Foundations of Applied Statistics and R

Thomas E. Love, Ph.D.

2024-09-15

Table of contents

Preface

> **Warning**
>
> A full draft of this book will be available as soon as possible.
> This Preface, and Chapters 1-12 and the Appendix materials are in a sufficiently final form to justify reading them. I'm actively revising Chapters 13-23 and new versions will appear before I declare a complete draft to be available.

This Book

1. This book is broken down into multiple chapters. Use the table of contents on the left side of the screen to navigate between chapters, or use the right side to navigate within the current chapter.
2. You can also search the book, using an automated index.
3. Any of the code provided can be copied to the clipboard using the Copy icon at the top right of the code block.
4. If, for some awful reason, you want to print or save this document, we recommend waiting until mid-January of 2025, when a "final" version will be available. That final version will remain available until 2025-06-01.

What You'll Find Here

This book provides a series of examples using R to work through issues that are likely to come up in PQHS/CRSP/MPHP 431. The main purpose is to give 431 students a set of common materials on which to draw during the course. In class, we will sometimes:

- reiterate points made in this document,
- amplify what is here,
- simplify the presentation of things done here,
- use new examples to show some of the same techniques,
- refer to issues not mentioned in this document.

We assume that you will read the materials as you need them, just as you will attend classes and try to learn from them. We welcome feedback of all kinds on this document or anything else.

The 431 Course online

The **online home** for Dr. Love's 431 course in Fall 2024 is

https://thomaselove.github.io/431-2024

Go there for all information related to the course.

All of the code and text in this book is posted online as HTML, and it is also possible to download a PDF version of the document from the down arrow next to the title at the top left of this screen.

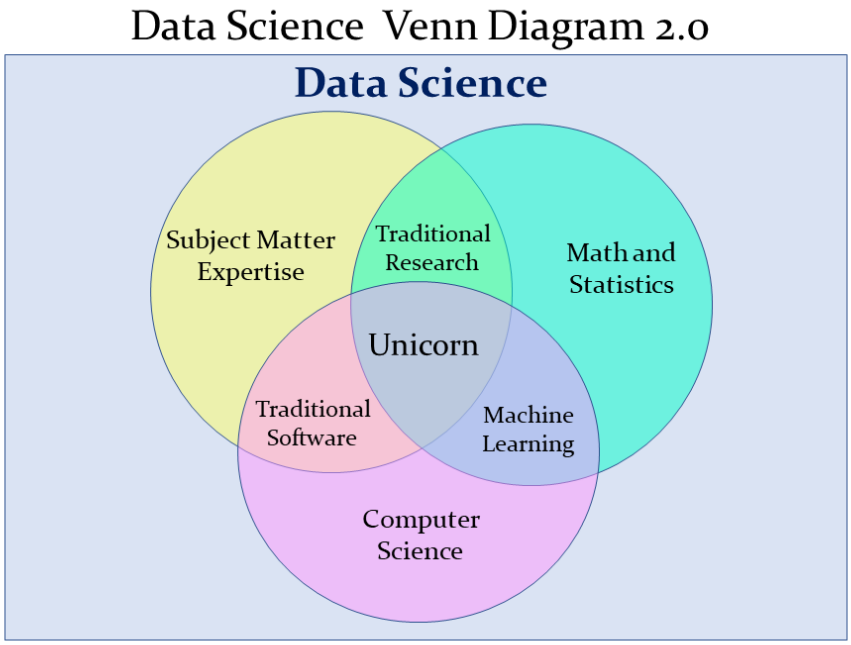
All data and R code related to this book are available to you through our course web site.

Part I

Getting Started

1 Introduction

The definition of **data science** can be a little slippery. One fairly modern view of data science is exemplified by Steven Geringer's 2014 Venn diagram.



- The field encompasses ideas from mathematics and statistics and from computer science, but with a heavy reliance on subject-matter knowledge. In our case, this includes clinical, health-related, medical or biological knowledge.
- As Gelman and Nolan (2017) suggest, the experience and intuition necessary for good statistical practice are hard to obtain, and teaching data science provides an excellent opportunity to reinforce statistical thinking skills across the full cycle of a data analysis project.
- The principal form in which computer science (coding/programming) play a role in this course is to provide a form of communication. You'll need to learn how to express your ideas not just orally and in writing, but also through your code.

Data Science is a **team** activity. Everyone working in data science brings some part of the necessary skill set, but no one person can cover all three areas alone for excellent projects.

> [The individual who is truly expert in all three key areas (mathematics/statistics, computer science and subject-matter knowledge) is] a mythical beast with magical powers who's rumored to exist but is never actually seen in the wild.
>
> http://www.kdnuggets.com/2016/10/battle-data-science-venn-diagrams.html

1.1 Data Science Project Cycle

A typical data science project can be modeled as follows, which comes from the introduction to the amazing book **R for Data Science**, which is a key text for this course (Wickham, Çetinkaya-Rundel, and Grolemund 2024).

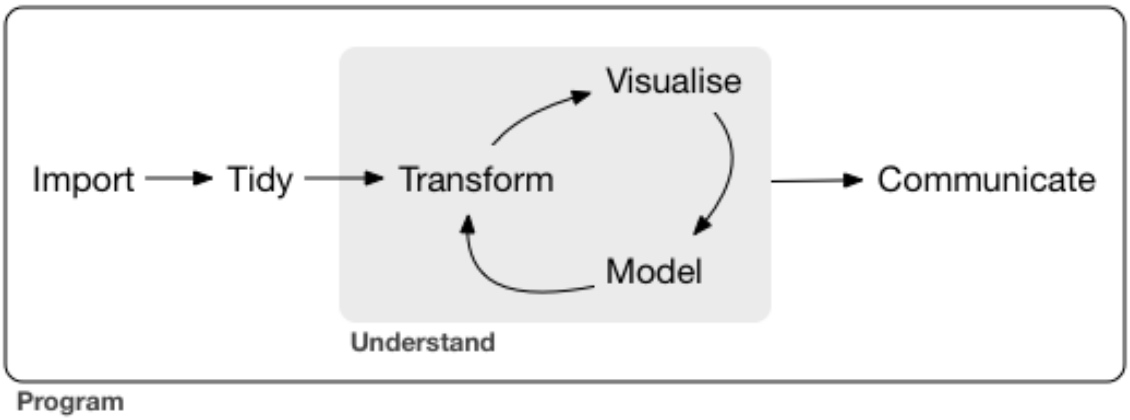


Figure 1.1: Source: R for Data Science: Introduction

This diagram is sometimes referred to as the Krebs Cycle of Data Science. For more on the steps of a data science project, we encourage you to read the Introduction of Wickham, Çetinkaya-Rundel, and Grolemund (2024).

As Gelman, Hill, and Vehtari (2021) suggest, statistical analysis itself cycles through a series of steps:

- Model building, starting with simple linear models of the form $y = a + bx +$ error, and expanding through additional predictors, interactions and transformations.
- Model fitting, which includes data manipulation, programming, and the use of algorithms to estimate regression coefficients and their uncertainties and to make probabilistic predictions.

- Understanding model fits, which involves graphics, more programming, and an active investigation of the (imperfect) connections between measurements, parameters, and the underlying objects of study.
- Criticism, which is not just about finding flaws and identifying questionable assumptions, but is also about considering directions for improvement of models. Or, if nothing else, limiting the claims that might be made by a naive reading of a fitted model.

The next step is to return to model building, possibly including new data.

1.2 Data Science and the 431 Course

We'll discuss each of these elements in the 431 course, focusing at the start on understanding our data through transformation, modeling and (especially in the early stages) visualization. In 431, we learn how to get things done.

- We get people working with R and R Studio and Quarto, even if they are completely new to coding. A gentle introduction is provided at Ismay and Kim (2024)
- We learn how to use the `tidyverse` (see http://www.tidyverse.org/), an array of tools in R (mostly developed by Hadley Wickham and his colleagues at R Studio) which share an underlying philosophy to make data science faster, easier, more reproducible and more fun. A critical text for understanding the tidyverse is Wickham, Çetinkaya-Rundel, and Grolemund (2024). Tidyverse tools facilitate:
  - **importing** data into R, which can be the source of intense pain for some things, but is really quite easy 95% of the time with the right tool.
  - **tidying** data, that is, storing it in a format that includes one row per observation and one column per variable. This is harder, and more important, than you might think.
  - **transforming** data, perhaps by identifying specific subgroups of interest, creating new variables based on existing ones, or calculating summaries.
  - **visualizing** data to generate actual knowledge and identify questions about the data - this is an area where R really shines, and we'll start with it in class.
  - **modeling** data, taking the approach that modeling is complementary to visualization, and allows us to answer questions that visualization helps us identify.
  - and last, but definitely not least, **communicating** results, models and visualizations to others, in a way that is reproducible and effective.
- We also use the `easystats` collection of R packages (Lüdecke et al. 2022) (see https://easystats.github.io/easystats/) to help us with modeling and reporting key elements of our statistical work. This framework builds on what the tidyverse does to help us in the final stages of preparing an analysis, as we'll see.

- Some programming/coding is an inevitable requirement to accomplish all of these aims. If you are leery of coding, you'll need to get past that, with the help of this course and our stellar teaching assistants. Getting started is always the most challenging part, but our experience is that most of the pain of developing these new skills evaporates by early October.

1.3 What The Course Is and Isn't

The 431 course is about **getting things done**. In developing this course, we adopt a modern approach that places data at the center of our work. Our goal is to teach you how to do truly reproducible research with modern tools. We want you to be able to collect and use data effectively to address questions of interest.

The curriculum includes more on several topics than you might expect from a standard graduate introduction to biostatistics.

- data gathering
- data wrangling
- exploratory data analysis and visualization
- multivariate modeling
- communication
- both Bayesian and frequentist approaches to regression models

It also nearly completely avoids formalism and is extremely applied - this is absolutely **not** a course in theoretical or mathematical statistics, and this book reflects that approach.

Our work does not require highly advanced mathematics. There's very little of the mathematical underpinnings here:

$$f(x) = \frac{e^{-(x-\mu)^2/(2\sigma^2)}}{\sigma\sqrt{2\pi}}$$

Instead, this book (and the course) focus on how we get R to do the things we want to do, and how we interpret the results of our work. Prior programming knowledge is desirable in some ways, but by no means required.

In 431 (and its follow-up, 432), we hope to provide you with the tools you need to work effectively with regression models. This includes understanding, using and assessing the fit of linear models (in 431) and generalized linear models (in 432) incorporating a wide range of ideas. We want you to understand what regression can and cannot do, and we want you to have a deeper sense of how to get started (and when to stop) with a regression analysis. Note that some of our motivation for the revisions found in this book as compared to prior versions of 431 comes from Gelman, Hill, and Vehtari (2021).

I should also mention that this isn't a book about *big* data, by which I mean data that can't fit easily into a single analytic file residing on a laptop computer. We work with small and medium-sized data files here, and will do so throughout 431 and its follow-up course, 432.

2 Graphical Summaries

2.1 R setup for this chapter

> **Note**
>
> This section loads all needed R packages for this chapter. Appendix A lists all R packages used in this book, and also provides R session information.

```
library(ggdist)
library(ggpubr)
library(glue)
library(janitor)
library(knitr)
library(naniar)
library(palmerpenguins)
library(patchwork)

library(easystats)
library(tidyverse)
```

2.2 Data from an R package: The Palmer Penguins

> **Note**
>
> Appendix C provides further guidance on pulling data from other systems into R, while Appendix D gives more information (including download links) for all data sets used in this book.

The data in the `palmerpenguins` package includes several measurements of interest for adult foraging penguins observed on islands in the Palmer Archipelago near Palmer Station, Antarctica. Dr. Kristen Gorman and the Palmer Station Long Term Ecological Research (LTER)

Program collected the data and made them available[1]. The data describe three species of penguins, called Adelie, Chinstrap and Gentoo. See Horst, Hill, and Gorman (2020) for more details on the `palmerpenguins` package and data.

Once we've used the `library()` function to load in the `palmerpenguins` package, a data set will become available called `penguins`. To view it, we can simply type in the name.

```
penguins
```

```
# A tibble: 344 x 8
   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
   <fct>   <fct>              <dbl>         <dbl>             <int>       <int>
 1 Adelie  Torgersen           39.1          18.7               181        3750
 2 Adelie  Torgersen           39.5          17.4               186        3800
 3 Adelie  Torgersen           40.3          18                 195        3250
 4 Adelie  Torgersen           NA            NA                  NA          NA
 5 Adelie  Torgersen           36.7          19.3               193        3450
 6 Adelie  Torgersen           39.3          20.6               190        3650
 7 Adelie  Torgersen           38.9          17.8               181        3625
 8 Adelie  Torgersen           39.2          19.6               195        4675
 9 Adelie  Torgersen           34.1          18.1               193        3475
10 Adelie  Torgersen           42            20.2               190        4250
# i 334 more rows
# i 2 more variables: sex <fct>, year <int>
```

For example, the first row describes a penguin of the Adelie species, from the island of Torgeson, with a bill length of 39.1 mm, and so forth.

The `penguins` data are represented in R using a **tibble**, a special type of R data frame that prints only the first ten rows of our data, and has some other attractive features, as compared to regular R data frames.

Within the `penguins` tibble, we see 344 rows, each representing a different penguins and 8 columns, each representing a different variable which helps characterize the penguins. In all, the data contain eight variables (columns) which describe each penguin's:

- species (Adelie, Chinstrap or Gentoo)
- island (Biscoe, Dream or Torgerson)
- bill length, in mm
- bill depth, in mm
- flipper length, in mm

[1]Two fun facts: (1) Male Gentoo and Adelie penguins "propose" to females by giving them a pebble. (2) The Adelie penguin was named for his wife by Jules Dumont d'Urville, who also rediscovered the Venus de Milo.

- body mass, in g
- sex (either female or male, although we'll see that some of the rows are missing this information)
- year of observation (either 2007, 2008 or 2009)

There are also some rows which contain indicators of missing data - these are the `NA` values we see, for example, in the data for the fourth penguin. It will turn out that missingness in the `penguins` tibble is confined to the variables describing the size and sex of the penguins.

2.3 What is in our tibble?

How many rows and columns are there in the `penguins` tibble?

```
dim(penguins)
```

```
[1] 344   8
```

```
nrow(penguins)
```

```
[1] 344
```

```
ncol(penguins)
```

```
[1] 8
```

There are many ways to get a quick understanding of what's contained in the `penguins` tibble, including `glimpse()` (shown below) as well as `str()` and `data_peek()`. Each of these provides similar information on the types of variables (factors, integers, numbers) and some of their values in the tibble.

```
str(penguins)
```

```
tibble [344 x 8] (S3: tbl_df/tbl/data.frame)
 $ species          : Factor w/ 3 levels "Adelie","Chinstrap",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ island           : Factor w/ 3 levels "Biscoe","Dream",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ bill_length_mm   : num [1:344] 39.1 39.5 40.3 NA 36.7 39.3 38.9 39.2 34.1 42 ...
 $ bill_depth_mm    : num [1:344] 18.7 17.4 18 NA 19.3 20.6 17.8 19.6 18.1 20.2 ...
 $ flipper_length_mm: int [1:344] 181 186 195 NA 193 190 181 195 193 190 ...
 $ body_mass_g      : int [1:344] 3750 3800 3250 NA 3450 3650 3625 4675 3475 4250 ...
 $ sex              : Factor w/ 2 levels "female","male": 2 1 1 NA 1 2 1 2 NA NA ...
 $ year             : int [1:344] 2007 2007 2007 2007 2007 2007 2007 2007 2007 2007 ...
```

The **naniar** package provides us with several ways to identify and count missing values in the data. For instance, the `penguins` tibble contains 19 missing values.

```
n_miss(penguins)
```

```
[1] 19
```

Which variables contain missing values?

```
miss_var_summary(penguins)
```

```
# A tibble: 8 x 3
  variable          n_miss pct_miss
  <chr>              <int>    <num>
1 sex                   11    3.20
2 bill_length_mm         2    0.581
3 bill_depth_mm          2    0.581
4 flipper_length_mm      2    0.581
5 body_mass_g            2    0.581
6 species                0    0
7 island                 0    0
8 year                   0    0
```

We can also break down missing values within each observation (or case) in the data.

```
miss_case_table(penguins)
```

```
# A tibble: 3 x 3
  n_miss_in_case n_cases pct_cases
           <int>   <int>     <dbl>
1              0     333    96.8
2              1       9     2.62
3              5       2     0.581
```

Here, for instance, we see 2 rows (penguins) each missing five values, 9 missing just 1, and the remaining 333 are complete.

2.4 Visualizing A Quantity

> There is no single best way to display a data set. Looking at data in unexpected ways can lead to discovery. – Gelman, Hill, and Vehtari (2021)

Let's look at some fundamental ways of plotting data describing the distribution of one of our variables. We'll start with the flipper length, measured in mm. We will make use of the `ggplot2` package, which is part of the `tidyverse` meta-package, to construct our plots.

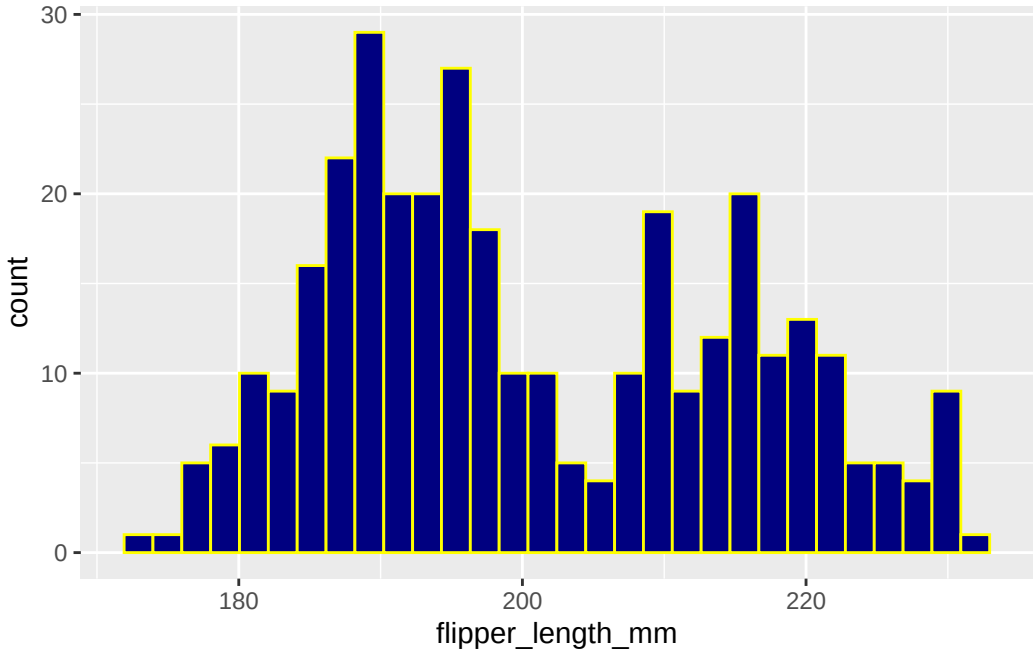
2.4.1 Histogram

Here, for example, is a histogram of the flipper lengths for our penguins. Note that we first specify the tibble (data frame = penguins, here), and then the x axis (flipper lengths), before creating the histogram with navy blue fill in each bar, and yellow lines around the bar.

```
ggplot(data = penguins, aes(x = flipper_length_mm)) +
  geom_histogram(fill = "navy", col = "yellow")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
Warning: Removed 2 rows containing non-finite outside the scale range
(`stat_bin()`).
```

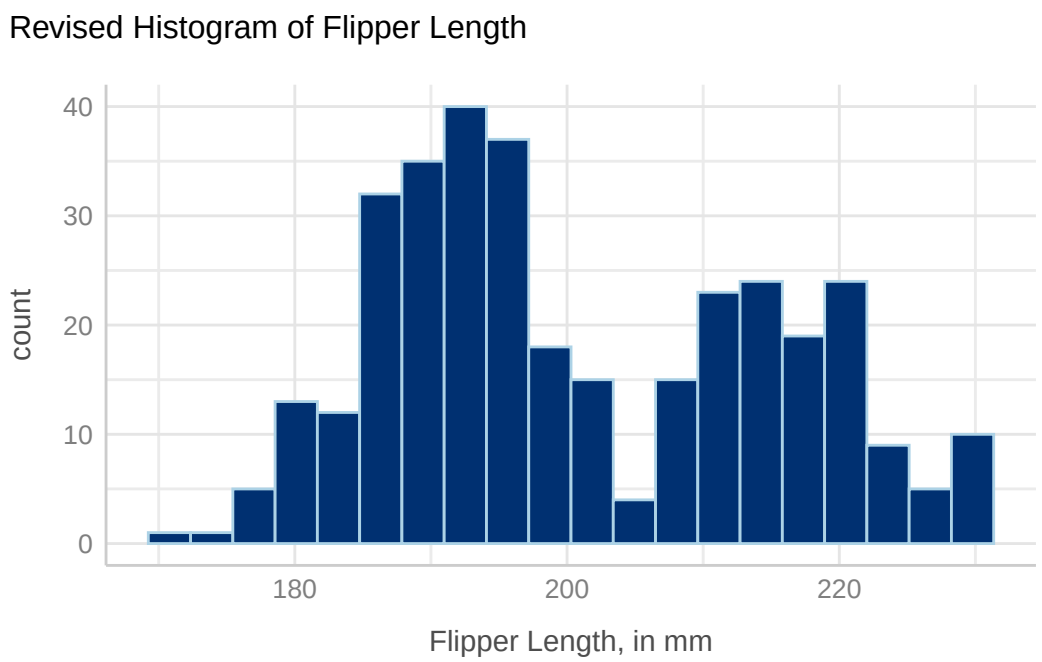


This code produces both a message (which we might ignore) and a warning (which is more worrisome.)

- The warning alerts us that two rows have not been plotted, and this is because we have missing values for flipper length for two of our penguins. So, perhaps we should filter our data down to only those penguins with complete information on flipper length.
- The message suggests that we ought to specify the number of bins (bars) in which we divide the flipper lengths. We can do so by either specifying a bin width or the number of bins.

We address each of these concerns (and adjust a few other things) in the revised histogram shown below.

```
penguins |>
  filter(complete.cases(flipper_length_mm)) |>
  ggplot(aes(x = flipper_length_mm)) +
  geom_histogram(bins = 20, fill = "#003071", color = "#ACD2E6") +
  labs(
    x = "Flipper Length, in mm",
    title = "Revised Histogram of Flipper Length"
  ) +
  theme_lucid()
```



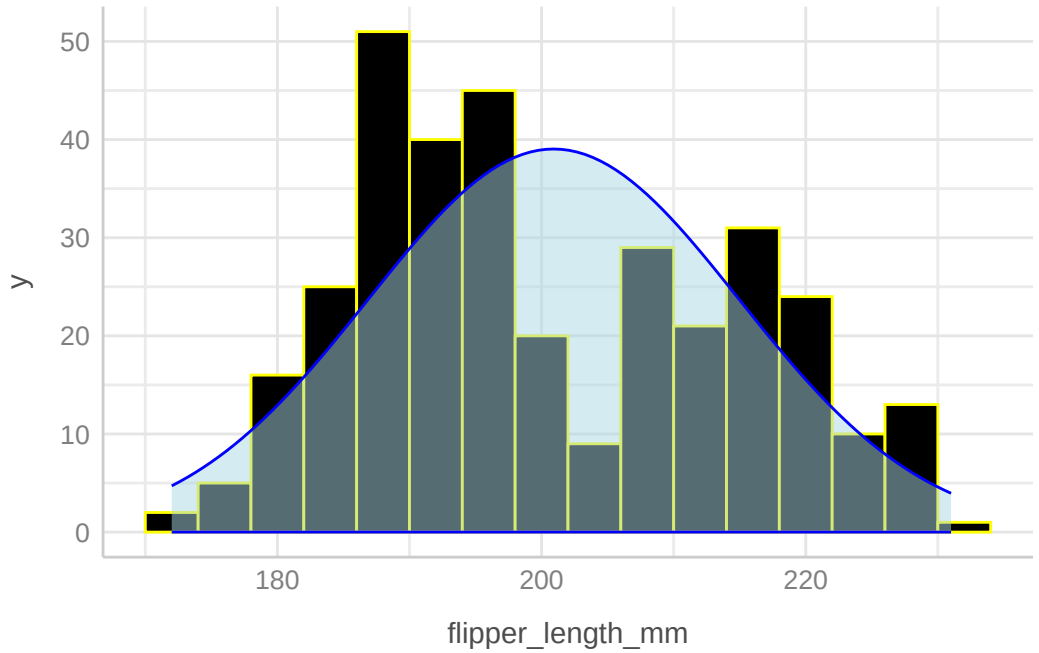
- To eliminate the penguins with missing information on flipper length, we filtered the data to only contain the cases (rows) with complete information on the `flipper_length_mm` variable. We'll use complete case approaches to deal with missingness for a while, until Section 17.7.
- The new color scheme follows CWRU's branding (visual identity) found at https://case.edu/brand/visual-identity/color. Specifically, we've used hexadecimal notation to identify CWRU Blue (HEX 003071) and CWRU Light Blue (HEX A6D2E6) as our desired fill and color, respectively.
- We've used a different theme than the default ggplot2 choice, called `theme_lucid()` which happens to be a favorite of mine. (My other favorite, used in most of this book, is `theme_bw()` which is very similar.) This results in a change in the background color from gray to white, among other things.
- We've also adjusted the x-axis label and added a main title to the plot.

2.4.2 Histogram with Normal Curve

Sometimes we might want to build a histogram with a curve superimposed to show what a Normal (or Gaussian) distribution with the same mean and standard deviation might be. My favorite way to do this is shown below.

```
bw = 4 # specify width of bins in histogram

ggplot(penguins, aes(flipper_length_mm)) +
  geom_histogram(binwidth = bw,
                 fill = "black", col = "yellow") +
  stat_function(fun = function(x)
    dnorm(x, mean = mean(penguins$flipper_length_mm,
                         na.rm = TRUE),
          sd = sd(penguins$flipper_length_mm,
                  na.rm = TRUE)) *
      length(penguins$flipper_length_mm) * bw,
    geom = "area", alpha = 0.5,
    fill = "lightblue", col = "blue") +
  theme_lucid()
```

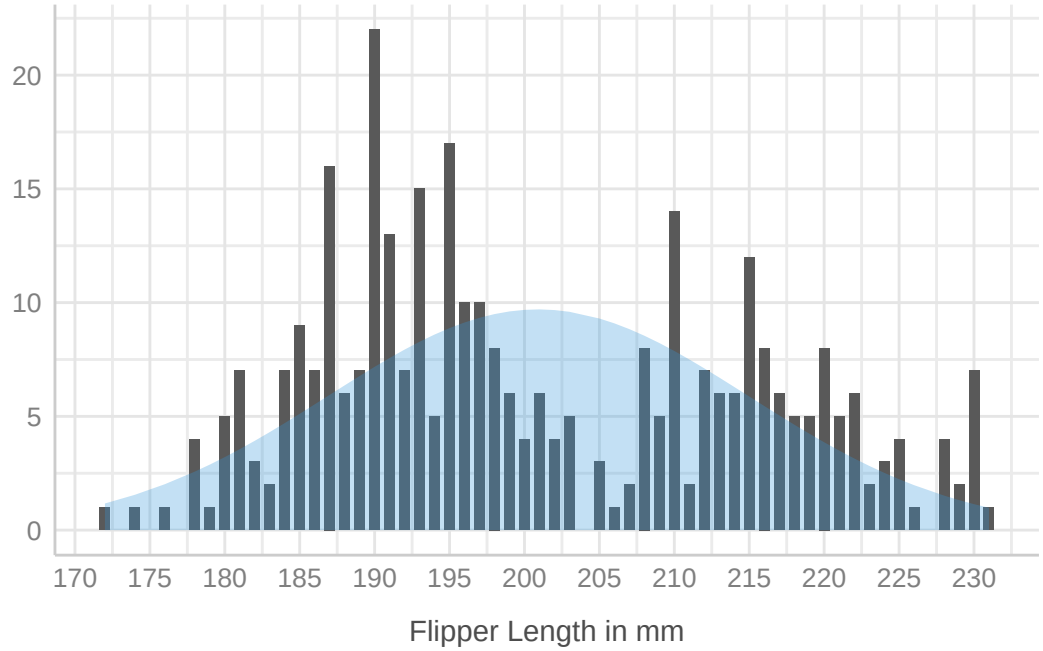
```
Warning: Removed 2 rows containing non-finite outside the scale range
(`stat_bin()`).
```



2.4.3 An alternative approach

Another way to do this produces a histogram using some tools from the **easystats** family of R packages, but the underlying plot is still a `ggplot2` plot, which can be adjusted in the same way as another such plot.

```
result <- describe_distribution(penguins$flipper_length_mm)

plot(result, centrality = "mean", dispersion = TRUE,
  dispersion_style = "curve"
) +
  labs(x = "Flipper Length in mm") +
  theme_lucid()
```

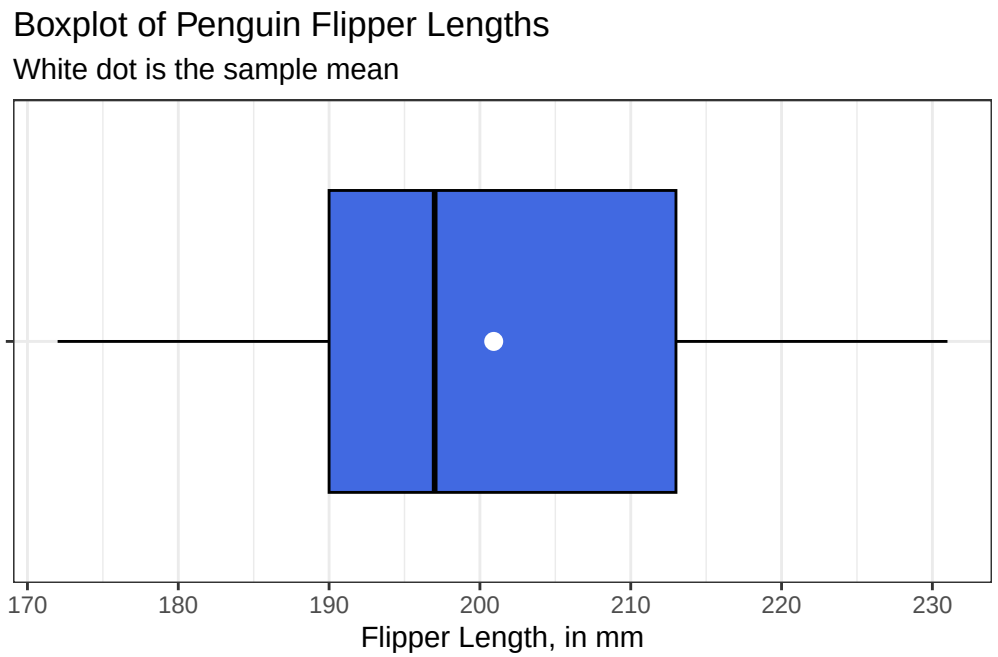


> **Note**
>
> On the good side, the `plot()` function used here already (by default) removes any missing values of flipper length before plotting.
> On the bad side, I don't know of a way to adjust the binwidths in this plot, which is something I prefer to be able to control. Hence, I generally use my approach.

2.4.4 Boxplot

```
penguins |>
  filter(complete.cases(flipper_length_mm)) |>
  ggplot(aes(x = flipper_length_mm, y = "")) +
  geom_boxplot(fill = "royalblue", color = "black") +
  stat_summary(
    fun = mean, geom = "point",
    shape = 16, size = 3, col = "white"
  ) +
  labs(
    x = "Flipper Length, in mm", y = "",
    title = "Boxplot of Penguin Flipper Lengths",
    subtitle = "White dot is the sample mean"
```

```
  ) +
  theme_bw()
```



This boxplot shows the middle 50% of the data (from the 25th percentile to the 75th percentile) as a box, with a vertical line in the middle indicating the median of the data (the 50th percentile.) The "whiskers" extend from the box out to the most extreme values seen in the data which do not qualify as potential outliers (unusual values) using a method due to Tukey (1977) and described in this book as part of Section 3.5.

I like to add an indication of the location of the sample **mean** of the data to my boxplots, so I have done so here.

The values plotted by this boxplot can be identified using the following R code:

```
fivenum(penguins$flipper_length_mm)
```

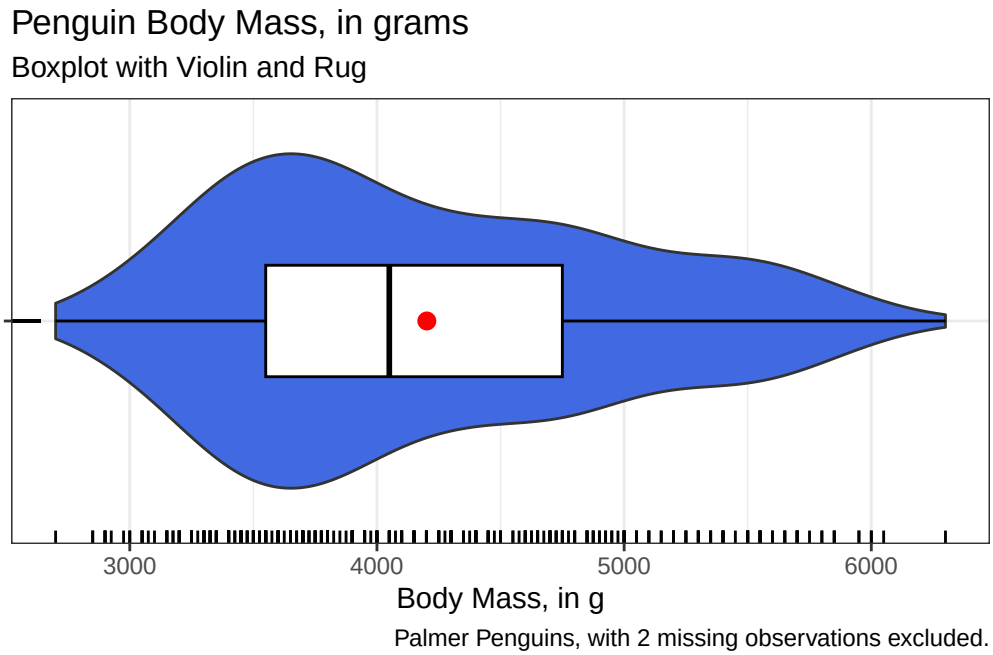
```
[1] 172 190 197 213 231
```

```
mean(penguins$flipper_length_mm, na.rm = TRUE)
```

```
[1] 200.9152
```

There are many ways to extend the fundamental idea of a boxplot. Here, for example, is a somewhat fancier option, now looking at the body mass (in g) of the penguins.

```
penguins |>
  filter(complete.cases(body_mass_g)) |>
  ggplot(aes(x = body_mass_g, y = "")) +
  geom_violin(fill = "royalblue") +
  geom_boxplot(width = 0.3, color = "black") +
  stat_summary(
    fun = mean, geom = "point",
    shape = 16, size = 3, col = "red"
  ) +
  geom_rug() +
  labs(
    x = "Body Mass, in g", y = "",
    title = "Penguin Body Mass, in grams",
    subtitle = "Boxplot with Violin and Rug",
    caption = "Palmer Penguins, with 2 missing observations excluded."
  ) +
  theme_bw()
```



2.4.5 Normal Q-Q Plot

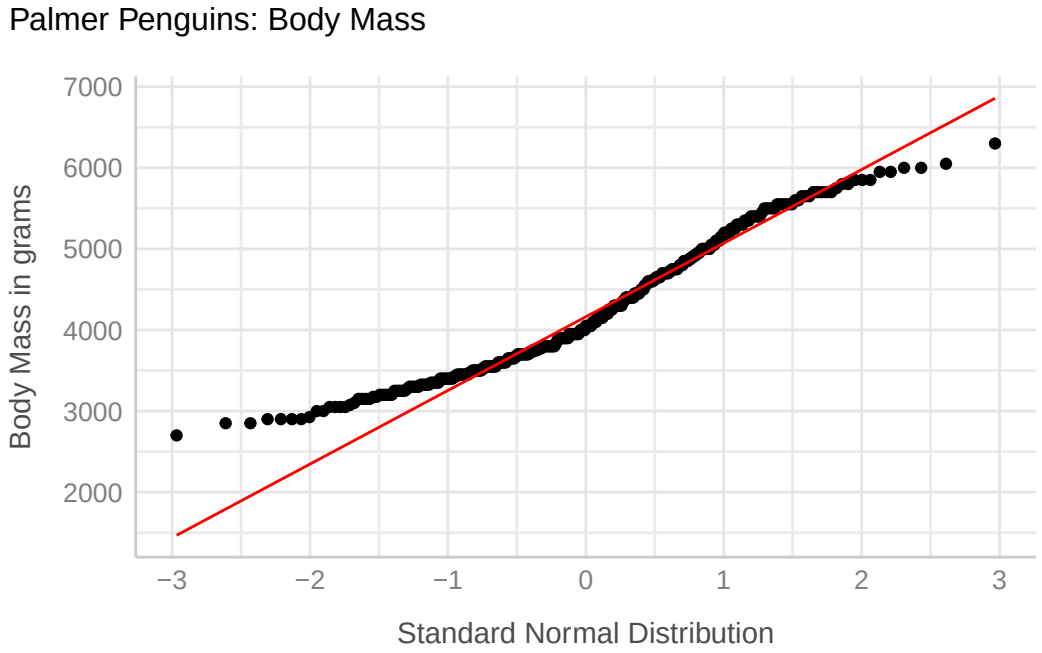
As it turns out, it will often be useful for us to consider whether or not a data set is well fit by a Normal distribution (sometimes called a Gaussian distribution.) One important tool for doing this is called a Normal probability plot or Normal Q-Q plot.

> **Note**
>
> The Central Limit Theorem from probability tells us that the sum of many small, independent random variables will produce a variable which approximated the Normal distribution. As a result, data that can be thought of as sums of many smaller additive factors often approximate a Normal distribution.
> A major appeal of the Normal distribution is that data that are Normally distributed can be well summarized by the mean and standard deviation of the sample.

A Normal Q-Q plot will follow a straight line when the data are (approximately) Normally distributed. When the data have a different shape, the plot will reflect that. The purpose of a Normal Q-Q plot is to help point out distinctions from a Normal distribution. A Normal distribution is symmetric and has certain expectations regarding its tails. The Normal Q-Q plot can help us identify data as well approximated by a Normal distribution, or not, because of:

- skew (including distinguishing between right skew and left skew) which is indicated by monotonic curves away from a straight line in the Normal Q-Q plot
- behavior in the tails (which could be heavy-tailed [more outliers than expected, shown as reverse "s" shapes] or light-tailed [shown as "s" shapes in the plot.])

```
penguins_cc <- penguins |>
  drop_na()

ggplot(penguins_cc, aes(sample = body_mass_g)) +
  geom_qq() +
  geom_qq_line(col = "red") +
  labs(
    y = "Body Mass in grams",
    x = "Standard Normal Distribution",
    title = "Palmer Penguins: Body Mass"
  ) +
  theme_lucid()
```

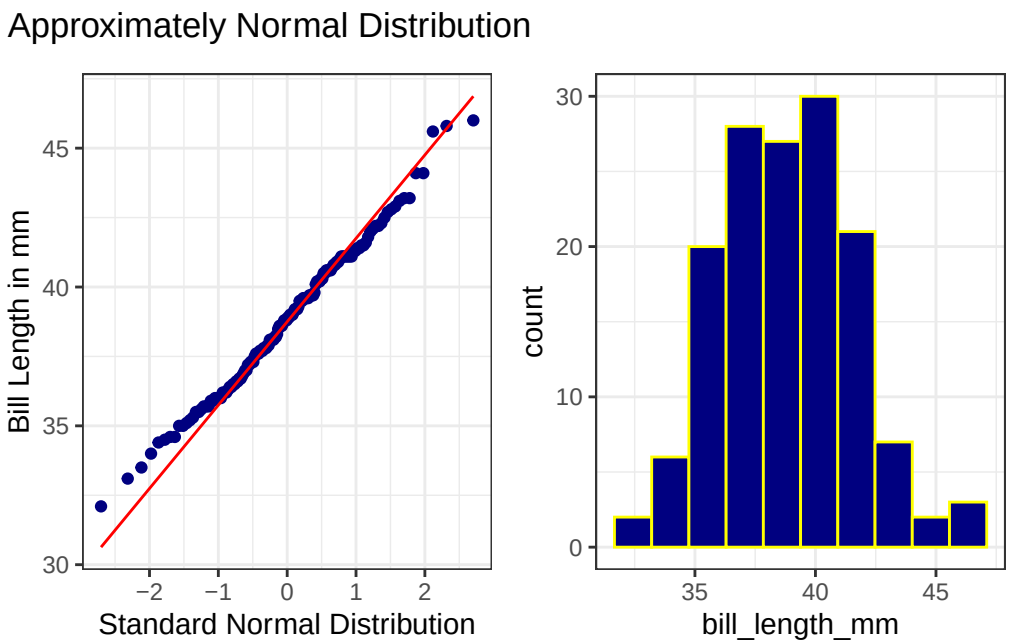


Here we see the points at the left of the plot curve above the red line, and the points at the right curve below the red line. This is the "s" shape I mentioned earlier, which indicates data that are somewhat more light-tailed than we'd expect from a Normal distribution.

Below are a couple of additional examples of Normal Q-Q plots in the `penguins` data, next to histograms to help us better understand the shape of a distribution.

```
dat1 <- penguins_cc |>
  filter(species == "Adelie")

p1 <- ggplot(dat1, aes(sample = bill_length_mm)) +
  geom_qq(col = "navy") +
  geom_qq_line(col = "red") +
  labs(
    y = "Bill Length in mm",
    x = "Standard Normal Distribution"
  ) +
  theme_bw()

p2 <- ggplot(dat1, aes(x = bill_length_mm)) +
  geom_histogram(bins = 10, col = "yellow", fill = "navy") +
  theme_bw()
```

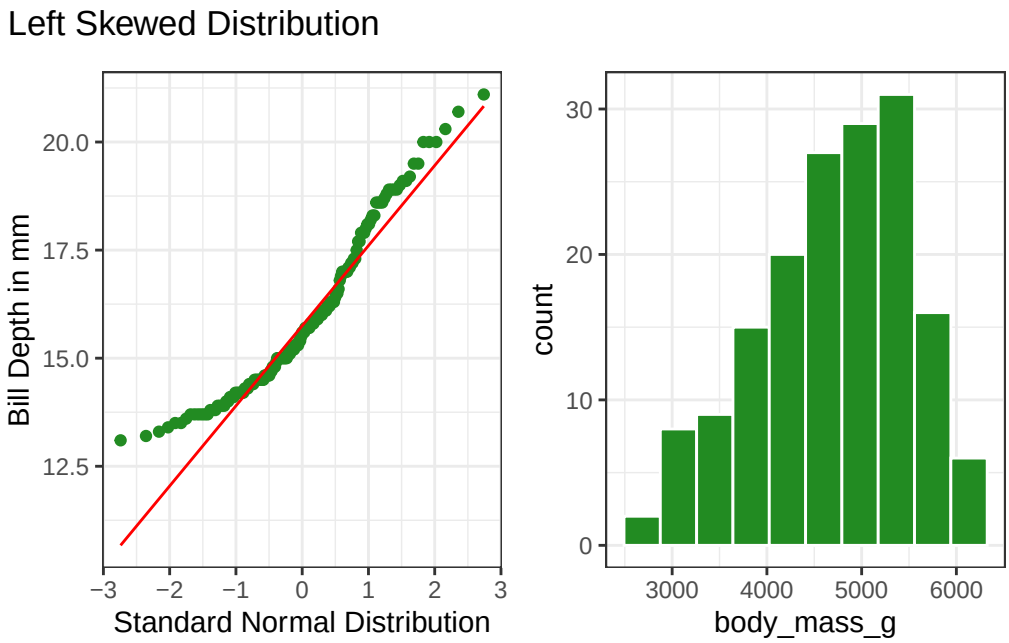
```
p1 + p2 +
  plot_annotation(title = "Approximately Normal Distribution")
```



While the Normal distribution isn't a perfect fit to these data, the points are generally quite close to the red line in the Normal Q-Q plot, and we see a mostly symmetric histogram to back up this finding.

```
dat2 <- penguins_cc |>
  filter(island == "Biscoe")

p3 <- ggplot(dat2, aes(sample = bill_depth_mm)) +
  geom_qq(col = "forestgreen") +
  geom_qq_line(col = "red") +
  labs(
    y = "Bill Depth in mm",
    x = "Standard Normal Distribution"
  ) +
  theme_bw()

p4 <- ggplot(dat2, aes(x = body_mass_g)) +
  geom_histogram(bins = 10, col = "white", fill = "forestgreen") +
  theme_bw()
```

```
p3 + p4 +
  plot_annotation(title = "Left Skewed Distribution")
```



Here, note that the points at the left and right side of the Normal Q-Q plot bend up away from the curve (more so at the bottom of the distribution) and this indicates skew.

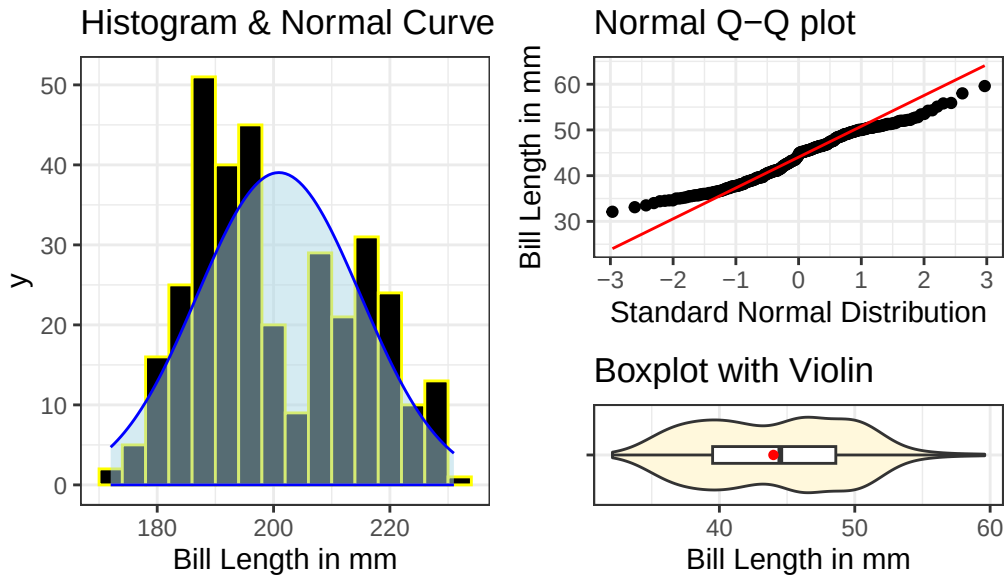
2.4.6 Three Plots at Once

Seeing more than one representation of the data at a time can help us use each to say something about the center, spread and shape of the distribution. The `patchwork` package allows us to place multiple plots into a single figure, as follows:

```
bw = 4 # specify width of bins in histogram

p1 <- ggplot(penguins, aes(x=flipper_length_mm)) +
  geom_histogram(binwidth = bw,
                 fill = "black", col = "yellow") +
  stat_function(fun = function(x)
    dnorm(x, mean = mean(penguins$flipper_length_mm,
                         na.rm = TRUE),
          sd = sd(penguins$flipper_length_mm,
                  na.rm = TRUE)) *
```

```
      length(penguins$flipper_length_mm) * bw,
    geom = "area", alpha = 0.5,
    fill = "lightblue", col = "blue"
    ) +
  labs(
    x = "Bill Length in mm",
    title = "Histogram & Normal Curve"
  ) +
  theme_bw()

p2 <- ggplot(penguins_cc, aes(sample = bill_length_mm)) +
  geom_qq() +
  geom_qq_line(col = "red") +
  labs(
    y = "Bill Length in mm",
    x = "Standard Normal Distribution",
    title = "Normal Q-Q plot"
  ) +
  theme_bw()

p3 <- ggplot(penguins_cc, aes(x = bill_length_mm, y = "")) +
  geom_violin(fill = "cornsilk") +
  geom_boxplot(width = 0.2) +
  stat_summary(
    fun = mean, geom = "point",
    shape = 16, col = "red"
  ) +
  labs(
    y = "", x = "Bill Length in mm",
    title = "Boxplot with Violin"
  ) +
  theme_bw()

p1 + (p2 / p3 + plot_layout(heights = c(2, 1))) +
  plot_annotation(
    title = "Palmer Penguins: Bill Length in mm"
  )
```

```
Warning: Removed 2 rows containing non-finite outside the scale range
(`stat_bin()`).
```



Here, we have a histogram with a superimposed Normal distribution curve, a Normal Q-Q plot, and a boxplot with violin to choose from, each describing the bill lengths in mm for all of the penguins with data on this measure.

- The histogram shows fewer observations near the mean of the data, but more (on both sides) as we move towards the tails. The edges of the distribution appear to have lighter tails than we'd expect if these data were Normally distributed.
- The S shape in the Normal Q-Q plot suggests symmetric but light-tailed data as compared to a Normal distribution.
- We see a lack of outliers in the boxplot, with the mean and median fairly close to one another, which indicates some symmetry but light-tailed shape.

2.4.7 Stem-and-Leaf Display

Prior to computers, a stem-and-leaf display was a useful tool for sorting through a batch of data by hand in a way that left you with an approximate histogram of the data values. I include it here out of nostalgia, primarily.

```
stem(penguins_cc$bill_depth_mm)
```

```
  The decimal point is at the |
```

```
13 | 1234
13 | 5567777778889999
14 | 001112222223334444
14 | 5555555566666778889
15 | 00000000001112222333344
15 | 5566677777888899999
16 | 0000111111222333344
16 | 555566666677888899
17 | 00000000000011111112222223333333334
17 | 55555566667788888888899999999999
18 | 000001111111122222333344444
18 | 55555555556666666666777777888888899999999
19 | 00000000011111122223344444
19 | 55555566677888999
20 | 0000001333
20 | 567778
21 | 11122
21 | 5
```

We see that the minimum bill depth is 13.1 and the maximum is 21.5.

Here's another stem-and-leaf display. Note the change in the location of the decimal point.

```
stem(penguins_cc$flipper_length_mm)
```

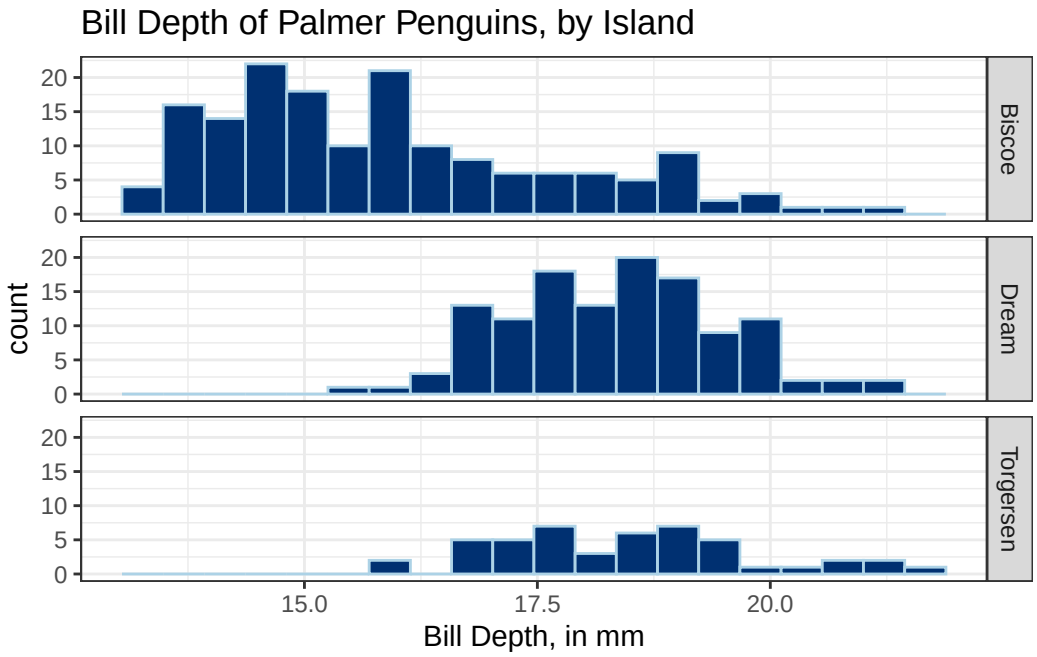
```
The decimal point is 1 digit(s) to the right of the |

17 | 24
17 | 68888
18 | 0000111111222334444444
18 | 55555555566666677777777777777778888889999999
19 | 00000000000000000000011111111111122222223333333333333344444
19 | 55555555555555555666666666677777777778888888899999
20 | 0000111111222233333
20 | 5556778888888899999
21 | 000000000000001122222223333344444
21 | 55555555555566666677778888899999
22 | 00000001111122222233444
22 | 55556888899
23 | 00000001
```

The minimum flipper_length is 172 mm and the maximum is 231.

2.5 Comparing Multiple Quantities

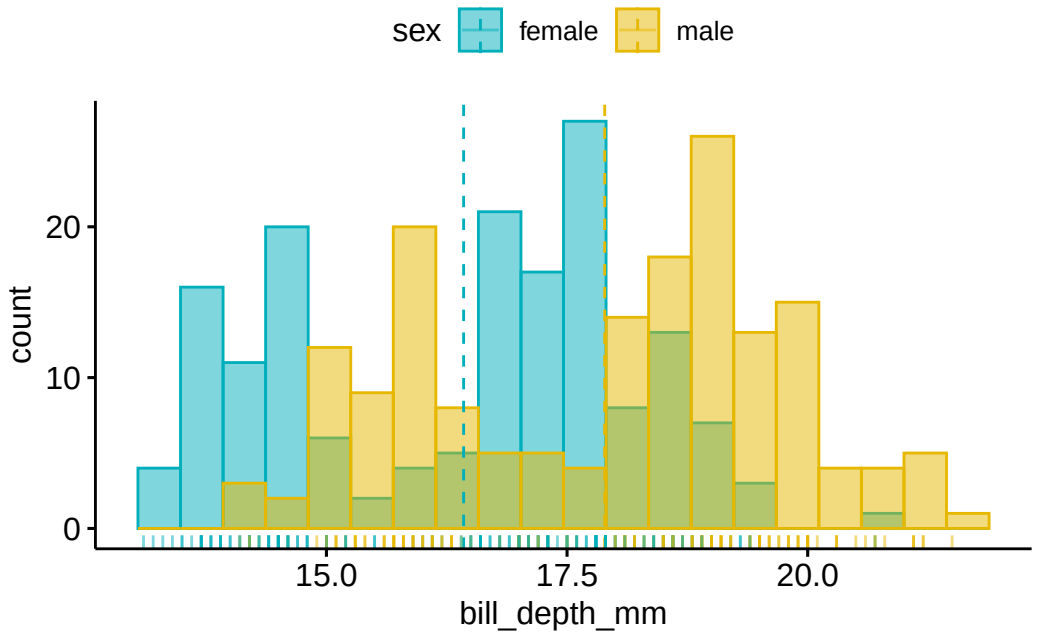
2.5.1 Faceted Histograms

```
ggplot(data = penguins_cc, aes(x = bill_depth_mm)) +
  geom_histogram(bins = 20, fill = "#003071", color = "#ACD2E6") +
  facet_grid(island ~ .) +
  labs(
    x = "Bill Depth, in mm",
    title = "Bill Depth of Palmer Penguins, by Island"
  ) +
  theme_bw()
```



2.5.2 Overlapping Histograms

```
gghistogram(penguins_cc,
  x = "bill_depth_mm",
  add = "mean", rug = TRUE, bins = 20,
  color = "sex", fill = "sex",
  palette = c("#00AFBB", "#E7B800")
)
```

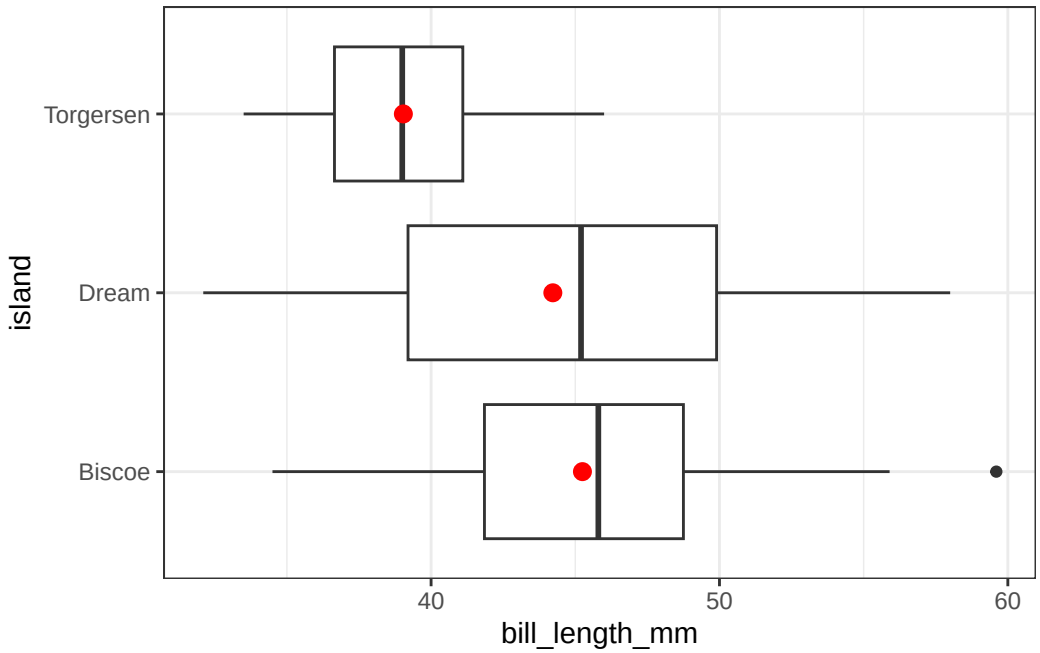


The `gghistogram()` function comes from the `ggpubr` package. Read more about its approaches to plotting distributions at https://rpkgs.datanovia.com/ggpubr/#distribution.

2.5.3 Comparison Boxplot, by Island

```
ggplot(
  data = penguins_cc,
  aes(x = bill_length_mm, y = island)
) +
  geom_boxplot() +
  stat_summary(
    fun = mean, geom = "point",
    shape = 16, size = 3, col = "red"
```

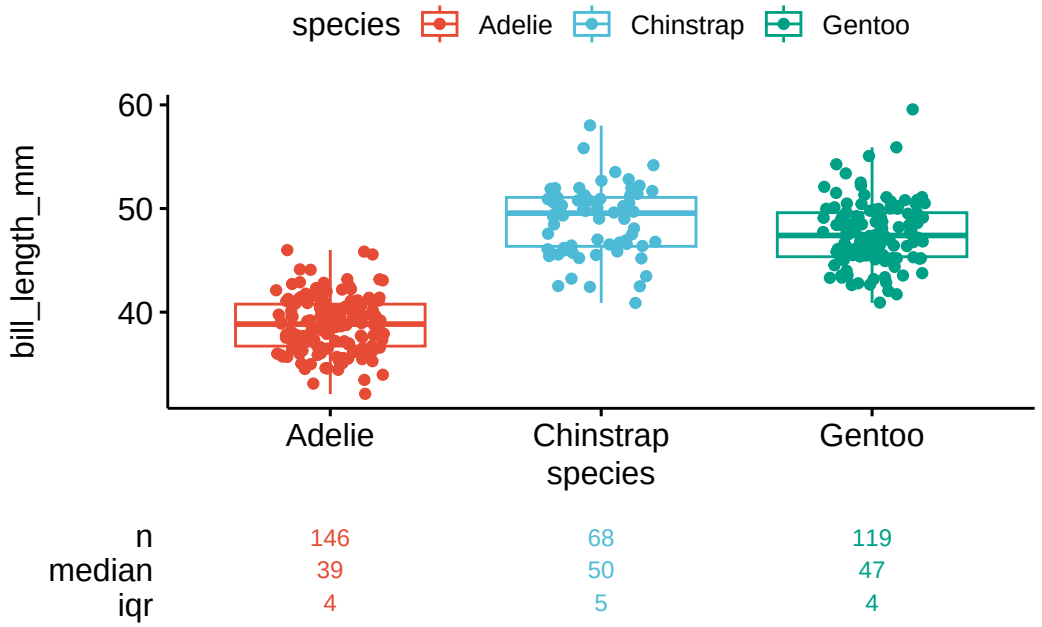
```
  ) +
  theme_bw()
```



The penguins coming from Torgeson island appear to have shorter bill lengths than do the penguins from the other two islands. Note also that there is a dot (indicating a potential outlier) in the Biscoe data, outside the whiskers for that plot. More on outliers in Section 3.5.

2.5.4 Adding Summary Statistics

```
ggsummarystats(
  penguins_cc,
  x = "species", y = "bill_length_mm",
  ggfunc = ggboxplot, add = "jitter",
  color = "species", palette = "npg"
)
```



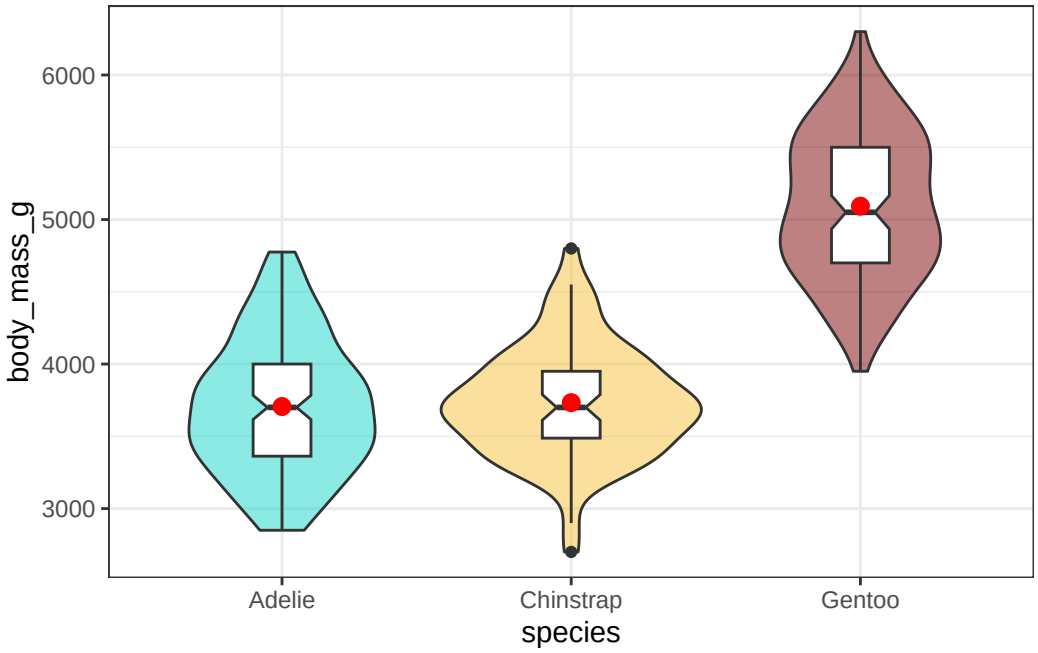
Adelie penguins appear smaller, on average, than do the penguins of the other two species.

The `ggsummarystats()` tool is part of the `ggpubr` package, and more details on this tool are available here.

2.5.5 Box and Violin Plot, by Species

```
ggplot(
  data = penguins_cc,
  aes(y = body_mass_g, x = species)
) +
  geom_violin(aes(fill = species)) +
  geom_boxplot(width = 0.2, notch = TRUE) +
  stat_summary(
    fun = mean, geom = "point",
    shape = 16, size = 3, col = "red"
  ) +
  scale_fill_viridis_d(
    option = "turbo", begin = 0.3,
    alpha = 0.5
  ) +
```

```
  guides(fill = "none") +
  theme_bw()
```



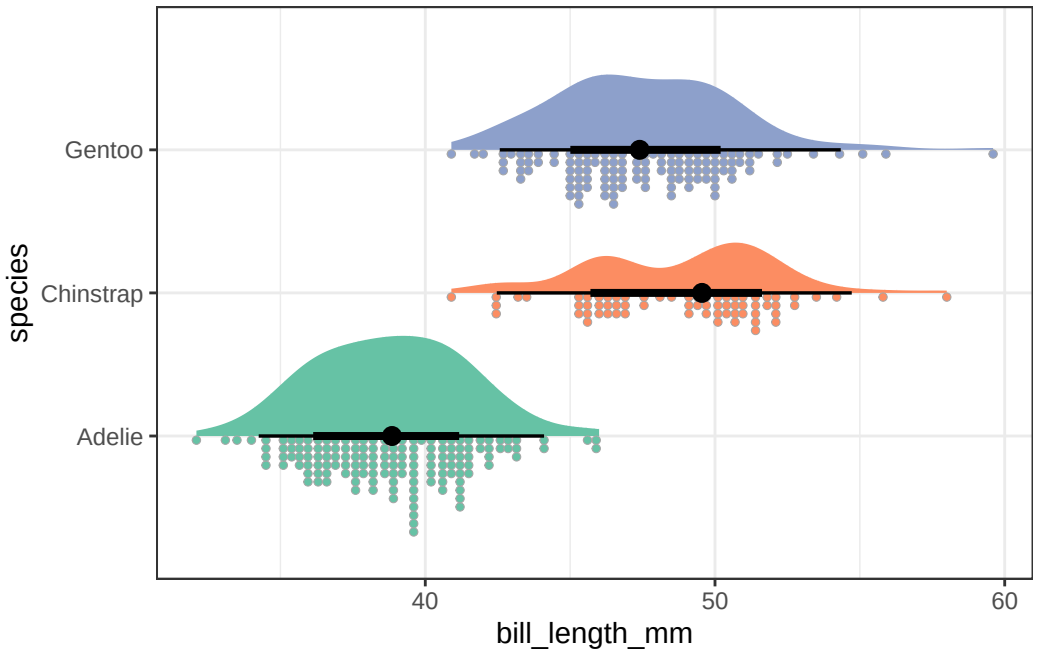
We have included a notch in each boxplot here to help compare the groups. The idea is that if the notches of two boxes do not overlap, this suggests a fairly large difference in the medians of the two groups. Here, this just provides a little more information to suggest that the Gentoo penguins have larger body mass than either of the other two species.

2.5.6 Rain Cloud Plot, by Species

A rain cloud plot is another way of visualizing the distributions of several groups on the same outcome simultaneously.

```
ggplot(
  data = penguins_cc,
  aes(
    y = species, x = bill_length_mm,
    fill = species
  )
) +
  stat_slab(aes(thickness = after_stat(pdf * n)), scale = 0.7) +
  stat_dotsinterval(side = "bottom", scale = 0.7, slab_linewidth = NA) +
```

```
  scale_fill_brewer(palette = "Set2") +
  guides(fill = "none") +
  theme_bw()
```



At the top of the plot is a half-violin plot, which displays a smoothed estimate of the values in the data. On the bottom, a dot chart shows the individual values grouped into small bins. In between, we show a dot at the group median, surrounded by a set of quantile intervals. The broader line shows an interval covering the middle 66% of the data, while the finer line shows an interval covering the middle 95% of the data.

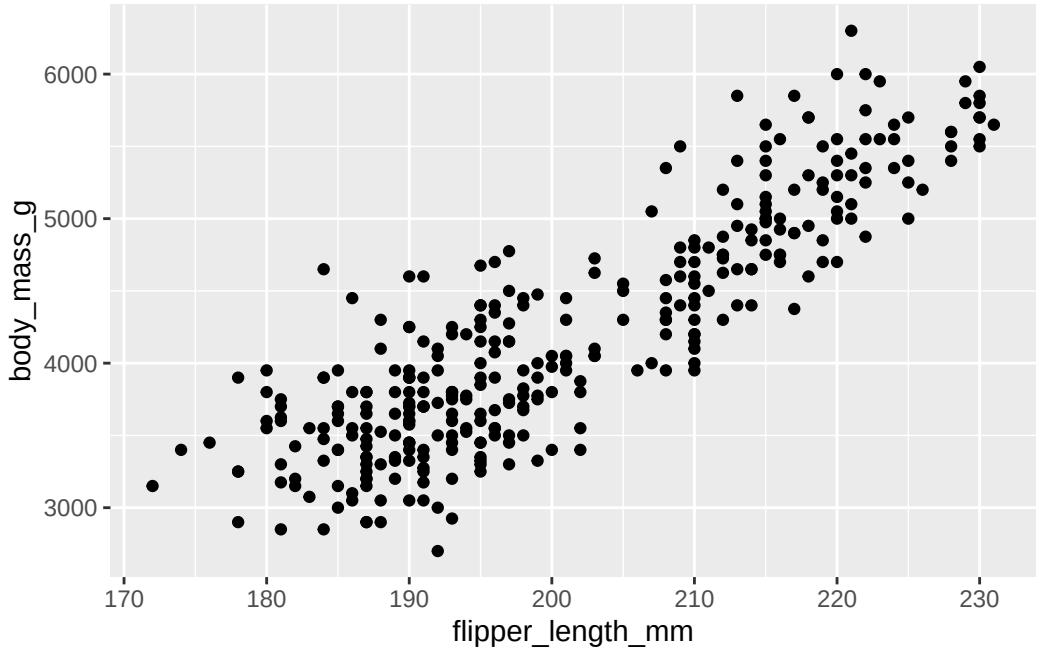
This implementation of rain cloud plots comes to us from the `ggdist` package, which contains several tools to help plot distributions and uncertainty. More information from that site on rain cloud plots and related options is available here.

