boxplotdeluxe <- ggplot(data = d,
                          aes(x = MoneyGroup,
                              y = Sysjust)) +
  # don't plot outliers twice
  geom_boxplot(outlier.shape = NA) +
  # add some random noise to the x position of the points to reduce overlap
  geom_point(shape = 1,
             position = position_jitter(width = 0.2, height = 0)) +
  xlab("bank note") +
  scale_x_discrete(labels = c("without", "with")) +
  ylab("system justification score")
p_boxplotdeluxe
```

The setting `outlier.shape = NA` in the `geom_boxplot()` command suppresses the drawing of potential outliers in the boxplot. If we didn't do this, we would be plotting such

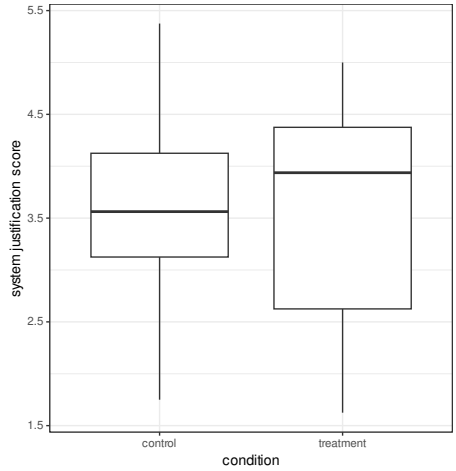


Figure 9.2: Comparison of the system justification scores in both conditions in Klein et al.'s (2014) replication of Caruso et al. (2013, Experiment 1). Data from the Abington sample.

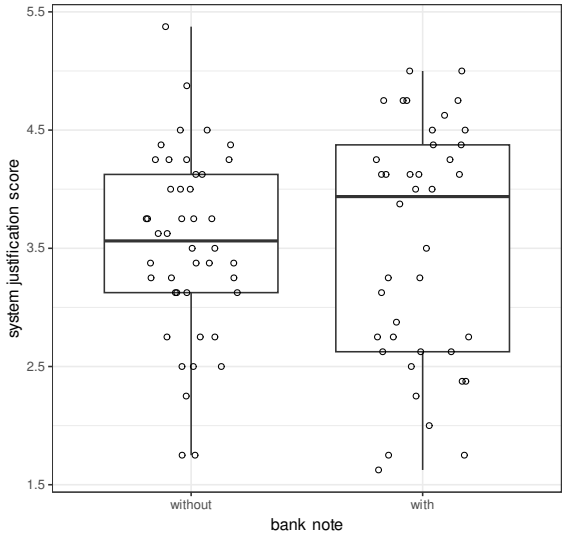


Figure 9.3: The same data but with the individual data points overlaid.

potential outliers twice: once as part of the boxplot, and once as individual data points. (There are no such outliers in this dataset, though.) Moreover, the data points are jittered horizontally so that they don't overlap as much.

Let's also construct a table with the basic descriptive statistics for both of the experiment's conditions.

```
d |>
  group_by(MoneyGroup) |>
  summarise(n = n(),
            mean = mean(Sysjust),
            median = median(Sysjust),
            stdev = sd(Sysjust))

# A tibble: 2 x 5
  MoneyGroup     n  mean median stdev
  <chr>      <int> <dbl>  <dbl> <dbl>
1 control       44  3.53   3.56 0.781
2 treatment     40  3.53   3.94 1.02
```

Let's turn to the matter of modelling these data in a general linear model. The basic equation is the same as the simple regression equation in Chapter 8:

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \tag{9.1}$$

for $i = 1, 2, \ldots, n$. This time, x_i is a **dummy variable** that encodes whether the i-th participant is part of one group or the other. There are two common methods for encoding group membership using dummy variables when there are just two groups: treatment coding and sum coding.

In **treatment coding**, we designate one group as the 'treatment group' and the other as the 'control group'. Then, we set $x_i = 1$ if the i-th participant is part of the treatment group and $x_i = 0$ if the i-th participant is part of the control group. In our current example, the treatment group has already been identified, and we just need to add another variable to the tibble that encodes this information numerically:

```
d$n.MoneyGroup <- ifelse(d$MoneyGroup == "treatment", yes = 1, no = 0)
```

Now, we can use the newly created dummy variable like the finer-grained predictor in Chapter 8:

```
money.lm <- lm(Sysjust ~ n.MoneyGroup, data = d)
coef(money.lm)

 (Intercept) n.MoneyGroup
   3.5340909   -0.0059659
```

Uncertainty estimates can be obtained just like in Chapter 8. Here, we'll skip straight to the standard errors and confidence intervals based on the assumption that the ε_i were sampled i.i.d. from a normal distribution. But there's an exercise towards the end of

this chapter where you verify the results obtained using a bootstrap that doesn't assume identical, normal error distributions.

```
summary(money.lm)$coefficients

               Estimate Std. Error   t value   Pr(>|t|)
(Intercept)   3.5340909    0.13633 25.923055 2.8981e-41
n.MoneyGroup -0.0059659    0.19756 -0.030198 9.7598e-01

confint(money.lm)

                2.5 %  97.5 %
(Intercept)   3.26289 3.80529
n.MoneyGroup -0.39898 0.38705
```

What do these parameter estimates actually refer to? Inspecting Equation 9.1, we notice that if $x_i = 0$, then

$$y_i = \widehat{\beta}_0 + \widehat{\varepsilon}_i,$$

or, put differently,

$$\widehat{y}_i = \widehat{\beta}_0.$$

That is, the estimated intercept term shows us the estimated conditional mean for the group coded as 0, which should come as no surprise given what we've seen in the previous chapter. From the previous chapter, we also know that $\widehat{\beta}_0 + \widehat{\beta}_1$ gives us the estimated conditional mean for the group coded as 1. The estimated β_1 parameter, then, tells us how much higher the estimated mean for the 1-group is compared to the 0-group. In our example,

$$y_i = 3.53 - 0.006x_i + \widehat{\varepsilon}_i.$$

So from the output above, we can glean that the mean of the control group is 3.53 ± 0.14, whereas the estimated mean difference between the treatment and the control groups is -0.006 ± 0.20. It is usually the latter estimate that is relevant.

Exercise 9.1. How would the parameter estimates change if we coded the treatment group as 0 and the control group as 1? Try to answer this question without using the computer. ♢

The second commonly used method for coding dummy variables is **sum coding**. In sum coding, the value of the dummy variable is $+0.5$ for the participants in one condition and -0.5 for the participants in the other conditions. (Alternatively, ± 1 is used.)

```
d$n.MoneyGroup2 <- ifelse(d$MoneyGroup == "treatment", yes = 0.5, no = -0.5)
money.lm2 <- lm(Sysjust ~ n.MoneyGroup2, data = d)
summary(money.lm2)$coefficients

                Estimate Std. Error   t value   Pr(>|t|)
(Intercept)    3.5311080    0.09878 35.747017 8.9152e-52
n.MoneyGroup2 -0.0059659    0.19756 -0.030198 9.7598e-01

confint(money.lm2)
```

```
                  2.5 %  97.5 %
(Intercept)    3.33460 3.72761
n.MoneyGroup2 -0.39898 0.38705
```

We can use these estimates to compute the conditional means for the treatment and control groups. The estimated regression equation now is given by

$$y_i = 3.53 - 0.006x_i + \widehat{\varepsilon}_i.$$

Due to rounding, this is the same equation as before, but the actual estimate for the intercept is slightly different between the two models. To obtain the estimate for the conditional mean for the treatment group, we use $x_i = 0.5$; to obtain the estimated conditional mean for the control group, we use $x_i = -0.5$:

$$\widehat{y}_{\text{treatment}} = 3.53 - 0.006 \cdot 0.5$$

and

$$\widehat{y}_{\text{control}} = 3.53 + 0.006 \cdot 0.5.$$

The difference between these estimates is exactly the parameter estimate for the slope:

$$\widehat{y}_{\text{treatment}} - \widehat{y}_{\text{control}} = (3.53 - 0.006 \cdot 0.5) - (3.53 + 0.006 \cdot 0.5) = -0.006.$$

As for the interpretation of the intercept estimate, consider the following:

$$\begin{aligned}
\widehat{\beta}_0 &= \frac{\widehat{\beta}_0 + \widehat{\beta}_0}{2} \\
&= \frac{(\widehat{\beta}_0 + \widehat{\beta}_1 \cdot 0.5) + (\widehat{\beta}_0 - \widehat{\beta}_1 \cdot 0.5)}{2} \\
&= \frac{\widehat{y}_{\text{treatment}} + \widehat{y}_{\text{control}}}{2}.
\end{aligned}$$

That is, when using sum-coding, the estimated intercept corresponds to the average of the conditional means for both groups, that is, to the **grand mean**.

Tip 9.2. If you want to use treatment coding, you don't actually have to code the dummy variables yourself—R can take care of this for you. However, by default, R codes such variables in alphabetical order. For instance, if your two groups are called 'French' and 'German', the French group would be coded as 0 and would be incorporated into the intercept estimate. But if your two groups are called 'Französisch' and 'Deutsch', the German (Deutsch) group would be coded as 0. I recommend you code your dummy variables by hand, so you know what's going on. ♢

Exercise 9.3. How would the parameter estimates change if we coded the treatment group as 1 and the control group as -1? How could you interpret the slope estimate? Try to answer these questions without using the computer. ♢

Exercise 9.4. One of the other findings that Klein et al. (2014) sought to replicate was the gambler's fallacy as studied by Oppenheimer et al. (2009). Klein et al. (2014) summarise this experiment as follows:

> "Oppenheimer & Monin (2009) investigated whether the rarity of an independent, chance observation influenced beliefs about what occurred before that event. Participants imagined that they saw a man rolling dice in a casino. In one condition, participants imagined witnessing three dice being rolled and all came up 6's. In a second condition two came up 6's and one came up 3. In a third condition, two dice were rolled and both came up 6's. All participants then estimated, in an open-ended format, how many times the man had rolled the dice before they entered the room to watch him. Participants estimated that the man rolled dice more times when they had seen him roll three 6's than when they had seen him roll two 6's or two 6's and a 3. For the replication, the condition in which the man rolls two 6's was removed leaving two conditions."

You can find the data for this replication study in `Klein2014_gambler.csv`. Analyse these data in light of the research question, but only consider the data from the `ufl` sample. Summarise your findings in two to three sentences. ♢

9.2 Differences between more than two groups

In Vanhove (2017a), I investigated how German gender assignment by Belgian speakers of Dutch is affected by their native dialect. For example, I wanted to find out if speakers of Belgian Dutch dialects would more often pick the German masculine article *der* for *Knie* ('knee') if, in their own Dutch dialect, *knie* was masculine rather than feminine. (Belgian Dutch dialects differ somewhat in terms of gender assignment, though speakers don't seem to be really aware of this. German *Knie* is neuter, by the way.) It turned out that the German gender assignments of the informants hardy betrayed any influence of their own dialect at all. In a follow-up study (Vanhove, 2019), I tested my hunch that speakers of a Belgian Dutch dialect don't rely on their own dialect when assigning gender to German nouns because they lack metalinguistic knowledge about grammatical gender in their own dialect. To this end, I devised a small experiment with three conditions:

- Strategy: In the first condition, participants were furnished with a simple strategy for figuring out the grammatical gender of nouns in their own dialect.
- Information: In the second condition, participants were told that their own dialect, like German but unlike Standard Dutch, makes a three-way adnominal gender distinction. They weren't told how they could identify the grammatical gender of nouns in their own dialect, however.
- No information: Participants in this condition were provided with accurate information about some grammatical aspect of their dialect that was irrelevant to the task at hand.

Then, the participants assigned German gender to 29 German nouns with Dutch cognates, and I tallied how often their German gender assignments were congruent with the nouns' gender in the participants' native dialect. I expected that participants in the

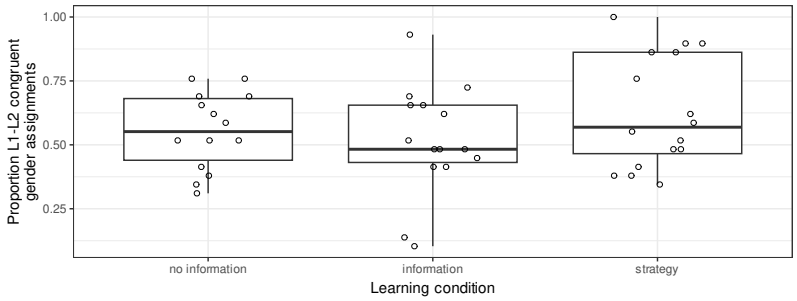


Figure 9.4: Proportion of L1-congruent gender assignments in L2 German by experimental condition in Vanhove (2019).

strategy condition would provide more congruent responses than those in the other conditions.

Let's read in the data and draw some boxplots (Figure 9.4). Note the use of the `limits` parameter in the `scale_x_discrete()` call to change the order of the boxplots to something more meaningful than the default alphabetical one.

```
d <- read_csv(here("data", "Vanhove2018.csv"))
ggplot(d,
       aes(x = Condition,
           y = ProportionCongruent)) +
  geom_boxplot(outlier.shape = NA) +
  geom_point(shape = 1,
             position = position_jitter(width = 0.2, height = 0)) +
  scale_x_discrete(limits = c("no information", "information", "strategy")) +
  xlab("Learning condition") +
  ylab("Proportion L1-L2 congruent\ngender assignments")
```

A descriptive numerical summary is obtained easily enough:

```
d |>
  group_by(Condition) |>
  summarise(N = n(),
            Mean = mean(ProportionCongruent),
            Median = median(ProportionCongruent),
            StDev = sd(ProportionCongruent))

# A tibble: 3 x 5
  Condition          N  Mean Median StDev
  <chr>          <int> <dbl>  <dbl> <dbl>
1 information       15 0.517  0.483 0.213
2 no information    14 0.554  0.552 0.150
3 strategy          16 0.627  0.569 0.219
```

In the previous section, we recoded a variable with two levels (treatment and control) as a binary dummy variable. When we have a variable with k levels, we need $k-1$ binary dummy variables to code all conditions unambiguously. In our present example, we will use treatment coding to identify both whether participants were assigned to the strategy condition and whether they were assigned to the information condition; if both dummy variables read 0, then the participant was assigned to the no information condition:

```
d$Strategy <- ifelse(d$Condition == "strategy", 1, 0)
d$Information <- ifelse(d$Condition == "information", 1, 0)

# quick check
xtabs(~ Strategy + Condition, data = d)

        Condition
Strategy information no information strategy
       0          15             14        0
       1           0              0       16

xtabs(~ Information + Condition, data = d)

           Condition
Information information no information strategy
          0           0             14       16
          1          15              0        0
```

The general linear model can deal with several predictors: The principles from the previous chapters carry over, just with $d = 3$ β parameters rather than $d \in \{1, 2\}$. So we can add both dummy variables to the model. The resulting model equation looks like this:

$$y_i = \beta_0 + \beta_1 \cdot x_{1,i} + \beta_2 \cdot x_{2,i} + \varepsilon_i,$$

for $i = 1, \ldots, n$. $x_{1,i}$ represents the value of the i-th participant on the first dummy variable; $x_{2,i}$ represents their value on the second dummy variable.

```
mod.lm <- lm(ProportionCongruent ~ Information + Strategy, data = d)
mod.lm


Call:
lm(formula = ProportionCongruent ~ Information + Strategy, data = d)

Coefficients:
(Intercept)  Information     Strategy
     0.5542      -0.0369       0.0730
```

The interpretation of these parameter estimates is analogous to what we've discussed in the previous section. The intercept estimate is the conditional mean for the group that was coded as 0 on both dummy variables, i.e., the no information condition. The estimate for information is the difference between the estimated conditional means for

the information and the no information conditions; the estimate for strategy is the difference between the estimated conditional means for the strategy and the no information conditions. You can verify this by reconstructing the condition means in the descriptive statistics using the model output.

We can estimate the uncertainty about these parameter estimates using bootstrap methods (see the exercise below). If we're comfortable assuming that the residuals were all drawn from the same normal distribution, we can use `summary()` as before to obtain standard errors:

```
summary(mod.lm)$coefficients
```

```
             Estimate Std. Error  t value   Pr(>|t|)
(Intercept)  0.554187   0.053006 10.45526 2.9207e-13
Information -0.036946   0.073701 -0.50129 6.1878e-01
Strategy     0.072968   0.072581  1.00533 3.2049e-01
```

Confidence intervals can be computed using `confint()`. The confidence interval about the intercept estimate isn't of too much interest here, but it is computed by default. That doesn't mean you have to report it in your papers, though. (Relevance!)

```
confint(mod.lm, level = 0.90)
```

```
                 5 %     95 %
(Intercept)  0.46503 0.643340
Information -0.16091 0.087016
Strategy    -0.04911 0.195046
```

Tip 9.5 (Obtaining directly interpretable regression coefficients)**.** While we used treatment coding here, other coding systems exist, and these may be more directly useful for you, depending on what you want to find out. For instance, we could code the dummy variables in such a way that $\widehat{\beta}_2$ estimates the difference between the conditional means for the third and second condition rather than between those for the third and first condition. Schad et al. (2020) explain how this can be accomplished; also see the entry on my blog *Tutorial: Obtaining directly interpretable regression coefficients by recoding categorical predictors* (5 May 2020). This technique is extremely useful as it often enables you to read off the answers to your research questions directly from the model output, as opposed to your having to piece together an answer based on several parameter estimates. ♢

Tip 9.6. *Know what your parameter estimates refer to.* It is vital that you know what the numbers in the output of your model literally mean before you try to interpret them in terms of your subject matter. ♢

***Exercise 9.7** (Recoding categorical predictors)**.** Read Example 1 on `https://janhove.github.io/posts/2020-05-05-contrast-coding/`. Now recode the conditions in the Vanhove (2017a) data in such a way that the β_0 parameter expresses the grand mean of `ProportionCongruent` (that is, the mean of the three condition means), the β_1 parameter expresses the difference between the mean in the `no information` condition and the mean of the means of the other two conditions, and the β_2 parameter expresses the

difference between the mean of the `information` condition and that of the `strategy` condition. Fit the model and verify if the estimated parameters are correct by comparing them to the condition means. ♢

***Exercise 9.8** (Semiparametric bootstrap without i.i.d. assumption)**.** In the analyses in this chapter, we estimated standard errors and constructed confidence intervals based on the assumption that the residuals were i.i.d. normal. We can check whether we obtain similar results when making different assumptions. To that end, we can run a semi-parametric bootstrap in which the bootstrapped residuals in a given condition are only sampled from that condition. This way, we don't assume that the residuals are drawn from a normal distribution, and we acknowledge the possibility that the residuals in different conditions are drawn from different distributions.

The code below is similar to that of the previous semiparametric bootstraps; the only thing that was added is the `group_by(Condition)` call in the *for*-loop. This splits up the dataset into groups by `Condition` so that the resampling of the `Residual` values on the next line only happens within each condition. In other words, a residual value in the information condition can only be reassigned to another observation in the information condition, and similarly for the other conditions.

Copy this code (ideally by typing rather than copy-pasting it; you'll learn more this way) and run it. How would the conclusions you draw from this analysis differ from the ones you would draw from the analysis in the main text?

```
# Read in data, fit model
d <- read_csv(here("data", "Vanhove2018.csv"))
d$Strategy <- ifelse(d$Condition == "strategy", 1, 0)
d$Information <- ifelse(d$Condition == "information", 1, 0)
mod.lm <- lm(ProportionCongruent ~ Information + Strategy, data = d)

# Extract predictions, residuals
d$Prediction <- predict(mod.lm)
d$Residual <- resid(mod.lm)

# Semi-parametric bootstrap w/o homoskedasticity assumption
n_bootstrap <- 20000
bs_b <- matrix(nrow = n_bootstrap, ncol = 3)

for (i in 1:n_bootstrap) {
  d <- d |>
    group_by(Condition) |>
    mutate(bs_outcome = Prediction + sample(Residual, replace = TRUE))
  bs_mod <- lm(bs_outcome ~ Information + Strategy, data = d)
  bs_b[i, ] <- coef(bs_mod)
}

apply(bs_b, 2, sd)
```

```
apply(bs_b, 2, quantile, prob = c(0.025, 0.975))
```

Now adapt this code in order to double-check the confidence intervals from the money priming example. ♢

Chapter 10

Differences between differences

Often, researchers aren't so much interested in how one predictor variable relates to some outcome. Rather, they're interested in how the relationship between one predictor and the outcome differs depending on another predictor. That is, they're interested in the **interaction** between two predictors.

Consider Figure 10.1, which shows four examples of what the joint effect of reading experience and word frequency on reading speed could look like. In three cases, the lines are not parallel to each other; in these cases, the effects of reading experience and word frequency on reading speed interact. In one case, the lines do run in parallel, and the effects of reading experience and word frequency on reading speed do not interact; that is, they are **additive**.

10.1 Interactions between two binary predictors

Berthele (2012) ask Swiss teachers in training to rate the academic potential of a German-speaking boy based on a short recording in which he spoke French. About half of the future teachers were told that the boy's name was Luca (a typical Swiss name); the rest were told that the boy's name was Dragan (a name suggesting a Balkan migration background). Moreover, for about half of the participants, the recording contained code-switches from German; for about half, it didn't. The author wanted to find out how the purported name and the presence or absence of code-switches affected the teacher trainees' judgements of the boy's academic potential.

As ever, the first step is to plot the data. We'll try a boxplot first, see Figure 10.2. The code-switching conditions are separated by means of facetting; this is achieved in the last line of the following code snippet.

```
d <- read_csv(here("data", "berthele2012.csv"))
ggplot(d,
       aes(x = Name,
           y = Potential)) +
  geom_boxplot(outlier.shape = NA) +
  geom_point(shape = 1,
```

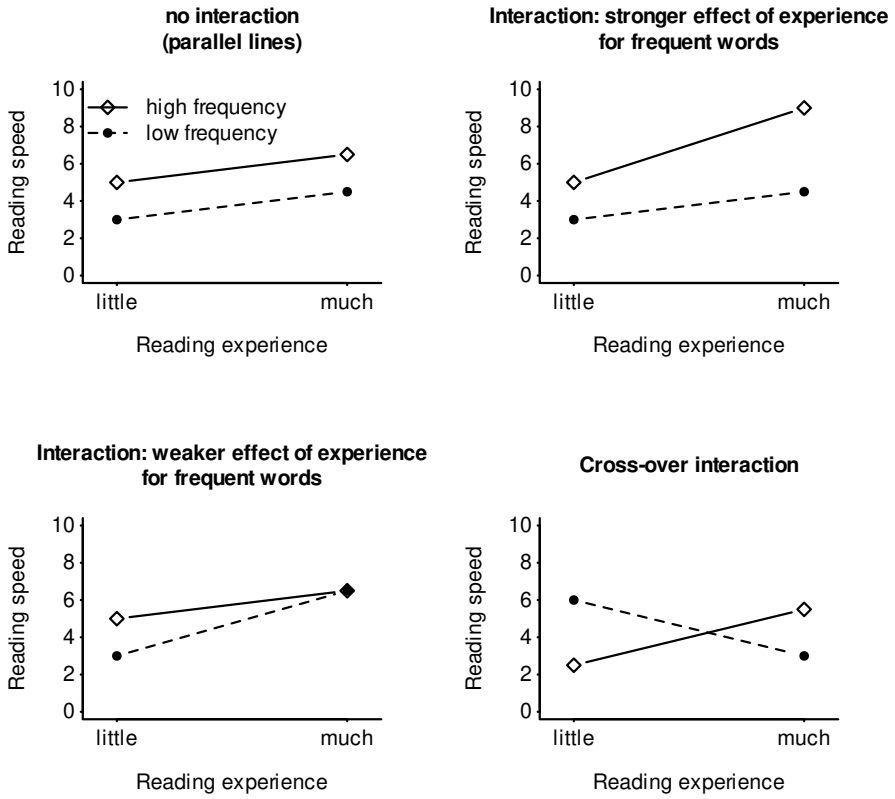


Figure 10.1: If the effects of reading experience and word frequency on reading speed interact, then the effect of reading experience on reading speed differs for different levels of word frequency. Or, equivalently, the effect of word frequency on reading speed differs for different levels of reading experience. This is reflected in the non-parallel lines. (The units on the y axis in this example are arbitrary.)

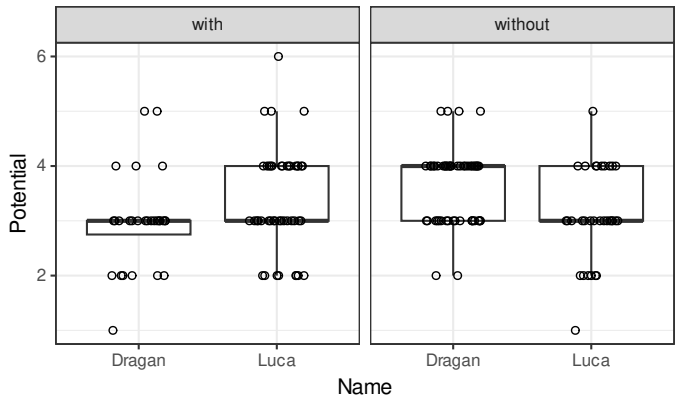


Figure 10.2: A first attempt at plotting the data. The patterns in the data aren't so clear because the data are too coarse for boxplots.

```
             position = position_jitter(width = 0.2, height = 0)) +
  facet_grid(cols = vars(CS))
```

While boxplots are a reasonable default, Figure 10.2 shows that these data are too coarse to plot in this way. Alternatively, we could compute the mean potential rating for each combination of predictor variables and plot these means. But then we wouldn't know how the data underlying these means are distributed; Figure 10.3. Such information is useful both to yourself and to your readers as they help you and them gauge if the means are a relevant indicator of the tendencies in the data.

```
summary_berthele <- d |>
  group_by(Name, CS) |>
  summarise(n = n(),
            MeanRating = mean(Potential),
            StdRating = sd(Potential),
            .groups = "drop")
summary_berthele
```

```
# A tibble: 4 x 5
  Name   CS          n MeanRating StdRating
  <chr>  <chr>   <int>      <dbl>     <dbl>
1 Dragan with       28       2.96     0.881
2 Dragan without    51       3.63     0.692
3 Luca   with       44       3.36     0.942
4 Luca   without    32       3.09     0.856
```

```
ggplot(summary_berthele,
       aes(x = Name,
           y = MeanRating,
           linetype = CS,
```

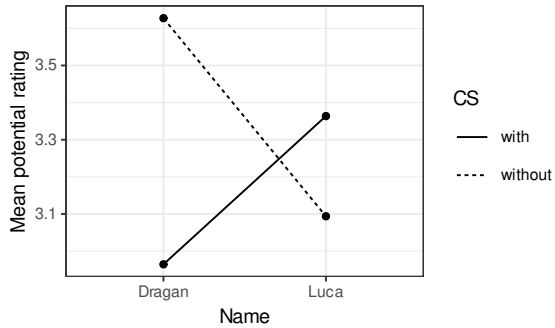


Figure 10.3: The trends in the data are clearer here, but we can't glean the distribution of the data from this plot.

```
          group = CS)) +
  geom_point() +
  geom_line() +
  ylab("Mean potential rating")
```

Luckily, we can have the best of both worlds. With the following commands, we plot the raw data as in Figure 10.2 and then add the mean trends to them; Figure 10.4.

```
ggplot(d,
       aes(x = Name,
           y = Potential)) +
  geom_point(shape = 1, colour = "grey50",
             position = position_jitter(width = 0.2, height = 0)) +
  geom_point(shape = 8, size = 3, colour = "blue",
             data = summary_berthele,      # Data from different dataframe
             aes(x = Name, y = MeanRating)) +
  geom_line(colour = "blue",
            data = summary_berthele,       # Data from different dataframe
            aes(x = Name, y = MeanRating, group = CS)) +
  facet_grid(cols = vars(CS))
```

Tip 10.1 (Try out several plots)**.** For group comparisons, boxplots are often a good choice, but it's often possible to improve on them. Don't hesitate to try out alternative plots.

Incidentally, it can take some time to find a good way to visualise your data. For the last graph, I had to tinker with the colour of the data points as well as with the colour, shape and size of the means. But ultimately, all of this is time well spent. ♢

We now turn our attention to the matter of modelling these data in the general linear model. The plots suggest that the purported name and the presence or absence of code-switches interact: if code-switches are present, 'Dragan's' academic potential is rated worse than 'Luca's', but if they are absent, the order is flipped. We want our model

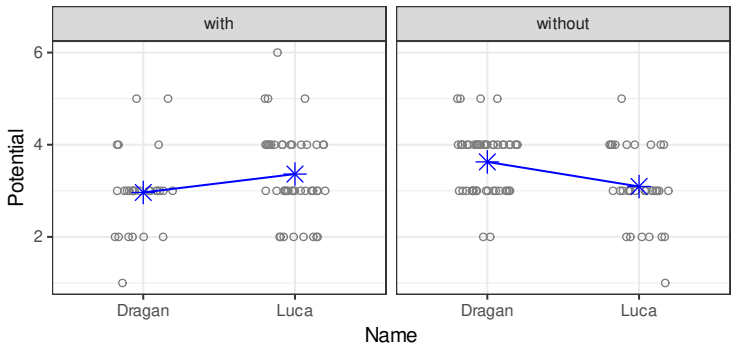


Figure 10.4: The best of both worlds.

to capture this interplay between the two predictors. To that end, we will need four β parameters.

- β_0, the intercept, captures the baseline of the ratings.
- β_1 captures the difference between the cells depending on the purported name (Dragan vs Luca).
- β_2 captures the difference between the cells depending on the presence or absence of code-switches.
- β_3 adjusts β_1 and β_2: How much larger or smaller is the difference between Dragan and Luca if there are code-switches compared to when there are no code-switches? Or, equivalently, how much larger or smaller is the difference between the presence and absence of code-switches depending on whether the purported name is Dragan or Luca?

The model equation is

$$y_i = \beta_0 + \beta_1 \cdot x_{1,i} + \beta_2 \cdot x_{2,i} + \beta_3 \cdot (x_{1,i} \cdot x_{2,i}) + \varepsilon_i,$$

$i = 1, \ldots, 155$. If we want to use treatment coding, the predictors can be coded as follows:

- $x_{1,i}$ indicates whether the i-th participant was told that the boy's name was Dragan (1) or Luca (0).
- $x_{2,i}$ indicates whether the i-th participant rated a recording with (1) or without (0) code-switches.
- Consequently, β_0 represents the average rating by participants who've purportedly heard Luca (0) talk without code-switches (0).

The term $(x_{1,i} \cdot x_{2,i})$ may surprise you, but it does the job: The interaction term is a new variable that is the pointwise product of the two predictor variables. For the four cells in the present design, this new variable takes on two values:

- Luca (0) without code-switches (0): $x_{1,i} \cdot x_{2,i} = 0 \cdot 0 = 0$.

- Luca (0) with code-switches (1): $x_{1,i} \cdot x_{2,i} = 0 \cdot 1 = 0$.
- Dragan (1) without code-switches (0): $x_{1,i} \cdot x_{2,i} = 1 \cdot 0 = 0$.
- Dragan (1) with code-switches (1): $x_{1,i} \cdot x_{2,i} = 1 \cdot 1 = 1$.

Let's add the dummy variables and their product to the tibble:

```
d <- d |>
  mutate(
    Dragan = ifelse(Name == "Dragan", 1, 0),
    WithCS = ifelse(CS == "with", 1, 0),
    DraganWithCS = Dragan * WithCS
  )
```

Now fit a linear model with these variables:

```
potential.lm <- lm(Potential ~ Dragan + WithCS + DraganWithCS, data = d)
potential.lm


Call:
lm(formula = Potential ~ Dragan + WithCS + DraganWithCS, data = d)

Coefficients:
 (Intercept)        Dragan        WithCS  DraganWithCS
       3.094         0.534         0.270        -0.933
```

We can use the estimated coefficients to reconstruct the cell means we computed earlier when drawing the graphs. In the following sums, rounding errors were corrected:

- Not Dragan (so Luca), without code-switches:

$$\widehat{y} = 3.09 + (0.53 \cdot 0) + (0.27 \cdot 0) + (-0.93 \cdot 0) = 3.09.$$

- Dragan, without code-switches:

$$\widehat{y} = 3.09 + (0.53 \cdot 1) + (0.27 \cdot 0) + (-0.93 \cdot 0) = 3.63.$$

- Not Dragan (so Luca), with code-switches:

$$\widehat{y} = 3.09 + (0.53 \cdot 0) + (0.27 \cdot 1) + (-0.93 \cdot 0) = 3.36.$$

- Dragan, with code-switches:

$$\widehat{y} = 3.09 + (0.53 \cdot 1) + (0.27 \cdot 1) + (-0.93 \cdot 1) = 2.96.$$

Since we're using treatment coding, the estimate of 0.53 for `Dragan` ($\widehat{\beta}_1$) *only* pertains to the recording without code-switches (the level coded as 0). I n order to obtain difference

between the estimated conditional means between Luca and Dragan when the recording contains code-switches, you need to include the interaction terms: $0.53 - 0.93 = -0.40$. Similarly, the estimate of 0.27 for `WithCS` ($\widehat{\beta}_2$) only pertains to ratings of 'Luca' (the level coded as 0). In order to obtain the difference between the estimated conditional means between recordings with vs without code-switches for Dragan, you again need to include the interaction term: $0.27 - 0.93 = -0.66$.

In the next exercise, you will learn to interpret the estimated parameters for a model that uses a different coding scheme. See Schad et al. (2020) and my blog post on recoding predictors (5 May 2020) for more details.

Exercise 10.2 (Different coding scheme)**.** Consider the following fictitious experiment and analysis. Eighty participants are randomly assigned to the four cells of a two-by-two design, each cell corresponding to one of the combinations of two binary predictor variables (Variable 1: A vs B, Variable 2: X vs Y). Then, their performance on some task is measured.

For the analysis, the analyst uses sum-coding: `Var1` reads $+0.5$ if the participant was assigned to a B-cell and -0.5 if they were assigned to an A-cell; `Var2` reads $+0.5$ if the participant was assigned to a Y-cell and -0.5 if they were assigned to an X-cell. The `Interaction` term is the pointwise product of `Var1` and `Var2`. The estimated parameter coefficients are as follows:

```
(Intercept)        Var1        Var2 Interaction
    9.65565    -0.92376    -2.69891    -5.53757
```

1. Compute the mean outcome value for each of the four cells.
2. Explain what the estimated `(Intercept)` coefficient represents.
3. Explain what the estimated `Var1` and `Var2` coefficients represent.
4. Explain what the estimated `Interaction` coefficient represents. ♢

Tip 10.3. When in doubt as to the correct literal interpretation of your model's estimated parameters, sit down and do the calculations like in the previous exercise. In fact, also do them if you're *not* in doubt. ♢

We can obtain standard errors and confidence intervals just like before by using the `summary()` and `confint()` commands. These computations are based on the assumption that the errors are i.i.d. normal, but you can always use bootstrapping to check if a different set of assumptions leads to the same conclusion.

```
summary(potential.lm)$coefficients

             Estimate Std. Error t value   Pr(>|t|)
(Intercept)   3.09375    0.14796 20.9090 1.9477e-46
Dragan        0.53370    0.18876  2.8274 5.3289e-03
WithCS        0.26989    0.19446  1.3879 1.6722e-01
DraganWithCS -0.93305    0.27672 -3.3719 9.4836e-04
```

```
confint(potential.lm, level = 0.90)

                   5 %     95 %
(Intercept)   2.848871  3.33863
Dragan        0.221305  0.84610
WithCS       -0.051948  0.59172
DraganWithCS -1.391020 -0.47508
```

Because we used treatment coding here, the estimates for `Dragan` and `WithCS` aren't too relevant. The study's main result is that the presence of code-switches is some 0.93 ± 0.28 points more detrimental to ratings of 'Dragan's' academic potential than it is to ratings of 'Luca's' academic potential. Equivalently, the *absence* of code-switches is some 0.93 ± 0.28 more beneficial to ratings of 'Dragan's' academic potential than it is to ratings of 'Luca's' academic potential.

***Exercise 10.4.** The main finding in Berthele (2012) was the interaction effect of -0.9 ± 0.3 points (90% CI: $[-1.39, -0.48]$). Double-check this estimated standard error and the 90% confidence interval using a semiparametric bootstrap that does not assume that the errors are drawn from the same distribution, as explained in Exercise 9.8 on page 213.

Tip: You can group the data by two variables by using `group_by(variable1, variable2)`. ♢

10.2 Interactions between a binary and a continuous predictor

Sometimes, researchers want to find out if the relationship between a more or less continuous predictor and the outcome differs between groups. This type of research question, too, can be addressed in a general linear model. The idea is the same as in the previous section: Dummy-code the group variable, compute its pointwise product with the continuous predictor, and feed the dummy-coded group variable, the continuous predictor, and their pointwise product into the model.

This case is covered in an exercise. But you should be aware of a common analytical strategy that does *not* work. This doomed strategy is to fit several models in order to gauge the relationship between the continuous predictor and the outcome separately for each group, and to conclude that if this relationship is statistically significant in one group but not in the other, there must be an interaction between the groups and the continuous predictor. Gelman & Stern (2006) and Nieuwenhuis et al. (2011) explain why this is a terrible idea.

Exercise 10.5. The question tackled in this exercise is a bit silly, but I couldn't find a fairly easy dataset where this type of analysis makes more sense.

First use the following code to read in the data from my PhD thesis again.

```
cognates <- read_csv(here("data", "vanhove2014_cognates.csv"))
background <- read_csv(here("data", "vanhove2014_background.csv"))
d <- cognates |>
```

```
  left_join(background)
```

We want to answer the silly 'research' question if the relationship between the participants' English skills (variable `English.Overall`) and their performance on the spoken Swedish cognate recognition task (variable `CorrectSpoken`) differs between men and women. Don't fly blind but plot first:

```
# plot not shown in lecture notes
ggplot(d,
       aes(x = English.Overall,
           y = CorrectSpoken)) +
  geom_point(shape = 1) +
  facet_grid(cols = vars(Sex))
```

What do you suspect is going on here? Fix the problem.

Once you've fixed the problem, add a dummy variable `n.Male` to the data frame/tibble that has the value 1 if the participant is a man and 0 if the participant is a women. Then fit the interaction model like so:

```
int.mod <- lm(CorrectSpoken ~ n.Male * CorrectSpoken, data = d)
```

Explain the literal meaning of each of the four estimated β parameters in this model. Further calculate the model prediction for a man with a score of 1 on the `English.Overall` predictor without using `predict()`. Compare it to the model prediction for a man with a score of 0 on this predictor. ♢

10.3 More complex interactions

It's possible to fit interactions between two continuous predictors; see the blog entry *Interactions between continuous variables* (26 June 2017). It's also possible to fit interactions between three or more predictors. However, it can be difficult to make sense of three-way, four-way, etc. interactions, and I don't have any datasets that call for such an analysis.

10.4 About non-cross-over interactions

The mere fact that a statistical model suggests that two predictor variables interact in their effect on some outcome variable does *not* imply that these predictor variables interact in their effect on the *construct* that this outcome variable represents. This is particularly important to appreciate if the interaction in question is not a cross-over interaction, that is, if the relative order between two groups or conditions doesn't change depending on the other predictor. For instance, if we observe a non-cross-over interaction between reading experience and word frequency on reading speed, this does not imply that reading experience and word frequency have non-additive effects on the cognitive construct that reading speed represents, viz., cognitive effort. The reason is that, for non-cross-over interactions, it is possible to monotonically transform the data so that the interaction disappears. By the same token, if there is *no* interaction, it is typically possible to

Table 10.1: Fictitious data of fuel use for two drivers and two cars.

Car	Driver	Litres per 100 kilometres	Miles per gallon
Car 1	Driver A	6.5	36.2
	Driver B	7.0	33.6
Car 2	Driver A	5.5	42.8
	Driver B	6.0	39.2

monotonically transform the data so that an interaction appears. See Wagenmakers et al. (2012a) for further explanation.

Example 10.6. A straightforward example may help make the problem more concrete. Let's say we want to compare the fuel use of two drivers in two cars. Fuel use is typically expressed in either litres of fuel needed to travel 100 kilometres or in miles that can be travelled using one gallon of fuel; see Table 10.1.

When fuel use is expressed in litres per 100 kilometres, the data in this fictitious example shows two clear effects: Driver B needs half a litre per 100 kilometres more than does Driver A (regardless of the car), and Car 1 needs a litre per 100 kilometres more than does Car 2 (regardless of the driver). So there is no interaction term needed to model fuel use in this example when fuel use is expressed in litres per 100 kilometres.

But when the same fuel use is expressed in miles per gallon, we observe that Driver B can cover 2.6 miles per gallon more than Driver A when driving Car 1, but that the difference is 3.6 miles per gallon for Car 2. That is, the effects of Driver and Car aren't additive when the data are expressed in miles per gallon, and an interaction term would be needed to account for the 1-mile/gallon difference between the differences.

By the same token, a significant non-cross-over interaction in response latencies when they are expressed in milliseconds per item may disappear when the latencies are expressed in items per second or when the latencies are log- or otherwise transformed. ♢

Chapter 11

Several predictors in one model

As we've seen in the previous chapters, the general linear model can accommodate multiple predictor variables at once, yielding **multiple regression models** as opposed to simple regression models. Mathematically and computationally, nothing really changes when adding more predictor variables to the model: the model estimates the β parameters by minimising the sum of squared residuals, and inference is achieved by making assumptions about the distribution of the errors. Conceptually, however, multiple regression models often pose researchers a number of challenges. These mainly relate to the decision whether to include several predictor variables in the model and, relatedly, to how to interpret the estimated regression coefficients. Here, I don't use 'interpret' to refer to subject-matter interpretations, but to more basic statistical interpretations: What do all these numbers literally mean? Clearly, before we can lend subject-matter interpretations to the output of statistical models, we need to understand what they mean literally.

In this chapter, we'll write simulations to illustrate the consequences of including several predictors in a handful of key scenarios. To anticipate the take-home message, the parameter estimates that the general linear model produces first and foremost describe associations in the data. Often, however, researchers want to lend a causal interpretation to these associations. For instance, we're usually not content with finding out that the experimental group outperforms the control group on average—we want to know whether the experimental group outperforms the control group on average *because* they're the experimental group. Importantly, *statistical tools cannot prove claims about causality*. This goes even for so-called causal models. But what we can do is make assumptions about the causal connections between our variables and reason about whether and how we can model these statistically. Whether these causal assumptions are reasonable is a subject-matter issue, not a statistical one.

When discussing assumptions about causal connections, it's useful to draw **directed acyclic graphs** (DAGS). For an introduction to DAGS, see Rohrer (2018) and McElreath (2020). We will use DAGS throughout this chapter to represent causal connections between variables ($x, y, z, \ldots$). Throughout, we're interested in estimating the causal influence that x exerts on y. Our guiding questions are, first, whether this is at all possible, and, second, if so, which variables we should include in the model.

Figure 11.1: In this scenario, z influences both x and y. As a result, z acts as a confounder if we're interested in the causal influence of x on y.

11.1 Accounting for confounders

First, we consider the scenario shown in Figure 11.1. We're interested in the causal influence of x on y. However, it is possible that a third variable, z, influences both x and y. If you want to, you can replace these abstract variable names by something more specific (e.g., x = participation in a content- and language-integrated programme, y = proficiency in the target language, z = the parents' socio-economic status). But I think it's ultimately more useful to embrace the abstraction, as this makes it clearer that the lessons drawn from this scenario are valid more generally.

```
Error in loadNamespace(x):  there is no package called 'shape'
```

Nevertheless, we'll make this scenario a bit more concrete by fleshing out the causal connections between the three variables. To keep this simple, we'll assume that the z variable was drawn from a normal distribution with mean 0 and standard deviation 1:

$$z_i \overset{\text{i.i.d.}}{\sim} \text{Normal}(0, 1^2),$$

$i = 1, \ldots, n$. Nothing hinges on the precise distribution of the z values, though.

Next, we assume that a one-unit increase in the z variable causes an increase of 1.2 units in the x variable. There is, however, some variability in the x variable that is unrelated to z. We express this additional variability in an error term τ, which we assume is also drawn from a normal distribution with mean 0 and standard deviation 1:

$$\begin{aligned} x_i &= 0 + 1.2 \cdot z_i + \tau_i, \\ \tau_i &\overset{\text{i.i.d.}}{\sim} \text{Normal}(0, 1^2), \end{aligned} \tag{11.1}$$

$i = 1, \ldots, n$. The numbers 1.2 in the first line and 1 in the second line were chosen arbitrarily—you can play around with these numbers at home. The 0 in the first line merely means that the mean of the x variable is 0, but little hinges on this.

Additionally, we assume that the y variable is described by the equation below. The influence of x on y is such that a one-unit increase in x leads to a 0.6-unit increase in y. The z variable has a negative causal effect on y, but we don't want to estimate this effect.

Again, the numbers 5.2, 0.6 and -1.3 were chosen arbitrarily.

$$y_i = 5.2 + 0.6 \cdot x_i - 1.3 \cdot z_i + \varepsilon_i, \qquad (11.2)$$
$$\varepsilon_i \overset{\text{i.i.d.}}{\sim} \text{Normal}(0, 1^2),$$

$i = 1, \ldots, n$.

Let's simulate a dataset with 100 observations of these three variables. If we don't specify any further parameters, the `rnorm(n)` call generates n observations from a normal distribution with mean 0 and standard deviation 1:

```
n <- 100
z <- rnorm(n)
x <- 0 + 1.2*z + rnorm(n)
y <- 5.2 + 0.6*x - 1.3*z + rnorm(n)
d <- tibble(y, x, z)
```

In order to visualise the pairwise associations between several continuous variables at once, we can draw a **scatterplot matrix**. The file `scatterplot_matrix.R` in the `functions` directory defines a custom-made function for plotting scatterplot matrices. Let's read it in and plot the simulated data; see Figure 11.2.

```
source(here("functions", "scatterplot_matrix.R"))
scatterplot_matrix(d)
```

We will now compare two strategies for analysing these simulated data. If we were analysing real data, we wouldn't do this—we'd only run the analysis that makes most sense. But in this chapter, we want to use simulated data in order to find out what the most sensible strategy is.

In the first model, we ignore the z variable:

```
dag1.lm1 <- lm(y ~ x, data = d)
summary(dag1.lm1)$coefficients
```

```
               Estimate Std. Error    t value     Pr(>|t|)
(Intercept)  5.30809703 0.13363095 39.7220634 3.167290e-62
x           -0.04188284 0.08624531 -0.4856246 6.283173e-01
```

The resulting parameter estimates are to be interpreted in exactly the same way as explained in Chapter 8:

- If we were to take a large number of observations for which all x values were 0, then, according to the model, we'd expect the mean of their y values to be 5.31 ± 0.13.
- If we were to take a large number of observations for which all x values were 1, then, according to the model, their mean y value is expected to be 0.04 ± 0.09 lower than that of the $x = 0$ group.

This interpretation is absolutely fine! But it does not involve any causal claims.

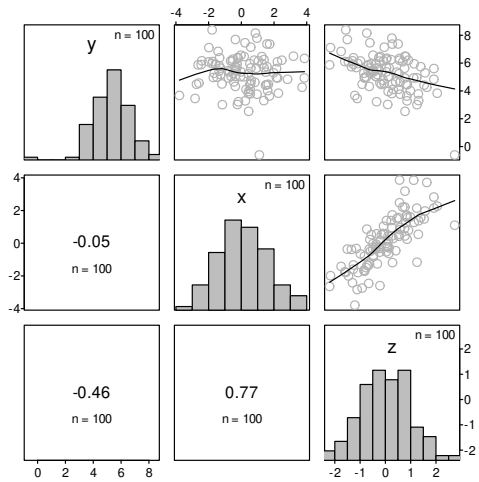


Figure 11.2: A scatterplot matrix of the three simulated variables. The numbers in the lower triangle are Pearson correlation coefficients and the number of data pairs they are based on. The trend lines in the scatterplots in the upper triangle are scatterplot smoothers. For more information, seehttps://janhove.github.io/posts/2019-11-28-scatterplot-matrix/.

In the second model, we include z as a predictor:

```
dag1.lm2 <- lm(y ~ x + z, data = d)
summary(dag1.lm2)$coefficients
```

```
              Estimate Std. Error   t value     Pr(>|t|)
(Intercept)  5.2221331  0.1018272 51.284267 4.387833e-72
x            0.6333719  0.1024041  6.185028 1.474785e-08
z           -1.4042437  0.1638714 -8.569183 1.633004e-13
```

These parameter estimates, too, are to be interpreted as outlined in Chapter 8:

- If we were to take a large number of observations for which all x *and* z values were 0, then, according to the model, we'd expect the mean of their y values to be 5.22 ± 0.10.

- If we were to take a large number of observations for which all x values were 1 and all z values were 0, then, according to the model, their mean y value is expected to be 0.63 ± 0.10 higher than that of the $x = 0, z = 0$ group.

- If we were to take a large number of observations for which all x values were 0 and all z values were 1, then, according to the model, their mean y value is expected to be 1.40 ± 0.16 lower than that of the $x = 0, z = 0$ group.

The second point does *not* contradict the interpretation of the parameter estimates of the first model: Just because parameters with the same name occur in both models (`(Intercept)`, `x`) doesn't mean that these have the same interpretation. (Recall the Greek letter fallacy from Remark 8.6 on page 172.) Specifically, the parameter estimates in the second model can only be interpreted correctly when taking into account the other variable in the model, z. Similarly, to interpret the parameter estimates in the first model, you must not implicitly assume a fixed value for the z variable that was not included in the model.

Now compare the parameter estimates of the second models with those in Equation 11.2 on page 227. The estimated parameters are quite close to the true parameters we used to generate the simulated data. In fact, the discrepancies between the estimates and the true values are purely due to chance. We can verify this by simulating lots of datasets using the same parameters as used in Equations 11.1 and 11.2 and analysing them in the same way as we did here. Below, we define the function `generate_dag1()`, which by default generates 10,000 such datasets, containing 100 observations each. Each dataset is analysed twice: Once without taking the z variable into account, and once including this variable in the model. For each simulated dataset and each model, the estimated `x` parameter is extracted.

```
generate_dag1 <- function(
  n = 100,           # number of observations per dataset
  sims = 10000,      # number of datasets
  z_x = 1.2,         # influence z -> x
  x_y = 0.6,         # influence x -> y
  z_y = -1.3,        # influence z -> y
  baseline_y = 5.2   # baseline y
) {
  est_lm1 <- vector(length = sims)
  est_lm2 <- vector(length = sims)

  for (i in 1:sims) {
    z <- rnorm(n)
    x <- z_x*z + rnorm(n)
    y <- baseline_y + x_y*x + z_y*z + rnorm(n)
    mod.lm1 <- lm(y ~ x)
    mod.lm2 <- lm(y ~ x + z)
    est_lm1[[i]] <- coef(mod.lm1)[[2]]
    est_lm2[[i]] <- coef(mod.lm2)[[2]]
  }

  tibble(`Model 1` = est_lm1,
         `Model 2` = est_lm2)
}
```

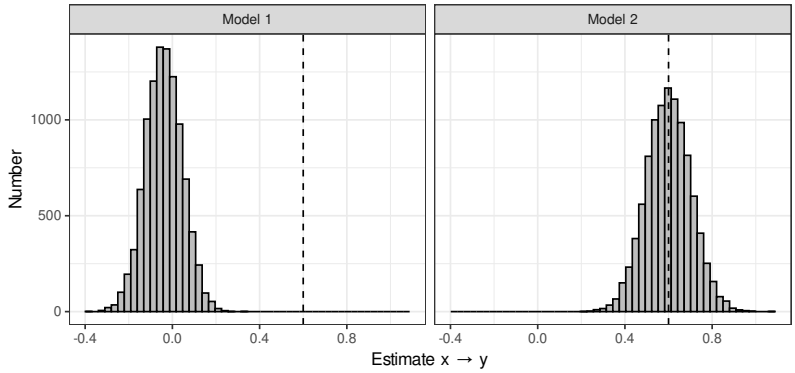


Figure 11.3: Model 1 does not estimate the causal effect of x on y in an unbiased way. Depending on the parameter values chosen in Equations 11.1 and 11.2, the bias could be an overestimate or, like here, an underestimate. Model 2, by contrast, does provide an unbiased estimate of the effect of x on y: On average, the estimated parameter values correspond exactly to the true parameter value.

The code below runs this function using the default settings and then plots the estimated parameters obtained by both models; Figure 11.3.

```
est_dag1 <- generate_dag1()

est_dag1 |>
  pivot_longer(cols = everything(),
               names_to = "Model",
               values_to = "Estimate") |>
  ggplot(aes(x = Estimate)) +
  geom_histogram(bins = 50,
                 fill = "grey", colour = "black") +
  geom_vline(xintercept = 0.6, linetype = "dashed") +
  facet_grid(cols = vars(Model)) +
  xlab("Estimate x → y") +
  ylab("Number")
```

This simulation confirms that the second model yields an unbiased estimate of the causal effect of x on y (0.6), but the first doesn't:

```
apply(est_dag1, 2, mean)

    Model 1     Model 2
-0.03925505  0.59990690
```

The model without the confounding variable (z) isn't wrong. If you want to estimate the average difference in the y variable between groups differing in x, this is the model you

need. But if we assume the causal structure shown in Figure 11.1, we cannot interpret its parameter estimates causally.

It seems clear what conclusion we ought to draw from the considerations above: if confounding variables are at play and we want to make causal claims, we need to include these confounders in the analysis. In practice, however, things aren't so simple.

- We may not know all confounders or we may not have been able to measure all of them.
- Even if we did measure the confouders, we probably didn't measure them perfectly.
- It is possible that the confounders exert some nonlinear influence, whereas we only considered linear effects.

In the exercises below, you'll explore the first two complicating factors. To anticipate the take-home message:

Tip 11.1. *Don't pin your hopes on statistical tools to neutralise the effect of confounding variables.* This would require you to have perfectly measured all confounders and to have specified the functional form of their causal effects correctly. *Statistical tools are no substitute for a solid research design that neutralises confounders.* ♢

Exercise 11.2 (Unknown confounders)**.** In Figure 11.4, one confounder, u, was added to our DAG, but for some reason, it wasn't measured. We assume that the following causal relationships are at play:

$$
\begin{aligned}
z_i &\sim \text{Normal}(0, 1^2), \\
u_i &\sim \text{Normal}(0, 1^2), \\
x_i &= 0 + 1.2 \cdot z_i + 0.9 \cdot u_i + \tau_i, \\
y_i &= 5.2 + 0.6 \cdot x_i - 1.3 \cdot z_i + 2.5 \cdot u_i + \varepsilon_i, \\
\tau_i &\sim \text{Normal}(0, 1^2), \\
\varepsilon_i &\sim \text{Normal}(0, 1^2),
\end{aligned}
$$

$i = 1, \ldots, n$, with all $z_i, u_i, \varepsilon_i, \tau_i$ independent. We can generate a dataset governed by these equations that contains 50 observations as follows. Note that while we simulate the u variable, we don't add it to the dataset since it wasn't measured.

```
n <- 50
z <- rnorm(n)
u <- rnorm(n)
x <- 0 + 1.2*z + 0.9*u + rnorm(n)
y <- 5.2 + 0.6*x - 1.3*z + 2.5*u + rnorm(n)
d <- tibble(y, x, z)
```

```
Error in loadNamespace(x):  there is no package called 'shape'
```

While we cannot include u in the model, we can include z:

Figure 11.4: The confounder z was measured; the confounder u wasn't.

```
exercise1.lm <- lm(y ~ x + z, data = d)
summary(exercise1.lm)$coefficients
```

```
             Estimate Std. Error   t value     Pr(>|t|)
(Intercept)  4.567136  0.2911181 15.688258 2.621409e-20
x            2.203009  0.3004465  7.332450 2.564975e-09
z           -2.845092  0.4471081 -6.363321 7.571485e-08
```

1. Explain what the parameter estimate for `x` literally means.

2. Adapt the `generate_dag1()` function and show that the `exercise1.lm` model does not provide an unbiased estimate of the causal effect of x on y, even though it includes the measured confounder z.

3. Would more data help solve this problem? Justify your answer by means of a simulation in which each dataset features 200 instead of 50 observations. ♢

Exercise 11.3 (Measurement error on the confounder)**.** Figure 11.5 depicts another scenario where z confounds the causal relationship between x and y. This time, however, we didn't measure z itself. Instead, we obtained an **indicator** z_m of z. This indicator represents the **construct** z imperfectly. This is quite usual: constructs such as working memory capacity, L2 writing skills, L1 vocabulary knowledge, intelligence, socioeconomic status etc., cannot be observed directly and have to be inferred from test results, questionnaire responses etc. It is therefore crucial to understand the effect of measurement error on statistical control.

We assume that the same causal links are at play as earlier. The only difference is that we include z_m instead of z in the dataset. To construct z_m, we take the values of z and add some Gaussian noise to it (from a normal distribution with mean 0 and standard

Figure 11.5: The z variable confounds the causal relationship between x and y, but it wasn't measured directly. Instead, we need to make do with a proxy variable z_m that captures z imperfectly.

deviation 0.3).

$$
\begin{aligned}
z_i &\sim \text{Normal}(0, 1^2), \\
z_{m,i} &= z_i + \psi_i, \\
x_i &= 0 + 1.2 \cdot z_i + \tau_i, \\
y_i &= 5.2 + 0.6 \cdot x_i - 1.3 \cdot z_i + \varepsilon_i, \\
\psi_i &\sim \text{Normal}(0, 0.3^2), \\
\tau_i &\sim \text{Normal}(0, 1^2), \\
\varepsilon_i &\sim \text{Normal}(0, 1^2),
\end{aligned}
\tag{11.3}
$$

$i = 1, \ldots, n$, with $z_i, \psi_i, \tau_i, \varepsilon_i$ independent.

```
Error in loadNamespace(x):  there is no package called 'shape'
```

Let's simulate the data.

```
n <- 50
z <- rnorm(n)
z_m <- z + rnorm(n, sd = 0.3)
x <- 0 + 1.2*z + rnorm(n)
y <- 5.2 + 0.6*x - 1.3*z + rnorm(n)
d <- tibble(y, x, z_m)
```

This time, we include z_m in the analysis:

```
exercise2.lm <- lm(y ~ x + z_m, data = d)
summary(exercise2.lm)$coefficients
```

```
              Estimate Std. Error   t value     Pr(>|t|)
(Intercept)  4.9841809  0.1366345 36.478213 3.937925e-36
x            0.3311196  0.1281096  2.584660 1.291282e-02
z_m         -1.1481046  0.2106489 -5.450324 1.808439e-06
```

1. Explain what the parameter estimate for `x` literally means.

Figure 11.6: In this scenario, the values of x were assigned randomly and independently of z. This is typical of experiments in which the participants are randomly assigned to the conditions.

2. Adapt the `generate_dag1()` function and show that the `exercise2.lm` model does not provide an unbiased estimate of the causal effect of x on y, even though it includes an indicator of the confounder (i.e., z_m).

3. Would more data help solve this problem? Justify your answer by means of a simulation in which each dataset features 200 instead of 50 observations.

4. Would it be better not to take into account the confounder at all? That is, should we just fit the model without the indicator of z?

5. What would happen if we had a more reliable indicator for z? To answer this question, rerun your simulation but use a standard deviation of 0.1 instead of 0.3 for ψ in Equation 11.3. ♢

Tip 11.4. If you read about a study that claims that the participants' 'intelligence' or 'socio-economic status' has been controlled for, mentally change this to 'an imperfect indicator of the participants' intelligence/socio-economic status'. ♢

11.2 Control variables in randomised experiments

In Section 11.1, we discussed a scenario in which z causally affects both x and y. This scenario is typical for observational (or correlational) studies and quasi-experiments. In the present section, we turn our attention to randomised experiments, that is, experiments in which the values of x are manipulated by the researchers based on random assignment. To reflect this, the x variable is represented as x_r in Figure 11.6 (r for *randomised*). A typical example for the more abstract scenario we consider here is an experiment in which participants are randomly assigned to the conditions. In this case, x would be a categorical variable. But without any loss of generality, we'll discuss a continuous x. This makes the simulations a bit easier; it doesn't affect the conclusions we will draw. The z variable would then be a variable that we're not really interested in but of which we suspect that it correlates with the outcome, y. The textbook case is a pretest/posttest experiment, where x represents the experimental condition, z the pretest performance, and y the posttest performance.

```
Error in loadNamespace(x):  there is no package called 'shape'
```

We assume that the causal links between the variables are described by the following equations:

$$\begin{aligned} x_i &\sim \text{Normal}(0, 1^2), \\ z_i &\sim \text{Normal}(0, 1^2), \\ y_i &= 5.2 + 0.3 \cdot x_i + 0.9 \cdot z_i + \varepsilon_i, \\ \varepsilon_i &\sim \text{Normal}(0, 1^2), \end{aligned} \tag{11.4}$$

$i = 1, \ldots, n$ with the x_i, z_i, ε_i independent. Let's simulate a dataset conforming to these equations:

```
n <- 100
x <- rnorm(n)
z <- rnorm(n)
y <- 5.2 + 0.3*x + 0.9*z + rnorm(n)
d <- tibble(y, x, z)
```

```
# Not shown in script
scatterplot_matrix(d)
```

We again fit one model with and one model without z. Both models yield similar though different estimates for the `x` parameter: 0.35 ± 0.14 and 0.38 ± 0.10.

```
dag2.lm1 <- lm(y ~ x, data = d)
summary(dag2.lm1)$coefficients
```

```
            Estimate Std. Error   t value     Pr(>|t|)
(Intercept) 5.400303  0.1419751 38.036964 1.715328e-60
x           0.354734  0.1403484  2.527525 1.308589e-02
```

```
dag2.lm2 <- lm(y ~ x + z, data = d)
summary(dag2.lm2)$coefficients
```

```
             Estimate Std. Error   t value     Pr(>|t|)
(Intercept) 5.3635827 0.09747464 55.025416 5.939051e-75
x           0.3805143 0.09632729  3.950223 1.481253e-04
z           0.9883193 0.09373457 10.543807 9.012801e-18
```

The simulations below will confirm that *both* models yield unbiased estimates of the causal influence of x on y. By this standard, neither model is bad. That said, if we have the z variable at our disposal, the second model is to be preferred. The reason for this will become clearer once we've simulated a couple of thousand datasets. To this end, we could write a new function, `generate_dag2()`. But we can also reuse `generate_dag1()` and set the parameter value of `z_x` to 0:

```
est_dag2 <- generate_dag1(x_y = 0.3, z_y = 0.9, z_x = 0)
```

As shown in Figure 11.7, both models yield unbiased estimates of the causal influence of x on y in Equation 11.4. But these estimates vary less from sample to sample in model

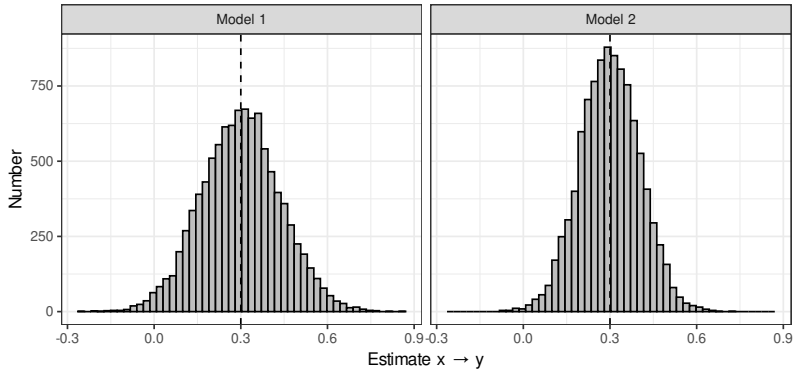


Figure 11.7: Both models yield unbiased estimates of the causal influence of x on y (0.3). But these estimates vary less from sample to sample in the second model than in the first model. In other words, on average, the second model yields more precise estimates.

2, that is, on average, they are closer to the true parameter value than the estimates that model 1 yields.

```
est_dag2 |>
  pivot_longer(cols = everything(),
               names_to = "Model",
               values_to = "Estimate") |>
  ggplot(aes(x = Estimate)) +
  geom_histogram(bins = 50,
                 fill = "grey", colour = "black") +
  geom_vline(xintercept = 0.3, linetype = "dashed") +
  facet_grid(cols = vars(Model)) +
  xlab("Estimate x → y") +
  ylab("Number")
```

We can check this numerically. The means of the estimates in both models correspond to the true value (making allowance for simulation error). But compared to the estimates by the first model, the standard deviation of the estimates by the second model is about 25% lower.

```
apply(est_dag2, 2, mean)

  Model 1   Model 2
0.3006039 0.3001774

apply(est_dag2, 2, sd)

  Model 1   Model 2
0.1373611 0.1029079
```

What's happened here is that by including the z variable in the model—even though it doesn't act as a confounder for the causal link between x and y—we have reduced the error variance and have hence increased the precision of our estimate. So even if you don't actually care about z, it can pay dividends to still collect it and include it in your analysis.

Tip 11.5 (Choosing control variables)**.** When designing an experiment, one or two additional variables that you suspect to be strongly correlated to the outcome may be useful. This is particularly true if they're not too strongly related to each other. Else they would both essentially be doing the same job. Such variables typically are *so* obviously related to the outcome that they are completely uninteresting in and of themselves (e.g., pretest performance). But don't collect umpteen additional variables on the off-chance that they might be related to the outcome and help you reduce the error variance. Moreover, the decision whether or not to include some additional variable in your analysis should be taken *before* you analyse your data—don't run several models and then report the one that works 'best'. You completely invalidate your inferential results this way. ♢

Exercise 11.6 (A pretest/posttest experiment (Part 1))**.** The data for this exercise stem from a study by Hicks (2021). She investigated how well 260 children could learn German–English cognates, that is, word pairs such as *Pfeffer–pepper* and *Haus–house*. There were three waves of data collection: T1, T2, and T3. After the first wave, an intervention was undertaken with 120 of the children, the goal of which was to impart to them a greater awareness of cognate correspondences. The other 140 children served as the control group. Here we're interested in the question if the intervention bore fruit, that is, if children who took part in the intervention were better able to learn German–English cognates.

For the time being, we'll pretend that the assignment of children to experimental condition was done at random and on an individual basis. We're interested in the T3 data (`T3cog`); the pretest scores from T1 serve as the control variable (`T1cog`); we ignore the T2 data.

Read in the data, retaining just the columns that we actually need:

```
hicks <- read_csv(here("data", "hicks2021.csv")) |>
  select(ID, Class, Group, T1cog, T3cog)
```

One option to visualise these data is to draw a scatterplot with the pretest scores along the x axis and the posttest scores along the y axis and with different colours depending on the experimental condition. Further, regression lines for both conditions can be added. These are derived from separate simple regression models for the two conditions:

```
ggplot(hicks,
       aes(x = T1cog, y = T3cog,
           colour = Group)) +
  geom_point(shape = 1) +
  geom_smooth(se = FALSE, method = "lm") +
  xlab("Pretest score") +
```

```
  ylab("Posttest score")
```

Based on this plot, how would you answer the research question? What aspect of the visualisation did you base your answer on?

Now fit a linear model of the form `outcome ~ condition + control`. Don't forget to create a dummy variable for the condition variable. Interpret the model in terms of the research question. ♢

Exercise 11.7 (A pretest/posttest experiment (Part II))**.** The children in the study by Hicks (2021) were pupils in classes. They were assigned to the experiment's conditions in whole classes rather than on an individual basis. This induces a dependency between different data points, threatening the validity of the inferential results obtained in the previous part of the exercise.

There are a couple of possible solutions to this problem (see Vanhove, 2020a, for a comparison). The easiest—and possibly the best—is to compute the mean pretest and mean posttest score for each cluster (class) and run the analysis using these averages instead. That is, assuming you named the dummy variable `n.Group`:

```
hicks_byclass <- hicks |>
  group_by(n.Group, Class) |>
  summarise(mean_T1 = mean(T1cog),
            mean_T3 = mean(T3cog))
byclass.lm <- lm(mean_T3 ~ n.Group + mean_T1, hicks_byclass)
```

Interpret the output of this model in terms of the research question. How would you report the finding in an article? Focus only on what's important. ♢

Exercise 11.8 (A pretest/posttest experiment (Part III))**.** In Exercise 11.7, we still assumed that random assignment was used, but on the level of the classes rather than on the level of the individual pupils. Look up in Hicks' article how the children were actually assigned to the different conditions. Briefly discuss plausible consequences. ♢

11.3 Posttreatment variables

We now turn our attention to scenarios where z is a posttreatment variable. This means that, when we draw a DAG, we can arrive at z starting in x by following arrows. In the context of a randomised experiment with x as the predictor of interest, this means that z was collected after the randomisation. As a consequence, z may be influenced by x.

First consider the DAG in Figure 11.8. There are two causal paths from x to y: one is direct ($x \to y$), one is **mediated** by z ($x \to z \to y$). A dataset conforming to this DAG can be simulated as follows:

```
Error in loadNamespace(x):  there is no package called 'shape'
```

```
n <- 200
x <- rnorm(n)
```

Figure 11.8

```
z <- 0.4*x + rnorm(n)
y <- 0.6*x + 1.2*z + rnorm(n)
d <- tibble(x, y, z)
```

What's the causal influence of x on y? That is, if you increase x by one unit, what change in y does this cause?

The answer to this question is *not* 0.6 units. While the direct causal effect of x on y is indeed such that a one-unit increase results in a 0.6-unit increase in y, x also influences y via z. A one-unit increase in x results in a 0.4-unit increase in z; and a one-unit increase in z results in a 1.2-unit increase in y. So in addition to the 0.6-unit increase that a one-unit increase in x causes directly in y, it also results in a $0.4 \cdot 1.2 = 0.48$-unit increase via z, for a total causal effect of $0.6 + 0.48 = 1.08$ units increase in y per one-unit increase in x!

If we want to estimate the total causal influence of x on y, we shouldn't close any causal paths going from x to y by controlling for z, that is, we should fit a simple regression model that does *not* include z. If, however, we want to estimate the causal effect of x on y that is not mediated by z, then we *do* need to control for z. Which model you want to run depends entirely on what you want to estimate.

```
total.lm <- lm(y ~ x, data = d)
direct.lm <- lm(y ~ x + z, data = d)
summary(total.lm)$coefficients[, 1:2]
```

```
             Estimate Std. Error
(Intercept) 0.2120306  0.1148429
x           1.1811532  0.1234403
```

```
summary(direct.lm)$coefficients[, 1:2]
```

```
              Estimate Std. Error
(Intercept) 0.02354685 0.07126880
x           0.70678600 0.08017829
z           1.25055066 0.06901613
```

Now consider the DAG in Figure 11.9. Here, both x and y causally affect z ($x \rightarrow z \leftarrow y$). A variable in which two or more causal variables clash together is known as a **collider**. In a sense, colliders are the opposite of confounders: As long as colliders are *not* controlled

Figure 11.9

for, their presence does not bias the causal estimates of interest. But once they are controlled for, they may bias these causal estimates.

```
Error in loadNamespace(x):  there is no package called 'shape'
```

By way of example, let's assume the data are generated following these equations:

$$
\begin{aligned}
x_i &\sim \text{Normal}(0, 1^2), \\
y_i &= 0.4x_i + \varepsilon_i, \\
z_i &= 0.5x_i + 0.8y_i + \tau_i, \\
\varepsilon_i &\sim \text{Normal}(0, 1^2), \\
\tau_i &\sim \text{Normal}(0, 1^2),
\end{aligned}
$$

$i = 1, \ldots, n$ with the $x_i, \varepsilon_i, \tau_i$ independent.

The function `generate_posttreatment()` is a slight adaptation of `generate_dag1()`.

```
generate_posttreatment <- function(
  n = 100,       # number of observations per dataset
  sims = 10000, # number of datasets
  x_y = 0.4,     # influence x -> y
  x_z = 0.5,     # influence x -> z
  y_z = 0.8      # influence y -> z
) {
  est_lm1 <- vector(length = sims)
  est_lm2 <- vector(length = sims)

  for (i in 1:sims) {
    x <- rnorm(n)
    y <- x_y*x + rnorm(n)
    z <- x_z*x + y_z*y + rnorm(n)
    mod.lm1 <- lm(y ~ x)
    mod.lm2 <- lm(y ~ x + z)
    est_lm1[[i]] <- coef(mod.lm1)[[2]]
    est_lm2[[i]] <- coef(mod.lm2)[[2]]
  }
```

Figure 11.10

```
  tibble(`Model 1` = est_lm1,
         `Model 2` = est_lm2)
}
```

As this simulation shows, the model without the collider (z) is able to estimate the causal effect of x on y (0.4) in an unbiased way. The estimates for the model with the collider, by contrast, are biased in this respect.

```
est_collider <- generate_posttreatment()
apply(est_collider, 2, mean)

     Model 1      Model 2
0.3997570388 0.0005970591
```

Nonetheless, the model with the collider is well-suited if you want to estimate the mean difference in the y variable between two groups that differ in the x variable but whose z values are all the same. As always, whether the model is justified depends on what you want to find out and on your assumptions. It also bears pointing out that colliders are sometimes inadvertently controlled for during the design stage. See Rohrer (2018).

Next, z is also a posttreatment variable in the DAG in Figure 11.10. This time, however, it is only indirectly influenced by x. In order to check whether it would be best to include z as a variable in the model, let's assume the following equations describe the causal links between the variables:

$$
\begin{aligned}
x_i &\sim \text{Normal}(0, 1^2), \\
y_i &= 0.4x_i + \varepsilon_i, \\
z_i &= y_i + \tau_i, \\
\varepsilon_i &\sim \text{Normal}(0, 1^2), \\
\tau_i &\sim \text{Normal}(0, 1^2),
\end{aligned}
$$

$i = 1, \ldots, n$ with the $x_i, \varepsilon_i, \tau_i$ independent.

```
Error in loadNamespace(x):  there is no package called 'shape'
```

We can reuse the `generate_posttreatment()` function and just specify the new parameter values. Again, the model without the additional variable provides an unbiased

Figure 11.11

estimate of the causal effect of x on y, whereas the model with this additional variable doesn't.

```
est_proxy_y <- generate_posttreatment(x_y = 0.4, x_z = 0, y_z = 1)
apply(est_proxy_y, 2, mean)
```

```
  Model 1   Model 2
0.3986262 0.1993806
```

In the scenario depicted in Figure 11.10, including the additional variable will bias the estimate of interest towards zero. By spelling out what the literal meaning is of the estimated parameter for `x` in the second model, you should see why this is the case.

The situation is only slightly different in the scenario depicted in Figure 11.11. Here, z is directly affected by x, but not by y. We'll simulate datasets conforming to the following equations:

$$
\begin{aligned}
x_i &\sim \text{Normal}(0, 1^2), \\
y_i &= 0.4x_i + \varepsilon_i, \\
z_i &= x_i + \tau_i, \\
\varepsilon_i &\sim \text{Normal}(0, 1^2), \\
\tau_i &\sim \text{Normal}(0, 1^2),
\end{aligned}
$$

$i = 1, \ldots, n$ with the $x_i, \varepsilon_i, \tau_i$ independent.

```
Error in loadNamespace(x):  there is no package called 'shape'
```

As the simulation results show, both models yield unbiased estimates of the causal effect of x on y. But notice that the estimates vary more from sample to sample for the second model than for the first. So a given estimate resulting from the model without the additional variable is more likely to be closer the actual parameter value than a given estimate resulting from the model with the additional variable. It's hard to imagine a research question where the second model would be preferred in this scenario.

```
est_proxy_x <- generate_posttreatment(x_y = 0.4, x_z = 1, y_z = 0)
apply(est_proxy_x, 2, mean)
```

```
  Model 1   Model 2
0.3997494 0.4004841
```

Figure 11.12

Figure 11.13

```
apply(est_proxy_x, 2, sd)
```

```
  Model 1   Model 2
0.1022453 0.1466505
```

In sum, including posttreatment variables as predictors in the analysis is usually a bad idea. In randomised experiments, there luckily exists a simple trick for preventing that the predictors you include in the model are posttreatment variables: just collect the predictors before carrying out the intervention!

Exercise 11.9. Consider Figure 11.12. Assume that you want to obtain an unbiased and maximally precise estimate the total causal effect of x on y and that all relationships shown are linear and additive. Which variables would you include in a general linear model as predictors? Justify your answer. ♢

```
Error in loadNamespace(x):  there is no package called 'shape'
```

Exercise 11.10. Same question for Figure 11.13. ♢

```
Error in loadNamespace(x):  there is no package called 'shape'
```

Exercise 11.11. Let's say that instead of merely observing and measuring all the variables in Figure 11.12, we devise a randomised experiment in which we randomly assign the participants to values of x.

1. Draw the updated DAG.
2. Which predictors would you in the general linear model now? Justify your answer. ♢

Exercise 11.12 (Interpreting model estimates (1))**.** Vanhove et al. (2019) had 1,000 short French texts written by children rated for their lexical diversity on a 9-point scale by between 2 and 18 raters each. For the purposes of this exercise, we're interested in modelling these human ratings in terms of the length of the text (the logarithmically transformed number of tokens (using base 2), `log2_ntokens`) and the type–token ratio (`TTR`), an easily computed metric of a text's lexical diversity.