2.6 Scatterplots of Associations

> When reporting data and analysis, you should always imagine yourself in the position of the reader of the report. – Gelman, Hill, and Vehtari (2021)

2.6.1 Building a basic scattterplot

```
ggplot(
  data = penguins_cc,
  aes(x = flipper_length_mm, y = body_mass_g)
) +
  geom_point()
```



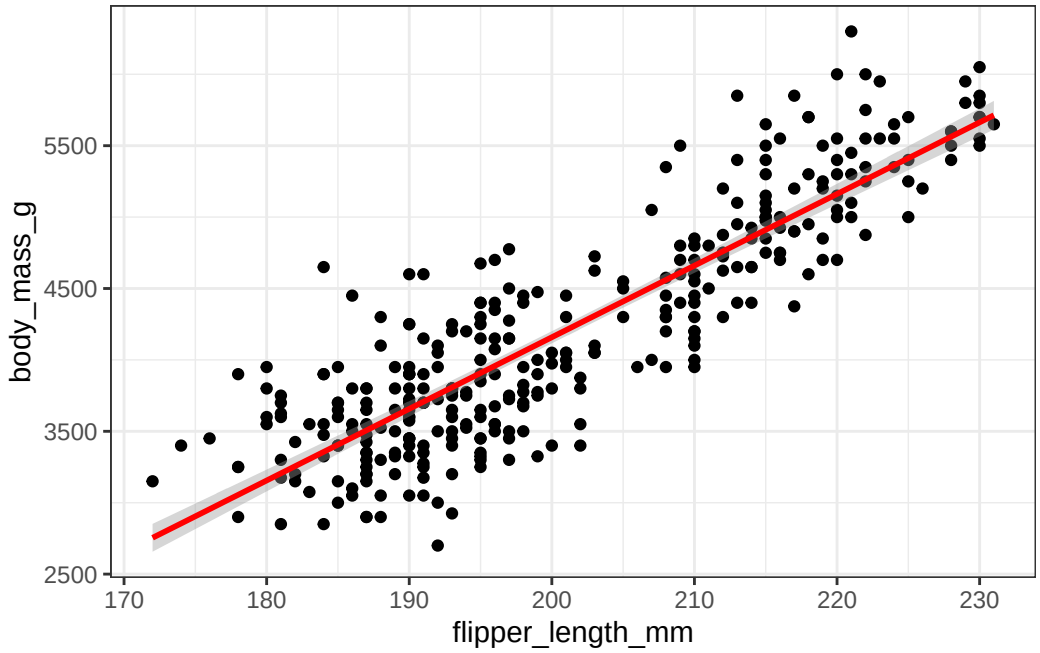
We see here a generally increasing relationship between these variables. On average, penguins with larger flipper lengths tend to have larger body mass.

2.6.2 Adding a straight line fit

The method of least squares (among other approaches) can be used to identify the intercept (value of y when x = 0) and slope (amount that y increases when x increases by 1) of a fitted line. In building a scatterplot, we can add this line by adding a call to `geom_smooth()` asking for a linear model (fit with `lm`), as follows:

```
ggplot(
  data = penguins_cc,
  aes(x = flipper_length_mm, y = body_mass_g)
```

```
) +
  geom_point() +
  geom_smooth(
    method = "lm",
    formula = y ~ x, se = TRUE,
    col = "red"
  ) +
  theme_bw()
```



The straight line fit through the data here uses the method of *least squares*. The line is selected to minimize the sum of the squared vertical distance (on the y-axis) from the data points to the regression line. For now, it's important to realize that this least squares line always goes through the point at the mean of our x variable (here, flipper length) and the mean of our y variable (here, body mass.)

The positive slope of this association again indicates that, in general, penguins with larger flipper lengths also have larger body masses.

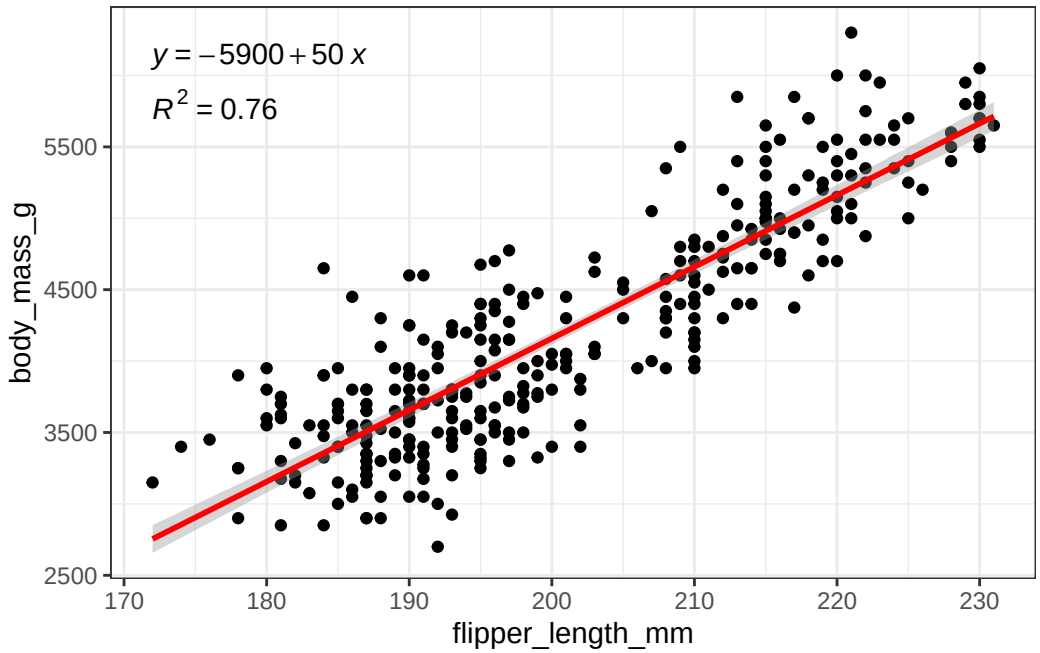
If you're interested, the equation for the slope and intercept of a least squares line fit to a scatter plot like this is found in Appendix E.

se Ribbon

In the R code above, we're including a request that a ribbon be plotted (with `se = TRUE`) around the regression line, to give us some idea of how much uncertainty we should perceive around the positioning of that line.

2.6.3 Adding the fitted equation

```
ggplot(
  data = penguins_cc,
  aes(x = flipper_length_mm, y = body_mass_g)
) +
  geom_point() +
  geom_smooth(
    method = "lm",
    formula = y ~ x, se = TRUE,
    col = "red"
  ) +
  stat_regline_equation() +
  stat_cor(aes(label = after_stat(rr.label)),
    label.x = 172, label.y = 5800
  ) +
  theme_bw()
```



Also, our model's R^2 value is 76%, or 0.76.

> **Tip**
>
> The `stat_regline_equation()` and `stat_cor()` functions which place these results in the plot come from the `ggpubr` package. Additional details are available at the ggpubr website.

2.6.4 Interpreting the model's R^2 value

The R^2 value above is the square of the Pearson correlation coefficient (see Appendix E for formulas), which we'll return to when we study association in Chapter 11. For now, the R^2 value can range from 0 to 1, with larger values indicating a stronger fit of the model to the data. We can interpret this R^2 value as indicating that our linear regression model using `flipper_length_mm` as a predictor accounts for 76% of the variation in our outcome, `body_mass_g`.

The correlation coefficient, r_{xy} can be determined here by recognizing that it is the square root of R^2 with the same sign as the slope of the regression line.

Here our regression line has a positive slope (in fact the slope is 50) and so the correlation coefficient $r_{xy} = \sqrt{0.76} = +0.87$ will also be positive.

```r
cor(
  penguins_cc$flipper_length_mm,
  penguins_cc$body_mass_g
)
```

```
[1] 0.8729789
```

2.6.5 Interpreting the Regression Equation

We have seen that our least squares regression estimate for these data is:

- body mass = -5900 + 50 flipper length + error.

The **intercept** here is not especially interesting. The value of -5900 estimates the average body mass of a penguin with a flipper length of 0 mm, and this is well beyond the range of our data[2]

The **slope** of the equation, on the other hand, is quite interesting. The safest and most sensible approach to describing a slope involves a comparison, for example:

"When comparing any two penguins who differ by 1 mm in flipper length, we observe a body mass that is, on average, 50 grams larger for the penguin with larger flipper length."

or

"Under our fitted model, the average difference in body mass, comparing two penguins with a one mm difference in flipper length, is 50 grams."

or

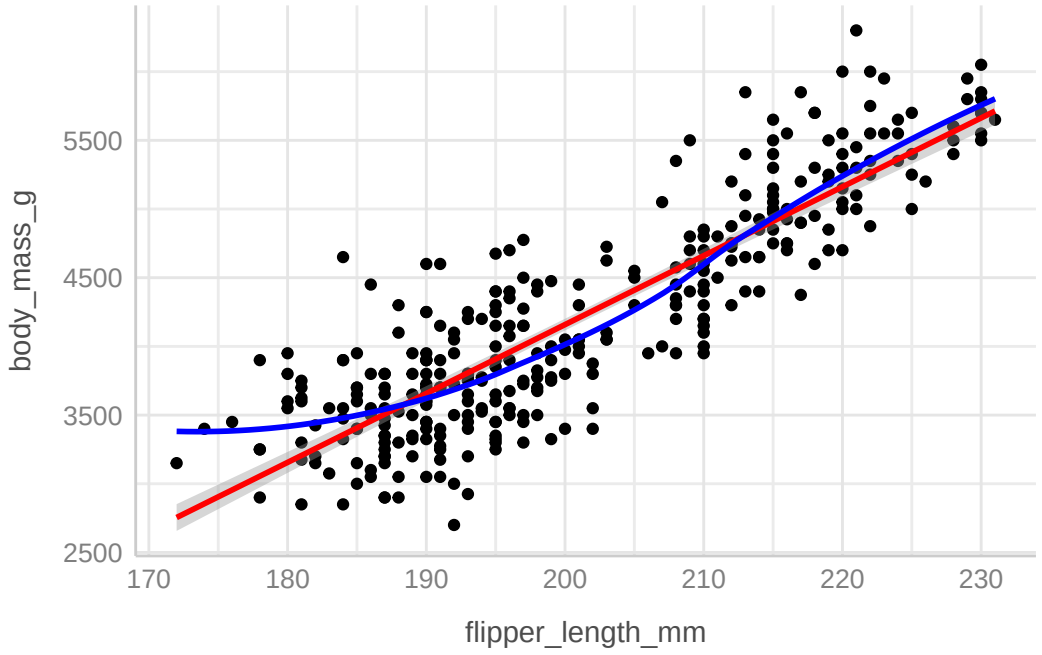
"If a penguin named Jack's flipper length is one mm larger than a penguin named Harry, then on average, Jack's body mass will be 50 grams larger than Harry's, according to our model."

> **Note**
>
> It may be tempting to report that the estimated *effect* of one mm of flipper length on a penguin's body mass is 50 grams, but we cannot justify that without many assumptions (related to causality) that don't hold in this case.
> The idea of an effect is really reserved for a change associated with some sort of treatment or intervention. But we don't have the ability to intervene on penguins to make their flippers longer.

[2]For instance, all penguins in our data have flipper lengths between 172 and 231 mm.

2.6.6 Add loess smooth

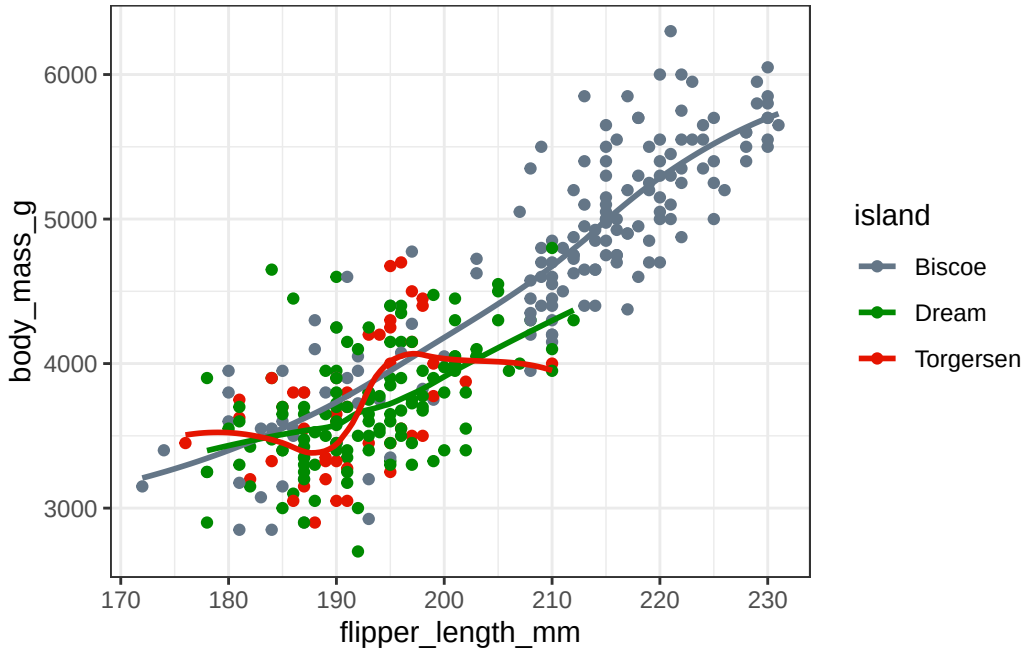
```
ggplot(
  data = penguins_cc,
  aes(x = flipper_length_mm, y = body_mass_g)
) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, se = TRUE, col = "red") +
  geom_smooth(method = "loess", formula = y ~ x, se = FALSE, col = "blue") +
  theme_lucid()
```



2.6.7 Smooths within each island

```
ggplot(
  data = penguins_cc,
  aes(
    x = flipper_length_mm, y = body_mass_g,
    col = island
  )
) +
```

```
  geom_point() +
  geom_smooth(
    method = "loess", formula = y ~ x, se = FALSE
  ) +
  scale_color_metro_d() +
  theme_bw()
```



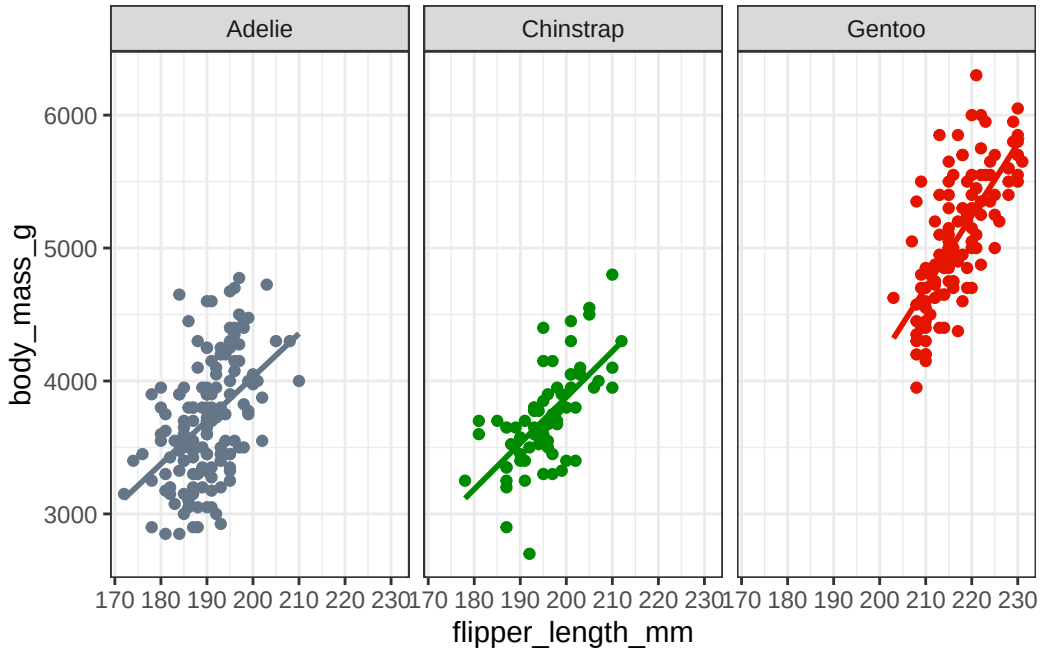
One thing we spot here is that we have a much broader range of flipper lengths in the penguins from Biscoe than in those from Dream or Torgeson, each of which seem to only have penguins with flippers between about 180 and 210 mm long.

For all three islands, we have a generally positive association between body mass and flipper length, especially in the Biscoe and Dream island groups.

2.6.8 Linear Model faceted by species

```
ggplot(
  data = penguins_cc,
  aes(
    x = flipper_length_mm, y = body_mass_g,
    col = species
```

```
  )
) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE) +
  scale_color_metro_d() +
  facet_wrap(~species) +
  guides(color = "none") +
  theme_bw()
```



Here, we use the `facet_wrap()` function to produce what are often referred to as *small multiples* - a series of closely related graphs. It is very clear that the slopes of the flipper length and body mass association are all similar here, but also that the Gentoo penguins are generally much larger (in terms of each variable) than the other two species.

2.7 A Few Key Points

These comments are highly motivated by Gelman, Hill, and Vehtari (2021), Chapter 2, which is available to you online (pdf).

> Never display a graph you can't talk about.
>
> The goal of any graph is communication to self or others.

Gelman, Hill, and Vehtari (2021) describe three common uses of graphics:

- Exploratory analyses of raw data
- Graphs of fitted models
- Graphs presenting final results

R (and ggplot tools in particular) makes it relatively easy to make attractive plots in all three of these settings.

Finally, it's almost always useful to think about a graph in terms of what comparison it makes, and to design the graph so that the comparison you want to make is shown accurately and helpfully.

2.8 For More Information

1. An introduction to ggplot2 at the ggplot2 web site.
2. Relevant sections of R for Data Science include:

- Data Visualization
- The Visualization unit, which includes sections on Layers, Exploratory Data Analysis, and Communication that are worth your time.

3. Relevant sections of the R Graphics Cookbook include:

- Summarized Data Distributions, which introduces some key code for building histograms and related items like density curves, frequency polygons, box plots, violin plots and dot plots.
- Scatter Plots which introduces several useful code blocks to improve and augment your scatterplots.

4. Relevant sections of OpenIntro Stats (pdf) include:

- Section 2 on Summarizing Data
- Section 4.1 on the Normal distribution

5. For an introductory example covering some key aspects of data, consider reviewing Chapters 1-3 of Introduction to Modern Statistics, 2nd Edition (Çetinkaya-Rundel and Hardin 2024) which discuss an interesting case study on using stents to prevent strokes. We'll reference this again at the end of our next Chapter.
6. Another great resource on data and measurement issues is Chapter 2 of Gelman, Hill, and Vehtari (2021), available in PDF at this link. We'll reference this again at the end of our next Chapter.

3 Numerical Summaries

3.1 R setup for this chapter

> **Note**
>
> This section loads all needed R packages for this chapter. Appendix A lists all R packages used in this book, and also provides R session information. Appendix B describes the 431-Love.R script, and demonstrates its use.

```
library(janitor)
library(knitr)
library(naniar)
library(palmerpenguins)

library(easystats)
library(tidyverse)

source("data/Love-431.R")
```

3.2 Data require structure and context

Descriptive statistics are concerned with the presentation, organization and summary of data, as suggested in Norman and Streiner (2014). This includes various methods of organizing and graphing data to get an idea of what those data can tell us.

As Vittinghoff et al. (2012) suggest, the nature of the measurement determines how best to describe it statistically, and the main distinction is between **numerical** and **categorical** variables. Even this is a little tricky - plenty of data can have values that look like numerical values, but are just numerals serving as labels.

As Bock, Velleman, and De Veaux (2004) point out, the truly critical notion, of course, is that data values, no matter what kind, are useless without their contexts. The Five W's (Who, What [and in what units], When, Where, Why, and often How) are just as useful for

establishing the context of data as they are in journalism. If you can't answer Who and What, in particular, you don't have any useful information.

In general, each row of a data frame corresponds to an individual (respondent, experimental unit, record, or observation) about whom some characteristics are gathered in columns (and these characteristics may be called variables, factors or data elements.) Every column / variable should have a name that indicates *what* it is measuring, and every row / observation should have a name that indicates *who* is being measured.

3.3 Data Source: The Palmer Penguins, again

> **Note**
>
> Appendix C provides further guidance on pulling data from other systems into R, while Appendix D gives more information (including download links) for all data sets used in this book.

Again, we'll use the `penguins` tibble from the `palmerpenguins` package, as we did in Chapter 2.

```
penguins
```

```
# A tibble: 344 x 8
   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
   <fct>   <fct>              <dbl>         <dbl>             <int>       <int>
 1 Adelie  Torgersen           39.1          18.7               181        3750
 2 Adelie  Torgersen           39.5          17.4               186        3800
 3 Adelie  Torgersen           40.3          18                 195        3250
 4 Adelie  Torgersen           NA            NA                  NA          NA
 5 Adelie  Torgersen           36.7          19.3               193        3450
 6 Adelie  Torgersen           39.3          20.6               190        3650
 7 Adelie  Torgersen           38.9          17.8               181        3625
 8 Adelie  Torgersen           39.2          19.6               195        4675
 9 Adelie  Torgersen           34.1          18.1               193        3475
10 Adelie  Torgersen           42            20.2               190        4250
# i 334 more rows
# i 2 more variables: sex <fct>, year <int>
```

```
n_miss(penguins)
```

```
[1] 19
```

Almost all serious statistical analyses have to deal with missing data. Data values that are missing are indicated in R, and to R, by the symbol `NA`.

To simplify our work in this chapter, we'll focus on a version of the `penguins` data which includes only complete observations, with known values for all 8 variables in the tibble. To create such a version, which filters the observations down to those penguins without any missing data on any variables, we use the `drop_na()` function, like this:

```
penguins_cc <- penguins |> drop_na()

dim(penguins_cc)
```

```
[1] 333   8
```

```
n_miss(penguins_cc)
```

```
[1] 0
```

We will filter to complete cases by dropping missing values for much of this book, postponing the use of imputation-based approaches to dealing with missing data until Section 17.7.

Until that time, we will make the formal assumption that all missing data are *missing completely at random*, a term we'll define more carefully in Section 17.7. If that MCAR assumption holds, then a complete case analysis is appropriate.

3.4 Numerical Summaries

Visit Appendix E as you like for additional information about statistical formulas relevant to this chapter and the book more generally.

3.4.1 Quantitative Variables

Variables recorded in numbers that we use as numbers are called **quantitative**. Familiar examples include incomes, heights, weights, ages, distances, times, and counts. All quantitative variables have measurement units, which tell you how the quantitative variable was measured. Without units (like miles per hour, angstroms, yen or degrees Celsius) the values of a quantitative variable have no meaning.

- It does little good to be told the price of something if you don't know the currency being used.

- You might be surprised to see someone whose age is 72 listed in a database on childhood diseases until you find out that age is measured in months.
- Often just seeking the units can reveal a variable whose definition is challenging - just how do we measure "friendliness", or "success," for example.
- Quantitative variables may also be classified by whether they are **continuous** or can only take on a **discrete** set of values. Continuous data may take on any value, within a defined range. Suppose we are measuring height. While height is really continuous, our measuring stick usually only lets us measure with a certain degree of precision. If our measurements are only trustworthy to the nearest centimeter with the ruler we have, we might describe them as discrete measures. But we could always get a more precise ruler. The measurement divisions we make in moving from a continuous concept to a discrete measurement are usually fairly arbitrary. Another way to think of this, if you enjoy music, is that, as suggested in Norman and Streiner (2014), a piano is a *discrete* instrument, but a violin is a *continuous* one, enabling finer distinctions between notes than the piano is capable of making. Sometimes the distinction between continuous and discrete is important, but usually, it's not.

3.4.2 `summary()` for a data frame / tibble

```
summary(penguins)
```

```
      species          island    bill_length_mm  bill_depth_mm
 Adelie   :152   Biscoe   :168   Min.   :32.10   Min.   :13.10
 Chinstrap: 68   Dream    :124   1st Qu.:39.23   1st Qu.:15.60
 Gentoo   :124   Torgersen: 52   Median :44.45   Median :17.30
                                 Mean   :43.92   Mean   :17.15
                                 3rd Qu.:48.50   3rd Qu.:18.70
                                 Max.   :59.60   Max.   :21.50
                                 NA's   :2       NA's   :2
 flipper_length_mm  body_mass_g       sex           year
 Min.   :172.0     Min.   :2700   female:165   Min.   :2007
 1st Qu.:190.0     1st Qu.:3550   male  :168   1st Qu.:2007
 Median :197.0     Median :4050   NA's  : 11   Median :2008
 Mean   :200.9     Mean   :4202                Mean   :2008
 3rd Qu.:213.0     3rd Qu.:4750                3rd Qu.:2009
 Max.   :231.0     Max.   :6300                Max.   :2009
 NA's   :2         NA's   :2
```

As we can see, the `summary()` function, when applied to a tibble or data frame provides counts of each level for factor (categorical) variables like `species`, `sex` and `island`. If values are missing, as in the `sex` variable, they are indicated with counts of NA's (NA stands for Not Available.)

For quantities like `bill_length_mm`, `bill_depth_mm`, `flipper_length_mm`, and `body_mass_g`, we also see counts of missing values gathered as NA's. In addition, we get a Mean of the sample, and a five-number summary of its distribution.

Summary Statistic	Description
Min or *Minimum*	Smallest non-missing value in data.
1st Quartile or *Q1*	25th percentile (median of lower half of data.)
Median or *Q2*	Middle value when data are sorted in order.
3rd Quartile or *Q3*	75th percentile (median of upper half of data.)
Max or *Maximum*	Largest non-missing value in data.
Mean	Sum of the values divided by the number of values.

3.4.3 `fivenum()` for a five-number summmary

Note that we can use the `fivenum()` function to obtain these first five values.

```
penguins |>
  select(bill_length_mm) |>
  fivenum()
```

```
[1] 32.10 39.20 44.45 48.50 59.60
```

As noted in Chapter 2, these are the summaries indicated by an ordinary boxplot.

Another, perhaps simpler, way to obtain this information is:

```
fivenum(penguins$bill_length_mm)
```

```
[1] 32.10 39.20 44.45 48.50 59.60
```

We could, for instance, summarize any distribution using uncertainty intervals built from these quantiles. For example, 39.2 to 48.5 is a central 50% interval for these data.

3.4.4 More summaries with `lovedist()`

We can obtain some additional summaries for any particular variable using a function called `lovedist()` from the `Love-431.R` script which is described in Appendix B.

```
penguins |>
  reframe(lovedist(bill_length_mm))
```

```
# A tibble: 1 x 10
      n  miss  mean    sd   med   mad   min   q25   q75   max
  <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1   344     2  43.9  5.46  44.4  7.04  32.1  39.2  48.5  59.6
```

This approach adds four new summaries to the mean and five-number summary available from `summary()`. Formulas are available in Appendix E.

Summary Statistic	Description
n or count	number of values contained in the variable (including missing)
miss	number of missing values
sd	standard deviation, a measure of variation
mad	median absolute deviation, also a measure of variation

We often summarize the center and spread (variation) in our data with a two-part summary.

- Should the data approximate a Normal distribution (bell-shaped curve,) the mean and standard deviation are commonly used to describe the variation.
- The median and mad are generally better options when a Normal distribution isn't a good fit to our data.

We can also use `lovedist()` to look at stratified summaries within groups defined by a factor. Here, we also show the use of the `kable()` function (from the `knitr` package) to tidy up the table and reduce the number of decimal places shown.

```
penguins |>
  reframe(lovedist(bill_length_mm), .by = species) |>
  kable(digits = 2)
```

species	n	miss	mean	sd	med	mad	min	q25	q75	max
Adelie	152	1	38.79	2.66	38.80	2.97	32.1	36.75	40.75	46.0

species	n	miss	mean	sd	med	mad	min	q25	q75	max
Gentoo	124	1	47.50	3.08	47.30	3.11	40.9	45.30	49.55	59.6
Chinstrap	68	0	48.83	3.34	49.55	3.63	40.9	46.35	51.08	58.0

3.4.5 Mean and SD, Median and MAD

As mentioned, one natural option (especially appropriate when the data follow a Normal distribution) is to use the sample mean and standard deviation.

```
mean_sd(penguins$bill_length_mm)
```

```
     -SD     Mean      +SD
38.46235 43.92193 49.38151
```

The `mean_sd` function shows us the sample mean $\pm$ the standard deviation. Should the data follow a Normal distribution,

- the sample mean $\pm$ 1 standard deviation forms an interval which covers approximately 68% of the data.
- The sample mean $\pm$ 2 standard deviations will cover 95% of data which follow a Normal distribution.
- The sample mean $\pm$ 3 standard deviations covers approximately 99.7% of the data for a Normal distribution.

> **Note**
>
> The standard deviation is also defined as the square root of the **variance**, which is another important measure of variation.
>
> ```
> var(penguins$bill_length_mm, na.rm = TRUE)
> ```
>
> ```
> [1] 29.80705
> ```
>
> ```
> sd(penguins$bill_length_mm, na.rm = TRUE)
> ```
>
> ```
> [1] 5.459584
> ```
>
> ```
> sd(penguins$bill_length_mm, na.rm = TRUE)^2
> ```
>
> ```
> [1] 29.80705
> ```

> As Gelman, Hill, and Vehtari (2021) point out, the variance is interpretable as the mean of the squared difference from the mean.
> The standard deviation is more often used to measure variation than the variance because the standard deviation has the same units of measurement as the mean.

When we don't have much data, or when we're unwilling to conclude that the data match up well with a Normal distribution, a more robust pair of estimates is the median and the MAD.

```
median_mad(penguins$bill_length_mm)
```

```
    -MAD   Median     +MAD
37.40765 44.45000 51.49235
```

As noted in Appendix E, the mad is scaled by R to take the same value as the standard deviation if the data are really Normally distributed, much as a Normal distribution's mean is equal to its median.

We can often interpret a median/mad combination in an analogous way to our interpretation of the mean and standard deviation, as a set of building blocks for intervals which describe the center and spread of our data, even when the data show skew or other non-Normality. As Gelman, Hill, and Vehtari (2021) point out,

> … median-based summaries are also frequently more computationally stable (than means and standard deviations,) and more robust against outlying values.

3.4.6 Coefficient of Variation

The coefficient of variation is the sample standard deviation divided by the mean. It quantifies the variation in the data related to the mean, and is useful only when describing quantities that have a meaningful zero value.

```
penguins_cc |> summarise(mean = mean(flipper_length_mm),
                         var = var(flipper_length_mm),
                         sd = sd(flipper_length_mm),
                         cv = sd / mean)
```

```
# A tibble: 1 x 4
   mean   var    sd     cv
  <dbl> <dbl> <dbl>  <dbl>
1  201.  196.  14.0 0.0697
```

We can use the `coef_var()` function from the `datawizard` package in the `easystats` family to compute this value.

```
coef_var(penguins$flipper_length_mm)
```

```
[1] NA
```

Since we have some missing data, this summary needs to be told to remove it.

```
coef_var(penguins$flipper_length_mm, remove_na = TRUE)
```

```
[1] 0.0699883
```

```
coef_var(penguins_cc$flipper_length_mm)
```

```
[1] 0.06974164
```

Notice that `penguins_cc` contains only the 333 observations with complete data on all 8 variables included in the `penguins` data, while removing the NA in the second function below only removes the 2 penguins witth missing values of `flipper_length_mm` from the original set of 344 penguins.

3.4.7 Standard Error of the Sample Mean

The standard error of the sample mean is the final summary we'll discuss here for a quantitative variable like `bill_depth_mm`. It is the sample standard deviation divided by the square root of the sample size.

```
penguins_cc |> summarise(n = n(),
                         mean = mean(bill_depth_mm),
                         sd = sd(bill_depth_mm),
                         se_mean = sd / sqrt(n))
```

```
# A tibble: 1 x 4
      n  mean    sd se_mean
  <int> <dbl> <dbl>   <dbl>
1   333  17.2  1.97   0.108
```

Generally, a standard error is the estimated standard deviation of an estimate, and is a good way of tellinng us about uncertainty in our estimate.

3.4.8 `describe_distribution()`

Another approach, from the datawizard package within the `easystats` family, is `describe_distribution()`, which provides some additional summaries for quantities within our data frame.

```
describe_distribution(penguins) |> kable(digits = 2)
```

Variable	Mean	SD	IQR	Min	Max	Skewness	Kurtosis	n	n_Missing
bill_length_mm	43.92	5.46	9.30	32.1	59.6	0.05	-0.88	342	2
bill_depth_mm	17.15	1.97	3.12	13.1	21.5	-0.14	-0.91	342	2
flipper_length_mm	200.92	14.06	23.25	172.0	231.0	0.35	-0.98	342	2
body_mass_g	4201.75	801.95	1206.25	2700.0	6300.0	0.47	-0.72	342	2
year	2008.03	0.82	2.00	2007.0	2009.0	-0.05	-1.50	344	0

This approach adds four new summaries to those we've seen before, specifically:

Summary Statistic	Description
IQR	inter-quartile range = 75th - 25th percentile
Range	combination of minimum and maximum, or the difference between the maximum and minimum values
Skewness	a measure of the asymmetry in the distribution of our data (positive values indicate right skew, negative values indicate left skew)
Kurtosis	a measure of tail behavior in the distribution of our data (positive values indicate fatter tails than a Normal distribution, while negative values indicate thinner tails)

See Appendix E for more on the measures for skew and kurtosis used by `describe_distribution()`. In practice, I very rarely look at those two summaries, preferring to use visualizations to build my understanding of the shape of a distribution.

3.4.9 Augmenting a `describe_distribution()`

```
set.seed(431)
penguins |>
  select(bill_depth_mm, body_mass_g) |>
  describe_distribution(ci = 0.95, iterations = 500,
                        quartiles = FALSE) |>
  kable(digits = 2)
```

Variable	Mean	SD	IQR	CI_low	CI_high	Min	Max	Skewness	Kurtosis	n	n_Missing
bill_depth_mm	17.15	1.97	3.12	16.95	17.35	13.1	21.5	-0.14	-0.91	342	2
body_mass_g	4201.75	801.95	1206.25	4119.36	4283.31	2700.0	6300.0	0.47	-0.72	342	2

- In estimating the mean of bill depth, our sample yields a point estimate of 17.15 mm and a 95% bootstrap confidence interval which extends from 16.95 to 17.35 mm.

Here, we augment the `describe_description()` results by adding a bootstrap **confidence interval** for the mean of the quantities of interest, and make room for that in the table by dropping the information about the individual quartiles. The bootstrap approach (which we'll discuss later in some detail) requires us to set a random seed to help with replication, and to specify which level of confidence we want to use (here, I used 95%.)

Another approach we can take is to find a bootstrap **confidence interval** for the median of the data, as follows. Note that the median and MAD become the focus of this table, rather than the mean and standard deviation.

```
penguins |>
  select(bill_depth_mm, body_mass_g) |>
  describe_distribution(centrality = "median",
                        ci = 0.95, iterations = 500,
                        quartiles = FALSE) |>
  kable(digits = 2)
```

Variable	Median	MAD	IQR	CI_low	CI_high	Min	Max	Skewness	Kurtosis	n	n_Missing
bill_depth_mm	17.3	2.22	3.12	17.02	17.80	13.1	21.5	-0.14	-0.91	342	2
body_mass_g	4050.0	889.56	1206.25	3900.00	4188.12	2700.0	6300.0	0.47	-0.72	342	2

- In estimating the median of bill depth, our sample yields a point estimate of 17.3 mm and a 95% bootstrap confidence interval which extends from 17.02 to 17.80 mm.

Other options for using `describe_distribution()` can be found on its web page.

3.4.10 Finding the most common value (mode)

```
penguins |>
  select(flipper_length_mm) |>
  table()
```

```
flipper_length_mm
172 174 176 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194
  1   1   1   4   1   5   7   3   2   7   9   7  16   6   7  22  13   7  15   5
195 196 197 198 199 200 201 202 203 205 206 207 208 209 210 211 212 213 214 215
 17  10  10   8   6   4   6   4   5   3   1   2   8   5  14   2   7   6   6  12
216 217 218 219 220 221 222 223 224 225 226 228 229 230 231
  8   6   5   5   8   5   6   2   3   4   1   4   2   7   1
```

It appears that 190, which happens 22 times in our data, is the most common value. We can obtain this directly using `distribution_mode()` from the `datawizard` package in `easystats`.

```
distribution_mode(penguins$flipper_length_mm)
```

```
[1] 190
```

Another related estimate to the mode is the highest maximum a posteriori (or MAP) estimate (which indicates the "peak" of a posterior distribution in a Bayesian analysis, as we'll see.)

To obtain a MAP estimate of the mode, we can use the `map_estimate()` function from the `datawizard` package in `easystats`.

```
map_estimate(penguins$flipper_length_mm)
```

```
MAP Estimate

Parameter | MAP_Estimate
------------------------
x         |       191.15
```

3.5 What makes an outlier?

An outlier is a value far away from the center of a distribution, that is deserving of additional attention, in part to ensure that the value is valid and was reliably measured and captured, but also because such outliers can substantially influence our models and summaries.

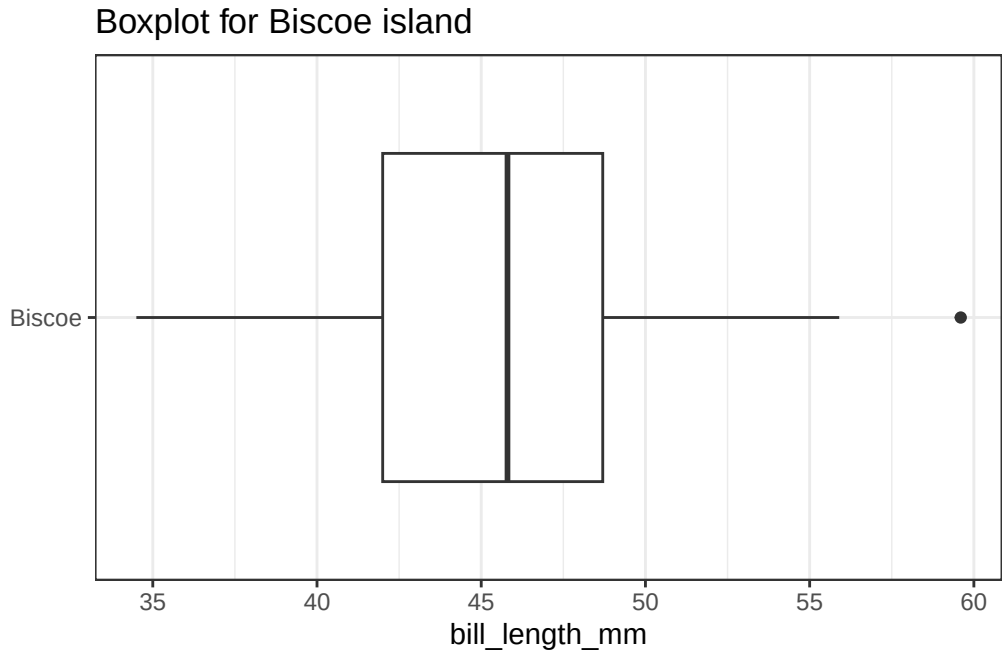
One approach, used, for instance, in the development of boxplots, comes from Tukey (1977), and uses the data's 25th and 75th percentiles, plus the difference between the two: the interquartile range:

- Mark as outliers any values which are outside the range of **Q25 - 1.5 (IQR)** to **Q75 + 1.5 (IQR)**

- Mark as serious outliers (or "far out") any values outside the range of **Q25 - 3 IQR** to **Q75 + 3 IQR**.

For example, consider this boxplot of bill length for penguins coming from the Biscoe island.

```
penguins |>
  filter(complete.cases(island, bill_length_mm)) |>
  filter(island == "Biscoe") |>
  ggplot( aes(x = bill_length_mm, y = "Biscoe") ) +
  geom_boxplot() +
  theme_bw() + labs(y = "", title = "Boxplot for Biscoe island")
```



Note that one value (near 60) is identified by a dot outside of the plot's whiskers. This suggests the point is an outlier (by Tukey's standards.)

Here's the five-number summary for these penguins:

```
dat1 <- penguins |>
  filter(complete.cases(island, bill_length_mm)) |>
  filter(island == "Biscoe")

fivenum(dat1$bill_length_mm)
```

```
[1] 34.5 42.0 45.8 48.7 59.6
```

```
IQR(dat1$bill_length_mm)
```

```
[1] 6.7
```

We see that the minimum value in this cut of our data is 34.5 and the maximum is 59.6.

Our Q25 is 42 and our Q75 is 48.7, so the IQR = 6.7. So, the bounds for outliers extend to all points which are either:

- below Q25 - 1.5 IQR = 42 - 1.5(6.7) = 31.95, or
- above Q75 + 1.5 IQR = 48.7 + 1.5(6.7) = 58.05

And the bounds for serious outliers extend to points:

- below Q25 - 3.0 IQR = 42 - 3(6.7) = 21.9, or
- above Q75 + 3.0 IQR = 48.7 + 3(6.7) = 68.8

Our maximum value is identified by the boxplot as an outlier because it meets the standard Tukey established for being an outlier.

> **Note**
>
> - The values of (Q25 - 1.5 IQR) and (Q75 + 1.5 IQR) are sometimes referred to as the inner fences of the data.
> - (Q25 - 3 IQR) and (Q75 + 3 IQR) are then referred to as the outer fences.

3.6 Describing Categories

3.6.1 Qualitative (Categorical) Variables

Qualitative or categorical variables consist of names of categories. These names may be numerical, but the numbers (or names) are simply codes to identify the groups or categories into which the individuals are divided. Categorical variables with two categories, like yes or no, up or down, or, more generally, 1 and 0, are called **binary** variables. Those with more than two-categories are sometimes called **multi-categorical** variables.

3.6.2 Nominal vs. Ordinal Categories

- When the categories included in a variable are merely names, and come in no particular order, we sometimes call them **nominal** variables. The most important summary of such a variable is usually a table of frequencies, and the mode becomes an important single summary, while the mean and median are essentially useless.

In the `penguins` data, `species` is a nominal variable with multiple unordered categories. So is `island`.

- The alternative categorical variable (where order matters) is called **ordinal**, and includes variables that are sometimes thought of as falling right in between quantitative and qualitative variables. An example in the `penguins` data is the `year` variable.
- Answers to questions like "How is your overall physical health?" with available responses Excellent, Very Good, Good, Fair or Poor, which are often coded as 1-5, certainly provide a perceived *order*, but a group of people with average health status 4 (Very Good) is not necessarily twice as healthy as a group with average health status of 2 (Fair).
- Sometimes we treat the values from ordinal variables as sufficiently scaled to permit us to use quantitative approaches like means, quantiles, and standard deviations to summarize and model the results, and at other times, we'll treat ordinal variables as if they were nominal, with tables and percentages our primary tools.
- Note that all binary variables (like `sex` in the penguins data) may be treated as either ordinal, or nominal.

Lots of variables may be treated as either quantitative or qualitative, depending on how we use them. For instance, we usually think of age as a quantitative variable, but if we simply use age to make the distinction between "child" and "adult" then we are using it to describe categorical information. Just because your variable's values are numbers, don't assume that the information provided is quantitative.

3.6.3 Counting Categories

One of the most important things we can do with any categorical variable, like `species`, `sex` or `island` in our `penguins` data, is to count it.

```
penguins |> count(species)
```

```
# A tibble: 3 x 2
  species       n
  <fct>     <int>
1 Adelie      152
```

```
2 Chinstrap    68
3 Gentoo      124
```

We see we have more Adelie penguins than Gentoo, and fewer Chinstrap than either of the other species. Of our total of 152 + 68 + 124 = 344 penguins in this sample, 152 (44.2%) are Adelie penguins, for example. We also see that each penguin fits into one of these three species.

3.6.4 Species and Island

Do the counts vary meaningfully by `island` on which they were spotted?

```
penguins |> count(species, island)
```

```
# A tibble: 5 x 3
  species   island        n
  <fct>     <fct>     <int>
1 Adelie    Biscoe       44
2 Adelie    Dream        56
3 Adelie    Torgersen    52
4 Chinstrap Dream        68
5 Gentoo    Biscoe      124
```

Yes. Only Adelie penguins are found on all three islands. The Dream island has only Adelie and Chinstrap penguins, while the Biscoe island has only Adelie and Gentoo, with the Torgeson island home to only the Adelie.

3.6.5 Creating a small table

Can we build a table to show the association of, say, species and sex?

```
table(penguins$species, penguins$sex, useNA = "ifany")
```

```
           female male <NA>
  Adelie       73   73    6
  Chinstrap    34   34    0
  Gentoo       58   61    5
```

Since there are some missing values in `sex`, the `table()` function requires us to indicate what we want to do about those missing values.

3.6.6 Using `tabyl()`

A related function that I use frequently for building contingency tables like this is called `tabyl()` which comes from the **janitor** package.

```
penguins |> tabyl(species, island)
```

```
   species Biscoe Dream Torgersen
    Adelie     44    56        52
 Chinstrap      0    68         0
    Gentoo    124     0         0
```

The `tabyl()` package generates tabyls in a way that lets us pull them directly into a tibble, which will be useful down the road. We also can add in a large array of "adorning" features to the result.

Here, we add row and column totals and a title to our tabyl.

```
penguins |>
  tabyl(species, island) |>
  adorn_totals(where = c("row", "col")) |>
  adorn_title() |>
  kable()
```

	island			
species	Biscoe	Dream	Torgersen	Total
Adelie	44	56	52	152
Chinstrap	0	68	0	68
Gentoo	124	0	0	124
Total	168	124	52	344

Next, we create a tabyl which includes both counts and row-wise percentages.

```
penguins |>
  tabyl(species, island) |>
  adorn_percentages(denominator = "row") |>
  adorn_pct_formatting() |>
  adorn_ns(position = "front")
```

```
   species       Biscoe       Dream  Torgersen
    Adelie  44  (28.9%) 56  (36.8%) 52 (34.2%)
 Chinstrap   0   (0.0%) 68 (100.0%)  0  (0.0%)
    Gentoo 124 (100.0%)  0   (0.0%)  0  (0.0%)
```

> **Tip**
>
> More on `tabyl()` can be found in this article and the janitor webpage.

3.6.7 Using `data_tabulate()`

The **datawizard** package provides a function called `data_tabulate()` which produces similar kinds of tables.

```
data_tabulate(penguins$species, penguins$island)
```

```
penguins$species | Biscoe | Dream | Torgersen | <NA> | Total
-----------------+--------+-------+-----------+------+------
Adelie           |     44 |    56 |        52 |    0 |   152
Chinstrap        |      0 |    68 |         0 |    0 |    68
Gentoo           |    124 |     0 |         0 |    0 |   124
<NA>             |      0 |     0 |         0 |    0 |     0
-----------------+--------+-------+-----------+------+------
Total            |    168 |   124 |        52 |    0 |   344
```

Here, we add some column percentages, and drop the column of NA results.

```
data_tabulate(penguins$species, penguins$island,
              remove_na = TRUE, proportions = "col"
)
```

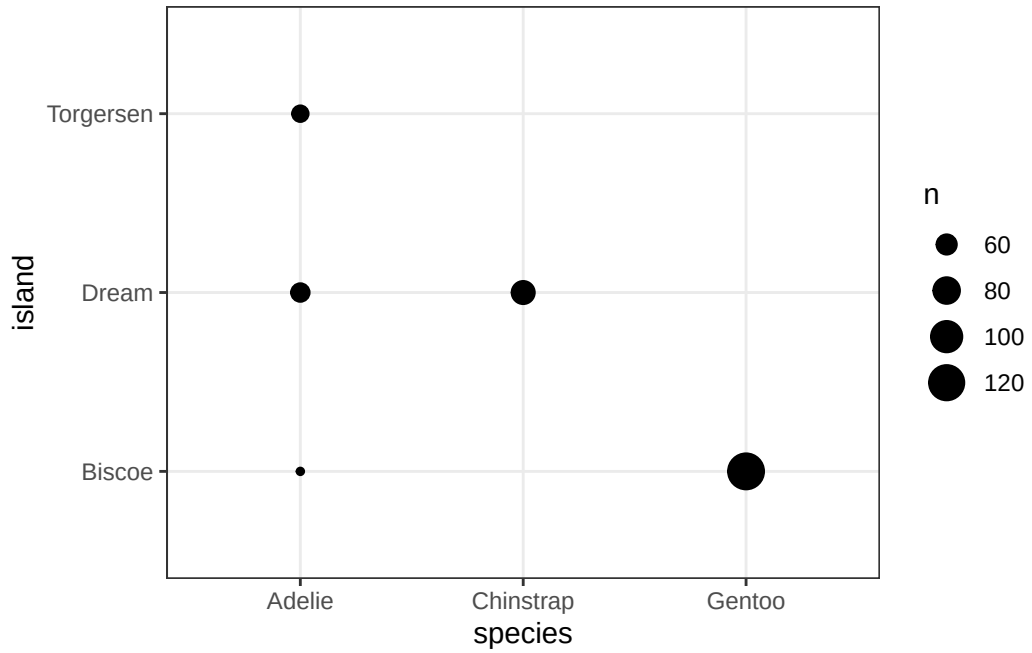
```
penguins$species |      Biscoe |      Dream |   Torgersen | Total
-----------------+-------------+------------+-------------+------
Adelie           |  44 (26.2%) | 56 (45.2%) | 52 (100.0%) |   152
Chinstrap        |   0  (0.0%) | 68 (54.8%) |  0   (0.0%) |    68
Gentoo           | 124 (73.8%) |  0  (0.0%) |  0   (0.0%) |   124
-----------------+-------------+------------+-------------+------
Total            |         168 |        124 |          52 |   344
```

> **Tip**
>
> More on `data_tabulate()` can be found on its web page.

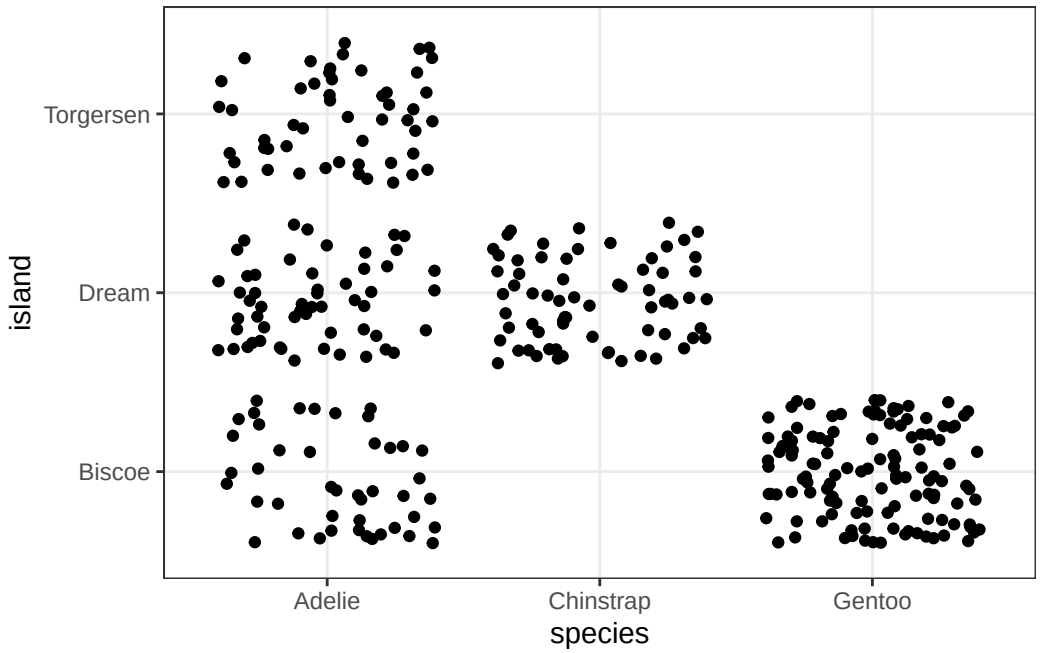
3.6.8 Plotting Counts

The **ggplot** approach to plotting cross-tabular data usually makes use of the `geom_count()` or `geom_jitter()` functions, as follows:

```
ggplot(penguins, aes(x = species, y = island)) +
  geom_count() +
  theme_bw()
```




```
ggplot(penguins, aes(x = species, y = island)) +
  geom_jitter() +
  theme_bw()
```



3.7 For More Information

1. For an introductory example covering some key aspects of data, consider reviewing Chapters 1-3 of Introduction to Modern Statistics, 2nd Edition (Çetinkaya-Rundel and Hardin 2024) which discuss an interesting case study on using stents to prevent strokes.
2. Another great resource on data and measurement issues is Chapter 2 of Gelman, Hill, and Vehtari (2021), available in PDF at this link.
3. Relevant sections of OpenIntro Stats (pdf) include:

- Section 2 on Summarizing Data
- Section 4.3 on the Binomial distribution

4. Visit Appendix E as you like for additional information about statistical formulas relevant to this chapter and the book more generally.
5. Wikipedia on Descriptive Statistics and on Summary Statistics and on Outliers.

4 Data Wrangling

4.1 R setup for this chapter

> **Note**
>
> This section loads all needed R packages for this chapter. Appendix A lists all R packages used in this book, and also provides R session information. Appendix B describes the 431-Love.R script, and demonstrates its use.

```
library(janitor)
library(knitr)
library(naniar)

library(easystats)
library(tidyverse)

source("data/Love-431.R")
```

4.2 Data from a .csv file: Cleveland Neighborhoods

> **Note**
>
> Appendix C provides further guidance on pulling data from other systems into R, while Appendix D gives more information (including download links) for all data sets used in this book.

We will demonstrate ideas in this chapter using a tibble containing information on neighborhoods within the city of Cleveland, Ohio. Here, we ingest the data from a `.csv` (comma-separated version) file called `cle_nbd.csv` into an R tibble we'll call `cle_neigh`, then print the result.

```
cle_neigh <- read_csv("data/cle_nbd.csv",
  show_col_types = FALSE
)

cle_neigh
```

```
# A tibble: 34 x 9
   nbhd_id nbhd_name    pop20 pop10 pop_grow location age65plus income22 unemp22
     <dbl> <chr>        <dbl> <dbl> <chr>    <chr>        <dbl>    <dbl> <chr>
 1       1 BELLAIRE-PU~ 13823 13380 Rise     West         13.8     41324 Middle
 2       2 BROADWAY-SL~ 18854 22331 Fall     Central      11.8     32896 Middle
 3       3 BROOKLYN CE~  8315  8948 Fall     Central      11.8     32308 Middle
 4       4 BUCKEYE SHA~ 11419 12470 Fall     East         22.5     35402 Middle
 5       5 CENTRAL      11955 12306 Fall     Central       7.41    16258 High
 6       6 CLARK FULTON  7625  8509 Fall     Central      10.6     33006 Middle
 7       7 COLLINWOOD ~  9616 11542 Fall     East         13.5     30580 Middle
 8       8 CUDELL        9125  9287 Fall     West         10.3     29417 High
 9       9 CUYAHOGA VA~  1404  1371 Rise     Central       1.54    20266 High
10      10 DETROIT-SHO~ 11285 11577 Fall     Central      12.0     45235 Low
# i 24 more rows
```

```
n_miss(cle_neigh)
```

```
[1] 0
```

Key sources for these data were U.S. Census Bureau (2020) and NEOCANDO (2024). The data in the `cle_neigh` tibble describe each of the 34 neighborhoods (technically statistical planning areas) of the City of Cleveland. For each neighborhood, the `cle_neigh` tibble contains 9 variables, described in the table below.

Variable	Description
nbhd_id	alphabetical order (1-34) of Cleveland neighborhoods
nbhd_name	name of neighborhood
pop20	Total Population, from Decennial Census 2020
pop10	Total Population, from Decennial Census 2010
pop_grow	Rise if Total population increased from 2010 to 2020, else Fall
location	neighborhood's geographic location (East, Central or West)
age65plus	% of residents ages 65 and higher, from ACS
income22	Median household income, from ACS

Variable	Description
unemp22	Unemployment rate as Low (< 10%), Middle (10-15%), High (above 15%), from ACS

> **Note**
>
> - ACS refers to 5-year estimates from the American Community Survey, covering 2018-2022.

4.3 Key Functions for Managing Data

The `dplyr` package, part of the `tidyverse`, provides a set of functions (verbs) which we will use frequently in this course to manage data.

4.3.1 The pipe

There are two main pipes used in R. They are tools for clearer coding that is easier to read, when you are putting multiple steps together into chain.

For example, the code below uses the `|>` pipe:

```
cle_neigh |> head()
```

```
# A tibble: 6 x 9
  nbhd_id nbhd_name     pop20 pop10 pop_grow location age65plus income22 unemp22
    <dbl> <chr>         <dbl> <dbl> <chr>    <chr>        <dbl>    <dbl> <chr>
1       1 BELLAIRE-PUR~ 13823 13380 Rise     West          13.8     41324 Middle
2       2 BROADWAY-SLA~ 18854 22331 Fall     Central       11.8     32896 Middle
3       3 BROOKLYN CEN~  8315  8948 Fall     Central       11.8     32308 Middle
4       4 BUCKEYE SHAK~ 11419 12470 Fall     East          22.5     35402 Middle
5       5 CENTRAL       11955 12306 Fall     Central        7.41    16258 High
6       6 CLARK FULTON   7625  8509 Fall     Central       10.6     33006 Middle
```

applies the `head()` function to the `cle_neigh` data. Specifically, the `|>` pipes the information from the `cle_neigh` data into the function `head()`. As we'll see, we can string a series of pipes together to apply a chain of actions to our data.

The `head()` function simply shows the first 6 rows of the data, while the `tail()` function shows the last 6 rows.

Another, similar, pipe is %>%, which does almost the same thing as |> and is read the same way. Here, we pipe the cle_neigh data into the tail() function, and also ask to show the last 8 rows of the data, instead of the default six.

```
cle_neigh %>% tail(8)
```

```
# A tibble: 8 x 9
  nbhd_id nbhd_name     pop20 pop10 pop_grow location age65plus income22 unemp22
    <dbl> <chr>         <dbl> <dbl> <chr>    <chr>        <dbl>    <dbl> <chr>
1      27 OLD BROOKLYN  32315 32009 Rise     Central      13.3     48061 Middle
2      28 SAINT CLAIR-~  5139  6876 Fall     East         16.7     29612 Middle
3      29 STOCKYARDS     9522 10411 Fall     Central      12.3     33553 Middle
4      30 TREMONT        7699  7975 Fall     Central       9.44    58716 Low
5      31 UNION-MILES ~ 15625 19004 Fall     East         20.1     32788 High
6      32 UNIVERSITY C~  9558  7939 Rise     East         14.9     24780 Low
7      33 WEST BOULEVA~ 18971 18888 Rise     West          9.75    37973 Middle
8      34 WOODLAND HIL~  5625  6678 Fall     East         19.0     25696 Middle
```

Wickham, Çetinkaya-Rundel, and Grolemund (2024) has much more on the pipe in its Pipes section.

4.3.2 select()

We use the select() function to choose specific variables (columns) from our tibble.

```
cle_neigh |>
  select(nbhd_name, pop_grow)
```

```
# A tibble: 34 x 2
   nbhd_name               pop_grow
   <chr>                   <chr>
 1 BELLAIRE-PURITAS        Rise
 2 BROADWAY-SLAVIC VILLAGE Fall
 3 BROOKLYN CENTRE         Fall
 4 BUCKEYE SHAKER          Fall
 5 CENTRAL                 Fall
 6 CLARK FULTON            Fall
 7 COLLINWOOD NOTTINGHAM   Fall
 8 CUDELL                  Fall
 9 CUYAHOGA VALLEY         Rise
10 DETROIT-SHOREWAY        Fall
# i 24 more rows
```

As you can see, this version of the data includes only the two variables we selected.

4.3.3 filter()

In contrast, we use the `filter()` function to choose specific subjects (rows) from our tibble.

```
cle_neigh |>
  filter(pop_grow == "Rise" & location == "West")
```

```
# A tibble: 5 x 9
  nbhd_id nbhd_name     pop20 pop10 pop_grow location age65plus income22 unemp22
    <dbl> <chr>         <dbl> <dbl> <chr>    <chr>        <dbl>    <dbl> <chr>
1       1 BELLAIRE-PUR~ 13823 13380 Rise     West          13.8     41324 Middle
2      12 EDGEWATER      6000  5851 Rise     West          13.5     52709 Low
3      17 HOPKINS         366   215 Rise     West          14.5     29497 Middle
4      19 JEFFERSON     17351 16548 Rise     West          11.3     49908 Low
5      33 WEST BOULEVA~ 18971 18888 Rise     West           9.75    37973 Middle
```

Here, we've selected only those rows that have rising population growth and are located in the West.

We can combine `filter()` and `select()` to choose both specific rows (here, those with incomes over 50,000) and columns (neighborhood ID code, name, income, unemployment and location.)

```
cle_neigh |>
  filter(income22 > 50000) |>
  select(nbhd_id, nbhd_name, income22, unemp22, location)
```

```
# A tibble: 6 x 5
  nbhd_id nbhd_name     income22 unemp22 location
    <dbl> <chr>            <dbl> <chr>   <chr>
1      11 DOWNTOWN         61467 Low     Central
2      12 EDGEWATER        52709 Low     West
3      20 KAMMS CORNERS    63280 Low     West
4      22 LEE-HARVARD      53278 Middle  East
5      26 OHIO CITY        62890 Low     Central
6      30 TREMONT          58716 Low     Central
```

4.3.4 mutate()

We use the `mutate()` function to create new variables that are combinations of the old ones. For instance, here we create a `pop_chg` variable which is the percentage change in population from `pop10` to `pop20`.

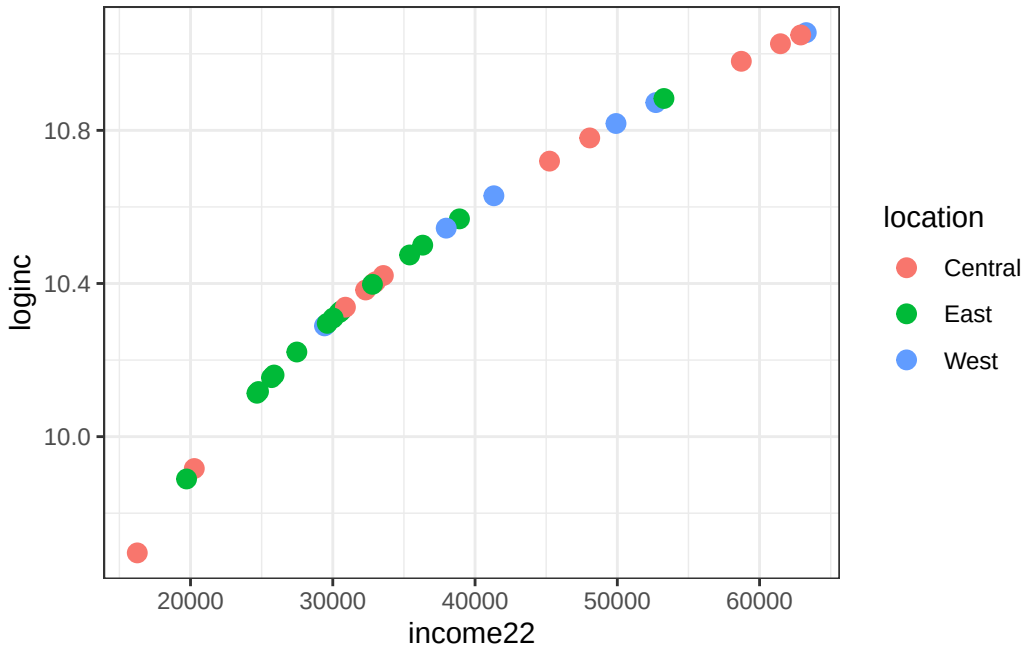
```
cle_neigh |>
  mutate(pop_chg = 100 * (pop20 - pop10) / pop20) |>
  select(nbhd_name, pop20, pop10, pop_grow, pop_chg) |>
  tail(4)
```

```
# A tibble: 4 x 5
  nbhd_name         pop20 pop10 pop_grow pop_chg
  <chr>             <dbl> <dbl> <chr>      <dbl>
1 UNION-MILES PARK  15625 19004 Fall     -21.6
2 UNIVERSITY CIRCLE  9558  7939 Rise      16.9
3 WEST BOULEVARD    18971 18888 Rise       0.438
4 WOODLAND HILLS     5625  6678 Fall     -18.7
```

Here, we use the `tail(4)` function to show the last four rows in this new set of data.

Another common use of `mutate()` is to create transformations of our data, using functions like the square root, inverse or logarithm.

```
cle_neigh |>
  mutate(loginc = log(income22)) |>
  ggplot(aes(x = income22, y = loginc, col = location)) +
  geom_point(size = 3) +
  theme_bw()
```



Note that `log()` is the natural logarithm in R, which we prefer to `log10()` (the base 10 logarithm) for most of our work because coefficients on the natural log scale will be more easily interpreted, as we'll see later.

4.3.5 Creating Factors with `mutate()`

Sometimes we will want to change all of the character variables in a data set into factor variables in R. Factors are used for categorical variables, variables that have a fixed and known set of possible values. They are also useful when you want to display character vectors in a non-alphabetical order. To do so, my favorite approach is:

```
cle_neigh |>
  mutate(across(where(is.character), as_factor))
```

```
# A tibble: 34 x 9
   nbhd_id nbhd_name    pop20 pop10 pop_grow location age65plus income22 unemp22
     <dbl> <fct>        <dbl> <dbl> <fct>    <fct>        <dbl>    <dbl> <fct>
 1       1 BELLAIRE-PU~ 13823 13380 Rise     West          13.8     41324 Middle
 2       2 BROADWAY-SL~ 18854 22331 Fall     Central       11.8     32896 Middle
 3       3 BROOKLYN CE~  8315  8948 Fall     Central       11.8     32308 Middle
 4       4 BUCKEYE SHA~ 11419 12470 Fall     East          22.5     35402 Middle
 5       5 CENTRAL      11955 12306 Fall     Central        7.41    16258 High
```

```
 6       6 CLARK FULTON  7625  8509 Fall     Central      10.6     33006 Middle
 7       7 COLLINWOOD ~  9616 11542 Fall     East         13.5     30580 Middle
 8       8 CUDELL        9125  9287 Fall     West         10.3     29417 High
 9       9 CUYAHOGA VA~  1404  1371 Rise     Central       1.54    20266 High
10      10 DETROIT-SHO~ 11285 11577 Fall     Central      12.0     45235 Low
# i 24 more rows
```

Note that the `nbhd_name`, `pop_grow` and `unemp22` variables are all now factors in R. We'll discuss factors in more detail as the semester moves along.

4.3.6 arrange()

```
cle_neigh |>
  mutate(pop_chg = 100 * (pop20 - pop10) / pop20) |>
  select(nbhd_name, pop20, pop10, pop_grow, pop_chg) |>
  arrange(pop_chg) |>
  head()
```

```
# A tibble: 6 x 5
  nbhd_name             pop20 pop10 pop_grow pop_chg
  <chr>                 <dbl> <dbl> <chr>      <dbl>
1 SAINT CLAIR-SUPERIOR   5139  6876 Fall       -33.8
2 GLENVILLE             21137 27394 Fall       -29.6
3 MOUNT PLEASANT        14015 17320 Fall       -23.6
4 UNION-MILES PARK      15625 19004 Fall       -21.6
5 FAIRFAX                5167  6239 Fall       -20.7
6 COLLINWOOD NOTTINGHAM  9616 11542 Fall       -20.0
```

By default, the arrangement here is sorted from lowest (here, most negative) to highest. To sort from highest to lowest, we could substitute

```
arrange(desc(pop_chg))
```

into the code above.

4.3.7 summarise() and group_by()

```r
cle_neigh |>
  group_by(location) |>
  summarise(
    n = n(), mean = mean(income22), sd = sd(income22),
    med = median(income22), mad = mad(income22)
  ) |>
  kable(digits = 2)
```

location	n	mean	sd	med	mad
Central	12	39628.58	15556.57	33279.5	18509.52
East	15	31038.13	7954.50	30023.0	6415.21
West	7	43444.00	12561.80	41324.0	16879.40

4.4 Describing the Data

Consider the differences between the population in 2020 and the population in 2010 across these neighborhoods.

```r
cle_neigh |>
  reframe(lovedist(pop20 - pop10))
```

```
# A tibble: 1 x 10
      n  miss  mean    sd   med   mad   min    q25   q75   max
  <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>
1    34     0 -719. 1684.  -519  819. -6257 -1052.  70.5  3838
```

4.4.1 Are population changes associated with location?

```r
cle_neigh |>
  reframe(lovedist(pop20 - pop10), .by = location)
```

```
# A tibble: 3 x 11
  location     n  miss   mean    sd    med   mad   min     q25   q75   max
  <chr>    <int> <int>  <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl> <dbl> <dbl>
1 West         7     0   101.  487.   149   436.  -758   -39.5  297    803
2 Central     12     0  -171. 1641.  -288.  696. -3477  -696.   101.  3838
3 East        15     0 -1540. 1774. -1053  1014. -6257 -1857   -698   1619
```

4.4.2 Re-ordering the levels of a factor

```
cle_neigh |>
  tabyl(unemp22)
```

```
 unemp22  n   percent
    High  9 0.2647059
     Low 11 0.3235294
  Middle 14 0.4117647
```

That's not so helpful. We really want to show these categories in a natural order (either High then Middle then Low, or the opposite.) We can use the `forcats` package and the `fct_relevel()` function to help.