```
d <- read_csv(here("data", "text_ratings.csv"))
lexdiv.lm <- lm(mean_rating ~ log2_ntokens + TTR, data = d)
summary(lexdiv.lm)$coefficients
```

```
              Estimate Std. Error   t value      Pr(>|t|)
(Intercept)  -4.092804  0.4580283 -8.935701  1.918579e-18
log2_ntokens  1.215198  0.0473475 25.665512 6.119808e-112
TTR           3.836416  0.3305577 11.605889  2.670688e-29
```

1. For each claim, decide whether it is correct. Justify your answers.
   - The model output shows that human raters are sensitive to differences in the type–token ratio when ratings texts for their lexical diversity.
   - The model output shows that there is a positive linear relationship between the TTR values and the mean ratings: texts with higher TTR values tend to receive higher ratings than texts with lower TTR values.
2. Draw a scatterplot matrix of these variables and revise your answer to the previous question if needed.
3. Explain what each of the three estimated model parameters literally means.
4. According to this model, what mean rating would you expect for a text consisting of 64 tokens with a TTR of 0.7? ♢

Exercise 11.13 (Interpreting model estimates (2))**.** We're given a data set `d` with the outcome `CorrectSpoken` and four predictors:

- `n.Sex`: 0.5 for men, -0.5 for women (sum coding).
- `NrLang`: Number of languages spoken for each participant (varies from 1 to 5).

- `DS.Span`: Score on the backward digit span, a working memory test (varies from 2 to 8, more is better).
- `Raven.Right`: Score on a test of fluid intelligence (varies from 0 to 35, more is better).

We fit the following model:

```
mod.lm <- lm(CorrectSpoken ~ NrLang + DS.Span + n.Sex*Raven.Right,
             data = d)
summary(mod.lm)$coefficients
```

```
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)          8.554      1.532    5.58  1.0e-07
NrLang               1.115      0.353    3.16  1.9e-03
DS.Span             -0.097      0.330   -0.29  7.7e-01
n.Sex               -5.017      1.716   -2.92  4.0e-03
Raven.Right          0.286      0.048    6.01  1.3e-08
n.Sex:Raven.Right    0.201      0.088    2.28  2.4e-02
```

1. What does the estimate of -5.02 for the `n.Sex` parameter refer to? That is, what does it mean literally?
2. Let's say we wanted to fit a similar model but one that has an intercept with a more useful interpretation. Explain how we could achieve this. Also explain how we can interpret the intercept in this new model.
3. Claim: We can glean from the model that, at least in this data set, participants with a high working memory capacity do not outperform participants with a low working memory capacity in terms of the `CorrectSpoken` variable.
   Is this claim correct? (Yes or no.) Justify your answer.
4. What gets computed here?

   ```
   8.554 + 1.115*3 - 0.097*4 + 0.286*25
   ```

   ```
   [1] 19
   ```

5. What, according to the model, is the expected average difference in `CorrectSpoken` between women with a `Raven.Right` score of 20 and women with a `Raven.Right` score of 30, keeping `NrLang` and `DS.Span` constant. ♢

The following exercise isn't specific to this chapter. Rather, you can draw on what we've covered so far to tackle it

Exercise 11.14 (Analysing an experiment)**.** From the abstract of Vanhove (2016):

> "This article investigates whether learners are able to quickly discover simple, systematic graphemic correspondence rules between their L1 and an unknown but closely related language in a setting of receptive multilingualism.
>
> (...)

> Eighty L1 German speakers participated in a translation task with written Dutch words, most of which had a German cognate. In the first part of the translation task, participants were shown 48 Dutch words, among which either 10 cognates containing the digraph ‹oe› (always corresponding to a German word with ‹u›) or 10 cognates with the digraph ‹ij› (corresponding to German ‹ei›). During this part, participants were given feedback in the form of the correct translation. In the second (feedback-free) part of the task, participants were shown another 150 Dutch words, among which 21 cognates with ‹oe› and 21 cognates with ‹ij›."

What I wanted to know was whether the exposure and feedback to ten words containing a specific interlingual orthographic correspondence (i.e., ‹oe›–‹u› or ‹ei›–‹ij›) was sufficient for the participants to pick up on this correspondence and use their knowledge in translating new words containing the correspondence.

First, we read in the data and compute, for each participant, the proportion of words in the second (feedback-free) part of the task which they translated correctly in each category (cognates containing ‹oe›, cognates containing ‹ij›, other cognates, non-cognates).

```
d <- read_csv(here("data", "correspondencerules.csv"))
d_perParticipant <- d |>
  filter(Block != "training") |>
  group_by(Subject, LearningCondition, Category, WSTRight) |>
  summarise(ProportionCorrect = mean(Correct == "yes"),
            .groups = "drop")
```

I've also retained a measure of the participants' L1 (German) vocabulary knowledge, namely the `WSTRight` variable. Use `View(d_perParticipant)` to inspect the structure of the cleaned-up and restructured dataset.

Analyse this dataset to answer the research question. ♢

11.4 *For the sake of completeness

11.4.1 Collinearity

You may come across the term **collinearity** (or **multicollinearity**) in studies, reviews, or while browsing the internet. I won't dwell too much on this term here and instead refer you to Vanhove (2021a), in case someone ever lobs it in your direction.

11.4.2 Multiple R^2 and adjusted R^2

One line in the `summary()` output that we've so far ignored is the one containing `Multiple R-squared` (R^2) and `Adjusted R-squared` (R^2_{adj}). Let's take a closer look at these numbers now. We'll use a dataset from Vanhove et al. (2019). The file `helascot_ratings.csv` contains between 2 and 18 ratings on a 9-point scale of lexical richness in short texts written by children. Here, we focus on argumentative French texts, written at the second mea-

surement point and judged by raters with French as their native language ("bi-French" or "mono-French"). For each text, we calculate the average rating:

```
ratings <- read_csv(here("data", "helascot_ratings.csv"))
ratings_per_text <- ratings |>
  filter(Text_Language == "French") |>
  filter(Text_Type == "arg") |>
  filter(Time == 2) |>
  filter(Rater_NativeLanguage %in% c("bi-French", "mono-French")) |>
  group_by(Text) |>
  summarise(mean_rating = mean(Rating))
```

The file `helascot_metrics.csv` contains a heap of quantified lexical features for each rated text. We add these to the tibble with the mean ratings:

```
metrics <- read_csv(here("data", "helascot_metrics.csv"))
ratings_per_text <- ratings_per_text |>
  left_join(metrics, by = "Text")
```

We want to run a regression model that captures how the average rating relates to selected text features. The first feature is the number of tokens in the rated text (`nTokens`); the second is the Guiraud index, which is calculated as:

$$\text{Guiraud} = \frac{\text{number of types}}{\sqrt{\text{number of tokens}}}.$$

Figure 11.14 shows a scatterplot matrix of the three variables. By the way, I prefer to put the outcome at the top left and the predictors to the right.

```
ratings_per_text |>
  select(mean_rating, Guiraud, nTokens) |>
  scatterplot_matrix(labels = c("Mean rating", "Guiraud",
                                "Number of tokens"))
```

The histogram for `nTokens` shows positive skew. Since this variable can only take strictly positive values, we can apply a logarithm to counteract this. Here we'll use the base-2 logarithm, though any other base would have worked just as well. The nice thing about base 2 is that I can tell at a glance that the value 4 represents texts of length 16 (since $2^4 = 16$), 5 represents length 32 (twice as long), and 6 represents length 64 (again twice as long).

```
ratings_per_text$log2.nTokens <- log2(ratings_per_text$nTokens)
```

Now let's finally turn to the R^2 values. First, we'll fit a linear model with the three variables and then apply the `summary()` function to the model object:

```
ratings.lm <- lm(mean_rating ~ log2.nTokens + Guiraud,
                 data = ratings_per_text)
summary(ratings.lm)
```

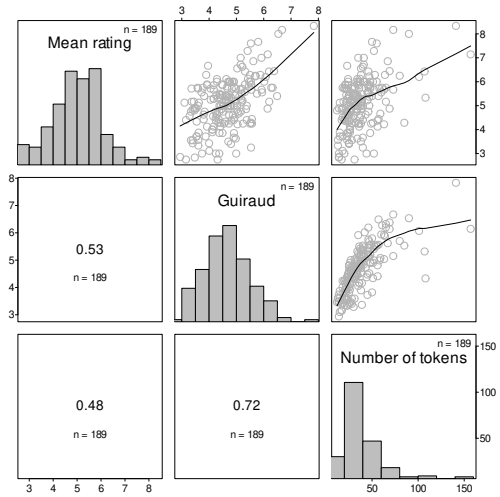


Figure 11.14: Scatterplot matrix with the mean text ratings, the Guiraud values, and the number of *tokens* per text. Since the distribution of token counts is right-skewed, we'll work with their logarithms instead of the raw values.

```
Call:
lm(formula = mean_rating ~ log2.nTokens + Guiraud, data = ratings_per_text)

Residuals:
        Min          1Q      Median          3Q         Max
-2.45984631 -0.64527774  0.03141186  0.60663167  2.08136848

Coefficients:
              Estimate Std. Error t value   Pr(>|t|)
(Intercept)  1.2234740  0.5040317 2.42738 0.01616150
log2.nTokens 0.3492755  0.1595304 2.18940 0.02981276
Guiraud      0.4497623  0.1282107 3.50799 0.00056611

Residual standard error: 0.9015863 on 186 degrees of freedom
Multiple R-squared:   0.29429, Adjusted R-squared:  0.2867017
F-statistic: 38.78217 on 2 and 186 DF,  p-value: 8.360733e-15
```

The first number (`Multiple R-squared`, usually written R^2) indicates the proportion of variance in the outcome that the estimated regression equation captures in this dataset. This is often called the 'explained variance', but strictly speaking the model doesn't really 'explain' anything—that's up to the researchers. To see where this number comes from, we can calculate the variance of the outcome, which is about 1.14:

```
var(ratings_per_text$mean_rating)

[1] 1.139576226
```

The variance of the model residuals is about 0.80:

```
var(resid(ratings.lm))

[1] 0.8042103604
```

So of the variance in the outcome variable, about $\frac{0.80}{1.14} = 71\%$ remains after taking into account the linear relationships with the predictors. In other words, the model captures 29% of the outcome variance:

```
1 - var(resid(ratings.lm)) / var(ratings_per_text$mean_rating)

[1] 0.2942899807
```

The formula for computing multiple R^2, then, is

$$R^2 = 1 - \frac{s^2(\boldsymbol{\varepsilon})}{s^2(\boldsymbol{y})},$$

where $s^2(\boldsymbol{y})$ is the sample variance of the outcome variable and $s^2(\boldsymbol{\varepsilon})$ is the sample variance of the model's residuals.

There are other ways of calculating R^2, but for the general linear model they all give the same answer—not so for the generalised linear model, which we haven't covered yet. See Kvålseth (1985) for details.

One issue with R^2 is that it can only increase as you add more predictors to the model. This happens even if the predictors have nothing to do with the outcome: by pure chance, the estimated regression coefficient for the link between an irrelevant predictor and the outcome will almost never be exactly 0 in the sample. So in the *sample*, an irrelevant predictor will still capture a little variance in the outcome, even though it doesn't in the *population*. We can easily test this by adding a random variable with no relation to the outcome to the dataset and the model:

```
ratings_per_text$noise <- rnorm(n = nrow(ratings_per_text))
ratings.lm2 <- lm(mean_rating ~ log2.nTokens + Guiraud + noise,
                  data = ratings_per_text)
summary(ratings.lm2)$r.squared

[1] 0.2946953536
```

The R^2 value is now a bit higher than before, even though the new predictor is completely irrelevant. To counter this problem, the R^2 value is sometimes corrected downwards.

This is achieved by using the estimate $\widehat{\sigma}_\varepsilon^2$ from page 161 in lieu of $s^2(\boldsymbol{\varepsilon})$:

$$\begin{aligned} R^2_{\text{adj}} &= 1 - \frac{\widehat{\sigma}_\varepsilon^2}{s^2(\boldsymbol{y})} \\ &= 1 - \frac{(n-1)s^2(\boldsymbol{\varepsilon})}{(n-p)s^2(\boldsymbol{y})} \\ &= 1 - \frac{(n-1)R^2}{n-p}, \end{aligned}$$

where n is the number of data points and p the number of estimated parameters. In our case, this yields for the `ratings.lm` model:

```
1 - (1 - summary(ratings.lm)$r.squared)*(189 - 1)/(189 - 3)
```

```
[1] 0.2867017009
```

and for the `ratings.lm2` model:

```
1 - (1 - summary(ratings.lm2)$r.squared)*(189 - 1)/(189 - 4)
```

```
[1] 0.2832579809
```

These are the `Adjusted R-squared` values reported in the `summary()` output.

Personally, I'm not a huge fan of R^2 or R^2_{adj}; see *Why reported R^2 values are often too high* (22 April 2016). Many researchers also seem to think that R^2 tells them how much variance the fitted regression model will capture in a new sample. But that's not true: R^2_{adj} estimates how much variance a model with the same predictors *but with newly estimated parameters* will capture in a new sample—under the assumption that both the original and the new sample are random draws from the same population. If you're really interested in the predictive power of a model, you're better off reading up on the principles of predictive modelling. See the reading recommendations at the end of this chapter.

11.5 Summary

- A multiple regression model is not just multiple simple regressions run at once. The meaning of the parameter estimates changes if you add or remove predictors to or from the model. Also see Morrissey & Ruxton (2018) and Vanhove (2021a).
- The decision which predictors to include in a model depends on what it is exactly you want to estimate and how you think different variables may be causally related. In my experience, people's difficulties with regression models aren't so much statistical in nature as due to their not having worked out what they actually want to find out.
- Know what the literal meaning of the estimated model parameters is before you interpret them in terms of the subject matter.
- As shown in the exercises, plots of the actual data may help prevent both you and your readership from interpreting the model output incorrectly.

11.6 *Further reading

On the limits of statistical control in observational studies, see Christenfeld et al. (2004), Huitema (2011, Part VII) and Westfall & Yarkoni (2016). On the utility of statistical control in randomised experiments, see Vanhove (2015) and references therein. For an accessible introduction to DAGS, confounders and colliders, see Rohrer (2018). Readers interested in using regression models for prediction will benefit from reading Shmueli (2010) and Kuhn & Johnson (2013).

Part IV

Significance testing

Chapter 12

The logic of significance testing

This chapter explains the basic logic behind p-value-based inferential statistics.[1] It does so by explicitly linking the computation of p-values to the random assignment of participants to conditions in experimental research. If you have ever taken an introductory statistics class, chances are p-values were explained to you in a different fashion, presumably by making assumptions about how the observations in the sample were sampled from a larger population and by making reference to the central limit theorem. For the explanation in this chapter, however, we're going to take a different tack and we will ignore the sampling method and the larger population. Instead, we're going to leverage what we know about how the observations, once sampled, were *assigned* to the different conditions of an experiment. The advantages of this approach are that it connects the design of a study more explicitly to the analysis of its data and that it is less math-intensive while permitting one to illustrate several key concepts about inferential statistics.

12.1 Randomisation as a basis for inference

Imagine the following experiment. To study the effect of alcohol on speaking speed, ten students (L1 German, L2 English) are randomly assigned to either the control or the experimental condition (five each); they don't know which condition they're assigned to. (Ten participants is obviously a very low number of participants, but it keeps things more tractable here.) Participants in the experimental condition drink one pint of ordinary beer; those in the control condition drink one pint of alcohol-free beer. Afterwards, they watch a video clip and relate what happens in it in English. This description is taped, and two independent raters who don't know which condition the participants were assigned to count the number of syllables uttered by the participants during the first minute. The mean of these two counts serves as the verbal fluency/speech rate variable.

By chance, Sandra, Nicole, Maria, Yves, and Daniel were assigned to the experimental group, while Nadja, Laura, Lukas, Thomas, and Michael ended up in the control group. Figure 12.1 shows their speech rates. On average, the experimental group uttered 4.3 syllables per second, compared to the 3.7 syllables per second uttered on average by the

[1] Large parts of this chapter are copied or adapted from the chapter *Inferential statistics 101* in my booklet *Quantitative methodology: An introduction*.

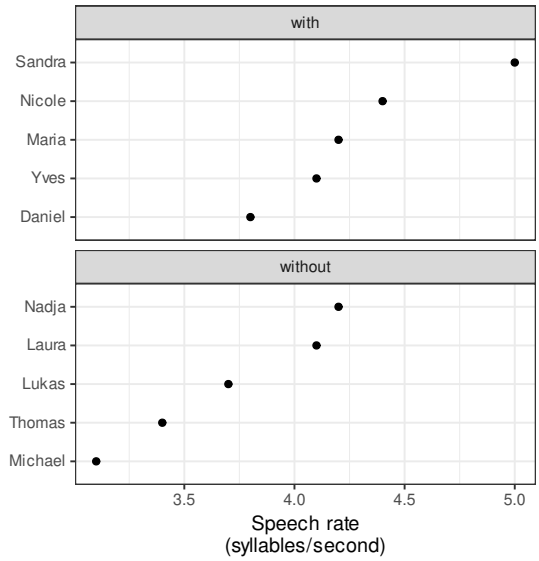


Figure 12.1: Results from a fictitious experiment.

control group. But can we conclude from this mean difference of 0.6 syllables per second that the drinking alcoholic versus alcohol-free beer affects speech rate in an L2—or might it just be due to random chance?

The participants were assigned to one of the groups at random. This ensures that the results were not systematically biased. For instance, the control group happened to contain only two women, whereas the experimental group contained three. But this difference is purely accidental. The aim of randomisation is not to create perfectly equivalent groups, but rather to prevent systematic bias—whether from known or unknown confounding variables. See Vanhove (2015) on this common misunderstanding.

In addition, this is a double-blind experiment: neither the participants themselves nor the staff evaluating the data knew who was assigned to which condition. This helps to avoid distortions of the results due to expectancy effects on the part of the participants or on the part of the researchers.

Of course, we could have refined this design further—for example, by ensuring that the participants' regional background was balanced across the two groups (e.g., one from Graubünden, one from Zurich, and one from Berne in each group), or by measuring participants' speaking speed before the experiment ('pre-test'), so that we could take it into account in the analysis. But even without such refinements, thanks to randomisation and blinding, this design still permits valid inferences.

The difference between the group means is 0.6 syllables per second. Because we carried out a randomised experiment and thus prevented systematic bias in the results, we might conclude that this difference was at least partly *caused* by our experimental manipulation: drinking a pint of alcoholic beer increases speech rate. Before making such a causal claim, however, we need to consider a more trivial explanation: perhaps the difference is simply

due to chance. This is our **null hypothesis** (H_0), which we contrast with a deliberately vague **alternative hypothesis** (H_A):

- H_0: The difference between the two means is *only* due to chance.
- H_A: The difference is *not only* due to chance.

In the so-called 'frequentist' tradition of null hypothesis testing, one calculates how likely it is to observe the given pattern (here: the difference between the group means), or an even more extreme pattern, if the null hypothesis were in fact true. This probability is called the p-value (p for *probability*). If this probability is small, the usual conclusion is that the assumption that the null hypothesis holds is probably not justified, and that some systematic effect is also at play. In practice, one typically works with an arbitrary threshold α (often $\alpha = 0.05$), below which the p-value is considered too small.

Before turning to some conceptual problems with this approach and discussing a few complications, let us first look at a method for calculating the p-value. If we assume the null hypothesis is true, then the difference between the groups is purely the result of randomisation—in other words, chance. Under this assumption, Michael would also have produced 3.1 syllables per second had he been in the experimental group; likewise Sandra would have produced 5.0 syllables per second had she been in the control group. So if the randomisation had assigned Sandra rather than Michael to the experimental group, and alcohol had no effect on speaking speed, then the experimental group's mean would have been 3.92 and the control group's mean 4.08. In that case we would have found the experimental group to speak 0.16 syllables per second faster.

Altogether there are 252 possible ways in which the experimental and control groups could have been formed. This number can be calculated using the binomial coefficient (see Section 3.3.3):[2]

$$\binom{10}{5} = \frac{10!}{(10-5)!5!} = \frac{3628800}{120^2} = 252.$$

Or in R:

```
choose(n = 10, k = 5)

[1] 252
```

Nine of these 252 possibilities are shown in Figure 12.2. For each case, we can calculate the size of the group difference. The R code is not important here, which is why I do not show it; only the logic matters.

Figure 12.3 shows all 252 possible group differences that could have arisen if the null hypothesis were actually true. On average, under the null hypothesis, the group difference would be 0. But depending on which participant was assigned to which condition, one could have observed smaller, or indeed larger, differences between the groups. From this figure we can see how unusual it would be to obtain group differences at least as large as the one we actually found, if the null hypothesis held. The dashed lines indicate

[2]For larger groups this number becomes too large to present clearly. For instance, there are 137,846,528,820 possible ways to split a group of 40 participants into two groups of equal size.

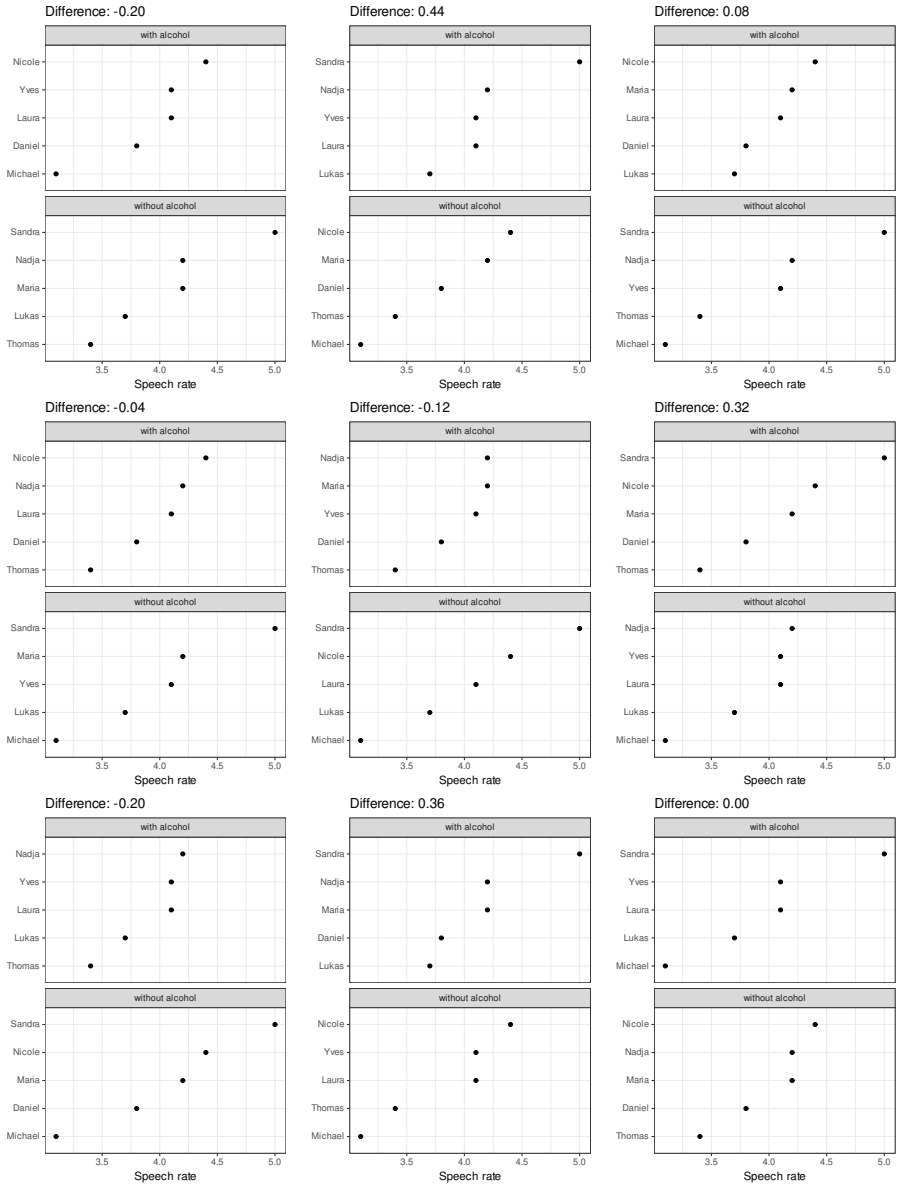


Figure 12.2: Nine of the 252 possible outcomes under the assumption that any difference between the conditions is due only to the random assignment.

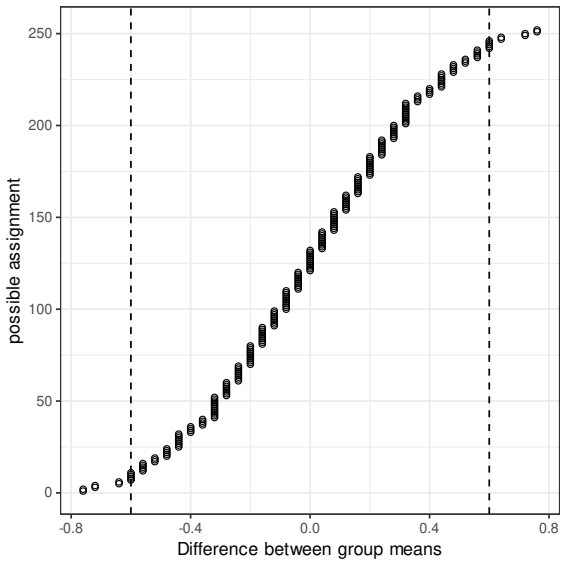


Figure 12.3: The differences between the group means in all 252 possible assignments. The vertical dashed lines indicate the actually observed difference of 0.6 syllables per second and its opposite number (-0.6).

that in eleven out of 252 cases, the mean difference is at least 0.6 syllables per second in favour of the experimental group, and in another eleven out of 252 cases, it is at least 0.6 syllables per second in favour of the control group. Therefore, even if the null hypothesis were in fact true in our example, we would still obtain a group difference of 0.6 syllables per second—or even larger—in 22 out of the 252 possible samples, i.e., $22/252 = 8.7\%$ of the time. This is our p-value.

Using the threshold $\alpha = 0.05$, we would find that $p = 0.087 > \alpha$ and conclude that a difference of 0.6 or more is still likely enough to occur under the null hypothesis of chance alone. Hence, we would see little need to revisit the assumption that the results may be due to chance alone. Crucially, this doesn't mean that we have shown H_0 to be true. It's just that H_0 would account reasonably well for these data. The absence of evidence for a difference is not the same as evidence for the absence of a difference.

Remark 12.1 (One- and two-sided p-values)**.** In the example above, we computed a two-sided p-value: we counted the rerandomisations that resulted in differences as least as extreme as the one observed in either direction. If we had hypothesised in advance that we would observe a difference in favour of the *with alcohol* condition, we could have computed a one-sided (right-sided) p-value and only have counted the eleven rerandomisations resulting in a mean difference as least as large as 0.6. The resulting p-value would have been $p_r = \frac{11}{252} = 0.044$.

Conversely, if we had hypothesised in advance that we would observe a difference in favour of the *without alcohol* condition, we could have computed a *left*-sided p-value and

only have counted the rerandomisations resulting a mean difference at *most* as large as 0.6. The resulting p-value would have been $p_\ell = \frac{245}{252} = 0.972$.

Since we didn't specify the direction of the expected difference in advance, we need to consider both possibilities simultaneously and compute a two-sided p-value. One way to compute two-sided p-values is to count the rerandomisations resulting in a mean difference at least as large in absolute value as the difference actually observed (in our case: $p = \frac{22}{252}$). Alternatively, a two-sided p-value can be computed as

$$p = 2\min(p_\ell, p_r).$$

Both of these valid ways coincide in 'nice' cases (e.g., exhaustive randomisation tests with an equal number of participants in both conditions), but they may yield somewhat different results in other cases. ♢

Remark 12.2 (Link to the research design)**.** The hypothesis test we just conducted is a **randomisation test**. Its use is legitimised by the research design, more specifically by the unrestricted randomisation and blinding:

- If it had been the case that we could never have assigned Daniel to the 'with alcohol' group for medical reasons, then there would not have been 252 possible outcomes under the null hypothesis, but only 126: all permutations with Daniel in the 'with alcohol' group would not have been possible. In the analysis, we would then have to take this into account by ignoring those permutations.
- If it had been the case that Michael and Laura had insisted on being tested in the same condition, then again there would not have been 252 possible outcomes under the null hypothesis, but only 126: all permutations with Michael and Laura in different conditions would have been impossible. This too would need to be accounted for in the analysis.
- If, for the purpose of balancing the two groups, we had wanted to assign at least one woman and one man to each condition, then the two permutations with only men or only women in a condition would not have been possible. Such a design may be sensible in some circumstances, but this restriction would also need to be taken into account. ♢

Remark 12.3 (Random assignment vs random sampling)**.** This experiment and its analysis illustrate the difference between **random assignment** and **random sampling**: we randomly assigned our participants to conditions, but we did not randomly select them from some population. In many introductions to inferential statistics, p-values are instead introduced by imagining drawing two random samples from two populations and asking whether the populations have the same mean. However, as already discussed in Section 5.4, real-world data rarely come from random samples; experiments with random assignment are, however, quite common.

Some caution is needed, though. Even if we had found more convincing evidence for an effect of the experimental manipulation, we still could not automatically conclude that the manipulation would have an effect in a particular *population*. Such a conclusion

would be at least more defensible if we had both randomly sampled participants from that population (*random sampling*) and then randomly assigned them to conditions (*random assignment*). Without random sampling, any generalisation to a population relies on a (often implicit) logical argument, rather than a statistical fact. This does not mean such a conclusion is necessarily wrong. But it is important to recognise that this is not a statistical question.

This nuance corresponds to the distinction between **internal validity** (Is the difference or effect observed in this sample attributable to the experimental manipulation?) and **external validity** (Can this finding be generalised beyond the sample?) Someone interested in the effectiveness of teaching methods would need to consider the external validity of the study. But for experimental psychologists, external validity is not necessarily so important (Mook, 1983): for them it can be more important to show that a manipulation can produce an effect at all, without having to establish the boundaries of that effect yet. ♢

Remark 12.4 (Statistical vs scientific hypotheses)**.** The null and alternative hypotheses formulated for the test are statistical hypotheses. These carry surprisingly little scientific content: the null hypothesis merely states that the observed patterns are purely due to chance; the alternative hypothesis states that they are not purely due to chance. It does not tell us what else, besides chance, might be causing the pattern. In our example, a few possible drivers of any difference between the groups could be:

- Alcohol lowers inhibitions when speaking, which in turn might increase speaking speed.
- People who have consumed a pint of alcoholic beer might rely on simpler structures and set phrases, which they can articulate faster than more complex structures or novel expressions.
- Alcohol impairs articulatory motor control.
- The raters somehow became aware of the participants' group assignments, and consciously or unconsciously allowed this knowledge to influence their ratings.

The key point is this: Even if a pattern is highly unlikely under the null hypothesis, a significance test does not tell you why the pattern might have occurred.

It is also worth noting that the alternative hypothesis is statistically vague. For example, it does not specify how large a possible alcohol-induced change in speaking speed might be. When a two-sided test is used, it doesn't even say whether this change would be an acceleration or a slowdown. ♢

12.2 A template for exact p-values

With the conceptual foundations out of the way, we're ready to treat p-values a bit more formally.

Definition 12.5 (p-value)**.** An exact p-value is a random variable P with the following property: if the null hypothesis is true, then for all $\alpha \in (0, 1)$ it holds that $\mathbb{P}(P \leq \alpha) \leq \alpha$.

A random variable P works as an approximate p-value if, under the null hypothesis, $\mathbb{P}(P \leq \alpha) \approx \alpha$ for all $\alpha \in (0,1)$. $\diamondsuit$

In principle, then, p-values can be used to bound the probability that we claim that something is going on in the data if in fact nothing is going on: We pick some value for α (typically 0.05). If the H_0 is in fact true, then we'd only observe p-values lower than α with a probability of at most α. In practice, however, things aren't so simple, and we'll discuss some complications shortly.

Remark 12.6 (Further quality criteria)**.** The constant random variable $P \equiv 1$ is, by Definition 12.5, a valid exact p-value, because

$$\mathbb{P}(P \leq \alpha) = 0 \leq \alpha$$

for all $\alpha \in (0,1)$. However, it is completely useless as a p-value, since it always has the same distribution—regardless of whether the null hypothesis is true or not. Similarly, a random variable $P \sim \mathrm{Unif}((0,1))$ is also a valid exact p-value, because

$$\mathbb{P}(P \leq \alpha) = \alpha \leq \alpha$$

for all $\alpha \in (0,1)$. But this p-value, too, always has the same distribution, whether or not the null hypothesis is true. What we actually want are p-values that are only rarely 'small' when the null hypothesis holds, but much more often small when it does not. $\diamondsuit$

The randomisation test we used in the previous section results in exact p-values, as the following lemma shows.

Lemma 12.7. Under the null hypothesis, exhaustive rerandomisation results in a p-value such that $\mathbb{P}(p \leq \alpha) \leq \alpha$ for each $\alpha \in (0,1)$. $\diamondsuit$

Proof. We'll first consider left-sided p-values. Assume that there are M possible rerandomisations. Sort these ascendingly by the value of the test statistic they result in (e.g., the mean difference in our example), breaking ties randomly if necessary. Note that, after this sorting, the left-sided p-value that the i-th rerandomisation would have resulted in (call it p_i) is at least $i/M, i = 1, \ldots, M$: there are at least i out of M rerandomisations with test statistics that are at most as large as as the one in the i-th rerandomisation; an even higher value than i/M is possible if several rerandomisations give rise to the same test statistic value.

Now, for $\alpha \in (0,1)$, compute $\kappa := \lfloor \alpha M \rfloor$. ($\lfloor \cdot \rfloor$ rounds down the number to the nearest integer.) Under the null hypothesis, we were equally likely to have generated each of the M possible randomisations. Hence, the probability that we generated a randomisation

resulting in a p-value no larger than α is

$$\begin{aligned}
\mathbb{P}(p \leq \alpha) &= \frac{\text{\# rerandomisations with } p_i \leq \alpha}{M} \\
&\leq \frac{\text{\# rerandomisations with } i/M \leq \alpha}{M} \\
&= \frac{\text{\# rerandomisations with } i \leq \kappa}{M} \\
&= \frac{\kappa}{M} \\
&\leq \alpha.
\end{aligned}$$

For right-sided p-values, sort the rerandomisations descendingly by the test statistic value and proceed analogously. For two-sided p-values, sort the rerandomisations by the absolute value of the test statistic and again proceed analogously.[3] □

Lemma 12.7 is a special case of Lemma 12.8, the proof of which can be found in Dümbgen (2016, p. 5).

Lemma 12.8. Let X be a random variable with distribution function F_0. Then the following inequalities hold:

$$\mathbb{P}(F_0(X) \leq \alpha) \leq \alpha, \tag{p_ℓ}$$

$$\mathbb{P}(1 - F_0(X-) \leq \alpha) \leq \alpha, \tag{p_r}$$

$$\mathbb{P}(2 \cdot \min\{F_0(X), 1 - F_0(X-)\} \leq \alpha) \leq \alpha, \tag{p}$$

for all $\alpha \in (0,1)$, where $F_0(t-) := \mathbb{P}(X < t)$ (and so $1 - F_0(t-) = \mathbb{P}(X \geq t)$). ◇

Lemma 12.8 shows that we can use the following template when computing p-values:

- First, decide which quantity X you are interested in. In the example above, we were interested in the difference between the sample means, but soon we will consider other quantities as well.
- Next, determine how this quantity X would be distributed if the null hypothesis were true. This gives us the distribution function F_0 of X. In the example above, we obtained F_0 by going through all possible assignments, and the value $F_0(r)$ corresponds to the proportion of assignments that result in a mean group difference of at most r units in favour of the alcohol group.
- According to the lemma, we can now compute three types of p-values: a left-sided (p_ℓ), a right-sided (p_r), and a two-sided (p); see Remark 12.1.

Exercise 12.9. Let's say that, in the speech rate example, we hypothesised that drinking a pint of alcoholic beer will result in a faster speech rate in some people but in a slower speech rate in others. It may make more sense to compare the variances of the speech

[3] When using the alternative definition of two-sided p-values, consider that $\mathbb{P}(p_\ell \leq \alpha/2) \leq \alpha/2$ and $\mathbb{P}(p_r \leq \alpha/2) \leq \alpha/2$. Hence $\mathbb{P}(\min(p_\ell, p_r) \leq \alpha/2) \leq \alpha/2 + \alpha/2 = \alpha$.

rates in the two conditions instead of the means. Explain how you could accomplish this using a randomisation test. Would you compute a left-sided, a right-sided or a two-sided p-value? ♢

12.3 On the meaning of p-values

Thanks to Lemma 12.8, we can obtain a valid p-value by computing the probability of observing the pattern we actually saw (in our example: a group difference of 0.6 syllables per second) *or an even more extreme pattern*, *assuming the null hypothesis is true*. In our example, we generated all alternative group differences under the assumption that the observed difference arose purely by chance. Other definitions or interpretations of the p-value risk being incorrect. To prevent some common misunderstandings:

- The p-value is *not* the probability that the null hypothesis is true. We cannot conclude that there is a 8.7% probability that H_0 holds.
- The p-value is also *not* the complement of the probability that the alternative hypothesis is true. In our example, we *cannot* conclude that H_A is true with 91.3% probability.
- The p-value is not the complement of the probability that the observed result would replicate in a new study. (I have no idea where this misunderstanding comes from, but I once saw it in a review report.)

These—and other—misunderstandings are surprisingly common, even in introductory statistics courses aimed at psychology or linguistics students. See Goodman (2008) and Greenland et al. (2016) for further discussion of common misinterpretations.

Often, researchers distinguish between 'statistically significant' and 'statistically non-significant' p-values. A p-value below a given threshold α is considered statistically significant. For a statistically significant p-value, researchers typically conclude that the data are unlikely under the null hypothesis, and therefore reject it in favour of the alternative. The significance threshold, denoted α, can in principle be chosen arbitrarily; in the social and human sciences, it is almost universally set at $\alpha = 0.05$—generally for no reason other than that a human hand has five fingers.

Tip 12.10. In statistics, 'significance' is a technical term and should not be confused with the everyday sense of practical or theoretical significance or importance. Try to avoid this ambiguity in your own writing. ♢

12.4 Type I and Type II Errors

As much as we might wish otherwise, significance tests do not offer certainty. In traditional null hypothesis significance testing (NHST), we distinguish between two types of potential mistakes.

The first type of mistake is rejecting a null hypothesis that is actually true. If you follow the conventional α threshold of 5%, you will incorrectly reject a true null hypothesis at

most 5% of the time—assuming the data are analysed correctly. This type of error is called a **Type I error** (or a false positive, i.e., finding something that isn't there). Because α is defined by the researcher—or, in practice, de facto set at 0.05—the frequency of Type I errors is, in principle, controlled: at most $100\alpha\%$ (i.e., de facto 5%) of all true null hypotheses would be wrongly rejected if the data are correctly analysed. In practice, however, complications can arise. Exercise 12.11 deals with one such complication; more subtle ones are discussed in Chapter 17.

Exercise 12.11 (Multiple testing)**.** A research team conducts an experiment in 30 school classes simultaneously. In each school class, the children are assigned to the control or experimental group using complete randomisation. That is, the researchers are in effect conducting 30 parallel and independent experiments.

Assume that in all of these classes, the null hypothesis is true. Further assume that, if the null hypothesis is true, there is exactly a 5% chance of observing a statistically significant difference between the control and experimental conditions.

What is the probability of obtaining a statistically significant difference in at least one of the 30 experiments? ♢

If H_0 is false (i.e., the pattern did not arise purely by chance), there is still the possibility of obtaining a p-value above the α threshold. In such cases, you would fail to reject the null hypothesis even though you ideally should. This type of mistake is called a **Type II error** (or a false negative, i.e., failing to detect something that is actually there). The probability of a Type II error is denoted β (not to be confused with the βs in regression models); its complement, $1 - \beta$, is called the statistical **power** of a test.

The probability β cannot be set arbitrarily because it depends on four factors:

1. The strength of the true effect if the null hypothesis does not hold (e.g., how much moderate alcohol consumption actually affects speech rate): the stronger the effect (i.e., the larger the difference), the higher the power.
2. The magnitude of the error variance (e.g., how much participants differ from each other within each group): the larger the error variance, the lower the power.
3. The sample size: the more data you have, the higher the power.
4. The choice of test and whether its assumptions are approximately met.

Power calculations (see Chapter 15) allow you to estimate the power of a significance test.

Remark 12.12 (Interpreting non-significant p-values)**.** Due to the possibility of a Type II error, a non-significant result does not allow you to conclude that there is a difference nor that there is *no* difference: it is always possible that you simply failed to detect the difference. If you read somewhere that "A and B did not differ significantly and are therefore the same (or 'essentially the same')", this is usually just convenient rhetoric: absence of evidence is not evidence of absence. Schmidt (1996) refers to this fallacy as "the most devastating of all to the research enterprise" (p. 126). ♢

Exercise 12.13. In a research field, a large number of experiments are conducted. In 45% of these, the null hypothesis is true, and in 55%, the alternative hypothesis is true. In 5% of the experiments in which the null hypothesis is true, this null hypothesis is incorrectly rejected; in 60% of the experiments in which the alternative hypothesis is true, the null hypothesis is correctly rejected.

We randomly select an experiment in which the null hypothesis was rejected. What is the probability that, in this experiment, the null hypothesis was rejected correctly? ♢

12.5 Randomisation tests for group differences

Randomisation tests have the advantage of making few assumptions. In the calculations above, our only real assumption was that participants were assigned to conditions at random. In practice, however, it is usually too cumbersome to enumerate all possible assignments. Instead, one typically generates only m random assignments. To do this, one can randomly permute the group labels using `sample()`.

To compute the left-sided p-value, count the number of assignments ℓ in which the observed pattern is at most as large as in the actual data. Then calculate

$$p_\ell = \frac{\ell + 1}{m + 1}.$$

In other words, we include the actually observed pattern among the randomly generated ones. We do so because it could also have arisen purely by chance. This also ensures that $p_\ell \neq 0$.

To compute the right-sided p-value, count the number of assignments r in which the observed pattern is at least as large as in the actual data. Then calculate

$$p_r = \frac{r + 1}{m + 1}.$$

For the two-sided p-value, calculate

$$p = 2 \cdot \min\{p_\ell, p_r\}.$$

While this procedure produces valid p-values, the specific result depends on the random assignments generated: different random draws may give different results. The larger m is, the smaller the randomness.

Below is an example using the data from Klein et al. (2014), which we already analysed in Chapter 9. To illustrate that the procedure isn't limited to mean differences, we analyse the difference in the medians.