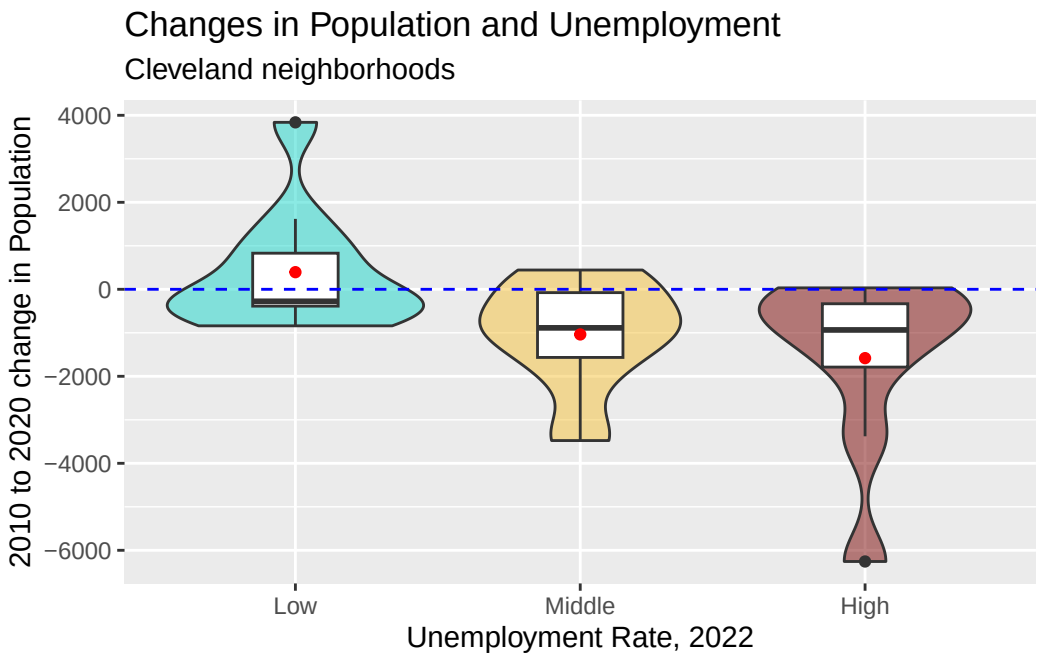
```
cle_neigh <- cle_neigh |>
  mutate(
    unemp22 =
      fct_relevel(unemp22, "Low", "Middle", "High")
  )

cle_neigh |>
  tabyl(unemp22)
```

```
 unemp22  n   percent
     Low 11 0.3235294
  Middle 14 0.4117647
    High  9 0.2647059
```

4.4.3 Plotting three groups

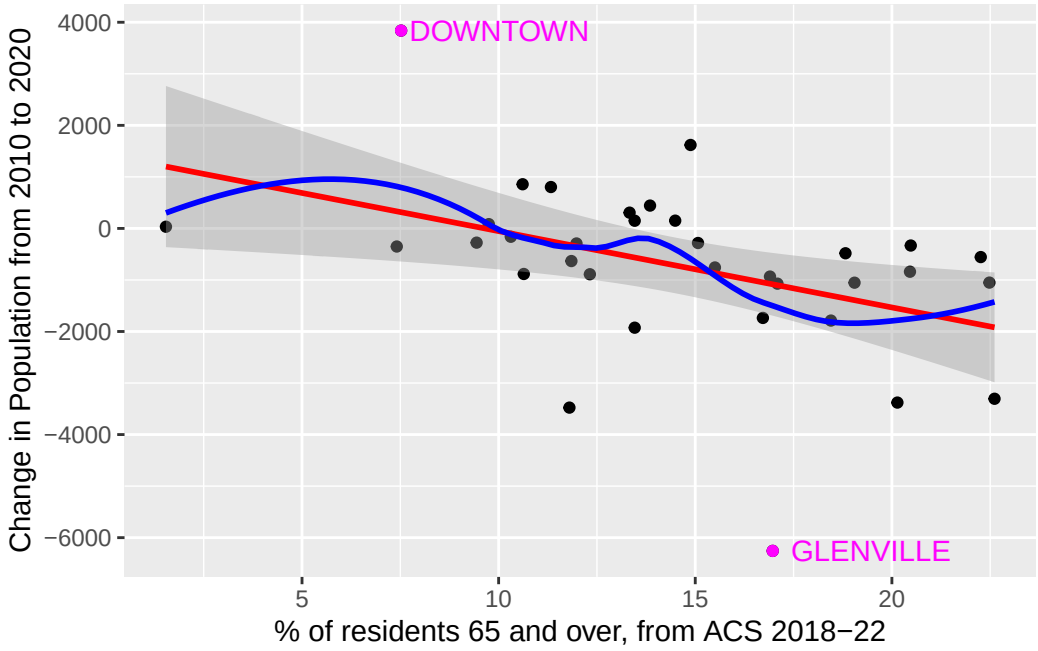
```
cle_neigh <- cle_neigh |>
  mutate(pop_diff = pop20 - pop10)

ggplot(data = cle_neigh, aes(
  x = unemp22, y = pop_diff,
  fill = unemp22
)) +
  geom_violin() +
```

```
  geom_boxplot(fill = "white", width = 0.3) +
  stat_summary(fun = "mean", geom = "point", col = "red") +
  geom_hline(
    yintercept = 0, col = "blue",
    linetype = "dashed"
  ) +
  scale_fill_viridis_d(
    option = "turbo",
    begin = 0.3, alpha = 0.5
  ) +
  guides(fill = "none") +
  labs(
    title = "Changes in Population and Unemployment",
    y = "2010 to 2020 change in Population",
    x = "Unemployment Rate, 2022",
    subtitle = "Cleveland neighborhoods"
  )
```



4.4.4 Association of Population Change and % 65+

```
ggplot(cle_neigh, aes(x = age65plus, y = pop_diff)) +
  geom_point() +
  geom_smooth(
    method = "lm", formula = y ~ x,
    se = TRUE, col = "red"
  ) +
  geom_smooth(
    method = "loess", formula = y ~ x,
    se = FALSE, col = "blue"
  ) +
  geom_text(
    data = cle_neigh |>
      filter(pop_diff > 3000 | pop_diff < -6000),
    aes(label = nbhd_name),
    nudge_x = 2.5, col = "magenta"
  ) +
  geom_point(
    data = cle_neigh |>
      filter(pop_diff > 3000 | pop_diff < -6000),
    col = "magenta"
  ) +
  labs(
    y = "Change in Population from 2010 to 2020",
    x = "% of residents 65 and over, from ACS 2018-22"
  )
```



4.5 Working with Factors

The forcats package is an important piece of the tidyverse, containing many functions which solve common problems with factors. This PDF cheat sheet is available to help you decide which of the `forcats` functions might be most useful to help you solve your current problem

Examples of `forcats` functions that we will use in this course include:

- `fct_recode()`, which lets us change factor levels by hand
- `fct_relevel()`, which lets us reorder factor levels by hand
- `fct_infreq()`, which lets us reorder factor levels by their frequency
- `fct_collapse()`, which lets us collapse factor levels into manually defined groups
- `fct_lump()`, which lets us lump uncommon factor levels together into an "other" level

The best place to learn more about factors is the chapter on factors in Wickham, Çetinkaya-Rundel, and Grolemund (2024).

The best place to start learning more about forcats functions is this Introduction to forcats.

4.6 For More Information

1. The dplyr page is a comprehensive source of descriptions for the tools in the `dplyr` package.
2. The R Graphics Cookbook has an excellent chapter on Getting Your Data into Shape which is well worth your time.
3. R for Data Science (see Wickham, Çetinkaya-Rundel, and Grolemund (2024)) provides an excellent chapter on Data Transformation which goes into further detail on many of the issues discussed here. Other key chapters with something to say about these issues include:

- Workflow: basics
- Code Style and also Data tidying at https://r4ds.hadley.nz/data-tidy
- Factors

4. The `datawizard` package within the easystats eco-system (see Lüdecke et al. (2022)) has many useful functions.

- In particular, the Coming from Tidyverse page provides a nice vignette describing many of the easystats approaches for wrangling data.

5. The `styler` package helps to format code according to the tidyverse style guide, which has some appealing features. I try to use this style throughout this book, to improve readability of the work, and I do so by applying a special add-in within R Studio.

Part II

Comparing Quantities

5 Comparing Paired Samples

5.1 R setup for this chapter

> **Note**
>
> This section loads all needed R packages for this chapter. Appendix A lists all R packages used in this book, and also provides R session information. Appendix B describes the 431-Love.R script, and demonstrates its use.

```
library(infer)
library(knitr)
library(naniar)
library(patchwork)
library(rstanarm)

library(easystats)
library(tidyverse)

source("data/Love-431.R")
theme_set(theme_bw())
```

5.2 Data: Lead in the Blood of Children

> **Note**
>
> Appendix C provides further guidance on pulling data from other systems into R, while Appendix D gives more information (including download links) for all data sets used in this book.

Morton et al. (1982) studied the absorption of lead into the blood of children in a matched-sample study. The "exposed" subjects were 33 children whose parents worked in a battery manufacturing factory (where lead was used) in Oklahoma. Each child with a lead-exposed

parent was then matched to another child of the same age with similar exposure to traffic who lived in the same neighborhood, but whose parents did not work in lead-related industries.

The complete study thus contains 66 children, arranged into 33 matched pairs. A sample of whole blood from each child provides the outcome of interest, which is lead content, measured in mg/dl.

Mainly, we want to estimate the difference in lead content between the exposed and control children, and then use that sample estimate to make inferences about the difference in lead content between the population of all children like those in the exposed group and the population of all children like those in the control group.

The data are available in several places, including the `PairedData` package in R and Pruzek and Helmreich (2009), but here we ingest the `bloodlead.csv` file from our website. A table of the first few pairs of observations (blood lead levels for one child exposed to lead and the matched control) is shown below.

```
bloodlead <- read_csv("data/bloodlead.csv", show_col_types = FALSE)

bloodlead
```

```
# A tibble: 33 x 3
   pair  exposed control
   <chr>   <dbl>   <dbl>
 1 P01        38      16
 2 P02        23      18
 3 P03        41      18
 4 P04        18      24
 5 P05        37      19
 6 P06        36      11
 7 P07        23      10
 8 P08        62      15
 9 P09        31      16
10 P10        34      18
# i 23 more rows
```

So, for instance, the first pair includes an exposed child whose blood lead level was 38 mg/dl and an unexposed (control) child whose blood lead level was 16 mg/dl. Using the `n_miss()` function from the `naniar` package, we see complete data in all 33 rows.

```
n_miss(bloodlead)
```

```
[1] 0
```

5.3 Paired Differences

Since we are interested in comparing the differences between the exposed and control children, we first create a column of these paired differences.

```
bloodlead <- bloodlead |>
  mutate(difference = exposed - control)

bloodlead |> head()
```

```
# A tibble: 6 x 4
  pair  exposed control difference
  <chr>   <dbl>   <dbl>      <dbl>
1 P01        38      16         22
2 P02        23      18          5
3 P03        41      18         23
4 P04        18      24         -6
5 P05        37      19         18
6 P06        36      11         25
```

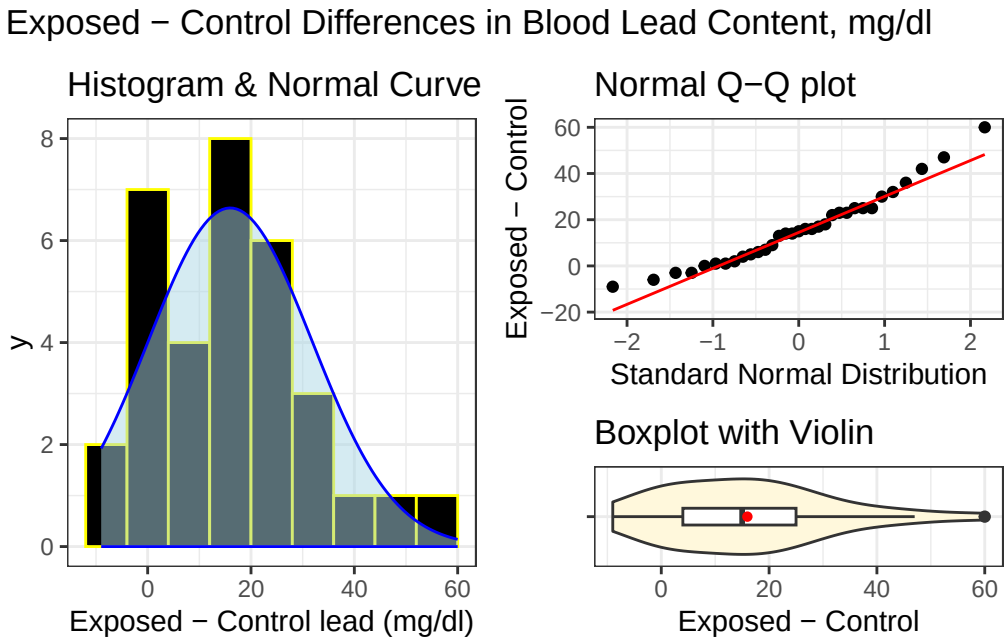
Again, the first pair includes an exposed child with blood lead level of 38 mg/dl and an unexposed (control) child measured at 16 mg/dl. So that paired difference is 38 - 16 = 22 mg/dl.

5.3.1 Visualizing the sample

To help our understanding of the distribution of our paired differences, we might, for instance, prepare a histogram, a boxplot (with violin) or a Normal Q-Q plot. Below, we show all three summaries, using the `patchwork` package to bind together the three individual `ggplot2` plots.

```
bw <- 8 # specify width of bins in histogram

p1 <- ggplot(bloodlead, aes(difference)) +
  geom_histogram(binwidth = bw, fill = "black", col = "yellow" ) +
  stat_function(fun = function(x) {
    dnorm(x, mean = mean(bloodlead$difference, na.rm = TRUE),
          sd = sd(bloodlead$difference, na.rm = TRUE) ) *
        length(bloodlead$difference) * bw},
    geom = "area", alpha = 0.5, fill = "lightblue", col = "blue") +
  labs(x = "Exposed - Control lead (mg/dl)",
```

```
    title = "Histogram & Normal Curve")

p2 <- ggplot(bloodlead, aes(sample = difference)) +
  geom_qq() +
  geom_qq_line(col = "red") +
  labs(y = "Exposed - Control", x = "Standard Normal Distribution",
    title = "Normal Q-Q plot")

p3 <- ggplot(bloodlead, aes(x = difference, y = "")) +
  geom_violin(fill = "cornsilk") +
  geom_boxplot(width = 0.2) +
  stat_summary(fun = mean, geom = "point", shape = 16, col = "red") +
  labs(y = "", x = "Exposed - Control", title = "Boxplot with Violin")

p1 + (p2 / p3 + plot_layout(heights = c(2, 1))) +
  plot_annotation(title = "Exposed - Control Differences in Blood Lead Content, mg/dl")
```



The center of the distribution appears to be around 15 mg/dl according to these plots. The data range from below 0 (around -10) to 60 mg/dl. In terms of shape, each of these plots seems to suggest an approximately Normal distribution. There's a single candidate outlier out to the right of the distribution (at 60 mg/dl), but that doesn't seem to be a major concern at this stage.

5.3.2 Numerical Summaries

Here are a few of the more useful numerical summaries we might consider in developing our understanding of the paired Exposed - Control differences in blood lead levels.

```
bloodlead |>
  reframe(lovedist(difference)) |>
  kable(digits = 2)
```

n	miss	mean	sd	med	mad	min	q25	q75	max
33	0	15.97	15.86	15	14.83	-9	4	25	60

Across these 33 paired differences, the mean and median are within one mg/dl of each other. The standard deviation is near 16 mg/dl and the mad just a bit less than 15 mg/dl. The quartiles match our expectations from the plots.

5.4 Estimating the Mean Difference

Next, we will provide a point estimate and confidence interval for the average difference. Of course, we might consider the average to be the mean or the median (or even something else.) To start, we'll focus on obtaining an estimate and sense of uncertainty around that estimate for the **mean** of the paired differences.

We saw previously that the sample mean of those 33 differences is about 16 mg/dl, and that the distribution of those paired differences in blood lead content are approximately Normally distributed. In light of those findings, how can we put an interval around that estimate that expresses our uncertainty in a reasonable way?

5.4.1 Using a linear model

The most general approach is to fit a linear model to the data. When working with one sample, like these paired differences contained in the `difference` column in the `bloodlead` tibble, we will start by running this model:

```
fit1 <- lm(difference ~ 1, data = bloodlead)
```

Here, the use of the number one indicates to R that we wish to fit a model for `difference` using only an intercept term. What does this model produce?

```
fit1
```

```
Call:
lm(formula = difference ~ 1, data = bloodlead)

Coefficients:
(Intercept)
      15.97
```

The fit here provides an estimate, which turns out to be the sample mean of the paired differences, or 15.97 mg/dl.

To obtain a confidence interval around this estimate, we might consider the `confint()` function.

```
confint(fit1, level = 0.95)
```

```
               2.5 %  97.5 %
(Intercept) 10.34469 21.5947
```

This provides a 95% confidence interval for the mean of the paired differences. We might state this combination of results as follows:

- Our sample of 33 pairs of exposed and control children shows an average difference in blood lead levels of 15.97 mg/dl with a 95% confidence interval ranging from (10.34, 21.59) mg/dl.
- Suppose Harry receives the exposure and Ron does not, but in all other ways they are similar, in the same way that our matched pairs were formed. Then our model suggests that Harry's blood level will be 15.97 mg/dl larger than Ron's.

> **Note**
>
> We might further describe this point and interval estimate combination as a result from a paired-samples comparison, via an intercept-only linear model. The result we have obtained here, as it turns out, is identical that obtained by building a paired t-test for the mean difference.

What if we instead wanted to see a 90% confidence interval for the mean?

```
confint(fit1, level = 0.9)
```

```
                 5 %     95 %
(Intercept) 11.29201 20.64738
```

- Now, our sample of 33 pairs of exposed and control children shows an average difference in blood lead levels of 15.97 mg/dl with a 90% confidence interval ranging from (11.29, 20.65) mg/dl.

> **Note**
>
> - Note that the 90% confidence interval is thinner than the 95% confidence interval. Why do you think that is true?

For more details on our linear model, we might use the `summary()` function in R, as follows:

```
summary(fit1)
```

```
Call:
lm(formula = difference ~ 1, data = bloodlead)

Residuals:
   Min     1Q Median     3Q    Max
-24.97 -11.97  -0.97   9.03  44.03

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   15.970      2.762   5.783 2.04e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 15.86 on 32 degrees of freedom
```

Here's what we find in this summary:

1. A restatement of the fitted model.
2. A five-number summary (minimum, 25th percentile, median, 75th percentile and maximum) of the model residuals (which are the observed - predicted errors made by the model.)

    - So the median of the errors we made in using this model to predict a difference appears to be that our prediction was 0.97 mg/dl too high.

3. Some information about the coefficient of the model, specifically its point estimate (15.97) and its standard error (2.76).
    - Remember we can obtain an approximate 95% confidence interval for this estimate by adding and subtracting two times its standard error.
    - We also see a t statistic, p value (that's the `Pr(>|t|)` piece) and some codes related to statistical significance. I'll ignore all of that, at least until Chapter 15.

4. Finally we see an estimated residual standard error, which is an estimate of an important quality called σ, or sigma.

The `summary()` function after a model is a very common approach in R, and it provides lots of useful information.

However, in this book, we will make heavy use of a few alternative approaches available in the **easystats** ecosystem of packages, specifically `model_parameters()` and `model_performance()`, to obtain and amplify the information we get from `summary()`.

First, let's look at what `model_parameters()` provides…

```
fit1 |>
  model_parameters(ci = 0.95) |>
  kable(digits = 2)
```

Parameter	Coefficient	SE	CI	CI_low	CI_high	t	df_error	p
(Intercept)	15.97	2.76	0.95	10.34	21.59	5.78	32	0

This adds the 95% confidence interval for our estimate, which remains (10.34, 21.59) rounded to two decimal places.

What if we wanted to see a 90% confidence interval, instead of 95%?

```
fit1 |>
  model_parameters(ci = 0.9) |>
  kable(digits = 2)
```

Parameter	Coefficient	SE	CI	CI_low	CI_high	t	df_error	p
(Intercept)	15.97	2.76	0.9	11.29	20.65	5.78	32	0

5.4.2 Using `t.test()`

The one-sample t-test on the paired differences produces the same confidence interval and other summaries as the paired-samples test comparing the exposed to the control directly, as we see below.

```
fit2 <- t.test(bloodlead$difference, conf.level = 0.95)

fit2
```

```

    One Sample t-test

data:  bloodlead$difference
t = 5.783, df = 32, p-value = 2.036e-06
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 10.34469 21.59470
sample estimates:
mean of x
  15.9697
```

Again, we estimate the mean of the paired differences to be 15.97 mg/dl, with 95% confidence interval (10.34, 21.59) mg/dl, just as we did in our regression model.

We can use `model_parameters()` again here to get this information.

```
fit2 |>
  model_parameters() |>
  kable(digits = 2)
```

Parameter	Mean	mu	Difference	CI	CI_low	CI_high	t	df_error	p	Method	Alternative
bloodlead$difference	15.97	0	15.97	0.95	10.34	21.59	5.78	32	0	One Sample t-test	two.sided

We'll ignore the mu, t, df_error and p values shown here for now.

Another identical approach includes both the `exposed` and `control` variables, but specifies a paired samples comparison, like this:

```r
fit3 <- t.test(bloodlead$exposed, bloodlead$control,
  paired = TRUE, conf.level = 0.95
)

fit3
```

```
    Paired t-test

data:  bloodlead$exposed and bloodlead$control
t = 5.783, df = 32, p-value = 2.036e-06
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 10.34469 21.59470
sample estimates:
mean difference
        15.9697
```

Again, we see the same results.

```r
fit3 |>
  model_parameters()
```

```
Paired t-test

Parameter         |             Group | Difference | t(32) |      p |         95% CI
-------------------------------------------------------------------------------------
bloodlead$exposed | bloodlead$control |      15.97 |  5.78 | < .001 | [10.34, 21.59]

Alternative hypothesis: true mean difference is not equal to 0
```

Note that the paired t-test is also equivalent to the result from our linear model.

What if we wanted to see a 90% confidence interval, instead of 95%? Unfortunately, we need to rerun the t test specifying this confidence level in order to get that result.

```r
fit3_90 <- t.test(bloodlead$exposed, bloodlead$control,
  paired = TRUE, conf.level = 0.90
)

fit3_90 |>
  model_parameters(ci = 0.90)
```

```
Paired t-test

Parameter         |             Group | Difference | t(32) |      p |         90% CI
-------------------------------------------------------------------------------------
bloodlead$exposed | bloodlead$control |      15.97 |  5.78 | < .001 | [11.29, 20.65]

Alternative hypothesis: true mean difference is not equal to 0
```

Again, our point estimate remains 15.97 mg/dl but our 90% confidence interval for the mean ranges from 11.29 to 20.65 mg/dl.

So our linear regression model, and our paired t test each produce the same point estimate and 95% confidence interval for the true mean of the paired differences, specifically:

Method	Estimate	95% CI
`lm`: difference ~ 1	15.97	(10.34, 21.59)
paired t test	15.97	(10.34, 21.59)

5.4.3 Estimating Cohen's d statistic

Now, let's consider the value of an effect size measurement, Cohen's *d*, in this situation.

```
cohens_d(difference ~ 1, data = bloodlead, ci = 0.95)
```

```
Cohen's d |       95% CI
------------------------
1.01      | [0.58, 1.42]
```

The Cohen's d calculation for a single sample mean is just the size of that mean, divided by its standard deviation.

Here, we have a sample mean of 15.97, and a sample standard deviation of 15.86, which produces Cohen's d = 15.97/15.86 which is approximately 1.01. The `cohens_d` function also provides a 95% confidence interval for this estimate.

So here, Cohen's d indicates that the paired differences are approximately the same size (on average) as their standard deviation. That's a pretty large gap from 0, as it turns out. The general guidelines for interpreting this effect size suggested by Cohen (1988) are:

- 0.2 indicates a small effect
- 0.5 indicates a moderate effect
- 0.8 indicates a large effect

So our value of 1.01 indicates a large difference from 0, by Cohen's standard.

The **easystats** package enables us to produce a series of effect size indices. Some alternatives and discussion are available on this web page. See Cohen (1988) for further details.

5.5 Bootstrap confidence intervals

5.5.1 Bootstrap CI for the mean

This approach comes from the infer package.

```
set.seed(431)

x_bar <- bloodlead |>
  observe(response = difference, stat = "mean")

res4 <- bloodlead |>
  specify(response = difference) |>
  generate(reps = 1000, type = "bootstrap") |>
  calculate(stat = "mean") |>
  get_confidence_interval(level = 0.95, type = "percentile")

res4 <- res4 |>
  mutate(pt_est = x_bar$stat) |>
  relocate(pt_est)

res4 |> kable(digits = 2)
```

pt_est	lower_ci	upper_ci
15.97	10.79	21.34

Note that changing the seed changes the result:

```
set.seed(12345)

x_bar <- bloodlead |>
  observe(response = difference, stat = "mean")

res4_new1 <- bloodlead |>
```

```
  specify(response = difference) |>
  generate(reps = 1000, type = "bootstrap") |>
  calculate(stat = "mean") |>
  get_confidence_interval(level = 0.95, type = "percentile")

res4_new1 <- res4_new1 |>
  mutate(pt_est = x_bar$stat) |>
  relocate(pt_est)

res4_new1 |> kable(digits = 2)
```

pt_est	lower_ci	upper_ci
15.97	10.85	21.12

Our answer will also change if we change the number of repetitions through the bootstrap process.

```
set.seed(431)

x_bar <- bloodlead |>
  observe(response = difference, stat = "mean")

res4_new2 <- bloodlead |>
  specify(response = difference) |>
  generate(reps = 2000, type = "bootstrap") |>
  calculate(stat = "mean") |>
  get_confidence_interval(level = 0.95, type = "percentile")

res4_new2 <- res4_new2 |>
  mutate(pt_est = x_bar$stat) |>
  relocate(pt_est)

res4_new2 |> kable(digits = 2)
```

pt_est	lower_ci	upper_ci
15.97	10.67	21.27

5.5.2 Bootstrap CI for the median

```
set.seed(431)

x_med <- bloodlead |>
  observe(response = difference, stat = "median")

res5 <- bloodlead |>
  specify(response = difference) |>
  generate(reps = 1000, type = "bootstrap") |>
  calculate(stat = "median") |>
  get_confidence_interval(level = 0.95, type = "percentile")

res5 <- res5 |>
  mutate(pt_est = x_med$stat) |>
  relocate(pt_est)

res5 |> kable(digits = 2)
```

pt_est	lower_ci	upper_ci
15	6.98	23

5.6 Bayesian linear model

Bayesian inference is an excellent choice for virtually every regression model, even when using weakly informative default priors (as we will do), because it yields estimates which are stable, and because it helps us present the uncertainty associated with our estimates in useful ways.

Once the **rstanarm** package is loaded in R, we can run any linear model fit with `lm` as a Bayesian model using `stan_glm()` with no other changes required, except to set a random seed before running the model. Here's an example.

```
set.seed(431)
fit6 <- stan_glm(difference ~ 1, data = bloodlead, refresh = 0)
```

```
fit6
```

```
stan_glm
```

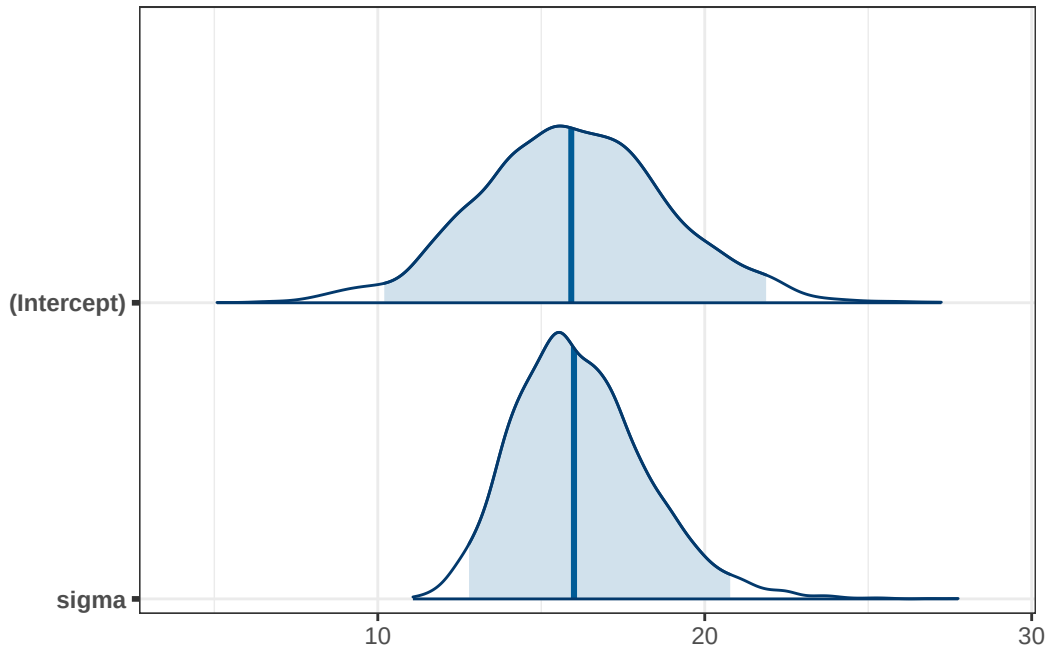
```
 family:       gaussian [identity]
 formula:      difference ~ 1
 observations: 33
 predictors:   1
------
            Median MAD_SD
(Intercept) 15.9    2.8

Auxiliary parameter(s):
      Median MAD_SD
sigma 16.0    2.0

------
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

```
post6 <- describe_posterior(fit6, ci = 0.95)
print_md(post6, digits = 2)
```

Table 5.10: Summary of Posterior Distribution

Parameter	Median	95% CI	pd	ROPE	% in ROPE	Rhat	ESS
(Intercept)	15.91	[10.20, 21.88]	100%	[-1.59, 1.59]	0%	1.000	2481.00

```
plot(fit6, plotfun = "areas", prob = 0.95)
```




```
fit6 |>
  model_parameters() |>
  kable(digits = 2)
```

Parameter	Median CI	CI_low	CI_high	pd	Rhat	ESS	Prior_Distribution	Prior_Location	Prior_Scale
(Intercept)	15.91	0.95	10.2	21.88	1	1	2481.47	normal	15.97	39.66

5.7 Wilcoxon signed rank test

A rank-based procedure called the Wilcoxon signed rank test can also be used to yield a confidence interval statement about the population pseudo-median, a measure of the population distribution's center (but not the population's mean).

Sometimes, a non-parametric alternative is desirable if the data are symmetric (in the sense that the mean and median are close) but show substantial differences in tail behavior from a Normal distribution. In that case, you will often see a report of a Wilcoxon signed rank test, as shown below:

```
wilcox.test(difference ~ 1, data = bloodlead,
  conf.int = TRUE, conf.level = 0.95, exact = FALSE)
```

```
    Wilcoxon signed rank test with continuity correction

data:  difference
V = 499, p-value = 1.155e-05
alternative hypothesis: true location is not equal to 0
95 percent confidence interval:
  9.999943 21.499970
sample estimates:
(pseudo)median
      15.49996
```

As it turns out, if you're willing to assume the population is symmetric (but not necessarily Normally distributed) then the population pseudo-median is actually equal to the population median.

Note that the pseudo-median here (15.5) is actually slightly closer here to the sample mean (15.97) than it is to the sample median (15).

There are two problems with the Wilcoxon signed-rank approach which make it something I use very rarely in practical work.

1. Converting the data to ranks is a strong transformation that loses a lot of the granular information in the data.
2. The parameter estimated here, called the **pseudo-median** is not the same as either the sample mean or sample median, and is thus a challenge to interpret.

As a result, bootstrap approaches, though more computationally intricate, are usually better choices for my work.

> **Note**
>
> The pseudo-median of a particular distribution G is the median of the distribution of (u + v)/2, where both u and v have the same distribution (G).
>
> - If the distribution G is symmetric, then the pseudomedian is equal to the median.
> - If the distribution is skewed, then the pseudomedian is not the same as the median.
> - For any sample, the pseudomedian is defined as the median of all of the midpoints of pairs of observations in the sample.

5.8 Sign test

We can also answer the question: How many of the paired differences (out of the 33 pairs) are positive?

```
bloodlead |> count(exposed > control)
```

```
# A tibble: 2 x 2
  `exposed > control`     n
  <lgl>               <int>
1 FALSE                   5
2 TRUE                   28
```

and here is a reasonably appropriate 95% confidence interval for this proportion.

```
binom.test(x = 28, n = 33, conf.level = 0.95)
```

```

    Exact binomial test

data:  28 and 33
number of successes = 28, number of trials = 33, p-value = 6.619e-05
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.6810102 0.9489113
sample estimates:
probability of success
             0.8484848
```

The confidence interval provided doesn't relate back to our original population means. It's just showing the confidence interval around the probability of the exposed mean being greater than the control mean for a pair of children. We'll return to studying confidence intervals for proportions in Chapter 13.

5.9 Comparing the Results

Method	Point Estimate	95% Uncertainty Interval
Linear fit (`lm`)	$\bar{x} = 15.97$	(10.34, 21.59)
t test	$\bar{x} = 15.97$	(10.34, 21.59)
Bootstrap mean	$\bar{x} = 15.97$	(10.79, 21.34)
Bootstrap median	median(x) = 15	(6.98, 23)
Bayesian fit	$\bar{x} = 15.91$	(10.20, 21.88)
Signed Rank	pseudo-med(x) = 15.5	(10, 21.5)

Do you see large differences here? Is it surprising that the Bayesian fit's estimates are closer to 0 than the linear fit?

5.9.1 Assumptions

1. All approaches assume that the data are independent samples from a distribution of paired differences in blood lead levels.
2. All approaches work better with larger samples.
3. All approaches assume that the paired differences are a random sample from a population or process with a fixed distribution of such differences.
4. The linear fit assumes the distribution of differences can be modeled well by a Normal distribution, as does the equivalent t-test.
5. The Wilcoxon signed rank test assumes a symmetric distribution, but not necessarily a Normal one.
6. The bootstrap has even more minor distributional requirements, essentially requiring only that the mean (or median) be an appropriate choice of summary for the center of the distribution.
7. The Bayesian assumptions also include a set of weakly informative prior distributions for the parameters in the model.

None of the methods dominates all of the rest. We will focus most of our effort in this book on least squares linear models fit with `lm()` and on Bayesian alternatives fit with `stan_glm()`.

5.9.2 General Advice

We have described several approaches to estimating a confidence interval for the center of a distribution of quantitative data.

1. The most commonly used approach uses the t distribution to estimate a confidence interval for a population/process mean. This requires some extra assumptions, most particularly that the underlying distribution of the population values is at least approximately Normally distributed. This is identical to the result we get from an intercept-only linear regression model.

2. A more modern and very general approach uses the idea of the bootstrap to estimate a confidence for a population/process parameter, which could be a mean, median or other summary statistic. The bootstrap, and the underlying notion of resampling is an important idea that lets us avoid some of the assumptions (in particular Normality) that are required by other methods. Bootstrap confidence intervals involve random sampling, so that the actual values obtained will differ a bit across replications.
3. A Bayesian linear model can be used to fit credible intervals, and has some advantages and disadvantages as compared to the intercept-only least squares fit.
4. The Wilcoxon signed-rank method is one of a number of inferential tools which transform the data to their ranks before estimating a confidence interval. This avoids some assumptions, but yields inferences about a less-familiar parameter - the pseudo-median.

Most of the time, the bootstrap provides a reasonably adequate confidence interval estimate of the population value of a parameter (mean or median, most commonly) from a distribution when our data consists of a single sample of quantitative information.

5.10 For More Information

1. The infer package has a nice place to get started at Getting to know infer.
2. Relevant sections of OpenIntro Stats (pdf) include:

- Section 7 - inference for numerical data (except we postpone most of the ideas in section 7.4 until 432.)

6 Comparing Two Groups

6.1 R setup for this chapter

> **Note**
>
> Appendix A lists all R packages used in this book, and also provides R session information. Appendix B describes the 431-Love.R script, and demonstrates its use.

```
library(ggdist)
library(infer)
library(knitr)
library(MKinfer)
library(patchwork)
library(readxl)
library(rstanarm)

library(easystats)
library(tidyverse)

source("data/Love-431.R")
theme_set(theme_bw())
```

6.2 Comparing Two Groups

In making a choice between two alternatives, questions such as the following become paramount.

- Is there a status quo?
- Is there a standard approach?
- What are the costs of incorrect decisions?
- Are such costs balanced?

The process of comparing the means/medians/proportions/rates of the populations represented by two independently obtained samples can be challenging, and such an approach is not always the best choice. Often, specially designed experiments can be more informative at lower cost (i.e. smaller sample size). As one might expect, using these more sophisticated procedures introduces trade-offs, but the costs are typically small relative to the gain in information.

When faced with such a comparison of two alternatives, a test based on **paired** data is often much better than a test based on two distinct, independent samples. Why? If we have done our experiment properly, the pairing lets us eliminate background variation that otherwise hides meaningful differences.

6.3 Data from an Excel (.xlsx) file: Parkinson's Disease Trial

> **Note**
>
> Appendix C provides further guidance on pulling data from other systems into R, while Appendix D gives more information (including download links) for all data sets used in this book.

We will demonstrate ideas in this chapter using a tibble containing information on a **simulated** trial assessing the effect of lixisenatide (as compared to a placebo) on the progression of motor disability in persons with Parkinson's disease. This example is motivated by Meissner et al. (2024), but simplifies the study considerably, and uses **fake** data.

The information we have is gathered in an Excel file called `park_rct.xlsx` within our `data` folder. We will use a function from the `readxl` package in R to help us ingest this information into a tibble called `park_rct`, using the code below.

```
park_rct <- read_xlsx("data/park_rct.xlsx") |>
  mutate(across(where(is.character), as_factor)) |>
  mutate(Person = as.character(Person))

park_rct
```

```
# A tibble: 152 x 5
   Person Treatment    MDS3_base MDS3_12m MDS3_delta
   <chr>  <fct>            <dbl>    <dbl>      <dbl>
 1 SUB001 Placebo             18       19          1
 2 SUB002 Placebo             21       25          4
 3 SUB003 Lixisenatide         0        0          0
```

```
 4 SUB004 Placebo             25       43         18
 5 SUB005 Placebo             22        9        -13
 6 SUB006 Lixisenatide        15        5        -10
 7 SUB007 Lixisenatide        20       30         10
 8 SUB008 Lixisenatide         2        0         -2
 9 SUB009 Placebo             23       32          9
10 SUB010 Lixisenatide        16       16          0
# i 142 more rows
```

Variables in the `park_rct` tibble are described below. A reminder that these are fake (simulated) data.

Variable	Description
Person	participant code: ranges from SUB001 to SUB152
Treatment	Placebo or Lixisenatide
MDS3_base	Baseline MDS-UPDRS part III score
MDS3_12m	MDS-UPDRS part III score after 12 months
MDS3_delta	Change (MDS3_12m minus MDS3_base)

The study aims to assess the effect of lixisenatide (vs. placebo) on the progression of motor disability in persons with early (diagnosed less than 3 years) Parkinson's disease. The primary outcome (end point) of interest for us is the change from baseline in scores on the Movement Disorder Society–Unified Parkinson's Disease Rating Scale (MDS-UPDRS) part III (range, 0 to 132, with higher scores indicating greater motor disability), which was assessed in patients in the on-medication state at 12 months. More details on the MDS-UPDRS are available at Goetz, Christopher G., et al. (2008).

6.4 Key Questions for Comparing with Independent Samples

Can you answer these questions for our study?

1. What is the **population** under study?
2. What is the **sample**? Is it representative of the population?
3. Who are the subjects / **individuals** within the sample?
4. What **data** are available on each individual?

6.4.1 RCT Caveats

The placebo-controlled, double-blind randomized clinical trial, especially if pre-registered, is often considered the best feasible study for assessing the effectiveness of a treatment. While that's not always true, it is a very solid design. The primary caveat is that the patients who are included in such trials are rarely excellent representations of the population of potentially affected patients as a whole.

6.5 Exploratory Data Analysis

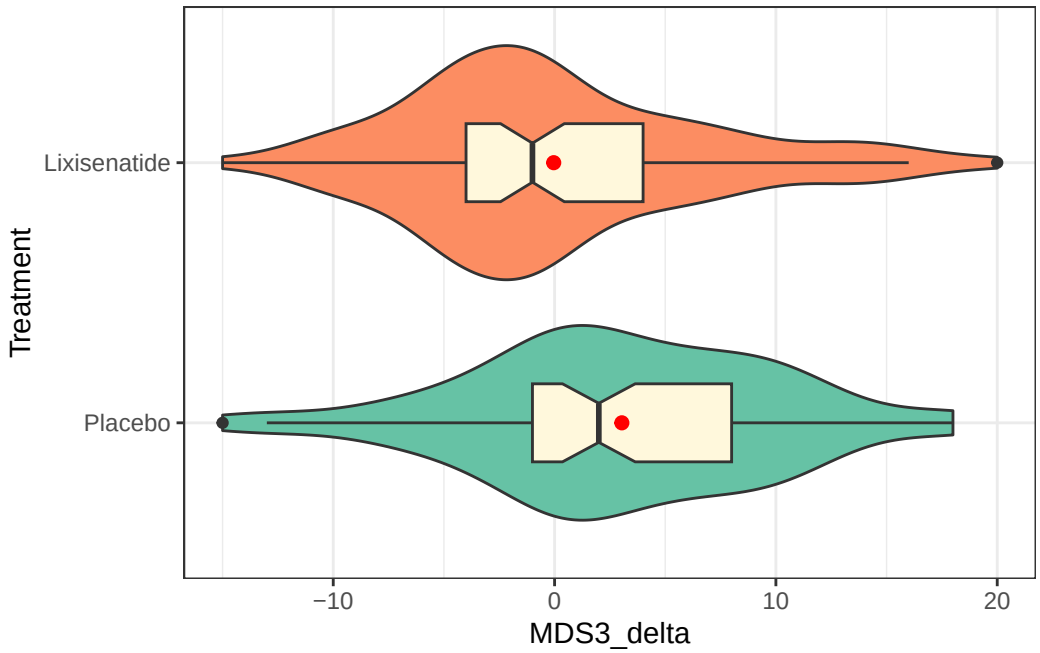
We have two independent samples here, where the placebo patients aren't matched in any way to the Lixisenatide patients. The Lixisenatide group has two more subjects, so the design isn't (quite) balanced (which would require equal sample sizes in each sample) although it's close.

6.5.1 Visualizing Two Independent Samples

A common choice for visualizing the distributions of two independent samples is to fit a comparison boxplot. Here, we'll augment the plot with a violin to show the shape of the distribution, and a red point to indicate the means in each group. We'll also fill in the colors of the violins with two distinct colors from the "Set2" palette in R[1].

```
ggplot(park_rct, aes(y = Treatment, x = MDS3_delta, fill = Treatment)) +
  geom_violin() +
  geom_boxplot(width = 0.3, fill = "cornsilk", notch = TRUE) +
  stat_summary(fun = mean, geom = "point", size = 2, col = "red") +
  scale_fill_brewer(palette = "Set2") +
  guides(fill = "none")
```

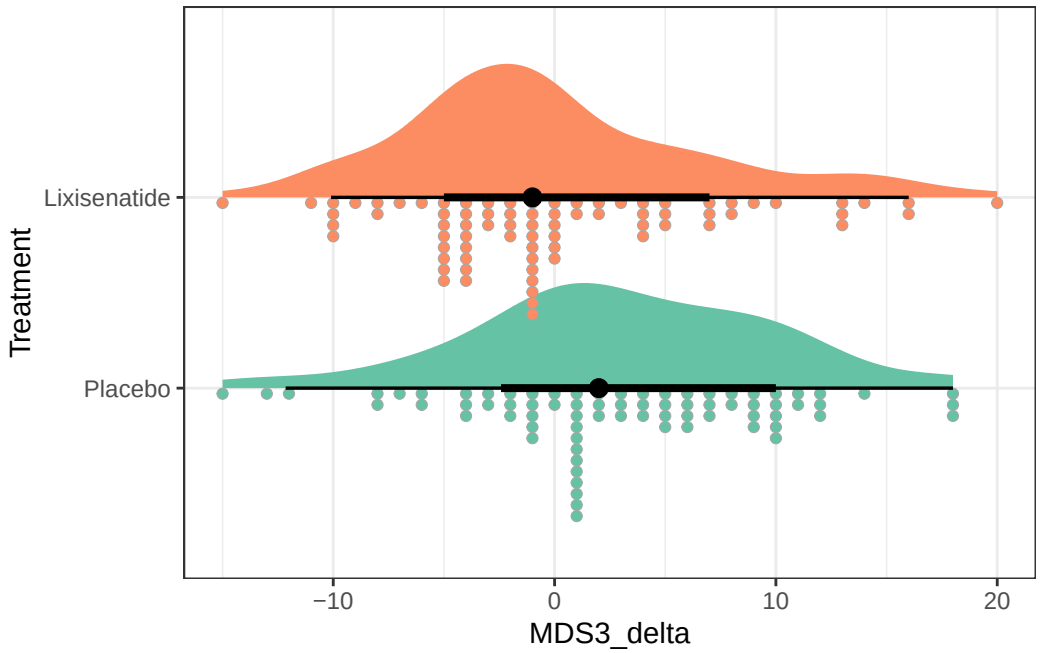
[1]Emil Hvitfeldt provides a nearly comprehensive list of palettes in R here if you're interested.



Another approach we might consider is the use of `stat_slab()` and `slat_dotsinterval()` from the ggdist package to simultaneously display:

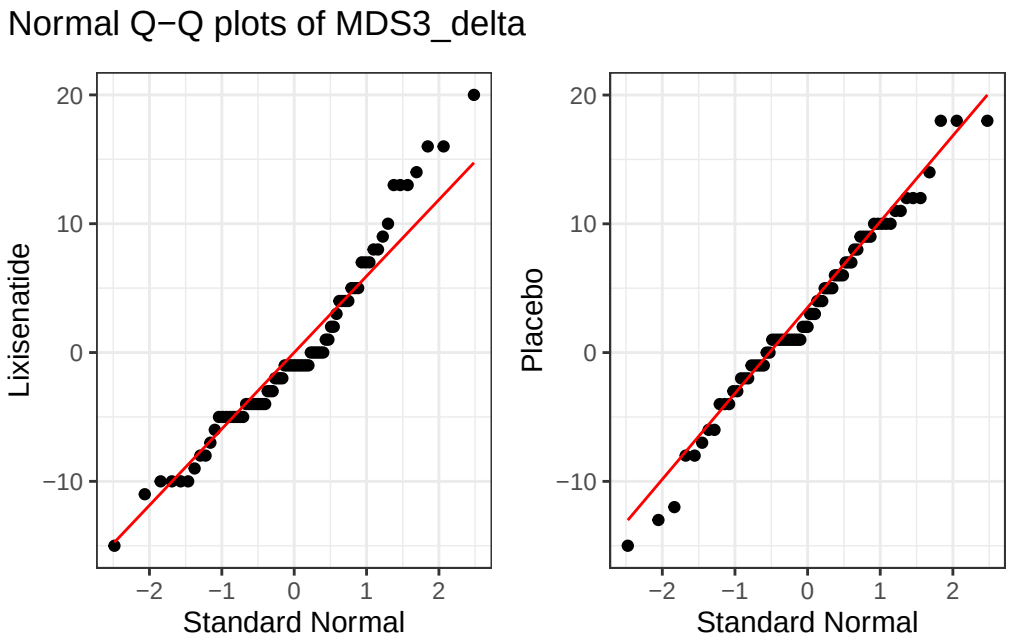
- a smoothed histogram of the data's distribution,
- a rain cloud plot showing the distribution in a dotplot, and
- a line with a point showing the median of the data along with (by default) changes in line size which indicate 66% and 95% intervals for the data.

```
ggplot(park_rct, aes(y = Treatment, x = MDS3_delta, fill = Treatment)) +
  stat_slab(aes(thickness = after_stat(pdf * n)), scale = 0.7) +
  stat_dotsinterval(side = "bottom", scale = 0.7, slab_linewidth = NA) +
  scale_fill_brewer(palette = "Set2") +
  guides(fill = "none") +
  theme_bw()
```



Finally, we show separate Normal Q-Q plots of each of the two treatment groups, joined together using the `patchwork` package into a single figure. Most of the points in each treatment group track with the expectations of a Normal distribution, as indicated by the red line being a reasonably good fit to the data.

```
p1 <- park_rct |>
  filter(Treatment == "Lixisenatide") |>
  ggplot(aes(sample = MDS3_delta)) +
  geom_qq() +
  geom_qq_line(col = "red") +
  labs(y = "Lixisenatide", x = "Standard Normal")

p2 <- park_rct |>
  filter(Treatment == "Placebo") |>
  ggplot(aes(sample = MDS3_delta)) +
  geom_qq() +
  geom_qq_line(col = "red") +
  labs(y = "Placebo", x = "Standard Normal")

p1 + p2 +
  plot_annotation(title = "Normal Q-Q plots of MDS3_delta")
```



6.5.2 Numerical Summaries

```
park_rct |>
  reframe(lovedist(MDS3_delta), .by = Treatment) |>
  kable(digits = 2)
```

Treatment	n	miss	mean	sd	med	mad	min	q25	q75	max
Placebo	75	0	3.04	6.84	2	5.93	-15	-1	8	18
Lixisenatide	77	0	-0.04	6.93	-1	5.93	-15	-4	4	20

We see that the mean change in the placebo group is about 3 points on the scale, indicating somewhat higher motor disability at 12 months than at baseline for those subjects. In the Lixisenatide group, we have a slightly negative mean change very close to zero (the median is -1) indicating that the subjects in that group have (very) slightly lower disability at 12 months than they did at baseline. So the direction of the effect we're seeing appears to be somewhat favorable for Lixisenatide as compared to placebo, at least in terms of the point estimate of the difference in means.

How much uncertainty do we have in those estimates? One way to start thinking about that is to think about the variation within each group. Our measures of spread shown above appear

fairly similar across the two treatment groups, in terms of standard deviations (6.8 vs. 6.9), MAD (both 5.9) and even the interquartile range (IQR).

6.6 Using a linear model

To start, we'll build a 95% confidence interval for the difference in treatment means using a linear model, with the outcome `MDS3_delta` being predicted by the `Treatment` group.

```
fit1 <- lm(MDS3_delta ~ Treatment, data = park_rct)

fit1
```

```

Call:
lm(formula = MDS3_delta ~ Treatment, data = park_rct)

Coefficients:
          (Intercept)  TreatmentLixisenatide
                3.040                -3.079
```

The model equation is MDS3_delta = 3.04 - 3.08 (Treatment = Lixisenatide), which implies that the model estimates that:

- subjects on placebo will have MDS3_delta of 3.04, on average.
- subjects on Lixisenatide will have MDS3_delta of 3.04 - 3.08 = -0.04, on average.

and of course, these are just the sample means in each group.

Here's one more detailed summary of this fit.

```
summary(fit1)
```

```

Call:
lm(formula = MDS3_delta ~ Treatment, data = park_rct)

Residuals:
    Min      1Q  Median      3Q     Max
-18.040  -4.040  -0.961   4.039  20.039

Coefficients:
```

```
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)            3.0400     0.7951   3.824 0.000192 ***
TreatmentLixisenatide -3.0790     1.1171  -2.756 0.006573 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.886 on 150 degrees of freedom
Multiple R-squared:  0.0482,    Adjusted R-squared:  0.04186
F-statistic: 7.597 on 1 and 150 DF,  p-value: 0.006573
```

We'll note that the `Multiple R-squared` statistic indicates that the predictors in this model (here we have only Treatment as a predictor) account for about 4.8% of the variation in our outcome (MDS3_delta) using this linear model.

6.6.1 Obtaining t-based confidence intervals

One way to obtain confidence intervals for the coefficients in our `fit1` model is to use the `confint()` function to obtain confidence intervals using the t distribution.

```
confint(fit1, level = 0.95)
```

```
                          2.5 %     97.5 %
(Intercept)            1.468992  4.6110084
TreatmentLixisenatide -5.286228 -0.8716937
```

The point estimate of the (Lixisenatide - Placebo) treatment effect is -3.08, with 95% CI (-5.29, -0.87).

Another way (and this is my usual approach nowadays) to obtain these results is to use `model_parameters()` as applied to our fitted model, `fit1`.

```
fit1 |>
  model_parameters(ci = 0.95) |>
  kable(digits = 2)
```

Parameter	Coefficient	SE	CI	CI_low	CI_high	t	df_error	p
(Intercept)	3.04	0.80	0.95	1.47	4.61	3.82	150	0.00
TreatmentLixisenatide	-3.08	1.12	0.95	-5.29	-0.87	-2.76	150	0.01

If we like, we can also obtain and present contrasts from the model as follows. In this relatively simple case, all of these approaches give the same conclusions.

```
cons <- estimate_contrasts(fit1, contrast = "Treatment", ci = 0.95)
cons |> kable(digits = 2)
```

Level1	Level2	Difference	CI_low	CI_high	SE	df	t	p
Placebo	Lixisenatide	3.08	0.87	5.29	1.12	150	2.76	0.01

6.6.2 Obtaining bootstrapped confidence intervals

So far, we have produced t-based confidence intervals for the coefficients in model `fit1`. We could also bootstrap the confidence interval for the difference in means shown by our linear model with the following augmentations to the `model_parameters()` command.

```
set.seed(43123)

model_parameters(fit1, bootstrap = TRUE, iterations = 2000,
  ci = 0.95, centrality = "median", ci_method = "quantile")
```

```
Parameter                | Coefficient |         95% CI |      p
-------------------------------------------------------------------
(Intercept)              |        3.03 | [ 1.42,  4.55] | < .001
Treatment [Lixisenatide] |       -3.07 | [-5.21, -0.82] | 0.009


Uncertainty intervals (equal-tailed) are naıve bootstrap intervals.
```

The resulting estimate still describes a difference in means, but it shows the median result of that difference across 2000 bootstrapped samples. We see now that our estimated effect of Lixisenatide as compared to Placebo is -3.07 points on the MDS3_delta scale, with a 95% confidence interval stretching from (-5.21, -0.82). Note, too, that we set a random seed here to allow us to replicate the results.

6.7 t test for Two Independent Samples

6.7.1 Assuming equal population variances

If we assume equal population variances, a t test produces identical results to our linear model `fit1`, as demonstrated below.

```
fit2 <- t.test(MDS3_delta ~ Treatment,
  data = park_rct, var.equal = TRUE)

fit2 |>
  model_parameters(ci = 0.95) |>
  kable(digits = 2)
```

Parameter	Group	Mean_Group1	Mean_Group2	Difference	CI	CI_low	CI_high	t	df_error	p	Method	Alternative
MDS3_delta	Treatment	3.04	-0.04	3.08	0.95	0.87	5.29	2.76	150	0.01	Two Sample t-test	two.sided

6.7.2 Cohen's d assuming equal population variances

We can also produce an estimate of Cohen's d statistic while assuming equal variances (i.e. making use of a pooled t test.)

```
cohens_d(MDS3_delta ~ Treatment, data = park_rct, pooled_sd = TRUE)
```

```
Cohen's d |       95% CI
------------------------
0.45      | [0.12, 0.77]

- Estimated using pooled SD.
```

Cohen's d estimates the standardized difference between the mean `MDS3_delta` scores across the two Treatment groups. Here, we have a value of 0.45 with a 95% CI of (0.12, 0.77).

This Cohen's d indicates that the differences between the means are approximately 45% of the size (on average) as their pooled standard deviation. As we've seen, the general guidelines for interpreting this effect size suggested by Cohen (1988) are:

- 0.2 indicates a small effect
- 0.5 indicates a moderate effect
- 0.8 indicates a large effect

So our observed d = 0.45 indicates a small to moderate difference from 0.

6.7.3 Estimated without pooling the standard deviation

We can also estimate the confidence interval without assuming equal population variances (i.e. without *pooling* the standard deviation) across our two Treatments. This is actually the default `t.test()` approach in R, and the code looks like this:

```
fit3 <- t.test(MDS3_delta ~ Treatment, data = park_rct)

fit3 |>
  model_parameters(ci = 0.95) |>
  kable(digits = 2)
```

Parameter	Group	Mean_Group1	Mean_Group2	Difference	CI	CI_low	CI_high	t	df_error	p	Method	Alternative
MDS3_delta	Treatment	3.04	-0.04	3.08	0.95	0.87	5.29	2.76	149.97	0.01	Welch Two Sample t-test	two.sided

Our new 95% confidence interval using this Welch two-sample t test procedure is (0.87, 5.29) for the size of the Placebo - Lixisenatide difference. The Cohen's d estimate of the standardized difference in means can also be obtained while not assuming a pooled standard deviation using the following code.

```
cohens_d(MDS3_delta ~ Treatment, data = park_rct, pooled_sd = FALSE)
```

```
Cohen's d |       95% CI
------------------------
0.45      | [0.12, 0.77]

- Estimated using un-pooled SD.
```

Note that this result doesn't change materially as compared to the Cohen's d when assuming equal variances in the two Treatment groups and this is because:

1. The sample sizes in the two Treatment groups are quite close, and
2. The sample standard deviations in the two Treatment groups are also quite similar.

6.8 Bayesian linear regression

Let's now fit a linear model to describe the difference in `MDS3_delta` between the two Treatment groups, using a Bayesian approach with the default choice of prior, which is weakly informative.

```
set.seed(431)
fit4 <- stan_glm(MDS3_delta ~ Treatment, data = park_rct, refresh = 0)

## refresh = 0 avoids printing the iteration updates
```

```
post4 <- describe_posterior(fit4, ci = 0.95)

print_md(post4, digits = 2)
```

Table 6.7: Summary of Posterior Distribution

Parameter	Median	95% CI	pd	ROPE	% in ROPE	Rhat	ESS
(Intercept)	3.03	[1.48, 4.54]	100%	[-0.70, 0.70]	0%	0.999	4143.00
TreatmentLixisenatide	-3.08	[-5.30, -0.85]	99.72%	[-0.70, 0.70]	0%	1.000	4181.00

Our estimate of the difference in means between the two treatment groups is -3.08 with a 95% credible interval ranging between (-5.30, -0.85). We note again that this is essentially the same conclusion that we drew from the t test approach.

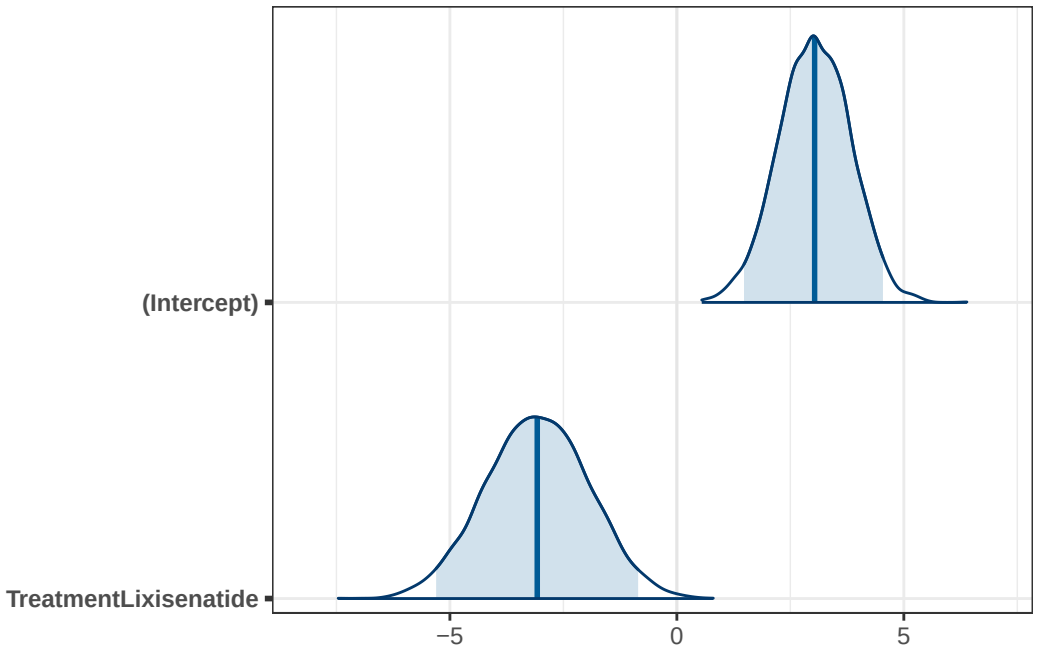
The difference here is that the model produces a credible interval rather than a confidence interval, which changes (a bit) the nature of what we can conclude about that interval.

Specifically, we could now conclude that there is a 95% chance that the true value of the difference in means between the Treatment groups is in the range (-5.30, -0.85), but only **if we were willing to assume that the prior distribution we established for the difference in means was appropriate.**

Moreover, from the `pd` result, we estimate a 99.72% chance that the true difference in means is in the direction we have observed in this model (that Lixisenatide does better than Placebo), and that with a default choice of minimal practical effect size (as shown by the ROPE and % in ROPE materials) that there is almost no chance that the effects is ignorably small.

We can show some graphs of the simulated results from our posterior distribution of our model's coefficients, for example…

```
plot(fit4, plotfun = "areas", prob = 0.95,
  pars = c("(Intercept)", "TreatmentLixisenatide"))
```



And we can summarize the Bayesian fit with the `model_parameters()` command as follows:

```
fit4 |>
  model_parameters() |>
  kable(digits = 2)
```

Parameter	Median	CI	CI_low	CI_high	pd	Rhat	ESS	Prior_Distribution	Prior_Location	Prior_Scale
(Intercept)	3.03	0.95	1.48	4.54	1	1	4143.11	normal	1.48	17.59
TreatmentLixisenatide	-3.08	0.95	-5.30	-0.85	1	1	4180.84	normal	0.00	35.06

The nice part of this last bit of output is that it displays the actual prior distributions assumed for each coefficient (both the intercept and the treatment effect.)

6.9 Using the Bootstrap

6.9.1 Bootstrap CI for Means

We can use the approach available in the `infer` package demonstrated below to obtain a 95% percentile bootstrap confidence interval for the difference in means between the two Treatment groups as follows. Note the need to set a random seed for replicability.

```
set.seed(431)
park_rct |>
  specify(MDS3_delta ~ Treatment) |>
  generate(reps = 1000, type = "bootstrap") |>
  calculate(stat = "diff in means",
    order = c("Placebo", "Lixisenatide") ) |>
  get_confidence_interval(level = 0.95, type = "percentile")
```

```
# A tibble: 1 x 2
  lower_ci upper_ci
     <dbl>    <dbl>
1    0.809     5.36
```

Again, we have a fairly similar estimated confidence interval (0.81, 5.36) to the ones we saw previously.

Another approach to doing essentially the same thing is available through the `boot.t.test()` function in the `MKinfer` package, as follows:

```
set.seed(431)
boot.t.test(MDS3_delta ~ Treatment, var.equal = TRUE, R = 2000,
  data = park_rct, conf.level = 0.95)
```

```
    Bootstrap Two Sample t-test

data:  MDS3_delta by Treatment
number of bootstrap samples:  2000
bootstrap p-value = 0.008
bootstrap difference of means (SE) = 3.102324 (1.134524)
95 percent bootstrap percentile confidence interval:
 0.9155498 5.2608788
```

```
Results without bootstrap:
t = 2.7562, df = 150, p-value = 0.006573
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.8716937 5.2862284
sample estimates:
     mean in group Placebo mean in group Lixisenatide
                3.04000000                -0.03896104
```

If we want to run the bootstrapped t test without assuming equal population variances, we can adapt the `boot.t.test()` function accordingly, like this:

```
set.seed(431)
boot.t.test(MDS3_delta ~ Treatment, var.equal = FALSE, R = 4000,
  data = park_rct, conf.level = 0.95)
```

```
    Bootstrap Welch Two Sample t-test

data:  MDS3_delta by Treatment
number of bootstrap samples:  4000
bootstrap p-value = 0.009
bootstrap difference of means (SE) = 3.074508 (1.107546)
95 percent bootstrap percentile confidence interval:
 0.8585108 5.2140563

Results without bootstrap:
t = 2.7567, df = 149.97, p-value = 0.006564
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.872082 5.285840
sample estimates:
     mean in group Placebo mean in group Lixisenatide
                3.04000000                -0.03896104
```

Again, our results stay similar to what we've seen in the past.

6.9.2 Bootstrap CI for Medians

We might also want to find a bootstrap confidence interval for the difference in **medians** rather than means across the two Treatment groups. We saw earlier that the median `MDS3_delta` in

the Placebo group was +2 and the median in the Lixisenatide group was -1, a difference of 3 points on the `MDS3_delta` scale. The following code from the `infer` package is a reasonable choice to obtain a bootstrap for this difference in medians.

```
set.seed(431)
park_rct |>
  specify(MDS3_delta ~ Treatment) |>
  generate(reps = 2500, type = "bootstrap") |>
  calculate(stat = "diff in medians",
            order = c("Placebo", "Lixisenatide") ) |>
  get_ci(level = 0.95, type = "percentile")
```

```
# A tibble: 1 x 2
  lower_ci upper_ci
     <dbl>    <dbl>
1        2      6.5
```

All of our previous estimates have focused on a difference in means, while this instead finds a difference in medians - and this explains some of the difference we observe.

6.10 Wilcoxon Rank Sum Test

Finally, you may wish to run a Wilcoxon rank sum test. This test compares the locations without using either the mean or the median of the original differences, but instead first ranks the observed `MDS3_delta` values regardless of Treatment group, and then compares the centers (locations) of those distributions. Here's how you could obtain this test (and 95% confidence interval) in R.

```
wilcox.test(MDS3_delta ~ Treatment, data = park_rct, exact = FALSE,
  conf.int = TRUE, conf.level = 0.95)
```

```
    Wilcoxon rank sum test with continuity correction

data:  MDS3_delta by Treatment
W = 3789.5, p-value = 0.0008758
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
 1.999936 5.999975
sample estimates:
```

```
difference in location
              3.999955
```

Of course, these results aren't describing either the difference in means or the difference in medians of the treatment groups on our outcome, so they're meaningfully harder to think about.

6.11 Our Results

In this study, we compared the `MDS3_delta` levels across two Treatment groups (Lixisenatide and Placebo), obtaining a series of 95% uncertainty intervals. Specifically, we found the following results (in each case using "Lixisenatide - Placebo" as the order for our difference, so that negative values (which indicate less motor disability) are good news for the Lixisenatide drug.

Method	Point Est.	95% Uncertainty Interval	What do we estimate?
linear fit with `lm()`	-3.08	(-5.29, -0.87)	diff. in means
bootstrap CI after `lm()` fit	-3.07	(-5.21, -0.82)	diff. in means
pooled t test	-3.08	(-5.29, -0.87)	diff. in means, same as `lm()`
Welch test	-3.08	(-5.29, -0.87)	diff. in means, unpooled std. dev.
Bayesian fit	-3.08	(-5.30, -0.85)	diff. in means, credible interval
percentile bootstrap	-3.08	(-5.36, -0.81)	diff. in means
bootstrap, pooled sd	-3.10	(-5.26, -0.92)	diff. in means
bootstrap, unpooled sd	-3.07	(-5.21, -0.86)	diff. in means
bootstrap, medians	-3	(-6.5, -2)	diff. in *medians*

None of these look wildly different from any of the others, and this is mostly because of the following four reasons:

- the sample size is similar in each Treatment group,
- the sample variance is similar in each Treatment group,

- the difference in means across the two Treatment groups is similar to the difference in medians we observe here,
- the distribution of `MDS3_delta` is fairly similar in each Treatment group, and in both cases is fairly well approximated by a Normal distribution, and
- we used a weakly informative prior distribution in our Bayesian models.

6.12 Paired vs. Independent Samples

One area that consistently trips students up in this course is the thought process involved in distinguishing studies comparing means that should be analyzed using dependent (i.e. paired or matched) samples (as we discussed in Chapter 5) and those which should be analyzed using independent samples (as we discussed in this chapter.)

A paired samples analysis uses additional information about the sample to pair/match subjects receiving the various exposures. That additional information is not part of an independent samples analysis (unpaired testing situation.) The reasons to do this are to (a) increase statistical power, and/or (b) reduce the effect of confounding.

1. In the design of experiments, blocking is the term often used for the process of arranging subjects into groups (blocks) that are similar to one another. Typically, a blocking factor is a source of variability that is not of primary interest to the researcher An example of a blocking factor might be the sex of a patient; by blocking on sex, this source of variability is controlled for, thus leading to greater accuracy.

2. If the sample sizes are not balanced (not equal), the samples must be treated as independent, since there would be no way to precisely link all subjects. So, if we have 10 subjects receiving exposure A and 12 subjects receiving exposure B, a paired samples analysis (such as a paired t test) is not correct.

3. The key element is a meaningful link between each observation in one exposure group and a specific observation in the other exposure group. Given a balanced design, the most common strategy indicating paired samples involves two or more repeated measures on the same subjects. For example, if we are comparing outcomes before and after the application of an exposure, and we have, say, 20 subjects who provide us data both before and after the exposure, then the comparison of results before and after exposure should use a paired samples analysis. The link between the subjects is the subject itself - each exposed subject serves as its own control.

4. The second most common strategy indicating paired samples involves deliberate matching of subjects receiving the two exposures. A matched set of observations (often a pair, but it could be a trio or quartet, etc.) is determined using baseline information and then (if a pair is involved) one subject receives exposure A while the other member of the pair receives exposure B, so that by calculating the paired difference, we learn about the

effect of the exposure, while controlling for the variables made similar across the two subjects by the matching process.

5. In order for a paired samples analysis to be used, we need (a) a link between each observation across the exposure groups based on the way the data were collected, and (b) a consistent measure (with the same units of measurement) so that paired differences can be calculated and interpreted sensibly.

If the samples are collected to facilitate a dependent samples analysis, the correlation of the outcome measurements across the groups will often be moderately strong and positive. If that's the case, then the use of a dependent samples analysis will reduce the effect of baseline differences between the exposure groups, and thus provide a more precise estimate. But even if the correlation is quite small, a dependent samples analysis should provide a more powerful estimate of the impact of the exposure on the outcome than would an independent samples analysis with the same number of observations.

6.13 Summary: Specifying A Two-Sample Study Design

These questions will help specify the details of the study design involved in any comparison of two populations on a quantitative outcome, perhaps with means.

1. What is the outcome under study?
2. What are the (in this case, two) treatment/exposure groups?
3. Were the data collected using matched / paired samples or independent samples?
4. Are the data a random sample from the population(s) of interest? Or is there at least a reasonable argument for generalizing from the sample to the population(s)?
5. What is the confidence level we require here?
6. Are we doing one-sided or two-sided testing/confidence interval generation?
7. If we have paired samples, did pairing help reduce nuisance variation?
    - Also, if we have paired samples, what does the distribution of sample paired differences tell us about which inferential procedure to use?
8. If we have independent samples, what does the distribution of each individual sample tell us about which inferential procedure to use?

6.14 For More Information

1. Chapter 20 of Introduction to Modern Statistics (Çetinkaya-Rundel and Hardin 2024) does a great job discussing the application of point estimates and confidence intervals in the case of two independent samples, using a randomization test, as well as the t-test and bootstrap approaches shown in this chapter.

2. The infer package website has a lot more on methods for performing statistical inference of the type we discuss here, especially the bootstrap approaches.
3. Should you need the formula for the t test comparing two independent samples, I encourage you to look at Wikipedia's page on Student's t test in the two-sample t-tests section.
4. Similarly, you could consider the Wikipedia page on the Wilcoxon signed rank test for an example or two and relevant formulas.
5. The Factors chapter and the Spreadsheets chapter in (Wickham, Çetinkaya-Rundel, and Grolemund 2024) can be helpful in extending your understanding of some of the things I've done in this chapter.
6. The Get Started with Bayesian Analysis vignette from the `easystats` family of packages (Makowski, Ben-Shachar, and Lüdecke 2019) is extremely helpful and includes some nice examples.

7 Transformation

7.1 R setup for this chapter

> **Note**
>
> Appendix A lists all R packages used in this book, and also provides R session information. Appendix B describes the 431-Love.R script, and demonstrates its use.