```
klein <- read_csv(here("data", "Klein2014_money_abington.csv"))
median_difference <- tapply(klein$Sysjust, klein$MoneyGroup, median) |>
  diff()
median_difference
```

```
treatment
    0.375

m <- 19999
median_diffs_H0 <- replicate(m, {
  permuted_group <- sample(klein$MoneyGroup)
  tapply(klein$Sysjust, permuted_group, median) |> diff()
})
l <- sum(median_diffs_H0 <= median_difference)
r <- sum(median_diffs_H0 >= median_difference)
(p_l <- (l + 1) / (m + 1))

[1] 0.88245

(p_r <- (r + 1) / (m + 1))

[1] 0.1719

(p <- 2 * min(p_l, p_r))

[1] 0.3438
```

Exercise 12.14 (Difference between more than two group means)**.** In Section 9.2, we saw how to model differences between more than two groups. We can now test the null hypothesis that the differences between the three condition means in the study by Vanhove (2019) are purely due to chance. Read the data in again and calculate the three condition means. Instead of computing the difference between these means, this time compute the variance of the three means. (It does not matter whether you calculate the sample variance or the population variance. You could also compute the standard deviation.) Adapt the code section above to test this null hypothesis and calculate an appropriate p-value.

Hint: Use a right-sided p-value. (Why?)

Incidentally, when comparing just two groups, one could also compute the variance (or standard deviation) of the statistics to be compared rather than their difference. If a two-sided test is used, this will give the same result, because the variance (or standard deviation) is a strictly monotonically increasing function of the absolute difference. ♢

Exercise 12.15 (Experiment with blocking)**.** We conduct a pretest–posttest experiment with random assignment. Instead of allocating the $2n$ participants completely at random to the two groups, we first order them according to their pretest scores and then form pairs as follows:

$$(1,2)(3,4)(5,6)(7,8)\ldots(2n-1,2n).$$

Within each pair, we randomly assign one participant to the control group and the other to the experimental group. This technique is called **blocking**. It can increase the power of an experiment and is underused in our field.

1. There are $\binom{2n}{n}$ possible ways of dividing $2n$ participants into two equally sized groups. How many possible allocations are there in an experiment with $2n$ participants if a blocking factor groups the participants into pairs?

2. Suppose the null hypothesis states that the group means differ only due to chance. Explain in words (but precisely!) how this null hypothesis can be tested using a randomisation test. ♢

Exercise 12.16 (Experiment with school classes)**.** For an experiment with two conditions, sixteen school classes of twenty pupils each are recruited. Among others, the following methods suggest themselves for assigning pupils to conditions:

1. All 320 pupils are allocated to the two conditions by complete randomisation. That is, 160 children are randomly assigned to the control group and 160 to the experimental group. The class they belong to is ignored, so it is possible that in a given class more children end up in the experimental than in the control condition.

2. Within each class, complete randomisation is used. In every class there are therefore ten pupils in the control condition and ten in the experimental condition.

3. The classes themselves are assigned to conditions. That is, all the children from eight classes are placed in the control condition, and all the children from the other eight classes are placed in the experimental condition.

For each of the three methods, calculate how many possible allocations there are. Also describe, for each method, a suitable randomisation test for comparing the group means. ♢

Example 12.17 (Wilcoxon rank-sum test)**.** In Exercise 9.4 on page 208 you examined a dataset on the *gambler's fallacy*. As you noticed there, the variable `RollsImagined` is strongly skewed. Instead of comparing means, we could compare medians, but here I'd like to introduce an alternative. An intuitive way of capturing whether the values in one group tend to be higher than those in the other is to randomly draw one observation from each group and calculate the probability that the first observation is higher than the second.

The actual computation of this probability is conceptually straightforward: if there are n_1 observations in the first group and n_2 in the second, then we carry out all $n_1 n_2$ pairwise comparisons. We then calculate the proportion of comparisons in which the first value is greater than the second; in case of a tie, we count the comparison as half a success. The home-made function `cles()` performs this calculation; its name refers to the term **common-language effect size** (McGraw & Wong, 1992), though this statistic is more commonly known as the **probability of superiority**:

```
# ufl data only; ignore NAs
d <- read_csv(here("data", "Klein2014_gambler.csv")) |>
  filter(Sample == "ufl") |>
  filter(!is.na(RollsImagined))
```

```
# split observations by condition
x1 <- d$RollsImagined[d$Condition == "three6"]
x2 <- d$RollsImagined[d$Condition == "two6"]

# load cles() and compute
source(here("functions", "cles.R"))
cles(x1, x2)
```

```
[1] 0.665254
```

If we draw one observation from the `three6` group and one from the `two6` group, then the first has a 66.5% chance of being larger than the second. With a randomisation test we could now test the null hypothesis that this figure deviates from 50% only by chance. The practical problem is that the calculation implemented in `cles()` is fairly slow: if we were to run thousands of rerandomisations and apply `cles()` each time, we would be waiting a very long time for the result.

Fortunately, it turns out that there is a less intuitive but computationally more efficient way of performing the calculation, which in the end gives the same result. We combine the observations from both groups into a single vector, convert them to ranks, and sum the ranks of the first group. For our data, this rank sum is $R = 3987.5$:

```
rank_sum <- function(x, y) {
  ranks <- rank(c(x, y))
  sum(ranks[1:length(x)])
}
rank_sum(x1, x2)
```

```
[1] 3987.5
```

For fixed group sizes n_1, n_2 the `cles` value can be calculated directly from the rank sum. This is useful computationally as it is faster to compute the rank sum than to carry out the necessary $n_1 n_2$ comparisons.

```
n1 <- length(x1)
n2 <- length(x2)
(rank_sum(x1, x2) - n1*(n1 + 1)/2) / (n1 * n2)
```

```
[1] 0.665254
```

We now carry out a randomisation test on the rank sum. The logic is exactly the same as before.

```
m <- 19999
rank_sums_H0 <- replicate(m, {
  permuted_group <- sample(d$Condition)
  x_H0 <- d$RollsImagined[permuted_group == "three6"]
  y_H0 <- d$RollsImagined[permuted_group == "two6"]
  rank_sum(x_H0, y_H0)
```

```
})
l <- sum(rank_sums_H0 <= rank_sum(x1, x2))
r <- sum(rank_sums_H0 >= rank_sum(x1, x2))
p_l <- (l + 1) / (m + 1)
p_r <- (r + 1) / (m + 1)
(p <- 2 * min(p_l, p_r))

[1] 0.0029
```

So we obtain $p = 0.003$.

This randomisation test corresponds to the **Wilcoxon rank-sum test**, which is implemented in R as `wilcox.test()`. In practice, however, this test usually relies on an approximation (see `Details` in `?wilcox.test`):

```
wilcox.test(x1, x2)


 Wilcoxon rank sum test with continuity correction

data:  x1 and x2
W = 2276, p-value = 0.00201
alternative hypothesis: true location shift is not equal to 0
```

The output shows a statistic W (for Wilcoxon), but this is *not* the rank sum. Instead, it is in fact the test statistic of the equivalent **Mann–Whitney test**, usually denoted U in the statistical literature. This statistic has a simpler relationship to the `cles` value:

```
cles(x1, x2) * (n1 * n2) # CLES to U

[1] 2276.5

wilcox.test(x1, x2)$statistic / (n1 * n2) # U to CLES

       W
0.665254
```

◇

***Exercise 12.18** (Confidence interval for the probability of superiority)**.** Use a nonparametric bootstrap to compute a 95% confidence interval for the probability of superiority computed in Remark 12.17.

Hint: Compute the rank sum or the U statistic and convert it to the probability of superiority/`cles` value. Using `cles()` would take too much time. ◇

12.6 Permutation tests for associations

To compute p-values for patterns other than group differences, we can use procedures very similar to randomisation tests. These are called **permutation tests**; because of their

similarity, the two terms are often used interchangeably. On page 200 you calculated the correlation coefficient for the relationship between two indicators of cognitive control in a sample of 34 participants (data from Poarch et al., 2019).

```
poarch <- read_csv(here("data", "poarch2018.csv"))
cor_poarch <- cor(poarch$Flanker, poarch$Simon)
cor_poarch
```

```
[1] 0.462355
```

The procedure for calculating p-values is always essentially the same, and follows the template we can use thanks to Lemma 12.8: we choose a test statistic, generate its distribution under the null hypothesis, and then compute p_ℓ, p_r, or a two-sided p-value. When calculating a p-value for a correlation coefficient, the task is to work out the probability of observing the correlation we found in the sample, or an even stronger one, if in the population there were in fact no relationship at all between the two variables. In other words, the null hypothesis tested is that the two variables are independent.

To calculate this probability, we first need to generate the distribution of the correlation coefficient we would expect in this sample if the two variables (`Flanker` and `Simon`) were independent (H_0). To this end, we can permute the observed values of one variable, i.e. shuffle them randomly, without permuting the observed values of the other variable. It doesn't matter which variable you permute. This breaks the systematic association between the two variables. Figure 12.4 shows the observed association alongside two possible associations that could arise if we randomly shuffle the Simon variable. The R code below shows how you can produce this figure yourself.

```
p1 <- ggplot(poarch,
             aes(x = Flanker,
                 y = Simon)) +
  geom_point(shape = 1) +
  ggtitle("Observed data")

# If no further parameters are set in sample(),
# the input is permuted at random.
p2 <- ggplot(poarch,
             aes(x = Flanker,
                 y = sample(Simon))) +
  geom_point(shape = 1) +
  ggtitle("One possible permutation")

p3 <- ggplot(poarch,
             aes(x = Flanker,
                 y = sample(Simon))) +
  geom_point(shape = 1) +
  ggtitle("Another possible permutation")
```

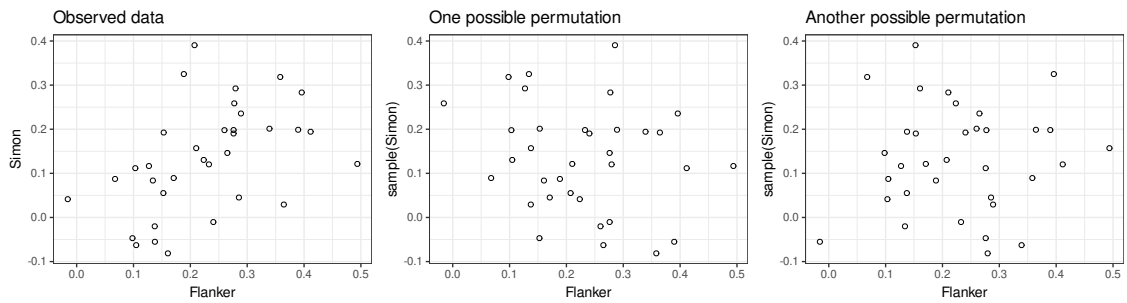


Figure 12.4: Left: The observed association between Flanker and Simon in the study by Poarch et al. (2018). Centre and right: By permuting one variable (here: Simon) independently of the other, the systematic association between them is broken. This corresponds to the null hypothesis. To see how the correlation coefficients are distributed under the null hypothesis, one can repeat this permutation a few thousand times and recalculate the correlation each time.

```
library(patchwork) # arrange plots side by side
p1 + p2 + p3
```

With 34 observations there are almost 300 sextillion (a 3 followed by 38 zeros) possible permutations of the Simon variable:

$$34! = 34 \cdot 33 \cdot 32 \cdot \cdots \cdot 3 \cdot 2 \cdot 1 \approx 2.95 \cdot 10^{38}.$$

It is impossible to compute the correlation coefficient for all of these. So instead we settle for 19,999 permutations:

```
m <- 19999
cor_H0 <- replicate(m, {
  cor(poarch$Flanker, sample(poarch$Simon))
})
```

The distribution of correlation coefficients under the null hypothesis can be shown in a histogram; Figure 12.5 shows a slightly more elaborate version. Strictly speaking, the actually observed correlation coefficient must also be included in this distribution, since it too could have arisen under the null. In total, then, the distribution contains 20,000 correlation coefficients.

```
df_cor_H0 <- tibble(cor_H0) |>
  add_row(cor_H0 = cor_poarch)
ggplot(df_cor_H0,
       aes(cor_H0)) +
  geom_histogram(fill = "grey", colour = "black",
                 breaks = seq(-1, 1, 0.05)) +
  geom_vline(xintercept = cor_poarch, linetype = 2) +
```

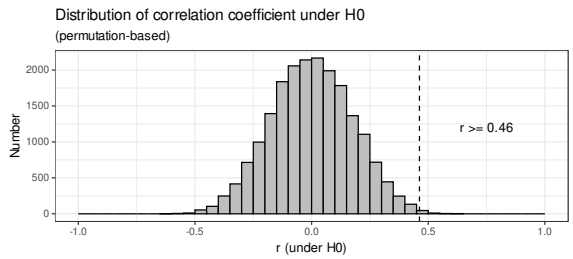


Figure 12.5: The distribution of the correlation coefficient for this sample with 34 obsevations when permuting one of the variables randomly.

```
ggtitle("Distribution of correlation coefficient under H0",
        subtitle = "(permutation-based)") +
xlab("r (under H0)") +
ylab("Number") +
annotate("text", x = 0.75, y = 1200, label = "r >= 0.46")
```

From this distribution we can read off how unusual correlation coefficients of $r \geq 0.46$ would be if there were no systematic association between the two variables. For the one-sided test (H_A: the correlation is positive), we look at the proportion of correlation coefficients generated under H_0 that are equal to 0.46 or larger:

```
r <- sum(df_cor_H0$cor_H0 >= cor_poarch)
(p_r <- (r + 1) / (m + 1))

[1] 0.00265
```

That is, 0.26% ($p = 0.0026$). If you run these calculations yourself, you will find a slightly different result. This is because your 19,999 random permutations won't be the same as mine.

For the two-sided test (H_A: the correlation is not equal to 0, but could be either positive or negative), we also need to look at how many of the correlation coefficients generated under H_0 are equal to 0.46 or smaller.

```
l <- sum(df_cor_H0$cor_H0 <= cor_poarch)
p_l <- (l + 1) / (m + 1)
(p <- 2 * min(p_l, p_r))

[1] 0.0053
```

So 0.5% ($p = 0.005$). The probability of observing a correlation this strong, or stronger, if the null hypothesis were in fact true, is therefore rather small. (Of course, we could have skipped the calculation of `p_l`, since this value is clearly larger than `p_r`.)

Exercise 12.19. Compute Kendall's τ for the association between the Flanker and Simon data. Test the null hypothesis that it differs from 0 only by chance using a self-written permutation test. ♢

12.7 The binomial test

In the final two sections of this chapter, we take a closer look at two more applications of the p-value template: the binomial test and Fisher's exact test.

Example 12.20 (ABX tests)**.** We want to find out whether a learner of English can tell the difference between the phonemes /ɛ/ (as in *bet*) and /æ/ (as in *bat*). To this end, we set up an ABX test: we compile a list of 30 minimal pairs whose components differ only in these phonemes (*bed–bad*, *fed–fad*, *peck–pack*, and so on). In total, 60 words are recorded twice each by a native speaker. For each minimal pair, the learner hears one recording of the /ɛ/-word and one of the /æ/-word, in random order. Afterwards, the learner hears a second recording of one of the two words. Their task is to decide whether the final recording ('X') belongs to the same category as the first ('A') or the second ('B') recording. The learner classifies 21 out of 30 recordings correctly ($X = 21$). This could suggest that they are, in some sense, able to distinguish /ɛ/ from /æ/. But we should also consider the possibility that they were just guessing.

If the learner were simply guessing, then for each minimal pair there would be a probability of $p_0 = 0.5$ of being correct—whether by randomly picking 'A' or 'B', or by using a simple strategy such as alternating between the two. In this case, the correctness of one answer would tell us nothing about the next, i.e., the answers would be independent. So the null hypothesis can be stated as:

$$H_0 : X \sim \text{Binomial}(30, 0.5).$$

The usual alternative hypothesis would simply be that the success probability is not equal to $p_0 = 0.5$. Here, however, we are only interested in the alternative that the success probability is *greater* than $p_0 = 0.5$.

We can test this null hypothesis with a **binomial test**. Figure 12.6 shows the Binomial$(30, 0.5)$ distribution. We can now read off, or easily calculate, how probable it would be to observe at least $X = 21$:

$$\mathbb{P}(X \geq 21) = 1 - \mathbb{P}(X \leq 20),$$

which we can compute with `pbinom()`:

```
1 - pbinom(20, 30, 0.5)

[1] 0.021387
```

A one-sided binomial test therefore gives a right-tailed p-value of about 0.02.

With `binom.test()` we get the same result:

```
binom.test(21, 30, 0.5, alternative = "greater")


 Exact binomial test

data:  21 and 30
number of successes = 21, number of trials = 30,
```

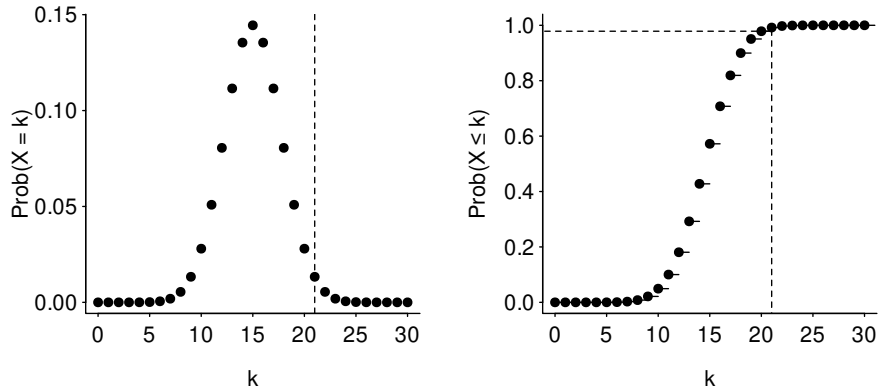


Figure 12.6: Left: Probability of k successes under the Binomial(30, 0.5) distribution. Right: Cumulative distribution function of the same distribution.

```
p-value = 0.0214
alternative hypothesis: true probability of success is greater than 0.5
95 percent confidence interval:
 0.534927 1.000000
sample estimates:
probability of success
                   0.7
```

If we use a significance threshold of $\alpha = 0.05$, we would conclude that the learner was not just lucky and is, to some extent, able to perceive the difference between the two phonemes. ♢

The binomial test from the example above is an exact test for comparing a binomial parameter with a hypothetical reference value (in this case $p_0 = 0.5$). 'Exact' here means that the requirement in Definition 12.5 is met. In the example, a right-tailed test was used, since we were only interested in the alternative that the success probability is greater than p_0. If we were only interested in the alternative that the success probability is less than p_0, then a left-tailed test would be appropriate. For this, we would calculate $\mathbb{P}(X \leq x)$, where x is the observed number of successes.

For a two-sided test, we can use Lemma 12.8 to calculate a valid p-value as

$$2 \cdot \min\{\mathbb{P}(X \leq x), \mathbb{P}(X \geq x)\}.$$

It is, however, possible to derive a different valid p-value for the two-sided test that gives the test more power. Let X be the random variable representing the number of successes in n trials, and let p_0 be the probability of success under H_0. Now define the random variable

$$\widetilde{X} := \mathbb{P}(Y = X),$$

where $Y \sim \text{Binomial}(n, p_0)$, with X, Y independent. For the observed number of successes x, we accordingly calculate the value $\widetilde{x} := \mathbb{P}(Y = x)$. With Lemma 12.8 we obtain a valid p-value by calculating $F_0(\widetilde{x})$. Here, F_0 is the distribution of the random variable $\widetilde{X}$ under the null hypothesis.

Example 12.21 (Two-sided binomial test)**.** Suppose the null hypothesis is

$$X \sim \text{Binomial}(17, 1/3)$$

and we observe three successes. We can now calculate a first valid two-sided p-value as follows:

```
p_l <- pbinom(3, 17, 1/3)
p_r <- 1 - pbinom(2, 17, 1/3)
(2 * min(p_l, p_r))

[1] 0.260845
```

However, this test has less statistical power than the slightly more complicated alternative. For this, we first calculate $\mathbb{P}(Y = k)$ for each $k = 0, \ldots, 17$:

```
prob_k <- dbinom(0:17, 17, 1/3)
```

The distribution function F_0 of the random variable $\widetilde{X}$ at a value r corresponds to the sum of probabilities $\mathbb{P}(Y = k)$ with $\mathbb{P}(Y = k) \leq r$; see Figure 12.7. We are interested in $F_0(\mathbb{P}(Y = 3))$ and thus obtain

```
sum(prob_k[prob_k <= prob_k[4]])

[1] 0.205897
```

An easier way is to use `binom.test()`:

```
binom.test(3, 17, 1/3)$p.value

[1] 0.205897
```

$\diamondsuit$

Binomial tests admittedly do not occur all that often. However, they nicely illustrate how significance tests can be carried out when the distribution of the outcome of a trial or an experiment under the null hypothesis is known: You simply calculate how unusual the observed result, or even more extreme results, would be under this distribution. In the preceding sections we estimated the distributions under the null hypothesis through brute-force computation; here, by contrast, we could use one of the classic probability distributions.

12.8 Fisher's exact test

To conclude this chapter, we take a closer look at the logic of Fisher's exact test.

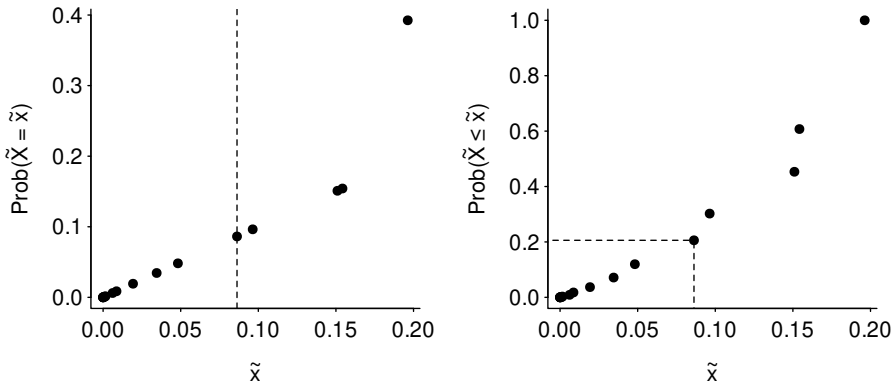


Figure 12.7: We define $\widetilde{X} := \mathbb{P}(Y = X)$, where X, Y are i.i.d. Binomial(17, 1/3) distributed. The left panel shows the probabilities $\mathbb{P}(\widetilde{X} = \widetilde{x})$; the right panel the distribution function of $\widetilde{X}$.

Definition 12.22 (Hypergeometric distribution)**.** Consider an urn that contains m white balls and n black balls. We randomly draw $k \leq m + n$ balls from this urn without replacement. Then the number of white balls drawn (w) follows a **hypergeometric distribution** with parameters m, n, k.[4] We write $w \sim \text{Hypergeometric}(m, n, k)$. ♢

Example 12.23 (Classification task)**.** Once again, we want to investigate whether a learner of English perceives the difference between the phonemes /ɛ/ (as in *bet*) and /æ/ (as in *bat*). This time we set up a different experiment. We provide the learner with a total of 30 audio files. Twenty of these files are recordings of the word *bet*, the remaining ten are recordings of the word *bat*. The learner's task is to select those twenty audio files that are recordings of *bet*. The learner *must* select exactly twenty files. The results are summarised in a contingency table:

	Classified as *bet*	Classified as *bat*	Row total
Is *bet*	$n_{11} = 15$	$n_{12} = 5$	$n_{1+} = 20$
Is *bat*	$n_{21} = 5$	$n_{22} = 5$	$n_{2+} = 10$
Column total	$n_{+1} = 20$	$n_{+2} = 10$	$n_{++} = 30$

An important feature of this experiment is that the row and column totals were fixed in advance: Exactly $n_{1+} = 20$ of the 30 files are recordings of *bet*, and exactly $n_{+1} = 20$ of the files had to be classified as *bet*. Once one of the entries $n_{11}, n_{12}, n_{21}, n_{22}$ is known, the other three are determined. The random variable X that we consider is the n_{11} entry. Under the null hypothesis, namely that the learner merely selected 20 files at random from the 30, this random variable is hypergeometrically distributed with parameters n_{1+}, n_{2+} and n_{+1}:

$$H_0 : X \sim \text{Hypergeometric}(n_{1+}, n_{2+}, n_{+1}).$$

[4]Depending on the reference work, this distribution is parameterised differently. Here I use the parameterisation used in R.

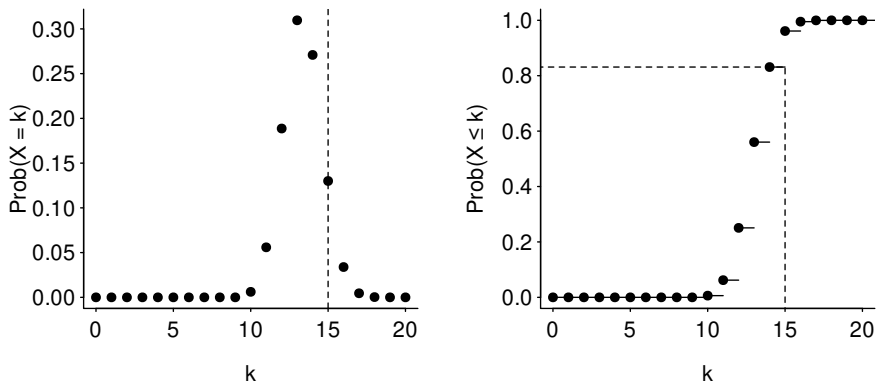


Figure 12.8: Left: Probability of $n_{11} = k$ for $n_{11} \sim$ Hypergeometric$(20, 10, 20)$. Right: The corresponding distribution function.

If the learner really is able to distinguish *bet* from *bat*, we would expect n_{11} to be larger rather than smaller. Accordingly, we calculate a right-sided p-value here, exactly as with the binomial test.

Figure 12.8 shows the Hypergeometric(20, 10, 20) distribution. We can now read off, or simply calculate, how likely an n_{11} entry of at least 15 would be under this distribution:

```
1 - phyper(15 - 1, 20, 10, 20)

[1] 0.168747
```

We obtain the same result by passing the observed table to the `fisher.test()` function:

```
fisher.test(
  rbind(c(15, 5),
        c(5, 5)),
  alternative = "greater"
)$p.value

[1] 0.168747
```

Incidentally, the same result is obtained if we focus on the categorisation of the *bat* recordings and adjust the parameters of the hypergeometric distribution accordingly, since it also holds that

$$H_0 : n_{22} \sim \text{Hypergeometric}(n_{2+}, n_{1+}, n_{+2}) :$$

```
1 - phyper(5 - 1, 10, 20, 10)

[1] 0.168747
```

```
fisher.test(
  rbind(c(5, 5),
        c(5, 15)),
  alternative = "greater"
)$p.value
```

```
[1] 0.168747
```

$\diamondsuit$

For the two-sided test we can proceed in much the same way as with the binomial test: We define the random variable

$$\widetilde{X} := \mathbb{P}(Y = X),$$

where $Y \sim \text{Hypergeometric}(n_{1+}, n_{2+}, n_{+1})$ with X, Y independent, and calculate $\widetilde{x} := \mathbb{P}(Y = n_{11})$. The p-value is then given by $F_0(\widetilde{x})$, where F_0 denotes the distribution function of the random variable $\widetilde{X}$ under the null hypothesis. If the `alternative` parameter of the `fisher.test()` function is left unspecified, this value is computed.

Fisher's exact test is called 'exact' because the p-values it produces satisfy the condition in Definition 12.5 precisely. Its actual scope of application is rather limited, since it is extremely rare for both the row and column totals of the contingency table to have been fixed in advance of data collection. Nevertheless, this test is often used to evaluate contingency tables for which only (for example) the row totals were fixed beforehand, or for which neither the row nor the column totals were predetermined. In such cases the test is still exact (in the sense of the definition), but it can be rather **conservative**: other tests (e.g., **Boschloo's test**) are also exact depending on the research design, but have greater power. The literature on the analysis of seemingly simple 2×2 contingency tables is surprisingly extensive; for an overview, see Fagerland et al. (2017). In the blog post *Exact significance tests for* 2×2 *tables* (10 September 2024), I present Boschloo's test.

Chapter 13

The t-test

Although randomisation and permutation tests are valid significance tests whose assumptions can be justified by the research design in controlled experiments, you rarely encounter them in practice. This is largely down to historical reasons and inertia. Back in the day, it was far too labour-intensive to grind through a few thousand permutations of the data in order to generate the distribution of a group difference or a correlation coefficient under the null hypothesis. Instead, statisticians developed significance tests whose mathematical derivations were more intricate but which could be carried out much more quickly. Examples include the t-test and the F-test. The reason these tests became so popular is that their results often agree with those of randomisation and permutation tests:

> "the statistician does not carry out this very simple and very tedious process [= permuting data], but his conclusions have no justification beyond the fact that they agree with those which could have been arrived at by this elementary method." (Fisher, 1936)

In what follows, we introduce the t-test, followed by analysis of variance and the F-test. That said, especially in straightforward controlled experiments such as those in the previous chapter, I would not automatically fall back on these pre-computer approximations, but would instead seriously consider a randomisation or permutation test.

The randomisation tests in Chapter 12 assume that any observed differences, under the null hypothesis, are the result of random assignment. The t-test takes a different angle: here the assumption is that any observed differences, under the null hypothesis, can be attributed to random sampling from a population. More precisely, the **one-sample t-test** is based on the null hypothesis that we have n i.i.d. observations $X_1, \ldots, X_n$, drawn from a normal distribution with known mean μ and unknown variance σ^2. As we have already seen (Theorem 6.12 on page 141), under these assumptions it holds that

$$T := \frac{\overline{X} - \mu}{s/\sqrt{n}} \sim t_{n-1},$$

where $\overline{X}$ is the sample mean of the n observations and s the sample standard deviation. We can now test whether the observed data are compatible with a particular population

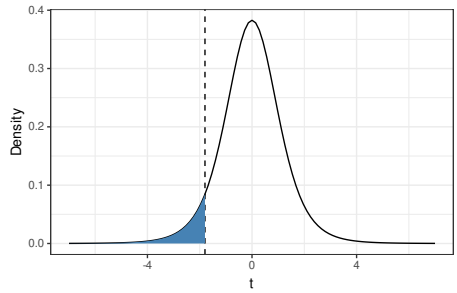


Figure 13.1: A probability density function of the $t(6)$ distribution. The dashed line shows the observed t value of -1.80. The area of the coloured patch underneath the density function corresponds to the left-sided p value corresponding to $t(6) = -1.80$.

mean μ_0. Concretely, we calculate

$$T_0 := \frac{\overline{X} - \mu_0}{s/\sqrt{n}}$$

and then check how extreme this value is under a t_{n-1} distribution.

Example 13.1 (One-sample t-test)**.** We observe seven values with a sample mean of $\overline{X} = -2.4$ and a sample standard deviation of $s = 5$. We set up the null hypothesis that these seven values were randomly drawn from a normal distribution with mean $\mu_0 = 1$ and unknown variance. The t-value for this null hypothesis is

$$\frac{-2.4 - 1}{5/\sqrt{7}} \approx -1.80.$$

For a two-sided t-test, we first calculate the left- and right-tailed p-values. In other words, we look up how often we would encounter values of -1.80 or smaller, and values of -1.80 or larger, under a $t(6)$ distribution; see Figure 13.1. Using Lemma 12.8, we obtain the two-sided p-value by doubling the smaller of these two probabilities:

```
t0 <- (-2.4 - 1)/(5/sqrt(7))
(p_l <- pt(t0, df = 7 - 1))

[1] 0.0610513

(p_r <- pt(t0, df = 7 - 1, lower.tail = FALSE))

[1] 0.938949

(p <- 2 * min(p_l, p_r))

[1] 0.122103
```

That is, we obtain a two-sided p-value of about 0.12. ♢

You can carry out a one-sample t-test using the R function `t.test()`:

```
x <- rnorm(10, mean = 2, sd = 4)
t.test(x, mu = 1) # tests if the mean of the normal distribution is 1


 One Sample t-test

data:  x
t = 1.293, df = 9, p-value = 0.228
alternative hypothesis: true mean is not equal to 1
95 percent confidence interval:
 -0.268102  5.650306
sample estimates:
mean of x
   2.6911
```

The above procedure can be adapted so that it is suitable for comparing two group means. This results in the **two-sample t-test**. The null hypothesis here is that we drew two independent samples:

- $\boldsymbol{X}_1 = (X_{1,1}, \ldots, X_{1,n_1})$ where

  $$X_{1,1}, \ldots, X_{1,n_1} \sim \mathcal{N}(\mu_1, \sigma^2).$$

  We denote the sample mean and sample variance of $\boldsymbol{X}_1$ as $\overline{X}_1$ and s_1^2, respectively.

- $\boldsymbol{X}_2 = (X_{2,1}, \ldots, X_{2,n_2})$ where

  $$X_{2,1}, \ldots, X_{2,n_1} \sim \mathcal{N}(\mu_2, \sigma^2).$$

  We denote the sample mean and sample variance of $\boldsymbol{X}_2$ as $\overline{X}_2$ and s_2^2, respectively.

That is, the samples are assumed to have been drawn from normal distribution with the same unknown variance σ^2.

Under this assumption, the difference $\overline{X}_1 - \overline{X}_2$ is normally distributed, too. The mean of this normal distribution is

$$\mathbb{E}(\overline{X}_1 - \overline{X}_2) = \mathbb{E}(\overline{X}_1) - \mathbb{E}(\overline{X}_2) = \mu_1 - \mu_2.$$

Since the samples are independent, its variance is

$$\begin{aligned}
\mathrm{Var}(\overline{X}_1 - \overline{X}_2) &= \mathrm{Var}(\overline{X}_1) + \mathrm{Var}(\overline{X}_2) \\
&= \frac{\sigma^2}{n_1} + \frac{\sigma^2}{n_2} \\
&= \sigma^2 \left(\frac{1}{n_1} + \frac{1}{n_2} \right).
\end{aligned}$$

Hence,

$$\overline{X}_1 - \overline{X}_2 \sim \mathcal{N}\left(\mu_1 - \mu_2, \sigma^2 \left(\frac{1}{n_1} + \frac{1}{n_2} \right) \right).$$

The common variance σ^2 can be estimated from the sample variances s_1^2, s_2^2 as

$$s^2 = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}.$$

(By taking the expected value on both sides, you can verify that $\mathbb{E}(s^2) = \sigma^2$.) This statistic is called the **pooled variance**. It can be shown that

$$T := \frac{(\overline{X}_1 - \overline{X}_2) - (\mu_1 - \mu_2)}{s\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \sim t_{n-2}.$$

We'll skip the derivation of this result, but the next example illustrates it by means of a simulation. In short, for any hypothetical difference between the population means, we can test whether the data are compatible with that difference.

Example 13.2 (Simulation)**.** The code snippet below generates 20,000 simulations of two samples with three and five observations, respectively. These samples are drawn from the same normal distribution, hence $\mu_1 - \mu_2 = 0$. The t value is computed using the `t.test()` function. The parameter setting `var.equal = TRUE` specifies that we assume that the samples are drawn from normal distributions with the same variance.

```
n_runs <- 20000
ttest_one_run <- function(n1, n2, sd1, sd2) {
  x1 <- rnorm(n1, mean = 0, sd = 3)
  x2 <- rnorm(n2, mean = 0, sd = 3)
  t.test(x1, x2, var.equal = TRUE)$statistic
}
n1 <- 3
n2 <- 5
t_values <- replicate(n_runs, ttest_one_run(n1, n2, 3, 3))
```

If we visualise these 20,000 t values, we see that their distribution corresponds to a t distribution with $3 + 5 - 2 = 6$ degrees of freedom; see Figure 13.2.

```
t_values |>
  tibble() |>
  ggplot(aes(t_values)) +
  stat_ecdf(colour = "darkblue") +
  stat_function(fun = pt, args = list(df = n1 + n2 - 2),
                colour = "red", linetype = "dashed") +
  xlab("t") +
  ylab("cumulative probability")
```

$\diamondsuit$

Example 13.3 (Two-sample t-test)**.** We observe two samples. Our null hypothesis is that these are independently drawn from normal distributions with the same unknown variance and with means such that $\mu_1 - \mu_2 = -2$. For the first sample, we obtain $n_1 = 7, \overline{X}_1 = 3.4, s_1^2 = 4$. For the second sample, we obtain $n_2 = 10, \overline{X}_2 = 9, s_2^2 = 5$.

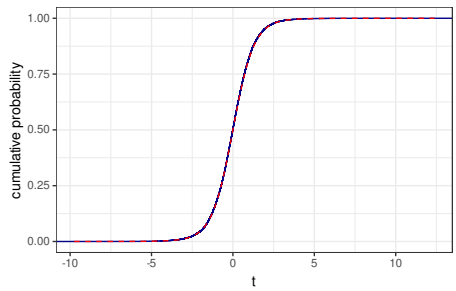


Figure 13.2: The blue curve shows the empirical distribution function of the 20,000 simulated t values; the red dashed curved shows the distribution function of the t_6 distribution.

We compute the pooled variance

$$s^2 = \frac{(7-1)4 + (10-1)5}{7+10-2} = 4.6$$

and the t value

$$T = \frac{3.4 - 9 - (-2)}{\sqrt{S^2}\sqrt{\frac{1}{7} + \frac{1}{10}}} \approx -3.41.$$

Now we compare this t value to a t_{15} distribution:

```
pooled_var <- ((7 - 1)*4 + (10 - 1)*5) / (7 + 10 - 2)
t0 <- (3.4 - 9 - (-2)) / (sqrt(pooled_var) * sqrt(1/7 + 1/10))
p_l <- pt(t0, df = 7 + 10 - 2)
p_r <- pt(t0, df = 7 + 10 - 2, lower.tail = FALSE)
(p <- 2 * min(p_l, p_r))

[1] 0.00390913
```

That is, $p = 0.004$. If we use threshold of $\alpha = 0.05$, then we'd reject the null hypothesis that these samples are drawn from normal distributions with the same variance and whose means differ by two units. ♢

Remark 13.4 (t-test as a linear model)**.** The code snippet below shows three equivalent ways to carry out a two-sample t-test.

```
n1 <- 12
n2 <- 28
group <- rep(c("A", "B"), times = c(n1, n2))
score <- c(rnorm(n = n1, mean = 3, sd = 1),
           rnorm(n = n2, mean = 4, sd = 1))
d <- tibble(group, score)

# t.test(x1, x2, ...)
t.test(score[group == "A"], score[group == "B"],
```

```
       mu = 0, var.equal = TRUE)


 Two Sample t-test

data:  score[group == "A"] and score[group == "B"]
t = -3.423, df = 38, p-value = 0.0015
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.83386 -0.47085
sample estimates:
mean of x mean of y
  3.11090   4.26326
```

```
# t.test(outcome ~ predictor, ...)
t.test(score ~ group, data = d, mu = 0, var.equal = TRUE)
```

```


 Two Sample t-test

data:  score by group
t = -3.423, df = 38, p-value = 0.0015
alternative hypothesis: true difference in means between group A and group B is not equal
95 percent confidence interval:
 -1.83386 -0.47085
sample estimates:
mean in group A mean in group B
        3.11090         4.26326
```

```
# lineares Modell
mod.lm <- lm(score ~ group, data = d)
summary(mod.lm)$coefficients
```

```
            Estimate Std. Error  t value    Pr(>|t|)
(Intercept)  3.11090   0.281659 11.04491 2.00850e-13
groupB       1.15236   0.336647  3.42304 1.49677e-03
```

The upshot is that you can think of a two-sample t-test as a linear model that only models a group difference and that assumes that the errors are i.i.d. normal.

By the way, a one-sample t-test corresponds to a linear model in which only the intercept is modelled and that assumes that the errors are i.i.d. normal. ♢

Exercise 13.5 (Different variances)**.** In Example 13.2, increase the standard deviation of the normal distribution from which the larger sample is drawn from 3 to 9. Compare the simulated distribution function with that of a t_6 distribution. What do you notice? In this situation, would p-values based on a t_6 distribution be correctly calibrated, too large, or too small?

Now reduce the standard deviation for the larger sample to 1. What do you observe this time?

Summarise your findings.

Hint: If the conclusion is not immediately clear, try making the larger group even larger. ♢

Remark 13.6 (Welch's t-test)**.** Exercise 13.5 highlights that the null hypothesis of the two-sample t-test assumes equality of the population variances. However, it is also possible to test the null hypothesis that the two samples come from normal distributions that may have different variances σ_1^2, σ_2^2, but whose means differ by a hypothetical value. For this, one can use **Welch's t-test**. In Welch's procedure, the pooled variance is calculated differently, leading to different t-values. Moreover, the degrees of freedom are computed in a different way and need not be whole numbers.

The following example tests with simulated data whether the means of the normal distributions from which the samples were drawn differ by 0.5 units—once with the usual t-test, and once with Welch's version:

```
x1 <- rnorm(5, mean = 2, sd = 3)
x2 <- rnorm(8, mean = 3, sd = 6)
t.test(x1, x2, mu = 0.5, var.equal = TRUE) # standard t-test


 Two Sample t-test

data:  x1 and x2
t = -0.8264, df = 11, p-value = 0.426
alternative hypothesis: true difference in means is not equal to 0.5
95 percent confidence interval:
 -7.19834  3.99536
sample estimates:
mean of x mean of y
  0.15778   1.75927

t.test(x1, x2, mu = 0.5, var.equal = FALSE) # Welch's t-test


 Welch Two Sample t-test

data:  x1 and x2
t = -0.8969, df = 10.67, p-value = 0.39
alternative hypothesis: true difference in means is not equal to 0.5
95 percent confidence interval:
 -6.77778  3.57481
sample estimates:
mean of x mean of y
  0.15778   1.75927
```

Ruxton (2006) recommends that Welch's t-test should always be used whenever one would otherwise run a two-sample t-test. Personally, in many cases I would now favour a randomisation test, as its null hypothesis is often easier to justify. ♢

Remark 13.7 (Non-normal data)**.** The null hypothesis tested by the t-test is rather unrealistic, since it assumes that the data come from a normal distribution. Normal distributions are idealised mathematical constructs. In reality, genuine data are never drawn from perfectly normal distributions. For example, they may only take values within a bounded range, or they may only take on a countable number of values.

For sufficiently large samples, however, the t-test can be regarded as an approximation to randomisation and permutation tests, as Fisher's quotation at the start of this chapter suggested. There is, however, no guarantee that this approximation will already be accurate enough for a specific, finite sample; see, for instance, Hesterberg (2014, Section 4.4). For simpler designs—such as those in Chapter 12—I would now often prefer randomisation or permutation tests. For more complex designs and models, this option is not always available, which makes it worthwhile to be familiar with these approximation methods nonetheless. ♢

Remark 13.8 (t-tests for other parameter estimates)**.** t-tests can also be used to test null hypotheses for parameter estimates that do not concern group means. We've seen many examples of this in the `summary()` outputs in the previous chapters:

- Page 182: The null hypotheses tested here state that the parameters for `(Intercept)` and `AOA` are actually equal to 0. The first of these is of little substantive interest, since few people are concerned with the intercept, and no one would seriously claim that it ought to be exactly 0. The second is also implausible, as it implies that `AOA` has no linear relationship at all with `GJT`.
- Page 206: Once again, the null hypotheses state that the parameters for `(Intercept)` and `n.Condition` in the population are equal to 0. The t- and p-values for the estimated parameter `n.Condition` could also be obtained using the `t.test()` function, just as we have done above.
- Page 212: Again, the null hypotheses under test are that the parameters in the population are equal to 0.
- Page 221: Likewise. Here, the significance test for `DraganWithCS` might be of genuine interest.

The above tests are exact if we assume i.i.d. normally distributed errors; otherwise, they should be understood as approximations, cf. Remark 13.7.

We can also compute a p-value for a correlation coefficient using a t-test. For instance, with the permutation tests in Section 12.6 on page 270, we found a p-value of 0.005 for the correlation between the Flanker and Simon data in Poarch et al. (2019). Using `cor.test()`, we obtain essentially the same result ($t(32) = 2.95$, $p = 0.0059$):

```
cor.test(poarch$Flanker, poarch$Simon)
```

```
 Pearson's product-moment correlation

data:  poarch$Flanker and poarch$Simon
t = 2.95, df = 32, p-value = 0.0059
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.147206 0.692281
sample estimates:
     cor
0.462355
```

Incidentally, you obtain the same p-value if you analyse these data within a regression model:

```r
# Output not shown
poarch1.lm <- lm(Flanker ~ Simon, data = poarch)
poarch2.lm <- lm(Simon ~ Flanker, data = poarch)
summary(poarch1.lm)
summary(poarch2.lm)
```

In this case, the null hypothesis is that the correlation coefficient, or equivalently the β coefficient, is equal to 0, and that the errors are i.i.d. normally distributed. If the errors are not normally distributed, the test should again be seen as an approximation. $\diamondsuit$

Chapter 14

Analysis of variance

With a two-sample t-test we can compare two groups. If instead we want use significance tests to compare more than two groups with respect to an outcome, it might seem natural to run multiple t-tests (or an alternative such as randomisation tests). For instance, if we wanted to compare three groups, we could first test groups A and B, then A and C, and finally B and C. The problem with this approach is that the probability of finding at least one significant difference when the null hypothesis is in fact true is greater than α. The simulations below demonstrate this problem, followed by some possible solutions.

We begin by generating data for which we know the null hypothesis literally holds. Here we assume random sampling, but even if we assumed random assignment instead, the problem and the solution would remain unchanged. The following command creates a dataset (a tibble) named `d`. It contains 60 observations of two variables: `group` (three levels with 20 observations each) and `score`. The latter variable was generated from a normal distribution whose properties do not depend on `group`. The null hypothesis therefore holds in these simulated data. Figure 14.1 displays the data.

```
d <- tibble(
  group = rep(c("A", "B", "C"), times = 20),
  score = rnorm(n = 3*20, mean = 10, sd = 3)
)
```

We could now split this dataset three times: Once keeping only the data from groups A and B, once keeping only groups A and C, and once keeping only groups B and C. Each time, we would then perform a Student's t-test.

Your output will look different, of course, since the data were generated randomly. As for the R code, note the use of `%in%` and the placeholder `_`. The latter passes the tibble created before the preceding `|>` to a specific parameter in the next function (here the `data` parameter of `t.test()`).

```
# A vs. B
d |>
  filter(group %in% c("A", "B")) |>
  t.test(score ~ group, data = _, var.equal = TRUE)
```

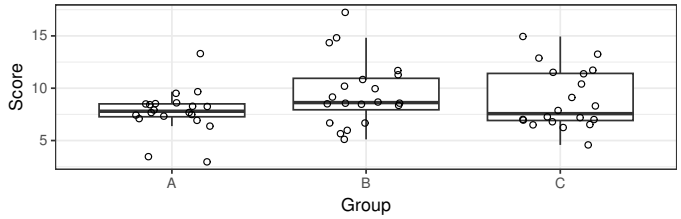


Figure 14.1: Three random samples with 20 observations each are drawn from the same normal distribution. Your graph will look a bit different to this one.

```
 Two Sample t-test

data:  score by group
t = -2.076, df = 38, p-value = 0.0447
alternative hypothesis: true difference in means between group A and group B is not equal
95 percent confidence interval:
 -3.4924301 -0.0436345
sample estimates:
mean in group A mean in group B
        7.77105         9.53908

# A vs. C
d |>
  filter(group %in% c("A", "C")) |>
  t.test(score ~ group, data = _, var.equal = TRUE)


 Two Sample t-test

data:  score by group
t = -1.39, df = 38, p-value = 0.173
alternative hypothesis: true difference in means between group A and group C is not equal
95 percent confidence interval:
 -2.70895  0.50371
sample estimates:
mean in group A mean in group C
        7.77105         8.87367

# B vs. C
d |>
  filter(group %in% c("B", "C")) |>
  t.test(score ~ group, data = _, var.equal = TRUE)
```

```
 Two Sample t-test

data:  score by group
t = 0.6988, df = 38, p-value = 0.489
alternative hypothesis: true difference in means between group B and group C is not equal to 0
95 percent confidence interval:
 -1.26214  2.59297
sample estimates:
mean in group B mean in group C
        9.53908         8.87367
```

Even though the null hypothesis literally holds true in this simulation, we would reject it if we follow the logic of significance testing: for the comparison between groups A and B, we observe a p-value smaller than $\alpha = 0.05$ ($p = 0.045$).

It is, of course, expected that we sometimes reject null hypotheses that are actually true—this is not the problem here. The real issue is that this type I error rate, which is supposed to be at most $\alpha = 0.05$, is actually substantially higher.

We can illustrate this with a simulation. The following commands run the same scenario as above—three groups with 20 observations each from the same normal distribution, and three t-tests—20,000 times. Each time, the p-values of the individual t-tests are stored. At the end, the smallest of the three p-values is calculated.

```r
n_runs <- 20000
pval_ab <- vector(length = n_runs)
pval_ac <- vector(length = n_runs)
pval_bc <- vector(length = n_runs)

for (i in 1:n_runs) {
  sim_df <- tibble(
    group = rep(c("A", "B", "C"), times = 20),
    score = rnorm(n = 3*20, mean = 10, sd = 3)
  )

  pval_ab[i] <- t.test(score ~ group, data = sim_df |>
                         subset(group %in% c("A", "B")))$p.value

  pval_ac[i] <- t.test(score ~ group, data = sim_df |>
                         subset(group %in% c("A", "C")))$p.value

  pval_bc[i] <- t.test(score ~ group, data = sim_df |>
                         subset(group %in% c("B", "C")))$p.value
}

# smallest of these 3 p values
```

```
min_pval <- pmin(pval_ab, pval_ac, pval_bc)
```

If the null hypothesis holds, then for any $\alpha \in (0, 1)$ at most $100\alpha\%$ of p-values should fall below α (Definition 12.5). If we plot histograms of the p-values from the individual t-tests, we see that this is indeed the case: the p-values are—for this type of data and this test—uniformly distributed on $(0, 1)$. Any deviations from perfect uniformity that we observe are due to randomness; increasing the number of simulations will reduce these deviations.

However, if we plot the distribution of the smallest of the three p-values (`min_pval`), we notice that these are *not* uniformly distributed; see Figure 14.2 on the facing page. If we were to base our inferences about whether the three groups differ on the smallest of the three p-values, we would reject the null hypothesis not in at most $\alpha = 5\%$ of cases, but in about 12% of cases when comparing three groups:

```
mean(min_pval <= 0.05)

[1] 0.12055
```

This finding is true in general. When multiple significance tests are used to examine a single null hypothesis, the probability that at least one of them produces a significant p-value increases—even if the null hypothesis is true and each test is correctly performed; we've already seen this in Exercise 12.11 on page 265. The probability of observing at least one significant p-value under the null hypothesis is called the **familywise Type-I error rate**; see Figure 14.3 on the facing page.

In situations like the example above, it is clear that a null hypothesis has been tested multiple times. Moreover, in such cases the problem can usually be addressed relatively easily. In other situations, such **multiple comparisons** are less obvious or harder to account for when interpreting results—for instance, if only a selection of the conducted significance tests is reported in a research paper. See Chapter 17 for further discussion.

Exercise 14.1. Let P_1, P_2, P_3 be independent random variables, each uniformly distributed on $(0, 1)$.

1. Calculate the probability that at least one of these random variables takes a value of at most 0.05.

   Hint: You've already done the hard part in Exercise 12.11.

2. The value you've calculated is greater than the 12% we obtained above. This is not because we estimated the latter number using a simulation. Why, then, do these numbers differ? ♢

14.1 First solution: Randomisation tests

The simplest way to address the multiple comparisons problem is to test the null hypothesis with a single ('omnibus') test rather than multiple separate tests. Most commonly, this is done using analysis of variance (ANOVA) and the F-test (see the next section).

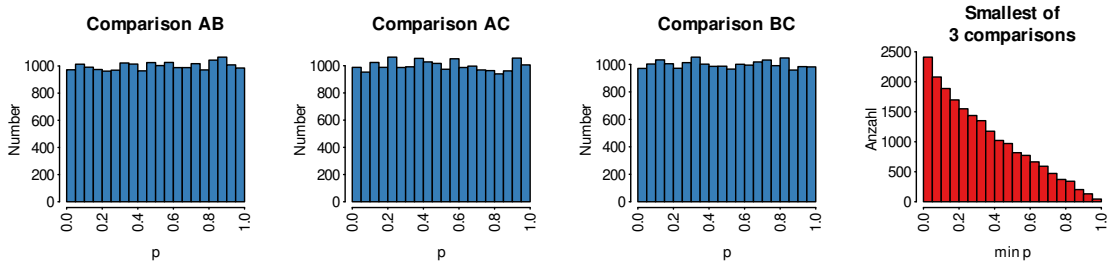


Figure 14.2: If the null hypothesis is true, we should observe $p \leq \alpha$ with probability at most α for all $\alpha \in (0,1)$. For the p-values for the individual t-tests, this is the case (blue histograms): the p-values follow a uniform distribution on $(0,1)$. But the distribution of the smallest of the three p-values is right-skewed (red histogram): the probability that at least one of the three p-values is no larger than α can be considerably larger than α.

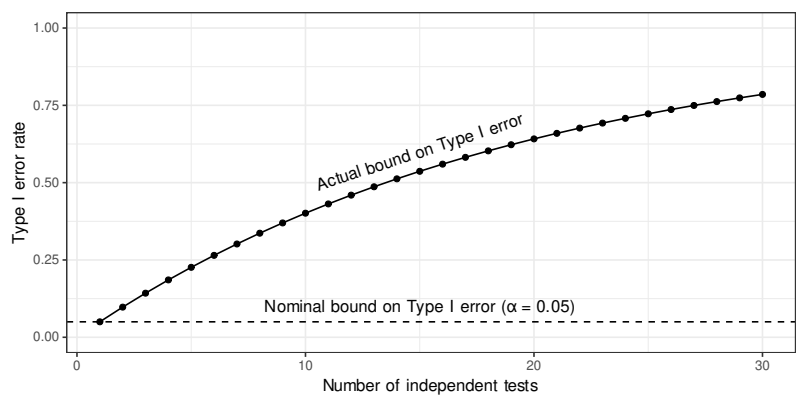


Figure 14.3: If all null hypotheses are true and several independent significance tests are carried out, the probability of finding at least one significant result increases. This probability is the familywise Type-I error rate. Two tests are independent if the outcome of one test doesn't give you any information as to the outcome of the other.

However, the same goal can also be achieved with a randomisation test, which in my opinion is easier to follow. If you have worked through Exercise 12.14, you have already implemented this test. For clarity, we will discuss this approach again here.

The randomisation test assumes the same null hypothesis as in Chapter 12: the participants (or whatever was observed) were assigned to the groups at random, and the differences between the group means that we are calculating here are simply the result of this random assignment.

```
d |>
  group_by(group) |>
  summarise(group_mean = mean(score))
```

```
# A tibble: 3 x 2
  group group_mean
  <chr>      <dbl>
1 A           7.77
2 B           9.54
3 C           8.87
```

The crucial question now is: how likely would it be to observe group means that differ this much—or even more—if the null hypothesis were actually true? To answer this question, we first need a single number that summarises how strongly these three means differ. The variance of the group means is a natural choice for this purpose.

```
d |>
  group_by(group) |>
  summarise(group_mean = mean(score)) |>
  select(group_mean) |>
  var()
```

```
           group_mean
group_mean   0.797414
```

The variance of the sample means is therefore 0.797. Your result will differ slightly because the data were generated randomly. To generate the distribution of the variances of the sample means under the null hypothesis, we can randomly reassign the observations to the group labels a few thousand times and calculate the variance of the sample means for each permuted dataset. The logic is identical to the permutation tests from the previous chapter; the only difference is that here we consider the variance of the group means rather than the difference between two group means.

```
m <- 19999
var_means_H0 <- replicate(m, {
  tapply(d$score, sample(d$group), mean) |> var()
})
```

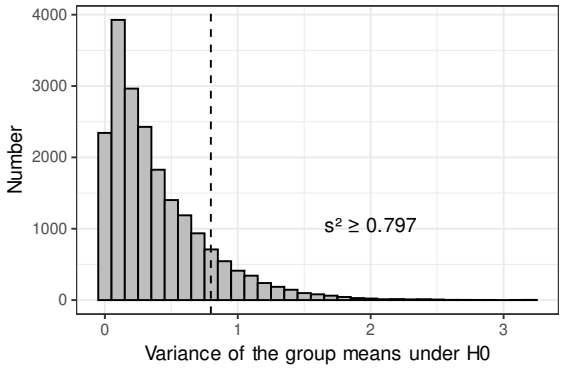


Figure 14.4: The distribution of the variances of the group means under rerandomisation.

The distribution of these variances (`var_means_H0`) is shown in Figure 14.4. We calculate the corresponding p_r-value, that is, a right-tailed p-value, as follows:[1]

```
(sum(var_means_H0 >= 0.7974136) + 1) / (m + 1)

[1] 0.13055
```

In this case, the p-value is therefore 0.131. This value will vary slightly from one run to another since it is based on 'only' 20,000 of the nearly 578 quadrillion possible permutations. However, these differences should be minimal.

14.2 Second solution: Analysis of variance and F-tests

Randomisation tests were historically too cumbersome to perform. As in the previous chapter, there are mathematical tricks that allow us to obtain an approximation with much less brute-force computation. These techniques are called **analysis of variance** (ANOVA) and the **F-test**. Since research articles are often full of ANOVAs and F-tests, we'll take a closer look at these techniques here.

14.2.1 Partitioning the variance

The first step in an ANOVA is to partition the variance in the outcome into two parts: variance *between* groups and variance *within* groups. To do this, we first calculate the **total sum of squares**, that is, the sum of the squared differences between the observations and their overall mean. Your numbers will differ, of course, because the data were randomly generated.

```
(sq.total <- sum((d$score - mean(d$score))^2))

[1] 461.642
```

[1] A left-tailed p-value is rarely computed in this context, since it corresponds to the alternative hypothesis that the group means differ *less* than expected under the null. One application of such a left-tailed p-value is to detect data fabrication (Simonsohn, 2013).

To calculate the variance within the groups, known as the **residual sum of squares**, we can subtract each observation's group mean. Alternatively, we can model the data using a general linear model and compute the sum of squares of the residuals:

```
d.lm <- lm(score ~ group, data = d)
(sq.rest <- sum((resid(d.lm))^2))

[1] 429.745
```

The sum of squares explained by the model is then:

```
(sq.group <- sq.total - sq.rest)

[1] 31.8965
```

In addition to the intercept, two parameters were needed to model the effect of the group factor. You can check this by inspecting the model `d.lm`. On average, each additional parameter accounts for a sum of squares of 15.9:

```
(meanSq.group <- sq.group / 2)

[1] 15.9483
```

There are 60 independent data points, and the model already estimates three parameters (intercept + 2 group parameters). In principle, we could still estimate 57 more parameters. This wouldn't make sense in practice, but it would be possible. We cannot estimate more parameters than there are observations: in a general linear model, the number of estimable parameters is limited by the number of data points. On average, these 57 parameters would account for a sum of squares of 7.54:

```
(meanSq.rest <- sq.rest / (60 - 2 - 1))

[1] 7.53939
```

The key insight is that, if the null hypothesis holds, the two group parameters should, on average, explain no more variance than the remaining 57 unmodelled parameters. In other words: if H_0 is true, `meanSq.group` and `meanSq.rest` should be roughly equal. Another way to express this is that the ratio of `meanSq.group` to `meanSq.rest` should be 1 under the null hypothesis. This ratio is called F. In our case, F equals 2.12:

```
(f.value <- meanSq.group / meanSq.rest)

[1] 2.11533
```

14.2.2 The F-test

Because of sampling error, F will never be exactly 1. The next question, therefore, is: how unusual would an F value of 2.12 or larger be if the null hypothesis were true?

If the errors are i.i.d. normally distributed, then this F value follows a specific distribution, namely an F **distribution**.

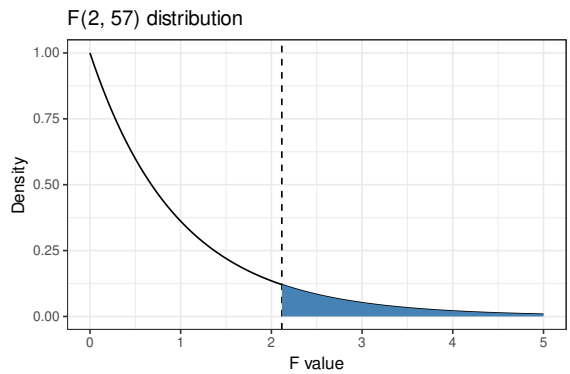


Figure 14.5: The F distribution with 2 and 57 degrees of freedom. The dashed line shows the F value we obtained. The p value corresponding to this F value is the aread of the patch that is coloured in.

Definition 14.2 (F distribution)**.** Let $X_1, \ldots, X_k, X_{k+1}, \ldots, X_{k+\ell} \sim \mathcal{N}(0,1)$ be independent random variables. Then the distribution of

$$\frac{\frac{1}{k}\sum_{i=1}^{k} X_i^2}{\frac{1}{\ell}\sum_{i=k+1}^{k+\ell} X_i^2}$$

is called the F distribution with k and ℓ degrees of freedom; it is denoted $F_{k,\ell}$ or $F(k,\ell)$. $\diamondsuit$

F distributions have two parameters ('degrees of freedom'): the number of parameters used to model the effect (here: 2), and the number of observations not yet used up (here: 57). Figure 14.5 shows an $F(2,57)$ distribution.

The area under the curve to the right of the dashed line represents the probability of observing an F value of 2.12 or higher if the null hypothesis is true—assuming the errors are normally distributed. It can be calculated as follows:

```
pf(f.value, df = 2, df2 = 60 - 2 - 1, lower.tail = FALSE)

[1] 0.129963
```

This is almost the same result as obtained from the randomisation test.

14.2.3 Analysis of variance in R

Fortunately, you don't need to carry out all these steps by hand. You can just model the data in a general linear model (as we did above) and then apply the `anova()` function to the model:

```
anova(d.lm)

Analysis of Variance Table
```

```
Response: score
          Df Sum Sq Mean Sq F value Pr(>F)
group      2   31.9  15.948   2.115   0.13
Residuals 57  429.7   7.539
```

You would report this result as follows: $F(2, 57) = 2.12$, $p = 0.13$.

Exercise 14.3 (t- vs F-test)**.** Revisit the money priming data analysed in Chapter 9. Test the group difference using a standard two-sample t-test and also using an ANOVA. Square the t value and compare it to the F value. ♢

14.3 Analysis of variance with multiple predictors

Analysis of variance can also be applied when there are multiple predictors in a model. The data from the Dragan/Luca study by Berthele (2012) (see Chapter 10) can be analysed in an ANOVA as follows:

```
berthele <- read_csv(here("data", "berthele2012.csv"))
berthele.lm1 <- lm(Potential ~ CS * Name, data = berthele)
anova(berthele.lm1)
```

```
Analysis of Variance Table

Response: Potential
           Df Sum Sq Mean Sq F value   Pr(>F)
CS          1   1.76   1.755   2.505 0.115571
Name        1   0.36   0.364   0.520 0.471895
CS:Name     1   7.97   7.965  11.369 0.000948
Residuals 151 105.79   0.701
```

Each row in the table shows how many parameters were needed to model a particular predictor (`Df`) and the associated sum of squares, as well as its F- and p-values. However, be careful: if we change the order of the predictors, a few numbers will change:

```
berthele.lm2 <- lm(Potential ~ Name * CS, data = berthele)
anova(berthele.lm2)
```

```
Analysis of Variance Table

Response: Potential
           Df Sum Sq Mean Sq F value   Pr(>F)
Name        1   0.79   0.786   1.121 0.291338
CS          1   1.33   1.334   1.904 0.169673
Name:CS     1   7.97   7.965  11.369 0.000948
Residuals 151 105.79   0.701
```

This happens because `Sum Sq` is calculated sequentially. If we were to perform the ANOVA manually, we would:

1. calculate the total sum of squares;

```
(ss.total <- sum((berthele$Potential - mean(berthele$Potential))^2))

[1] 115.871
```

2. remove the effect of the first variable and calculate the sum of squares it accounts for;

```
cs.lm <- lm(Potential ~ CS, data = berthele)
(ss.cs <- ss.total - sum(resid(cs.lm)^2))

[1] 1.755
```

3. remove the effect of the second variable and calculate the sum of squares it accounts for;

```
name.lm <- lm(Potential ~ CS + Name, data = berthele)
(ss.name <- ss.total - ss.cs - sum(resid(name.lm)^2))

[1] 0.364399
```

4. remove the interaction effect and calculate its sum of squares;

```
interaction.lm <- lm(Potential ~ CS + Name + CS:Name, data = berthele)
(ss.interaction <- ss.total - ss.cs - ss.name -
sum(resid(interaction.lm)^2))

[1] 7.96514
```

5. calculate the remaining sum of squares;

```
(ss.rest <- ss.total - ss.cs - ss.name - ss.interaction)

[1] 105.786
```

6. compute the F-values for all effects.

```
ss.cs / (ss.rest/151)

[1] 2.5051

ss.name / (ss.rest/151)

[1] 0.520144

ss.interaction / (ss.rest/151)

[1] 11.3695
```

Depending on whether we enter `CS` or `Name` as the first variable in the model, the sum of squares attributed to this variable changes. This is because `CS` and `Name` are somewhat correlated in this dataset: there are more data points for Dragan without code-switches than with, and more data points for Luca with code-switches than without:

```
xtabs(~ CS + Name, data = berthele)

         Name
CS        Dragan Luca
  with        28   44
  without     51   32
```

Part of the variation in the `Potential` values therefore cannot be unambiguously assigned to one predictor or the other. If `CS` is added first to the model, all of this ambiguous variation is attributed to `CS`; if `Name` is added first, it is attributed to `Name`. As a result, the sums of squares for these variables change depending on which variable is entered first. However, the result for the last effect (the interaction) remains unchanged.

Remark 14.4 (Type-I, Type-II, Type-III sum of squares)**.** When the sums of squares and the corresponding F- and p-values are computed sequentially (as above), this is called the **Type-I sum of squares**. This method is particularly useful if you are interested in the effect that is entered last in the model but want to account for other variables first.

An alternative approach is somewhat creatively called **Type-II sum of squares**. Here, effects are entered into the model in all possible orders (with interactions always added after main effects), and the F- or p-value for an effect is taken as the value obtained when the effect is entered last. In our example, the results would look as follows:

- `CS`: We take the values for `CS` from the ANOVA table of model `berthele.lm2`, because here `CS` is entered as the last main effect: $F(1, 151) = 1.9$, $p = 0.17$.
- `Name`: We take the values for `Name` from the ANOVA table of model `berthele.lm1`, because here `Name` is entered as the last main effect: $F(1, 151) = 0.52$, $p = 0.47$.
- The interaction is always entered after the main effects, so its result is identical in both models: $F(1, 151) = 11.4$, $p < 0.001$.

These results can be calculated automatically using the `Anova()` function (with a capital A) from the `car` package:

```
car::Anova(berthele.lm1)

Anova Table (Type II tests)

Response: Potential
          Sum Sq  Df F value   Pr(>F)
CS          1.33   1   1.904 0.169673
Name        0.36   1   0.520 0.471895
CS:Name     7.97   1  11.369 0.000948
Residuals 105.79 151
```

There is also a **Type-III sum of squares** calculation, but this is generally discouraged. Both Type-I and Type-II sums of squares can be useful depending on the situation, and if you are only interested in the interaction, or if each 'cell' has the same number of

observations, the choice doesn't matter. However, please do not discard data just to achieve equal cell sizes in order to avoid choosing between these methods! ♢

Exercise 14.5 (Randomisation test)**.** Write your own randomisation test and test the null hypothesis that the factors `CS` and `Name` only appear to interact due to random assignment. ♢

Remark 14.6 (Terminology)**.** In the research literature, several abbreviations have become standard to distinguish between different types of analysis of variance.

- One-way ANOVA is ANOVA with a single predictor. Most often, this is a grouping variable with two or more levels.
- Two-way ANOVA is ANOVA with two predictors. Frequently, the interaction between the predictors is of interest. If one variable has two levels and the other four, this is often called a 2×4 *two-way* ANOVA. *Three-way*, *four-way*, etc., ANOVA also exist: you simply add more predictors to the model.
- ANCOVA stands for **analysis of covariance**. This is simply an ANOVA in which at least one continuous control variable is included.
- RM-ANOVA stands for **repeated-measures analysis of variance**. This approach can be used when, for example, multiple measurements of the same variable are collected per participant. RM-ANOVA is not discussed here, as more flexible tools now exist for handling dependency structures in the data (mixed models).
- MANOVA stands for **multivariate analysis of variance**. It is an extension of ANOVA to multiple outcome variables. Everitt & Hothorn (2011) write the following about MANOVA in the preface to their book *An introduction to applied multivariate analysis with R*:

  > "But there is an area of multivariate statistics that we have omitted from this book, and that is multivariate analysis of variance (MANOVA) and related techniques (...). There are a variety of reasons for this omission. First, we are not convinced that MANOVA is now of much more than historical interest; researchers may occasionally pay lip service to using the technique, but in most cases it really is no more than this. They quickly move on to looking at the results for individual variables. And MANOVA for repeated measures has been largely superseded by the models that we shall describe [in their book]." (p. vii-viii) ♢

Remark 13.7 also applies to F-tests and analysis of variance.

14.4 Planned comparisons and post-hoc tests

With a one-way analysis of variance, the aim is to answer the question of whether the group means (*any* of the group means) in the population differ from one another, or whether any observed differences are merely due to random assignment. If a small p-value is found, one tends to conclude that some of the group means do indeed differ.

However, ANOVA itself does not answer the natural follow-up question: which groups actually differ from each other? Is it groups A and B, or rather groups A and C, or B and C?

To answer such questions, researchers often perform significance tests following ANOVA. If these questions arise *after* the data have been collected and were not derived from theory beforehand (exploratory analysis), they are referred to as **post-hoc tests**. If the questions existed *prior* to the study (confirmatory analysis), they are called **planned comparisons**.

Common procedures for these follow-up tests bear names such as '*t*-tests with Bonferroni correction', '*t*-tests with Holm–Bonferroni correction', 'Fisher's least significant difference test', 'Scheffé test', and so on. The idea behind all of them is to control the increased global risk of committing a Type I error due to multiple testing.

However, additional tests are not always necessary or desirable. In my view, you need to consider the theory and hypotheses that underpin the study, and which data patterns would count as evidence for these:

- If the theory predicts that there will be *some* group differences (any differences at all), then a single ANOVA is sufficient. Any interesting differences can be reported descriptively—for example, by graphically illustrating the linear model used for the ANOVA. Such potential differences can then be tested in a new confirmatory study (see Bender & Lange, 2001, p. 344). If the ANOVA is not significant, one should refrain from additional tests so as not to inflate the familywise Type I error rate.
- If the theory predicts a *specific* group difference, or multiple separate theories are being tested that refer to different group means (e.g., A vs B, and C vs D), then ANOVA is not strictly necessary; pairwise comparisons suffice. Any interesting but unpredicted differences should be reported descriptively and left to a new confirmatory study.
- If the theory predicts that a particular difference *or* another difference will occur, one should consult the methods mentioned above. The ANOVA itself can be skipped, and one can directly run the relevant tests with an appropriate correction.
- If the theory predicts more complex group differences, for example, 'The combined mean of groups A and B is lower than the combined mean of groups B, C, and D', one should read up on methods designed to test such questions. See Schad et al. (2020) for guidance.
- If the theory predicts that a particular difference *and* another difference will occur, then in my view two pairwise comparisons suffice.

A brief introduction with many references is Bender & Lange (2001). Practical advice can be found in Ruxton & Beauchamp (2008), although it may be difficult to follow without prior experience with such analyses. A relevant blog post on the topic is *On correcting for multiple comparisons: Five scenarios* (1 April 2016).

Tip 14.7 (Be cautious with post-hoc explanations)**.**

It is often tempting, after the fact, to interpret certain patterns in the data theoretically. These patterns may, however, be entirely due to chance and may not replicate in a new study.

It is surprisingly easy to convince oneself, *after the fact*, that one had *predicted* a particular pattern. A major reason is that scientific theories and hypotheses are often vague and can correspond to multiple statistical hypotheses. To avoid deceiving oneself—and later the readers—into thinking that one predicted the patterns exactly as they occurred, one can preregister scientific and statistical hypotheses: specifying in advance the expected patterns and how the data will be analysed. For more information on preregistration, see `http://datacolada.org/64` , Wagenmakers et al. (2012b), and Chambers (2017). ♢

14.5 *F-test in the `summary()` output

We can now finally decipher the last line in the `summary()` output of an `lm()` model. Let us revisit the model `berthele.lm1`:

```
summary(berthele.lm1)


Call:
lm(formula = Potential ~ CS * Name, data = berthele)

Residuals:
    Min      1Q  Median      3Q     Max 
-2.0938 -0.6275  0.0357  0.3725  2.6364 

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)    
(Intercept)           2.964      0.158   18.74  < 2e-16 
CSwithout             0.663      0.197    3.37  0.00096 
NameLuca              0.399      0.202    1.97  0.05025 
CSwithout:NameLuca   -0.933      0.277   -3.37  0.00095 

Residual standard error: 0.837 on 151 degrees of freedom
Multiple R-squared:  0.087, Adjusted R-squared:  0.0689 
F-statistic:  4.8 on 3 and 151 DF,  p-value: 0.0032
```

The F-test ($F(3, 151) = 4.8$, $p = 0.003$) tests the null hypothesis that all predictors together actually explain no variance in the outcome at all. We can reconstruct the numbers using the `anova()` output:

```
anova(berthele.lm1)

Analysis of Variance Table

Response: Potential
           Df Sum Sq Mean Sq F value   Pr(>F)
```

```
CS          1   1.76   1.755   2.505 0.115571
Name        1   0.36   0.364   0.520 0.471895
CS:Name     1   7.97   7.965  11.369 0.000948
Residuals 151 105.79   0.701

meanSq.total <- (1.755 + 0.364 + 7.965) / 3
meanSq.rest <- 105.786 / 151
(f.value <- meanSq.total / meanSq.rest)

[1] 4.798
```

The p-value can then be calculated using the $F(3, 151)$ distribution:

```
pf(f.value, 3, 151, lower.tail = FALSE)

[1] 0.00319983
```

I've never encountered any situation in which this F-test is useful.

14.6 Interim summary

- The purpose of significance tests is to control the probability that we conclude the null hypothesis is false when in fact it is true.
- The null hypothesis is almost invariably that the observed pattern arose purely by chance, either due to random assignment in an experiment or because of random sampling from a population. When one is interested in group differences, this usually corresponds to the assumption that no such differences actually exist.
- p-values typically estimate the probability of observing the given pattern (e.g., a group difference or another parameter estimate) or even more extreme patterns if the null hypothesis were actually true. If this probability is low, i.e., below a pre-specified α threshold (commonly $\alpha = 0.05$), the observed pattern is considered incompatible with the null hypothesis, which is then rejected.
- Significance tests do not assess whether a difference or parameter is large; they merely test the null hypothesis that the parameter has a specific value (usually 0). Consequently, significance tests cannot be used to determine whether an observed difference should be regarded as large or small.
- $p > 0.05$ does *not* imply that the null hypothesis is true; $p \leq 0.05$ does *not* imply that it is false.
- Even if the null hypothesis is false, this does not necessarily mean that your scientific hypothesis is correct: there may be alternative theoretical explanations compatible with the data, or your data collection may be biased in some way.
- Randomisation and permutation tests, t-tests, and F-tests all serve the same purpose. t-tests are used to test the null hypothesis for parameter estimates (e.g., group differences or regression coefficients); F-tests test whether a predictor (often with

more than two levels) explains more variation in the data than would be expected by chance.

- t- and F-tests can be derived from the general linear model, assuming i.i.d. normal errors. They often serve as approximations to randomisation or permutation tests.
- There is not always a single 'correct' significance test for a given situation. When multiple reasonable tests exist, their results do not necessarily have to agree.
- There are significance tests we have not yet encountered. They differ in their computation from t- and F-tests, but their goal remains the same: to calculate the probability of observing the actual or an even more extreme pattern if it were due solely to chance.

Chapter 15

*Calculating statistical power

Even when there really is a difference between two groups or an association between certain variables at the population level, it is possible that a sample will fail to demonstrate this difference or association with a significance test. Similarly, in experiments with random allocation, a genuinely existing intervention effect may go undetected. This fact is illustrated in Figure 15.1 on the next page. The probability that a significance test detects a true effect is called its power. When planning studies, it can be useful to take this power into account. This chapter therefore explains how the power of a test can be calculated.

In the interest of honesty, I should add that I myself rarely conduct power analyses. The first reason is that, as you will see in this chapter, a meaningful power calculation requires already having quite a lot of information about the problem at hand. The second reason is that power analyses are mainly useful if one's conclusions are based primarily on p-values, which I increasingly try to avoid.

15.1 Calculating power through simulations

To calculate the power of the scenario illustrated in Figure 15.1, we can write a simulation. The simulation code below does the following:

1. We define the standard deviation of the two populations. In this simulation, the standard deviation is $\sigma = 2$ for both populations. Both distributions are assumed to be normally distributed, since this is an assumption of the t-test used in the analysis. (Here we assume random samples from populations; we could also design the simulation to mimic an experiment with random assignment, but in practice the differences are minimal.)
2. We define the difference between the means of the two normal distributions. Here, $\mu_B - \mu_A = 0.7$.
3. We set the number of observations per sample. Here, $n_A = n_B = 32$.
4. We draw a sample from each population 20,000 times.
5. These samples are modelled with a general linear model, and the p-value of the difference (based on a t-distribution) is stored. For this last step one could also

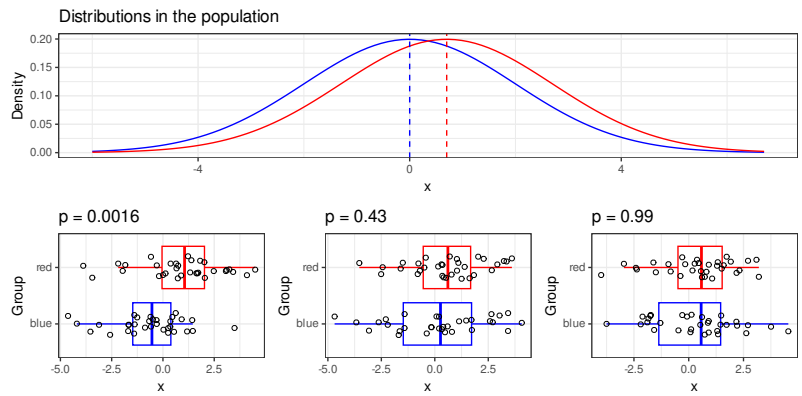


Figure 15.1: Top: Two normal distribution with $\sigma = 2$. The mean of the blue distribution is 0, that of the red is 0.7. Bottom: We draw three times two random samples with 32 observations each from these normal distributions. Even though there is a difference between the means of the distributions, there needn't be a significant difference between the sample means.

use the `t.test()` function, but the current code is easier to adapt to more complex designs.

6. The distribution of the p-values is visualised (Figure 15.2).

7. The proportion of significant p-values ($p \leq 0.05$) is calculated.

```
# Population parameters (play around with these!)
sd_group <- 2
difference <- 0.7

# Sample sizes (play around with these!)
n_groupA <- 32
n_groupB <- 32
group <- rep(c("A", "B"), times = c(n_groupA, n_groupB))

# Simulation
n_runs <- 20000
p_values <- replicate(n_runs, {
  score = c(rnorm(n = n_groupA, mean = 0, sd = sd_group),
            rnorm(n = n_groupB, mean = difference, sd = sd_group))
  mod.lm <- lm(score ~ group)
  summary(mod.lm)$coefficients[2, 4]
})

# Proportion of significant tests
mean(p_values <= 0.05)
```

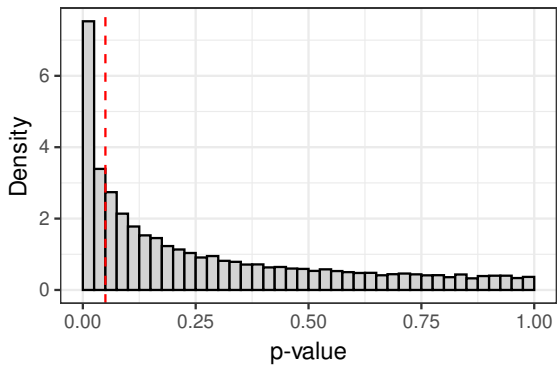


Figure 15.2: The distribution of the 20,000 p values in the power simulation.

```
[1] 0.273
```

Conclusion: If the populations are normally distributed with $\sigma = 2$ and there is a difference of 0.7 between their means, then there is (only) about a 27% chance of finding a significant difference when drawing random samples of 32 observations each. The significance test used here is a two-sided t-test assuming equal variances.

Exercise 15.1.

1. How does the power in the above example change if the normal distributions have standard deviations of $\sigma = 3$ instead of $\sigma = 2$?
2. How does the power change in this example if, instead of 32 observations in each sample, there are 54 observations in one sample and 10 in the other?
3. How much power would one have if the difference between the two populations were infinitesimal? Try to answer this question first without using a simulation. Then check your answer with a simulation (for instance with $\mu_B - \mu_A = 0.000001$). ♢

15.2 Analytical power calculation

For a few significance tests, including t-tests, the power can also be calculated analytically. The formulas for this are not shown here, but they are implemented in the `power.t.test()` function. For a difference of 0.7 units between normal distributions with $\sigma = 2$ and samples of 32 observations each, the power can be calculated as follows:

```
power.t.test(n = 32, delta = 0.7, sd = 2)

     Two-sample t test power calculation
```

```
              n = 32
          delta = 0.7
             sd = 2
      sig.level = 0.05
          power = 0.280414
    alternative = two.sided

NOTE: n is number in *each* group
```

The advantage of this function is that one can specify the desired power, leave the number of observations blank, and thereby learn how many observations per sample would be needed to achieve the desired power:

```
power.t.test(delta = 0.7, sd = 2, power = 0.90)


     Two-sample t test power calculation

              n = 172.516
          delta = 0.7
             sd = 2
      sig.level = 0.05
          power = 0.9
    alternative = two.sided

NOTE: n is number in *each* group
```

So, one would need 173 observations per sample to have a 90% chance of finding a significant difference when the populations differ by 0.7 units and have a standard deviation of 2.

Other parameters can also be left unspecified:

```
power.t.test(n = 40, sd = 2, power = 0.75)


     Two-sample t test power calculation

              n = 40
          delta = 1.19291
             sd = 2
      sig.level = 0.05
          power = 0.75
    alternative = two.sided

NOTE: n is number in *each* group
```

If you want to have 75% power, have 40 observations per group, and assume that the populations have a standard deviation of 2, then you must hope that the difference between the population means is at least 1.2 units.

To conduct a power calculation, you need to know three of the following four pieces of information:

- The number of observations.
- The effect size in the population (here: the difference between population means). This may be the expected or hoped-for effect size, but it may also be the smallest effect size one considers practically relevant.
- The variability in the population (here: the standard deviation of the distributions).[1]
- The desired power.

Figure 15.3 on the following page illustrates how these four factors interact. In my view, it is not particularly meaningful to speak of *the* power of a test: it is more sensible to say that a test, under certain assumptions (concerning the actual difference, the variability within groups, and the distribution of the residuals), would have a certain power, and under other assumptions a different one. Figure 15.3 makes this point quite clearly.

```
# Combinations of sample sizes, differences and standard deviations
# we want to compute the power for:
power.df <- expand.grid(
  sample_size = seq(from = 2, to = 100, by = 1),
  delta = seq(0.3, 1.5, by = 0.3),
  SD = c(0.5, 1, 1.5, 2)
)

# Compute power
power.df$Power <- power.t.test(n     = power.df$sample_size,
                               delta = power.df$delta,
                               sd    = power.df$SD)$power

# Add text
power.df$SD <- paste("Standard deviation of error:\n", power.df$SD)

# Plot
ggplot(power.df,
       aes(x = sample_size,
           y = Power,
           colour = factor(delta))) +
  geom_line() +
  ylim(0, 1) +
```

[1] In principle, only the ratio between the effect size and the variability is relevant; see Cohen (1977, 1992).

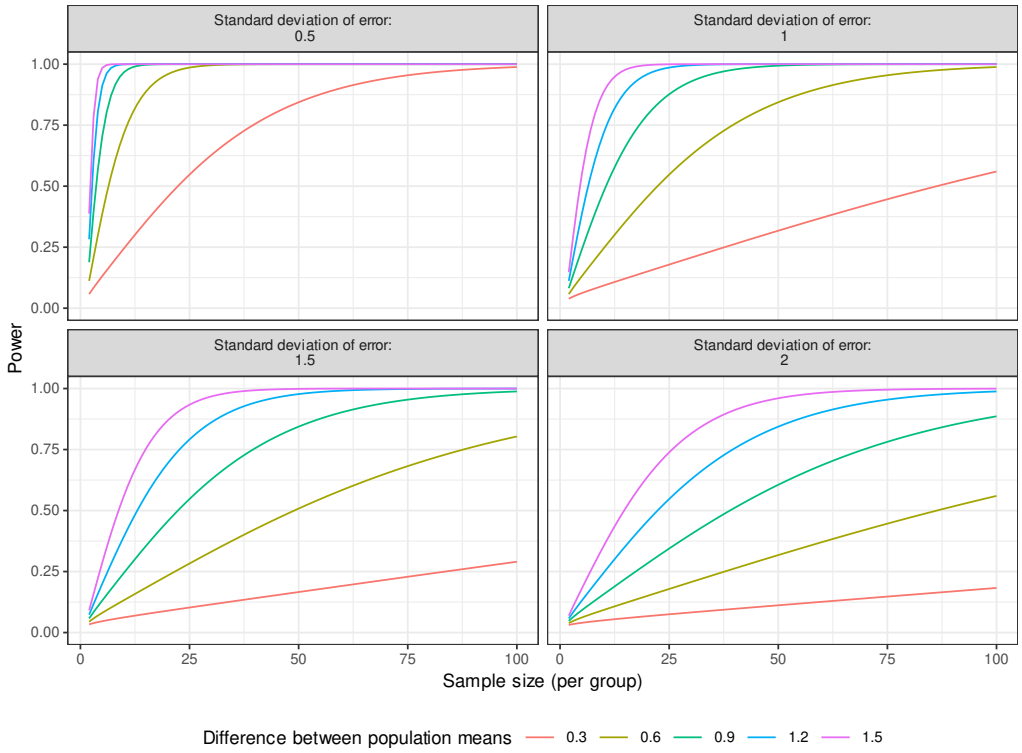


Figure 15.3: The power of a significance test depends on the effect size (here: the difference between the population means), the error variance (shown here as the standard deviation of the error distribution), and the amount of data.

```
facet_wrap(vars(SD)) +
xlab("Sample size (per group)") +
scale_colour_discrete(name = "Difference between population means") +
theme(legend.position = "bottom")
```

15.3 Further reading

Cohen (1992) is a concise introduction into power computations. Another useful article is Kelley et al. (2003). Two blog posts of mine on the subject of power are *Abandoning standardised effect sizes and opening up other roads to power* (14 July 2017) and *Increasing power and precision using covariates* (24 October 2017). A more advanced article, which is particularly relevant for experimental research in the language sciences, is Westfall et al. (2014).

Chapter 16

*Silly tests

When and whether significance tests are meaningful is a matter of debate. Less disputed, however, is that significance tests are often used in contexts where they are redundant. Abelson (1995) called these silly tests. Superfluous tests make research reports less accessible, since readers have to expend extra effort to separate the relevant information from the irrelevant. The following sections therefore discuss five types of superfluous significance tests, with the aim of helping you avoid them in your own work. See also Vanhove (2021b).

16.1 RM-ANOVA for pretest/posttest designs

A useful research design is the pretest–posttest design with both a control group and an experimental group. All participants first complete a task (pretest), are then randomly assigned to the experimental or control group, and after the intervention complete the same or a similar task again (posttest). Such data are often analysed using a repeated-measures analysis of variance (RM-ANOVA). The results might then be reported as follows:

> "A repeated-measures ANOVA yielded a non-significant main effect of Condition ($F(1,48) < 1$) but a significant main effect of Time ($F(1,48) = 154.6$, $p < 0.001$). In addition, the Condition $\times$ Time interaction was significant ($F(1,48) = 6.2$, $p = 0.016$)." [Based on a fictional example from Vanhove, 2015.]

The problem with this analysis is that three tests are reported ('main effect of Condition', 'main effect of Time', and 'Condition $\times$ Time interaction'), but only the interaction is of real interest: what matters is whether the experimental group improved more than the control group:

- That posttest performance across both groups is better than pretest performance ('main effect of Time') is uninformative: perhaps this reflects a practice effect, perhaps the posttest was easier than the pretest, or perhaps both groups learnt something simply from taking part in the experiment.
- That overall performance differs (or not) between the two groups across both measurement points ('main effect of Condition') is also irrelevant: the average differ-

ence may be negligible, yet participants in the experimental group may still have learnt more (for instance, if their pretest scores were lower than the control group's but their posttest scores higher).

- Only the interaction effect is informative here: is the effect of 'Time' different in size depending on 'Condition'?

This analysis can be simplified by analysing the *differences* between posttest and pretest performance for each participant using a t-test. The result would be unchanged, but the analysis is easier to follow and only a single (relevant) test is reported. Instead of a t-test, one could of course also use a Wilcoxon test or a randomisation test.

An even better alternative is to construct a linear model in which the posttest results are treated as the outcome, with both group membership and pretest performance included as predictors (= ANCOVA). This yields greater power and precision, and has the added advantage that pretest and posttest scores do not even need to be expressed on the same scale. See Vanhove (2015) for further details.

16.2 Balance tests

Even when participants have been randomly assigned to different groups, researchers often describe how these constructed groups differ with respect to variables such as age, language proficiency, or IQ. Significance tests are often used for this purpose. In Vanhove (2015), I discuss three reasons why such significance tests are superfluous; the two most important are these.

- These significance tests examine the null hypothesis that (for example) the mean age differs between the groups purely due to chance. But since participants were randomly assigned, we *know* that this null hypothesis is actually true. The test either tells us that the group difference is due to chance—something we already knew—or it claims there is a systematic difference, which would be incorrect (Type I error). The test therefore cannot provide any information that we do not already know and that is simultaneously true.

- Using such balance tests implies that researchers may think the goal of randomisation is to produce perfectly balanced groups. As we have already seen in Chapter 12, this is not the goal. The goal of randomisation is to prevent *systematic* bias. Random chance differences between groups are already accounted for in p-values and confidence intervals, so it is unnecessary for the groups to be perfectly balanced on background variables.

Important background variables can, of course, be incorporated in the analysis, but this is better achieved by including them as predictors in the model. If necessary, randomisation can also be restricted to make groups more comparable with respect to such variables (see Imai et al., 2008, Vanhove, 2015, and the `walkthrough_blocking()` function in the `cannonball` package).

16.3 Tautological tests

> "The 70 participants were divided into three proficiency groups according to their performance on a 20-point French-language test. The high-proficiency group consisted of participants with a score of 16 or higher ($n = 20$); the mid-proficiency group of participants with a score between 10 and 15 ($n = 37$); and the low-proficiency group of participants with a score of 9 or lower ($n = 13$). An ANOVA showed significant differences in the test scores between the high-, mid- and low-proficiency groups ($F(2, 67) = 133.5$, $p < 0.001$)." [Fictional example from the blog post *Silly significance tests: Tautological tests* (15 October 2014)]

The problem with this significance test is that it is entirely tautological. Similar to balance tests, such tautological tests cannot tell us anything that we did not already know and that is simultaneously true: we deliberately constructed the groups so that they do not overlap on a particular variable. It is therefore unsurprising that the null hypothesis ('any difference is due purely to chance') is not true here.

Tautological tests are symptomatic of a broader problem: the belief that predictors must always be categorical or that one should always create groups. By splitting continuous data into categories, information is lost, and with it statistical precision. See also *The problem with cutting up continuous variables and what to do when things aren't linear* (16 October 2015).

16.4 Tests unrelated to the research question

Quite frequently, significance tests are conducted that have little or nothing to do with the actual research question. A fictional example of this is when an intervention study is carried out and one compares not only the performance of the control and intervention groups, but also the performance of boys versus girls, or of participants from different socioeconomic backgrounds, and so on. Factors such as gender and social class may indeed influence performance, but this does not justify performing separate significance tests on them. If you believe that such factors are important, it is generally more sensible to include them directly in the model rather than running separate analyses for each.

For further guidance, see the blog post *Silly significance tests: Tests unrelated to the genuine research questions* (8 June 2015).

16.5 Tests for uninteresting main effects

Often, researchers are primarily interested in the interaction between two or more predictors. When testing this interaction in an analysis of variance, however, significance tests for the main effects are also reported. These main-effect tests are usually not of substantive interest. In my view, they may in fact make research reports harder to digest. If, for some reason, it is absolutely necessary to report significance tests for uninteresting main effects, they should, in my opinion, be placed in a table whose caption clearly indicates that only the interaction is the effect of genuine interest.

16.6 Conclusion

Just because you could perform a significance test, or because the software outputs one, does not mean you should report it. Always ask yourself how relevant each significance test would be for the questions you aim to answer. Ideally, in my view, there should be at most one significance test per research question. More generally, I believe that one should not report analyses whose relevance is not clear to oneself—even if there is a concern that reviewers might otherwise complain. See *In research, don't do things you don't see the point of* (18 February 2022).

Please do not misunderstand the previous paragraph: I am *not* suggesting that only significance tests yielding a significant p-value should be reported. And nor do I think that every analysis should be accompanied by a p-value.

Exercise 16.1. Read the introduction of a quantitative research article that is relevant to your current research project or one that you have studied in a seminar. Write down the research question(s) you consider most important in terms of what the study aims to answer. Then, count how many significance tests are reported or mentioned in the results section. (Sometimes it is only mentioned that a significance test was performed, without detailed reporting.) How many of these are directly relevant to answering the key research question(s) you identified?

Note: See Figure 4 in Vanhove (2021b)! ♢

Chapter 17

*Questionable research practices

When dealing with significance tests—whether because you use them yourself or because you frequently encounter them in the research literature—it is important to be aware of several common misapplications that can undermine the purported meaning of *p*-values. The overarching problem behind these questionable research practices is that researchers and journal editors tend to prefer certain results (read: significant results), and the research and publication process is therefore geared towards producing and publishing such findings. If one is sufficiently flexible during data collection, analysis, and reporting, it becomes very easy to find and report significant patterns without those 'findings' necessarily reflecting reality.

To inform yourself about questionable research practices related to significance testing, several key articles are highlighted here. Comprehensive overviews in book form include Chambers (2017, *The Seven Deadly Sins of Psychology*) and Ritchie (2021, *Science Fictions*).

17.1 Sterling et al. (1995) on publication bias

Sterling et al. (1995, 4 pages of text) counted how often significance tests reported significant *p*-values in medical and psychological journals. In the medical journals, the null hypothesis was rejected in 85% of cases, and in the psychology journals, this figure was a striking 96%.

As Sterling et al. explain, these numbers indicate that researchers and editors discriminate against non-significant findings. Even if the null hypothesis were never true, it would be impossible for so many tests to yield significant results. Their hypothesis is that researchers tend to write up studies that produce significant (rather than non-significant) results, and editors tend to reject articles reporting non-significant findings.

This **publication bias** can lead to the literature overestimating both the evidence for and the magnitude of a particular effect (for example, 'bilingualism leads to cognitive advantages'; see also de Bruin et al., 2015).

17.2 Kerr (1998) on *hypothesizing after the results are known*

Kerr (1998, 20 pages) discusses the reasons for and drawbacks of a common practice in scientific writing: researchers search their data for interesting patterns, but instead of presenting their findings as the outcome of exploratory analysis, the report is written as if the observed pattern had been predicted in advance. Kerr labels this practice **HARKing**—hypothesizing after the results are known. He identifies several costs of HARKing (p. 211), including:

- "Translating Type I errors into hard-to-eradicate theory."
- "Disguising post hoc explanations as a priori explanations (when the former tend also to be more ad hoc, and consequently, less useful)."
- "Not communicating valuable information about what did not work."
- "Encouraging retention of too-broad, disconfirmable old theory."
- "Inhibiting identification of plausible alternative hypotheses."
- "Implicitly violating basic ethical principles."

HARKing is closely linked to cherry picking. In this practice, researchers examine (explicitly or implicitly) different patterns—which could be group differences, correlations, or other effects—and report only the strongest or otherwise most striking ones. The problem is illustrated in the xkcd cartoon in Figure 17.1.

These and related issues are also discussed in an accessible manner by de Groot (2014, originally published in Dutch in 1956; 3.5 pages text + 3 pages commentary). See also Berthele (2019) for similar problems in applied linguistics.

17.3 Simmons et al. (2011) on undisclosed flexibility

Simmons et al. (2011, 7 pages) demonstrate, using both a real example and simulations, that certain common practices can lead to a very high probability of rejecting the null hypothesis even when it is true. The practices they examine include:

- analysing multiple outcomes but reporting only the one that yields the 'best' results;
- examining the data after the first wave of collection to check for significance, and collecting more data if no significant result is found;[1]
- including control variables in the analysis only if the result without them is not significant;[2]

[1] If you intend to do this yourself, see Lakens (2014) for guidance on how to analyse the data. Crucially, the relevant decisions must be made before data collection, not afterwards.

[2] The problem is not including control variables per se, but doing so selectively based on whether they lead to significance. Including relevant controls is good practice, but the decision should not depend on the outcome.

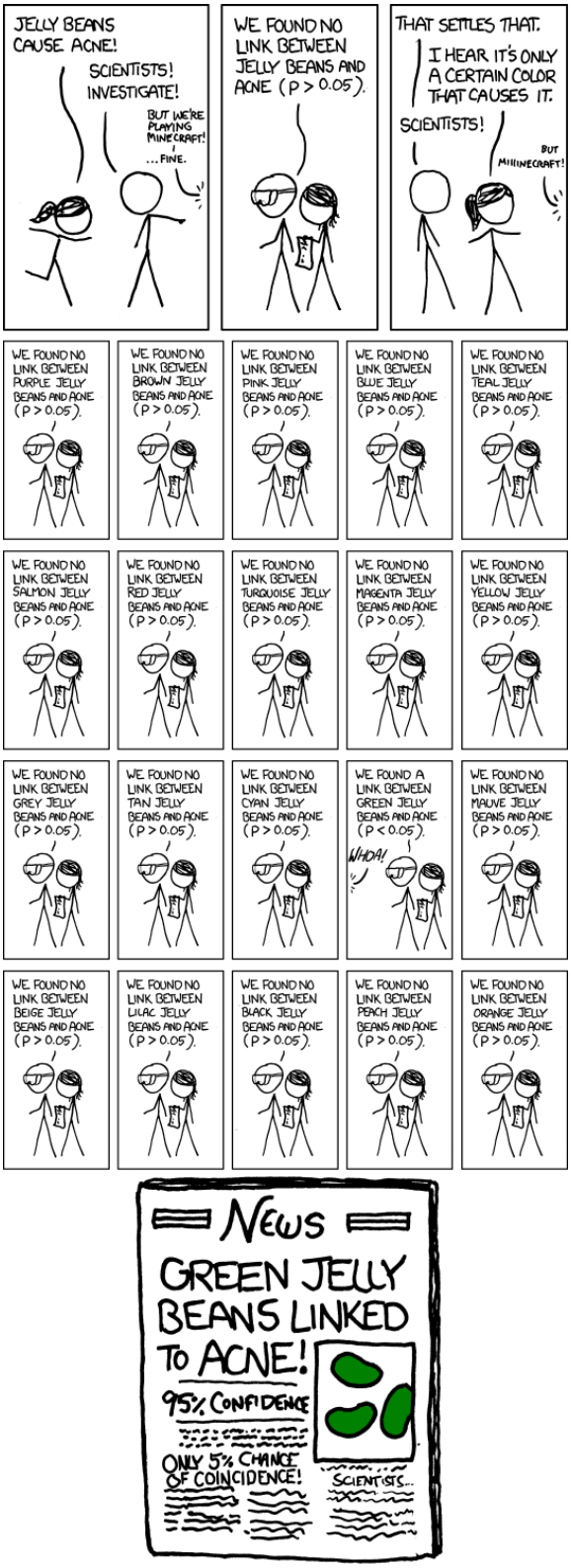


Figure 17.1: Source: `https://xkcd.com/882/`.

- designing multiple conditions but omitting some if doing so produces 'better' results.

Of course, other practices can also inflate the error rate—for example, handling outliers inconsistently. The overarching problem is that researchers treat significant results as 'good' or 'informative' and non-significant results as 'bad' or 'uninformative.' This mindset then justifies flexible decisions during data collection or analysis (e.g., including or omitting a control variable, or keeping or discarding certain outliers) when they lead to significance, even if such decisions would not have been made had the results been non-significant.

It is worth noting that the authors no longer endorse their second recommendation (*Authors must collect at least 20 observations per cell or else provide a compelling cost-of-data-collection justification*); see Simmons et al. (2018).

Complementary to this article are Gelman & Loken (2013) and Steegen et al. (2016), which show how much flexibility researchers have in data analysis (see also Poarch et al., 2019). Scientific hypotheses often correspond to multiple statistical hypotheses (and vice versa). This underdetermination can lead researchers, even unconsciously, to focus on patterns that seem compatible with their theory, while overlooking alternative analyses that might produce results incompatible with the theory. A similar phenomenon is documented in an ethnographic study by Peterson (2016).

Part V

Miscellaneous topics

Chapter 18

*Within-subjects experiments

In the experiments we have considered so far, participants were randomly assigned to the experimental conditions. In such cases, the value of the variable *Condition* varies *between* participants but remains constant within each participant. These are therefore called **between-subjects** experiments.

It is sometimes possible, however, to design an experiment in which all participants take part in multiple conditions. In this case, the variable *Condition* also varies *within* participants, since each participant contributes data points from several (often all) conditions. Such studies are called **within-subjects** experiments.

Compared with between-subjects experiments, within-subjects experiments have the advantage that an important source of error variance is completely controlled for—namely, the pre-existing differences between participants. As a result, within-subjects designs usually yield more precise estimates and greater statistical power than between-subjects designs with the same number of participants.

A potential drawback, however, is that the order in which participants experience the conditions may affect their performance. To prevent such order effects from biasing the comparison of conditions, researchers typically vary the order of conditions across participants—for example, through complete counterbalancing or by using Latin squares. For further details, see the lecture notes *Quantitative methodology: An introduction*, Chapter 7.

In applied linguistics and psycholinguistics, experiments are often set up in a more complex way than simply including a single between- or within-subjects factor. Participants frequently respond to stimuli, and conditions may vary either within stimuli or between stimuli. Nowadays, data from such more complex designs are usually analysed with so-called mixed-effects models. Nevertheless, it is often possible to restructure datasets from these studies so that they can be analysed with simpler methods. This chapter shows how.

18.1 Within-subjects experiments without order effects

van den Broek et al. (2018) investigated whether foreign-language vocabulary is learnt more effectively when learners practise words with context-rich or with context-poor tasks. Their first experiment involved 45 participants, who were asked to learn 104 words in an unfamiliar language (Swahili). During practice, learners had to translate the target words from Swahili into Dutch. For each participant, half of the words were presented with semantically informative contexts (e.g., *I'm going to the bakery. We've run out of* mkate, where *mkate* means 'bread') and half without such context (e.g., *We've run out of* mkate.). Moreover, each word was practised with a rich context by some participants and with a poor context by others. Thus, the factor *Context* varied both within participants and within words. The order of the practice items was randomised for each participant. A first post-test, covering 50 words (25 per condition), was administered immediately. A week later, a second post-test was given with the remaining 54 words (27 per condition). For each participant, a given word was tested either immediately or after a delay.

In what follows, we will examine the effect of learning condition on the accuracy of translations from Swahili into Dutch. The dataset is available as supplementary material for van den Broek et al. (2018) at `https://osf.io/eujyn/`, but I have also saved it locally as `VanDenBroek2018_Exp1.csv`. We load the file, keeping only the columns we actually need:

```
d <- read_csv(here("data", "VanDenBroek2018_Exp1.csv")) |>
  select(PP, Word, Translation, Condition, RepeatedTest, Test2Swa2NlLenient)
```

The column `PP` contains the participants' identification numbers. The columns `Word` and `Translation` contain the Swahili word and its correct Dutch translation. The column `Condition` indicates in which condition the participant practised the word: with context (`context`) or without context (`retrieval`). The column `RepeatedTest` shows whether the word already appeared in the immediate post-test; here we are only interested in the rows where the value is `FALSE`. The column `Test2Swa2NlLenient` indicates whether the word was translated correctly (`TRUE`) or not (`FALSE`).

We cannot simply analyse the data in their current form using a general linear model: because we have multiple observations per participant as well as multiple observations per vocabulary item, the crucial independence assumption would be violated. One way around this problem is to aggregate the data in a suitable way.

Exercise 18.1 (Participant-level summary)**.** Create a tibble `d_perPart` that contains, for each participant, the proportion of correctly translated words practised in the `retrieval` condition and the proportion of correctly translated words practised in the `context` condition. Only rows with `RepeatedTest == FALSE` should be included. The tibble should have three columns: `PP`, `retrieval`, and `context`.

For self-checking: for participant 42, you should obtain the following values:

```
d_perPart |> filter(PP == 42)

# A tibble: 1 x 3
```

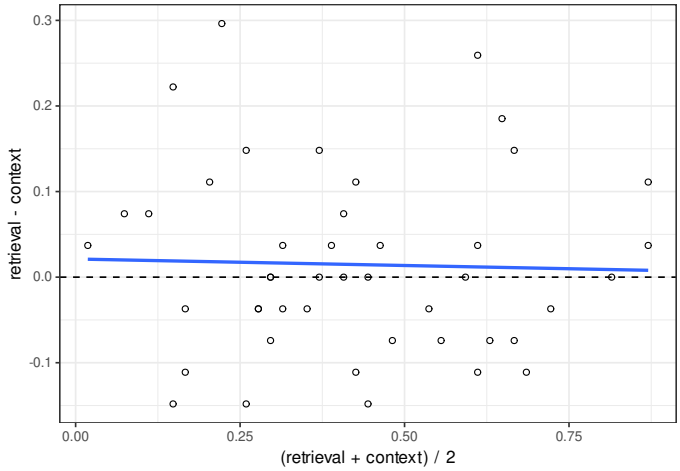


Figure 18.1: Tukey mean–difference plot.

```
     PP context retrieval
  <dbl>   <dbl>     <dbl>
1    42   0.370     0.444
```

◇

The question is to what extent the scores in the `retrieval` condition differ from those in the `context` condition. We add two new columns to `d_perPart`: one for the difference between the two conditions and one for their average:

```
d_perPart <- d_perPart |>
  mutate(difference = retrieval - context,
         average = (retrieval + context) / 2)
```

We can now draw a **Tukey mean–difference plot** (also known as a **Bland–Altman plot** in medical circles). This shows, for each participant, the difference between the two conditions in relation to their average performance across conditions. In Figure 18.1, a trend line has also been added to highlight this relationship. On average, participants seem to perform slightly better in the `retrieval` condition than in the `context` condition, but quite a few participants actually perform better in the `context` than in the `retrieval` condition. There doesn't seem to be a systematic relationship between the participants' average performance and the difference between their performance in the two conditions; this would have shown up as a sloping trend line.

We denote by X_i the result of the i-th participant in the `retrieval` condition, by Y_i their result in the `context` condition, and by $D_i := X_i - Y_i$ the difference between the two. The null hypothesis states that any observed difference between the `retrieval` and `context` conditions is due to chance. In the study by van den Broek et al. (2018), the words to be learned were randomly assigned to conditions for each participant. Order effects were also neutralised by randomising the sequence of words for each participant.

Under these circumstances, the null hypothesis implies that for a given participant i, observing D_i is just as likely as observing $-D_i$. In other words, the vector of differences $\boldsymbol{D} = (D_1, \ldots, D_n)$ is **sign-symmetric**.

The first test we'll discuss leverages this sign-symmetry: under the null hypothesis of sign-symmetry, positive and negative differences should be equally likely. Let n_+ be the number of participants who perform better in the `retrieval` condition than in the `context` condition, and n_- the number who perform worse. Cases where the difference is exactly zero are ignored. To avoid floating-point issues, we round the differences first.