```
library(car)
library(ggdist)
library(infer)
library(janitor)
library(knitr)
library(MKinfer)
library(naniar)
library(patchwork)
library(rstanarm)

library(easystats)
library(tidyverse)

source("data/Love-431.R")
theme_set(theme_bw())
```

7.2 Data from an .Rds file: DARWIN data

> **Note**
>
> Appendix C provides further guidance on pulling data from other systems into R, while Appendix D gives more information (including download links) for all data sets used in this book.

The DARWIN dataset (Diagnosis AlzheimeR WIth haNdwriting) includes handwriting data specifically designed for the early detection of Alzheimer's disease (AD). These data are found in the UC Irvine Machine Learning Repository and are derived from Cilia et al. (2022). Gathered into our `darwin.Rds` file are times to complete three handwriting tasks sampled from the original protocol of 25 such tasks:

- task 1 (a signature),
- task 3 (joining points with a vertical line four times) and
- task 25 (copying a 110-character paragraph)

The `.Rds` file we have is an R data set, which we can ingest directly with `read_rds()`.

```
darwin <- read_rds("data/darwin.Rds")

darwin
```

```
# A tibble: 174 x 5
   subject status time01 time03 time25
   <chr>   <fct>   <dbl>  <dbl>  <dbl>
 1 S_001   HC       5870  10845  61885
 2 S_002   AD       7840  12260  41990
 3 S_003   AD     156085  20205 159395
 4 S_004   AD       4160   3190  77170
 5 S_005   HC       6870  10400 124760
 6 S_006   AD       5685   5455 252180
 7 S_007   HC       5790   2195  36845
 8 S_008   AD       8445  10735 106760
 9 S_009   AD       9450  24140  72085
10 S_010   AD      17625   8790 122115
# i 164 more rows
```

```
n_miss(darwin)
```

```
[1] 0
```

Our data describe 5 characteristics, including a subject ID (`subject`) for each of 174 study participants. While the units of time are not clearly specified, they seem to be milliseconds, so, for example, the first subject completed task 1 in 5.87 seconds (5870 ms.) We see also that there are no missing values in the `darwin` data.

Note that the `status` variable takes two values:

- **HC** for Healthy Control
- **AD** for subject with Alzheimer's Disease.

```
darwin |> count(status)
```

```
# A tibble: 2 x 2
  status     n
  <fct>  <int>
1 HC        85
2 AD        89
```

In the rest of this chapter, we will explore the comparison of each of the three task times (gathered in `time01`, `time03` and `time 25`) between the subjects with Alzheimer's Disease and the Healthy Controls.

7.3 Visualizing the data for Task 3

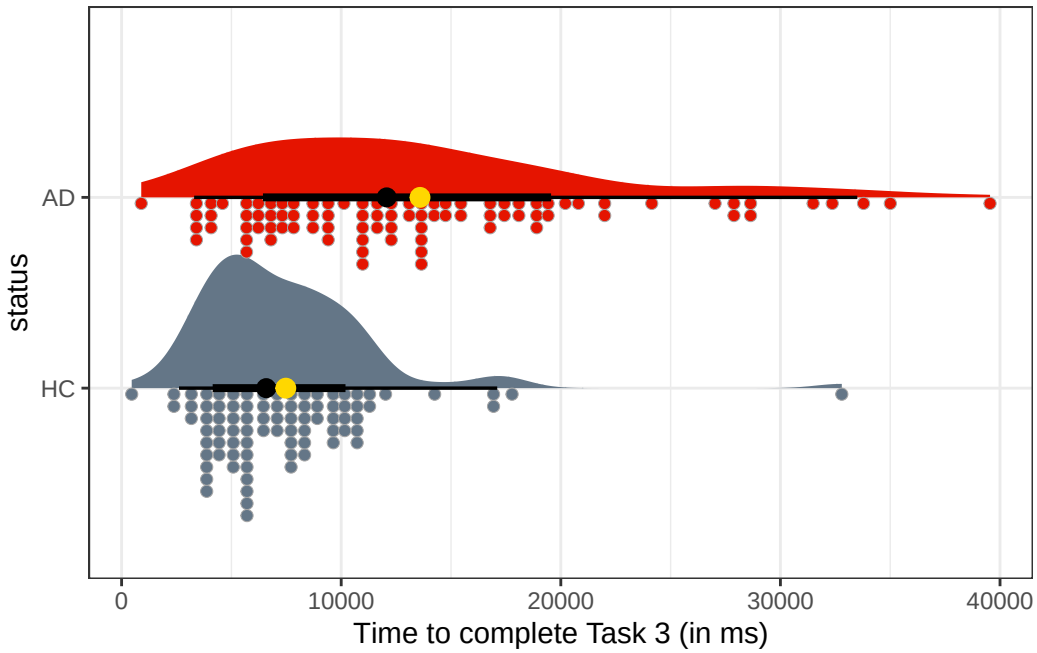
In looking at the data for Task 3, we have two independent samples of an outcome (`time03`) split into two groups by `status`.

```
darwin |>
  reframe(lovedist(time03), .by = status)
```

```
# A tibble: 2 x 11
  status     n  miss   mean    sd   med   mad   min   q25   q75   max
  <fct>  <int> <int>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 HC        85     0  7477. 4297.  6585 3306.   460  4830  9390 32795
2 AD        89     0 13589. 8183. 12070 7502.   895  7180 17605 39545
```

A plot of the data, like the one below, shows some right skew within each group, specifically a few times that are much longer than the rest. Does a comparison of sample means (shown as gold dots) seem particularly appropriate in this setting?

```
ggplot(darwin, aes(y = status, x = time03, fill = status)) +
  stat_slab(aes(thickness = after_stat(pdf * n)), scale = 0.7) +
  stat_dotsinterval(side = "bottom", scale = 0.7, slab_linewidth = NA) +
  stat_summary(fun = mean, geom = "point", size = 3, col = "gold") +
  scale_fill_metro_d() +
  guides(fill = "none") +
  labs(x = "Time to complete Task 3 (in ms)")
```



Our approaches for inference about two independent samples mostly anticipate a more symmetric distribution (with less substantial outliers and with the mean closer to the median) in each sample. Could we transform the data on time to complete Task 3 so as to obtain a comparison which might fit those assumptions?

7.4 Some Linear Transformations

Some commonly used transformations in statistics do not change the shape of our distribution in any meaningful way. These are *linear* transformations, where our transformed data is just a linear function of the original data.

Examples include:

- **centering** the data, by subtracting away its mean, so that the mean value of our transformed data becomes zero, which can be accomplished with the `center()` function from the `datawizard` package in `easystats`.

```
dat1 <- darwin |> mutate(ctime03 = center(time03))

dat1 |>
  reframe(lovedist(time03)) |>
  kable(digits = 2)
```

n	miss	mean	sd	med	mad	min	q25	q75	max
174	0	10603.36	7239.98	8585	4866.63	460	5568.75	13443.75	39545

```
dat1 |>
  reframe(lovedist(ctime03)) |>
  kable(digits = 2)
```

n	miss	mean	sd	med	mad	min	q25	q75	max
174	0	0	7239.98	-2018.36	4866.63	-10143.36	-5034.61	2840.39	28941.64

- **rescaling** the data to a new range, through division by a constant value, which can be accomplished by setting the desired minimum and maximum values with the `rescale()` function from the `datawizard` package in `easystats`.

```
dat1 <- dat1 |> mutate(rstime03 = rescale(time03, to = c(0, 100)))

dat1 |>
  reframe(lovedist(rstime03)) |>
  kable(digits = 2)
```

n	miss	mean	sd	med	mad	min	q25	q75	max
174	0	25.95	18.52	20.79	12.45	0	13.07	33.22	100

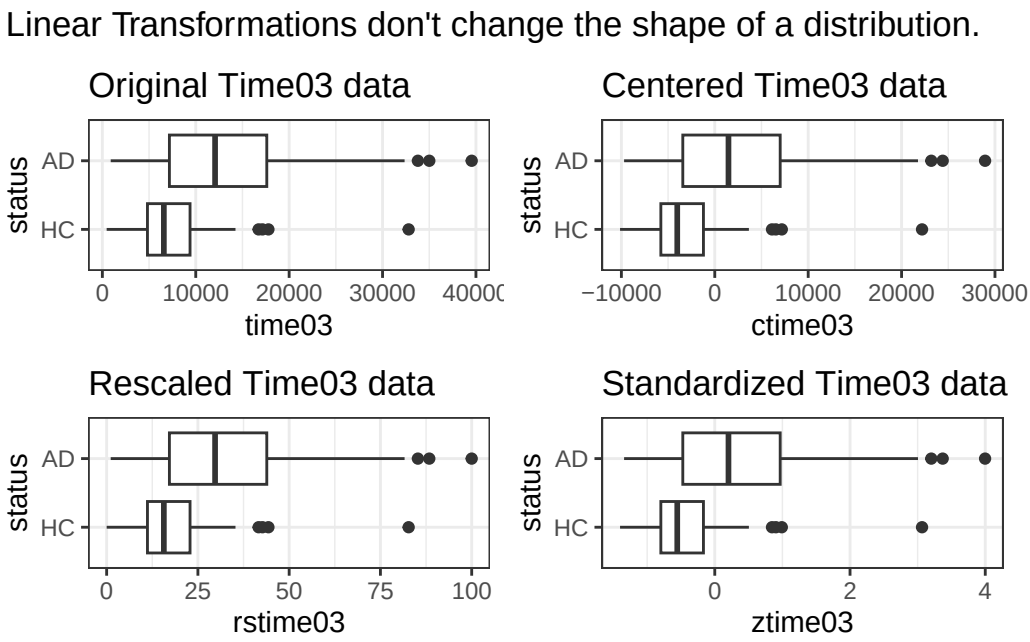
- **standardizing** the data, which involves both subtracting the mean and dividing by the standard deviation, to produce new data with a mean of 0 and a standard deviation of 1 (this is sometimes referred to as Z-scoring or normalization.) Here, we'll use the `standardize()` function from the `datawizard` package in `easystats`.

```
dat1 <- dat1 |> mutate(ztime03 = standardize(time03))

dat1 |>
  reframe(lovedist(ztime03)) |>
  kable(digits = 2)
```

n	miss	mean	sd	med	mad	min	q25	q75	max
174	0	0	1	-0.28	0.67	-1.4	-0.7	0.39	4

Note, however, that none of these linear transformations fix our problem with the *shape* of our distribution.

```
p1 <- ggplot(dat1, aes(x = time03, y = status)) +
  geom_boxplot() +
  labs(title = "Original Time03 data")

p2 <- ggplot(dat1, aes(x = ctime03, y = status)) +
  geom_boxplot() +
  labs(title = "Centered Time03 data")

p3 <- ggplot(dat1, aes(x = rstime03, y = status)) +
  geom_boxplot() +
  labs(title = "Rescaled Time03 data")

p4 <- ggplot(dat1, aes(x = ztime03, y = status)) +
  geom_boxplot() +
  labs(title = "Standardized Time03 data")

(p1 + p2) / (p3 + p4) +
  plot_annotation(title = "Linear Transformations don't change the shape of a distribution.")
```



So these linear transformations may slide the distribution back-and-forth along the x axis, and

may expand or contract the range of that distribution, but they don't address the problems we've had with the shape of the data.

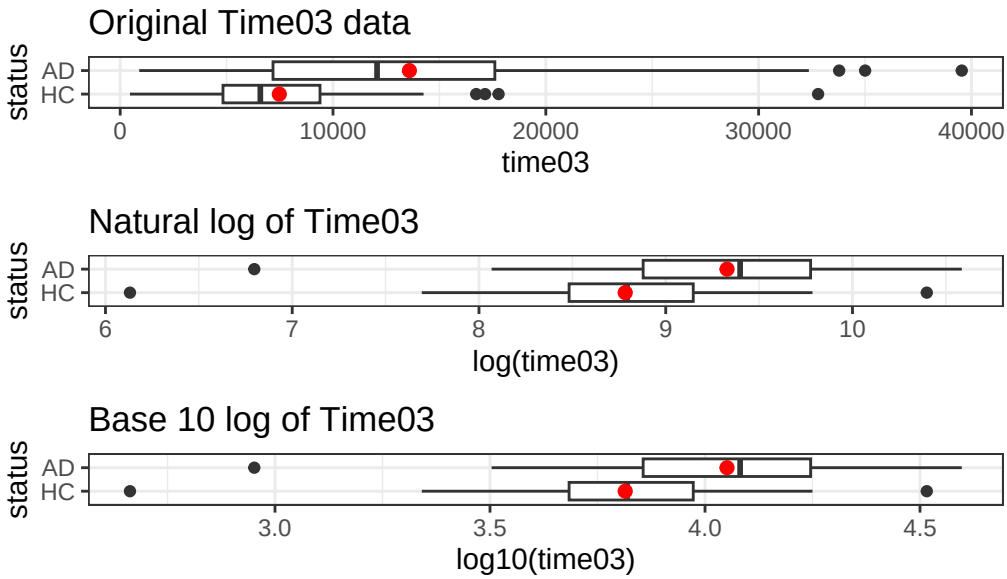
We need to think about some potential transformations that don't just add/subtract or multiply/divide by a constant. We need something non-linear.

7.5 The Logarithmic Transformation

The **logarithm** is one of the more useful non-linear transformations, when our data (as in this case) are entirely positive and show signs of right skew. In R (and in this class), the `log()` function refers to the natural logarithm, with base e, rather than `log10()`, which produces the base 10 logarithm. Either can be used to accomplish the same sort of reshaping of our data.

```
p1 <- ggplot(darwin, aes(x = time03, y = status)) +
  geom_boxplot() +
  stat_summary(geom = "point", fun = "mean", size = 2, col = "red") +
  labs(title = "Original Time03 data")

p2 <- ggplot(darwin, aes(x = log(time03), y = status)) +
  geom_boxplot() +
  stat_summary(geom = "point", fun = "mean", size = 2, col = "red") +
  labs(title = "Natural log of Time03")

p3 <- ggplot(darwin, aes(x = log10(time03), y = status)) +
  geom_boxplot() +
  stat_summary(geom = "point", fun = "mean", size = 2, col = "red") +
  labs(title = "Base 10 log of Time03")

p1 / p2 / p3 +
  plot_annotation(title = "Logarithmic transformations affect the shape of a distribution.")
```

Logarithmic transformations affect the shape of a distribution.



> **Note**
>
> 1. Each of the logged time distributions are more symmetric (much less right-skewed, with the mean closer to the median) than our original data.
> 2. The two logarithmic bases, though they produce different scales, produce the same distributional shapes as each other.

```
darwin |>
  reframe(lovedist(time03), .by = status) |>
  kable(digits = 2)
```

status	n	miss	mean	sd	med	mad	min	q25	q75	max
HC	85	0	7476.88	4297.36	6585	3306.20	460	4830	9390	32795
AD	89	0	13589.33	8182.96	12070	7501.96	895	7180	17605	39545

```
darwin |>
  reframe(lovedist(log(time03)), .by = status) |>
  kable(digits = 2)
```

status	n	miss	mean	sd	med	mad	min	q25	q75	max
HC	85	0	8.78	0.55	8.79	0.52	6.13	8.48	9.15	10.40
AD	89	0	9.33	0.66	9.40	0.66	6.80	8.88	9.78	10.59

```
darwin |>
  reframe(lovedist(log10(time03)), .by = status) |>
  kable(digits = 2)
```

status	n	miss	mean	sd	med	mad	min	q25	q75	max
HC	85	0	3.81	0.24	3.82	0.22	2.66	3.68	3.97	4.52
AD	89	0	4.05	0.29	4.08	0.28	2.95	3.86	4.25	4.60

7.5.1 Importance of the Log Transformation

— from Gelman, Hill, and Vehtari (2021)

> The line $y = a + bx$ can be used to express a more general class of relationships by allowing logarithmic transformations. The formula *log* $y = a + bx$ represents exponential growth (if $b > 0$) *or decline (if* $b < 0$*)*: y = A exp(bx) where A = exp(a). The parameter A is the value of y when x = 0 and the parameters b determines the rate of growth or decline. A one-unit difference in x corresponds to an additive difference of b in log y and thus a multiplicative factor of exp(b) in y.
>
> It is often helpful to model all-positive random variables on the logarithmic scale because it does not allow for values that are 0 or negative. The logarithmic transformation is non-linear and it pulls in the values at the high end, compressing the scale of the distribution.

7.6 Box-Cox to suggest Power Transformations

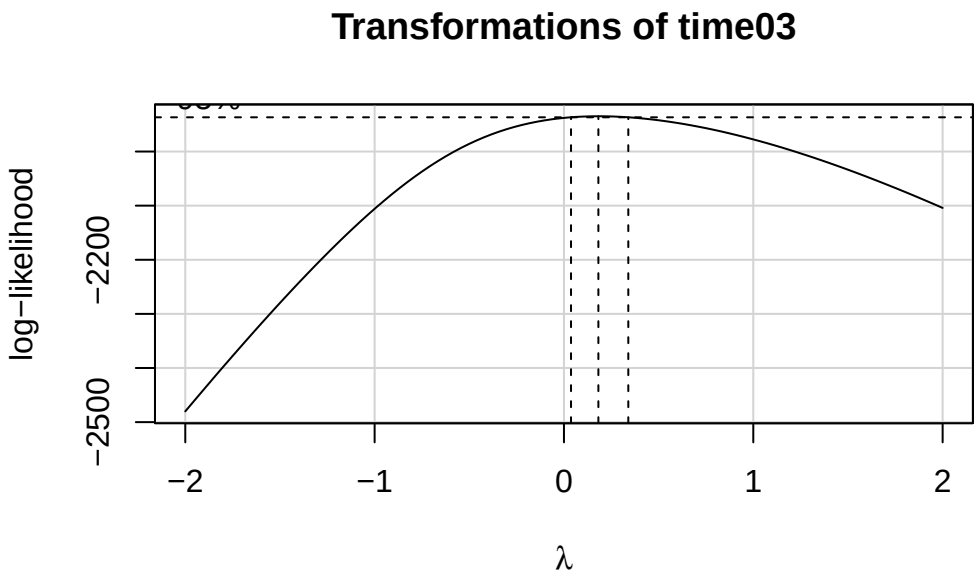
There are several other non-linear transformations we might consider in this situation, including an entire family of power distributions. Tukey's ladder of power transformations can guide our exploration.

Power (λ)	-2	-1	-1/2	0	1/2	1	2
Transformation	$1/y^2$	$1/y$	$1/\sqrt{y}$	log y	$\sqrt{y}$	y	y^2

The **Box-Cox plot**, sifts through the ladder of options to suggest a transformation (for our outcome) to achieve a more Normal distribution within each group, and linearize the outcome-predictor(s) relationship.

Here's how we might fit a Box-Cox plot to our data on Task 3.

```
fit3 <- lm(time03 ~ status, data = darwin)
boxCox(fit3, main = "Transformations of time03")
```



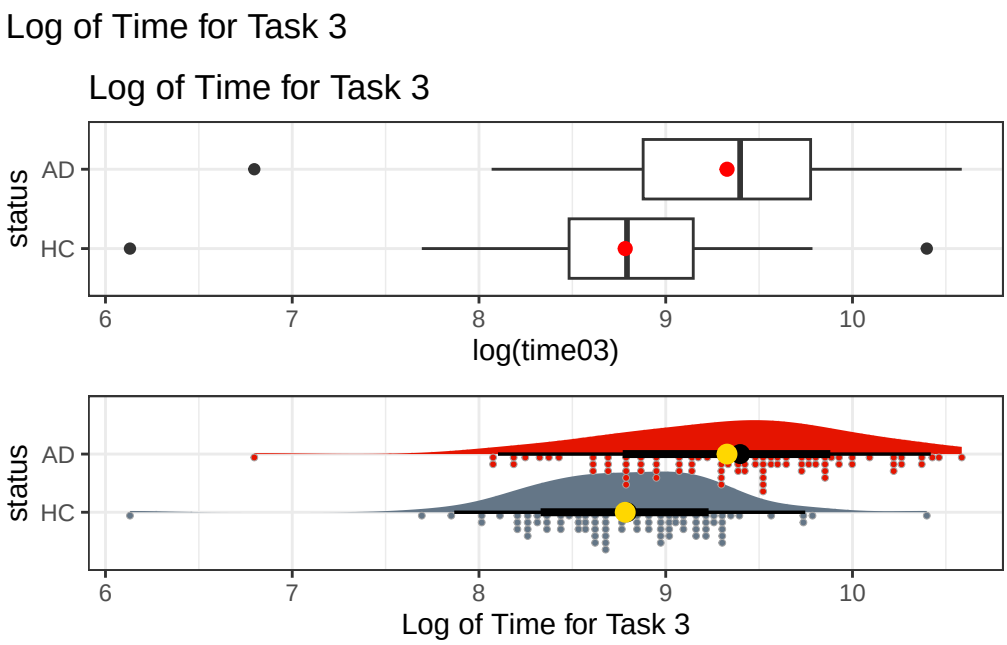

```
summary(powerTransform(fit3))$result
```

```
   Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
Y1 0.1836012        0.33   0.03146328    0.3357392
```

The power transformations which the Box-Cox plots suggests here has a power of 0.33, in other words, the cube root of our outcome. But note that the logarithm (power = 0) and square root (power = 0.5) surround this option, and in the interests of simplicity, I'd like to stick to one of those two options. First, I'll show the log transformation…

```
p1 <- ggplot(darwin, aes(x = log(time03), y = status)) +
  geom_boxplot() +
  stat_summary(geom = "point", fun = "mean", size = 2, col = "red") +
```

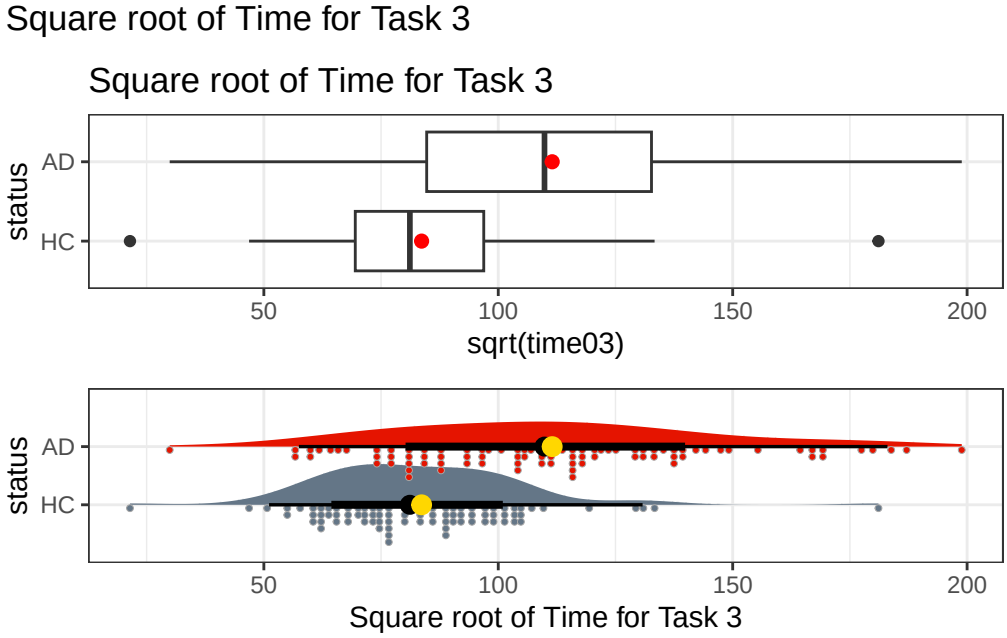
```
  labs(title = "Log of Time for Task 3")

p2 <- ggplot(darwin, aes(y = status, x = log(time03), fill = status)) +
  stat_slab(aes(thickness = after_stat(pdf * n)), scale = 0.7) +
  stat_dotsinterval(side = "bottom", scale = 0.7, slab_linewidth = NA) +
  stat_summary(fun = mean, geom = "point", size = 3, col = "gold") +
  scale_fill_metro_d() +
  guides(fill = "none") +
  labs(x = "Log of Time for Task 3")

p1 / p2 +
  plot_annotation(title = "Log of Time for Task 3")
```



Now, let's look at the square root transformation instead of the logarithm. Is this a meaningful improvement in terms of adherence to the assumptions required for making a comparison about the means?

```
p3 <- ggplot(darwin, aes(x = sqrt(time03), y = status)) +
  geom_boxplot() +
  stat_summary(geom = "point", fun = "mean", size = 2, col = "red") +
  labs(title = "Square root of Time for Task 3")
```

```
p4 <- ggplot(darwin, aes(y = status, x = sqrt(time03), fill = status)) +
  stat_slab(aes(thickness = after_stat(pdf * n)), scale = 0.7) +
  stat_dotsinterval(side = "bottom", scale = 0.7, slab_linewidth = NA) +
  stat_summary(fun = mean, geom = "point", size = 3, col = "gold") +
  scale_fill_metro_d() +
  guides(fill = "none") +
  labs(x = "Square root of Time for Task 3")

p3 / p4 +
  plot_annotation(title = "Square root of Time for Task 3")
```



It seems that either transformation (the square root or the logarithm) can do a pretty good job of generating distributions for the two status groups (AD and HC) which are less skewed, and have fewer outliers. I'll opt to use the logarithm here, because it is such a common choice for transformations in regression, but the square root would also work well in this specific instance.

7.6.1 A Few Caveats

1. Some of these transformations (like the logarithm) require the data to be positive. We can rescale the Y data by adding a constant to every observation in a data set without changing shape.

2. We can use a natural logarithm (`log` in R), a base 10 logarithm (`log10`) or even sometimes a base 2 logarithm (`log2`) to good effect in Tukey's ladder. All affect the association's shape in the same way, so we'll stick with `log` (base e).
3. Some re-expressions don't lead to easily interpretable results. Not many things that make sense in their original units also make sense in inverse square roots. There are times when we won't care, but often, we will.
4. If our primary interest is in making predictions, we'll generally be more interested in getting good predictions back on the original scale, and we can back-transform the point and interval estimates to accomplish this.

7.7 Making Inferences about Task 3

So, we'll transform our Task 3 times by taking their logarithms.

```
darwin <- darwin |> mutate(logtime03 = log(time03))
```

This produces means in the two exposure groups which are 8.8 and 9.3 for the HC and AD groups, respectively, with standard deviations of 0.55 and 0.66, respectively. It's appealing to be comparing transformed values which are between 0 and 100 in practice, and so we'll look to build transformations which accomplish this aim.

```
darwin |> group_by(status) |> reframe(lovedist(logtime03))
```

```
# A tibble: 2 x 11
  status     n  miss  mean    sd   med   mad   min   q25   q75   max
  <fct>  <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 HC        85     0  8.78 0.551  8.79 0.516  6.13  8.48  9.15  10.4
2 AD        89     0  9.33 0.656  9.40 0.656  6.80  8.88  9.78  10.6
```

7.8 Linear Model and Ordinary Least Squares

To start, we'll use ordinary least squares and a model fit with `lm()` to compare the logged times to complete Task 3 across the two `status` groups.

```
fit3 <- lm(logtime03 ~ status, data = darwin)

fit3 |> model_parameters(ci = 0.95) |>
  kable(digits = 2)
```

Parameter	Coefficient	SE	CI	CI_low	CI_high	t	df_error	p
(Intercept)	8.78	0.07	0.95	8.65	8.91	133.37	172	0
statusAD	0.55	0.09	0.95	0.36	0.73	5.92	172	0

```
estimate_contrasts(fit3, ci = 0.95, contrast = "status")
```

```
Marginal Contrasts Analysis

Level1 | Level2 | Difference |         95% CI |   SE | t(172) |      p
-------------------------------------------------------------------------
HC     |     AD |      -0.55 | [-0.73, -0.36] | 0.09 |  -5.92 | < .001

Marginal contrasts estimated at status
p-value adjustment method: Holm (1979)
```

The AD group has a higher (log time) by about 0.55 with a 95% uncertainty interval of (0.36, 0.73). This transformed time value is no longer expressed in milliseconds, of course, since we're now talking about a logarithm of time, rather than our original Task 3 time.

7.8.1 Back-transforming the model's predictions

Your model holds in the "transformed data" world. We cannot back-transform the regression coefficients in a rational way.

After fitting a model, it is useful to generate model-based estimates of the outcomes for different combinations of predictor values, and we can back-transform these predictions to get back to our original scale, as follows.

7.8.1.1 Predictions for The Average Across Many Subjects

If we consider the entire HC group, we would predict that the average log(time03) would be what, exactly?

```
estimate_expectation(fit3, data = "grid", ci = 0.95)
```

```
Model-based Expectation

status | Predicted |   SE |       95% CI
----------------------------------------
```

```
HC     |      8.78 | 0.07 | [8.65, 8.91]
AD     |      9.33 | 0.06 | [9.20, 9.46]

Variable predicted: logtime03
Predictors modulated: status
```

The model `fit3` and the `estimate_expectation()` function can be used to generate expectations for `logtime03`, which we can then exponentiate to obtain…

- our expectation in terms of the average time to complete Task 3 in milliseconds across all HC subjects, which is 6502 ms (exp(8.78)), with 95% uncertainty interval (5710, 7406) (from exp(8.65), exp(8.91).)

> Using R as a calculator
>
> Here, we use R as a calculator…
>
> ```
> round_half_up(exp(c(8.78, 8.65, 8.91)),0)
> ```
>
> ```
> [1] 6503 5710 7406
> ```
>
> Note that the `round_half_up()` function comes from the `janitor` package and is an effective way of rounding results for presentation. More on this function is available on the janitor reference page.

- our expectation in terms of the average time to complete Task 3 in milliseconds across all AD subjects turns out to be 11271 ms, with 95% uncertainty interval (9897, 12836).

> Again, using R as a calculator
>
> ```
> round_half_up(exp(c(9.33, 9.20, 9.46)),0)
> ```
>
> ```
> [1] 11271  9897 12836
> ```

7.8.1.2 Predictions for Individual Subjects

We can also make predictions for individual subjects with HC or AD, using the `estimate_prediction()` function applied to our regression model `fit3`.

```
estimate_prediction(fit3, data = "grid", ci = 0.95)
```

```
Model-based Prediction

status | Predicted |   SE |        95% CI
----------------------------------------
HC     |      8.78 | 0.61 | [7.58,  9.99]
AD     |      9.33 | 0.61 | [8.12, 10.53]

Variable predicted: logtime03
Predictors modulated: status
```

- our prediction of the time an individual HC subject would need to complete Task 3 (in milliseconds) turns out to be 6503 ms, with 95% uncertainty interval (1959, 21807).

```
round_half_up(exp(c(8.78, 7.58, 9.99)),0)
```

```
[1]  6503  1959 21807
```

- our prediction of the time an individual AD subject would need to complete Task 3 (in milliseconds) turns out to be 11271 ms, with 95% uncertainty interval (3361, 37421).

```
round_half_up(exp(c(9.33, 8.12, 10.53)),0)
```

```
[1] 11271  3361 37421
```

Note how much wider the uncertainty intervals are for individual subjects than for the average across all subjects with a certain status (HC or AD.)

7.8.2 Bootstrap CI for Linear Model Coefficients

Rather than using an OLS fit to obtain the t-distribution based confidence interval for our linear model, we could also bootstrap the confidence intervals in this model with …

```
model_parameters(fit3, bootstrap = TRUE, iterations = 2000,
  ci = 0.95, centrality = "median", ci_method = "quantile")
```

```
Parameter   | Coefficient |       95% CI |      p
-------------------------------------------------
(Intercept) |        8.78 | [8.66, 8.89] | < .001
status [AD] |        0.55 | [0.37, 0.73] | < .001


Uncertainty intervals (equal-tailed) are naıve bootstrap intervals.
```

7.9 Other approaches

7.9.1 t test (Welch - not pooling SD)

If we run R's default t test, without assuming equal population variances, on our logged times, we obtain the following, and we'll also add a calculation of a standardized difference (Cohen's d) for the effect size (difference in means on the log scale).

```
fit3a <- t.test(logtime03 ~ status, data = darwin)

fit3a |> model_parameters(ci = 0.95)
```

```
Welch Two Sample t-test

Parameter |  Group | status = HC | status = AD | Difference |         95% CI | t(169.24) |
-------------------------------------------------------------------------------------------
logtime03 | status |        8.78 |        9.33 |      -0.55 | [-0.73, -0.36] |     -5.94 | <

Alternative hypothesis: true difference in means between group HC and group AD is not equal t
```

```
cohens_d(logtime03 ~ status, data = darwin, pooled_sd = FALSE)
```

```
Cohen's d |         95% CI
--------------------------
-0.90     | [-1.21, -0.59]

- Estimated using un-pooled SD.
```

7.9.2 t test (with pooled SD)

The t test on our logged times assuming equal variances in the two groups produces the same result as our linear model fit with `lm()`, of course. Here, though, we can also add a calculation of a standardized difference for the effect size on the log scale.

```
fit3b <- t.test(logtime03 ~ status, data = darwin, var.equal = TRUE)

fit3b |> model_parameters(ci = 0.95)
```

```
Two Sample t-test

Parameter |  Group | status = HC | status = AD | Difference |         95% CI | t(172) |
-------------------------------------------------------------------------------------
logtime03 | status |        8.78 |        9.33 |      -0.55 | [-0.73, -0.36] |  -5.92 | < .0

Alternative hypothesis: true difference in means between group HC and group AD is not equal
```

```
cohens_d(logtime03 ~ status, data = darwin, pooled_sd = TRUE)
```

```
Cohen's d |         95% CI
--------------------------
-0.90     | [-1.21, -0.58]

- Estimated using pooled SD.
```

Because the two groups (HC and AD) have very similar sample sizes and, after transformation, similar standard deviations (and thus similar variances, since the variance is just the square of the standard deviation), it really doesn't matter whether or not we assume equal variances for these data.

7.9.3 Wilcoxon Signed Rank

We could, instead of transforming the data with the logarithm, use a non-parametric approach to compare the distributions (although this doesn't compare the means) such as a Wilcoxon signed rank test.

```
wilcox.test(time03 ~ status, data = darwin,
            conf.int = TRUE, conf.level = 0.95)
```

```
    Wilcoxon rank sum test with continuity correction

data:  time03 by status
W = 1829, p-value = 4.105e-09
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
 -6730 -3190
sample estimates:
difference in location
                 -4950
```

7.9.4 Bayesian linear model

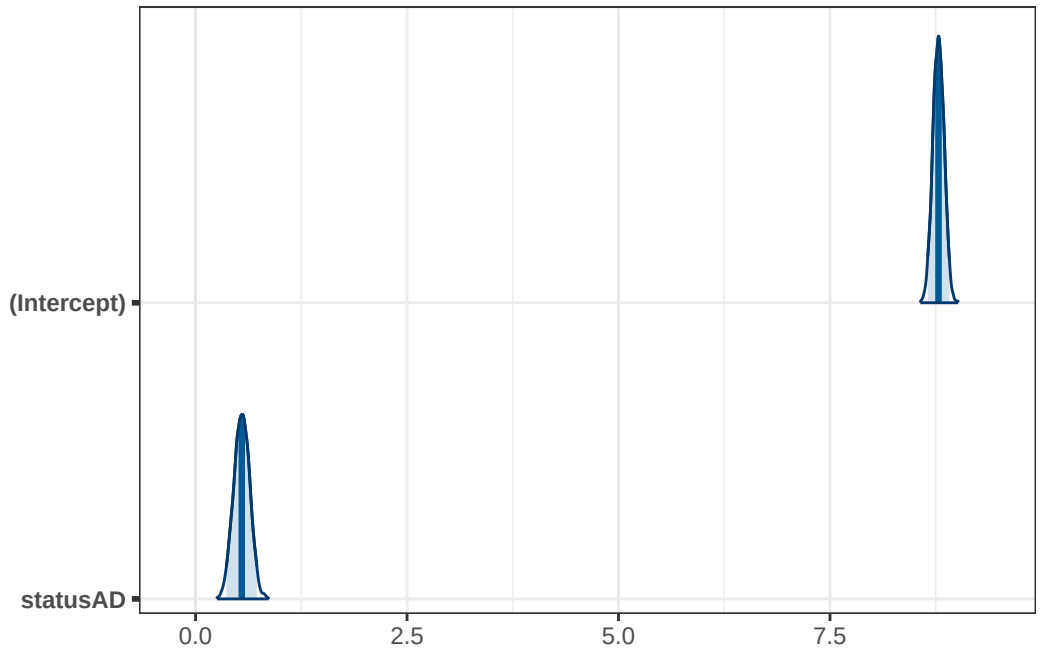
We could also fit a Bayesian linear model to our logged times for Task 3, as follows:

```
set.seed(431)
fit3c <- stan_glm(logtime03 ~ status, data = darwin, refresh = 0)

post3c <- describe_posterior(fit3c, ci = 0.95)

print_md(post3c, digits = 2)
```

Table 7.10: Summary of Posterior Distribution

Parameter	Median	95% CI	pd	ROPE	% in ROPE	Rhat	ESS
(Intercept)	8.78	[8.66, 8.91]	100%	[-0.07, 0.07]	0%	1.000	3621.00
statusAD	0.55	[0.36, 0.72]	100%	[-0.07, 0.07]	0%	0.999	3435.00

```
plot(fit3c, plotfun = "areas", prob = 0.95,
     pars = c("(Intercept)", "statusAD"))
```




```
fit3c |> model_parameters() |> kable(digits = 2)
```

Parameter	Median CI	CI_low	CI_high	pd	Rhat	ESS	Prior_Distribution	Prior_Location	Prior_Scale
(Intercept)	8.78	0.95	8.66	8.91	1	1	3620.57	normal	9.06	1.66
statusAD	0.55	0.95	0.36	0.72	1	1	3435.02	normal	0.00	3.31

As with an OLS model fit with `lm()`, we can back-transform expectations or predictions from this model, but not the coefficients themselves.

```
estimate_expectation(fit3c, data = "grid", ci = 0.95)
```

```
Model-based Expectation

status | Predicted |   SE |       95% CI
----------------------------------------
HC     |      8.78 | 0.07 | [8.66, 8.91]
AD     |      9.33 | 0.07 | [9.20, 9.46]

Variable predicted: logtime03
Predictors modulated: status
```

```
round_half_up(exp(c(8.78, 8.66, 8.91)),0) # for the mean across HC subjects
```

```
[1] 6503 5768 7406
```

```
round_half_up(exp(c(9.33, 9.20, 9.46)),0) # for the mean across AD subjects
```

```
[1] 11271  9897 12836
```

```
estimate_prediction(fit3c, data = "grid", ci = 0.95)
```

```
Model-based Prediction

status | Predicted |   SE |        95% CI
-----------------------------------------
HC     |      8.79 | 0.60 | [7.58,  9.97]
AD     |      9.33 | 0.61 | [8.13, 10.52]

Variable predicted: logtime03
Predictors modulated: status
```

```
round_half_up(exp(c(8.79, 7.58, 10.03)),0) # for an individual HC subject
```

```
[1]  6568  1959 22697
```

```
round_half_up(exp(c(9.32, 8.12, 10.56)),0) # for an individual AD subject
```

```
[1] 11159  3361 38561
```

7.9.5 Bootstrap CI for mean (time03) without transformation

Finally, we could use the bootstrap directly to build a confidence interval around the difference in means, without using a transformation. The first way I'll do this uses the tools from the `infer` package.

```
set.seed(431)
darwin |>
  specify(time03 ~ status) |>
  generate(reps = 1000, type = "bootstrap") |>
  calculate( stat = "diff in means", order = c("AD", "HC") ) |>
  get_confidence_interval( level = 0.95, type = "percentile" )
```

```
# A tibble: 1 x 2
  lower_ci upper_ci
     <dbl>    <dbl>
1    4270.    8048.
```

Or we could use the `boot.t.test()` approach from the `MKinfer` package.

```
set.seed(431)
boot.t.test(time03 ~ status, var.equal = TRUE, R = 2000,
  data = darwin, conf.level = 0.95)
```

```
    Bootstrap Two Sample t-test

data:  time03 by status
number of bootstrap samples:  2000
bootstrap p-value < 5e-04
bootstrap difference of means (SE) = -6115.905 (1087.369)
95 percent bootstrap percentile confidence interval:
 -8236.120 -3963.146

Results without bootstrap:
t = -6.1265, df = 172, p-value = 5.94e-09
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -8081.771 -4143.116
sample estimates:
mean in group HC mean in group AD
        7476.882        13589.326
```

Note the difference between the raw t test result (results without bootstrap) and the bootstrap in this setting.

We can run this without assuming equal population variances, as well.

```
set.seed(431)
boot.t.test(time03 ~ status, var.equal = FALSE, R = 2000,
  data = darwin, conf.level = 0.95 )
```

```
    Bootstrap Welch Two Sample t-test
```

```
data:  time03 by status
number of bootstrap samples:  2000
bootstrap p-value < 5e-04
bootstrap difference of means (SE) = -6130.079 (977.7301)
95 percent bootstrap percentile confidence interval:
 -8147.991 -4226.181

Results without bootstrap:
t = -6.2074, df = 134.42, p-value = 6.245e-09
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -8059.950 -4164.937
sample estimates:
mean in group HC mean in group AD
        7476.882        13589.326
```

7.9.6 Bootstrap CI for Medians

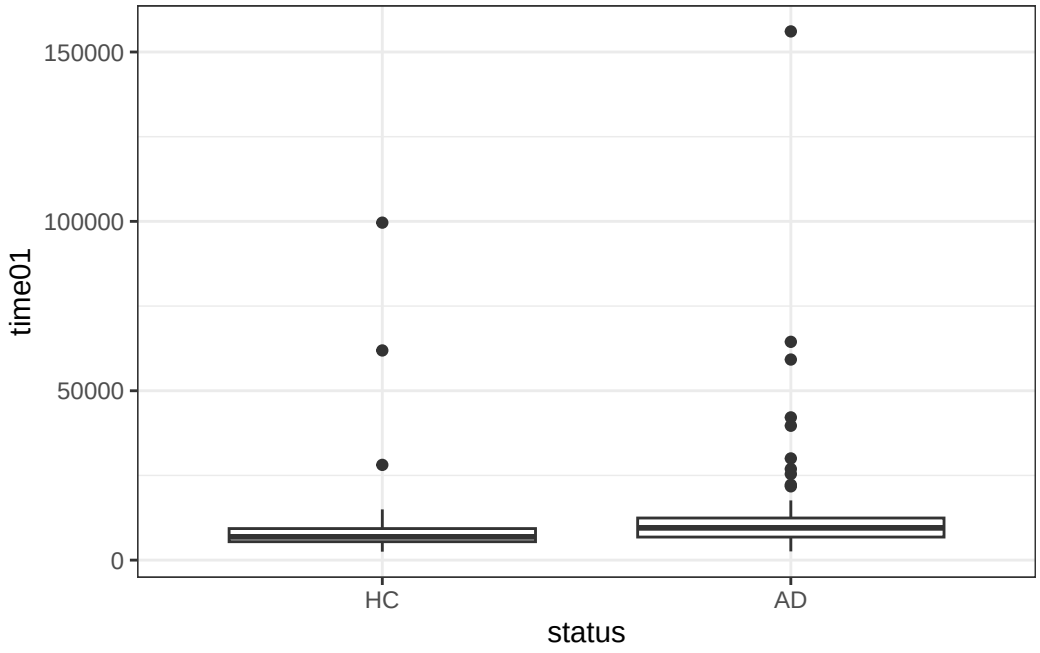
Instead of comparing mean times to complete Task 3, we could use the bootstrap to compare the medians directly.

```
set.seed(431)
darwin |>
  specify(time03 ~ status) |>
  generate(reps = 2500, type = "bootstrap") |>
  calculate(
    stat = "diff in medians",
    order = c("AD", "HC")
  ) |>
  get_ci(level = 0.95, type = "percentile")
```

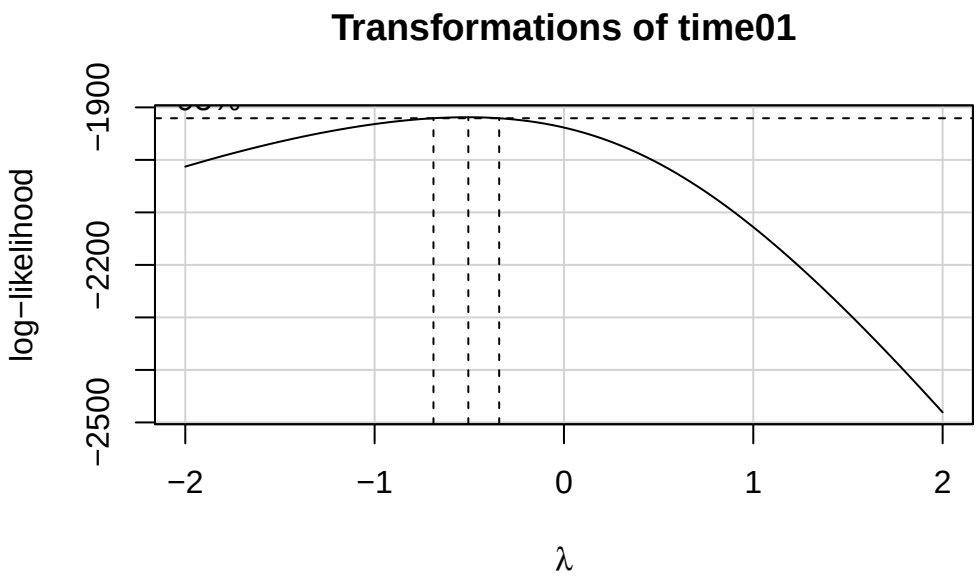
```
# A tibble: 1 x 2
  lower_ci upper_ci
     <dbl>    <dbl>
1     3275    7522.
```

7.10 What about Task 1?

```
ggplot(darwin, aes(x = status, y = time01)) +
  geom_boxplot()
```



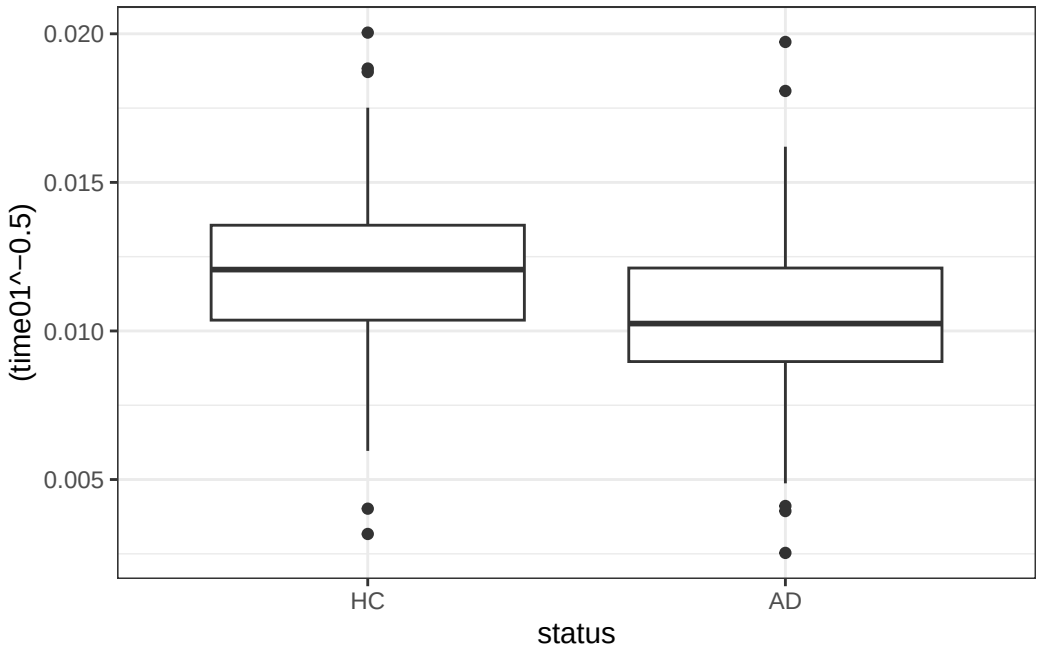

```
fit1 <- lm(time01 ~ status, data = darwin)
boxCox(fit1, main = "Transformations of time01")
```




```
summary(powerTransform(fit1))$result
```

```
    Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
Y1 -0.5110717        -0.5   -0.6849666   -0.3371769
```

```
ggplot(darwin, aes(x = status, y = (time01^-0.5))) +
  geom_boxplot()
```




```
darwin <- darwin |>
  mutate(trans_t01 = 1000 * time01^(-0.5))

darwin |>
  reframe(lovedist(trans_t01), .by = status)
```

```
# A tibble: 2 x 11
  status     n  miss  mean    sd   med   mad   min   q25   q75   max
  <fct>  <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 HC        85     0  12.0  2.81  12.1  2.37  3.17 10.4   13.6  20.0
2 AD        89     0  10.5  3.05  10.2  2.02  2.53  8.97  12.1  19.7
```

7.10.1 Linear Model

```
fit1 <- lm(trans_t01 ~ status, data = darwin)

fit1 |>
  model_parameters(ci = 0.95) |>
  kable(digits = 2)
```

Parameter	Coefficient	SE	CI	CI_low	CI_high	t	df_error	p
(Intercept)	11.98	0.32	0.95	11.36	12.61	37.64	172	0
statusAD	-1.45	0.45	0.95	-2.33	-0.57	-3.25	172	0

```
estimate_contrasts(fit1, ci = 0.95, contrast = "status")
```

```
Marginal Contrasts Analysis

Level1 | Level2 | Difference |       95% CI |   SE | t(172) |     p
-------------------------------------------------------------------
HC     |     AD |       1.45 | [0.57, 2.33] | 0.45 |   3.25 | 0.001

Marginal contrasts estimated at status
p-value adjustment method: Holm (1979)
```

7.10.2 Back-transformation of predictions and expectations...

Here our transformation is `trans_t01 = 1000 * time01^(-0.5)`. To back out of this transformation, we need to do a little algebra...

$$trans_{t01} = 1000 * time01^{-0.5} trans_{t01} = 1000 * \left(\frac{1}{\sqrt{time01}}\right) \frac{trans_{t01}}{1000} = \left(\frac{1}{\sqrt{time01}}\right) \frac{1000}{trans_{t01}} = \sqrt{time01} \left(\frac{100}{trans}\right.$$

We can obtain an expectation for `trans_t01` from `estimate_expectation()` and then apply this transformation in order to describe the result in terms of the time to complete Task 1 in milliseconds, as follows...

```
estimate_expectation(fit1, data = "grid", ci = 0.95)
```

```
Model-based Expectation

status | Predicted |   SE |         95% CI
-----------------------------------------
HC     |     11.98 | 0.32 | [11.36, 12.61]
AD     |     10.54 | 0.31 | [ 9.92, 11.15]

Variable predicted: trans_t01
Predictors modulated: status
```

So, for example, our expectation for the average across all HC subjects is 11.98 after our transformation. Backing out of this transformation, we have an estimate of 6968 milliseconds.

```
round_half_up((1000 / 11.98)^2, 0)
```

```
[1] 6968
```

To obtain the uncertainty interval for the expectation across all HC subjects, we apply the back-transformation to the endpoints of the interval, and then round the results to no decimal places as follows:

```
round_half_up(c((1000 / 10.54)^2, (1000 / 9.92)^2, (1000 / 11.15)^2), 0)
```

```
[1]  9002 10162  8044
```

and so our model's estimate for the time spent on Task 1 for HC subjects on average is 6968 milliseconds, with a 95% uncertainty interval of (6289, 7749) milliseconds.

For the average across AD subjects, we have a point estimate for the time spent on Task 1 of 9002 ms, and a 95% uncertainty interval of (8044, 10162) ms.

```
round_half_up(c((1000 / 11.36)^2, (1000 / 12.61)^2), 0)
```

```
[1] 7749 6289
```

We can obtain analogous results for individual predictions for HC and AD subjects, respectively, according to this model for transformed Task 1 times, with the following code:

```
estimate_prediction(fit1, data = "grid", ci = 0.95)
```

```
Model-based Prediction

status | Predicted |   SE |        95% CI
----------------------------------------
HC     |     11.98 | 2.95 | [6.16, 17.81]
AD     |     10.54 | 2.95 | [4.71, 16.36]

Variable predicted: trans_t01
Predictors modulated: status
```

```
round_half_up(c((1000 / 11.98)^2, (1000 / 6.16)^2, (1000 / 17.81)^2), 0) ## HC subjects
```

```
[1]  6968 26354  3153
```

```
round_half_up(c((1000 / 10.54)^2, (1000 / 4.71)^2, (1000 / 16.36)^2), 0) ## AD subjects
```

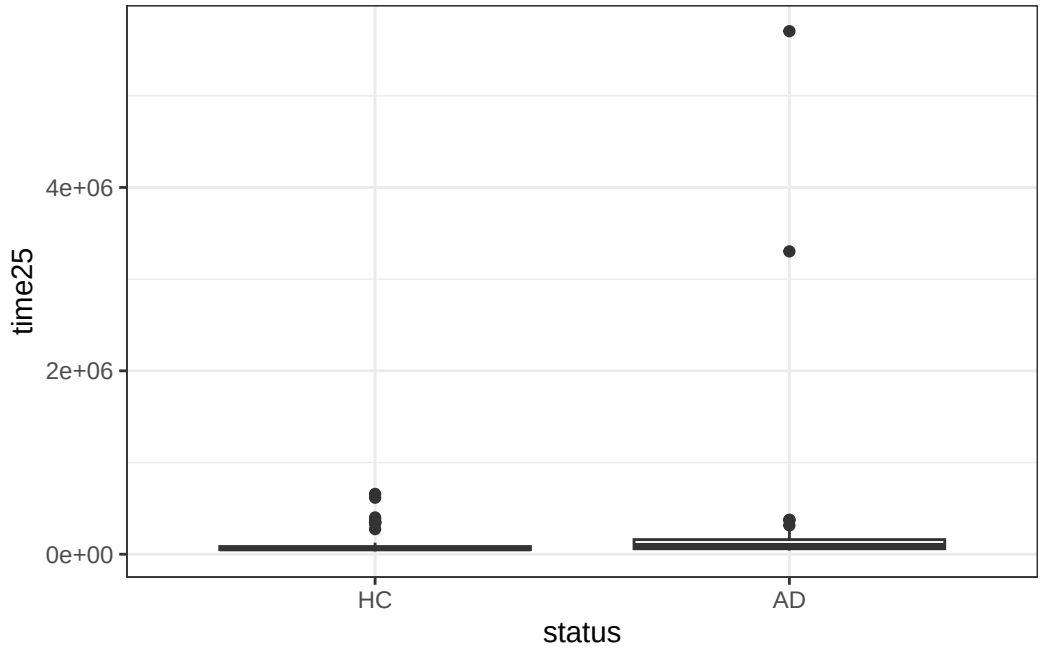
```
[1]  9002 45077  3736
```

All of our other approaches used for Task 3 would also work for Task 1, certainly.

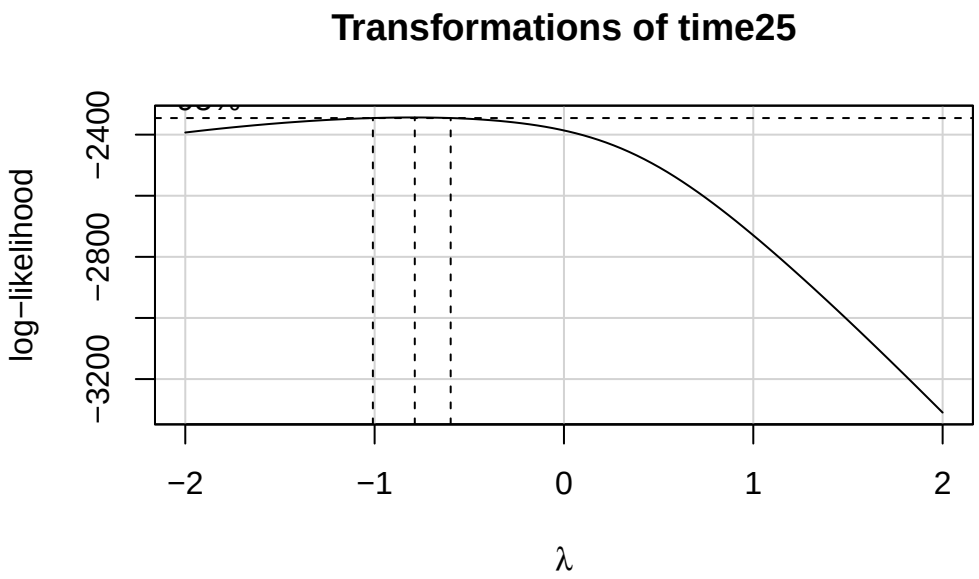
7.11 What about Task 25?

Finally, let's consider Task 25.

```
ggplot(darwin, aes(x = status, y = time25)) +
  geom_boxplot()
```



```
fit25 <- lm(time25 ~ status, data = darwin)
boxCox(fit25, main = "Transformations of time25")
```

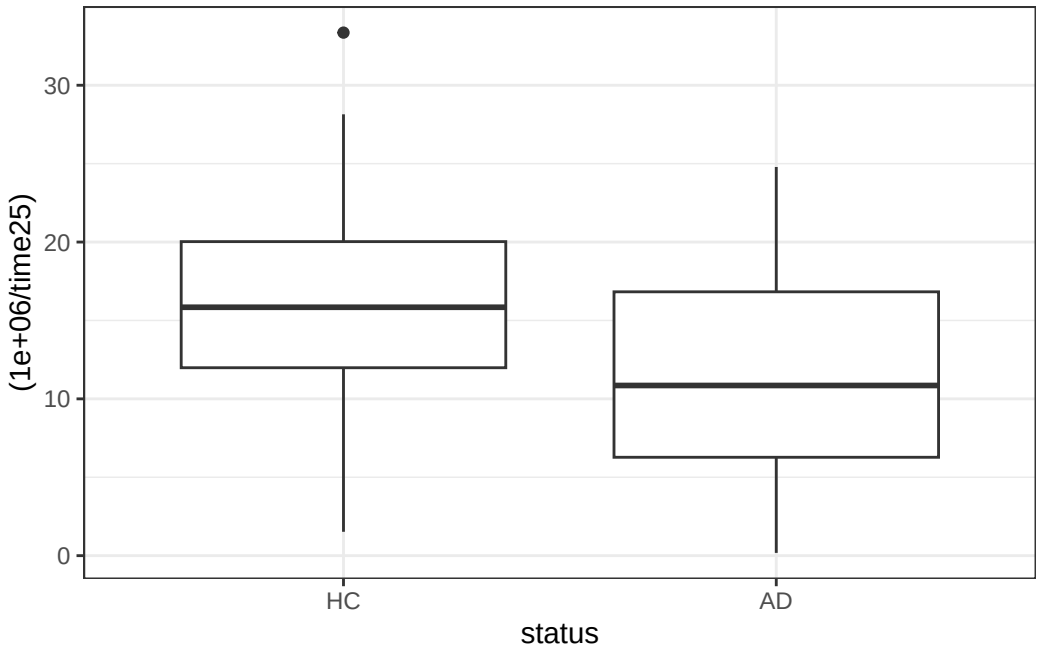



```
summary(powerTransform(fit25))$result
```

```
    Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
Y1 -0.7965846           -1    -1.002999   -0.5901699
```

Now, we'll use an inverse transformation and also multiply the values by 1,000,000 in order to get coefficients which fall between 1 and 100.

```
ggplot(darwin, aes(x = status, y = (1000000 / time25))) +
  geom_boxplot()
```




```
darwin <- darwin |>
  mutate(trans_t25 = 1000000 / time25)

darwin |>
  reframe(lovedist(trans_t25), .by = status)
```

```
# A tibble: 2 x 11
  status     n  miss  mean    sd   med   mad   min   q25   q75   max
  <fct>  <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 HC        85     0  15.3  6.84  15.8  5.74 1.52  12.0   20.0  33.4
2 AD        89     0  11.0  5.68  10.9  7.19 0.175  6.27  16.8  24.8
```

7.11.1 Linear Model

```
fit25 <- lm(trans_t25 ~ status, data = darwin)

fit25 |> model_parameters(ci = 0.95) |> kable(digits = 2)
```

Parameter	Coefficient	SE	CI	CI_low	CI_high	t	df_error	p
(Intercept)	15.25	0.68	0.95	13.91	16.59	22.41	172	0
statusAD	-4.27	0.95	0.95	-6.15	-2.40	-4.49	172	0

```
estimate_contrasts(fit25, ci = 0.95, contrast = "status")
```

```
Marginal Contrasts Analysis

Level1 | Level2 | Difference |       95% CI |   SE | t(172) |      p
-------------------------------------------------------------------
HC     |     AD |       4.27 | [2.40, 6.15] | 0.95 |   4.49 | < .001

Marginal contrasts estimated at status
p-value adjustment method: Holm (1979)
```

We could also bootstrap the confidence intervals in this model with …

```
model_parameters(fit25, bootstrap = TRUE, iterations = 2000,
  ci = 0.95, centrality = "median", ci_method = "quantile")
```

```
Parameter   | Coefficient |         95% CI |      p
-----------------------------------------------------
(Intercept) |       15.26 | [13.79, 16.68] | < .001
status [AD] |       -4.28 | [-6.16, -2.31] | < .001


Uncertainty intervals (equal-tailed) are naıve bootstrap intervals.
```

Working with the original `lm()` fit, we can obtain predictions or expectations and back-transform them as before. I'll show the average expectations here…

```
estimate_expectation(fit25, data = "grid", ci = 0.95)
```

```
Model-based Expectation

status | Predicted |   SE |         95% CI
-----------------------------------------
HC     |     15.25 | 0.68 | [13.91, 16.59]
```

```
AD     |     10.98 | 0.67 | [ 9.67, 12.29]

Variable predicted: trans_t25
Predictors modulated: status
```

Here are the back-transformed point estimate and 95% uncertainty interval for an individual prediction for a subject with status AD to complete Task 25, in milliseconds…

```
round_half_up(c((1000000 / 10.98), (1000000 / 9.67), (1000000 / 12.29)),0)
```

```
[1]  91075 103413  81367
```

so the point estimate is 91075 ms, with 95% uncertainty interval (81367, 103413).

However, we run into a challenge when we look at transforming out of the uncertainty interval for individual predictions.

```
estimate_prediction(fit25, data = "grid", ci = 0.95)
```

```
Model-based Prediction

status | Predicted |   SE |         95% CI
-----------------------------------------
HC     |     15.25 | 6.31 | [ 2.80, 27.71]
AD     |     10.98 | 6.31 | [-1.48, 23.43]

Variable predicted: trans_t25
Predictors modulated: status
```

For an individual AD subject, we have:

```
round_half_up(c((1000000 / 10.98), (1000000 / -1.48), (1000000 / 23.43)),0)
```

```
[1]   91075 -675676   42680
```

Our 95% uncertainty interval now includes negative times, and the uncertainty interval no longer includes the point estimate, and this causes multiple problems. This suggests that transforming our way out of a problem when making a comparison of this sort is not always going to produce useful results.

7.12 For More Information

1. Transformation is discussed in some detail as part of this vignette for the `modelbased` package vignette from the easystats meta-package.
2. You can learn more about model-based response estimates and uncertainty at this link which describes both the `estimate_expection()` and `estimate_prediction()` functions.
3. More on the Box-Cox family of power transformations and how they are used in R is available in this post from 2022 at Data Science Tutorials. A more general discussion of power transforms, including Box-Cox is available through Wikipedia.
4. Another nice introduction to transformations (using Stata, rather than R, but the ideas are still relevant) is available here.
5. This overview of functions in the janitor package includes explanations for several useful tools, including the `round_half_up()` function.

8 Weighting

8.1 R setup for this chapter

> **Note**
>
> Appendix A lists all R packages used in this book, and also provides R session information. Appendix B describes the 431-Love.R script, and demonstrates its use.

```
library(haven)
library(knitr)
library(naniar)

library(easystats)
library(tidyverse)

source("data/Love-431.R")
theme_set(theme_bw())
```

8.2 Data from a SAS file: NHANES 2015-16

> **Note**
>
> Appendix C provides further guidance on pulling data from other systems into R, while Appendix D gives more information (including download links) for all data sets used in this book.

In this chapter, we'll be using data from the 2015-16 administration of the NHANES (National Health and Nutrition Examination Survey.) See https://wwwn.cdc.gov/nchs/nhanes/continuousnhanes/overview.aspx?BeginYear=2015 for an overview of NHANES 2015-16.

Course Project B also uses NHANES data, and there we will use the `nhanesA` (see https://cran.r-project.org/web/packages/nhanesA/vignettes/Introducing_nhanesA.html) R package to pull in publicly available data.

Here, though, we'll use two SAS files (called `DEMO_I.xpt` and `BPX_I.xpt`) downloaded from the NHANES 2015-16 data repository to pull in the data we need. The files are known as SAS Transport files, and carry the extension `.xpt`. The `haven` package provides the `read_xpt()` function to ingest such files.

```
demo_i_raw <- read_xpt("data/DEMO_I.xpt")
bpx_i_raw <- read_xpt("data/BPX_I.xpt")
```

Now, we'll reduce the size of each raw file to include only the variables we plan to use.

```
demo_i <- demo_i_raw |>
  select(SEQN, RIDSTATR, RIAGENDR, RIDAGEYR, SIALANG,
         WTINT2YR, WTMEC2YR)

bpx_i <- bpx_i_raw |> select(SEQN, BPXSY1, BPXDI1)
```

Next, we'll merge the two files together (each contains an ID variable called SEQN which allows us to do this) using the `left_join()` function from the `dplyr` package within the `tidyverse` family.

```
nh1516 <- left_join(demo_i, bpx_i, by = "SEQN")
```

Several of the variables in `nh1516` are in fact categorical, and we'll account for that as follows.

```
nh1516 <- nh1516 |>
  mutate(RIDSTATR = factor(RIDSTATR),
         RIAGENDR = factor(RIAGENDR),
         SIALANG = factor(SIALANG),
         SEQN = as.character(SEQN))
```

8.3 What's in the Data?

We now have a tibble containing 9971 rows and 9 columns.

```
nh1516
```

```
# A tibble: 9,971 x 9
   SEQN  RIDSTATR RIAGENDR RIDAGEYR SIALANG WTINT2YR WTMEC2YR BPXSY1 BPXDI1
   <chr> <fct>    <fct>       <dbl> <fct>      <dbl>    <dbl>  <dbl>  <dbl>
```

```
 1 83732 2        1              62 1        134671.  135630.    128     70
 2 83733 2        1              53 1         24329.   25282.    146     88
 3 83734 2        1              78 1         12400.   12576.    138     46
 4 83735 2        2              56 1        102718.  102079.    132     72
 5 83736 2        2              42 1         17628.   18235.    100     70
 6 83737 2        2              72 2         11252.   10879.    116     58
 7 83738 2        2              11 1          9965.    9861.    102     36
 8 83739 2        1               4 1         44750.   46173.     NA     NA
 9 83740 2        1               1 1          9892.   10963.     NA     NA
10 83741 2        1              22 1         37043.   39353.    110     70
# i 9,961 more rows
```

From the About NHANES page...

> The NHANES interview includes demographic, socioeconomic, dietary, and health-related questions. The examination component consists of medical, dental, and physiological measurements, as well as laboratory tests administered by highly trained medical personnel.

The 9 variables included in `nh1516` are:

Variable	Description	Source
SEQN	respondent sequence number (code)	DEMO_I BPX_I
RIDSTATR	interview/examination status (1 = Interview only, 2 = Interview + MEC Exam)	DEMO_I
RIAGENDR	sex[1] (1 = Male, 2 = Female)	DEMO_I
RIDAGEYR	age in years at screening	DEMO_I
SIALANG	language used[2] in interview (1 = English, 2 = Spanish)	DEMO_I
WTINT2YR	full sample 2 year interview weight	DEMO_I
WTMEC2YR	full sample 2 year MEC[3] exam weight	DEMO_I
BPXSY1	first systolic blood pressure, in mm Hg	BPX_I
BPXDI1	first diastolic blood pressure, in mm Hg	BPX_I

[1]The name `RIAGENDR` is somewhat unfortunate, as this variable describes biological sex, rather than gender.
[2]Persons 16 years and older and emancipated minors were interviewed directly. A proxy provided information for survey participants who were under 16 and for participants who could not answer the questions themselves.
[3]MEC = Mobile Examination Center

8.3.1 Renaming factor levels

```
nh1516 |> count(RIAGENDR, SIALANG)
```

```
# A tibble: 4 x 3
  RIAGENDR SIALANG     n
  <fct>    <fct>   <int>
1 1        1        4239
2 1        2         653
3 2        1        4345
4 2        2         734
```

It would help to rename the factor levels from 1 and 2 to something more meaningful.

```
nh1516 <- nh1516 |>
  mutate(RIAGENDR =
           fct_recode(RIAGENDR, "Male" = "1", "Female" = "2"),
         SIALANG =
           fct_recode(SIALANG, "English" = "1", "Spanish" = "2"),
         RIDSTATR =
           fct_recode(RIDSTATR, "Int only" = "1",
                      "Exam and Int" = "2")
  )

nh1516 |> count(RIAGENDR, SIALANG)
```

```
# A tibble: 4 x 3
  RIAGENDR SIALANG     n
  <fct>    <fct>   <int>
1 Male     English  4239
2 Male     Spanish   653
3 Female   English  4345
4 Female   Spanish   734
```

8.3.2 Missing data?

The only missing values in our data are blood pressures. This is because not all subjects completed the MEC examination (where the blood pressures were recorded) as well as the interview.

```
miss_var_summary(nh1516)
```

```
# A tibble: 9 x 3
  variable n_miss pct_miss
  <chr>     <int>    <num>
1 BPXSY1     2826     28.3
2 BPXDI1     2826     28.3
3 SEQN          0      0
4 RIDSTATR      0      0
5 RIAGENDR      0      0
6 RIDAGEYR      0      0
7 SIALANG       0      0
8 WTINT2YR      0      0
9 WTMEC2YR      0      0
```

8.4 What is the average age?

8.4.1 Unweighted

Here's a summary of the age information in our sample.

```
nh1516 |>
  reframe(lovedist(RIDAGEYR)) |>
  kable(digits = 2)
```

n	miss	mean	sd	med	mad	min	q25	q75	max
9971	0	31.9	24.77	27	29.65	0	9	53	80

That's an accurate description of the ages shown in our **sample**, but each subject in NHANES is assigned a different sampling **weight** (called `WTINT2YR`) which allows this sample to represent the NHANES Target Population, which is…

> The NHANES (target) population is the non-institutionalized U.S. civilian population of all ages residing in all 50 states and Washington, D.C. The survey exmaines a nationally representative sample of about 5,000 persons per year.

- Source: CMS.gov on NHANES.

8.4.2 Weighting is important here

> In statistics it is common to reweight data or inferences so as to adapt to a target population. – Gelman, Hill, and Vehtari (2021)

From the description of the DEMO_I file:

> The 2-year sample weights (WTINT2YR, WTMEC2YR) should be used for all NHANES 2015-2016 analyses.

For a variable (like AGE) gathered through the interview process, we will use the `WTINT2YR` weight.

```
nh1516 |> select(SEQN, RIDAGEYR, WTINT2YR) |> head()
```

```
# A tibble: 6 x 3
  SEQN  RIDAGEYR WTINT2YR
  <chr>    <dbl>    <dbl>
1 83732       62  134671.
2 83733       53   24329.
3 83734       78   12400.
4 83735       56  102718.
5 83736       42   17628.
6 83737       72   11252.
```

So, for example, subject 83732 is 62 years old, and has a sampling weight of more than 134,600 people, while subject 83733 (aged 53) has a much smaller sampling weight of just over 24,300 people. In order to describe the NHANES target population, we must apply these weights in calculating summaries like the mean, median, standard deviation or MAD of age. So how might we do this?

8.4.3 Using `weighted_mean()`, etc.

A series of functions, called `weighted_mean()`, `weighted_median()` and so forth, are available from the `datawizard` package in the `easystats` family.

```
weighted_mean(nh1516$RIDAGEYR, weights = nh1516$WTINT2YR)
```

```
[1] 37.98659
```

```
weighted_sd(nh1516$RIDAGEYR, weights = nh1516$WTINT2YR)
```

```
[1] 22.60222
```

```
weighted_median(nh1516$RIDAGEYR, weights = nh1516$WTINT2YR)
```

```
[1] 37
```

```
weighted_mad(nh1516$RIDAGEYR, weights = nh1516$WTINT2YR)
```

```
[1] 28.1694
```

Compare these to the unweighted results (repeated below) and we see that the weighting makes a big difference - after weighting the sample appears to describe a much older population.

```
nh1516 |> summarise(
  mean = mean(RIDAGEYR), sd = sd(RIDAGEYR),
  median = median(RIDAGEYR), mad = mad(RIDAGEYR)
)
```

```
# A tibble: 1 x 4
   mean    sd median   mad
  <dbl> <dbl>  <dbl> <dbl>
1  31.9  24.8     27  29.7
```

8.5 Variation in age by preferred language?

Can we compare the ages we see by the language used in the interview?

8.5.1 Ignoring the weighting

```
means_by_group(nh1516$RIDAGEYR, by = nh1516$SIALANG)
```

```
# Mean of Age in years at screening by nh1516$SIALANG

Category |  Mean |    N |    SD |         95% CI |      p
-----------------------------------------------------------
English  | 31.45 | 8584 | 24.73 | [30.93, 31.97] | < .001
Spanish  | 34.67 | 1387 | 24.85 | [33.37, 35.98] | < .001
Total    | 31.90 | 9971 | 24.77 |                |

Anova: R2=0.002; adj.R2=0.002; F=20.258; p<.001
```

```
fit1 <- lm(RIDAGEYR ~ SIALANG, data = nh1516)

fit1 |> model_parameters(ci = 0.95) |> kable(digits = 2)
```

Parameter	Coefficient	SE	CI	CI_low	CI_high	t	df_error	p
(Intercept)	31.45	0.27	0.95	30.93	31.97	117.76	9969	0
SIALANGSpanish	3.22	0.72	0.95	1.82	4.63	4.50	9969	0

Ignoring the weighting, we see that the mean age of subjects for whom the interview is in Spanish is about 3.2 years older than those who were interviewed in English, with 95% CI (1.8, 4.6) years.

8.5.2 Weighted means within subgroups

Let's look at the mean age after weighting within each `SIALANG` group.

```
nh1516_ENG <- nh1516 |> filter(SIALANG == "English")

weighted_mean(nh1516_ENG$RIDAGEYR, weights = nh1516_ENG$WTINT2YR)
```

```
[1] 38.28088
```

```
weighted_sd(nh1516_ENG$RIDAGEYR, weights = nh1516_ENG$WTINT2YR)
```

```
[1] 22.68499
```

```
nh1516_ESP <- nh1516 |> filter(SIALANG == "Spanish")

weighted_mean(nh1516_ESP$RIDAGEYR, weights = nh1516_ESP$WTINT2YR)
```

```
[1] 34.21842
```

```
weighted_sd(nh1516_ESP$RIDAGEYR, weights = nh1516_ESP$WTINT2YR)
```

```
[1] 21.16711
```

8.5.3 Using `means_by_group()`

Another approach is available through the `means_by_group()` function.

```
means_by_group(nh1516$RIDAGEYR,
               by = nh1516$SIALANG,
               weights = nh1516$WTINT2YR)
```

```
# Mean of Age in years at screening by nh1516$SIALANG

Category |  Mean |     N |    SD |         95% CI |      p
-----------------------------------------------------------
English  | 38.28 | 3e+08 | 22.68 | [37.82, 38.74] | < .001
Spanish  | 34.22 | 2e+07 | 21.17 | [32.57, 35.87] | < .001
Total    | 37.99 |  9971 | 22.60 |                |

Anova: R2=0.002; adj.R2=0.002; F=21.691; p<.001
```

8.5.4 Comparing Weighted Means: Linear Model

```
fit1w <- lm(RIDAGEYR ~ SIALANG,
            weights = WTINT2YR, data = nh1516)

fit1w |> model_parameters(ci = 0.95) |> kable(digits = 2)
```

Parameter	Coefficient	SE	CI	CI_low	CI_high	t	df_error	p
(Intercept)	38.28	0.23	0.95	37.82	38.74	163.06	9969	0
SIALANGSpanish	-4.06	0.87	0.95	-5.77	-2.35	-4.66	9969	0

However, after incorporating the weighting, we see that the weighted mean age of subjects for whom the interview is in Spanish is almost 4.1 years YOUNGER than those who were interviewed in English, with 95% CI (2.4, 5.8) years.

So the weighting in fact alters the direction of our estimate. This is deliberate, as primary language is one of the strata on which the NHANES weights are based to match the sample to the target population.

8.6 What is the average systolic blood pressure (SBP)?

8.6.1 Unweighted SBP summaries

```
nh1516 |> reframe(lovedist(BPXSY1)) |> kable(digits = 2)
```

n	miss	mean	sd	med	mad	min	q25	q75	max
9971	2826	120.54	18.62	118	17.79	72	108	130	236

8.6.2 Weighted mean and SD

For the blood pressure readings, which are obtained through the MEC exam, rather than through the interview, we will need to use the MEC weights gathered in `WTMEC2YR` instead of the interview weights we used previously.

```
weighted_mean(nh1516$BPXSY1, weights = nh1516$WTMEC2YR)
```

```
Warning: Some `weights` were negative. Weighting not carried out.
```

```
[1] 120.5394
```

Whoops - we have some negative weights?

```
summary(nh1516$WTMEC2YR)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      0   12551   20281   31740   33708  242387
```

Actually, it looks like we have some zeros.

Are all of these coming from the people who were not examined?

```
nh1516 |>
  reframe(lovedist(WTMEC2YR), .by = RIDSTATR)
```

```
# A tibble: 2 x 11
  RIDSTATR        n  miss   mean     sd    med    mad   min    q25    q75    max
  <fct>       <int> <int>  <dbl>  <dbl>  <dbl>  <dbl> <dbl>  <dbl>  <dbl>  <dbl>
1 Exam and I~  9544     0 33160. 34178. 21034. 13256. 3419. 13395. 34969. 2.42e5
2 Int only      427     0     0      0      0      0     0      0      0  0
```

OK. So in order to apply the weights, we need to restrict ourselves to only those subjects who completed the MEC Examination.

```
nh1516_exam <- nh1516 |>
  filter(RIDSTATR == "Exam and Int")
```

```
nh1516_exam |> select(WTMEC2YR) |> summary()
```

```
    WTMEC2YR
 Min.   :  3419
 1st Qu.: 13395
 Median : 21034
 Mean   : 33160
 3rd Qu.: 34969
 Max.   :242387
```

OK. Now we shouldn't have any non-positive weights...

```
weighted_mean(nh1516_exam$BPXSY1, weights = nh1516_exam$WTMEC2YR)
```

```
[1] 120.6674
```

```
weighted_sd(nh1516_exam$BPXSY1, weights = nh1516_exam$WTMEC2YR)
```

```
[1] 17.45271
```

After weighting, the mean SBP is a little larger, and the standard deviation of SBP is a little smaller.

8.7 Variation in SBP by sex?

8.7.1 Ignoring the weighting

```
means_by_group(nh1516$BPXSY1, by = nh1516$RIAGENDR)
```

```
# Mean of Systolic: Blood pres (1st rdg) mm Hg by nh1516$RIAGENDR

Category |   Mean |    N |    SD |           95% CI |      p
-----------------------------------------------------------
Male     | 122.18 | 3498 | 18.15 | [121.56, 122.79] | < .001
Female   | 118.97 | 3647 | 18.93 | [118.37, 119.57] | < .001
Total    | 120.54 | 7145 | 18.62 |                  |

Anova: R2=0.007; adj.R2=0.007; F=53.393; p<.001
```

```
fit2 <- lm(BPXSY1 ~ RIAGENDR, data = nh1516)

fit2 |> model_parameters(ci = 0.95) |> kable(digits = 2)
```

Parameter	Coefficient	SE	CI	CI_low	CI_high	t	df_error	p
(Intercept)	122.18	0.31	0.95	121.56	122.79	389.56	7143	0
RIAGENDRFemale	-3.21	0.44	0.95	-4.07	-2.35	-7.31	7143	0

Ignoring the weighting, we see that the mean SBP for female subjects is about 3.2 mm Hg smaller than male subjects, with 95% CI for the difference of (2.4, 4.1) mm Hg.

8.7.2 Weighted Means by sex

Again, we'll restrict ourselves to those completing both the Exam and the interview.

```
means_by_group(nh1516_exam$BPXSY1,
               by = nh1516_exam$RIAGENDR,
               weights = nh1516_exam$WTMEC2YR)
```

```
# Mean of Systolic: Blood pres (1st rdg) mm Hg by nh1516_exam$RIAGENDR

Category |   Mean |     N |    SD |           95% CI |      p
-------------------------------------------------------------
Male     | 122.08 | 1e+08 | 16.42 | [121.50, 122.66] | < .001
Female   | 119.33 | 1e+08 | 18.28 | [118.77, 119.90] | < .001
Total    | 120.67 |  7145 | 17.45 |                  |

Anova: R2=0.006; adj.R2=0.006; F=44.449; p<.001
```

8.7.3 Comparing Weighted Means: Linear Model

```
fit2w <- lm(BPXSY1 ~ RIAGENDR,
            weights = WTMEC2YR, data = nh1516_exam)

fit2w |> model_parameters(ci = 0.95) |> kable(digits = 2)
```

Parameter	Coefficient	SE	CI	CI_low	CI_high	t	df_error	p
(Intercept)	122.08	0.30	0.95	121.50	122.66	413.58	7143	0
RIAGENDRFemale	-2.75	0.41	0.95	-3.55	-1.94	-6.67	7143	0

After weighting, the SBP difference is more like 2.75 mm Hg (95% CI: 1.9, 3.6) which is a bit smaller than what we saw without weighting.

8.8 For More Information

1. The National Center for Health Statistics, as part of its materials on the NHANES (National Health and Nutrition Examination Survey) publishes modules found here on why weights are calculated, how they are calculated, the importance of weights in making

estimates that are representative of the U.S. civilian non-institutionalized population, how to select the appropriate weight to use in your analysis, and other issues.

2. Thomas Lumley on Weights in Statistics from 2020 is a nice way of learning more about the various uses of the terms weights and weighting in statistics and data science.

3. Max Kuhn on Using case weights with tidymodels in R may be helpful to you for a general overview. Case weights are non-negative numbers used to specify how much each observation influences the estimation of a model.

4. Within R for Data Science, the Joins chapter provides more information about joining and merging R data frames.

9 Comparing Multiple Groups

9.1 R setup for this chapter

> **Note**
>
> Appendix A lists all R packages used in this book, and also provides R session information. Appendix B describes the 431-Love.R script, and demonstrates its use.

```
library(ggdist)
library(janitor)
library(knitr)
library(naniar)
library(rstanarm)

library(easystats)
library(tidyverse)

source("data/Love-431.R")
theme_set(theme_bw())
```

9.2 Data from a tab-separated file: Contrast Baths and Carpal Tunnel Syndrome

> **Note**
>
> Appendix C provides further guidance on pulling data from other systems into R, while Appendix D gives more information (including download links) for all data sets used in this book.

Study participants were randomly assigned to one of three treatment group protocols–contrast baths with exercise, contrast baths without exercise, and an exercise-only control treatment

group. Study participants were evaluated with hand volumetry, before and after treatment at two different data collection periods-pre- and postoperatively.

The original article is A randomized controlled study of contrast baths on patients with carpal tunnel syndrome by Robert G. Janssen, Deborah A. Schwartz, and Paul F Velleman. I sourced these data from the Data and Story Library.

Data were gathered on 76 participants (59 had complete data on our outcome) before Carpal Tunnel Release surgery. The changes in hand volume (the after treatment volume minus the before treatment volume) are the outcome of interest.

```
cbaths <- read_tsv("data/cbaths.txt",
                   show_col_types = FALSE) |>
  janitor::clean_names() |>
  mutate(treatment = factor(treatment))

miss_var_summary(cbaths)
```

```
# A tibble: 2 x 3
  variable     n_miss pct_miss
  <chr>         <int>    <num>
1 hand_vol_chg     17     22.4
2 treatment         0      0
```

```
## drop the rows with missing values

cbaths <- cbaths |>
  drop_na()
```

```
cbaths
```

```
# A tibble: 59 x 2
   treatment     hand_vol_chg
   <fct>                <dbl>
 1 Bath                    10
 2 Exercise                 0
 3 Bath                    10
 4 Bath                     5
 5 Exercise                 4
 6 Bath+Exercise            4
 7 Exercise                 2
 8 Bath                    -4
```

```
 9 Bath                    8
10 Bath+Exercise           0
# i 49 more rows
```

9.2.1 Summarizing Hand Volume Change by Treatment Group

Our data shows hand volume change (hand_vol_chg) for each subject, as well as which of the three treatments they received. Suppose we want to compare the means of these outcomes across the three treatment groups.

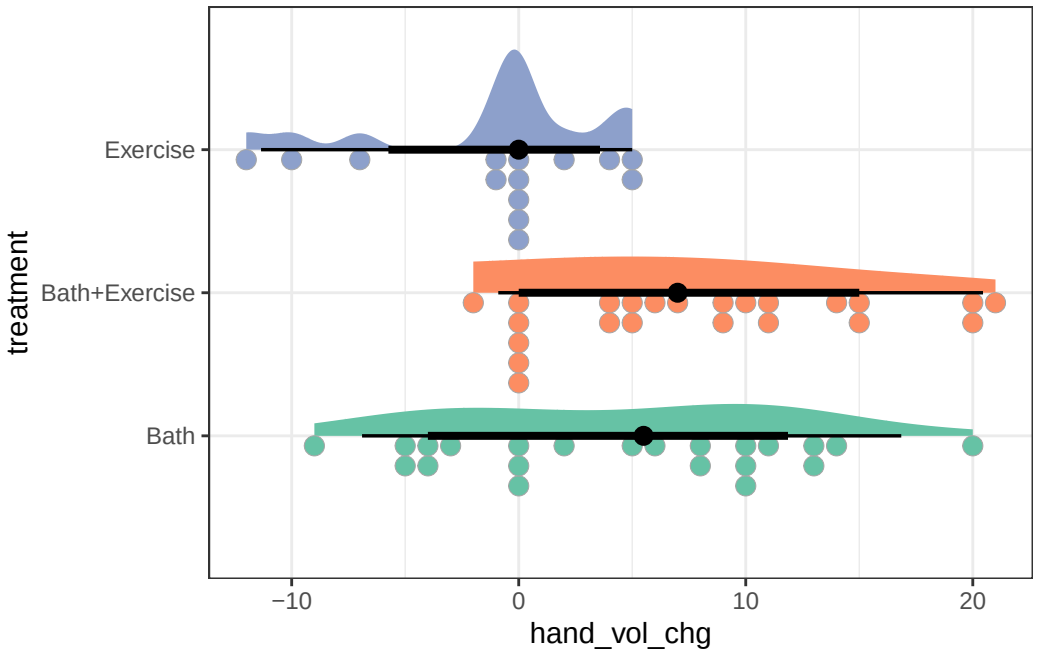
```
cbaths |>
  reframe(lovedist(hand_vol_chg), .by = treatment) |>
  kable(digits = 2)
```

treatment	n	miss	mean	sd	med	mad	min	q25	q75	max
Bath	22	0	4.55	7.76	5.5	8.15	-9	-2.25	10.0	20
Exercise	14	0	-1.07	5.18	0.0	2.22	-12	-1.00	1.5	5
Bath+Exercise	23	0	8.00	7.04	7.0	10.38	-2	2.00	12.5	21

Our sample sizes are fairly small in each group, so we'll keep that in mind as we plot the data, thinking about center, spread and shape.

9.2.2 Rain Cloud Plot

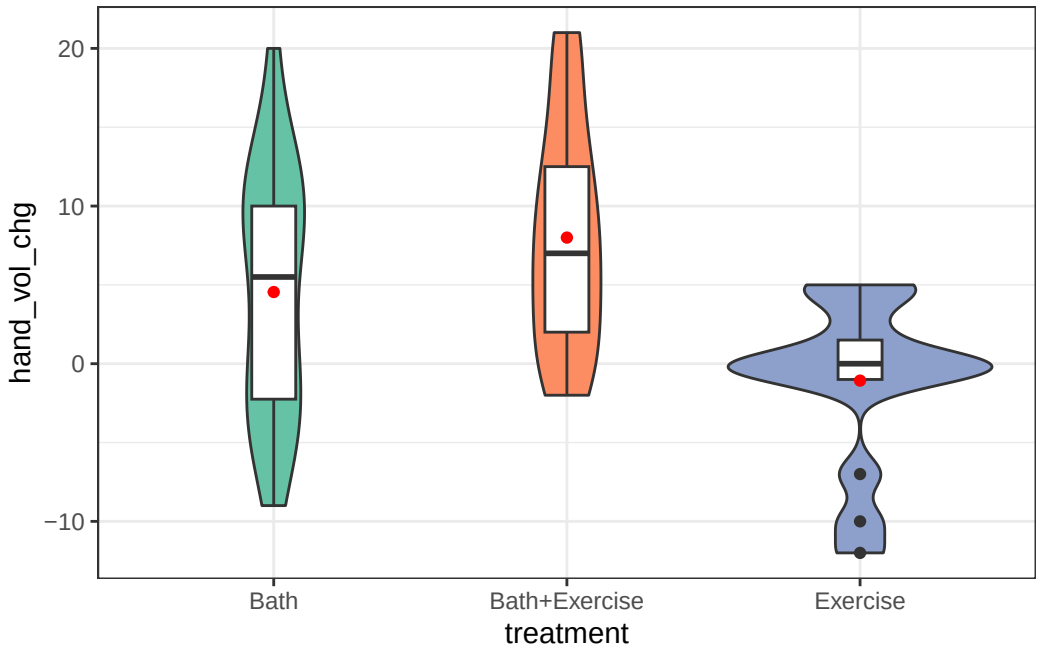
```
ggplot(cbaths, aes(y = treatment, x = hand_vol_chg, fill = treatment)) +
  stat_slab(aes(thickness = after_stat(pdf * n)), scale = 0.7) +
  stat_dotsinterval(side = "bottom", scale = 0.7, slab_linewidth = NA) +
  scale_fill_brewer(palette = "Set2") +
  guides(fill = "none")
```



With smaller sample sizes, it's a bit hard to make strong conclusions about the shape of the data in each group from these plots, or from the boxplots below.

9.2.3 Boxplot and Violin with Means

```
ggplot(cbaths, aes(x = treatment, y = hand_vol_chg)) +
  geom_violin(aes(fill = treatment)) +
  geom_boxplot(width = 0.15) +
  stat_summary(geom = "point", fun = "mean", col = "red") +
  scale_fill_brewer(palette = "Set2") +
  guides(fill = "none")
```



The data in the Bath and Bath+Exercise groups are highly spread out, I think, with the means and medians fairly close to each other. In the Exercise group, we see a few candidate outliers on the low end, and the mean is below even the 25th percentile as a result.

It's not really clear how willing we should be to assume that the data from each of the samples came from a Normal distribution here, as is often the case with small sample sizes. A linear model does make this assumption when comparing three (or more) group means, but happily is pretty robust to modest violations of that assumption. Let's run the OLS model and see what we get.

9.3 Comparing Means with a Linear Model

9.3.1 Linear Model Comparing 3 `treatment` means

Here we build an ordinary least squares (OLS) fit using `lm()` to predict the outcome based on which of the three treatment groups each subject is in. The predictions made turn out to just be the sample mean of the outcome (hand volume change) in each treatment group. Since we have a small sample size, let's use a 90% confidence interval here, rather than the default choice of 95%.

```
fit1 <- lm(hand_vol_chg ~ treatment, data = cbaths)

model_parameters(fit1, ci = 0.90) |> kable(digits = 2)
```

Parameter	Coefficient	SE	CI	CI_low	CI_high	t	df_error	p
(Intercept)	4.55	1.48	0.9	2.07	7.02	3.07	56	0.00
treatmentBath+Exercise	3.45	2.07	0.9	-0.01	6.92	1.67	56	0.10
treatmentExercise	-5.62	2.38	0.9	-9.59	-1.64	-2.36	56	0.02

9.3.2 Estimate means at each level from model

```
means1 <- estimate_means(fit1, ci = 0.90)
```

```
We selected `by = c("treatment")`.
```

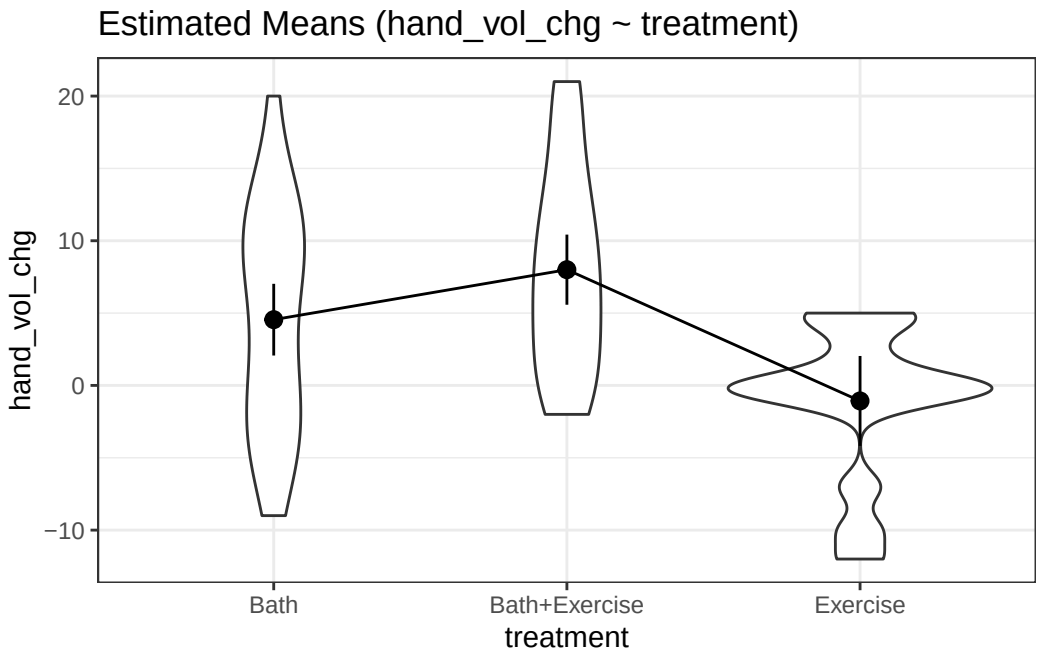
```
means1
```

```
Estimated Marginal Means

treatment      |  Mean |   SE |         90% CI
-----------------------------------------------
Bath           |  4.55 | 1.48 | [ 2.07,  7.02]
Bath+Exercise |  8.00 | 1.45 | [ 5.58, 10.42]
Exercise       | -1.07 | 1.86 | [-4.18,  2.03]

Marginal means estimated at treatment
```

It looks like, for example, the Bath+Exercise group has a 90% CI which is completely above the Bath group, which is in turn completely above the Exercise group.

Here is a violin plot of each group, on which we've included the sample means and the 90% confidence intervals.

```
plot(visualisation_recipe(means1,
                          show_data = "violin"))
```

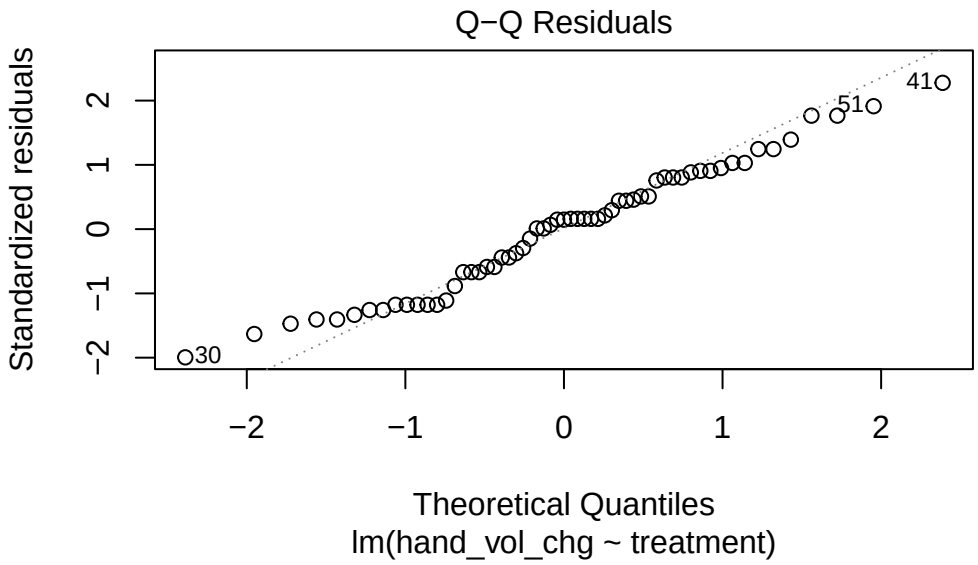


For more on using this approach to visualization when modeling a group of means, you might be interested in this modelbased package vignette.

9.3.3 Assessing Normality

One way to assess whether we have a meaningful problem is to look at some diagnostic plots for our linear model, which we'll see more of in Chapter 11. But for now, let's consider a Normal Q-Q plot of the regression residuals for our model `fit1` as a way of assessing whether non-Normality in our samples is a serious problem for our fit.

```
plot(fit1, which = 2)
```



With the possible exception of observation 30 (which may be an outlier), our Normal Q-Q plot of residuals looks pretty good. There's no evidence of strong violations of the Normality assumption within our fit. So we'll proceed.

9.3.4 Analysis of Variance Tables

To compare means of two or more independently sampled groups, we can use an analysis of variance (ANOVA) table, which can be generated in at least the following two ways:

```
anova(fit1)
```

```
Analysis of Variance Table

Response: hand_vol_chg
          Df  Sum Sq Mean Sq F value   Pr(>F)
treatment  2  716.16  358.08  7.4148 0.001391 **
Residuals 56 2704.38   48.29
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(aov(fit1))
```

```
            Df Sum Sq Mean Sq F value  Pr(>F)
treatment    2  716.2   358.1   7.415 0.00139 **
Residuals   56 2704.4    48.3
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Either one of these ANOVA tables can be used to estimate "eta-squared" (η^2), the proportion of variation explained by the model (which is the sum of squares attributed to the treatment groups - here, 716.16), divided by the total sum of squares (which is 716.16 + 2704.4 = 3420.56.)

We can also obtain an estimate for "eta-squared" (η^2), which is the proportion of variation in our outcome accounted for by the model, i.e. the proportion of variation captured by the means of the treatment groups. Here, the value of η^2 is estimated to be 21%.

```
eta_squared(fit1, ci = 0.90)
```

```
# Effect Size for ANOVA

Parameter | Eta2 |       90% CI
------------------------------
treatment | 0.21 | [0.09, 1.00]

- One-sided CIs: upper bound fixed at [1.00].
```

What if we want to directly compare each combination of means (Bath + Exercise to Bath, Exercise to Bath, and Bath + Exercise to Exercise) with 90% confidence intervals? We have two main approaches that we discuss here: Holm's method (which is a bit superior to the Bonferroni approach in that it's more powerful) and Tukey's method (which is appropriate if we pre-planned a set of pairwise comparisons.)

9.3.5 Pairwise Comparisons using Holm method

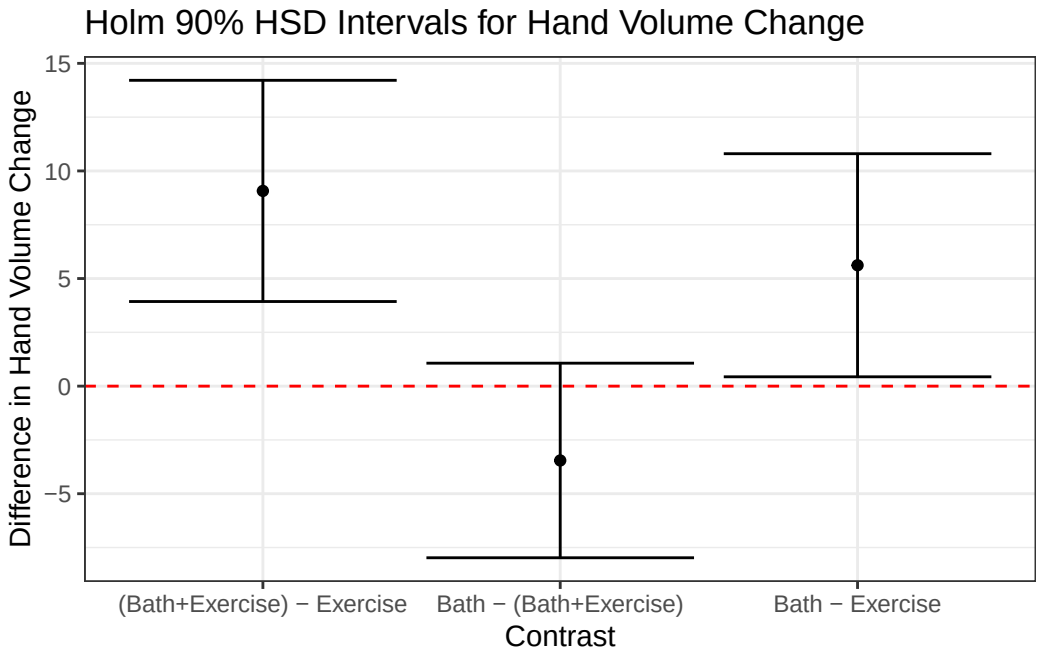
```
con1 <- estimate_contrasts(fit1, contrast = "treatment",
                           ci = 0.90, p.adjust = "holm")

con1 |> kable(digits = 2)
```

Level1	Level2	Difference	CI_low	CI_high	SE	df	t	p
(Bath+Exercise)	Exercise	9.07	3.93	14.21	2.36	56	3.85	0.00
Bath	(Bath+Exercise)	-3.45	-7.98	1.07	2.07	56	-1.67	0.10
Bath	Exercise	5.62	0.43	10.80	2.38	56	2.36	0.04

We see that, for example, the comparison of (Bath+Exercise) - Exercise has an estimated mean difference of 9.07, with a 90% confidence interval of (3.93, 14.21). It appears that the combination treatment led to larger hand volume changes, on average.

We can draw similar conclusions about the other two paired comparisons. It's often helpful to plot these comparisons, and we can do this as follows.

```
con1_tib <- tibble(con1) |>
  mutate(contr = str_c(Level1, " - ", Level2))

ggplot(con1_tib, aes(x = contr, y = Difference)) +
  geom_point() +
  geom_errorbar(aes(ymin = CI_low, ymax = CI_high)) +
  geom_hline(yintercept = 0, col = "red", lty = "dashed") +
  labs(title = "Holm 90% HSD Intervals for Hand Volume Change",
       x = "Contrast",
       y = "Difference in Hand Volume Change")
```



Note that only the Bath - (Bath + Exercise) comparison has a negative point estimate, and it also is the only comparison which has a 90% confidence interval which includes zero. Why might that be interesting?

9.3.6 Pairwise Comparisons using Tukey's HSD method

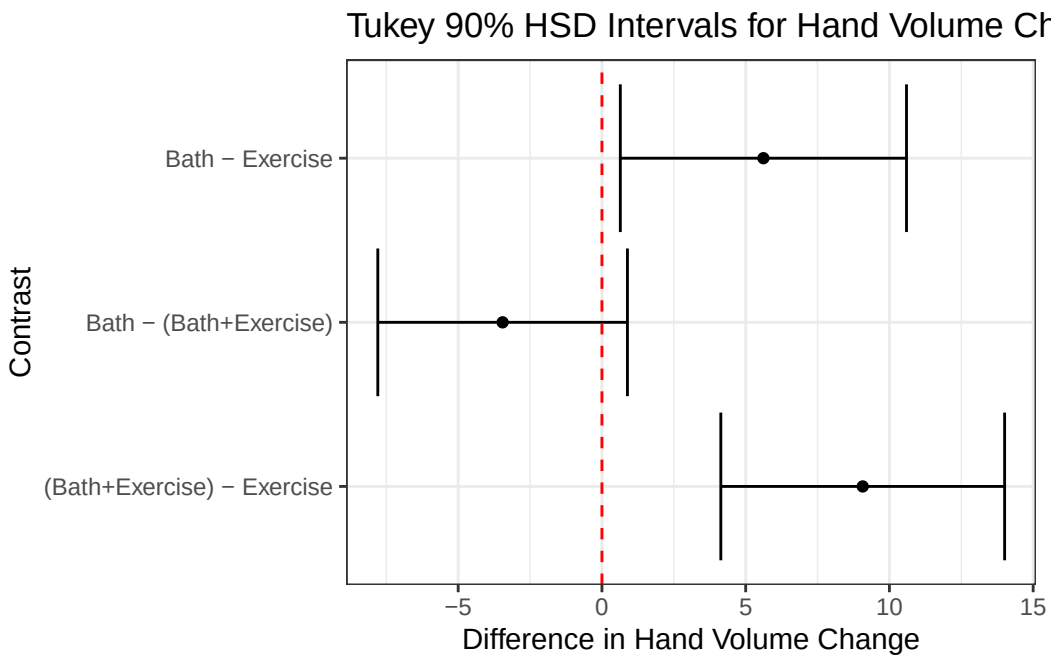
Another approach (more powerful than Holm's approach if we pre-plan our pairwise comparisons) is Tukey's Honestly Significant Differences method.

```
con1a <- estimate_contrasts(fit1,
  contrast = "treatment", p_adjust = "tukey", ci = 0.90
)

con1a |> kable(digits = 2)
```

Level1	Level2	Difference	CI_low	CI_high	SE	df	t	p
(Bath+Exercise)	Exercise	9.07	4.14	14.01	2.36	56	3.85	0.00
Bath	(Bath+Exercise)	-3.45	-7.80	0.89	2.07	56	-1.67	0.23
Bath	Exercise	5.62	0.64	10.59	2.38	56	2.36	0.06

Are there important differences in the findings from the Tukey and Holm procedures in this case?

```
con1a_tib <- tibble(con1a) |>
  mutate(contr = str_c(Level1, " - ", Level2))

ggplot(con1a_tib, aes(y = contr, x = Difference)) +
  geom_point() +
  geom_errorbar(aes(xmin = CI_low, xmax = CI_high)) +
  geom_vline(xintercept = 0, col = "red", lty = "dashed") +
  labs(title = "Tukey 90% HSD Intervals for Hand Volume Change",
       y = "Contrast",
       x = "Difference in Hand Volume Change")
```



9.4 Bayesian Model comparing 3 means

To fit a Bayesian linear model to compare the mean hand volume changes across our three treatment groups, we can use the following code.

```
set.seed(431)
fit2 <- stan_glm(hand_vol_chg ~ treatment,
                 data = cbaths, refresh = 0)
```

```
post2 <- describe_posterior(fit2, ci = 0.90)
print_md(post2, digits = 2)
```

Table 9.5: Summary of Posterior Distribution

Parameter	Median	90% CI	pd	ROPE	% in ROPE	Rhat	ESS
(Intercept)	4.58	[2.14, 7.01]	99.72%	[-0.77, 0.77]	0%	1.000	3141.00
treatmentBath+Exercise	3.45	[0.04, 6.87]	95.12%	[-0.77, 0.77]	7.82%	1.000	2951.00
treatmentExercise	-5.70	[-9.72, -1.74]	98.95%	[-0.77, 0.77]	0%	0.999	3201.00

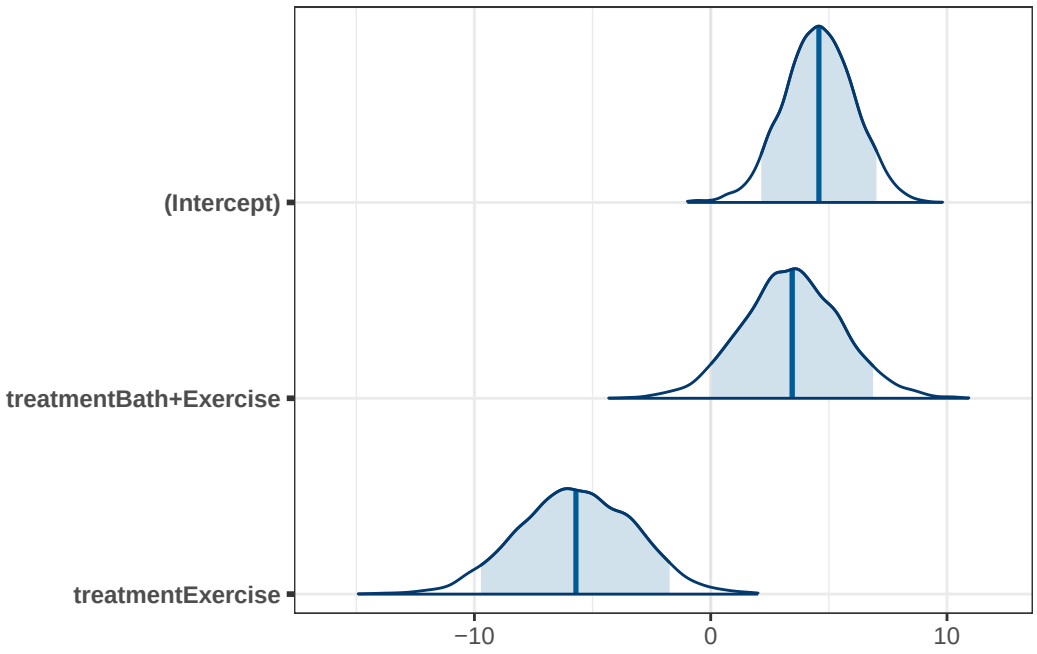
We see that the model selects the Bath only group as its baseline, with an estimated mean (based on the median of the posterior distribution) of 4.58, with 90% credible interval (2.14, 7.01).

The difference between Bath only and Bath+Exercise is estimated to be 3.45 (with higher values for the Bath+Exercise group) with 90% credible interval (0.04, 6.87).

The difference between Bath only and Exercise only is estimated to be 5.70 (with higher values for the Bath group) with 90% credible interval (1.74, 9.72).

Here is the plot of these three effects, showing the 90% credible intervals.

```
plot(fit2,
  plotfun = "areas", prob = 0.90,
  pars = c("(Intercept)", "treatmentBath+Exercise",
           "treatmentExercise"))
```




```
fit2 |> model_parameters(ci = 0.90) |> kable(digits = 2)
```

Parameter	Median	CI	CI_low	CI_high	pd	Rhat	ESS	Prior_Distribution	Prior_Location	Prior_Scale
(Intercept)	4.58	0.9	2.14	7.01	1.00	1	3141.37	normal	4.56	19.20
treatmentBath+Exercise	3.45	0.9	0.04	6.87	0.95	1	2950.71	normal	0.00	39.03

Parameter	Median	CI	CI_low	CI_high	pd	Rhat	ESS	Prior_Distribution	Prior_Location	Prior_Scale
treatment	Exercise - 5.70	0.9	- 9.72	-1.74	0.99	1	3201.18	normal	0.00	44.74

Again, we obtain the comparisons of the various treatment group means in terms of hand volume change.

9.5 Non-Normality?

What if we're not willing to assume a Normal distribution for each sample? Happily the ANOVA procedure is pretty robust and we'll usually be OK thinking about a linear model (Bayesian or OLS) in practical work, unless we have substantial skew. The natural solution in the case of skew might be to consider a transformation of our outcome, and we'll think about that more in our next chapter.

There are other options, which include extending our bootstrap approach to the case of 3 or more independent samples, working with a randomization approach, or the use of a rank transformation to create something called a Kruskal-Wallis test, but I'll leave those out of the conversation for now.

9.6 For More Information

1. Chapter 22 of Introduction to Modern Statistics is entitled "Inference for comparing many means" and does a very nice job of describing some key features of the randomization and ANOVA approaches. See Çetinkaya-Rundel and Hardin (2024).
2. Aickin and Gensler's 1996 article in the American Journal of Public Health called Adjusting for multiple testing when reporting research results: the Bonferroni vs Holm methods provides some logic for choosing Holm vs. Bonferroni approaches. Wikipedia also has nice explanations of the Bonferroni correction and the Holm approach, which may be of interest.
3. Wikipedia also describes and provides some of the mathematical foundations for Tukey's HSD test, which is also called several other things. Abdi and Williams (2010) has a nice summary (pdf) called Tukey's Honestly Significant Difference (HSD) Test.
4. The `lmboot` package in R (pdf) provides functions for bootstrap evaluation of ANOVA settings, and the author of the package, Megan Heyman, produced slides on Bootstrap in Linear Models: A Comprehensive R package in 2019.
5. The Kruskal-Wallis test is a rank-based test for assessing whether multiple samples come from the same distribution or not. The `kruskal.test()` function is the main tool in R. This page provides a pretty comprehensive example of its usage in R.

6. Visit this tutorial from the `bayestestR` package (part of `easystats`) for a nice introduction to the Bayesian model we're fitting here.

10 Storing Blood

10.1 R setup for this chapter

> **Note**
>
> Appendix A lists all R packages used in this book, and also provides R session information. Appendix B describes the 431-Love.R script, and demonstrates its use.

```
library(car)
library(ggdist)
library(janitor)
library(knitr)
library(naniar)
library(patchwork)
library(rstanarm)

library(easystats)
library(tidyverse)

source("data/Love-431.R")
theme_set(theme_bw())
```

10.2 Blood Storage data

> **Note**
>
> Appendix C provides further guidance on pulling data from other systems into R, while Appendix D gives more information (including download links) for all data sets used in this book.

The `storage.Rds` tibble we provide comes from the Cleveland Clinic Statistical Dataset Repository. The source[1] is Cata, Klein, and al. (2011). A version of the data is also available as

[1]Note that the first author's last name is Cata, not Cato, as the CCF repository suggests.

part of the medicaldata package in R (see Higgins (2023).)

From the CCF Repository, we have the following background:

> In cancer patients, perioperative blood transfusion has long been suspected of reducing long-term survival, but available evidence is inconsistent. An important factor associated with the deleterious effects of blood transfusion is the storage age of the transfused blood units. It is suspected that cancer recurrence may be worsened after the transfusion of older blood.
>
> This study evaluated the association between red blood cells (RBC) storage duration and biochemical prostate cancer recurrence after radical prostatectomy. Specifically, tested was the hypothesis that perioperative transfusion of allogeneic RBCs stored for a prolonged period is associated with earlier biochemical recurrence of prostate cancer after prostatectomy.
>
> Patients were assigned to 1 of 3 RBC age exposure groups on the basis of the terciles (ie, the 33rd and 66th percentiles) of the overall distribution of RBC storage duration if all their transfused units could be loosely characterized as of "younger," "middle," or "older" age.

The sample we study here includes data on 307 men who:

- had undergone radical prostatectomy and
- received transfusion during or within 30 days of the surgical procedure at Cleveland Clinic and
- had available PSA follow-up data, and
- who received solely allogeneic blood products (rather than a combination) and could be classified into one of the three RBC age exposure groups, and
- did not have any missing data on prostate volume.

The variable we'll study here is the prostate volume (`PVol`, in g) of the subjects in each of the three RBC Age Groups listed below, although this is just a convenient choice for us.

- `RBC_group` 1 was $\leq$ 13 days ("younger")
- `RBC_group` 2 was in between ("middle")
- `RBC_group` 3 was $\geq$ 18 days ("older")

```
storage <- read_rds("data/storage.Rds")

storage
```

```
# A tibble: 307 x 3
   id    RBC_group  PVol
   <chr> <fct>     <dbl>
 1 1     3          54
 2 2     3          43.2
 3 3     3         103.
 4 4     2          46
 5 5     2          60
 6 6     3          45.9
 7 7     3          42.6
 8 8     1          40.7
 9 9     1          45
10 10    2          67.6
# i 297 more rows
```

```
n_miss(storage)
```

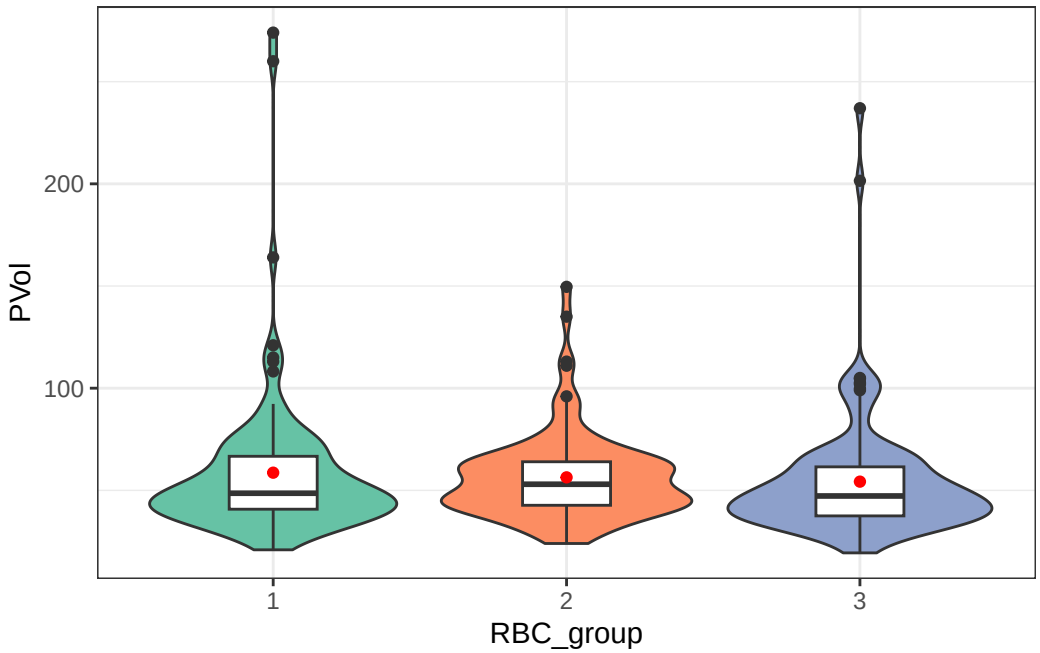
```
[1] 0
```

As noted, we have no missing data to overcome, and the `RBC_group` information (codes 1, 2 and 3) are classified as a factor, rather than a numeric value.

10.3 Exploratory Data Analysis

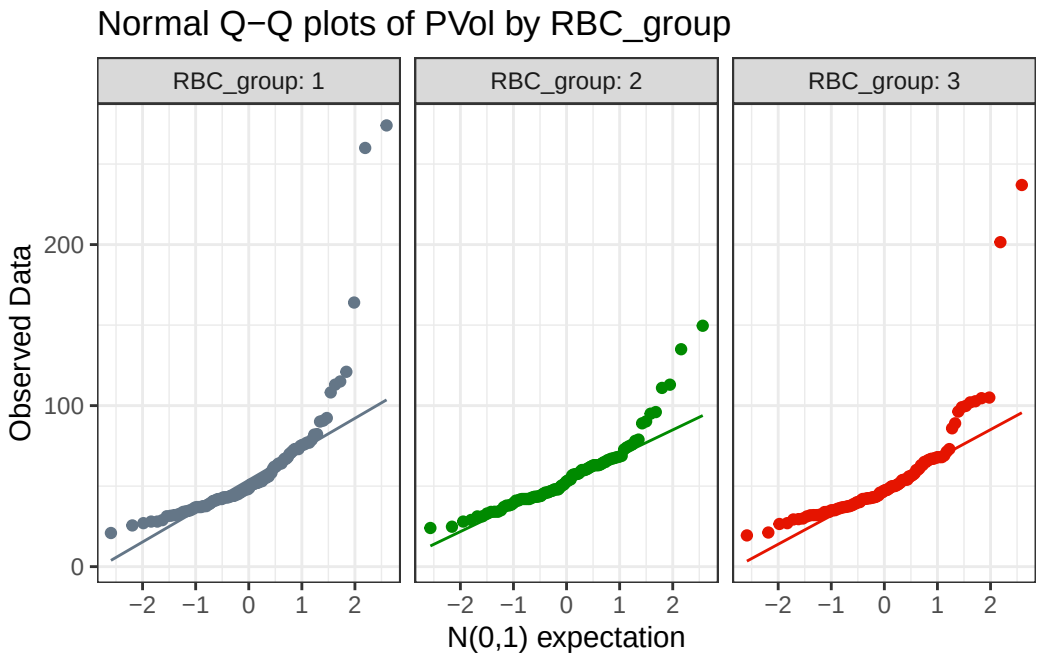
10.3.1 Visualization

```
ggplot(storage, aes(x = RBC_group, y = PVol)) +
  geom_violin(aes(fill = RBC_group)) +
  geom_boxplot(width = 0.3) +
  stat_summary(fun = mean, geom = "point", col = "red") +
  scale_fill_brewer(palette = "Set2") +
  guides(fill = "none")
```



As when we've worked with other comparisons using independent samples, our main question is whether each sample appears likely to have come from a Normal distribution, or not. Here, all three `RBC` groups show meaningful (right) skew on the `PVol` outcome. We can see this with other tools as well, such as a set of faceted Normal Q-Q plots.

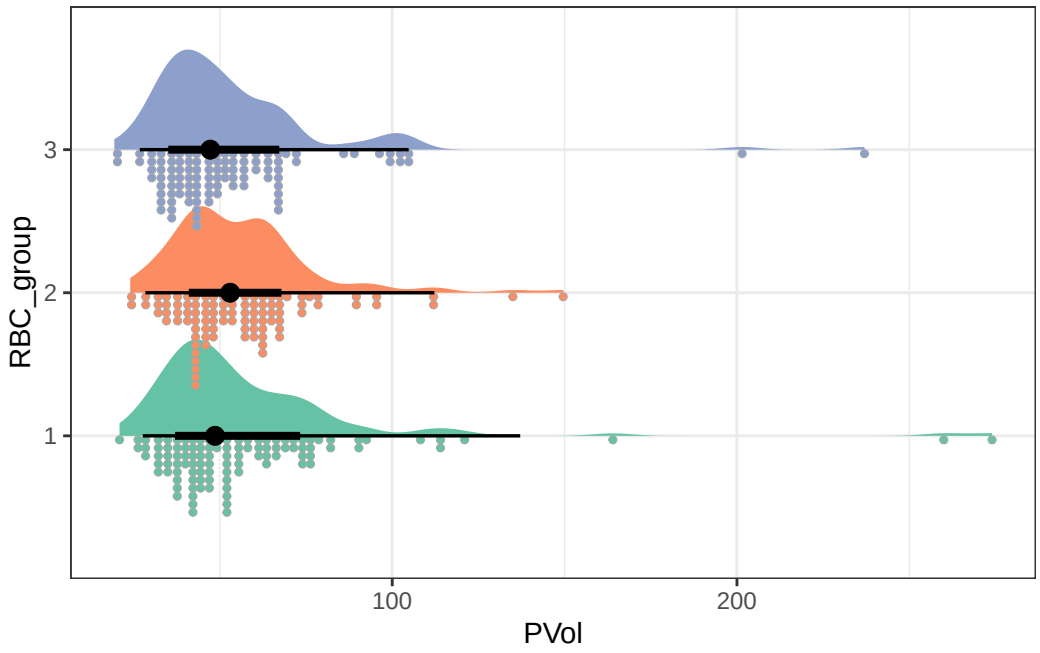
```
ggplot(storage, aes(sample = PVol, col = RBC_group)) +
  geom_qq() + geom_qq_line(aes(col = RBC_group)) +
  scale_color_metro_d() +
  facet_wrap(~ RBC_group, labeller = "label_both") +
  guides(color = "none") +
  labs(title = "Normal Q-Q plots of PVol by RBC_group",
       y = "Observed Data", x = "N(0,1) expectation")
```



All three plots curve up and away from their respective straight lines at both the top and bottom of the distribution, indicating right skew.

Another approach we might use here comes from the `ggdist` package.

```
ggplot(storage, aes(y = RBC_group, x = PVol, fill = RBC_group)) +
  stat_slab(aes(thickness = after_stat(pdf * n)), scale = 0.7) +
  stat_dotsinterval(side = "bottom", scale = 0.7, slab_linewidth = NA) +
  scale_fill_brewer(palette = "Set2") +
  guides(fill = "none") +
  theme_bw()
```



10.3.2 Numerical Summaries

Within each group, we see the mean `PVol` above the median `PVol`, also a sign of right skew.

```
storage |>
  reframe(lovedist(PVol), .by = RBC_group) |>
  kable(digits = 2)
```

RBC_group	n	miss	mean	sd	med	mad	min	q25	q75	max
3	104	0	54.27	30.06	47.25	15.64	19.4	37.50	61.50	237.0
2	97	0	56.37	21.27	53.00	16.31	24.0	42.70	64.00	149.6
1	106	0	58.66	36.72	48.60	17.20	20.9	40.78	66.67	274.0

10.4 Initial Linear Fit

Despite the apparent right skew, let's fit an OLS model (with `lm()`) first to the raw data, just for completeness.

```r
fit1 <- lm(PVol ~ RBC_group, data = storage)

summary(fit1)
```

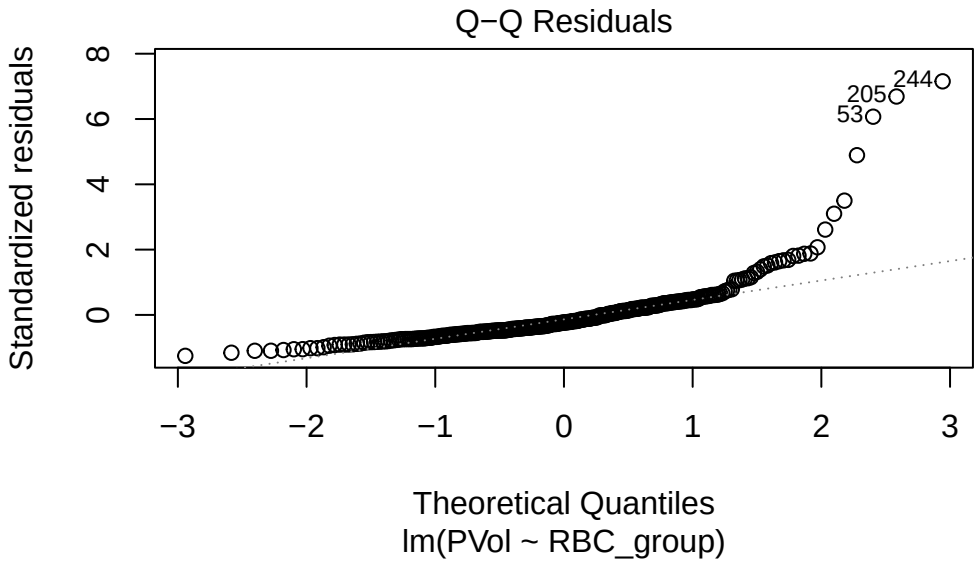
```
Call:
lm(formula = PVol ~ RBC_group, data = storage)

Residuals:
    Min      1Q  Median      3Q     Max
-37.756 -16.174  -7.056   7.989 215.344

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   58.656      2.937  19.968   <2e-16 ***
RBC_group2    -2.289      4.249  -0.539    0.591
RBC_group3    -4.382      4.174  -1.050    0.295
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 30.24 on 304 degrees of freedom
Multiple R-squared:  0.003615,  Adjusted R-squared:  -0.00294
F-statistic: 0.5515 on 2 and 304 DF,  p-value: 0.5767
```

Under the fitted model, how would we interpret the point estimates?

- The intercept shows the average value of `PVol` for subjects in `RBC_group1`, the group not otherwise identfied here, and that is 58.7 grams.
- The `RBC_group2` slope coefficient shows the average difference in `PVol` for subjects who were in `RBC_group2` as compared to the baseline category: `RBC_group1`. That mean difference is estimated to be -2.3 grams, so that we estimate the mean `PVol` of `RBC_group2` subjects to be 58.7 - 2.3 or 56.4 grams.
- Finally, the `RBC_group3` slope coefficient shows the average difference in `PVol` for subjects who were in `RBC_group3` as compared to `RBC_group1`. That mean difference is estimated to be -4.4 grams, so that we estimate the mean `PVol` of `RBC_group3` subjects to be 58.7 - 4.4 or 54.3 grams.

Does the assumption of Normality required by this fit work well, according to the diagnostic Q-Q plot of residuals shown below?

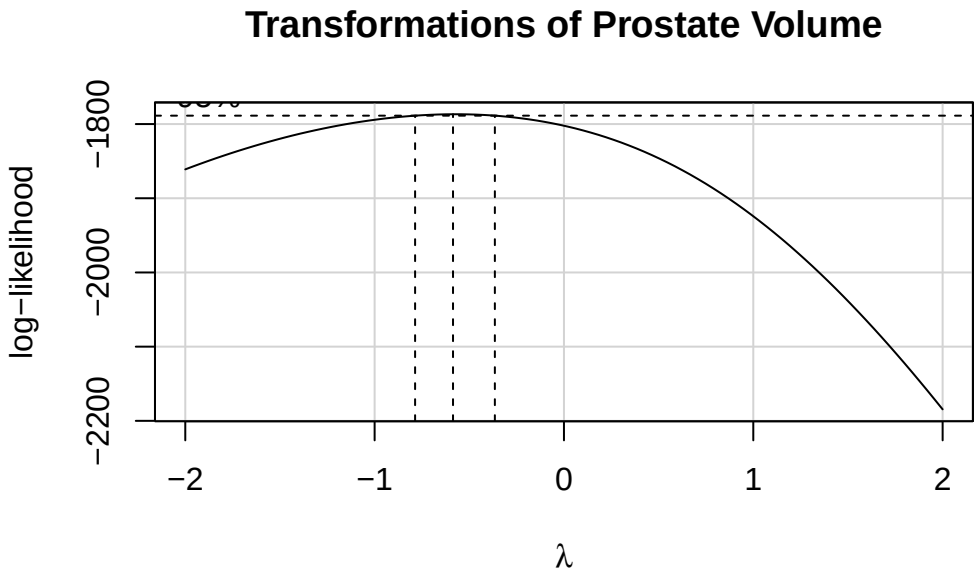
```
plot(fit1, which = 2)
```



No. We still have lots of right skew, especially on the higher end of the residuals. Let's consider transforming `PVol`.

10.5 Would a Transformation Be Useful?

```
boxCox(fit1, main = "Transformations of Prostate Volume")
```



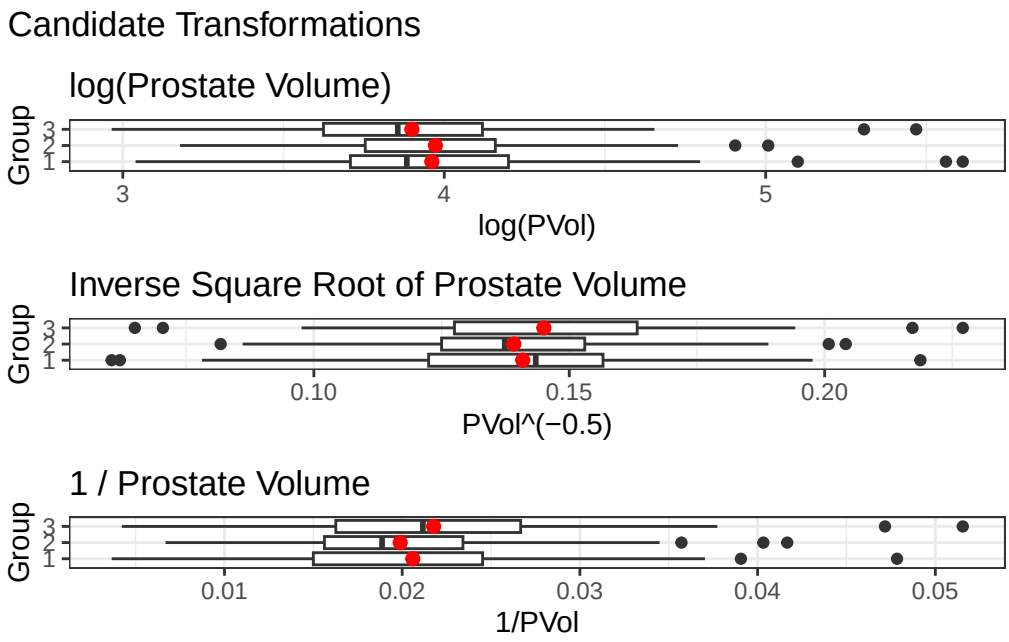

```
summary(powerTransform(fit1))$result
```

```
    Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
Y1 -0.5717863        -0.5   -0.7825024   -0.3610701
```

Given the suggested power of -0.5, we'll try an inverse square root, and compare it to the two nearest transformations, the log (power = 0) and the inverse (power = -1.) Here are a set of boxplots with means to help us.

```
p1 <- ggplot(storage, aes(x = log(PVol), y = RBC_group)) +
  geom_boxplot() +
  stat_summary(geom = "point", fun = "mean", size = 2, col = "red") +
  labs(title = "log(Prostate Volume)", y = "Group")

p2 <- ggplot(storage, aes(x = PVol^(-0.5), y = RBC_group)) +
  geom_boxplot() +
  stat_summary(geom = "point", fun = "mean", size = 2, col = "red") +
  labs(title = "Inverse Square Root of Prostate Volume", y = "Group")

p3 <- ggplot(storage, aes(x = 1/PVol, y = RBC_group)) +
  geom_boxplot() +
  stat_summary(geom = "point", fun = "mean", size = 2, col = "red") +
```

```
  labs(title = "1 / Prostate Volume", y = "Group")


p1 / p2 / p3 +
  plot_annotation(title = "Candidate Transformations")
```



I think we could possibly go with any of these transformations, but I will select the inverse square root (power = -0.5) for two reasons:

- it has candidate outliers on both sides of the distribution, so it looks a little more symmetric than the other choices, and
- I haven't used that transformation yet in this book.