```
d_perPart$difference <- round(d_perPart$difference, 7)
n_plus  <- sum(d_perPart$difference > 0)
n_minus <- sum(d_perPart$difference < 0)
```

The null hypothesis implies that

$$n_+ \sim \text{Binomial}(n_+ + n_-, 0.5).$$

With `binom.test()`, we can carry out the so-called **sign test**. In this case, the result ($p = 1$) is trivial because $n_+ = n_-$:

```
binom.test(n_plus, n_plus + n_minus, p = 0.5)


 Exact binomial test

data:  n_plus and n_plus + n_minus
number of successes = 19, number of trials = 38,
p-value = 1
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.333789 0.666211
sample estimates:
probability of success
                   0.5
```

The sign test is a valid test of the null hypothesis, but it ignores the magnitude of the differences. It could be the case, for example, that the numbers of positive and negative differences are roughly equal, yet the positive ones tend to be larger. A second way of testing the null hypothesis of sign symmetry is therefore to compute the mean (or the sum) of the observed differences (about 1.5 percentage points in this case) and then ask how often such a mean (or sum) would arise if the signs of the differences were randomly flipped.

We can approximate the distribution of the mean of the differences under the null hypothesis of sign symmetry by simulation. In the code below, `b` is a random sign vector, i.e., a vector of n independent entries drawn from $-1, 1$, with each value equally likely:

```
mean_diffs_H0 <- function(d, m) {
  n <- length(d)
  distr <- replicate(m, {
    b <- sample(c(-1, 1), n, replace = TRUE)
    mean(b * d)
  })
  distr
}

m <- 19999
mean_diffs_H0_distr <- mean_diffs_H0(d_perPart$difference, m)
```

As usual, we can then compute the one-sided and two-sided p-values:

```
l <- sum(mean_diffs_H0_distr <= mean(d_perPart$difference))
r <- sum(mean_diffs_H0_distr >= mean(d_perPart$difference))
p_l <- (l + 1) / (m + 1)
p_r <- (r + 1) / (m + 1)
(p <- 2 * min(p_l, p_r))

[1] 0.3631
```

That is, the observed mean difference of 1.5 percentage points isn't significant ($p = 0.36$).

Dümbgen (2016) refers to this procedure as the **sign-t test**. Like randomisation and permutation tests, it was historically too computationally intensive to be practical. A widely used approximation is the **paired-samples t-test**. Here we compute

$$T = \frac{\overline{D}}{s(\boldsymbol{D})/\sqrt{n}},$$

where $\overline{D}$ is the mean of the differences, $s(\boldsymbol{D})$ their standard deviation, and n the number of differences. If we assume not only sign symmetry but also that the differences are normally distributed, then

$$T \sim t_{n-1}.$$

From this t-distribution we can calculate a p-value. If the differences are not normally distributed, the test can still be used as an approximation:

```
n <- length(d_perPart$difference)
mean_diff <- mean(d_perPart$difference)
sd_diff <- sd(d_perPart$difference)
(t_stat <- mean_diff / (sd_diff / sqrt(length(d_perPart$difference))))

[1] 0.924321

p_l <- pt(t_stat, df = n - 1)
p_r <- 1 - pt(t_stat, df = n - 1)
2 * min(p_l, p_r)
```

```
[1] 0.360361
```

That is, $p = 0.36$.

Of course, this is more conveniently done with the `t.test()` function:

```
t.test(d_perPart$difference)


 One Sample t-test

data:  d_perPart$difference
t = 0.9243, df = 44, p-value = 0.36
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -0.0174871  0.0471167
sample estimates:
mean of x
0.0148148

# alternatively
# t.test(d_perPart$retrieval, d_perPart$context, paired = TRUE)
```

The `t.test()` function also outputs a confidence interval for the difference between the conditions. This confidence interval can be double-checked by bootstrapping the difference scores.

Both the sign-t test and the paired-samples t-test share a potential weakness: outliers can exert disproportionate influence on the result. One way to reduce this sensitivity is to work with the ranks of the differences rather than their raw values. The procedure is as follows:

1. Remove any differences where $D_i = 0$.
2. For the remaining differences, compute the ranks of their absolute values. Denote these ranks by $\widetilde{R}_i$.
3. Assign the sign of each difference to its rank:

$$R_i := \begin{cases} \widetilde{R}_i, & \text{if } D_i > 0, \\ -\widetilde{R}_i, & \text{if } D_i < 0. \end{cases}$$

4. Compute the sum of the signed ranks R_i.

In R:

```
signed_rank_sum <- function(d) {
  d <- d[d != 0]
  signed_ranks <- sign(d) * rank(abs(d))
  sum(signed_ranks)
```

```
}
signed_rank_sum(d_perPart$difference)

[1] 79
```

We then compare this statistic with the distribution it would have under the null hypothesis of sign symmetry. The logic and procedure are the same as for the sign-t test, except that we now work with the sum of signed ranks:

```
sum_signed_ranks_H0 <- function(d, m) {
  n <- length(d)
  distr <- replicate(m, {
    b <- sample(c(-1, 1), n, replace = TRUE)
    signed_rank_sum(b * d)
  })
  distr
}

m <- 19999
sum_signed_ranks_H0_distr <- sum_signed_ranks_H0(d_perPart$difference, m)

l <- sum(sum_signed_ranks_H0_distr <= signed_rank_sum(d_perPart$difference))
r <- sum(sum_signed_ranks_H0_distr >= signed_rank_sum(d_perPart$difference))
p_l <- (l + 1) / (m + 1)
p_r <- (r + 1) / (m + 1)
(p <- 2 * min(p_l, p_r))

[1] 0.5786
```

That is, $p = 0.58$.

This procedure corresponds to the **Wilcoxon signed-rank test**, which is implemented in R as `wilcox.test()`. This function often relies on an approximation, though—for example, when sample sizes are large, when some differences equal zero, or when several of the absolute ranks $\widetilde{R}_i$ are tied:

```
wilcox.test(d_perPart$difference)


 Wilcoxon signed rank test with continuity correction

data:  d_perPart$difference
V = 410, p-value = 0.569
alternative hypothesis: true location is not equal to 0

# or (with a slightly different result due rounding):
# wilcox.test(d_perPart$retrieval, d_perPart$context, paired = TRUE)
```

In the output, the test statistic `V` does not represent the sum of all R_i, but rather the larger of the sum of the positive R_i or the sum of the negative R_i. However, this statistic is directly related to the overall sum of all signed ranks:

```
n_not0 <- length(d_perPart$difference[d_perPart$difference != 0])
2 * 410 - n_not0 * (n_not0 + 1) / 2
```

```
[1] 79
```

Exercise 18.2. The file `correspondencerules.csv` contains data from the study by Vanhove (2016). Eighty native German speakers read Dutch words that had German cognates. Participants were randomly assigned to one of two learning conditions: about half received some Dutch words containing the digraph *oe*, which corresponds to the German *u* (e.g., *proesten–prusten* 'to snort'), while the other half received some Dutch words containing the digraph *ij*, corresponding to the German *ei* (e.g., *twijfel–Zweifel* 'doubt'). These learning conditions are recorded in the dataset under the column `LearningCondition` as `oe-u` or `ij-ei`.

Afterwards, all participants were tested on new Dutch words, including items with *oe* and items with *ij*. The research question was whether prior experience with a digraph (either *oe* or *ij*) helps participants decode new words containing that digraph. Thus, although participants were only assigned to one learning condition, they were tested in both conditions. This makes it a within-subjects experiment.

1. Read the dataset into R.
2. Keep only the rows where `Category` is either `oe cognate` or `ij cognate` and `Block != "Training"`. You should end up with 3360 rows. Name the resulting dataset `correspondences`.
3. For each participant (`Subject`), calculate the proportion of trials in which they correctly decoded the digraph (`CorrectVowel == "yes"`), separately for the two relevant word categories (`Category`).
4. Compute, for each participant, the difference between the proportion of correct decodings for the learned digraph and the proportion for the new digraph. A positive difference indicates that participants performed better on the already-learned digraph.

   Hint: Use `LearningCondition` as a grouping factor in the previous step. Then convert the resulting summary to a wide format and calculate differences using `ifelse()`.
5. Create and interpret a Tukey mean–difference plot.
6. Perform an appropriate significance test and draw a brief conclusion (1–3 sentences).

Remark 18.3 (Same data, different perspective)**.** We have so far looked at the data from van den Broek et al. (2018) and Vanhove (2016) from the perspective of the participants. However, in both studies it is also possible to view the data from the perspective of the

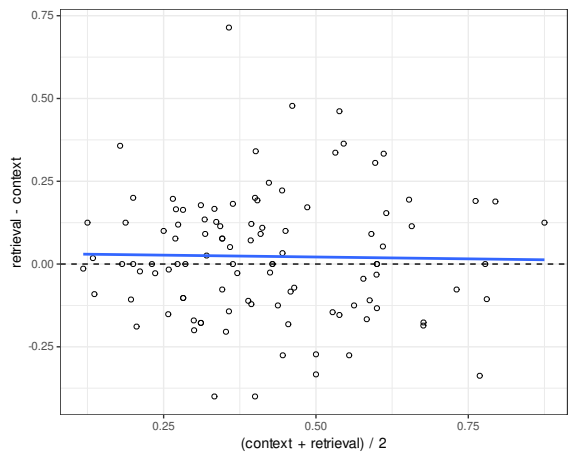


Figure 18.2: Another Tukey mean–difference plot of the van den Broek et al. (2018) data. This time, the data were aggregated by vocabulary item rather than by participant.

stimuli. In the study by van den Broek et al. (2018), all words were learnt in both the `retrieval` and `context` conditions, so we can also summarise and visualise the data as shown in Figure 18.2. Rather than subjecting the participant-level differences between conditions to a significance test, we could instead analyse the differences between conditions at the level of the words.

```
d_items <- d |>
  filter(!RepeatedTest) |>
  group_by(Word, Translation, Condition) |>
  summarise(prop_correct = mean(Test2Swa2NlLenient),
            .groups = "drop") |>
  mutate(Item = paste0(Word, " (", Translation, ")")) |>
  pivot_wider(names_from = "Condition", values_from = "prop_correct") |>
  mutate(Difference = retrieval - context)

ggplot(d_items,
       aes(x = (context + retrieval) / 2,
           y = Difference)) +
  geom_point(shape = 1) +
  geom_hline(yintercept = 0, linetype = "dashed") +
  geom_smooth(method = "lm", se = FALSE, formula = y ~ x) +
  xlab("(context + retrieval) / 2") +
  ylab("retrieval - context")
```

One potential advantage of a stimulus-level analysis is that the labels of the stimuli often convey more information than the typically anonymised participant IDs. Especially when the number of stimuli is still manageable, graphical representations like Figure 18.3 are useful. This figure shows, on the one hand, that there is a fairly consistent difference

between conditions within each category, and on the other hand, that some words were easy for everyone (*schrijven* 'to write'), while others were difficult for all participants (*schijf* 'slice', *schoen* 'shoe').

```
# correspondences is the dataset containg the Vanhove (2016) data
correspondences_items <- correspondences |>
  group_by(Item, Category, LearningCondition) |>
  summarise(prop_correct = mean(CorrectVowel == "yes"),
            .groups = "drop")

ggplot(correspondences_items,
       aes(x = prop_correct,
           y = reorder(Item, prop_correct),
           shape = LearningCondition)) +
  geom_point() +
  scale_shape_manual(values = c(1, 3),
                     name = "Learning condition") +
  facet_grid(rows = vars(Category),
             scales = "free_y") +
  xlab("Proportion of successful decodings") +
  ylab(element_blank()) +
  theme(legend.position = "bottom")
```

Which perspective is most informative will likely depend on the specific study. There is certainly no objection to displaying the data graphically from both perspectives. What you should *not* do, however, is run multiple significance tests on the same data and then report only the one yielding the lowest p-value. ♢

Exercise 18.4. Consider a pedagogical experiment along the following lines. We want to investigate the interplay between two factors A and B on pupils' performance. To that end, we recruit twelve classes with an average of twenty pupils per class. Factor A can only be manipulated practically between classes, not within classes. (Factor A may, for instance, be related to something the teachers do.) Hence, six classes are randomly assigned in their entirety to condition A_1, and the remaining six classes are assigned in their entirety to condition A_2. Factor B, by contrast, is easily manipulated within classes. Hence, within each class, we randomly assign half of the pupils to condition B_1 and half to condition B_2—that is, the assignment of factor B is blocked on class. Each pupils, then, is assigned to one A condition and one B condition.

The resulting dataset contains 360 entries—one for each pupil in the study. The research questions are as follows:

1. Averaged over the two levels of factor A, do pupils assigned to condition B_1 perform better than those assigned to condition B_2?

2. Does the difference in performance between conditions B_1 and B_2 depend on whether the pupils were assigned to condition A_1 or A_2?

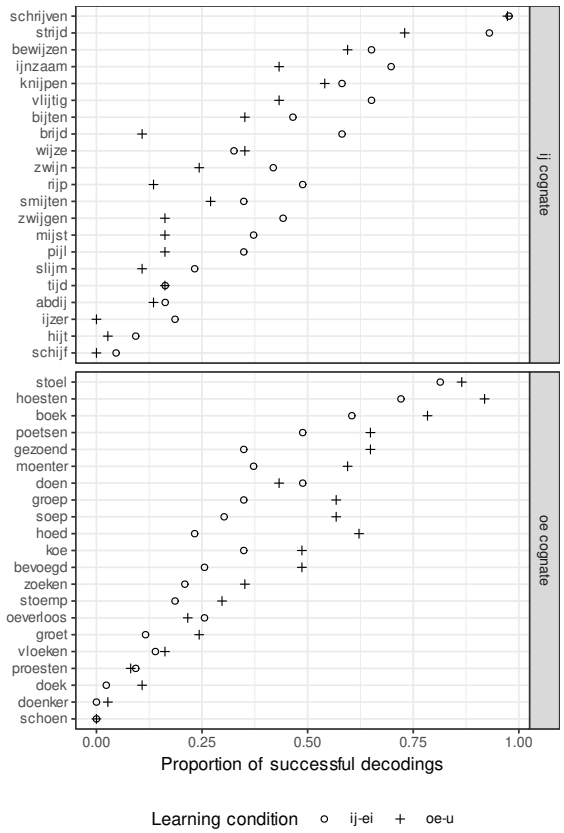


Figure 18.3: Proportion of correctly decodings per word in Vanhove (2016). The words are ordered by category and, within their category, by the average proportion of successful decodings.

1. Sketch a plot that would allow you to compare the pupils' performance according to the conditions they were assigned to. Explain how you could glean the answers to the two research questions from the plot.

2. You shouldn't analyse these data directly in a general linear model. Explain why.

3. Suggest a work-around that would allow you to use a general linear model to analyse these data. Explain how you can glean the answers to the two research questions from the model output.

4. How could you analyse if, averaged over the two levels of factor B, there's a difference in performance between condition A_1 and condition A_2? ♢

Remark 18.5 (RM-ANOVA and mixed models)**.** In the two examples in this chapter, only two conditions are compared. For within-subjects studies with more than two conditions, the standard approach in the past was usually a repeated-measures ANOVA. Today, studies with multiple observations per participant or per stimulus are often analysed using so-called mixed-effects models. In fact, both van den Broek et al. (2018) and Vanhove (2016) analysed their data with mixed-effects logistic models. However, the technical requirements for such models are considerable; the simpler alternatives discussed in this chapter are likely more accessible.

Regardless of which analysis method you choose, it is worth investing time and thought into graphical displays so that readers who are not familiar with your analysis approach can still understand your research report. ♢

18.2 AB/BA cross-over experiments

Consider a within-subjects experiment with two conditions, creatively labelled A and B, in which participants are randomly assigned to the two possible orders AB and BA.[1] This design is called a **AB/BA cross-over design**. We will consider condition A to be the intervention (or treatment) condition and B the control condition, but nothing hinges on these labels. Let's break down the systematic factors that contribute to the first and second measurements in both orders, see Table 18.1.

- We assume that there is some baseline β common to all measurements. This common baseline is of no further interest.

- There may be treatment effect τ that contributes to the measurements in condition A (i.e., the first measurement for order AB and the second for order BA), but not to those in condition B.[2]

- There may be an order effect ω that contributes to the second measurements but not to the first measurements.[3]

[1]This section is adapted from *Quantitative methodology: An introduction*, Chapter 7.

[2]If there is a treatment effect τ' that contributes to the measurements in condition B, this is equivalent to there being a treatment effect $\tau = -\tau'$ that contributes to the measurements in condition A.

[3]Similarly, if an order effect ω' contributes to the first measurements instead, write $\omega = -\omega'$.

Table 18.1: Systematic factors contributing to the measurements in a within-subjects experiment with two conditions. The meaning of the Greek symbols is explained in the running text.

Order	First measurement	Second measurement
AB	$\beta + \tau$	$\beta + \omega + \kappa$
BA	β	$\beta + \tau + \omega$

- There may be a carryover effect κ that affects the measurements in condition B but only if B follows A.[4]

In any analysis of these data, we need to assume that there are no carryover effects, that is, $\kappa = 0$. If $\kappa \neq 0$, a within-subjects design was the wrong choice. Under the assumption that $\kappa = 0$, we may estimate the value of τ by first computing, for each participant, the period difference, that is, the difference between their first and their second measurement. For the participants in the AB order, and ignoring any non-systematic effects, we obtain

$$d_{AB} = (\beta + \tau) - (\beta + \omega + \kappa) = \tau - \omega - \kappa = \tau - \omega,$$

since $\kappa = 0$ by assumption. For the participants in the BA order, we similarly obtain

$$d_{BA} = \beta - (\beta + \tau + \omega) = -\tau - \omega.$$

Observe that

$$\frac{d_{AB} - d_{BA}}{2} = \frac{(\tau - \omega) - (-\tau - \omega)}{2} = \tau.$$

The consequence of this is that, under the assumption of no carryover effects, the treatment effect τ can be estimated as half the mean difference in the period differences between the two orders. A randomisation test or some analytical approximation thereof can be used to obtain a p-value for the null hypothesis that $\tau = 0$, and the usual techniques (general linear model, bootstrapping) can be used to construct a confidence interval for τ.

```
set.seed(2025-09-12)
```

[4]Similarly, if a carryover κ' affects A when following B, write $\kappa = -\kappa'$.

Chapter 19

*Logistic regression

In language research, we often deal with binary outcome data such as accuracy (correct/incorrect), presence vs. absence, etc. It is possible to analyse such binary data in a general linear model by coding one level of the variable as 1 and the other as 0. The predicted values can then be interpreted as the estimated success probabilities (e.g., probability of a correct answer, of the presence of a given phoneme, etc.). This is referred to as the **linear probability model**, and it may be an appropriate tool when analysing the data from experiments with categorical predictors (see Hellevik, 2009; Huang, 2019). While the errors of such models are bound to violate the assumptions of equality of distributions and normality, bootstrapping techniques can be used to double-check the inferences drawn from such models under different sets of assumptions.

That said, researchers should be aware of some possible problems with the linear probability model (see Jaeger, 2008). An especially glaring problem is that linear probability models with continuous predictors can generate impossible predictions, namely probabilities smaller than 0 or greater than 1. Even for categorical predictors, such models can yield confidence intervals that lie partially outside of the $[0, 1]$ interval.

While the latter problem can be addressed in linear probability models by constructing the confidence intervals using bootstrap techniques, another option is to adjust the general linear model so that it can accommodate binary (and other) outcome types. This gives rise to the **generalised linear model** (I didn't come up with these names...). In our field, the most common type of generalised linear model beside the general linear model is the **logistic regression model** for dealing with binary outcome data, which we'll introduce in this chapter.

19.1 Categorical predictors

Tversky & Kahneman (1981) presented their participants with a hypothetical scenario and asked them to choose between one of two actions:

> "Imagine that the U.S. is preparing for the outbreak of a an unusual Asian disease, which is expected to kill 600 people. Two alternative programs to

Table 19.1: Number of choices for Programme A and Programme B depending on how the programmes were presented (Tversky & Kahneman, 1981).

	Programme A	Programme B
Gain framing	109	43
Loss framing	34	121

> combat the disease have been proposed. Assume that the exact scientific estimate of the consequences of the programs are as follows:"

For about half of the participants, the consequences of these programmes were framed as gains (*gain framing*):

> "If Program A is adopted, 200 people will be saved.
>
> If Program B is adopted, there is a 1/3 probability that 600 people will be saved, and a 2/3 probability that no people will be saved."

For the other participants, the consequences of these programmes were framed as losses (*loss framing*):

> "If Program A is adopted, 400 people will die.
>
> If Program B is adopted, there is a 1/3 probability that nobody will die, and a 2/3 probability that 600 people will die."

Note that the outcome of Programme A is the same regardless of how its consequences are framed, and the same goes for Programme B. Note further that both programmes have the same expected value in terms of lives saved, namely 200. The only difference is that this is a certainty in Programme A and a probabilistic expectation in Programme B. Table 19.1 shows how the participants' preferences varied between the framing conditions.

Let's reconstruct the dataset:

```
d <- tibble(
  framing = c(rep("gain", 109 + 43),
              rep("loss", 34 + 121)),
  choice = c(rep("A", 109), rep("B", 43),
             rep("A", 34), rep("B", 121))
)
```

We can compute the proportion of choices for Programme A (the sure option) like so:

```
d |>
  group_by(framing) |>
  summarise(proportion_sure = mean(choice == "A"))

# A tibble: 2 x 2
```

```
  framing proportion_sure
  <chr>             <dbl>
1 gain              0.717
2 loss              0.219
```

Using the linear probability model, the difference in the proportions can be retrieved, and uncertainty estimates for this difference can be computed:

```
d <- d |>
  mutate(sure_choice = ifelse(choice == "A", 1, 0),
         n.loss = ifelse(framing == "loss", 1, 0))  # treatment coding
tversky.lm <- lm(sure_choice ~ n.loss, data = d)
summary(tversky.lm)$coefficients
```

```
             Estimate Std. Error  t value    Pr(>|t|)
(Intercept)  0.717105  0.0351803  20.3837 6.99440e-59
n.loss      -0.497750  0.0495111 -10.0533 1.00341e-20
```

```
confint(tversky.lm)
```

```
                2.5 %    97.5 %
(Intercept)  0.647879  0.786332
n.loss      -0.595177 -0.400324
```

That is, the percentage of sure choices is just shy of 50 ± 5 percentage points lower in the loss framing condition than in the gain framing condition. The 95% confidence interval spans from -60 to -40 percentage points. You can double-check these confidence intervals using a semiparametric bootstrap that does not assume equality of distributions, but we won't do so here.

As explained in the introduction, a possible problem with the linear probability model is that the model generates predictions (and confidence intervals) that go outside the permissible range for probabilities. While this isn't the case here, we still need to work out a solution should we ever need it. One such solution is to not work directly with proportions or probabilities but to work with so-called log-odds instead. To understand these, we first need to understand what odds and odds ratios are.

Instead of saying that the proportion of sure choices under gain framing was 0.7171, we could also say that for each unsure choice, there are about 2.53 sure choices:

$$\frac{109}{43} = \frac{109 \Big/ (109+43)}{43 \Big/ (109+43)} = \frac{0.7171}{1-0.7171} \approx 2.53.$$

The **odds** of a sure choice in the gain framing condition, then, are 2.53 to 1. Similarly, the odds of a sure choice in the loss framing condition are

$$\frac{34}{121} = \frac{0.2194}{1-0.2194} \approx 0.28,$$

that is, for each unsure choice there were 0.28 sure choices.

To convert a proportion p to odds φ, then, we can use the formula

$$\varphi = \frac{p}{1-p}.$$

Given some odds φ, we can backtransform these to the proportion p using the formula

$$p = \frac{\varphi}{1+\varphi}.$$

In order to express the preference difference between the gain and loss framing conditions, we can compute an **odds ratio**. This shows that the odds of a sure choice in the loss framing condition are

$$\frac{34/121}{109/43} = \frac{0.28}{2.53} = 0.11$$

as large as in the gain framing condition. Equivalently, the odds of a sure choice are about 9 times as large in the gain framing condition as in the loss framing condition.

Odds and odds ratios cannot be negative. But by taking their (natural) logarithm, we can further transform them so that their theoretical range is the entire real axis. The natural logarithm of odds or odds ratios are called **log-odds**. In our example, the log-odds of a sure choice in the gain framing condition are

$$\ln 2.53 \approx 0.93,$$

compared to

$$\ln 0.28 \approx -1.27$$

in the loss framing condition. Due to the properties of logarithms ($\log a/b = \log a - \log b$), the difference between these log-odds is the logarithm of the odds ratio:

$$\ln 0.28 - \ln 2.53 = \ln \frac{0.28}{2.53} = \ln 0.11 \approx -2.21.$$

As these computations show, a proportion or probability p can be converted to log-odds η using the following formula:

$$\eta = \ln \varphi = \ln \frac{p}{1-p}.$$

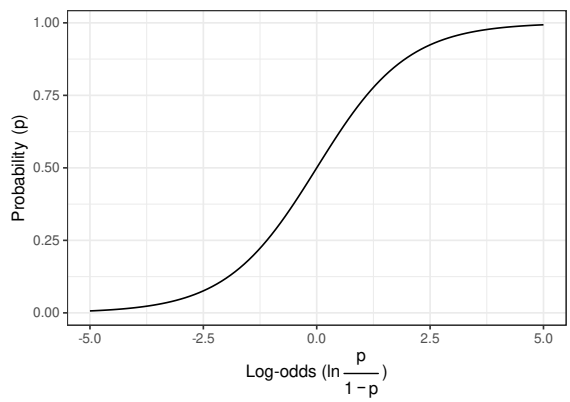


Figure 19.1: The logistic function converts log-odds into probabilities.

This function, which maps proportions to log-odds, is known as the **logit function**. To compute proportions or probabilities from log-odds, we need to find its inverse:

$$
\begin{aligned}
& & \eta &= \ln \frac{p}{1-p} \\
&\Rightarrow & \exp(\eta) &= \frac{p}{1-p} \\
&\Rightarrow & \exp(\eta) - p\exp(\eta) &= p \\
&\Rightarrow & \exp(\eta) &= p(1+\exp(\eta)) \\
&\Rightarrow & p &= \frac{\exp(\eta)}{1+\exp(\eta)} = \frac{1}{1/(\exp(\eta))+1} = \frac{1}{1+\exp(-\eta)}.
\end{aligned}
$$

For instance, given the log-odds of 0.93, we can compute the original proportion like so:

```
1 / (1 + exp(-0.93))
```

```
[1] 0.717075
```

The function for converting log-odds into probabilities is called the **logistic function**. It is implemented in R as `plogis()`:

```
plogis(0.93)
```

```
[1] 0.717075
```

Figure 19.1 shows that the logistic function looks like.

For our present example, we now let p_i denote the probability that the i-th participant prefers the sure programme, $i = 1, \ldots, 307$. Then we can conceive of each participant's decision as a Bernoulli experiment with success probability p_i. More formally, the choice of the i-th participant is modelled as a Bernoulli random variable $X_i \sim \text{Bernoulli}(p_i)$. Our goal is to estimate the p_is. We do so by modelling them as log-odds in a linear

model and then backtransforming these into probabilities:

$$
\begin{aligned}
X_i &\sim \text{Bernoulli}(p_i), \\
p_i &= \frac{1}{1+\exp(-\eta_i)}, \\
\eta_i &= \beta_0 + \beta_1 x_i,
\end{aligned}
$$

where $x_i = 1$ if the i-th participant was presented the scenario with loss framing and $x_i = 0$ otherwise and where $X_1, \ldots X_{307}$ are independent.

The model parameters are estimated using maximum likelihood estimation, which was mentioned at the end of Chapter 7. In brief, we pick as our estimates $\widehat{\beta}_0, \widehat{\beta}_1$ the β_0 and β_1 given which the data we have observed would be most likely to occur:

$$
\begin{aligned}
(\widehat{\beta}_0, \widehat{\beta}_1) &= \underset{(\beta_0,\beta_1)}{\arg\max} \prod_{i=1}^{307} p_i^{X_i}(1-p_i)^{1-X_i} \\
&= \underset{(\beta_0,\beta_1)}{\arg\max} \sum_{i=1}^{307} \left(X_i \log(p_i) + (1-X_i)\log(1-p_i)\right),
\end{aligned}
$$

where

$$
p_i = \frac{1}{1+\exp(-(\beta_0+\beta_1 x_i))}.
$$

Figure 19.2 shows that a $\widehat{\beta}_0$ value around 1 and a $\widehat{\beta}_1$ value around -2 maximise these data's likelihood.

The same result can be obtained more quickly using the `glm()` function. It works similarly to the `lm()` we're already familiar with, except that we need to specify a distribution family and a so-called link function. As mentioned above, we consider the participants' choices to be Bernoulli distributed. The Bernoulli distribution is a more specific instance of the binomial distribution, which explain the choice for the `family` parameter. The choice of the logit link function is due to our working with log-odds.

```
tversky.glm <- glm(sure_choice ~ n.loss, data = d,
                   family = binomial(link = "logit"))
summary(tversky.glm)$coefficients
```

```
             Estimate Std. Error  z value    Pr(>|z|)
(Intercept)  0.930148   0.180084  5.16509 2.40323e-07
n.loss      -2.199578   0.264776 -8.30732 9.78882e-17
```

The interpretation of the estimated model parameters is as follows:

- The log-odds of a sure choice when the `n.loss` value is 0 are about 0.93. This corresponds to odds of about 2.53 and to a probability of about 0.72:

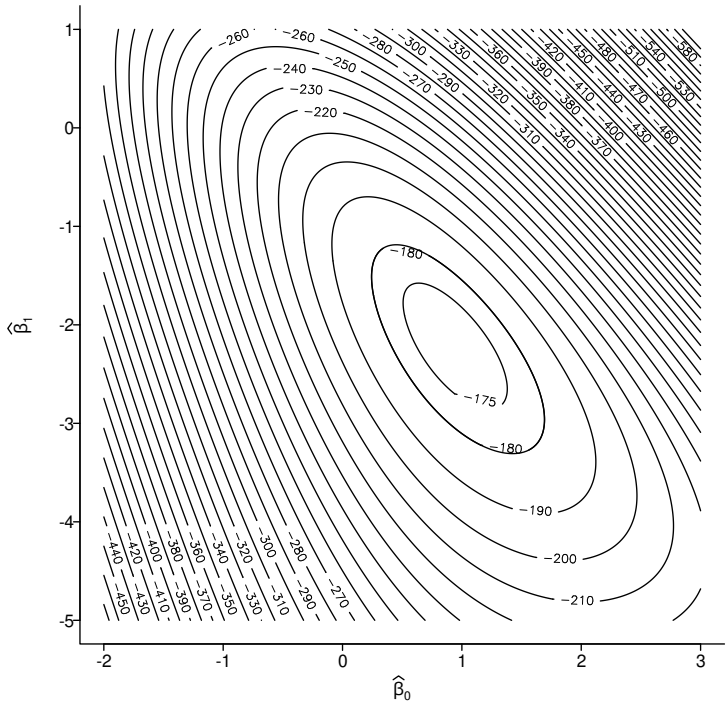


Figure 19.2: The log-likelihood of different choices for $\widehat{\beta}_0$ and $\widehat{\beta}_1$. The log-likelihood is maximised for $\widehat{\beta}_0$ values near 1 and $\widehat{\beta}_1$ values around -2.

```
exp(0.93)

[1] 2.53451

plogis(0.93)

[1] 0.717075
```

- The difference between the log-odds of a sure choice when the `n.loss` value is 1 compared to when it is 0 is -2.2. This corresponds to an odds ratio of 0.11:

```
exp(-2.2)

[1] 0.110803
```

- In order to obtain the estimated probability of a sure choice in the loss framing condition, we need to compute the log-odds of this first and then convert them into probabilities using the logistic function:

```
plogis(0.93 - 2.2)

[1] 0.219257
```

We had already computed all these numbers by hand. The added value of the model is that it produces standard errors for the estimated β parameters as well as approximate confidence intervals. Note that these are expressed in log-odds, too!

```
confint(tversky.glm)

                2.5 %   97.5 %
(Intercept)  0.585268  1.29314
n.loss      -2.730843 -1.69128
```

We can express these confidence intervals on the odds scale by exponentiation:

```
exp(confint(tversky.glm))

                2.5 %   97.5 %
(Intercept) 1.7954719 3.644196
n.loss      0.0651643 0.184283
```

That is, the 95% interval for the estimated odds ratio of 0.11 is about $[0.065, 0.184]$. While *odds* can be converted into probabilities, odds *ratios* cannot. If you want to estimate the effect of framing in percentage points and obtain a confidence interval that is also expressed in percentages, consider fitting a linear probability model.

Remark 19.1 (Inference in generalised linear models)**.** Under certain assumptions, the confidence intervals and p-values reported in the `lm()` output are exact. This is not the case for models fitted using `glm()`: For generalised linear models, only approximate confidence procedures (based on asymptotic results) are available. That said, it is possible to apply (parametric) bootstrap and permutation procedures that may be more reliable for smaller samples in particular; see Remarks 19.6 and 19.7. ♢

Remark 19.2 (Linear probability model done better)**.** Having introduced the `glm()` function, we can now fit the linear probability model in a slightly better way. Specifically, we assume that

$$\begin{aligned} X_i &\sim \text{Bernoulli}(p_i), \\ p_i &= \eta_i, \\ \eta_i &= \beta_0 + \beta_1 x_i, \end{aligned}$$

$i = 1, \ldots, n$. Compared to the logistic model, the only difference is that the link function is the identity (i.e., $\eta_i = p_i$) instead of the logit function. In contrast to the general linear models considered earlier, there aren't any error terms in this model! So instead of using `lm()`, which estimates error variance, we can use `glm()` with a suitable link function:

```
lpm.glm <- glm(sure_choice ~ n.loss, data = d,
               family = binomial(link = "identity"))
summary(lpm.glm)$coefficients

             Estimate Std. Error  z value    Pr(>|z|)
(Intercept)  0.717105  0.0365327  19.6291 8.72132e-86
n.loss      -0.497750  0.0493903 -10.0779 6.91925e-24
```

Compare the output of this model to the output of the model `tversky.lm`. ♢

Remark 19.3 (Significance tests for 2×2 tables)**.** If a full-fledged model isn't required, it is possible to calculate exact p-values for 2×2 contingency tables. See my blogpost *Exact significance tests for* 2×2 *tables* (10 September 2024) for details. ♢

Generalised linear models can accommodate multiple predictors, as in the following exercise.

Exercise 19.4 (Fitting interactions)**.** Keysar et al. (2012) extended the experiment by Tversky & Kahneman (1981) by having some participants complete the experiment in their native language (English) and others in a foreign language they were learning (Japanese). They were interested in finding out whether the framing effect that Tversky & Kahneman (1981) had observed was as pronounced when the participants deal with the dilemma in a foreign language as it was in their native language. That is, they were interested in the interaction between framing and language.

We read in the data and summarise it numerically:

```
d <- read_csv(here("data", "Keysar2012.csv"))
d |>
  group_by(language, framing) |>
  summarise(
    sure_choices = sum(sure_choice == 1),
    unsure_choices = sum(sure_choice == 0)
  )
```

```
# A tibble: 4 x 4
# Groups:   language [2]
  language framing sure_choices unsure_choices
  <chr>    <chr>          <int>          <int>
1 English  gain              24              7
2 English  loss              14             16
3 Japanese gain              13             17
4 Japanese loss              12             18
```

1. Create a dummy variable for `language` that has the value 0 for English and 1 for Japanese. Also create a dummy variable for `framing` that has the value 0 for gain framing and 1 for loss framing. (That is, use treatment coding.) Fit a logistic model with these dummy variables and their pointwise product (interaction) as predictors.

2. You should have obtained the following estimated model parameters:

```
            Estimate Std. Error
(Intercept)  1.23214   0.429562
n.language  -1.50041   0.565924
n.framing   -1.36568   0.564316
interaction  1.22847   0.770122
```

(a) Explain what each estimated model parameter means literally.

(b) Based on these coefficients, compute the estimated probability that a participant who read the dilemma in the loss-framing condition and in English picks the sure option. Verify your answer using the numerical summary above.

(c) Based on these coefficients, compute the estimated probability that a participant who read the dilemma in the loss-framing condition and in Japanese picks the sure option. Verify your answer using the numerical summary above.

3. Compute a 95% confidence interval for the interaction parameter and convert it to an odds ratio.

4. How would you answer Keysar et al.'s research question? ♢

Exercise 19.5 (Sum coding)**.** We refit Keysar et al.'s data using sum-coded predictors:

```
d <- d |>
  mutate(n.language = ifelse(d$language == "Japanese", yes = 0.5, no = -0.5),
         n.framing = ifelse(d$framing == "loss", yes = 0.5, no = -0.5),
         interaction = n.language*n.framing)
keysar.glm <- glm(sure_choice ~ n.language + n.framing + interaction,
                  data = d,
                  family = binomial(link = "logit"))
summary(keysar.glm)$coefficients[, 1:2]
```

```
             Estimate Std. Error
(Intercept)  0.106221   0.192530
n.language  -0.886171   0.385061
n.framing   -0.751438   0.385061
interaction  1.228474   0.770122
```

The estimate for the interaction parameter is the same as before but the estimates for the other parameters have changed. Explain what the parameter estimates for `(Intercept)`, `n.language` and `n.framing` mean. ♢

Remark 19.6 (Confidence intervals using parametric bootstrapping)**.** The confidence intervals for the model parameters rely on large-sample approximations. One option to verify them is to apply a parametric bootstrap. To this end, we generate new outcome data from the fitted model (using `simulate()`) and then refit the model on these new outcome data and store the parameter estimates we're interested in. The following code snippet constructs a 95% confidence interval for the interaction parameter using the parametric bootstrap and the percentile method.

```
B <- 20000
interaction_bs <- replicate(B, {
  choice_y <- unlist(simulate(keysar.glm))
  keysar_bs <- glm(choice_y ~ n.language + n.framing + interaction,
                   data = d, family = binomial(link = "logit"))
```

```
  coef(keysar_bs)[4]
})
# hist(interaction_bs) # not shown
quantile(interaction_bs, probs = c(0.025, 0.975))
```

```
     2.5%     97.5%
-0.252516  2.944902
```

The upper end of this confidence interval is a fair bit higher than the approximate one you can compute using `confint()`. ♢

Remark 19.7 (p-values in the generalised linear model)**.** Statistical inference in the generalised linear model relies on large-sample approximations, and different methods for computing p-values may give different results.

The first large-sample approximation method is the **Wald method**, which can be obtained using `summary()`. In the present example, the Wald p-value for the interaction term is $p = 0.111$ ($z = 1.60$).

```
summary(keysar.glm)$coefficients
```

```
             Estimate Std. Error   z value  Pr(>|z|)
(Intercept)  0.106221   0.192530  0.551709 0.5811477
n.language  -0.886171   0.385061 -2.301378 0.0213703
n.framing   -0.751438   0.385061 -1.951479 0.0510001
interaction  1.228474   0.770122  1.595168 0.1106746
```

The second large-sample approximation method is the **likelihood ratio test**. It tends to be more accurate than Wald's method and requires you to fit a null model. In the present example, this would be a model with main effects for language and framing but no interaction effect. A model comparison using the `anova()` function yields $p = 0.108$ ($\chi^2_1 = 2.59$). Note though that this analysis is not an analysis of variance that you may have encountered elsewhere but a so-called analysis of deviance. (We won't go into the concept of deviance, but it is closely related to the ratio of the likelihoods of the models being compared.)

```
null_model <- glm(sure_choice ~ n.language + n.framing,
                  data = d, family = binomial(link = "logit"))
anova(null_model, keysar.glm, test = "LRT")
```

```
Analysis of Deviance Table

Model 1: sure_choice ~ n.language + n.framing
Model 2: sure_choice ~ n.language + n.framing + interaction
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1       118      158.6
2       117      156.0  1    2.586    0.108
```

The third large-sample approximation method is the **score test**, which yields $p = 0.109$ ($\chi^2_1 = 2.57$):

```
anova(null_model, keysar.glm, test = "Rao")

Analysis of Deviance Table

Model 1: sure_choice ~ n.language + n.framing
Model 2: sure_choice ~ n.language + n.framing + interaction
  Resid. Df Resid. Dev Df Deviance   Rao Pr(>Chi)
1       118      158.6
2       117      156.0  1    2.586 2.571    0.109
```

Of these three large-sample approximations, the likelihood-ratio test is to be preferred. That said, it relies on the differences in the deviances following a χ^2 distribution under the null hypothesis, which may not be a reasonable assumption for small datasets.

Alternatively, we can use a technique similar to the parametric bootstrap: We generate new outcome data *under the fitted null model* and then refit the full model. We then count in how many cases the estimated interaction term in the models fitted on the null data was at least as extreme as the interaction term that we estimated for the actual data. This proportion serves as a p-value—in our case, $p = 0.114$.