10.5.1 Build in Transformation

Here are the means and standard deviations of the raw `PVol` values after our transformation.

```
storage |> group_by(RBC_group) |>
  summarise(mean = mean(PVol^(-0.5)), sd = sd(PVol^(-0.5)))
```

```
# A tibble: 3 x 3
  RBC_group  mean     sd
```

```
  <fct>     <dbl>  <dbl>
1 1         0.141 0.0274
2 2         0.139 0.0231
3 3         0.145 0.0272
```

I would want to multiply these values by a constant, say, 100, so that we can more easily look at the coefficients.

```
storage <- storage |>
  mutate(PV_trans = 100*(PVol^(-0.5)))

storage |> group_by(RBC_group) |>
  summarise(mean = mean(PV_trans), sd = sd(PV_trans))
```

```
# A tibble: 3 x 3
  RBC_group  mean    sd
  <fct>     <dbl> <dbl>
1 1          14.1  2.74
2 2          13.9  2.31
3 3          14.5  2.72
```

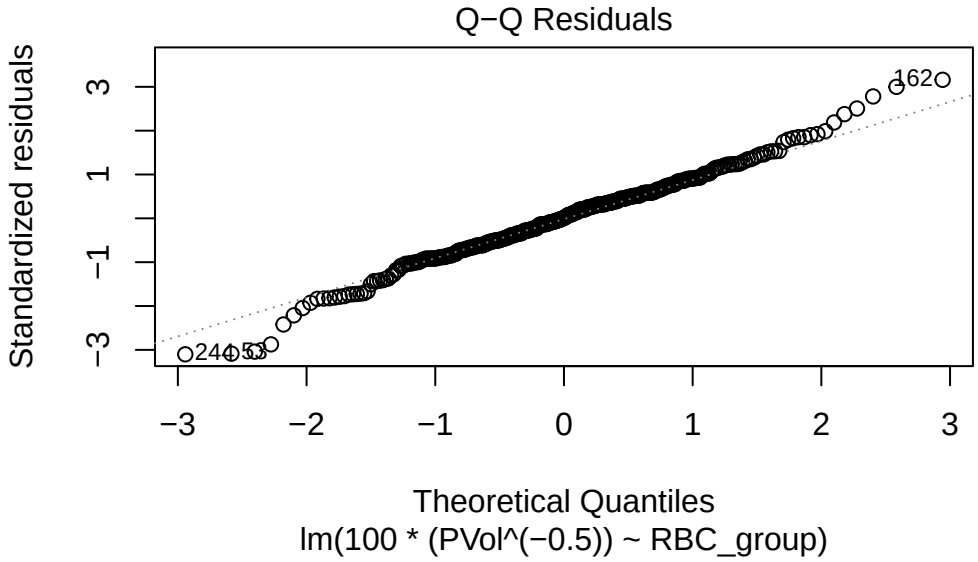
10.6 Linear Fit after Inverse Square Root Transformation

```
fit2 <- lm(100*(PVol^(-0.5)) ~ RBC_group, data = storage)

summary(fit2)
```

```
Call:
lm(formula = 100 * (PVol^(-0.5)) ~ RBC_group, data = storage)

Residuals:
    Min      1Q  Median      3Q     Max
-8.0487 -1.6050  0.0207  1.5161  8.1997

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  14.0899     0.2532  55.637   <2e-16 ***
RBC_group2   -0.1756     0.3664  -0.479    0.632
```

```
RBC_group3    0.4142     0.3599   1.151    0.251
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.607 on 304 degrees of freedom
Multiple R-squared:  0.008931,  Adjusted R-squared:  0.002411
F-statistic:  1.37 on 2 and 304 DF,  p-value: 0.2557
```

```r
plot(fit2, which = 2)
```



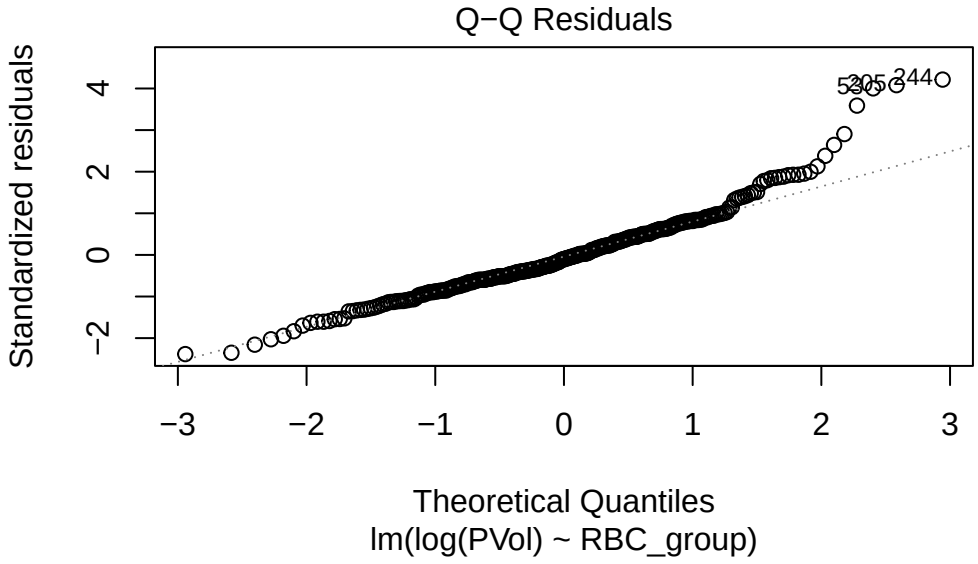
After this transformation, the Normal Q-Q plot of residuals is certainly closer to a Normal distribution. Is this also better than, say, the log transformation?

10.7 Linear Fit after Log Transformation

```r
fit3 <- lm(log(PVol) ~ RBC_group, data = storage)

summary(fit3)
```

```
Call:
lm(formula = log(PVol) ~ RBC_group, data = storage)

Residuals:
     Min       1Q   Median       3Q      Max
-0.93428 -0.23894 -0.04056  0.20666  1.65158

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.96155    0.03825 103.576   <2e-16 ***
RBC_group2   0.01084    0.05533   0.196    0.845
RBC_group3  -0.06199    0.05435  -1.141    0.255
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3938 on 304 degrees of freedom
Multiple R-squared:  0.006664,  Adjusted R-squared:  0.0001289
F-statistic:  1.02 on 2 and 304 DF,  p-value: 0.3619
```

```r
plot(fit3, which = 2)
```



I think I prefer the inverse square root transformation here.

10.8 Back-Transforming Predictions

Let's get that fit's (`fit2`) predictions for each `RBC_group`:

```
estimate_means(fit2, ci = 0.95)
```

```
We selected `by = c("RBC_group")`.

Warning in (function (object, at, cov.reduce = mean, cov.keep = get_emm_option("cov.keep"),
Auto-detection of the response transformation may be incorrect

Estimated Marginal Means

RBC_group | Mean |       SE |       95% CI
-----------------------------------------
1         | 0.14 | 2.53e-03 | [0.14, 0.15]
2         | 0.14 | 2.65e-03 | [0.13, 0.14]
3         | 0.15 | 2.56e-03 | [0.14, 0.15]

Marginal means estimated at RBC_group
```

Note that we could also use:

```
estimate_expectation(fit2, data = "grid", ci = 0.95)
```

```
Model-based Expectation

RBC_group | Predicted |   SE |         95% CI
-----------------------------------------------
1         |     14.09 | 0.25 | [13.59, 14.59]
2         |     13.91 | 0.26 | [13.39, 14.44]
3         |     14.50 | 0.26 | [14.00, 15.01]

Variable predicted: PVol
Predictors modulated: RBC_group
```

So, for example, our predicted **transformed** value of `PVol` among subjects in `RBC_group` 1 is 14.09, with 95% CI (13.59, 14.59).

To back-transform, we remember that the transformation we used was:

$$PVol_{transformed} = \frac{100}{\sqrt{PVol}}$$

To back out of the transformation we need to solve for `PVol` in terms of the transformed `PVol`. That is:

$$PVol_{transformed} = \frac{100}{\sqrt{PVol}} \frac{PVol_{transformed}}{100} = \frac{1}{\sqrt{PVol}} \sqrt{PVol} = \frac{100}{PVol_{transformed}} PVol = \left(\frac{100}{PVol_{transformed}} \right)$$

Again, our predicted **transformed** value of `PVol` among subjects in `RBC_group` 1 is 14.09, with 95% CI (13.59, 14.59).

Converting these values back to our original scale (in g) for `PVol` gives:

- point estimate `(100/14.09)^2` = 50.4
- lower bound `(100/14.59)^2` = 47, and
- upper bound `(100/13.59)^2` = 54.1

I'll leave it to you to do the same sort of calculation for the point estimate and 95% CI for RBC groups 2 and 3.

10.9 Pairwise Comparisons of Means

10.9.1 Bonferroni correction approach

```
con1 <- estimate_contrasts(fit2, contrast = "RBC_group",
                           ci = 0.95, p_adjust = "bonferroni")
```

```
Warning in (function (object, at, cov.reduce = mean, cov.keep = get_emm_option("cov.keep"),
Auto-detection of the response transformation may be incorrect
```

```
con1 |> kable(digits = 2)
```

Level1	Level2	Difference	CI_low	CI_high	SE	df	t	p
RBC_group1	RBC_group2	0.18	-0.71	1.06	0.37	304	0.48	1.00
RBC_group1	RBC_group3	-0.41	-1.28	0.45	0.36	304	-1.15	0.75
RBC_group2	RBC_group3	-0.59	-1.48	0.30	0.37	304	-1.60	0.33

Level1	Level2	Difference	CI_low	CI_high	SE	df	t	p

Note that each of these contrasts have confidence intervals which include zero. Is that still true with our other approaches?

10.9.2 Holm-Bonferroni approach

```
con2 <- estimate_contrasts(fit2, contrast = "RBC_group",
                           ci = 0.95, p_adjust = "holm")
```

```
Warning in (function (object, at, cov.reduce = mean, cov.keep = get_emm_option("cov.keep"),
Auto-detection of the response transformation may be incorrect
```

```
con2 |> kable(digits = 2)
```

Level1	Level2	Difference	CI_low	CI_high	SE	df	t	p
RBC_group1	RBC_group2	0.18	-0.71	1.06	0.37	304	0.48	0.63
RBC_group1	RBC_group3	-0.41	-1.28	0.45	0.36	304	-1.15	0.50
RBC_group2	RBC_group3	-0.59	-1.48	0.30	0.37	304	-1.60	0.33

10.9.3 Tukey HSD approach

```
con3 <- estimate_contrasts(fit2, contrast = "RBC_group",
                           ci = 0.95, p_adjust = "tukey")
```

```
Warning in (function (object, at, cov.reduce = mean, cov.keep = get_emm_option("cov.keep"),
Auto-detection of the response transformation may be incorrect
```

```
con3 |> kable(digits = 2)
```

Level1	Level2	Difference	CI_low	CI_high	SE	df	t	p
RBC_group1	RBC_group2	0.18	-0.69	1.04	0.37	304	0.48	0.88

Level1	Level2	Difference	CI_low	CI_high	SE	df	t	p
RBC_group1	RBC_group3	-0.41	-1.26	0.43	0.36	304	-1.15	0.48
RBC_group2	RBC_group3	-0.59	-1.46	0.28	0.37	304	-1.60	0.25

10.10 Bayesian Linear Model

We can, of course, run a Bayesian linear model on our transformed outcome, too.

```
set.seed(431)
fit4 <- stan_glm(PV_trans ~ RBC_group, data = storage, refresh = 0)
```

```
post4 <- describe_posterior(fit4, ci = 0.95)

print_md(post4, digits = 2)
```

Table 10.5: Summary of Posterior Distribution

Parameter	Median	95% CI	pd	ROPE	% in ROPE	Rhat	ESS
(Intercept)	14.09	[13.57, 14.61]	100%	[-0.26, 0.26]	0%	1.000	3480.00
RBC_group2	-0.17	[-0.91, 0.54]	68.40%	[-0.26, 0.26]	50.53%	1.000	3158.00
RBC_group3	0.41	[-0.32, 1.12]	86.10%	[-0.26, 0.26]	32.42%	1.000	3476.00

As before, we can obtain estimated predictions from this model with, for instance:

```
estimate_means(fit4, ci = 0.95)
```

```
We selected `by = c("RBC_group")`.

Estimated Marginal Means

RBC_group |  Mean |         95% CI |   pd
-----------------------------------------
1         | 14.09 | [13.57, 14.61] | 100%
2         | 13.92 | [13.40, 14.43] | 100%
3         | 14.51 | [14.00, 15.00] | 100%

Marginal means estimated at RBC_group
```

and again, we can back-transform these means and 95% credible intervals.

10.11 For More Information

1. Again, the Chapter on Inference for comparing many means in Çetinkaya-Rundel and Hardin (2024) is a good starting place for some people.
2. David Scott's sections on Box-Cox transformations, Tukey Ladder of Powers and on Analysis of Variance: One-Factor may be helpful to you.
3. Andrea Onofri had some useful examples using the `emmeans` package in her 2019 tutorial Stabilising transformations: how do I present my results?

11 Association and Regression

11.1 R setup for this chapter

> **Note**
>
> Appendix A lists all R packages used in this book, and also provides R session information. Appendix B describes the 431-Love.R script, and demonstrates its use.

```
library(GGally)
library(ggpubr)
library(ggstatsplot)
library(glue)
library(janitor)
library(knitr)
library(naniar)
library(patchwork)
library(rstanarm)

library(easystats)
library(tidyverse)

source("data/Love-431.R")
theme_set(theme_bw())
```

11.2 Data: US Wooden Roller Coasters

> **Note**
>
> Appendix C provides further guidance on pulling data from other systems into R, while Appendix D gives more information (including download links) for all data sets used in this book.

I was motivated to look at roller coaster[1] data by an older example from the Data and Story Library which also uses the Roller Coaster Database. Specifically, I used this URL to pull data on 110 currently operating (as of late June 2024) wooden roller coasters in the US meeting the following criteria:

- Classification = Roller Coaster
- Design = Sit Down
- Existing
- Location = United States
- Roller Coasters
- Current Status = Operating
- Type = Wood

```
coasters <- read_csv("data/coasters.csv", show_col_types = FALSE) |>
  janitor::clean_names()

head(coasters)
```

```
# A tibble: 6 x 10
  c_id  coaster_n          speed height duration opened park  town  state region
  <chr> <chr>              <dbl>  <dbl>    <dbl>  <dbl> <chr> <chr> <chr> <chr>
1 C-001 American Eagle      66      127      143   1981 Six ~ Gurn~ Illi~ Midwe~
2 C-002 American Thunder    48       82       NA   2008 Six ~ Eure~ Miss~ Midwe~
3 C-003 Apocalypse the Ri~  50.1     95      180   2009 Six ~ Vale~ Cali~ West
4 C-004 Arkansas Twister    NA       95       NA   1992 Magi~ Hot ~ Arka~ South
5 C-005 Beast               64.8    110      250   1979 King~ Mason Ohio  Midwe~
6 C-006 Big Dipper          NA       NA      100   1958 Camd~ Hunt~ West~ South
```

The 10 variables in the data are:

Variable	Description
`c_ID`	Coaster Identification Code
`coaster_N`	Name of Coaster
`speed`	Maximum speed, in miles per hour
`height`	Maximum height, in feet
`duration`	Length of ride, in seconds
`opened`	Year of opening
`park`	Name of amusement park
`town`	Name of town where park is located
`state`	State where park is located

[1]My favorite amusement park is Kennywood Park, just outside of Pittsburgh.

Variable	Description
`region`	Region of the US (4 categories)

11.2.1 Missing Data

The `naniar` package provides some excellent ways to count missing values in our data.

```
miss_var_summary(coasters)
```

```
# A tibble: 10 x 3
   variable  n_miss pct_miss
   <chr>      <int>    <num>
 1 duration      35    31.8
 2 speed         17    15.5
 3 height         9     8.18
 4 c_id           0     0
 5 coaster_n      0     0
 6 opened         0     0
 7 park           0     0
 8 town           0     0
 9 state          0     0
10 region         0     0
```

```
miss_case_table(coasters)
```

```
# A tibble: 4 x 3
  n_miss_in_case n_cases pct_cases
           <int>   <int>     <dbl>
1              0      64    58.2
2              1      32    29.1
3              2      13    11.8
4              3       1     0.909
```

As we've done in the past, we'll restrict ourselves to just the 64 coasters with complete data on all variables. We'll also set up the ages of the coasters, where age is just 2024 minus the year the coaster opened.

```
coast_cc <- coasters |>
  mutate(age = 2024 - opened) |>
  drop_na()

coast_cc
```

```
# A tibble: 64 x 11
   c_id  coaster_n   speed height duration opened park  town  state region   age
   <chr> <chr>       <dbl>  <dbl>    <dbl>  <dbl> <chr> <chr> <chr> <chr>  <dbl>
 1 C-001 American E~  66      127      143   1981 Six ~ Gurn~ Illi~ Midwe~    43
 2 C-003 Apocalypse~  50.1     95      180   2009 Six ~ Vale~ Cali~ West      15
 3 C-005 Beast        64.8    110      250   1979 King~ Mason Ohio  Midwe~    45
 4 C-007 Blue Streak  40       78      105   1964 Ceda~ Sand~ Ohio  Midwe~    60
 5 C-013 Classic Co~  50       55      105   1935 Wash~ Puya~ Wash~ West      89
 6 C-014 Coastersau~  32       40       50   2004 Lego~ Wint~ Flor~ South     20
 7 C-015 Comet        50       84      105   1946 Hers~ Hers~ Penn~ North~    78
 8 C-016 Comet        55       95      120   1994 Six ~ Quee~ New ~ North~    30
 9 C-017 Comet        25       37       84   1951 Wald~ Erie  Penn~ North~    73
10 C-019 Cyclone      60       75      110   1927 Luna~ Broo~ New ~ North~    97
# i 54 more rows
```

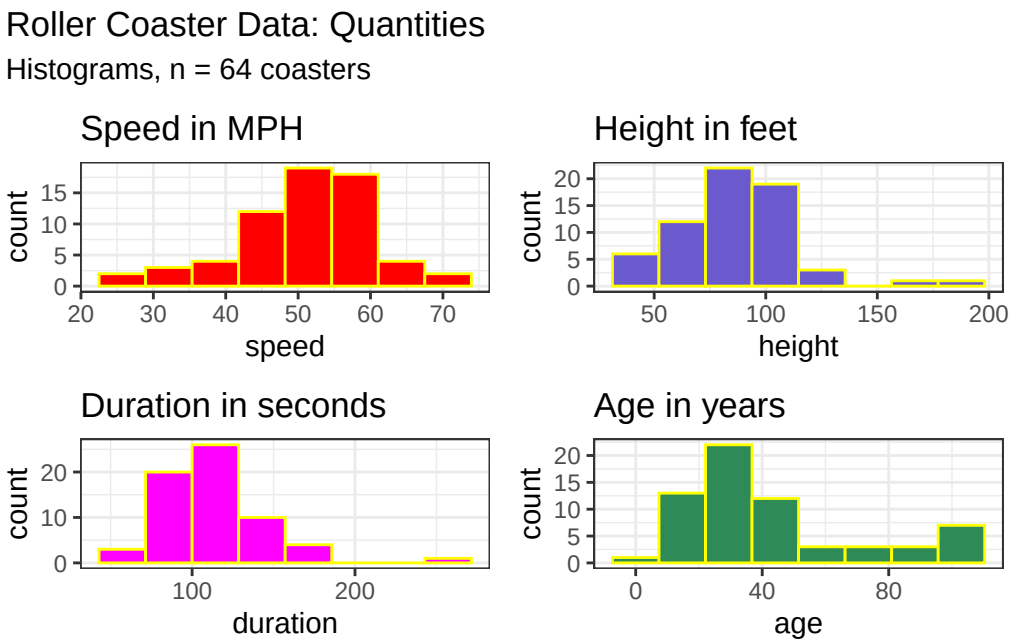
11.3 Exploratory Data Analysis

11.3.1 Visualizations

Although we're primarily interested in associations between variables, let's look briefly at the sample distributions for the four variables of interest.

```
p1 <- ggplot(coast_cc, aes(x = speed)) +
  geom_histogram(bins = 8, fill = "red", col = "yellow") +
  labs(title = "Speed in MPH")

p2 <- ggplot(coast_cc, aes(x = height)) +
  geom_histogram(bins = 8, fill = "slateblue", col = "yellow") +
  labs(title = "Height in feet")

p3 <- ggplot(coast_cc, aes(x = duration)) +
  geom_histogram(bins = 8, fill = "magenta", col = "yellow") +
  labs(title = "Duration in seconds")
```

```
p4 <- ggplot(coast_cc, aes(x = age)) +
  geom_histogram(bins = 8, fill = "seagreen", col = "yellow") +
  labs(title = "Age in years")

(p1 + p2) / (p3 + p4) +
  plot_annotation(title = "Roller Coaster Data: Quantities",
                  subtitle = "Histograms, n = 64 coasters")
```



11.3.2 Numeric Summaries

Here are distributional descriptions of our four main variables of interest.

```
res1 <- coast_cc |>
  reframe(lovedist(speed)) |>
  mutate(variable = "speed")
res2 <- coast_cc |>
  reframe(lovedist(height)) |>
  mutate(variable = "height")
res3 <- coast_cc |>
  reframe(lovedist(duration)) |>
  mutate(variable = "duration")
```

```
res4 <- coast_cc |>
  reframe(lovedist(age)) |>
  mutate(variable = "age")

rbind(res1, res2, res3, res4) |>
  relocate(variable) |>
  kable(digits = 2)
```

variable	n	miss	mean	sd	med	mad	min	q25	q75	max
speed	64	0	50.76	9.06	51.15	7.19	25	45.75	56.00	70
height	64	0	86.20	26.19	85.00	22.24	35	70.75	100.00	181
duration	64	0	112.67	31.71	111.00	28.17	50	90.00	120.00	250
age	64	0	42.72	27.57	33.50	20.76	1	25.00	51.25	104

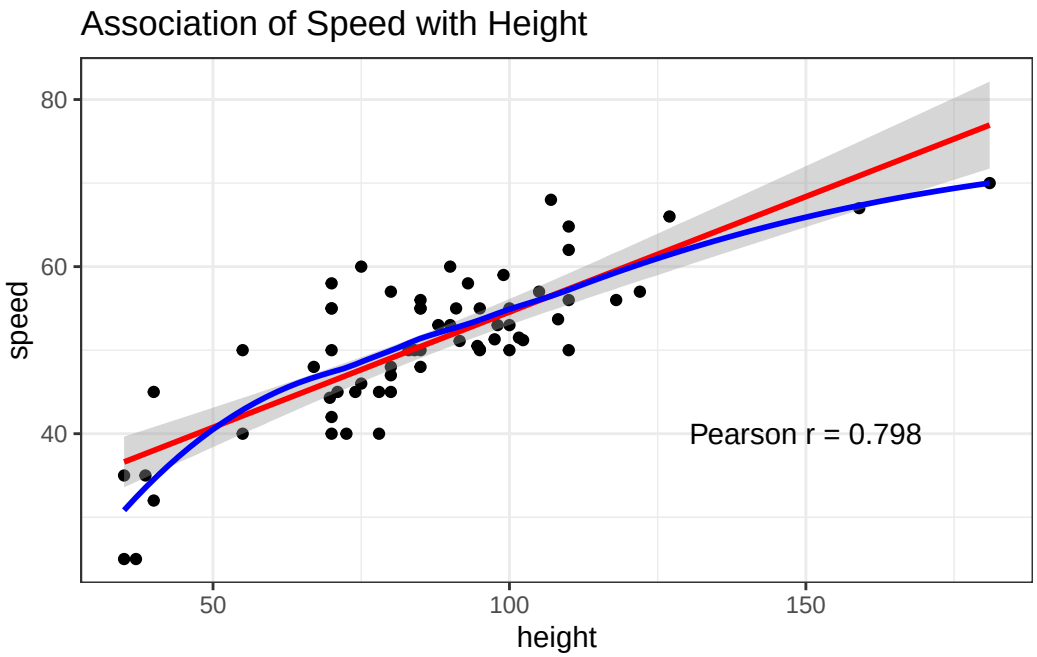
11.4 `height` - `speed` association

First, we'll consider the association of each coaster's `speed` with its `height`.

11.4.1 Scatterplot with Pearson correlation

Let's create a plot including both the linear fit (from `lm()`) and a loess smooth, as well as the Pearson correlation coefficient.

```
ggplot(coast_cc, aes(x = height, y = speed)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, col = "red") +
  geom_smooth(method = "loess", formula = y ~ x, se = FALSE, col = "blue") +
  labs(title = "Association of Speed with Height") +
  annotate("text", x = 150, y = 40,
           label = glue("Pearson r = ",
                        round(cor(coast_cc$height,
                                  coast_cc$speed),3)))
```

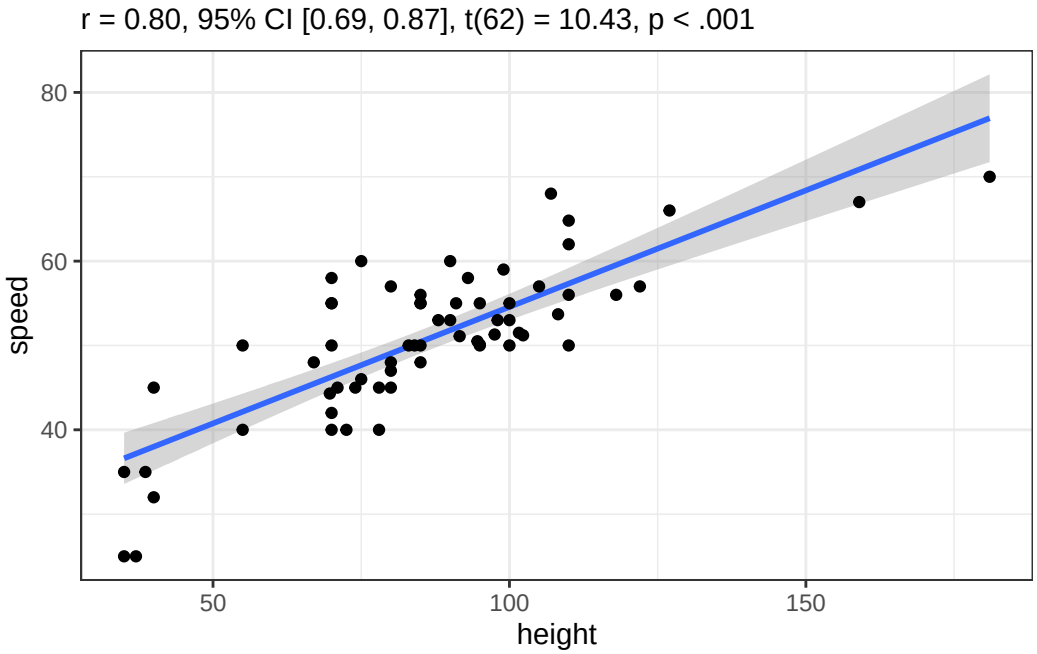


Another way to augment the scatterplot with similar information (other than the loess smooth) follows:

```
res6 <- cor_test(coast_cc, "height", "speed")

res6
```

```
Parameter1 | Parameter2 |    r |       95% CI | t(62) |         p
-----------------------------------------------------------------
height     |      speed | 0.80 | [0.69, 0.87] | 10.43 | < .001***

Observations: 64
```

```
plot(res6)
```



11.4.2 Spearman's rank correlation

This measure is the Pearson correlation of the rank scores of the two variables. It's mostly used for assessing whether or not an association is monotone.

```
cor_test(coast_cc, "height", "speed", method = "spearman")
```

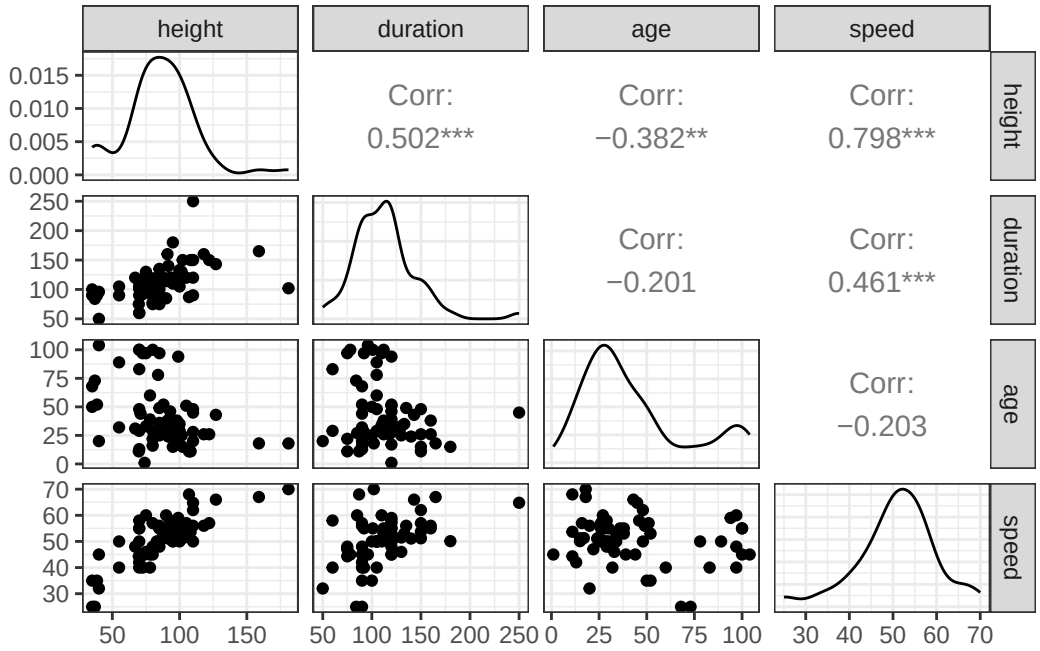
```
Parameter1 | Parameter2 |  rho |       95% CI |        S |         p
-------------------------------------------------------------------
height     |      speed | 0.72 | [0.57, 0.82] | 12371.84 | < .001***

Observations: 64
```

Numerous other measures of association are available. See this description, which is part of easystats' **correlation** package.

11.5 Scatterplot Matrix

We'll use the `ggpairs` function from the GGally package when I want to build a combined matrix of scatterplots and correlation coefficients, as displayed below:

```
ggpairs(coast_cc |> select(height, duration, age, speed))
```



11.6 Correlation Matrix

Behold the complete set of correlation coefficients using the Pearson's approach for the complete cases in the coasters data.

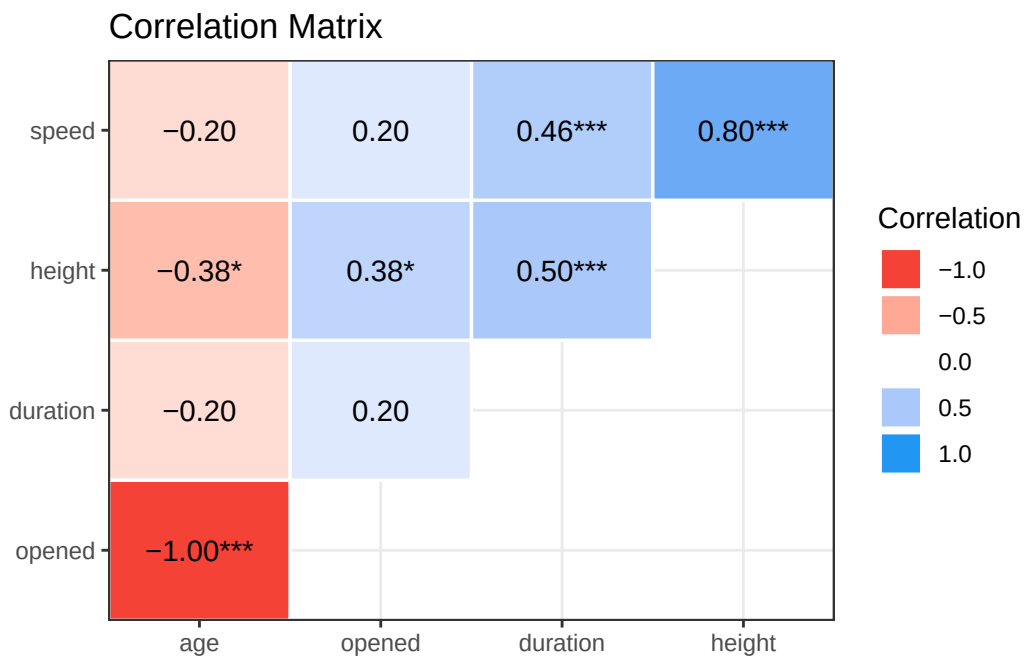
```
res_c <- correlation(coast_cc)

res_c
```

```
# Correlation Matrix (pearson-method)

Parameter1 | Parameter2 |     r |         95% CI | t(62) |         p
-------------------------------------------------------------------
```

```
speed      |     height |  0.80 | [ 0.69,  0.87] | 10.43 | < .001***
speed      |   duration |  0.46 | [ 0.24,  0.63] |  4.09 | < .001***
speed      |     opened |  0.20 | [-0.04,  0.43] |  1.63 | 0.429
speed      |        age | -0.20 | [-0.43,  0.04] | -1.63 | 0.429
height     |   duration |  0.50 | [ 0.29,  0.67] |  4.58 | < .001***
height     |     opened |  0.38 | [ 0.15,  0.57] |  3.26 | 0.011*
height     |        age | -0.38 | [-0.57, -0.15] | -3.26 | 0.011*
duration   |     opened |  0.20 | [-0.05,  0.43] |  1.62 | 0.429
duration   |        age | -0.20 | [-0.43,  0.05] | -1.62 | 0.429
opened     |        age | -1.00 | [-1.00, -1.00] |  -Inf | < .001***

p-value adjustment method: Holm (1979)
Observations: 64
```

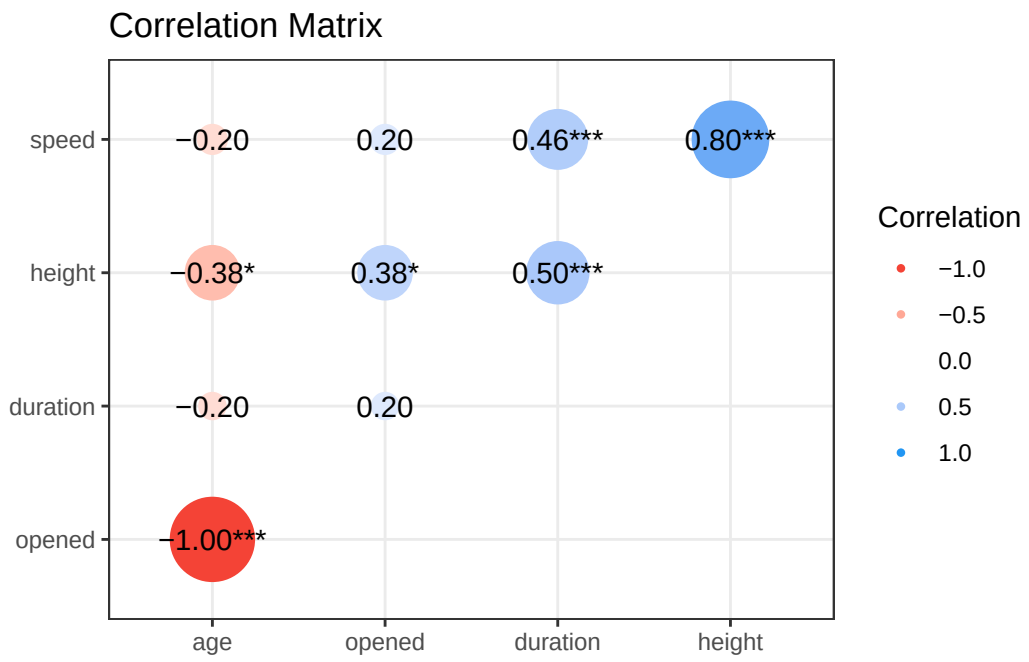
Once we have these results, we can plot them in several ways, in addition to the scatterplot matrix we've seen. These include:

```
plot(summary(res_c))
```



and

```
plot(summary(res_c), show_data = "points")
```



11.6.1 Using a different measure

We can also obtain a correlation matrix using Spearman's rank correlation, as follows:

```
res8 <- correlation(coast_cc, method = "spearman")

res8
```

```
# Correlation Matrix (spearman-method)

Parameter1 | Parameter2 |   rho |         95% CI |        S |         p
-------------------------------------------------------------------------
speed      |     height |  0.72 | [ 0.57,  0.82] | 12371.84 | < .001***
speed      |   duration |  0.45 | [ 0.23,  0.63] | 23898.81 | 0.001**
speed      |     opened |  0.17 | [-0.09,  0.40] | 36461.32 | 0.768
speed      |        age | -0.17 | [-0.40,  0.09] | 50898.68 | 0.768
height     |   duration |  0.60 | [ 0.41,  0.74] | 17308.98 | < .001***
height     |     opened |  0.39 | [ 0.15,  0.58] | 26851.92 | 0.010*
height     |        age | -0.39 | [-0.58, -0.15] | 60508.08 | 0.010*
```

```
duration   |     opened |  0.13 | [-0.12,  0.37] | 37811.21 | 0.768
duration   |        age | -0.13 | [-0.37,  0.12] | 49548.79 | 0.768
opened     |        age | -1.00 | [-1.00, -1.00] | 87360.00 | < .001***

p-value adjustment method: Holm (1979)
Observations: 64
```

11.7 Modeling Speed with Height

Let's start with a linear model (fit using ordinary least squares) to predict speed using height across our 64 coasters.

11.7.1 Fitting the Model

```
fit1 <- lm(speed ~ height, data = coast_cc)
```

11.7.2 Parameter Estimates

```
model_parameters(fit1, ci = 0.95) |> kable(digits = 2)
```

Parameter	Coefficient	SE	CI	CI_low	CI_high	t	df_error	p
(Intercept)	26.95	2.38	0.95	22.18	31.71	11.31	62	0
height	0.28	0.03	0.95	0.22	0.33	10.43	62	0

Good ways to interpret these coefficients include:

1. When comparing any two coasters whose heights are one foot apart, we expect a speed that is, on average, 0.28 miles per hour faster, for the taller coaster.
2. Under the fitted model, the average difference in coaster speed between two coasters whose height differs by ten feet is 2.8 MPH.

Note that we multiplied the height difference by ten, so we needed to multiply the average difference by 10, as well, in our second response.

11.7.3 Performance of the Model

How well does our model `fit1` perform? Some key summaries (applicable to any of the linear models we've fit with ordinary least squares so far in this book, too) are shown below.

```
model_performance(fit1)
```

```
# Indices of model performance

AIC     |    AICc |     BIC |    R2 | R2 (adj.) |  RMSE | Sigma
-----------------------------------------------------------------
403.859 | 404.259 | 410.336 | 0.637 |     0.631 | 5.416 | 5.503
```

While we'll focus on just a few of these measures in this chapter, specifically R2 and RMSE, here is a description of all of the summaries.

- AIC: Akaike's Information Criterion
- AICc: Second-order (or small sample) AIC with a correction for small sample sizes
- BIC: Bayesian Information Criterion

AIC, AICc and BIC are used when comparing one or more models for the same outcome. When comparing models fitted by maximum likelihood (like ordinary least squares linear models), the smaller the AIC or BIC, the better the fit. See the R documentation for these information criteria here.

- R2: r-squared value
- R2_adj: adjusted r-squared

The R-squared (R^2) measure, also called the coefficient of determination, describes how much of the variation in our outcome can be explained using our model (and its predictors.) R^2 falls between 0 and 1, and the closer it is to 1, the better the model fits our data. In a simple linear regression model such as we fit here (with one predictor), the R^2 value is also the square of the Pearson correlation coefficient. Note also that in discussing ANOVA previously, we called the R^2 value η^2.

An adjusted R-squared measure, is an index (so it's no longer a proportion of anything) used to compare different models (usually using different sets of predictors) that are fit to predict the same outcome. While adjusted R^2 usually falls between 0 and 1, it can also be negative, and its formula takes into account both the number of observations available and the number of predictors in the model. The idea is to reduce the temptation to overfit the data, by penalizing the R^2 value a little for each predictor. The adjusted R^2 measure is always no larger than R^2. See the documentation for these measures in the `performance` package (part of `easystats`.)

- RMSE: root mean squared error

- SIGMA: residual standard deviation, see insight::get_sigma()

The RMSE is the square root of the variance of the residuals and indicates the absolute fit of the model to the data (difference between observed data to model's predicted values). It can be interpreted as the standard deviation of the unexplained variance, and is in the same units as the response variable. Lower values indicate better model fit. (Source).

Linear models assume that their residuals are drawn from a Normal distribution with mean 0 and standard deviation equal to `sigma` (σ). The residual standard deviation `sigma` indicates the accuracy for a model to predict our outcome. σ also indicates that the predicted outcome will be within $\pm 1\sigma$ units of the observed outcome for approximately 68% of the data points, for example. See the discussion of `sigma` here.

11.8 Checking a Linear Model

The main assumptions of any linear model are:

- **linearity**: we assume that the outcome is linearly related to our predictor
- **constant variance (homoscedasticity)**: we assume that the variation of our outcome is about the same regardless of the value of our predictor
- **normal distribution**: we assume that the errors around the regression model at any specified values of the x-variables follow an approximately Normal distribution.

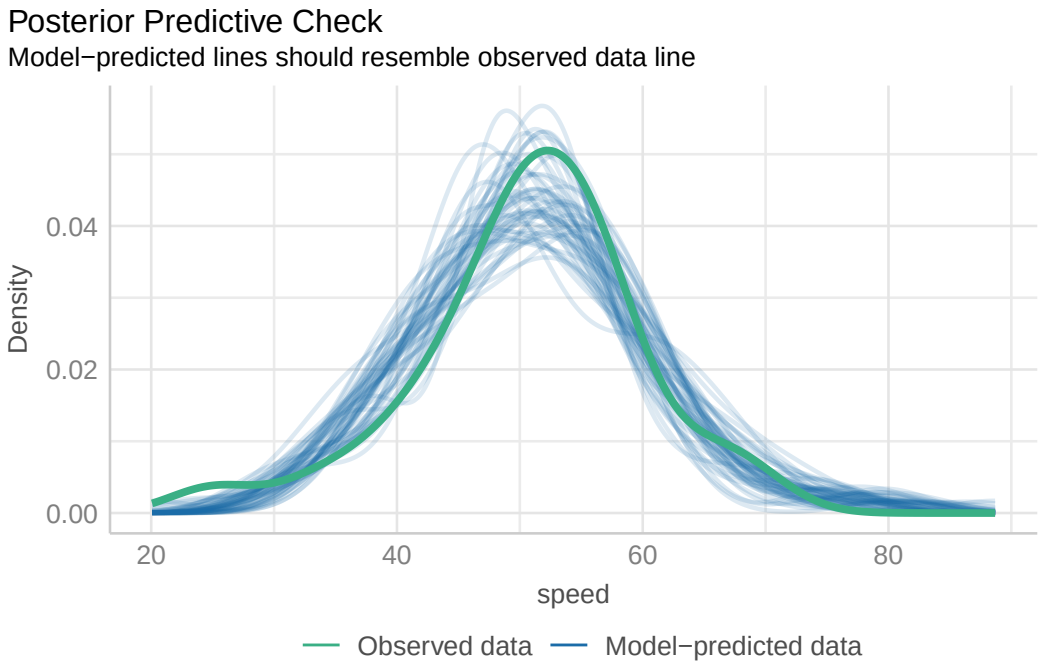
11.8.1 Posterior Predictive Checks

Next, we'll walk through what each of these plots is doing, one at a time.

```
diagnostic_plots <- plot(check_model(fit1, panel = FALSE))
```

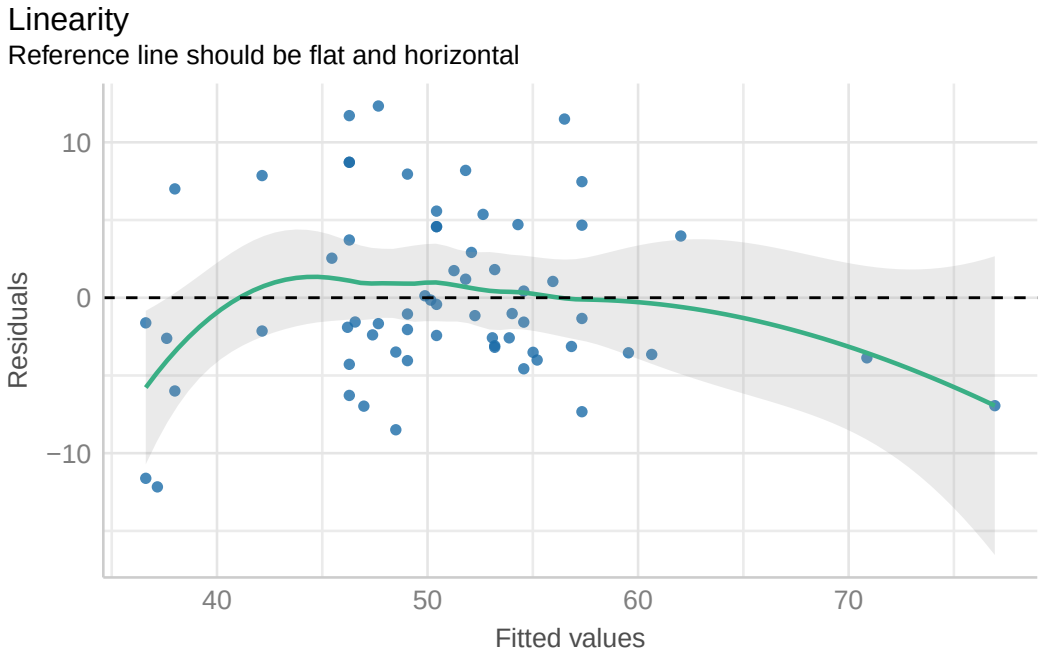
```
For confidence bands, please install `qqplotr`.
```

```
diagnostic_plots[[1]]
```



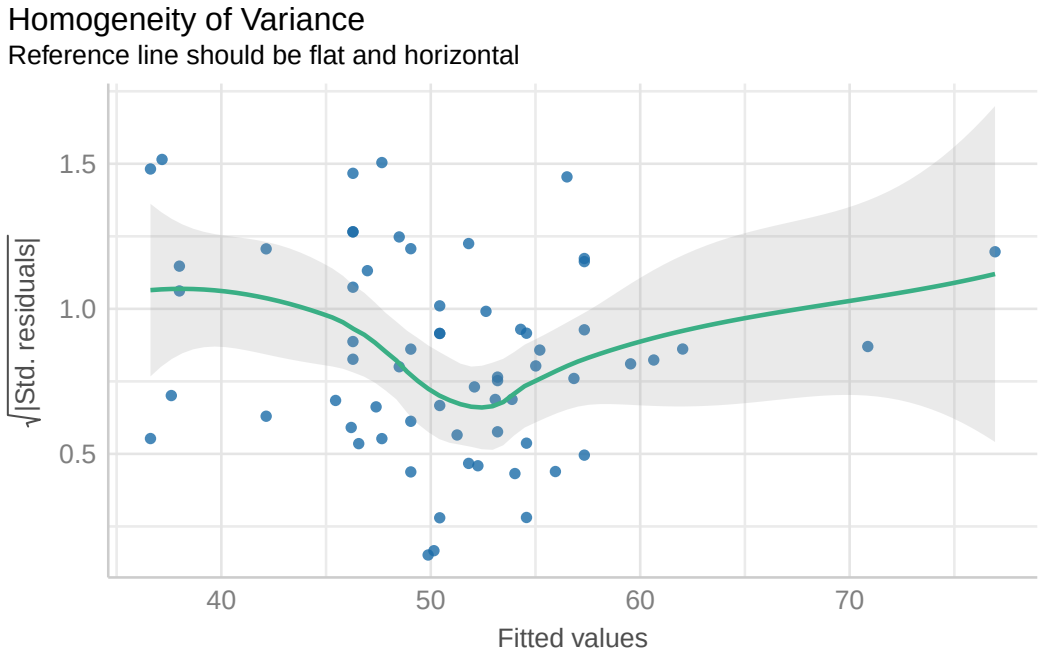
11.8.2 Checking Linearity

```
diagnostic_plots[[2]]
```



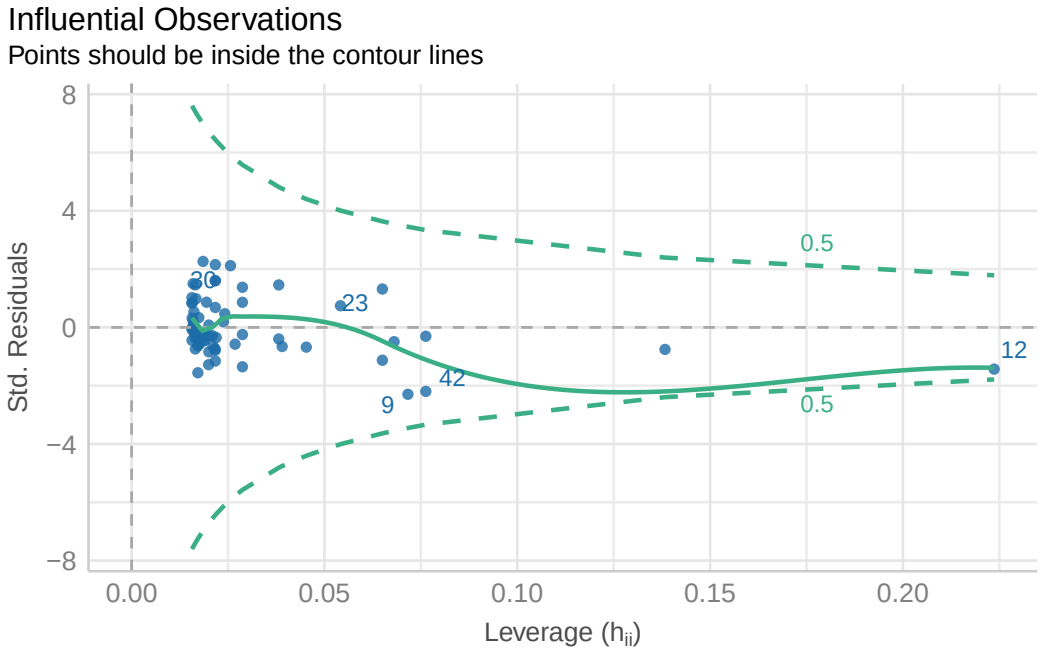
11.8.3 Checking for Homogeneity of Variance

```
diagnostic_plots[[3]]
```



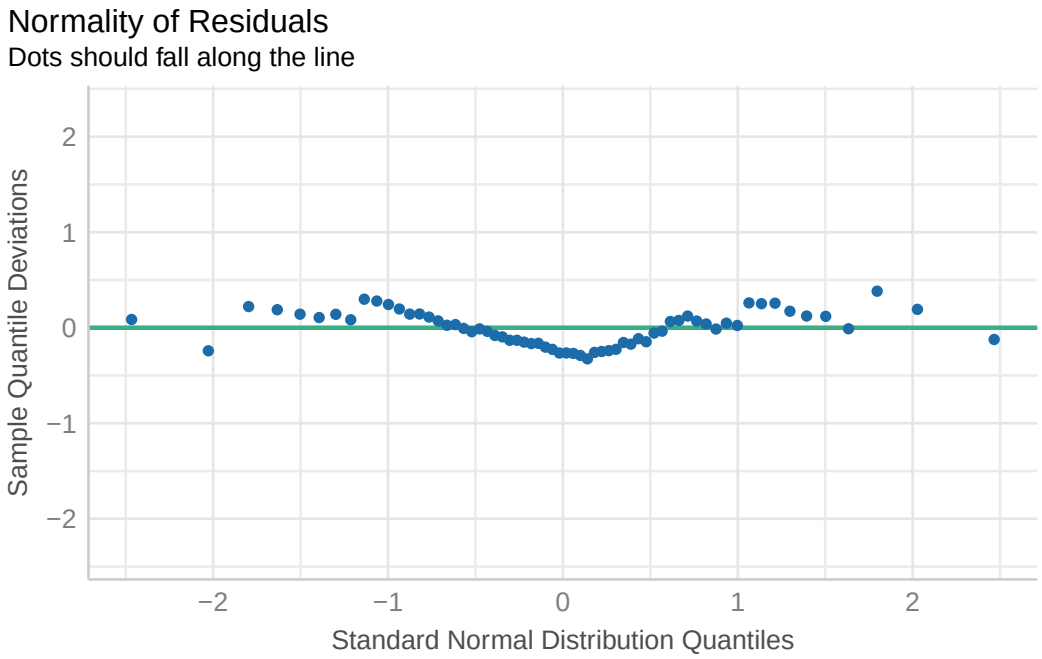
11.8.4 Checking for Influential Observations

```
diagnostic_plots[[4]]
```



11.8.5 Checking for Normality of Residuals

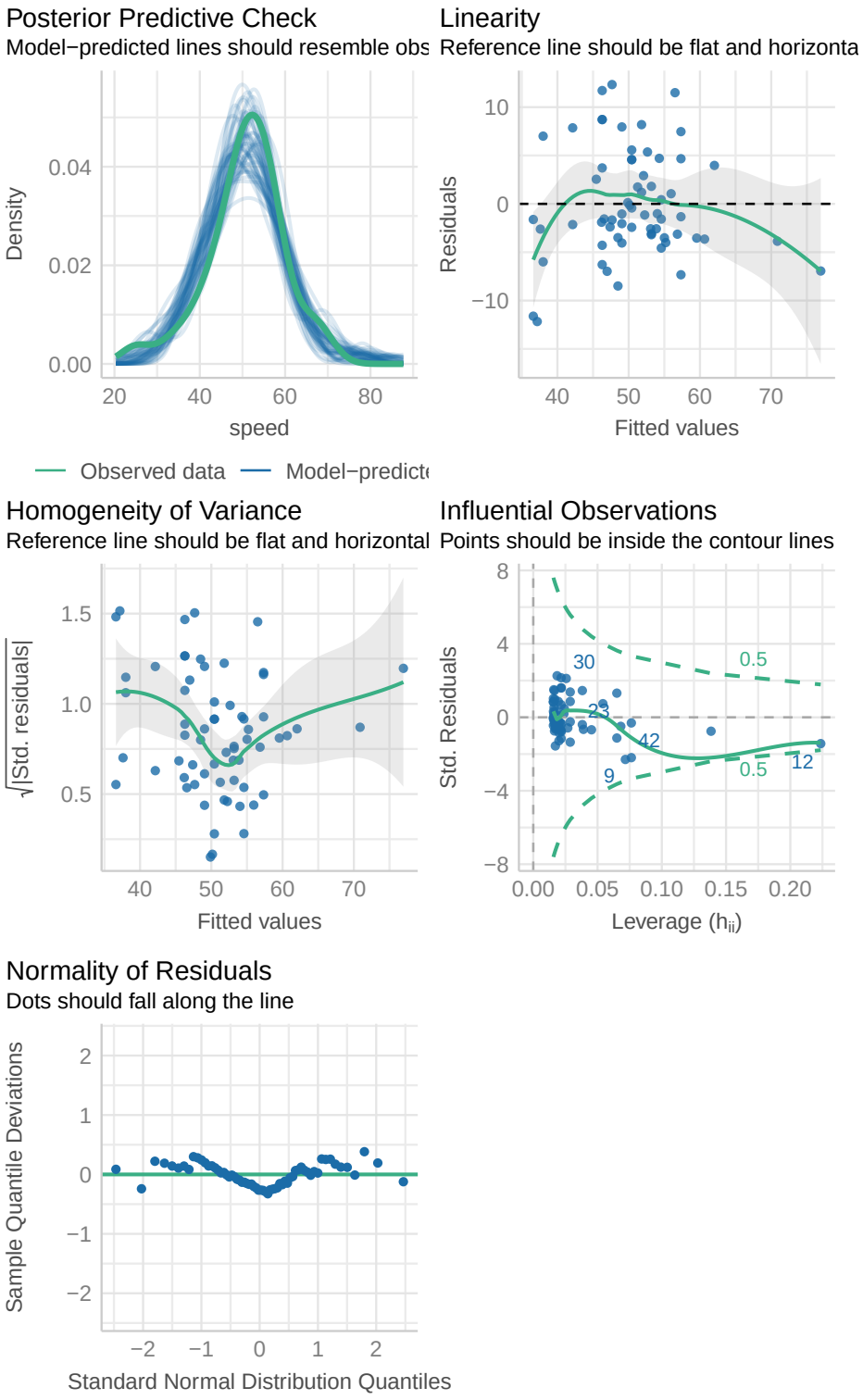
```
diagnostic_plots[[5]]
```



11.8.6 Running All of the Checks

> **Note**
>
> You will want to include
> `#| fig-height: 9`
> at the start of the R code chunk where you run `check_model()` in order to give the plots some vertical space to breathe.

```
check_model(fit1)
```


Posterior Predictive Check
Model−predicted lines should resemble obs
Density
0.04
0.02
0.00
20
40
60
80
speed
Observed data
Model−predicte
Linearity
Reference line should be flat and horizonta
Residuals
10
0
−10
40
50
60
70
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
1.5
1.0
0.5
40
50
60
70
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
8
4
0
−4
−8
0.00
0.05
0.10
0.15
0.20
Leverage (hii)
0.5
0.5
30
23
42
9
12
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
2
1
0
−1
−2
−2
−1
0
1
2
Standard Normal Distribution Quantiles

- See the check_model() web page at easystats (performance) for more details.

```
check_normality(fit1)
```

```
OK: residuals appear as normally distributed (p = 0.104).
```

```
check_heteroskedasticity(fit1)
```

```
OK: Error variance appears to be homoscedastic (p = 0.077).
```

11.9 Bayesian linear model for Speed with Height

11.9.1 Fitting the model

Here, we fit the Bayesian model, assuming weakly informative priors on the coefficients.

```
set.seed(431)
fit2 <- stan_glm(speed ~ height, data = coast_cc, refresh = 0)
```

11.9.2 Parameter Estimates

```
model_parameters(fit2, ci = 0.95) |> kable(digits = 2)
```

Parameter	Median	CI	CI_low	CI_high	pd	Rhat	ESS	Prior_Distribution	Prior_Location	Prior_Scale
(Intercept)	26.90	0.95	22.05	31.97	1	1	3770.92	normal	50.76	22.65
height	0.28	0.95	0.22	0.33	1	1	3516.32	normal	0.00	0.87

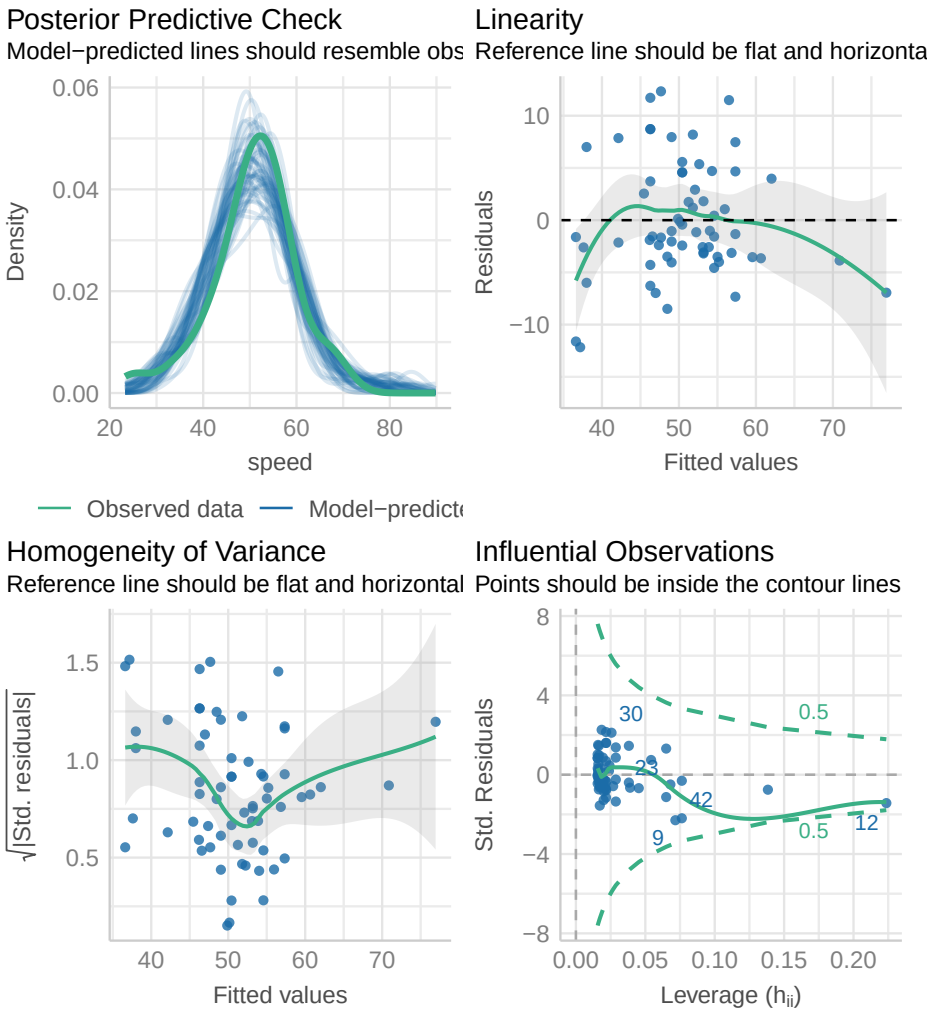
11.9.3 Performance of the Model

```
model_performance(fit2)
```

```
# Indices of model performance

ELPD     | ELPD_SE |   LOOIC | LOOIC_SE |    WAIC |    R2 | R2 (adj.) |  RMSE | Sigma
-------------------------------------------------------------------------------------
-202.245 |   5.520 | 404.491 |   11.039 | 404.407 | 0.630 |     0.614 | 5.416 | 5.552
```

11.9.4 Checking the Model

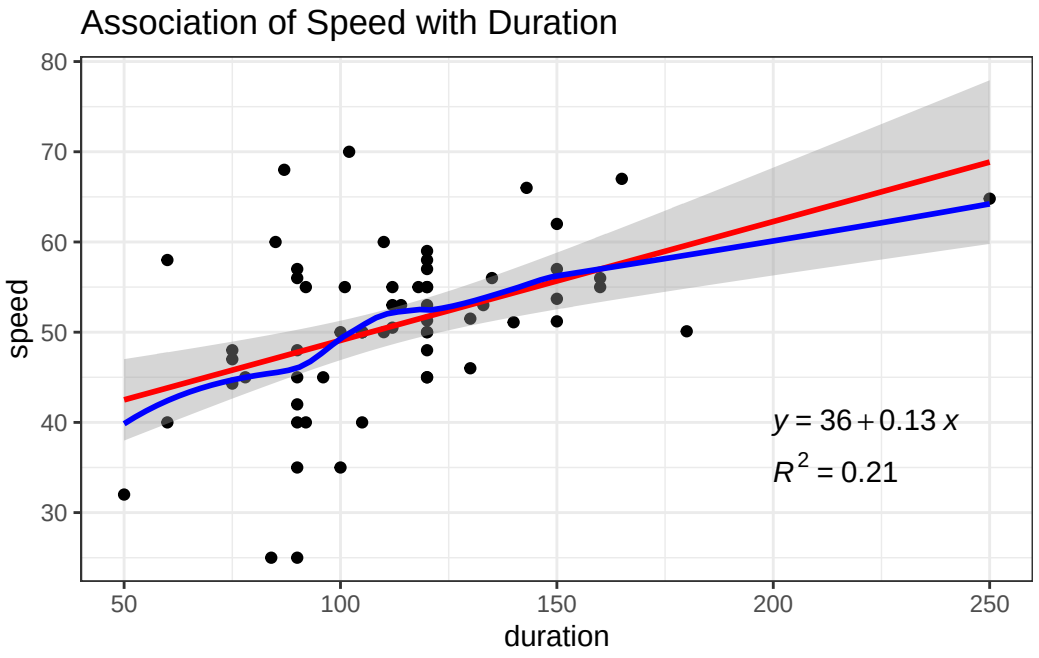
```
check_model(fit2)
```


Posterior Predictive Check
Model−predicted lines should resemble obs
Density
speed
Observed data
Model−predicte
Linearity
Reference line should be flat and horizonta
Residuals
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
Leverage (hii)
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
Standard Normal Distribution Quantiles

11.10 Predicting Speed with Duration

Next, consider the relationship between `duration` and `speed`.

```
ggplot(coast_cc, aes(x = duration, y = speed)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, col = "red") +
  geom_smooth(method = "loess", formula = y ~ x, se = FALSE, col = "blue") +
  labs(title = "Association of Speed with Duration") +
  stat_regline_equation(label.x = 200, label.y = 40) +
  stat_cor(aes(label = after_stat(rr.label)),
    label.x = 200, label.y = 35)
```



11.10.1 Ordinary Least Squares

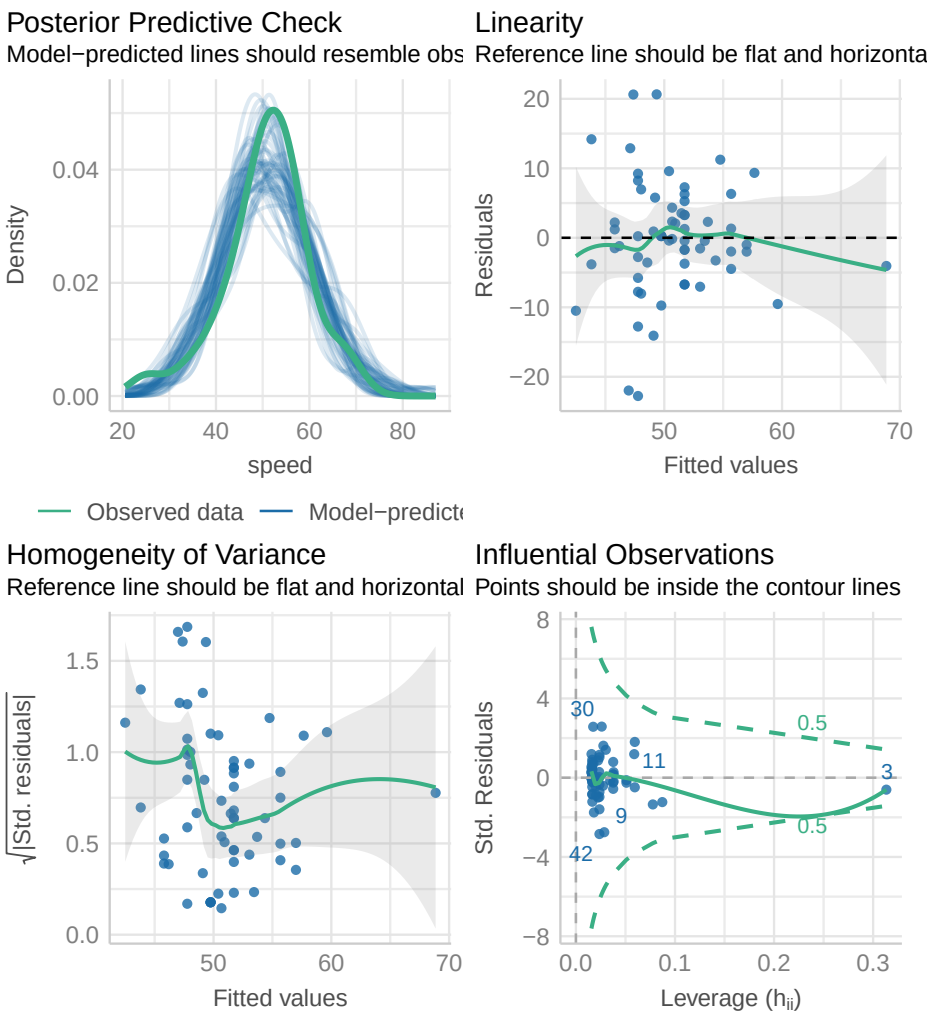
```
fit3 <- lm(speed ~ duration, data = coast_cc)

model_parameters(fit3, ci = 0.95) |> kable(digits = 2)
```

Parameter	Coefficient	SE	CI	CI_low	CI_high	t	df_error	p
(Intercept)	35.91	3.77	0.95	28.38	43.44	9.53	62	0
duration	0.13	0.03	0.95	0.07	0.20	4.09	62	0

```
model_performance(fit3) |> kable()
```

AIC	AICc	BIC	R2	R2_adjusted	RMSE	Sigma
453.4355	453.8355	459.9122	0.2126508	0.1999516	7.97771	8.105361

```
check_model(fit3)
```


Posterior Predictive Check
Model−predicted lines should resemble obs
Density
0.04
0.02
0.00
20
40
60
80
speed
Observed data
Model−predicte
Linearity
Reference line should be flat and horizonta
Residuals
20
10
0
−10
−20
50
60
70
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
1.5
1.0
0.5
0.0
50
60
70
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
8
4
0
−4
−8
30
11
9
42
0.5
0.5
3
0.0
0.1
0.2
0.3
Leverage (hii)
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
2
1
0
−1
−2
−3
−2
−1
0
1
2
Standard Normal Distribution Quantiles

11.10.2 Bayesian linear model fit

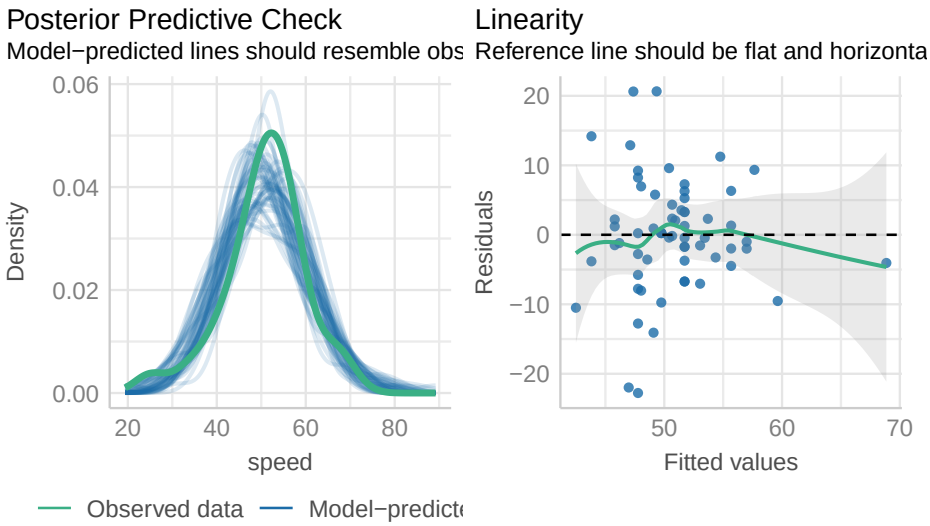
```
set.seed(431)
fit4 <- stan_glm(speed ~ duration, data = coast_cc, refresh = 0)

model_parameters(fit4, ci = 0.95) |> kable(digits = 2)
```

Parameter	Median	CI	CI_low	CI_high	pd	Rhat	ESS	Prior_Distribution	Prior_Location	Prior_Scale
(Intercept)	35.95	0.95	28.34	43.33	1	1	3722.28	normal	50.76	22.65
duration	0.13	0.95	0.07	0.20	1	1	3704.16	normal	0.00	0.71

```
model_performance(fit4)
```

```
# Indices of model performance

ELPD     | ELPD_SE |   LOOIC | LOOIC_SE |    WAIC |    R2 | R2 (adj.) |  RMSE | Sigma
-------------------------------------------------------------------------------------
-227.090 |   7.486 | 454.181 |   14.972 | 454.127 | 0.209 |     0.180 | 7.978 | 8.168
```

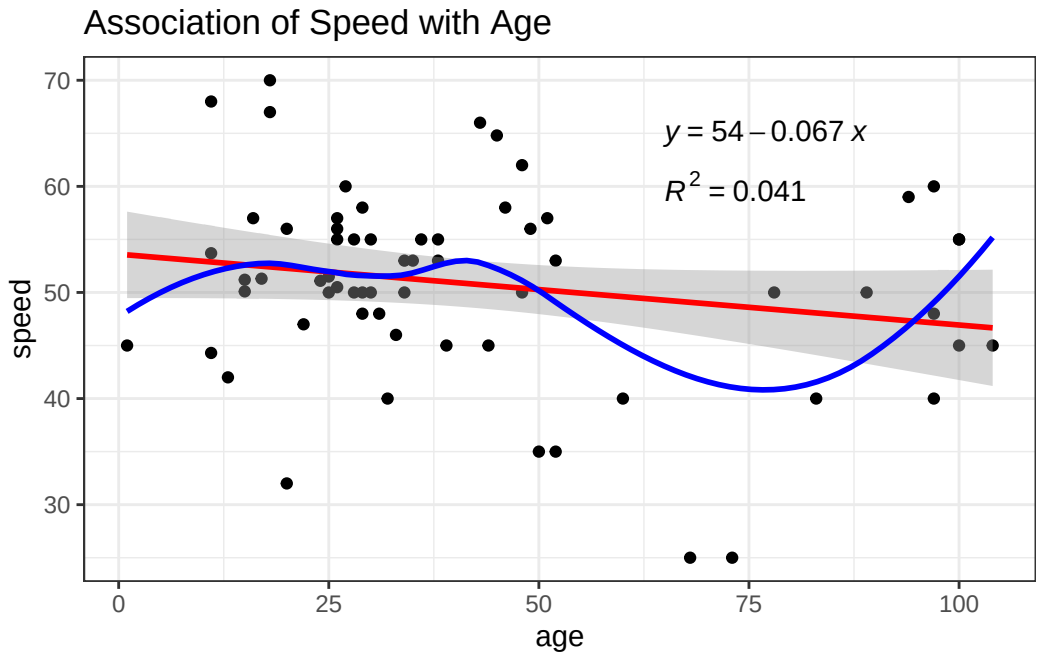
```
check_model(fit4)
```


Posterior Predictive Check
Model−predicted lines should resemble obs
Density
speed
Observed data
Model−predicte
Linearity
Reference line should be flat and horizonta
Residuals
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
Leverage (hii)
30
11
9
42
3
0.5
0.5
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
Standard Normal Distribution Quantiles

11.11 Predicting Speed with Age

Finally, are newer coasters faster? Consider the relationship between `age` and `speed`.

```
ggplot(coast_cc, aes(x = age, y = speed)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, col = "red") +
  geom_smooth(method = "loess", formula = y ~ x, se = FALSE, col = "blue") +
  labs(title = "Association of Speed with Age") +
  stat_regline_equation(label.x = 65, label.y = 65) +
  stat_cor(aes(label = after_stat(rr.label)),
    label.x = 65, label.y = 60)
```



11.11.1 Ordinary Least Squares Fit

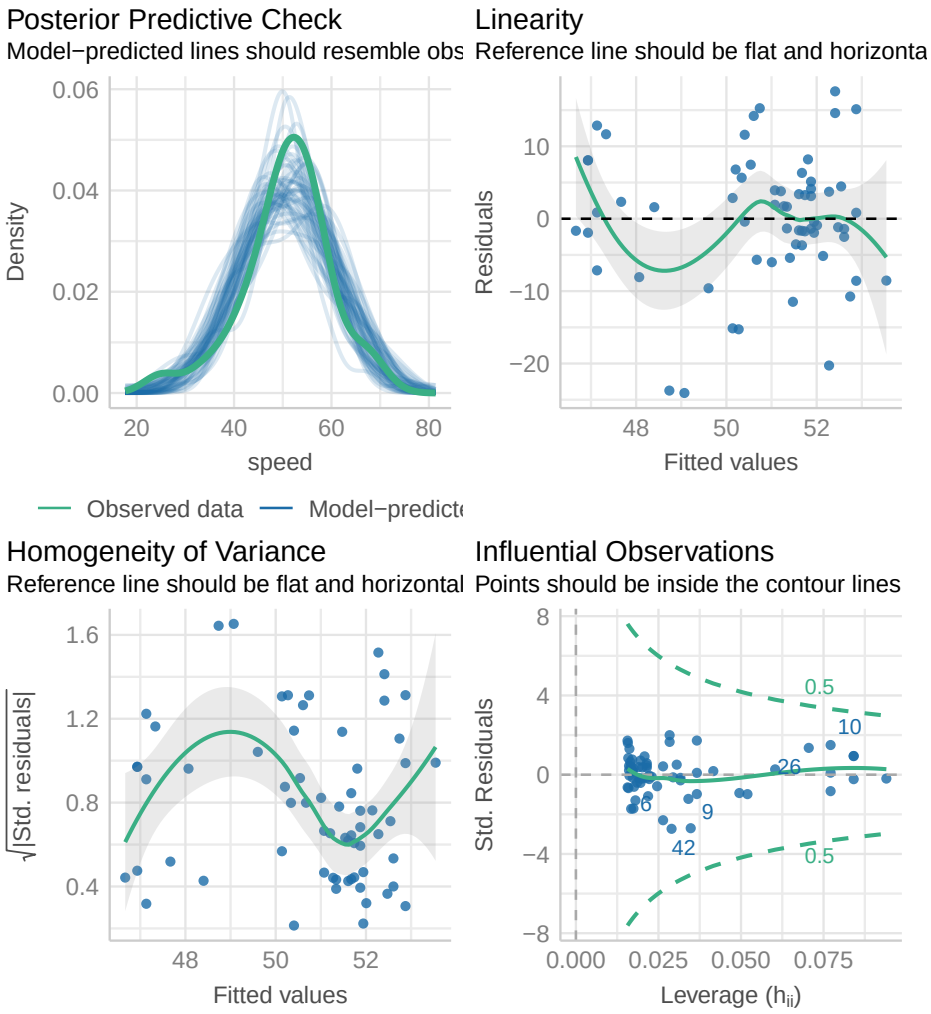
```
fit5 <- lm(speed ~ age, data = coast_cc)

model_parameters(fit5, ci = 0.95) |> kable(digits = 2)
```

Parameter	Coefficient	SE	CI	CI_low	CI_high	t	df_error	p
(Intercept)	53.61	2.07	0.95	49.47	57.75	25.86	62	0.00
age	-0.07	0.04	0.95	-0.15	0.01	-1.63	62	0.11

```
model_performance(fit5)
```

```
# Indices of model performance

AIC     |    AICc |     BIC |    R2 | R2 (adj.) |  RMSE | Sigma
-----------------------------------------------------------------
466.039 | 466.439 | 472.516 | 0.041 |     0.026 | 8.803 | 8.944
```

```
check_model(fit5)
```


Posterior Predictive Check
Model−predicted lines should resemble obs
Density
speed
Observed data
Model−predicte
Linearity
Reference line should be flat and horizonta
Residuals
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
Leverage (hii)
0.5
10
26
6
9
42
0.5
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
Standard Normal Distribution Quantiles

11.11.2 Bayesian linear model fit

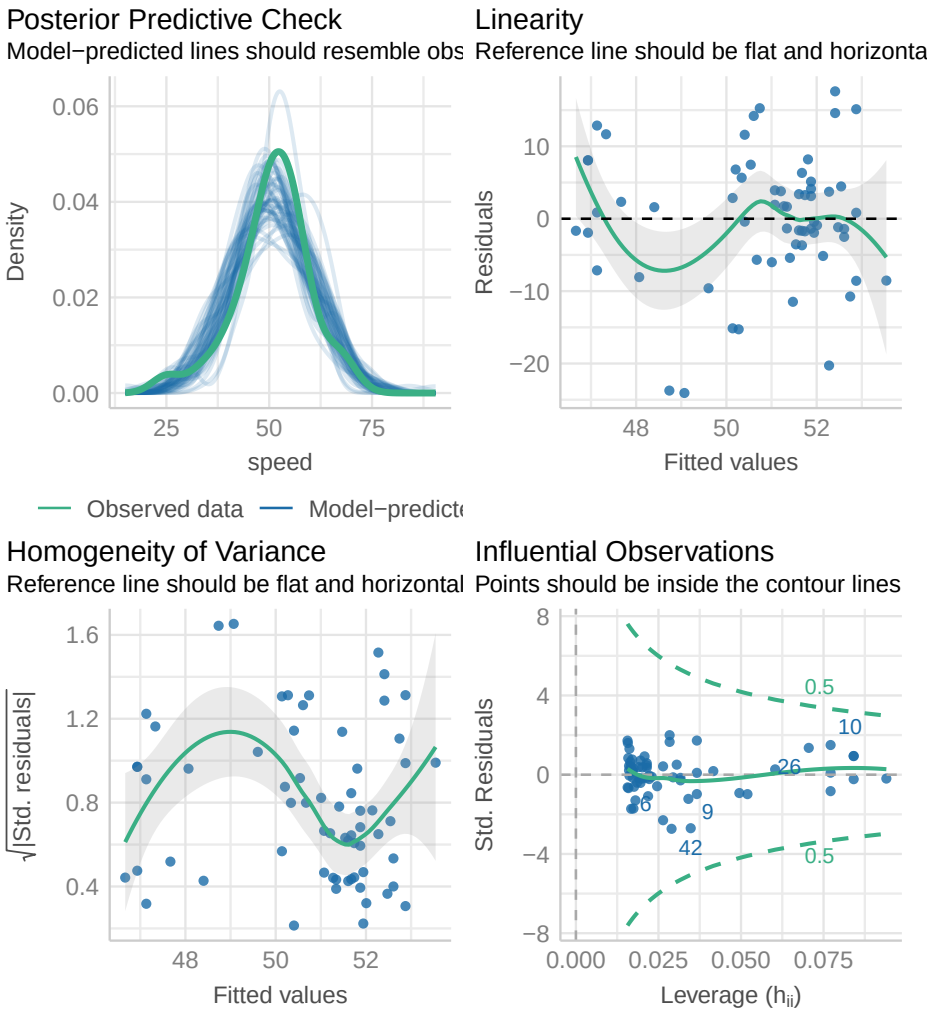
```
set.seed(431)
fit6 <- stan_glm(speed ~ age, data = coast_cc, refresh = 0)

model_parameters(fit6, ci = 0.95) |> kable(digits = 2)
```

Parameter	Median	CI	CI_low	CI_high	pd	Rhat	ESS	Prior_Distribution	Prior_Location	Prior_Scale
(Intercept)	53.54	0.95	49.57	57.64	1.00	1	3569.63	normal	50.76	22.65
age	-0.07	0.95	-0.15	0.01	0.95	1	3753.96	normal	0.00	0.82

```
model_performance(fit6)
```

```
# Indices of model performance

ELPD     | ELPD_SE |   LOOIC | LOOIC_SE |    WAIC |    R2 | R2 (adj.) |  RMSE | Sigma
-------------------------------------------------------------------------------------
-233.199 |   6.670 | 466.398 |   13.341 | 466.367 | 0.039 |    -0.003 | 8.803 | 8.971
```

```
check_model(fit6)
```


Posterior Predictive Check
Model−predicted lines should resemble obs
Density
speed
Observed data
Model−predicte
Linearity
Reference line should be flat and horizonta
Residuals
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
Leverage (hii)
0.5
10
26
6
9
42
0.5
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
Standard Normal Distribution Quantiles

11.12 For More Information

1. Chapter 7 of Çetinkaya-Rundel and Hardin (2024) is about linear regression with a single predictor.
2. This tutorial from the `bayestestR` package (part of `easystats`) uses simple linear regression with both `lm()` and Bayesian approaches.
3. Here is a nice tutorial using R on Simple Linear Regression that addresses some of this chapter's issues.
4. Wikipedia's page on Linear regression provides lots of useful details.
5. Plotting Functions for the correlation package
6. Correlation Types

12 Studying Craters

12.1 R setup for this chapter

> **Note**
>
> Appendix A lists all R packages used in this book, and also provides R session information. Appendix B describes the 431-Love.R script, and demonstrates its use.

```
library(glue)
library(haven)
library(janitor)
library(knitr)
library(naniar)
library(patchwork)
library(rstanarm)

library(easystats)
library(tidyverse)

source("data/Love-431.R")
theme_set(theme_bw())
```

12.2 Data from an SPSS File: Craters

> **Note**
>
> Appendix C provides further guidance on pulling data from other systems into R, while Appendix D gives more information (including download links) for all data sets used in this book.

These data are adapted from the Data and Story Library, which provides the following introduction to the data which were originally gathered from the Earth Impact Database:

> Meteor Crater in Arizona was the first recognized impact crater and was identified as such only in the 1920s. With the help of satellite images, more and more craters have been identified; now more than 180 are known. These, of course, are only a small sample of all the impacts the earth has experienced: Only 29% of earth's surface is land, and many craters have been covered or eroded away. Astronomers have recog-nized a roughly 35 million-year cycle in the frequency of cratering, although the cause of this cycle is not fully understood.
>
> The data hold information about craters. craters from the most recent 35Ma (million years) may be the more reliable data, and are suitable for analyses relating age and diameter.

Craters are large indentations in the ground, usually bowl-shaped, and caused by the impact of some sort of celestial object. Our data come from an SPSS format (`.sav`) and can be ingested into R using the `read_sav()` function from the `haven` package.

```
craters <- read_sav("data/craters.sav") |>
  janitor::clean_names()

craters
```

```
# A tibble: 168 x 10
   crater name           diameter     age n_s   latitude longitude buried drilled
    <dbl> <chr>             <dbl>   <dbl> <dbl>    <dbl>     <dbl> <dbl+> <dbl+l>
 1      1 "Haviland"        0.015 1   e-3 1 [N]     37.6     -99.2 1 [N]   1 [N]
 2      2 "Dalgaranga"      0.024 2.7 e-1 2 [S]    -27.6     117.  1 [N]   1 [N]
 3      3 "Sikhote Ali~     0.027 5.50e-5 1 [N]     46.1     135.  1 [N]   1 [N]
 4      4 "Campo Del C~     0.05  4   e-3 2 [S]    -27.6     -61.7 1 [N]   2 [Y]
 5      5 "Sobolev"         0.053 1   e-3 1 [N]     46.3     138.  1 [N]   2 [Y]
 6      6 "Veevers"         0.08  1   e+0 2 [S]    -23.0     125.  1 [N]   1 [N]
 7      7 "Ilumets\u00~     0.08  2   e-3 1 [N]     58.0      27.4 1 [N]   2 [Y]
 8      8 "Morasko"         0.1   1   e-2 1 [N]     52.5      16.9 1 [N]   1 [N]
 9      9 "Kaalij\u008~     0.11  4   e-3 1 [N]     58.4      22.7 1 [N]   1 [N]
10     10 "Wabar"           0.116 1.4 e-4 1 [N]     21.5      50.5 1 [N]   1 [N]
# i 158 more rows
# i 1 more variable: location <chr>
```

Columns in the data include:

Variable	Description
crater	numerical code (1 - 168)
name	name of crater

Variable	Description
diameter	diameter in km
age	age in millions of years (mA)
n_s	(N)orthern or (S)outhern hemisphere
latitude	Latitude, in degrees N or S of the equator
longitude	Longitude, in degrees W or E of the Greenwich meridian
buried	Yes or No
drilled	Yes or No
location	country (and sometimes state or province)

```
craters |> count(drilled)
```

```
# A tibble: 3 x 2
  drilled       n
  <dbl+lbl> <int>
1  1 [N]       67
2  2 [Y]       96
3 NA           5
```

```
str(craters)
```

```
tibble [168 x 10] (S3: tbl_df/tbl/data.frame)
 $ crater   : num [1:168] 1 2 3 4 5 6 7 8 9 10 ...
  ..- attr(*, "format.spss")= chr "F8.2"
 $ name     : chr [1:168] "Haviland" "Dalgaranga" "Sikhote Alin" "Campo Del Cielo" ...
  ..- attr(*, "format.spss")= chr "A42"
 $ diameter : num [1:168] 0.015 0.024 0.027 0.05 0.053 0.08 0.08 0.1 0.11 0.116 ...
  ..- attr(*, "format.spss")= chr "F8.2"
 $ age      : num [1:168] 1.0e-03 2.7e-01 5.5e-05 4.0e-03 1.0e-03 1.0 2.0e-03 1.0e-02 4.0e-03
  ..- attr(*, "format.spss")= chr "F8.2"
 $ n_s      : dbl+lbl [1:168] 1, 2, 1, 2, 1, 2, 1, 1, 1, 1, 2, 1, 2, 1, 1, 1, 2, 2, ...
   ..@ format.spss: chr "F8.0"
   ..@ labels     : Named num [1:2] 1 2
   .. ..- attr(*, "names")= chr [1:2] "N" "S"
 $ latitude : num [1:168] 37.6 -27.6 46.1 -27.6 46.3 ...
  ..- attr(*, "format.spss")= chr "F8.2"
 $ longitude: num [1:168] -99.2 117.3 134.7 -61.7 137.9 ...
  ..- attr(*, "format.spss")= chr "F8.2"
 $ buried   : dbl+lbl [1:168] 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
   ..@ format.spss: chr "F8.0"
```

```
  ..@ labels     : Named num [1:2] 1 2
  .. ..- attr(*, "names")= chr [1:2] "N" "Y"
 $ drilled  : dbl+lbl [1:168]  1,  1,  1,  2,  2,  1,  2,  1,  1,  1,  1,  2,  1,  1...
  ..@ format.spss: chr "F8.0"
  ..@ labels     : Named num [1:2] 1 2
  .. ..- attr(*, "names")= chr [1:2] "N" "Y"
 $ location : chr [1:168] "Kansas, U.S.A." "Western Australia, Australia" "Russia" "Argentina
  ..- attr(*, "format.spss")= chr "A36"
```

As we can see, several of the columns are *labeled*, and we likely want to eliminate those labels in our R code. To do so, we can use `zap_labels()` from the `haven` package.

```
craters <- craters |> zap_labels() |>
  mutate(buried = fct_recode(factor(buried), "No" = "1", "Yes" = "2"),
         drilled = fct_recode(factor(drilled), "No" = "1", "Yes" = "2"))

summary(craters)
```

```
     crater           name              diameter           age          
 Min.   :  1.00   Length:168         Min.   :  0.015   Min.   :   0.0001  
 1st Qu.: 42.75   Class :character   1st Qu.:  3.150   1st Qu.:  36.9000  
 Median : 84.50   Mode  :character   Median :  8.000   Median : 124.5000  
 Mean   : 84.50                      Mean   : 18.713   Mean   : 266.2618  
 3rd Qu.:126.25                      3rd Qu.: 18.250   3rd Qu.: 383.7500  
 Max.   :168.00                      Max.   :300.000   Max.   :2400.0000  
      n_s           latitude        longitude         buried    drilled  
 Min.   :1.000   Min.   :-34.72   Min.   :-156.633   No  :105   No  :67  
 1st Qu.:1.000   1st Qu.: 22.56   1st Qu.: -80.858   Yes : 61   Yes :96  
 Median :1.000   Median : 46.98   Median :  18.333   NA's:  2   NA's: 5  
 Mean   :1.208   Mean   : 33.94   Mean   :   2.892                       
 3rd Qu.:1.000   3rd Qu.: 56.68   3rd Qu.:  48.642                       
 Max.   :2.000   Max.   : 75.70   Max.   : 172.083                       
   location        
 Length:168         
 Class :character   
 Mode  :character   
```

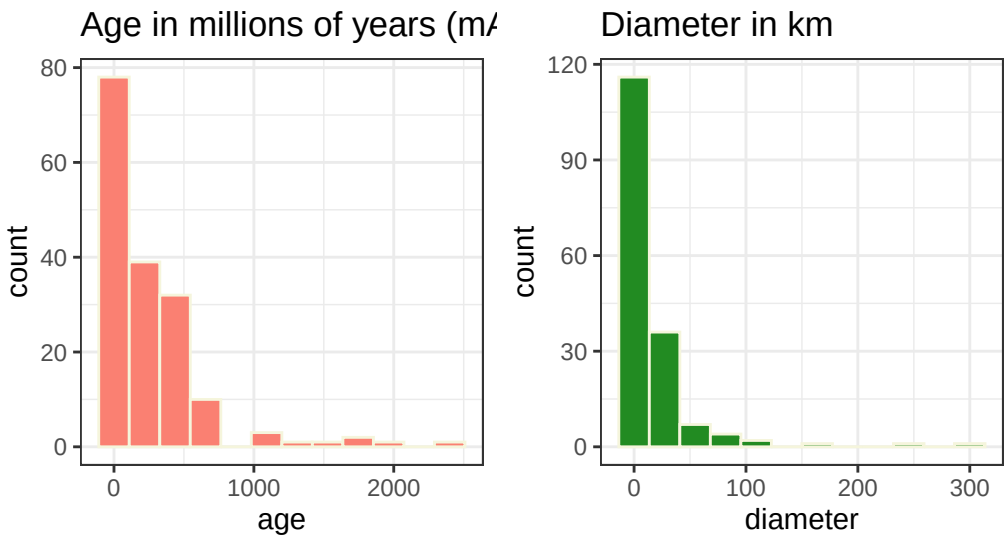
12.3 Association of age and diameter

```
res1 <- craters |> reframe(lovedist(age)) |> mutate(varname = "age")
res2 <- craters |> reframe(lovedist(diameter)) |> mutate(varname = "diameter")

rbind(res1, res2) |> relocate(varname) |> kable(digits = 2)
```

varname	n	miss	mean	sd	med	mad	min	q25	q75	max
age	168	0	266.26	379.03	124.5	183.69	0.00	36.90	383.75	2400
diameter	168	0	18.71	36.37	8.0	8.27	0.01	3.15	18.25	300

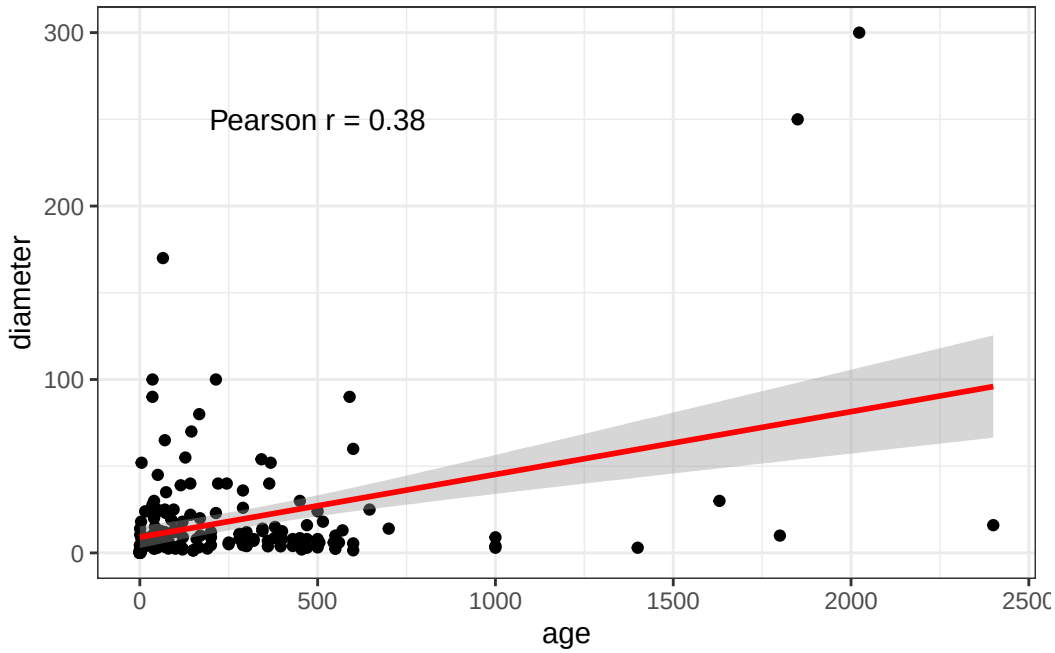
The age and diameter data each appear to be right skewed, with the sample mean substantially larger than the sample median, and the points clustered more closely together at the left of the histograms.

```
p1 <- ggplot(craters, aes(x = age)) +
  geom_histogram(bins = 12, fill = "salmon", col = "beige") +
  labs(title = "Age in millions of years (mA)")

p2 <- ggplot(craters, aes(x = diameter)) +
  geom_histogram(bins = 12, fill = "forestgreen", col = "beige") +
  labs(title = "Diameter in km")

(p1 + p2) +
  plot_annotation(title = "Age and Diameter of Craters",
                  subtitle = "Histograms, n = 168 craters")
```



When we look at the scatterplot of the association between age and diameter, most of the points are gathered in the bottom left of the plot.

```
ggplot(craters, aes(x = age, y = diameter)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, col = "red") +
  annotate("text", x = 500, y = 250,
           label = glue("Pearson r = ",
                        round(cor(craters$age, craters$diameter),2)))
```



In light of these results, it may be that a transformation (of either age, diameter or both) is in order.

12.4 Logarithmic Transformation of Each Variable

Consider taking the logarithm of both age and diameter.

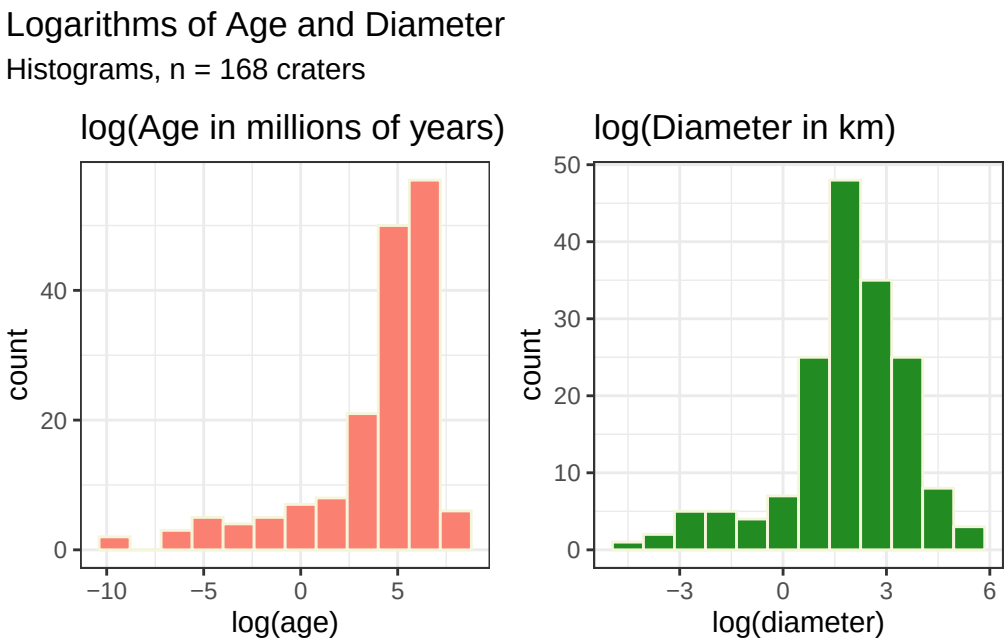
What does this imply? From Gelman, Hill, and Vehtari (2021) Section 3.4:

- The formula log y = a + b log x describes what is called *power-law growth* (if b $>$ 0) or decline (if b $<$ 0), with the formula $y = Ax^b$ where A = `exp(a)`.
- The parameter A is the value of y when x = 1
- The parameter b determines the rate of growth or decline.
- A one-unit difference in log x corresponds to an additive difference of b in log y.
- One example is a *power law*: Let y be the area of a square and x be its perimeter. Then $y = (x/4)^2$ and we can take the log of both sides to get log y = 2 (log x - log 4) = -2.8 + 2 log x.

12.4.1 Distributions after Transformation

We'd expect applying a log transformation to spread out the lower values, reducing the right skew in each variable. Is this what happens?

```
p3 <- ggplot(craters, aes(x = log(age))) +
  geom_histogram(bins = 12, fill = "salmon", col = "beige") +
  labs(title = "log(Age in millions of years)")

p4 <- ggplot(craters, aes(x = log(diameter))) +
  geom_histogram(bins = 12, fill = "forestgreen", col = "beige") +
  labs(title = "log(Diameter in km)")

(p3 + p4) +
  plot_annotation(title = "Logarithms of Age and Diameter",
                  subtitle = "Histograms, n = 168 craters")
```




```
craters <- craters |>
  mutate(logage = log(age), logdiameter = log(diameter))
```

```
res3 <- craters |> reframe(lovedist(logage)) |> mutate(varname = "logage")
res4 <- craters |> reframe(lovedist(logdiameter)) |> mutate(varname = "logdiameter")

rbind(res3, res4) |> relocate(varname) |> kable(digits = 2)
```

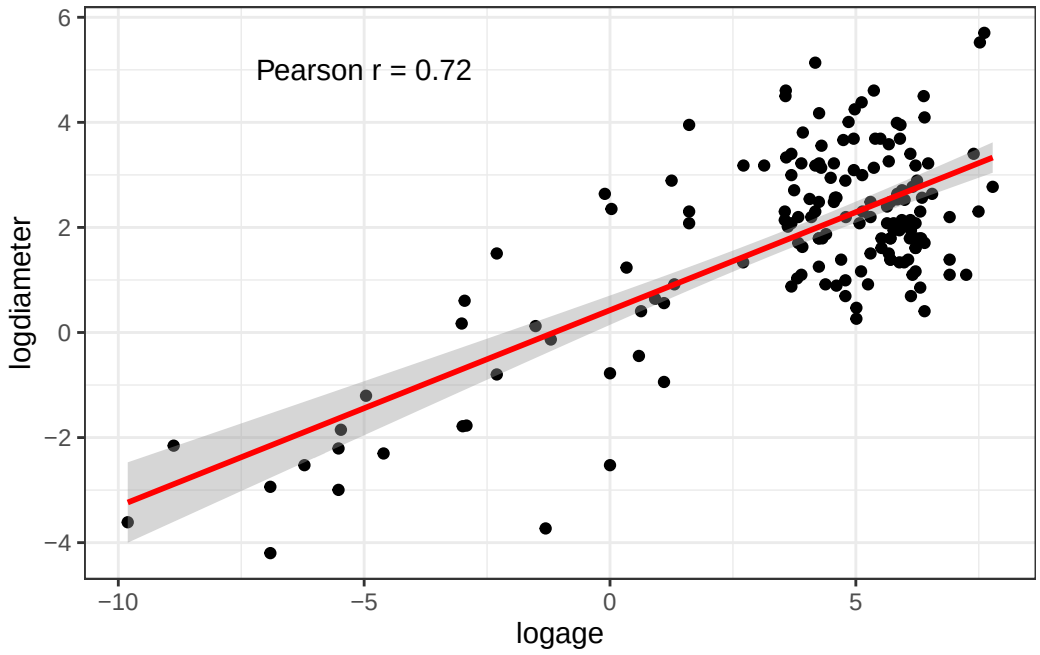
varname	n	miss	mean	sd	med	mad	min	q25	q75	max
logage	168	0	3.76	3.46	4.82	1.71	-9.81	3.61	5.95	7.78
logdiameter	168	0	1.83	1.79	2.08	1.36	-4.20	1.15	2.90	5.70

Note that the means and medians after the logarithmic transformation are substantially closer to each other, as is indicated by the reduction in skew we see in the histograms.

12.4.2 Association after Transformation

What happens when we look at the scatterplot of these transformed variables?

```
ggplot(craters, aes(x = logage, y = logdiameter)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, col = "red") +
  annotate("text", x = -5, y = 5,
           label = glue("Pearson r = ",
                        round(cor(craters$logage, craters$logdiameter),2)))
```



While we now have many points clustered in the top right of the plot, the overall fit of a linear model appears much better, as the points more closely cluster around the fitted line.

12.4.3 Comparing the Correlation Coefficients

We also see a larger Pearson correlation after this transformation, as we can also see from the `cor_test()` results below.

```
cor_test(craters, "age", "diameter")
```

```
Parameter1 | Parameter2 |    r |       95% CI | t(166) |         p
-------------------------------------------------------------------
age        |   diameter | 0.38 | [0.24, 0.50] |   5.25 | < .001***

Observations: 168
```

```
cor_test(craters, "logage", "logdiameter")
```

```
Parameter1 |  Parameter2 |    r |       95% CI | t(166) |         p
-----------------------------------------------------------------------
logage     | logdiameter | 0.72 | [0.64, 0.79] |  13.49 | < .001***

Observations: 168
```

We might also consider looking at the Spearman rank correlations:

```
cor_test(craters, "age", "diameter", method = "Spearman")
```

```
Parameter1 | Parameter2 |  rho |       95% CI |        S |         p
-------------------------------------------------------------------------
age        |   diameter | 0.33 | [0.19, 0.46] | 5.28e+05 | < .001***

Observations: 168
```

```
cor_test(craters, "logage", "logdiameter", method = "Spearman")
```

```
Parameter1 |  Parameter2 |  rho |       95% CI |        S |         p
-----------------------------------------------------------------------
logage     | logdiameter | 0.33 | [0.19, 0.46] | 5.28e+05 | < .001***

Observations: 168
```

Notice that the logarithmic transformation preserves the order of the observations, so that the ranks by age and diameter are identical to the ranks by log(age) and log(diameter), so the transformation has no impact on the Spearman rank correlation.

12.5 Untransformed Linear Model

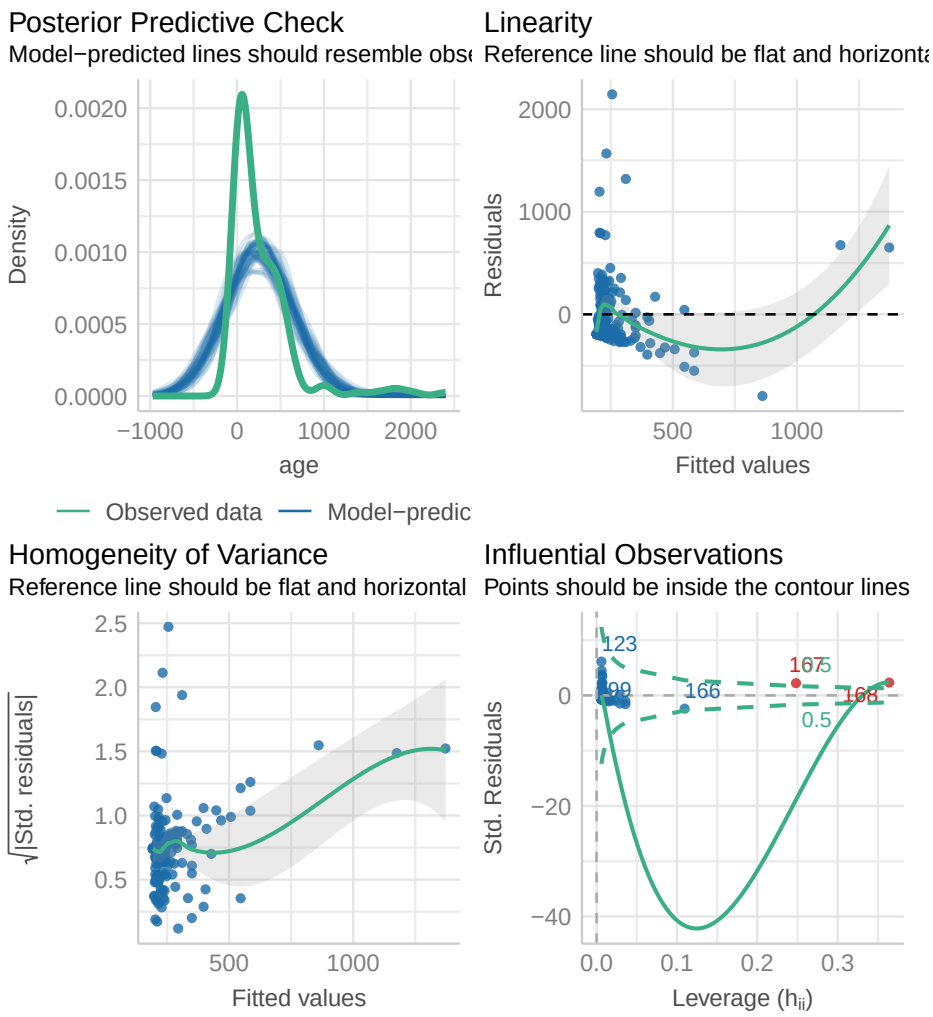
```
fit1 <- lm(age ~ diameter, data = craters)

model_parameters(fit1, ci = 0.95) |> kable(digits = 2)
```

Parameter	Coefficient	SE	CI	CI_low	CI_high	t	df_error	p
(Intercept)	192.71	30.57	0.95	132.35	253.06	6.30	166	0
diameter	3.93	0.75	0.95	2.45	5.41	5.25	166	0

```
model_performance(fit1)
```

```
# Indices of model performance

AIC      |     AICc |      BIC |    R2 | R2 (adj.) |    RMSE |   Sigma
-----------------------------------------------------------------------
2451.026 | 2451.172 | 2460.398 | 0.142 |     0.137 | 349.997 | 352.099
```

```
check_model(fit1)
```


Posterior Predictive Check
Model−predicted lines should resemble obs
Density
0.0020
0.0015
0.0010
0.0005
0.0000
−1000
0
1000
2000
age
Observed data
Model−predic
Linearity
Reference line should be flat and horizonta
Residuals
2000
1000
0
500
1000
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
2.5
2.0
1.5
1.0
0.5
500
1000
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
123
167
99
166
168
0.5
0
−20
−40
0.0
0.1
0.2
0.3
Leverage (hii)
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
5.0
2.5
0.0
−2.5
−3
−2
−1
0
1
2
3
Standard Normal Distribution Quantiles

12.6 Log-Log Regression Model

```
fit3 <- lm(logage ~ logdiameter, data = craters)

model_parameters(fit3, ci = 0.95) |> kable(digits = 3)
```

Parameter	Coefficient	SE	CI	CI_low	CI_high	t	df_error	p
(Intercept)	1.199	0.265	0.95	0.675	1.723	4.52	166	0
logdiameter	1.401	0.104	0.95	1.196	1.606	13.49	166	0

```
model_performance(fit3)
```

```
# Indices of model performance

AIC     |    AICc |     BIC |    R2 | R2 (adj.) |  RMSE | Sigma
-----------------------------------------------------------------
774.932 | 775.078 | 784.304 | 0.523 |     0.520 | 2.386 | 2.400
```

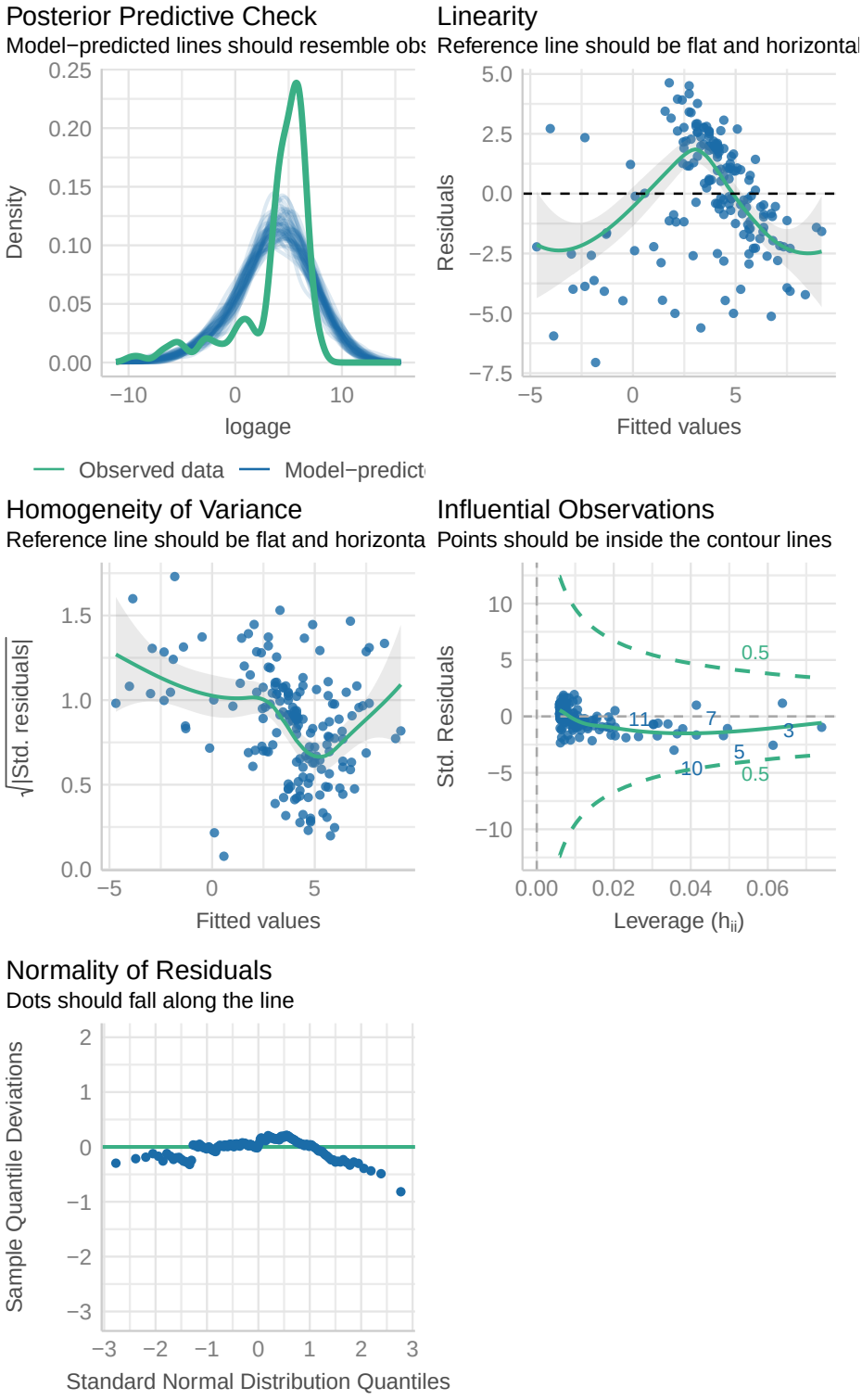
Our equation is $\log(\text{age}) = 1.2 + 1.4 \log(\text{diameter})$.

We can exponentiate both sides to see the relation between age and diameter on the untransformed scales.

$$\exp(\log y) = \exp(1.2 + 1.4 \log(\mathit{diameter})), \text{so } y = \mathit{exp}(1.2) \times x^{1.4}, \text{so } y = 3.32x^{1.4}$$

When increasing diameter by a factor of 2 (i.e., doubling the diameter) the estimated age is multiplied by $2^{1.4} = 2.64$. Halving the diameter causes the estimated age to be multiplied by $(1/2)^{1.4} = 0.38$, and so forth.

```
check_model(fit3)
```


Posterior Predictive Check
Model−predicted lines should resemble obs
Density
logage
Observed data
Model−predict
Linearity
Reference line should be flat and horizontal
Residuals
Fitted values
Homogeneity of Variance
Reference line should be flat and horizonta
√|Std. residuals|
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
Leverage (hii)
0.5
0.5
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
Standard Normal Distribution Quantiles

The fit is better here, although still not really as good as we might hope for. Would a Bayesian fit be helpful?

12.7 Bayesian Log-Log Regression

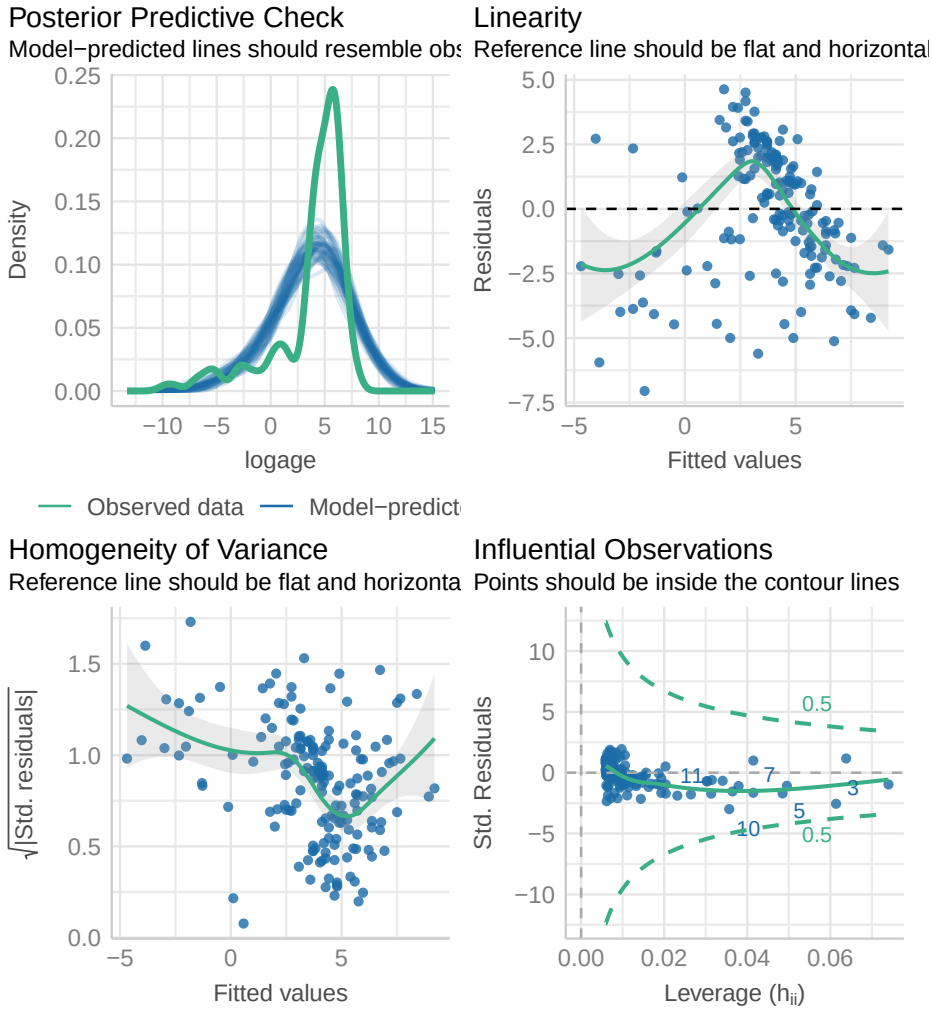
```
set.seed(431)
fit4 <- stan_glm(logage ~ logdiameter,
                 data = craters, refresh = 0)

model_parameters(fit4, ci = 0.95) |> kable(digits = 2)
```

Parameter	Median	CI	CI_low	CI_high	pd	Rhat	ESS	Prior_Distribution	Prior_Location	Prior_Scale
(Intercept)	1.21	0.95	0.67	1.72	1	1	3756.99	normal	3.76	8.66
logdiameter	1.40	0.95	1.20	1.61	1	1	3860.42	normal	0.00	4.84

```
model_performance(fit4)
```

```
# Indices of model performance

ELPD     | ELPD_SE |   LOOIC | LOOIC_SE |    WAIC |    R2 | R2 (adj.) |  RMSE | Sigma
-------------------------------------------------------------------------------------
-387.860 |   8.599 | 775.720 |   17.198 | 775.699 | 0.519 |     0.508 | 2.386 | 2.411
```

```
check_model(fit4)
```


Posterior Predictive Check
Model−predicted lines should resemble obs
Density
logage
Observed data
Model−predict
Linearity
Reference line should be flat and horizontal
Residuals
Fitted values
Homogeneity of Variance
Reference line should be flat and horizonta
√|Std. residuals|
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
Leverage (hii)
0.5
0.5
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
Standard Normal Distribution Quantiles

We still have some meaningful non-linearity, I think, but we'll leave it as is for now.

12.8 For More Information

1. Gelman, Hill, and Vehtari (2021) is a great, though not free, reference for many of the ideas discussed in this entire book, and especially this chapter.
2. More on `check_model()` in Visual check of model assumptions. Posterior predictive checks is another good resource.
3. Chapter 7 of Çetinkaya-Rundel and Hardin (2024) is about linear regression with a single predictor, and has sections on residuals and checking for outliers, for instance.
4. The Qualtrics support page has a walkthrough for residual plots in the case of simple linear regression that should provide some additional background.

Part III

Comparing Categories

13 Proportions and Rates

> ! This is a DRAFT version of this Chapter.
>
> This is a sketchy draft. I'll remove this notice when I post a version of this Chapter that is essentially finished.

13.1 R setup for this chapter

> i Note
>
> Appendix A lists all R packages used in this book, and also provides R session information.

```
library(Epi)
library(knitr)
library(medicaldata)
library(naniar)

library(easystats)
library(tidyverse)

source("data/Love-431.R")
theme_set(theme_bw())
```

13.2 Data: strep_tb data from the `medicaldata` R package

> **Note**
>
> Appendix C provides further guidance on pulling data from other systems into R, while Appendix D gives more information (including download links) for all data sets used in this book. Appendix B describes the 431-Love.R script, and demonstrates its use.

See pages 51-52 of R&OS for standard errors and confidence intervals for proportions, and for what to do when y = 0 or y = n

My source for these data is Higgins (2023).

strep_tb data from the `medicaldata` R package - we'll look at study arm (streptomycin or control) and the dichotomous outcome of improved (true, false) - will need to work with a logical variable, and we'll also keep the patient ID.

See https://higgi13425.github.io/medicaldata/ for more details.

```
strep <- medicaldata::strep_tb |>
  mutate(
    imp_f = factor(improved),
    imp_f = fct_recode(imp_f,
      "Improved" = "TRUE",
      "Worsened" = "FALSE"
    ),
    imp_f = fct_relevel(imp_f, "Improved")
  )
strep
```

```
# A tibble: 107 x 14
   patient_id arm     dose_strep_g dose_PAS_g gender baseline_condition
   <chr>      <fct>          <dbl>      <dbl> <fct>  <fct>
 1 0001       Control            0          0 M      1_Good
 2 0002       Control            0          0 F      1_Good
 3 0003       Control            0          0 F      1_Good
 4 0004       Control            0          0 M      1_Good
 5 0005       Control            0          0 F      1_Good
 6 0006       Control            0          0 M      1_Good
 7 0007       Control            0          0 F      1_Good
 8 0008       Control            0          0 M      1_Good
 9 0009       Control            0          0 F      2_Fair
10 0010       Control            0          0 M      2_Fair
```

```
# i 97 more rows
# i 8 more variables: baseline_temp <fct>, baseline_esr <fct>,
#   baseline_cavitation <fct>, strep_resistance <fct>, radiologic_6m <fct>,
#   rad_num <dbl>, improved <lgl>, imp_f <fct>
```

```
table(strep$arm, strep$imp_f)
```

```
              Improved Worsened
  Streptomycin       38       17
  Control            17       35
```

13.3 Estimating a Proportion

Within those who received Streptomycin, 38 improved and 17 did not out of 55 subjects. Can we estimate a confidence interval for the population proportion of all subjects?

13.3.1 Using a Bayesian augmentation

```
binom.test(x = 38 + 2, n = 55 + 4, conf.level = 0.95)
```

```
    Exact binomial test

data:  38 + 2 and 55 + 4
number of successes = 40, number of trials = 59, p-value = 0.008641
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.5436200 0.7937535
sample estimates:
probability of success
             0.6779661
```

13.3.2 SAIFS: single augmentation with an imaginary failure or success

The `saifs_ci()` function

```
`saifs_ci` <-
  function(x, n, conf.level=0.95, dig=3)
  {
    p.sample <- round(x/n, digits=dig)

    p1 <- x / (n+1)
    p2 <- (x+1) / (n+1)

    var1 <- (p1*(1-p1))/n
    se1 <- sqrt(var1)
    var2 <- (p2*(1-p2))/n
    se2 <- sqrt(var2)

    lowq = (1 - conf.level)/2
    tcut <- qt(lowq, df=n-1, lower.tail=FALSE)

    lower.bound <- round(p1 - tcut*se1, digits=dig)
    upper.bound <- round(p2 + tcut*se2, digits=dig)
    tibble(
      sample_x = x,
      sample_n = n,
      sample_p = p.sample,
      lower = lower.bound,
      upper = upper.bound,
      conf_level = conf.level
    )
  }
```

Using the `saifs_ci()` function from `Love-431.R`

```
saifs_ci(x = 38, n = 55, conf.level = 0.95, dig = 3)
```

```
# A tibble: 1 x 6
  sample_x sample_n sample_p lower upper conf_level
     <dbl>    <dbl>    <dbl> <dbl> <dbl>      <dbl>
1       38       55    0.691 0.552 0.821       0.95
```

13.4 Assessing the 2 x 2 table

This table is in **standard epidemiological format**, which means that:

- The rows of the table describe the "treatment" (which we'll take here to be arm).
  - The more interesting (sometimes also the more common) "treatment" is placed in the top row. That's Streptomycin here.
- The columns of the table describe the "outcome" (which we'll take here to be whether the subject improved or not.)
  - Typically, the more common or more interesting "outcome" is placed to the left. Here, we'll use "improved" on the left.

```
twoby2(table(strep$arm, strep$imp_f))
```

```
2 by 2 table analysis:
------------------------------------------------------
Outcome   : Improved
Comparing : Streptomycin vs. Control

             Improved Worsened    P(Improved) 95% conf. interval
Streptomycin       38       17         0.6909    0.5579   0.7984
Control            17       35         0.3269    0.2139   0.4644

                                  95% conf. interval
            Relative Risk: 2.1134    1.3773   3.2429
         Sample Odds Ratio: 4.6021    2.0389  10.3877
Conditional MLE Odds Ratio: 4.5304    1.8962  11.2779
    Probability difference: 0.3640    0.1754   0.5182

            Exact P-value: 0.0002
        Asymptotic P-value: 0.0002
------------------------------------------------------
```

13.5 Ebola Virus Study

The World Health Organization's Ebola Response Team published an article[1] in the October 16, 2014 issue of the New England Journal of Medicine, which contained some data I will use in this example, focusing on their Table 2.

Suppose we want to compare the proportion of deaths among cases that had a definitive outcome who were hospitalized to the proportion of deaths among cases that had a definitive outcome who were not hospitalized.

[1]WHO Ebola Response Team (2014) Ebola virus disease in West Africa: The first 9 months of the epidemic and forward projections. New Engl J Med 371: 1481-1495 doi: 10.1056/NEJMoa1411100

The article suggests that of the 1,737 cases with a definitive outcome, there were 1,153 hospitalized cases. Across those 1,153 hospitalized cases, 741 people (64.3%) died, which means that across the remaining 584 non-hospitalized cases, 488 people (83.6%) died.

Here is the initial contingency table, using only the numbers from the previous paragraph.

Initial Ebola Table	Deceased	Alive	Total
Hospitalized	741	–	1153
Not Hospitalized	488	–	584
Total			1737

Now, we can use arithmetic to complete the table, since the rows and the columns are each mutually exclusive and collectively exhaustive.

Ebola 2x2 Table	Deceased	Alive	Total
Hospitalized	741	412	1153
Not Hospitalized	488	96	584
Total	1229	508	1737

We want to compare the fatality risk (probability of being in the deceased column) for the population of people in the hospitalized row to the population of people in the not hospitalized row.

See sections 25.4 and 26.11 in the 2023 course notes.

```
twobytwo(741, 412, 488, 96,
         "Hosp", "Not Hosp", "Dead", "Alive",
         conf.level = 0.95)
```

```
2 by 2 table analysis:
------------------------------------------------------
Outcome   : Dead
Comparing : Hosp vs. Not Hosp

         Dead Alive    P(Dead) 95% conf. interval
Hosp      741   412     0.6427    0.6146   0.6698
Not Hosp  488    96     0.8356    0.8033   0.8635

                                  95% conf. interval
            Relative Risk:  0.7691    0.7271   0.8135
         Sample Odds Ratio:  0.3538    0.2756   0.4542
```

```
Conditional MLE Odds Ratio:  0.3540    0.2726   0.4566
    Probability difference: -0.1929   -0.2325  -0.1508

             Exact P-value: 0.0000
        Asymptotic P-value: 0.0000
------------------------------------------------------
```

```
twobytwo(412, 741, 96, 488,
         "Hosp", "Not Hosp", "Alive", "Dead",
         conf.level = 0.95)
```

```
2 by 2 table analysis:
------------------------------------------------------
Outcome   : Alive
Comparing : Hosp vs. Not Hosp

         Alive Dead    P(Alive) 95% conf. interval
Hosp       412  741      0.3573    0.3302   0.3854
Not Hosp    96  488      0.1644    0.1365   0.1967

                                  95% conf. interval
             Relative Risk: 2.1737    1.7823   2.6512
         Sample Odds Ratio: 2.8264    2.2016   3.6284
Conditional MLE Odds Ratio: 2.8248    2.1900   3.6678
    Probability difference: 0.1929    0.1508   0.2325

             Exact P-value: 0.0000
        Asymptotic P-value: 0.0000
------------------------------------------------------
```

13.6 For More Information

- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2797398/ talks about 2x2 tables
- OpenStats https://www.openintro.org/book/os/ Section 5 - foundations for inference about a proportion
- OpenStats https://www.openintro.org/book/os/ Section 6 - inference for categorical data (except but we don't do 6.3 in 431)

14 Cross-Tabulations

! This is a DRAFT version of this Chapter.

This is a sketchy draft. I'll remove this notice when I post a version of this Chapter that is essentially finished.

14.1 R setup for this chapter

i Note

Appendix A lists all R packages used in this book, and also provides R session information.

```
library(janitor)
library(knitr)
library(naniar)

library(easystats)
library(tidyverse)

theme_set(theme_bw())
```

14.2 Tattoo Example

i Note

Appendix C provides further guidance on pulling data from other systems into R, while Appendix D gives more information (including download links) for all data sets used in this book.

```
tats <- read_tsv("data/tattoos.txt", show_col_types = FALSE) |>
  mutate(across(where(is.character), as_factor)) |>
  janitor::clean_names()

glimpse(tats)
```

```
Rows: 626
Columns: 2
$ location        <fct> Commercial Parlor, Commercial Parlor, Commercial Parlo~
$ has_hepatitis_c <fct> Yes, Yes, Yes, Yes, Yes, Yes, Yes, Yes, Yes, Yes, Yes,~
```

The `tatoo.txt` data we ingest here into R comes from the Data and Story Library. The original source of the data is the University of Texas Southwestern Medical Center, and we observe 625 individuals categorized according to their tattoo status and whether or not they have a diagnosis of Hepatitis C. Specifically, the variables include:

- *location* in one of three groups:
  - (tattoo obtained in a) Commercial Parlor,
  - (tattoo obtained) Elsewhere, or
  - No Tattoo
- *has_hepatitis_c* status in two groups: Yes, No

```
tats |> count(location, has_hepatitis_c)
```

```
# A tibble: 6 x 3
  location          has_hepatitis_c     n
  <fct>             <fct>           <int>
1 Commercial Parlor Yes                17
2 Commercial Parlor No                 35
3 Elsewhere         Yes                 8
4 Elsewhere         No                 53
5 No Tattoo         Yes                22
6 No Tattoo         No                491
```

```
tats |> tabyl(location, has_hepatitis_c) |>
  adorn_title() |>
  kable()
```

	has_hepatitis_c	
location	Yes	No
Commercial Parlor	17	35
Elsewhere	8	53
No Tattoo	22	491

```
tats |>
  tabyl(location, has_hepatitis_c) |>
  adorn_percentages(denominator = "row") |>
  adorn_pct_formatting() |>
  adorn_ns(position = "front")
```

```
          location        Yes          No
 Commercial Parlor 17 (32.7%)  35 (67.3%)
         Elsewhere  8 (13.1%)  53 (86.9%)
         No Tattoo 22  (4.3%) 491 (95.7%)
```

```
data_tabulate(tats$location, tats$has_hepatitis_c,
  proportions = "col", include_na = FALSE
)
```

```
tats$location     |        Yes |          No |   <NA> | Total
------------------+------------+-------------+--------+------
Commercial Parlor | 17 (36.2%) |  35  (6.0%) | 0 (0%) |    52
Elsewhere         |  8 (17.0%) |  53  (9.2%) | 0 (0%) |    61
No Tattoo         | 22 (46.8%) | 491 (84.8%) | 0 (0%) |   513
<NA>              |  0  (0.0%) |   0  (0.0%) | 0 (0%) |     0
------------------+------------+-------------+--------+------
Total             |         47 |         579 |      0 |   626
```

14.3 Chi-Square Test

```
chisq.test(table(tats$location, tats$has_hepatitis_c))
```

```
Warning in stats::chisq.test(x, y, ...): Chi-squared approximation may be
incorrect
```

```
    Pearson's Chi-squared test

data:  table(tats$location, tats$has_hepatitis_c)
X-squared = 57.912, df = 2, p-value = 2.658e-13
```

A chi-square test of independence is a descriptive summary, like a correlation coefficient, so there's no outcome being modeled, really. This is reflected in the `xtabs()` function's approach.

```
tabx <- xtabs(~ location + has_hepatitis_c, data = tats)

tabx
```

```
                  has_hepatitis_c
location           Yes  No
  Commercial Parlor  17  35
  Elsewhere           8  53
  No Tattoo          22 491
```

```
summary(tabx)
```

```
Call: xtabs(formula = ~location + has_hepatitis_c, data = tats)
Number of cases in table: 626
Number of factors: 2
Test for independence of all factors:
    Chisq = 57.91, df = 2, p-value = 2.658e-13
    Chi-squared approximation may be incorrect
```

```
tat_tab <- tats |> tabyl(location, has_hepatitis_c)

tab_res1 <- chisq.test(tat_tab, tabyl_results = TRUE)
```

```
Warning in stats::chisq.test(., ...): Chi-squared approximation may be
incorrect
```

```
tab_res1
```

```
    Pearson's Chi-squared test

data:  tat_tab
X-squared = 57.912, df = 2, p-value = 2.658e-13
```

```
tab_res1$observed
```

```
          location Yes  No
 Commercial Parlor  17  35
         Elsewhere   8  53
         No Tattoo  22 491
```

```
tab_res1$expected
```

```
          location       Yes        No
 Commercial Parlor  3.904153  48.09585
         Elsewhere  4.579872  56.42013
         No Tattoo 38.515974 474.48403
```

```
tab_res1$residuals
```

```
          location       Yes         No
 Commercial Parlor  6.627811 -1.8883383
         Elsewhere  1.598143 -0.4553290
         No Tattoo -2.661238  0.7582168
```

```
tab_res1$stdres
```

```
          location       Yes        No
 Commercial Parlor  7.196963 -7.196963
         Elsewhere  1.749148 -1.749148
         No Tattoo -6.512978  6.512978
```

14.4 Personal Appearance Example

These data are also adapted from an example in the Data and Story Library. The data are an excerpt from the results of a GfK Roper Reports® Worldwide survey. In addition to grouping the subjects into five age groups, each was also asked how important their personal appearance is to them, on a seven-point scale.

The data are a contingency table of responses to this question by age decade for 5,844 consumers.

Personal Appearance	20-29	30-39	40-49	50-59	60plus	Total
1 - Not at all important	37	53	56	36	52	**234**
2	43	53	58	37	45	**236**
3	83	88	93	54	45	**363**
4 - Average importance	376	403	423	224	210	**1636**
5	312	317	270	150	106	**1155**
6	326	307	254	123	86	**1096**
7 - Extremely important	337	300	252	142	93	**1124**
Total	**1514**	**1521**	**1406**	**766**	**637**	**5844**

Rather than generating an R tibble with 5844 rows, here I'll just recreate the cross-tabulation in R, then analyze it.

```
persapp <-
  as.table(rbind (
    c(37, 53, 56, 36, 52),
    c(43, 53, 58, 37, 45),
    c(83, 88, 93, 54, 45),
    c(376, 403, 423, 224, 210),
    c(312, 317, 270, 150, 106),
    c(326, 307, 254, 123, 86),
    c(337, 300, 252, 142, 93)))

dimnames(persapp) <-
  list( appear= c("1", "2", "3", "4", "5", "6", "7"),
        age = c("20-29", "30-39", "40-49", "50-59", "60plus"))

persapp
```

```
      age
appear 20-29 30-39 40-49 50-59 60plus
```

```
     1    37    53    56    36    52
     2    43    53    58    37    45
     3    83    88    93    54    45
     4   376   403   423   224   210
     5   312   317   270   150   106
     6   326   307   254   123    86
     7   337   300   252   142    93
```

Now, let's look at the results from a χ^2 test of independence of the rows and columns from this contingency table.

```
out2 <- chisq.test(persapp)

out2
```

```
    Pearson's Chi-squared test

data:  persapp
X-squared = 120.83, df = 24, p-value = 6.914e-15
```

```
out2$observed
```

```
      age
appear 20-29 30-39 40-49 50-59 60plus
     1    37    53    56    36    52
     2    43    53    58    37    45
     3    83    88    93    54    45
     4   376   403   423   224   210
     5   312   317   270   150   106
     6   326   307   254   123    86
     7   337   300   252   142    93
```

```
out2$expected
```

```
      age
appear     20-29     30-39     40-49     50-59    60plus
     1  60.62218  60.90246  56.29774  30.67146  25.50616
     2  61.14031  61.42300  56.77892  30.93361  25.72416
     3  94.04209  94.47690  87.33368  47.58008  39.56725
```

```
     4 423.83710 425.79671 393.60301 214.43806 178.32512
     5 299.22485 300.60832 277.87988 151.39117 125.89579
     6 283.93977 285.25257 263.68515 143.65777 119.46475
     7 291.19370 292.54004 270.42163 147.32786 122.51677
```

```
out2$residuals
```

```
      age
appear      20-29      30-39      40-49      50-59     60plus
     1 -3.0339202 -1.0126167 -0.0396820  0.9621465  5.2459285
     2 -2.3199626 -1.0747345  0.1620508  1.0907250  3.8005169
     3 -1.1386502 -0.6663530  0.6063321  0.9307154  0.8636781
     4 -2.3236213 -1.1047681  1.4817456  0.6529731  2.3719674
     5  0.7385286  0.9454163 -0.4727057 -0.1130655 -1.7731913
     6  2.4960803  1.2876363 -0.5964354 -1.7235300 -3.0617357
     7  2.6843194  0.4361579 -1.1202303 -0.4389451 -2.6666809
```

```
out2$stdres
```

```
      age
appear       20-29       30-39       40-49       50-59      60plus
     1 -3.59740205 -1.20165904 -0.04647599  1.05347351  5.67227598
     2 -2.75133354 -1.27560078  0.18982947  1.19446962  4.11012547
     3 -1.36592442 -0.80000375  0.71845112  1.03098141  0.94479685
     4 -3.18122022 -1.51373842  2.00379056  0.82550794  2.96133317
     5  0.95784108  1.22715817 -0.60557414 -0.13541096 -2.09716160
     6  3.21713632  1.66094580 -0.75931957 -2.05129138 -3.59856010
     7  3.46999984  0.56427443 -1.43038477 -0.52396591 -3.14352177
```

14.5 For More Information

15 Statistical Significance

> ❗ I'm not posting a draft here until it's better than it is now.
>
> I'll remove this notice when I post a version of this Chapter that is essentially finished.

15.1 For More Information

Part IV

Linear Regression

16 Covariate Adjustment

> ! This is a DRAFT version of this Chapter.
>
> This is a sketchy draft. I'll remove this notice when I post a version of this Chapter that is essentially finished.

16.1 R setup for this chapter

> i Note
>
> Appendix A lists all R packages used in this book, and also provides R session information. Appendix B describes the 431-Love.R script, and demonstrates its use.

```
library(haven)
library(knitr)
library(naniar)
library(patchwork)
library(rstanarm)

library(easystats)
library(tidyverse)

source("data/Love-431.R")
theme_set(theme_bw())
```

16.2 Data ingest from Stata: Supraclavicular Nerve Block

> **Note**
>
> Appendix C provides further guidance on pulling data from other systems into R, while Appendix D gives more information (including download links) for all data sets used in this book.

This is adapted from the Supraclavicular[1] Nerve Block example at the Cleveland Clinic Statistical Dataset Repository. The source is Roberman et al. (2011). A version of the data is also available as part of the medicaldata package in R (see Higgins (2023).)

We have provided the data in a Stata data file (`.dta`) which we can ingest into R using the `read_dta()` function in the `haven` package, as follows:

```
supra <- read_dta("data/supraclav.dta")

head(supra)
```

```
# A tibble: 6 x 6
  subject onset_sensory group opioid_total   bmi   age
    <dbl>         <dbl> <dbl>        <dbl> <dbl> <dbl>
1       1             0     1           30  41.2    52
2       2             7     2          150  25.2    54
3       3            24     2            0  34.1    46
4       4             4     1           15  41.6    54
5       5            30     1           90  27.2    41
6       6             4     2           15  22.0    21
```

The data contain information on 103 patients, ages 18-70 years who ...

> were scheduled to undergo an upper extremity procedure suitable for supraclavicular anesthesia at the Cleveland Clinic. Patients were randomly assigned to either (1) combined group-ropivacaine and mepivacaine mixture; or (2) sequential group-mepivacaine followed by ropivacaine. A number of demographic and post-op pain medication variables (fentanyl, alfentanil, midazolam) were collected. The primary outcome is time to 4-nerve sensory block onset.
>
> This study investigates whether sequential supraclavicular injection of 1.5% mepivacaine followed 90 seconds later by 0.5% ropivacaine provides a quicker onset and a longer duration of analgesia than an equidose combination of the 2 local anesthetics.

[1] Supraclavicular refers to the area above the clavicle, or collarbone, on either side of the body.

> The primary outcome was time to 4-nerve sensory block onset, which was defined as time from the completion of anesthetic injection until development of sensory block to sharp pain in each of the 4 major nerve distributions: median, ulnar, radial, and musculocutaneous.

Our interest today is in determining whether the mixture or sequential group has a shorter time to 4-nerve sensory block[2], measured in minutes. We will first assess this without adjustment for any other variable, then consider whether adjusting for the total opioid consumption (measured in mg) might change our thinking about the association of group with block time.