```
m <- 19999
interaction_bs <- replicate(m, {
  choice_y <- unlist(simulate(null_model)) # simulate under null!
  keysar_bs <- glm(choice_y ~ n.language + n.framing + interaction,
                   data = d, family = binomial(link = "logit"))
  coef(keysar_bs)[4]
})
(sum(abs(interaction_bs) >= abs(coef(keysar.glm)[4])) + 1) / (m + 1)

[1] 0.11395
```

We could also carry out a bootstrap version of the likelihood ratio test, which yields a p-value of 0.111.

```
m <- 19999
lrt_bs <- replicate(m, {
  choice_y <- unlist(simulate(null_model)) # simulate under null!
  # fit both null and full model to data
  null_bs <- glm(choice_y ~ n.language + n.framing,
                 data = d, family = binomial(link = "logit"))
  keysar_bs <- glm(choice_y ~ n.language + n.framing + interaction,
                   data = d, family = binomial(link = "logit"))
  # compare fitted models
  anova(null_bs, keysar_bs, test = "LRT")[2, 4]
})
observed_lrt <- anova(null_model, keysar.glm, test = "LRT")[2, 4]
```

```
(sum(lrt_bs >= observed_lrt) + 1) / (m + 1)
```

```
[1] 0.11055
```

Note that with the bootstrap procedure, the results will vary slightly from run to run due to the randomness involved in simulating the new observations. ♢

19.2 Continuous predictors

You can also fit continuous predictors in the generalised linear model. To illustrate the procedure, we'll take a look at how well one of the participants in Vanhove & Berthele (2013) translated 181 words from languages she did not know depending on the orthographic overlap (expressed on a scale from 0 to 10) between these words and their translations.[1]

Let's read in the data and draw a scatterplot including a scatterplot smoother (Figure 19.3). Even though this smoother doesn't take into account that the data are binary (notice how it extends beyond the $[0, 1]$ interval), it highlights a monotonic association between orthographic overlap and the participant's translation success.

```
d <- read_csv(here("data", "VanhoveBerthele2013_oneparticipant.csv"))

ggplot(data = d,
       aes(x = Overlap,
           y = Correct)) +
  geom_point(shape = 1,
             # vertical jittering to better show the data points
             position = position_jitter(width = 0, height = 0.02)) +
  # scatterplot smoother without confidence band
  geom_smooth(se = FALSE)
```

We can fit these data in a logistic model:

```
translation.glm <- glm(Correct ~ Overlap, data = d,
                       family = binomial(link = "logit"))
summary(translation.glm)$coefficients
```

```
             Estimate Std. Error  z value    Pr(>|z|)
(Intercept) -3.313207   0.734010 -4.51384 6.36635e-06
Overlap      0.594433   0.114358  5.19801 2.01429e-07
```

Other than that the estimates are in log-odds, there is nothing new here:

- The `(Intercept)` estimates the log-odds of a correct translation if the degree of orthographic `Overlap` is 0. Expressed in odds, there is a 0.037–to–1 chance of a

[1] If we wanted to model the data of all participants, we would need logistic mixed-effects models, which are well outside the scope of these lecture notes.

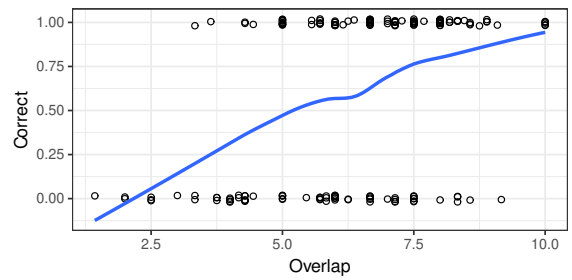


Figure 19.3: There seems to be a monotonous relationship between the degree of overlap and whether the participant correctly translated the word. Note, though, that scatterplot smoother extends beyond the $[0, 1]$ interval on the y axis.

correct translation for words with this little overlap; or about a probability of 0.035.

```
exp(-3.31)

[1] 0.0365162

plogis(-3.31)

[1] 0.0352297
```

Note that there are no words with 0 overlap in this dataset. If we wanted to, we could centre the overlap variable so that the intercept estimate becomes more informative.

- The odds ratio of a correct translation for words with an orthographic overlap of 1 vs. for words with an orthographic overlap of 0 is about 1.8. That is, the odds of correctly translating a word with an orthographic overlap of 1 are estimated to be about 1.8 as large as those of correctly translating a word with an orthographic overlap of 0.

```
exp(0.59)

[1] 1.80399
```

- To obtain the estimated probability of correctly translating a word given its overlap, first compute the relevant log-odds and then convert them to the probability scale. For instance, the estimated probability of correctly translating a word with an orthographic overlap of 7.23 is about 73%.

```
plogis(-3.313 + 0.594*7.23)

[1] 0.72743
```

Let's draw a plot that shows how, according to the model, the success probability varies with the degree of orthographic overlap (Figure 19.4). Note that the plotted curve isn't

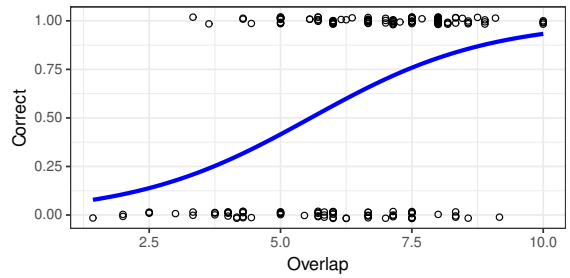


Figure 19.4: The same data, but this time the trend line is based on our logistic model.

straight. This is because we modelled the data linearly in log-odds space; by converting the result back to probability space via the logistic function, the trend line becomes nonlinear. That is, we could not *not* have found a nonlinear trend line. A full-fledged tutorial on drawing trend lines based on models is available from `https://janhove.github.io/visualise_uncertainty/`.

```
# quick function for computing the probabilities
model_to_prob <- function(x, model) {
  plogis(coef(model)[[1]] + coef(model)[[2]]*x)
}

ggplot(data = d,
       aes(x = Overlap,
           y = Correct)) +
  geom_point(shape = 1,
             position = position_jitter(width = 0, height = 0.02)) +
  stat_function(fun = model_to_prob, args = list(model = translation.glm),
                colour = "blue", linewidth = 1)
```

For further guidance on visualising logistic regression models, see `https://janhove.github.io/visualise_uncertainty/`.

19.3 Further reading

We've only scratched the surface. While we've covered the logit and the identity link function, other link functions exist (e.g., probit, complementary-log-log). We haven't mentioned overdispersion at all, either. Further, some useful extensions of logistic (binary) regression include multinomial regression and different kinds of ordinal regression. Agresti (2002) is the go-to reference for categorical data analysis.

Chapter 20

*Recommendations for self-study and tips

20.1 Books

Instead of repeating the reading recommendations from the last 300 pages, I'll limit myself here to three book suggestions:

- Bodo Winter's *Statistics for linguists: An introduction using R* aligns well in philosophy with this script. Winter (2019) also discusses the generalised linear model and mixed-effects models. These are useful, for example, when analysing data from a study in which different participants respond to multiple stimuli.
- *Regression and other stories* by Andrew Gelman, Jennifer Hill and Aki Vehtari offers a comprehensive, accessible introduction to working with regression models. It does not shy away from more realistic but correspondingly more complex examples. The book is freely available at `https://avehtari.github.io/ROS-Examples/` (Gelman et al., 2020/2023).
- Richard McElreath's *Statistical rethinking: A Bayesian course with examples in R and Stan* (2nd edition) is an excellent introduction to so-called Bayesian statistics. In my view, it is particularly recommended if you already have some experience working with quantitative data (McElreath, 2020).

20.2 Final tips

- Your first priority should be to formulate a clear research question or objective. Avoid technical terminology (e.g., 'significance', 'regression coefficient', 'interaction', 'correlation') that already presupposes a particular analytical method. Then develop a research design and choose an analytical method suited to answering your question. Ideally, you should already know in advance how you will answer the research question depending on the output of the analysis. In particular, you should know how the relationship or phenomenon of interest would be reflected in the data and in the analysis output, and how similar patterns might otherwise arise (e.g., influence of third variables, alternative theoretical explanations, etc.). See

https://www.the100.ci/2024/08/27/lets-do-statistics-the-other-way-around/ and the literature cited there.

- Schedule regular time for self-study. Even if it is only two hours every other Friday afternoon, this will help you more than trying to learn statistics intensively in the last months of your doctorate. With the book recommendations above and the reading lists in the previous chapters, you will not run out of resources.
- Statistical procedures often have misleading names. So-called causal models, for example, do not allow you to demonstrate causality; causality is an assumption of the model, not its result. Similarly, 'confirmatory' factor analysis is probably used more often for exploratory than for confirmatory purposes in practice. And as we have already discussed, a 'significant' finding can be entirely tautological or irrelevant. Do not let the names of methods guide your choice of tools too strongly.
- Do *not* assume that published analyses necessarily make sense. **(!!!)** In many articles, they do not. Moreover, many analyses are poorly reported. Too much irrelevant and distracting information is often included in the text, along with too many significance tests; see Vanhove (2021b). Consequently, you should not carry out or report analyses that are common in the literature if you judge them to add no value in the context of your study. Just because an analysis produces numbers does not mean all those numbers are relevant to you or your readership.
- Be sparing with significance tests. Ideally, in my view, the statistical hypotheses you test should be so closely tailored to the scientific hypotheses underlying the work that at most one significance test is conducted and reported per hypothesis. These tests should ideally be preregistered; see https://help.osf.io/article/330-welcome-to-registrations and https://datacolada.org/64.
- Use plenty of plots, and not only tables, both during analysis and in your reports. Aim to show the variability in your data (scatterplots, boxplots) as well as the uncertainty in your estimates (e.g., confidence intervals).
- Make your data and R code available online whenever possible. On the one hand, this saves you from having to include details in the report that you consider irrelevant: interested readers can consult the appendix themselves. On the other hand, it is entirely possible that in the coming years new statistical methods will be developed that allow your data to be analysed more effectively. If your data are freely accessible, they can be reanalysed, helping to ensure your study does not become obsolete. A practical platform for sharing data and scripts is https://osf.io/.
- Whether an analysis is justifiable depends primarily on whether it helps you answer your questions or learn something about the data. Do not follow a flowchart mechanically. Instead, think carefully about what you actually want to find out and whether the numbers in the output are relevant to that aim.

Appendices

Appendix A

Common error messages in R

Tip A.1. Solve problems step by step! If your entire screen is full of red error messages, start with the very first one. Often, the other errors are just consequences of the first. ♢

Tip A.2 (Debugging)**.** When asking someone for help, you should create a reproducible example that illustrates the problem. These examples should be as *short* as possible; see `https://stackoverflow.com/q/5963269/1331521`. Often, trying to reduce a problem to a minimal reproducible example helps you find the solution yourself. Another useful technique for debugging code is **rubber duck debugging**; see `https://rubberduckdebugging.com/`. It often also helps to take a break or go for a walk, so you can return the next day with a fresh perspective. ♢

object 'x' not found

This occurs when you try to access an object that is not in memory. Typographical errors are a common cause. It may also happen that you have not read in the object, or that it has a different name than you expect.

Example:

```
x <- c(1, 5, 4)
mean(X)
```

```
[1] 15
```

In the first line, an object named `x` is created, but the second line attempts to compute the mean of an object named `X`. R is case-sensitive!

If you are unsure which objects are currently in memory, you can use:

```
ls()
```

The same information is available in RStudio in the top-right window.

could not find function 'x'

You are trying to use a function that does not exist or is not accessible. First, check that you typed the function name correctly. If the function comes from an external package, make sure that package is loaded.

Example:

```
r.test(n = 50, r12 = 0.4)

Error in r.test(n = 50, r12 = 0.4):  could not find function "r.test"
```

The `r.test()` function is part of the `psych` package. Either load the package first with `library(psych)` or prefix the function with `psych::`.

Error in library(x) : there is no package called 'x'

You are trying to load a package that has not yet been installed.

Example:

```
library(ggjoy)

Error in library(ggjoy):  there is no package called 'ggjoy'
```

Solution: install the package first:

```
install.packages("ggjoy")
```

unexpected symbol

A common cause of this error is a missing comma.

Example:

```
> library(ggplot2)
> ggplot(data = iris
+        aes(x = Sepal.Width,
Error: unexpected symbol in:
"ggplot(data = iris
       aes"
```

A comma is missing after `data = iris`.

Example:

```
> ggplot(data = iris)
>        aes(x = Sepal.Width,
+            y = Sepal.Length)) +
Error: unexpected ')' in:
"       aes(x = Sepal.Width,
```

```
           y = Sepal.Length))"
```

The bracket at the end of the first line should be a comma.

Example:

```
> ggplot(data = iris,
+        aes(x = Sepal.Width,
+            y = Sepal.Length))) +
Error: unexpected ')' in:
"       aes(x = Sepal.Width,
           y = Sepal.Length)))"
```

There is one too many closing brackets on the third line.

It doesn't work, and there is no error message!

This usually happens when a bracket or similar character is missing.

Example: this command produces no error message but also no plot:

```
> ggplot(data = iris,
+        aes(x = Sepal.Width,
+            y = Sepal.Length) +
+   geom_point()
+
```

As you can see, the next line begins with a `+`, meaning the command is incomplete. The problem is a missing closing bracket on line 3: the last bracket closes `aes()`, but the `ggplot()` call is not yet closed. Correct version:

```
> ggplot(data = iris,
+        aes(x = Sepal.Width,
+            y = Sepal.Length)) +
+   geom_point()
>
```

Notice that the next line starts with `>`, indicating that the command has been executed.

Another example: this command also produces no error but no plot:

```
> ggplot(data = iris,
+        aes(x = Sepal.Width,
+            y = Sepal.Length))
> geom_point()
geom_point: na.rm = FALSE
stat_identity: na.rm = FALSE
position_identity
```

Problem: the plus sign after line 3 is missing.

The result of a calculation is 'NA'

You want to calculate a mean or similar, but the vector contains `NA` values (*not available*). R treats `NA` as an unknown number. If a vector contains an unknown value, its mean or standard deviation (etc.) will also be unknown (`NA`).

```
x <- c(5, 4, 18, NA, 3)
mean(x)

[1] NA
```

If you wish, you can calculate the mean ignoring unknown values:

```
mean(x, na.rm = TRUE)

[1] 7.5
```

For more information on handling missing data, see Graham (2009).

Appendix B

Software versions

I last compiled this script using the following software versions.

```
suppressWarnings(devtools::session_info(pkgs = "attached"))
```

```
- Session info ------------------------------------------
 setting  value
 version  R version 4.5.1 (2025-06-13)
 os       Ubuntu 25.10
 system   x86_64, linux-gnu
 ui       X11
 language (EN)
 collate  en_US.UTF-8
 ctype    en_US.UTF-8
 tz       Europe/Zurich
 date     2025-10-28
 pandoc   NA
 quarto   1.8.25 @ /usr/local/bin/quarto

- Packages ----------------------------------------------
 package      * version date (UTC) lib source
 cannonball   * 2025-02 2025-10-27 [1] Github (janhove/cannonball@627a15e)
 dagitty      * 0.3-4   2023-12-07 [1] CRAN (R 4.5.1)
 dplyr        * 1.1.4   2023-11-17 [1] CRAN (R 4.5.1)
 forcats      * 1.0.1   2025-09-25 [1] CRAN (R 4.5.1)
 ggplot2      * 4.0.0   2025-09-11 [1] CRAN (R 4.5.1)
 here         * 1.0.2   2025-09-15 [1] CRAN (R 4.5.1)
 knitr        * 1.50    2025-03-16 [1] CRAN (R 4.5.1)
 lubridate    * 1.9.4   2024-12-08 [1] CRAN (R 4.5.1)
 patchwork    * 1.3.2   2025-08-25 [1] CRAN (R 4.5.1)
 purrr        * 1.1.0   2025-07-10 [1] CRAN (R 4.5.1)
 RColorBrewer * 1.1-3   2022-04-03 [1] CRAN (R 4.5.1)
 readr        * 2.1.5   2024-01-10 [1] CRAN (R 4.5.1)
 readxl       * 1.4.5   2025-03-07 [1] CRAN (R 4.5.1)
```

```
 stringr      * 1.5.2   2025-09-08 [1] CRAN (R 4.5.1)
 tibble       * 3.3.0   2025-06-08 [1] CRAN (R 4.5.1)
 tidyr        * 1.3.1   2024-01-24 [1] CRAN (R 4.5.1)
 tidyverse    * 2.0.0   2023-02-22 [1] CRAN (R 4.5.1)

 [1] /home/jan/R/x86_64-pc-linux-gnu-library/4.5
 [2] /usr/local/lib/R/site-library
 [3] /usr/lib/R/site-library
 [4] /usr/lib/R/library
 * -- Packages attached to the search path.

------------------------------------------------------------
```

Bibliography

Abelson, Robert P. 1995. *Statistics as principled argument*. New York, NY: Psychology Press.

Agresti, Alan. 2002. *Categorical data analysis*. Hoboken, NJ: Wiley 2nd edn.

Albers, Casper J., Henk A. L. Kiers & Don van Ravenzwaaij. 2018. Credible confidence: A pragmatic view on the frequentist vs Bayesian debate. *Collabra: Psychology* 4(1). 31. doi:10.1525/collabra.149.

Baayen, R. Harald. 2008. *Analyzing linguistic data: A practical introduction to statistics using R*. Cambridge: Cambridge University Press.

Baayen, R. Harald & Maja Linke. 2020. Generalized additive mixed models. In Magali Paquot & Stefan Th. Gries (eds.), *A practical handbook of corpus linguistics*, 563–591. Cham, Switzerland: Springer Nature. doi:10.1107/978-3-030-46216-1_23.

Baguley, Thom. 2009. Standardized or simple effect size: What should be reported? *British Journal of Psychology* 100(3). 603–617. doi:10.1348/000712608X377117.

Bender, Ralf & Stefan Lange. 2001. Adjusting for multiple testing: when and how? *Journal of Clinical Epidemiology* 54(4). 343–349. doi:10.1016/S0895-4356(00)00314-0.

Berthele, Raphael. 2012. The influence of code-mixing and speaker information on perception and assessment of foreign language proficiency: An experimental study. *International Journal of Bilingualism* 16(4). 453–466. doi:10.1177/1367006911429514.

Berthele, Raphael. 2019. Policy recommendations for language learning: Linguists' contributions between scholarly debats and pseudoscience. *Journal of the European Second Language Association* 3(1). 1–11. doi:10.22599/jesla.50.

Broman, Karl W. & Kara H. Woo. 2017. Data organization in spreadsheets. *The American Statistician* doi:10.1080/00031305.2017.1375989.

Caruso, Eugene M., Kathleen D Vohs, Brittani Baxter & Adam Waytz. 2013. Mere exposure to money increases endorsement of free-market systems and social inequality. *Journal of Experimental Psychology: General* 142(2). 301–306. doi:10.1037/a0029288.

Chambers, Chris. 2017. *The seven deadly sins of psychology: A manifesto for reforming the culture of scientific practice*. Princeton, NJ: Princeton University Press.

Christenfeld, Nicholas J. S., Richard P. Sloan, Douglas Carroll & Sander Greenland. 2004. Risk factors, confounding, and the illusion of statistical control. *Psychosomatic Medicine* 66. 868–875. doi:10.1097/01.psy.0000140008.70959.41.

Clopper, C. J. & Egon Sharpe Pearson. 1934. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika* 26(4). 404–413. doi:10.2307/2331986.

Cohen, Jacob. 1977. *Statistical power analysis for the behavioral sciences*. New York, NY: Academic Press revised edition edn.

Cohen, Jacob. 1992. A power primer. *Psychological Bulletin* 112(1). 155–159. doi:10.1037/0033-2909.112.1.155.

de Bruin, Angela, Barbara Treccani & Sergio Della Sala. 2015. Cognitive advantage in bilingualism: An example of publication bias? *Psychological Science* 26(1). 99–107. doi:10.1177/0956797614557866.

de Groot, Adrianus Dingeman. 2014. The meaning of 'significance' for different types of research. *Acta Psychologica* 148. 188–194. doi:10.1016/j.actpsy.2014.02.001. Translated and annotated by Eric-Jan Wagenmakers, Denny Borsboom, Josine Verhagen, Rogier Kievit, Marjan Bakker, Angelique Cramer, Dora Matzke, Don Mellenbergh, and Han L. J. van der Maas.

DeKeyser, Robert, Iris Alfi-Shabtay & Dorit Ravid. 2010. Cross-linguistic evidence for the nature of age effects in second language acquisition. *Applied Psycholinguistics* 31. 413–438. doi:10.1017/S0142716410000056.

Desgrippes, Magalie, Amelia Lambelet & Jan Vanhove. 2017. The development of argumentative and narrative writing skills in Portuguese heritage speakers in Switzerland (HELASCOT project). In Raphael Berthele & Amelia Lambelet (eds.), *Heritage and school language literacy development in migrant children: Interdependence or independence?*, 83–96. Bristol: Multilingual Matters. doi:10.21832/9781783099054-006.

DiCiccio, Thomas J. & Bradley Efron. 1996. Bootstrap confidence intervals. *Statistical Science* 11(3). 189–212. doi:10.1214/ss/1032280214.

Dümbgen, Lutz. 2016. *Einführung in die Statistik*. Basel: Birkhäuser. doi:10.1007/978-3-0348-0004-4.

Efron, Bradley. 1979. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics* 7(1). 1–26. doi:10.1214/aos/1176344552.

Efron, Bradley & Robert J. Tibshirani. 1993. *An introduction to the bootstrap*. Boca Raton, FL: Chapman & Hall/CRC.

Ehrenberg, Andrew S. C. 1982. *A primer in data reduction: An introductory statistics textbook*. Chichester: Wiley.

Eriksen, Barbara A. & Charles W. Eriksen. 1974. Effects of noise letters upon the identification of a target letter in a nonsearch task. *Perception & Psychophysics* 16. 143–149. doi:10.3758/BF03203267.

Everitt, Brian & Torsten Hothorn. 2011. *An introduction to applied multivariate analysis with R*. New York: Springer.

Fagerland, Morten W., Stian Lydersen & Petter Laake. 2017. *Statistical analysis of contingency tables*. Boca Raton, FL: Chapman and Hall/CRC. doi:10.1201/9781315374116.

Fisher, Ronald Aylmer. 1936. "The coefficient of racial likeness" and the future of craniometry. *Journal of the Royal Anthropological Institute of Great Britain and Ireland* 66. 57–63. doi:10.2307/2844116.

Gelman, Andrew & Jennifer Hill. 2007. *Data analysis using regression and multilevel/hierarchical models*. New York, NY: Cambridge University Press.

Gelman, Andrew, Jennifer Hill & Aki Vehtari. 2020/2023. *Regression and other stories*. Cambridge University Press. `https://avehtari.github.io/ROS-Examples/`.

Gelman, Andrew & Eric Loken. 2013. The garden of forking paths: Why multiple comparisons can be a problem, even when there is no 'fishing expedition' or 'p-hacking' and the research hypothesis was posited ahead of time. `http://www.stat.columbia.edu/~gelman/research/unpublished/p_hacking.pdf`.

Gelman, Andrew & Hal Stern. 2006. The difference between "significant" and "not significant" is not itself statistically significant. *The American Statistician* 60(4). 328–331. doi:10.1198/000313006X152649.

Goodman, Steven. 2008. A dirty dozen: Twelve *p*-value misconceptions. *Seminars in Hematology* 45. 135–140. doi:10.1053/j.seminhematol.2008.04.003.

Graham, John W. 2009. Missing data analyis: Making it work in the real world. *Annual Review of Psychology* 60. 549–576. doi:10.1146/annurev.psych.58.110405.085530.

Greenland, Sander, Stephen J. Senn, Kenneth J. Rothman, John B. Carlin, Charles Poole, Steven N. Goodman & Douglas G. Altman. 2016. Statistical tests, *p* values, confidence intervals, and power: A guide to misinterpretations. *European Journal of Epidemiology* 31. 337–350. doi:10.1007/s10654-016-0149-3.

Healy, Kieran. 2019. *Data visualization: A practical introduction*. Princeton, NJ: Princeton University Press. Also freely available online from https://socviz.co/.

Hellevik, Ottar. 2009. Linear versus logistic regression when the dependent variable is a dichotomy. *Quality & Quantity* 43. 59–74. doi:10.1007/s11135-007-9077-3.

Hesterberg, Tim C. 2014. What teachers should know about the bootstrap: Resampling in the undergraduate statistics curriculum. arXiv. doi:10.48550/arXiv.14115279.

Hesterberg, Tim C. 2015. What teachers should know about the bootstrap: Resampling in the undergraduate statistics curriculum. *The American Statistician* 69(4). 371–386. doi:10.1080/00031305.2015.1089789.

Hicks, Nina Selina. 2021. Exploring systematic orthographic crosslinguistic similarities to enhance foreign language vocabulary learning. *Language Teaching Research* doi:10.1177/13621688211047353.

Hoekstra, Rink, Richard D. Morey, Jeffrey N. Rouder & Eric-Jan Wagenmakers. 2014. Robust misinterpretation of confidence intervals. *Psychonomic Bulletin & Review* 21(5). 1157–1164. doi:10.3758/s13423-013-0572-3.

Huang, Francis L. 2019. Alternatives to logistic regression models in experimental studies. *Journal of Experimental Education* doi:10.1080/00220973.2019.1699769.

Huff, Darrell. 1954. *How to lie with statistics*. New York: Norton.

Huitema, Bradley E. 2011. *The analysis of covariance and alternatives: Statistical methods for experiments, quasi-experiments, and single-case studies*. Hoboken, NJ: Wiley.

Imai, Kosuke, Gary King & Elizabeth A. Stuart. 2008. Misunderstandings between experimentalists and observationalists about causal inference. *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 171. 481–502. doi:10.1111/j.1467-985X.2007.00527.x.

Jaeger, T. Florian. 2008. Categorical data analysis: Away from ANOVAs (transformation or not) and towards logit mixed models. *Journal of Memory and Language* 59(4). 434–446. doi:10.1016/j.jml.2007.11.007.

Kelley, Ken, Scott E. Maxwell & Joseph R. Rausch. 2003. Obtaining power or obtaining precision. *Evaluation & the Health Professions* 26(3). 258–287. doi:10.1177/0163278703255242.

Kerr, Norbert L. 1998. HARKing: Hypothesizing after the results are known. *Personality and Social Psychology Review* 2(3). 196–217. doi:10.1207/s15327957pspr0203_4.

Keysar, Boas, Sayuri L. Hayakawa & Sun Gyu An. 2012. The foreign-language effect: Thinking in a foreign tongue reduces decision biases. *Psychological Science* 23(6). 661–668. doi:10.1177/0956797611432178.

Klein, Richard A., Kate A. Ratliff, Michelangelo Vianello, Reginald B. Adams Jr., Štěpán Bahník, Michael J. Bernstein, Konrad Bocian et al. 2014. Investigating variation in replicability: A "many labs" replication project. *Social Psychology* 45(3). 142–152. doi:10.1027/1864-9335/a000178.

Kuhn, Max & Kjell Johnson. 2013. *Applied predictive modeling*. New York: Springer. doi:10.1007/978-1-4614-6849-3.

Kvålseth, Tarald O. 1985. Cautionary note about R^2. *The American Statistician* 4(1). doi:10.2307/2683704.

Lakens, Daniël. 2014. Performing high-powered studies efficiently with sequential analyses. *European Journal of Social Psychology* 44. 701–710. doi:10.1002/ejsp.2023.

McElreath, Richard. 2020. *Statistical rethinking: A Bayesian course with examples in R and Stan*. Boca Raton, FL: CRC Press 2nd edn.

McGraw, Kenneth O. & S. P. Wong. 1992. A common language effect size statistic. *Psychological Bulletin* 111. 361–365.

Mook, Douglas G. 1983. In defense of external invalidity. *American Psychologist* 38. 379–387. doi:10.1037/0003-066X.38.4.379.

Morey, Richard D., Rink Hoekstra, Jeffrey N. Rouder, Michael D. Lee & Eric-Jan Wagenmakers. 2016. The fallacy of placing confidence in confidence intervals. *Psychonomic Bulletin & Review* 23(1). 103–123. doi:10.3758/s13423-015-0947-8.

Morrissey, Michael B. & Greame D. Ruxton. 2018. Multiple regression is not multiple regressions: The meaning of multiple regression and the non-problem of collinearity. *Philosophy, Theory, and Practice in Biology* 10(3). doi:10.3998/ptpbio.16039257.0010.003.

Nalborczyk, Ladislas, Paul-Christian Bürkner & Donald R. Williams. 2019. Pragmatism should not be a substitute for statistical literacy: A commentary on Albers, Kiers, and van Ravenzwaaij (2018). *Collabra: Psychology* 5(1). 13. doi:10.1525/collabra.197.

Nieuwenhuis, Sander, Birte U. Forstmann & Eric-Jan Wagenmakers. 2011. Erroneous analyses of interactions in neuroscience: A problem of significance. *Nature Neuroscience* 14. 1105–1107. doi:10.1038/nn.2886.

Noether, G. E. 1981. Why Kendall tau? *Teaching Statistics* 3(2). 41–43. doi:10.1111/j.1467-9639.1981.tb00422.x.

Oppenheimer, Daniel M., Tom Meyvis & Nicolas Davidenko. 2009. Instructional manipulation checks: Detecting satisficing to increase statistical power. *Journal of Experimental Social Psychology* 45. 867–872. doi:10.1016/j.jesp.2009.03.009.

Oppenheimer, Daniel M & Benoît Monin. 2009. The retrospective gambler's fallacy: Unlikely events, constructing the past, and multiple universes. *Judgment and Decision Making* 4(5). 326–334.

Pestana, Carlos, Amelia Lambelet & Jan Vanhove. 2017. Reading comprehension in Portuguese heritage speakers in Switzerland (HELASCOT project). In Raphael Berthele & Amelia Lambelet (eds.), *Heritage and school language literacy development in migrant children: Interdependence or independence?*, 58–82. Bristol: Multilingual Matters. doi:10.21832/9781783099054-005.

Peterson, David. 2016. The baby factory: Difficult research objects, disciplinary standards, and the production of statistical significance. *Socius* 2. 1–10. doi:10.1177/2378023115625071.

Poarch, Gregory J., Jan Vanhove & Raphael Berthele. 2019. The effect of bidialectalism on executive function. *International Journal of Bilingualism* 23(2). 612–628. doi:10.1177/1367006918763132.

Ritchie, Stuart. 2021. *Science fictions*. Penguin.

Rohrer, Julia M. 2018. Thinking clearly about correlations and causation: Graphical causal models for observational data. *Advances in Methods and Practices in Psychological Science* 1(1). 27–42. doi:10.1177/2515245917745629.

Ruxton, Graeme D. 2006. The unequal variance *t*-test is an underused alternative to Student's *t*-test and the Mann–Whitney *u* test. *Behavioral Ecology* 17. 688–690. doi:10.1093/beheco/ark016.

Ruxton, Graeme D. & Guy Beauchamp. 2008. Time for some a priori thinking about post hoc testing. *Behavioral Ecology* 19(3). 690–693. doi:10.1093/beheco/arn020.

Schad, Daniel J., Shravan Vasishth, Sven Hohenstein & Reinhold Kliegl. 2020. How to capitalize on a priori contrasts in linear (mixed) models: A tutorial. *Journal of Memory and Language* 110. doi:10.1016/j.jml.2019.104038.

Schmidt, Frank L. 1996. Statistical significance testing and cumulative knowledge in psychology: Implications for training of researchers. *Psychological Methods* 1. 115–129. doi:10.1037/1082-989X.1.2.115.

Schwertman, Neil C., A. J. Gilks & J. Cameron. 1990. A simple noncalculus proof that the median minimizes the sum of the absolute deviations. *The American Statistician* 44(1). 38–39. doi:10.1080/00031305.1990.10475690.

Senn, Stephen. 2011. Francis Galton and regression to the mean. *Significance* 8(3). 124–126. doi:10.1111/j.1740-9713.2011.00509.x.

Shmueli, Galit. 2010. To explain or to predict? *Statistical Science* 25(3). 289–310. doi:10.1214/10-STS330.

Simmons, Joseph P., Leif D. Nelson & Uri Simonsohn. 2011. False-positive psychology: Undisclosed flexibility in data collection and analysis allows presenting anything as significant. *Psychological Science* 22. 1359–1366. doi:10.1177/0956797611417632.

Simmons, Joseph P., Leif D. Nelson & Uri Simonsohn. 2018. False-positive citations. *Perspectives on Psychological Science* 13(2). 255–259. doi:10.1177/1745691617698146.

Simon, J. Richard. 1969. Reactions toward the source of stimulation. *Journal of Experimental Psychology* 81. 174–176. doi:10.1037/h0027448.

Simonsohn, Uri. 2013. Just post it: The lesson from two cases of fabricated data detected by statistics alone. *Psychological Science* 24(10). 1875–1888. doi:10.1177/0956797613480366.

Steegen, Sara, Francis Tuerlinckx, Andrew Gelman & Wolf Vanpaemel. 2016. Increasing transparency through a multiverse analysis. *Perspectives on Psychological Science* 11(5). 702–712. doi:10.1177/1745691616658637.

Sterling, Theodore D., W. L. Rosenbaum & J. J. Weinkam. 1995. Publication decision revisited: The effect of the outcome of statistical tests on the decision to publish and vice versa. *The American Statistician* 49. 108–112. `http://www.jstor.org/stable/2684823`.

Stocker, Ladina. 2017. The impact of foreign accent on credibility: An analysis of cognitive statement ratings in a Swiss context. *Journal of Psycholinguistic Research* 46(3). 617–628. doi:10.1007/s10936-016-9455-x.

Tversky, Amos & Daniel Kahneman. 1981. The framing of decisions and the psychology of choice. *Science* 211(4481). 453–458. doi:10.1126/science.7455683.

van den Broek, Gesa S. E., Atsuko Takashima, Eliane Segers & Ludo Verhoeven. 2018. Contextual richness and word learning: Context enhances comprehension but retrieval enhances retention. *Language Learning* 68(2). 546–585. doi:10.1111/lang.12285.

Vanhove, Jan. 2013. The critical period hypothesis in second language acquisition: A statistical critique and a reanalysis. *PLOS ONE* 8. e69172. doi:10.1371/journal.pone.0069172.

Vanhove, Jan. 2014. *Receptive multilingualism across the lifespan: Cognitive and linguistic factors in cognate guessing*: University of Fribourg dissertation. `http://doc.rero.ch/record/210293`.

Vanhove, Jan. 2015. Analyzing randomized controlled interventions: Three notes for applied linguists. *Studies in Second Language Learning and Teaching* 5. 135–152. doi:10.14746/ssllt.2015.5.1.7.

Vanhove, Jan. 2016. The early learning of interlingual correspondence rules in receptive multilingualism. *International Journal of Bilingualism* 20(5). 580–593. doi:10.1177/1367006915573338.

Vanhove, Jan. 2017a. The influence of standard and substandard Dutch on gender assignment in second language German. *Language Learning* 67(2). 431–460. doi:10.1111/lang.12230.

Vanhove, Jan. 2017b. Lexical richness of short French, German and Portugese texts written by children (technical report). `https://osf.io/vw4pc/`.

Vanhove, Jan. 2018. Checking the assumptions of your statistical method without getting paranoid. doi:10.31234/osf.io/zvawb.

Vanhove, Jan. 2019. Metalinguistic knowledge about the native language and language transfer in gender assignment. *Studies in Second Language Learning and Teaching* 9(2). 397–419. doi:10.14746/ssllt.2019.9.2.7.

Vanhove, Jan. 2020a. Capitalising on covariates in cluster-randomised experiments. *PsyArXiv Preprints* doi:10.31234/osf.io/ef4zc.

Vanhove, Jan. 2020b. When labeling L2 users as nativelike or not, consider classification errors. *Second Language Research* 36(4). 709–724. doi:10.1177/0267658319827055.

Vanhove, Jan. 2021a. Collinearity isn't a disease that needs curing. *Meta-Psychology* 5. doi:10.15626/MP.2021.2548.

Vanhove, Jan. 2021b. Towards simpler and more transparent quantitative research reports. *ITL - International Journal of Applied Linguistics* 172(1). 3–25. doi:10.1075/itl.20010.van.

Vanhove, Jan & Raphael Berthele. 2013. Factoren bij het herkennen van cognaten in onbekende talen: algemeen of taalspecifiek? *Taal & Tongval* 65. 171–210. doi:10.5117/TET2013.2.VANH.

Vanhove, Jan, Audrey Bonvin, Amelia Lambelet & Raphael Berthele. 2019. Predicting perceptions of the lexical richness of short French, German, and Portuguese texts using text-based indices. *Journal of Writing Research* 10(3). 499–525. doi:10.17239/jowr-2019.10.03.04.

Wagenmakers, Eric-Jan, Angelos-Miltiadis Krypotos, Amy H. Criss & Geoff Iverson. 2012a. On the interpretation of removable interactions: A survey of the field 33 years after Loftus. *Memory & Cognition* 40(2). 145–160. doi:10.3758/s13421-011-0158-0.

Wagenmakers, Eric-Jan, Ruud Wetzels, Denny Borsboom, Han L. J. van der Maas & Rogier A. Kievit. 2012b. An agenda for purely confirmatory research. *Perspectives on Psychological Science* 7(6). doi:10.1177/1745691612463078.

Weissgerber, Tracey L., Natasa M. Milic, Stacey J. Winham & Vesna D. Garovic. 2015. Beyond bar and line graphs: Time for a new data presentation paradigm. *PLOS Biology* 13(4). e1002128. doi:10.1371/journal.pbio.1002128.

Westfall, Jacob, David A. Kenny & Charles M. Judd. 2014. Statistical power and optimal design in experiments in which samples of participants respond to samples of stimuli. *Journal of Experimental Psychology: General* 143. 2020–2045. doi:10.1037/xge0000014.

Westfall, Jacob & Tal Yarkoni. 2016. Statistically controlling for confounding constructs is harder than you think. *PLOS ONE* 11(3). e0152719. doi:10.1371/journal.pone.0152719.

Wickham, Hadley, Mine Çetinkaya Rundel & Garrett Grolemund. 2023. *R for data science: Import, tidy, transform, visualize, and model data*. O'Reilly 2nd edn. `https://r4ds.hadley.nz/`.

Wieling, Martijn. 2018. Analyzing dynamic phonetic data using generalized additive mixed modeling: A tutorial focusing on articulatory differences between L1 and L2 speakers of English. *Journal of Phonetics* 70. 86–116.

Winter, Bodo. 2019. *Statistics for linguists: An introduction using R*. Routledge. doi:10.4324/9781315165547.