The variables of interest, then, are:

Variable	Description
`subject`	numeric subject ID code
`onset_sensory`	time in minutes to 4-nerve sensory block[3]
`group`	anesthetic group: either 1 = mixture, or 2 = sequential
`opioid_total`	total opioid consumption (in mg)

We have also collected age (in years) for all subjects and BMI (in kg/m^2) for all but three of the subjects.

```
miss_var_summary(supra)
```

```
# A tibble: 6 x 3
  variable      n_miss pct_miss
  <chr>          <int>    <num>
1 bmi                3     2.91
2 subject            0     0
3 onset_sensory      0     0
4 group              0     0
5 opioid_total       0     0
6 age                0     0
```

Let's express the `group` in terms of the names of the two anesthetic conditions the study applies.

[2] Onset_first_sensory was defined as time from the completion of anesthetic injection until development of first sensory block to sharp pain.

[3] Two subjects (subjects 16 and 84) failed to achieve complete sensory and motor block, and these were censored at 50 minutes. We'll ignore this here.

```
supra <- supra |>
  mutate(group =
           fct_recode(factor(group),
                      "Mixture" = "1", "Sequential" = "2"),
         subject = as.character(subject))

glimpse(supra)
```

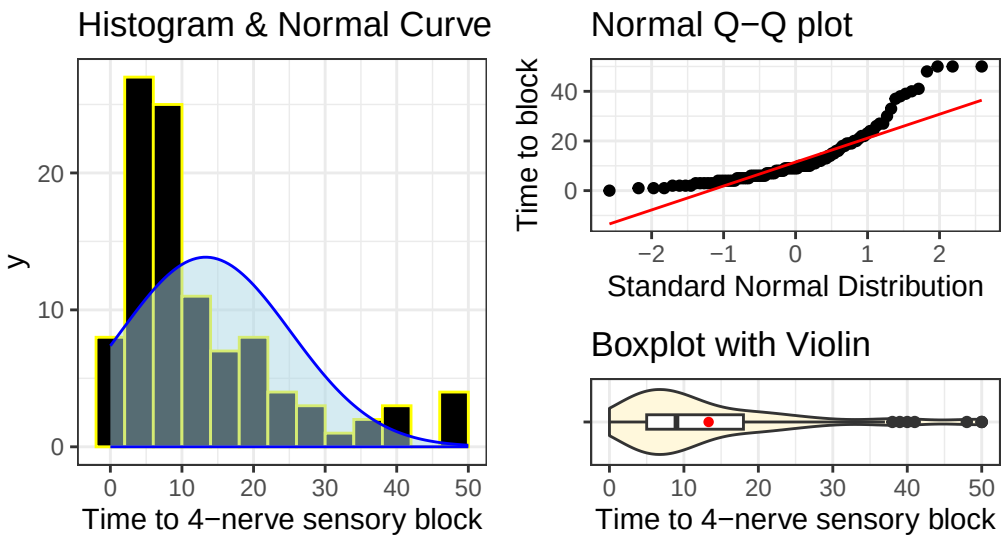
```
Rows: 103
Columns: 6
$ subject       <chr> "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11",~
$ onset_sensory <dbl> 0, 7, 24, 4, 30, 4, 12, 13, 27, 4, 3, 21, 9, 9, 5, 50, 7~
$ group         <fct> Mixture, Sequential, Sequential, Mixture, Mixture, Seque~
$ opioid_total  <dbl> 30.00, 150.00, 0.00, 15.00, 90.00, 15.00, 15.00, 60.00, ~
$ bmi           <dbl> 41.15, 25.22, 34.14, 41.57, 27.17, 22.05, 26.32, 24.69, ~
$ age           <dbl> 52, 54, 46, 54, 41, 21, 68, 61, 44, 28, 36, 60, 34, 64, ~
```

16.3 Exploratory Data Analysis

We'll start by looking at the distribution of our outcome, `onset_sensory`.

```
bw = 4 # specify width of bins in histogram

p1 <- ggplot(supra, aes(onset_sensory)) +
  geom_histogram(binwidth = bw,
                 fill = "black", col = "yellow") +
  stat_function(fun = function(x)
    dnorm(x, mean = mean(supra$onset_sensory,
                         na.rm = TRUE),
          sd = sd(supra$onset_sensory,
                  na.rm = TRUE)) *
      length(supra$onset_sensory) * bw,
    geom = "area", alpha = 0.5,
    fill = "lightblue", col = "blue"
    ) +
  labs(
    x = "Time to 4-nerve sensory block",
    title = "Histogram & Normal Curve"
  )
```

```r
p2 <- ggplot(supra, aes(sample = onset_sensory)) +
  geom_qq() +
  geom_qq_line(col = "red") +
  labs(
    y = "Time to block",
    x = "Standard Normal Distribution",
    title = "Normal Q-Q plot"
  )

p3 <- ggplot(supra, aes(x = onset_sensory, y = "")) +
  geom_violin(fill = "cornsilk") +
  geom_boxplot(width = 0.2) +
  stat_summary(
    fun = mean, geom = "point",
    shape = 16, col = "red"
  ) +
  labs(
    y = "", x = "Time to 4-nerve sensory block",
    title = "Boxplot with Violin"
  )

p1 + (p2 / p3 + plot_layout(heights = c(2, 1))) +
  plot_annotation(
    title = "Supraclavicular Nerve Block",
    subtitle = "Time to 4-nerve sensory block"
  )
```




```
supra |> reframe(lovedist(onset_sensory)) |>
  kable(digits = 2)
```

n	miss	mean	sd	med	mad	min	q25	q75	max
103	0	13.32	11.87	9	7.41	0	5	18	50

16.4 Time to Block by Group

16.4.1 Least Squares Linear Model

```
fit1 <- lm(onset_sensory ~ group, data = supra)

model_parameters(fit1, ci = 0.95)
```

```
Parameter          | Coefficient |   SE |         95% CI | t(101) |      p
-------------------------------------------------------------------------
(Intercept)        |       11.42 | 1.63 | [ 8.19, 14.66] |   7.00 | < .001
group [Sequential] |        3.83 | 2.32 | [-0.77,  8.43] |   1.65 | 0.102
```

```
Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
  using a Wald t-distribution approximation.
```

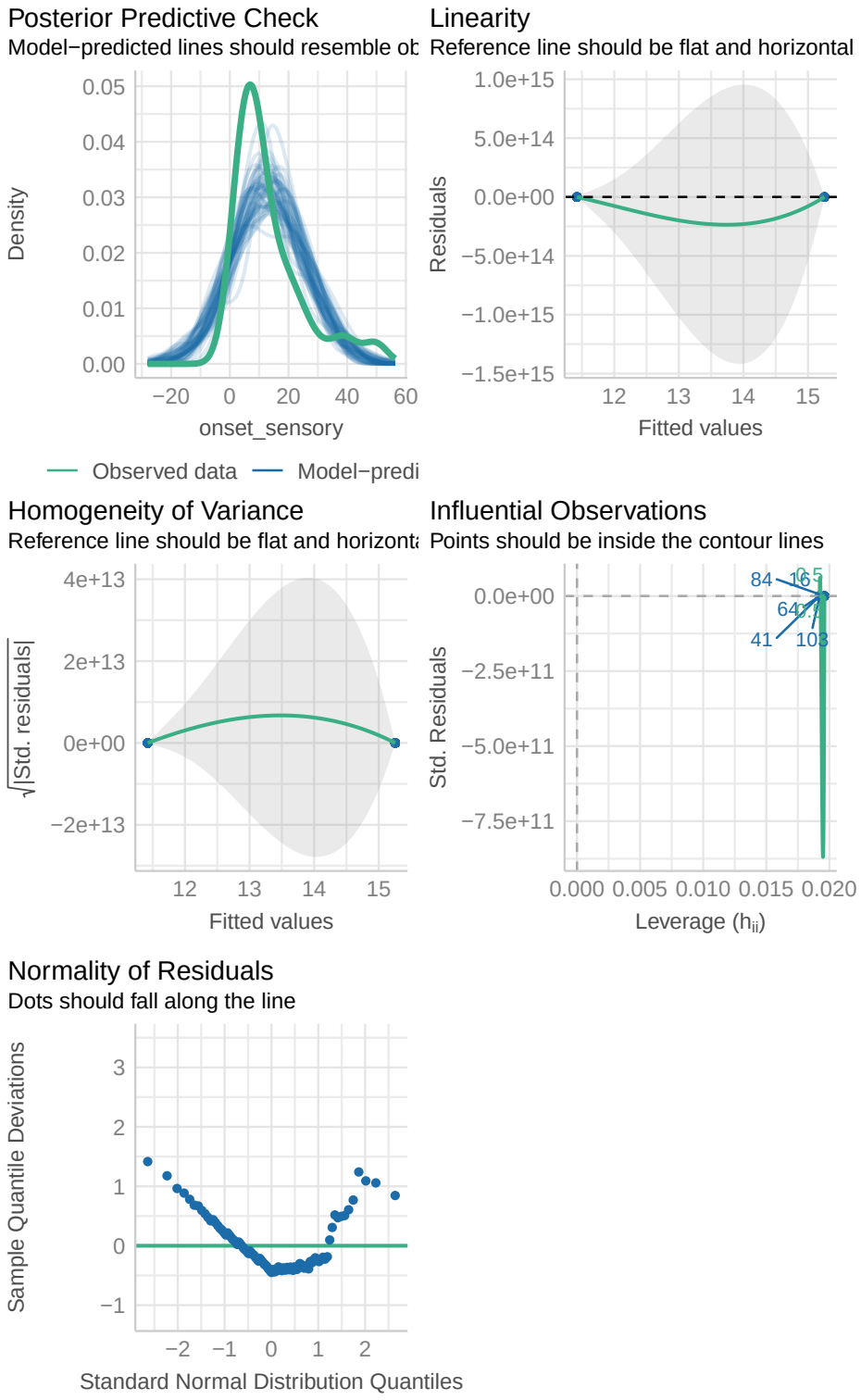
```
estimate_contrasts(fit1, contrast = "group")
```

```
Marginal Contrasts Analysis

Level1  |     Level2 | Difference |        95% CI |   SE | t(101) |     p
-------------------------------------------------------------------------
Mixture | Sequential |      -3.83 | [-8.43, 0.77] | 2.32 |  -1.65 | 0.102

Marginal contrasts estimated at group
p-value adjustment method: Holm (1979)
```

```
model_performance(fit1)
```

```
# Indices of model performance

AIC     |    AICc |     BIC |    R2 | R2 (adj.) |   RMSE |  Sigma
-----------------------------------------------------------------
804.175 | 804.418 | 812.079 | 0.026 |     0.017 | 11.655 | 11.769
```

```
check_model(fit1)
```


Posterior Predictive Check
Model−predicted lines should resemble ob
Density
0.05
0.04
0.03
0.02
0.01
0.00
−20
0
20
40
60
onset_sensory
Observed data
Model−predi
Linearity
Reference line should be flat and horizontal
Residuals
1.0e+15
5.0e+14
0.0e+00
−5.0e+14
−1.0e+15
−1.5e+15
12
13
14
15
Fitted values
Homogeneity of Variance
Reference line should be flat and horizonta
√|Std. residuals|
4e+13
2e+13
0e+00
−2e+13
12
13
14
15
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
0.0e+00
−2.5e+11
−5.0e+11
−7.5e+11
84
0.5
41
103
0.000
0.005
0.010
0.015
0.020
Leverage (hii)
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
3
2
1
0
−1
−2
−1
0
1
2
Standard Normal Distribution Quantiles

16.4.2 Bayesian linear model

```
set.seed(431)
fit2 <- stan_glm(onset_sensory ~ group, data = supra, refresh = 0)

model_parameters(fit2, ci = 0.95)
```

```
Parameter       | Median |         95% CI |     pd |  Rhat |     ESS |                    Pri
-------------------------------------------------------------------------------------------
(Intercept)     |  11.43 | [ 8.27, 14.66] |   100% | 1.001 | 4070.00 | Normal (13.32 +- 29.6
groupSequential |   3.79 | [-0.75,  8.32] | 94.95% | 1.001 | 4048.00 |  Normal (0.00 +- 59.0


Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
  using a MCMC distribution approximation.
```

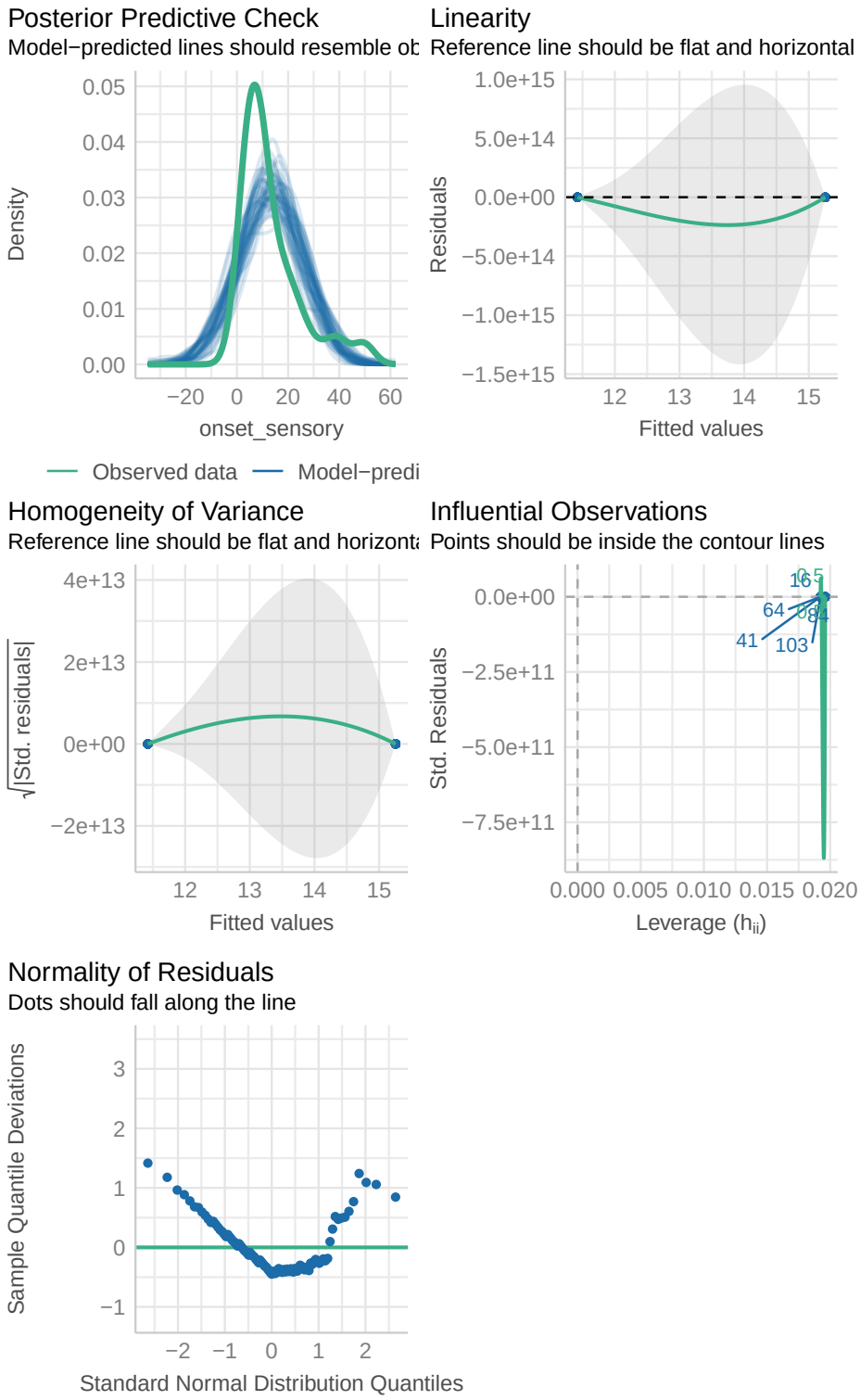
```
estimate_contrasts(fit2, contrast = "group")
```

```
Marginal Contrasts Analysis

Level1  |     Level2 | Difference |        95% CI |     pd | % in ROPE
-------------------------------------------------------------------------
Mixture | Sequential |      -3.79 | [-8.32, 0.75] | 94.95% |     1.16%

Marginal contrasts estimated at group
```

```
model_performance(fit2)
```

```
# Indices of model performance

ELPD     | ELPD_SE |   LOOIC | LOOIC_SE |    WAIC |    R2 | R2 (adj.) |   RMSE |  Sigma
-----------------------------------------------------------------------------------------
-402.665 |  10.611 | 805.331 |   21.222 | 805.304 | 0.026 |    -0.003 | 11.655 | 11.816
```

```
check_model(fit2)
```


Posterior Predictive Check
Model−predicted lines should resemble ob
Density
0.05
0.04
0.03
0.02
0.01
0.00
−20
0
20
40
60
onset_sensory
Observed data
Model−predi
Linearity
Reference line should be flat and horizontal
Residuals
1.0e+15
5.0e+14
0.0e+00
−5.0e+14
−1.0e+15
−1.5e+15
12
13
14
15
Fitted values
Homogeneity of Variance
Reference line should be flat and horizonta
√|Std. residuals|
4e+13
2e+13
0e+00
−2e+13
12
13
14
15
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
0.0e+00
−2.5e+11
−5.0e+11
−7.5e+11
165
64
84
41
103
0.000
0.005
0.010
0.015
0.020
Leverage (hii)
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
3
2
1
0
−1
−2
−1
0
1
2
Standard Normal Distribution Quantiles

16.5 Adjusting for Opioid Consumption

16.5.1 Least Squares Linear Model

```
fit3 <- lm(onset_sensory ~ group + opioid_total, data = supra)

model_parameters(fit3, ci = 0.95)
```

```
Parameter          | Coefficient |   SE |         95% CI | t(100) |      p
-------------------------------------------------------------------------
(Intercept)        |        8.35 | 1.85 | [ 4.68, 12.02] |   4.52 | < .001
group [Sequential] |        3.70 | 2.23 | [-0.71,  8.12] |   1.66 | 0.099
opioid total       |        0.06 | 0.02 | [ 0.02,  0.10] |   3.12 | 0.002


Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
  using a Wald t-distribution approximation.
```

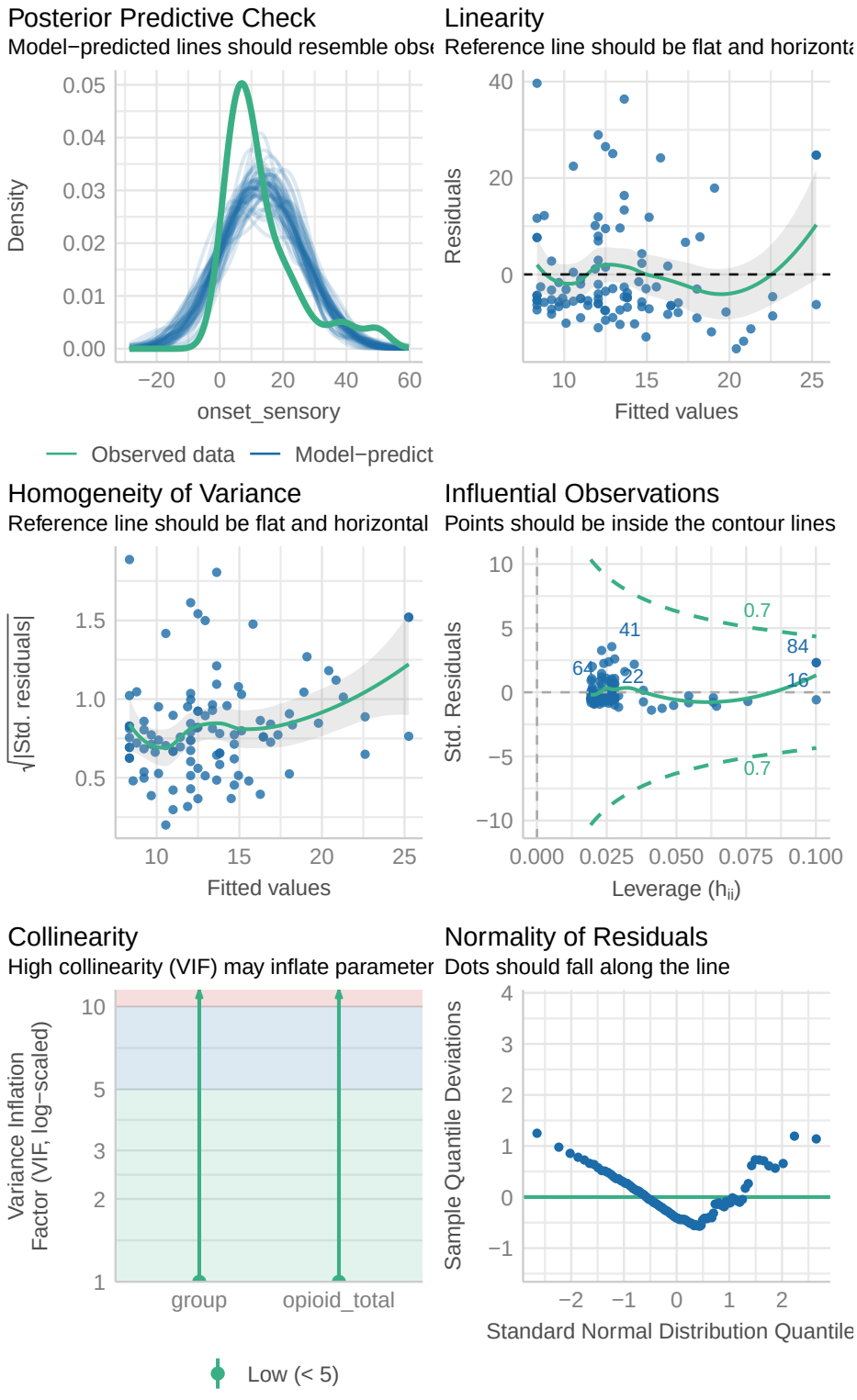
```
estimate_contrasts(fit3, contrast = "group")
```

```
Marginal Contrasts Analysis

Level1  |     Level2 | Difference |        95% CI |   SE | t(100) |     p
-------------------------------------------------------------------------
Mixture | Sequential |      -3.70 | [-8.12, 0.71] | 2.23 |  -1.66 | 0.099

Marginal contrasts estimated at group
p-value adjustment method: Holm (1979)
```

```
model_performance(fit3)
```

```
# Indices of model performance

AIC     |    AICc |     BIC |    R2 | R2 (adj.) |   RMSE |  Sigma
-----------------------------------------------------------------
796.609 | 797.017 | 807.148 | 0.113 |     0.095 | 11.126 | 11.291
```

```
check_model(fit3)
```


Posterior Predictive Check
Model−predicted lines should resemble obs
Density
onset_sensory
Observed data
Model−predict
Linearity
Reference line should be flat and horizonta
Residuals
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
Leverage (hii)
Collinearity
High collinearity (VIF) may inflate parameter
Variance Inflation Factor (VIF, log−scaled)
group
opioid_total
Low (< 5)
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
Standard Normal Distribution Quantile

16.5.2 Bayesian linear model

```
set.seed(431)
fit4 <- stan_glm(onset_sensory ~ group + opioid_total,
                 data = supra, refresh = 0)

model_parameters(fit4, ci = 0.95)
```

```
Parameter       | Median |         95% CI |     pd |  Rhat |     ESS |                   Pri
-------------------------------------------------------------------------------------------
(Intercept)     |   8.35 | [ 4.59, 12.00] |   100% | 0.999 | 4530.00 | Normal (13.32 +- 29.6
groupSequential |   3.75 | [-0.48,  8.16] | 95.45% | 1.000 | 5436.00 |  Normal (0.00 +- 59.0
opioid_total    |   0.06 | [ 0.02,  0.10] | 99.80% | 0.999 | 4662.00 |   Normal (0.00 +- 0.5

Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
  using a MCMC distribution approximation.
```

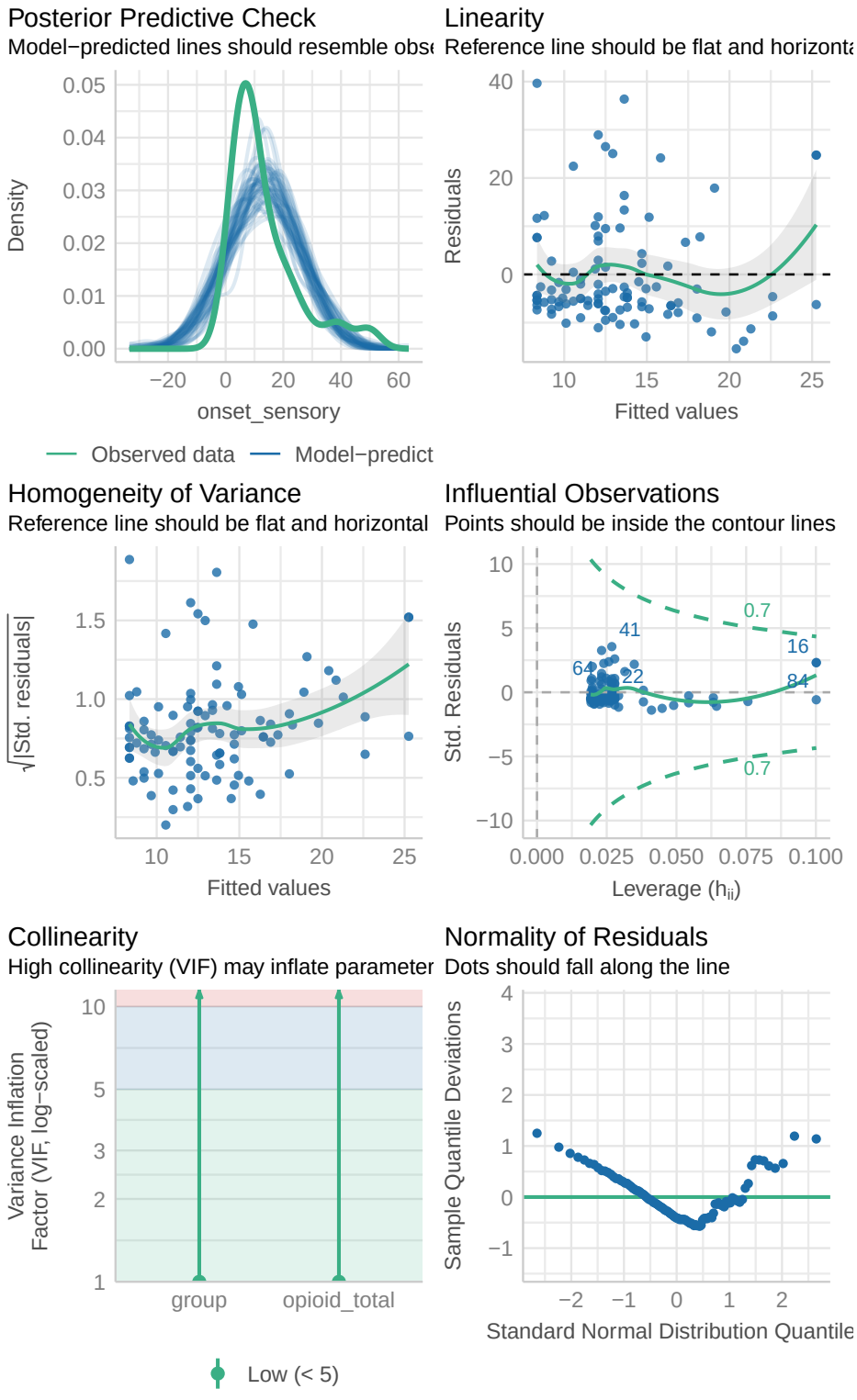
```
estimate_contrasts(fit4, contrast = "group")
```

```
Marginal Contrasts Analysis

Level1  |     Level2 | Difference |        95% CI |     pd | % in ROPE
-------------------------------------------------------------------------
Mixture | Sequential |      -3.75 | [-8.16, 0.48] | 95.45% |     0.97%

Marginal contrasts estimated at group
```

```
model_performance(fit4)
```

```
# Indices of model performance

ELPD     | ELPD_SE |   LOOIC | LOOIC_SE |    WAIC |    R2 | R2 (adj.) |   RMSE |  Sigma
-------------------------------------------------------------------------------------------
-399.338 |  10.630 | 798.676 |   21.260 | 798.573 | 0.118 |     0.053 | 11.126 | 11.315
```

```
check_model(fit4)
```


Posterior Predictive Check
Model−predicted lines should resemble obs
Density
onset_sensory
Observed data
Model−predict
Linearity
Reference line should be flat and horizonta
Residuals
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
Leverage (hii)
0.7
41
16
64
22
84
Collinearity
High collinearity (VIF) may inflate parameter
Variance Inflation Factor (VIF, log−scaled)
group
opioid_total
Low (< 5)
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
Standard Normal Distribution Quantile

16.6 What about a transformation of the outcome?

- discuss what to do here given that we have a zero in the outcome
- show the results on a multiplicative scale with exponentiate = TRUE

16.7 For More Information

431-book Chapter 16 reference OpenStats https://www.openintro.org/book/os/Section 8 - intro to linear regression

17 Nations of the World

> **!** This is a DRAFT version of this Chapter.
>
> This is a sketchy draft. I'll remove this notice when I post a version of this Chapter that is essentially finished.

17.1 R setup for this chapter

> **i** Note
>
> Appendix A lists all R packages used in this book, and also provides R session information. Appendix B describes the 431-Love.R script, and demonstrates its use.

```
library(broom)
library(car)
library(janitor)
library(knitr)
library(mice)
library(naniar)
library(patchwork)
library(rstanarm)

library(easystats)
library(tidyverse)

source("data/Love-431.R")
theme_set(theme_bw())
```

17.2 Data on Nations of the World

> **Note**
>
> Appendix C provides further guidance on pulling data from other systems into R, while Appendix D gives more information (including download links) for all data sets used in this book.

I collected these data from the World Health Organization and other sources, as of the end of May 2024. The primary resource was the Global Health Observatory from the WHO. Other resources included:

- The World Health Statistics Report
- Table of World Health Statistics 2024
- Gross Domestic Product Per Capita comes from the World Bank, accessed 2024-06-20.

```
nations <- read_csv("data/nations.csv", show_col_types = FALSE) |>
  mutate(
    UNIV_CARE = factor(UNIV_CARE),
    WHO_REGION = factor(WHO_REGION),
    C_NUM = as.character(C_NUM),
    GDP_PERCAP = GDP_PERCAP/1000
  ) |>
  janitor::clean_names()

glimpse(nations)
```

```
Rows: 192
Columns: 7
$ c_num        <chr> "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", ~
$ country_name <chr> "Afghanistan", "Albania", "Algeria", "Andorra", "Angola",~
$ iso_alpha3   <chr> "AFG", "ALB", "DZA", "AND", "AGO", "ATG", "ARG", "ARM", "~
$ uhc_index    <dbl> 41, 64, 74, 79, 37, 76, 79, 68, 87, 85, 66, 77, 76, 52, 7~
$ univ_care    <fct> no, yes, yes, no, no, no, yes, no, yes, yes, no, yes, no,~
$ gdp_percap   <dbl> 0.356, 6.810, 4.343, 41.993, 3.000, 19.920, 13.651, 7.018~
$ who_region   <fct> Eastern Mediterranean, European, African, European, Afric~
```

The data include information on 7 variables (listed below), describing 192 separate countries of the world.

> **Note**
>
> By Country, we mean a sovereign state that is a member of both the United Nations and the World Health Organization, in its own right. Note that Liechtenstein is not a member of the World Health Organization, but is in the UN. Liechtenstein is the only UN member nation not included in the `nations` data.

Variable	Description
`c_num`	Country ID (alphabetical by country_name)
`country_name`	Name of Country (according to WHO and UN)
`iso_alpha3`	ISO Alphabetical 3-letter code gathered from WHO
`uhc_index`	Universal Health Coverage: Service coverage index[1] - higher numbers indicate better coverage.
`univ_care`	Yes if country offers government-regulated and government-funded health care to more than 90% of its citizens as of 2023, else No
`gdp_percap`	GDP Per Capita per the World Bank in thousands of 2023 US dollars[2]
`who_region`	Six groups[3] according to regional distribution

17.3 Exploratory Data Analysis

Our interest is in understanding the association of `uhc_index` and `univ_care`, perhaps after adjustment for the country's `gdp_percap`.

17.3.1 EDA for our outcome

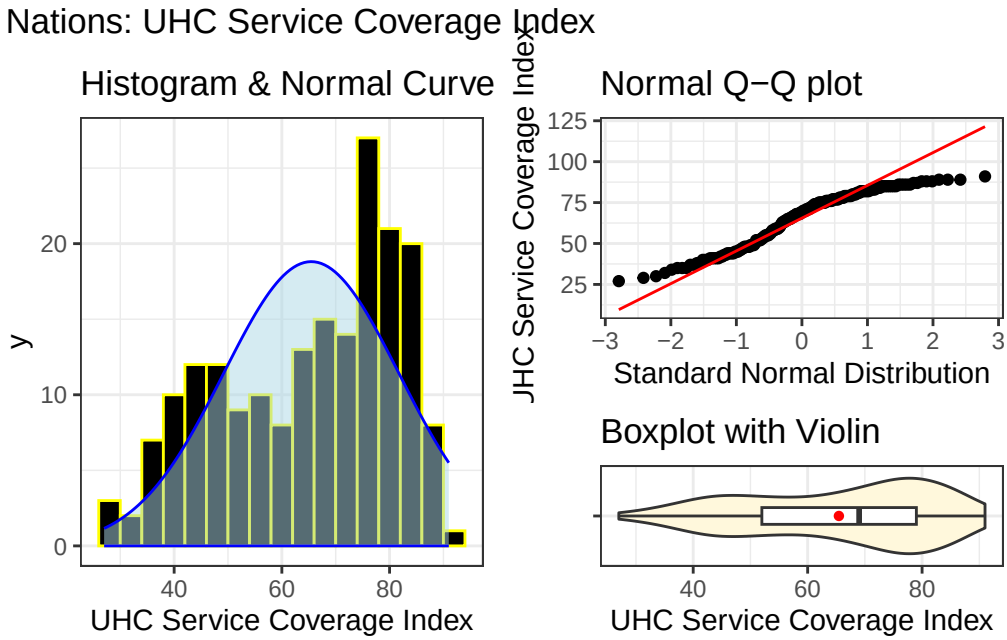
```
bw = 4 # specify width of bins in histogram

p1 <- ggplot(nations, aes(uhc_index)) +
  geom_histogram(binwidth = bw,
                 fill = "black", col = "yellow") +
  stat_function(fun = function(x)
    dnorm(x, mean = mean(nations$uhc_index,
```

[1] This is a measure of coverage of essential health services as expressed as the average score of 14 tracer indicators of health service coverage, measured in 2021.

[2] Some of the GDP data describes 2021, some 2022 and some 2023, depending on what was most recent and available.

[3] The WHO regions are African, Americas, Eastern Mediterranean, European, Southeast Asian and Western Pacific.

```
                          na.rm = TRUE),
          sd = sd(nations$uhc_index,
                  na.rm = TRUE)) *
      length(nations$uhc_index) * bw,
    geom = "area", alpha = 0.5,
    fill = "lightblue", col = "blue"
    ) +
  labs(
    x = "UHC Service Coverage Index",
    title = "Histogram & Normal Curve"
  )

p2 <- ggplot(nations, aes(sample = uhc_index)) +
  geom_qq() +
  geom_qq_line(col = "red") +
  labs(
    y = "UHC Service Coverage Index",
    x = "Standard Normal Distribution",
    title = "Normal Q-Q plot"
  )

p3 <- ggplot(nations, aes(x = uhc_index, y = "")) +
  geom_violin(fill = "cornsilk") +
  geom_boxplot(width = 0.2) +
  stat_summary(
    fun = mean, geom = "point",
    shape = 16, col = "red"
  ) +
  labs(
    y = "", x = "UHC Service Coverage Index",
    title = "Boxplot with Violin"
  )

p1 + (p2 / p3 + plot_layout(heights = c(2, 1))) +
  plot_annotation(
    title = "Nations: UHC Service Coverage Index"
  )
```

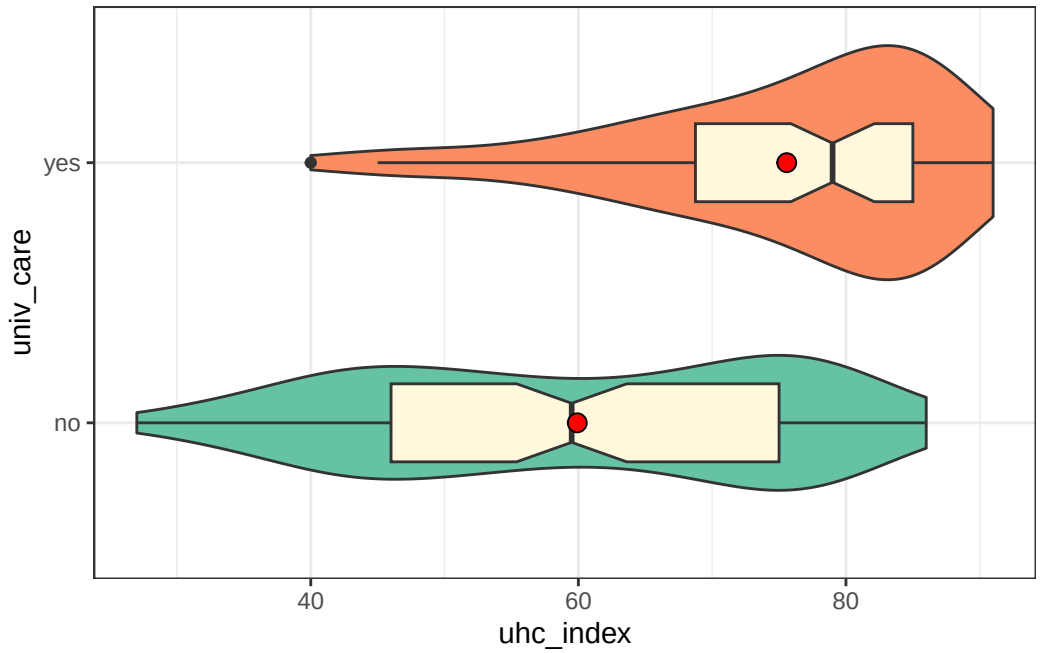



```
nations |>
  reframe(lovedist(uhc_index)) |>
  kable(digits = 2)
```

n	miss	mean	sd	med	mad	min	q25	q75	max
192	0	65.46	16.29	69	19.27	27	52	79	91

17.3.2 UHC Service Coverage by Universal Care Status

```
ggplot(nations, aes(y = univ_care, x = uhc_index,
                    fill = univ_care)) +
  geom_violin() +
  geom_boxplot(width = 0.3, fill = "cornsilk", notch = TRUE) +
  stat_summary(fun = mean, geom = "point", fill = "red",
               size = 3, shape = 21) +
  scale_fill_brewer(palette = "Set2") +
  guides(fill = "none")
```



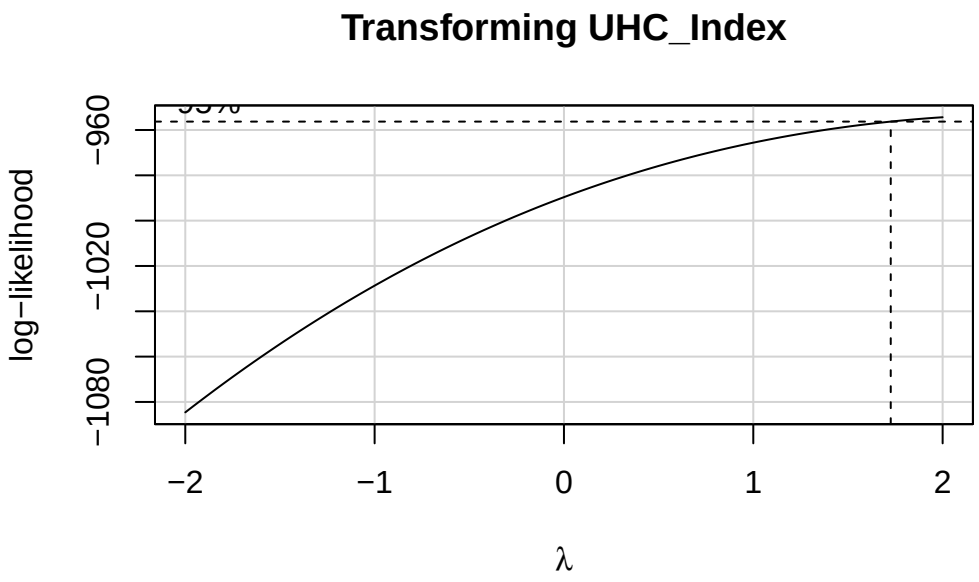

```
nations |>
  reframe(lovedist(uhc_index), .by = univ_care) |>
  kable(digits = 2)
```

univ_care	n	miss	mean	sd	med	mad	min	q25	q75	max
no	124	0	59.92	15.77	59.5	21.50	27	46.00	75	86
yes	68	0	75.57	11.79	79.0	9.64	40	68.75	85	91

17.4 Should we transform our outcome?

Eventually, we will consider including the `gdp_percap` predictor in our model, so let's include that here, as well as `univ_care` when predicting `uhc_index`.

```
fit1 <- lm(uhc_index ~ univ_care + gdp_percap,
           data = nations)

boxCox(fit1, main = "Transforming UHC_Index")
```




```
summary(powerTransform(fit1))$result
```

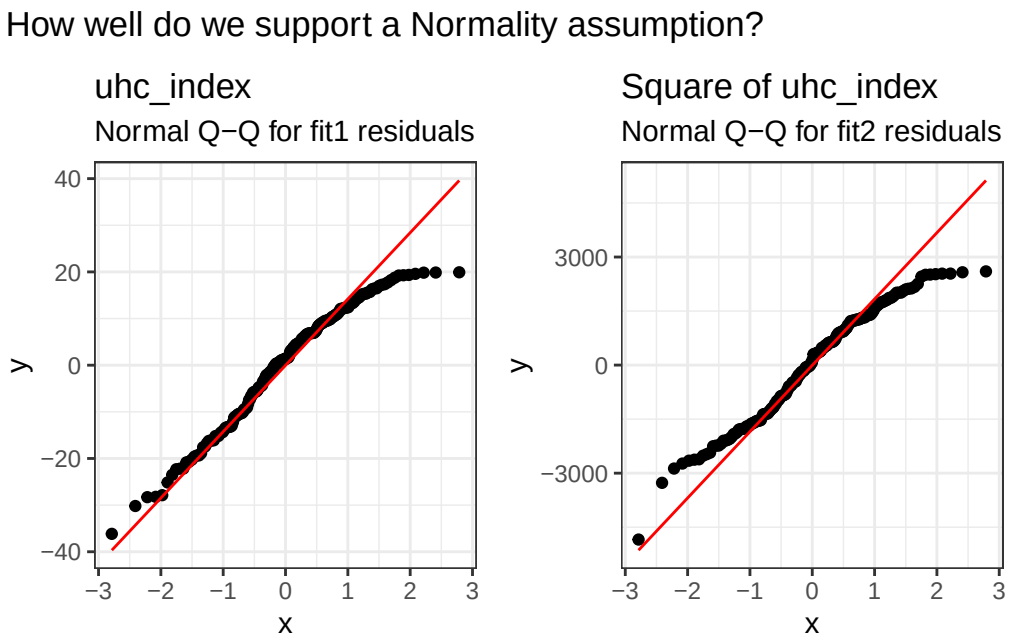
```
   Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
Y1  2.532079           2     1.908487     3.155671
```

17.4.1 Looking at Residual Plots

We might consider whether the residuals from our two models (with and without transforming the outcome) follow the assumption of a Normal distribution.

```
fit2 <- lm((uhc_index^2) ~ univ_care + gdp_percap,
           data = nations)

resids_df <- tibble(raw = fit1$residuals,
                    sqr = fit2$residuals)

p1 <- ggplot(resids_df, aes(sample = raw)) +
  geom_qq() + geom_qq_line(col = "red") +
  labs(title = "uhc_index",
       subtitle = "Normal Q-Q for fit1 residuals")
```

```
p2 <- ggplot(resids_df, aes(sample = sqr)) +
  geom_qq() + geom_qq_line(col = "red") +
  labs(title = "Square of uhc_index",
       subtitle = "Normal Q-Q for fit2 residuals")

p1 + p2 +
  plot_annotation("How well do we support a Normality assumption?")
```



In terms of providing a better match to the assumption of Normality in the residuals, the square transformation doesn't appear to be very helpful. If anything, our problem with light tails (fewer values far from the mean than expected) may get a little worse in fit2. So we'll skip the transformation and work with our original outcome in what follows.

17.5 Least Squares Linear Model

```
fit3 <- lm(uhc_index ~ univ_care, data = nations)

model_parameters(fit3, ci = 0.95)
```

```
Parameter       | Coefficient |   SE |         95% CI | t(190) |      p
```

```
-------------------------------------------------------------------------
(Intercept)     |       59.92 | 1.30 | [57.35, 62.49] |  46.04 | < .001
univ care [yes] |       15.65 | 2.19 | [11.34, 19.97] |   7.16 | < .001


Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
  using a Wald t-distribution approximation.
```

```
estimate_contrasts(fit3, contrast = "univ_care", ci = 0.95)
```

```
Marginal Contrasts Analysis

Level1 | Level2 | Difference |           95% CI |   SE | t(190) |      p
-------------------------------------------------------------------------
no     |    yes |     -15.65 | [-19.97, -11.34] | 2.19 |  -7.16 | < .001

Marginal contrasts estimated at univ_care
p-value adjustment method: Holm (1979)
```

17.5.1 Interpreting the Fit

When comparing two nations where one offers universal health care and the other does not, the country that offers universal care will, on average, have a UHC index score that is 15.7 points higher.

Under this fitted model, the average difference in UHC index score between a country that offers universal health care and one that does not is 15.7 points (95% uncertainty interval for the estimated difference is 11.34, 19.97 points.)

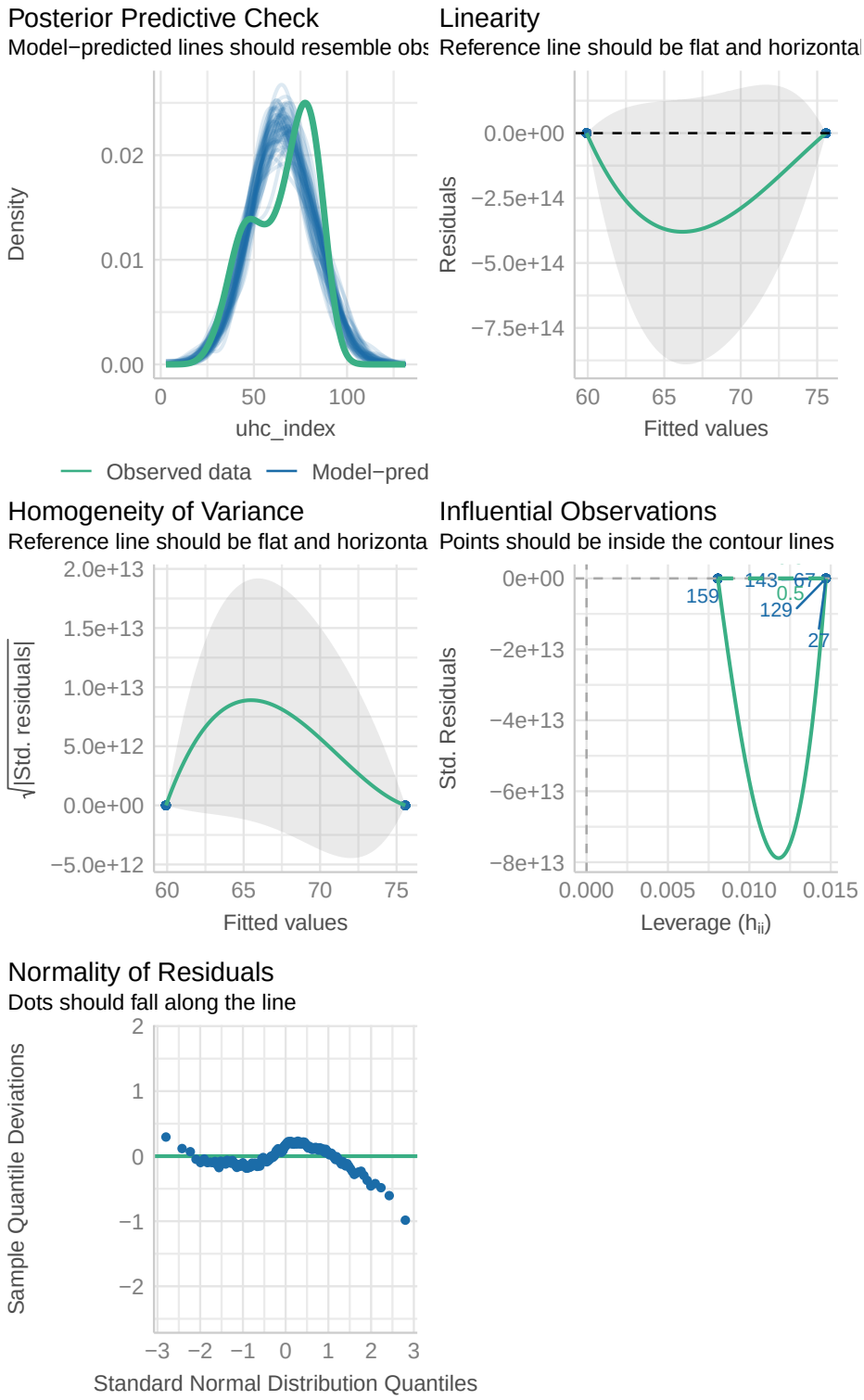
17.5.2 Performance of the Model

```
model_performance(fit3)
```

```
# Indices of model performance

AIC      |     AICc |      BIC |    R2 | R2 (adj.) |   RMSE |  Sigma
-----------------------------------------------------------------------
1575.544 | 1575.672 | 1585.317 | 0.212 |     0.208 | 14.417 | 14.493
```

17.5.3 Checking the Model

```
check_model(fit3)
```


Posterior Predictive Check
Model−predicted lines should resemble obs
Density
0.02
0.01
0.00
0
50
100
uhc_index
Observed data
Model−pred
Linearity
Reference line should be flat and horizontal
Residuals
0.0e+00
−2.5e+14
−5.0e+14
−7.5e+14
60
65
70
75
Fitted values
Homogeneity of Variance
Reference line should be flat and horizonta
√|Std. residuals|
2.0e+13
1.5e+13
1.0e+13
5.0e+12
0.0e+00
−5.0e+12
60
65
70
75
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
0e+00
−2e+13
−4e+13
−6e+13
−8e+13
159
143
67
0.5
129
27
0.000
0.005
0.010
0.015
Leverage (hii)
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
2
1
0
−1
−2
−3
−2
−1
0
1
2
3
Standard Normal Distribution Quantiles

17.6 Bayesian Linear Model

```
set.seed(431)
fit4 <- stan_glm(uhc_index ~ univ_care, data = nations, refresh = 0)

model_parameters(fit4, ci = 0.95)
```

```
Parameter    | Median |         95% CI |   pd |  Rhat |     ESS |                   Prior
-------------------------------------------------------------------------------------------
(Intercept)  |  59.92 | [57.29, 62.50] | 100% | 1.000 | 4052.00 | Normal (65.46 +- 40.72)
univ_careyes |  15.69 | [11.28, 19.94] | 100% | 1.000 | 3679.00 |  Normal (0.00 +- 84.92)


Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
  using a MCMC distribution approximation.
```

```
estimate_contrasts(fit4, contrast = "univ_care", ci = 0.95)
```

```
Marginal Contrasts Analysis

Level1 | Level2 | Difference |           95% CI |   pd | % in ROPE
-------------------------------------------------------------------
no     |    yes |     -15.69 | [-19.94, -11.28] | 100% |        0%

Marginal contrasts estimated at univ_care
```

17.6.1 Interpreting the Fit

Under this fitted Bayesian model (fit4) which uses a weakly informative prior, the estimated median difference in UHC index score between a country that offers universal health care and one that does not is 15.7 points (95% credible interval for the median difference ranges from 11.28 to 19.94 points.)

The estimated probability that the median (universal care - no universal care) difference in UHC index scores is positive is 100%, according to the fit4 model. None of the simulated results comparing the two groups (universal care yes vs. no) fall in the region of practical equivalence.

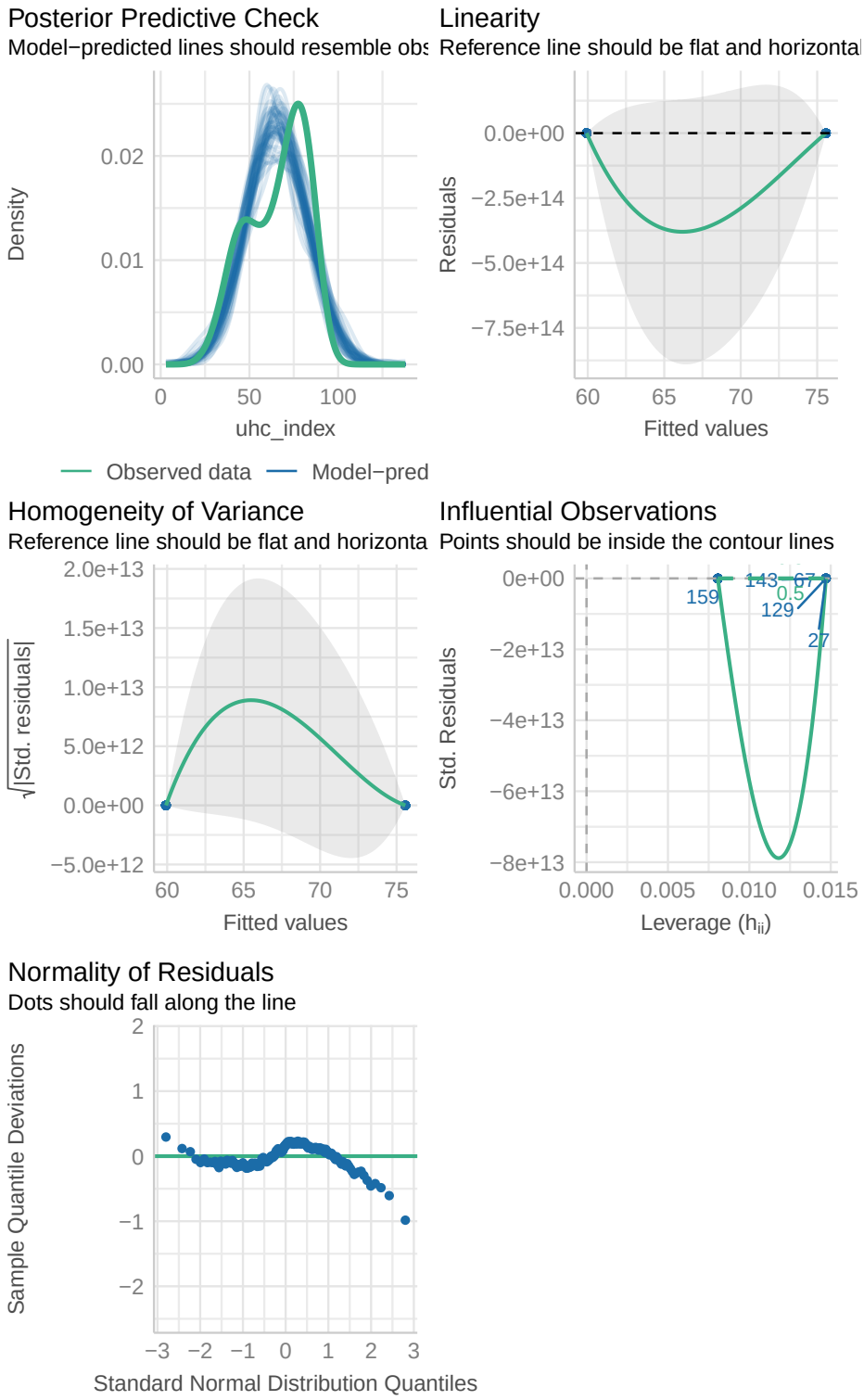
17.6.2 Performance of the Model

```
model_performance(fit4)
```

```
# Indices of model performance

ELPD     | ELPD_SE |    LOOIC | LOOIC_SE |     WAIC |    R2 | R2 (adj.) |   RMSE |  Sigma
-------------------------------------------------------------------------------------------
-787.518 |   7.559 | 1575.035 |   15.119 | 1575.027 | 0.212 |     0.202 | 14.417 | 14.518
```

17.6.3 Checking the Model

```
check_model(fit4)
```


Posterior Predictive Check
Model−predicted lines should resemble obs
Density
0.02
0.01
0.00
0
50
100
uhc_index
Observed data
Model−pred
Linearity
Reference line should be flat and horizontal
Residuals
0.0e+00
−2.5e+14
−5.0e+14
−7.5e+14
60
65
70
75
Fitted values
Homogeneity of Variance
Reference line should be flat and horizonta
√|Std. residuals|
2.0e+13
1.5e+13
1.0e+13
5.0e+12
0.0e+00
−5.0e+12
60
65
70
75
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
0e+00
−2e+13
−4e+13
−6e+13
−8e+13
159
143
67
0.5
129
27
0.000
0.005
0.010
0.015
Leverage (hii)
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
2
1
0
−1
−2
−3
−2
−1
0
1
2
3
Standard Normal Distribution Quantiles

17.7 Missing Data Mechanisms

We now tackle the issue of missing data.

My source for the following description of mechanisms is Chapter 25 of Gelman and Hill (2007), and that chapter is available at this link.

1. **MCAR = Missingness completely at random**. A variable is missing completely at random if the probability of missingness is the same for all units, for example, if for each subject, we decide whether to collect `gdp_percap` by rolling a die and refusing to answer if a "6" shows up. If data are missing completely at random, then throwing out cases with missing data does not bias your inferences.

> **Note**
>
> We have tacitly made the MCAR assumption in all of our prior dealings with missing data in this book.

2. **Missingness that depends only on observed predictors**. A more general assumption, called **missing at random** or **MAR**, is that the probability a variable is missing depends only on available information. Here, we would have to be willing to assume that the probability of non-response to `gdp_percap` depends only on the other, fully recorded variables in the data. It is often reasonable to model this process as a logistic regression, where the outcome variable equals 1 for observed cases and 0 for missing. When an outcome variable is missing at random, it is acceptable to exclude the missing cases (that is, to treat them as NA), as long as the regression adjusts for all of the variables that affect the probability of missingness.
3. **Missingness that depends on unobserved predictors**. Missingness is no longer "at random" if it depends on information that has not been recorded and this information also predicts the missing values. If a particular treatment causes discomfort, a patient is more likely to drop out of the study. This missingness is not at random (unless "discomfort" is measured and observed for all patients). If missingness is not at random, it must be explicitly modeled, or else you must accept some bias in your inferences.
4. **Missingness that depends on the missing value itself.** Finally, a particularly difficult situation arises when the probability of missingness depends on the (potentially missing) variable itself. For example, suppose that people with higher earnings are less likely to reveal them.

Essentially, situations 3 and 4 are referred to collectively as **non-random missingness**, and cause more trouble for us than 1 and 2.

17.8 Dealing with Missing Data

There are several available methods for dealing with missing data that are MCAR or MAR, but they basically boil down to:

- Complete Case (or Available Case) analyses
- Single Imputation
- Multiple Imputation

17.8.1 Complete Case (and Available Case) analyses

In **Complete Case** analyses, rows containing NA values are omitted from the data before analyses commence. This is the default approach for many statistical software packages, and may introduce unpredictable bias and fail to include some useful, often hard-won information.

- A complete case analysis can be appropriate when the number of missing observations is not large, and the missing pattern is either MCAR (missing completely at random) or MAR (missing at random.)
- Two problems arise with complete-case analysis:
    1. If the units with missing values differ systematically from the completely observed cases, this could bias the complete-case analysis.
    2. If many variables are included in a model, there may be very few complete cases, so that most of the data would be discarded for the sake of a straightforward analysis.
- A related approach is *available-case* analysis where different aspects of a problem are studied with different subsets of the data, perhaps identified on the basis of what is missing in them.

17.8.2 Single Imputation

In **single imputation** analyses, NA values are estimated/replaced *one time* with *one particular data value* for the purpose of obtaining more complete samples, at the expense of creating some potential bias in the eventual conclusions or obtaining slightly *less* accurate estimates than would be available if there were no missing values in the data.

- A single imputation can be just a replacement with the mean or median (for a quantity) or the mode (for a categorical variable.) However, such an approach, though easy to understand, underestimates variance and ignores the relationship of missing values to other variables.
- Single imputation can also be done using a variety of models to try to capture information about the NA values that are available in other variables within the data set.

- In this book, we will use the `mice` package to perform single imputations, as well as multiple imputation.

17.8.3 Multiple Imputation

Multiple imputation, where NA values are repeatedly estimated/replaced with multiple data values, for the purpose of obtaining mode complete samples *and* capturing details of the variation inherent in the fact that the data have missingness, so as to obtain *more* accurate estimates than are possible with single imputation.

How many imputations?

Quoting vanBuuren at https://stefvanbuuren.name/fimd/sec-howmany.html

> They take a quote from Von Hippel (2009) as a rule of thumb: the number of imputations should be similar to the percentage of cases that are incomplete. This rule applies to fractions of missing information of up to 0.5.
>
> White, Royston, and Wood (2011) suggest these criteria provide an adequate level of reproducibility in practice. The idea of reproducibility is sensible, the rule is simple to apply, so there is much to commend it. The rule has now become the de-facto standard, especially in medical applications.
>
> It is convenient to set m = 5 during model building, and increase m only after being satisfied with the model for the "final" round of imputation. So if calculation is not prohibitive, we may set m to the average percentage of missing data. The substantive conclusions are unlikely to change as a result of raising m beyond m = 5.

17.9 Nations and Missing Data

In our study, if we want to adjust our estimates for the impact of per-capita GDP, we must deal with the fact that these data are missing for four of the nations in our data.

```
miss_var_summary(nations)
```

```
# A tibble: 7 x 3
  variable     n_miss pct_miss
  <chr>         <int>    <num>
1 gdp_percap        4     2.08
2 c_num             0     0
3 country_name      0     0
```

```
4 iso_alpha3        0     0
5 uhc_index         0     0
6 univ_care         0     0
7 who_region        0     0
```

```
nations |> filter(is.na(gdp_percap)) |>
  select(country_name, iso_alpha3, gdp_percap, uhc_index, univ_care) |>
  kable()
```

country_name	iso_alpha3	gdp_percap	uhc_index	univ_care
Democratic People's Republic of Korea	PRK	NA	68	yes
Eritrea	ERI	NA	45	no
South Sudan	SSD	NA	34	no
Venezuela (Bolivarian Republic of)	VEN	NA	75	no

17.9.1 Complete Case Analysis

We can drop all of the missing values from a data set with `drop_na` or with `na.omit` or by filtering for `complete.cases`. Any of these approaches produces the same result.

> **Note**
>
> If we don't do anything about missing data, and simply feed the results to the `lm()` or `stan_glm()` functions, the machine will produce a fit on the complete cases.

```
fit5 <- lm(uhc_index ~ univ_care + gdp_percap,
           data = nations)
```

```
summary(fit5)
```

```

Call:
lm(formula = uhc_index ~ univ_care + gdp_percap, data = nations)

Residuals:
    Min      1Q  Median      3Q     Max
-36.161  -9.625   1.336   9.558  19.917

Coefficients:
```

```
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  57.02924    1.20223  47.436  < 2e-16 ***
univ_careyes 11.04387    1.98727   5.557 9.44e-08 ***
gdp_percap    0.27041    0.03414   7.920 2.14e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 12.5 on 185 degrees of freedom
  (4 observations deleted due to missingness)
Multiple R-squared:  0.4117,    Adjusted R-squared:  0.4053
F-statistic: 64.73 on 2 and 185 DF,  p-value: < 2.2e-16
```

```
model_parameters(fit5, ci = 0.95)
```

```
Parameter       | Coefficient |   SE |         95% CI | t(185) |      p
-------------------------------------------------------------------------
(Intercept)     |       57.03 | 1.20 | [54.66, 59.40] |  47.44 | < .001
univ care [yes] |       11.04 | 1.99 | [ 7.12, 14.96] |   5.56 | < .001
gdp percap      |        0.27 | 0.03 | [ 0.20,  0.34] |   7.92 | < .001


Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
  using a Wald t-distribution approximation.
```

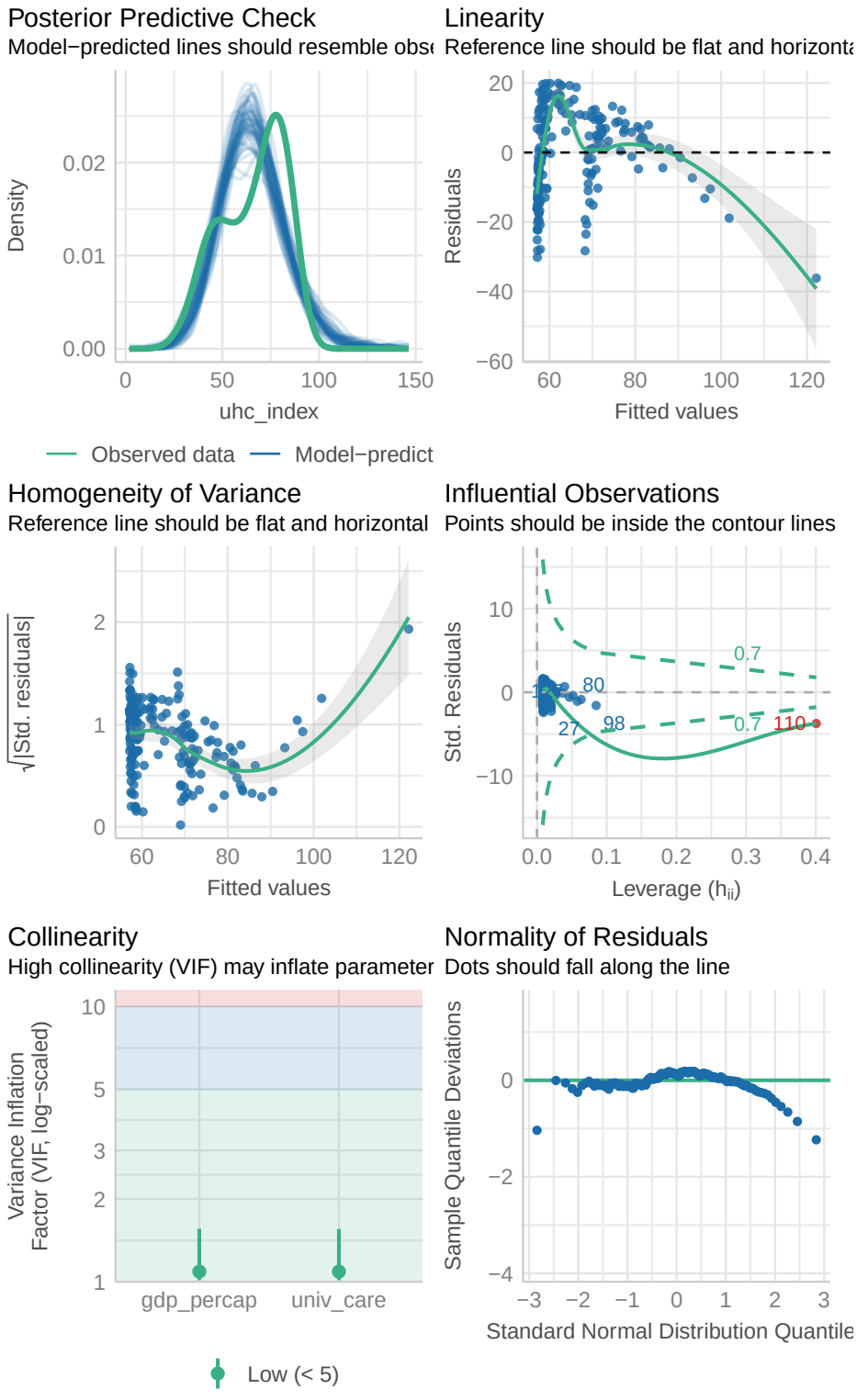
```
model_performance(fit5)
```

```
# Indices of model performance

AIC      |     AICc |      BIC |    R2 | R2 (adj.) |   RMSE |  Sigma
-----------------------------------------------------------------------
1488.247 | 1488.465 | 1501.192 | 0.412 |     0.405 | 12.402 | 12.503
```

Note the indication that this model fit has had four observations deleted due to missingness.

Finally, we check the model...

```
check_model(fit5)
```


Posterior Predictive Check
Model−predicted lines should resemble obs
Density
0.02
0.01
0.00
0
50
100
150
uhc_index
Observed data
Model−predict
Linearity
Reference line should be flat and horizonta
Residuals
20
0
−20
−40
−60
60
80
100
120
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
2
1
0
60
80
100
120
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
10
0
−10
0.7
80
27
98
0.7
110
0.0
0.1
0.2
0.3
0.4
Leverage (hii)
Collinearity
High collinearity (VIF) may inflate parameter
Variance Inflation Factor (VIF, log−scaled)
10
5
3
2
1
gdp_percap
univ_care
Low (< 5)
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
0
−2
−4
−3
−2
−1
0
1
2
3
Standard Normal Distribution Quantile

17.9.2 Single Imputation

Here is how I produced a single imputation of the nations data in R using the `mice` package, and some default choices, so that I could perform a regression analysis incorporating all of the nations, including those with missing `gdp_percap` values.

> **Note**
>
> By default, mice uses predictive mean matching (or pmm), for numeric data. It uses logistic regression imputation for binary data, polytomous regression imputation for unordered categorical data with more than 2 levels, and a proportional odds regression model for ordered categorical data with more than 2 levels.
> All of these methodologies will be explored in the 432 class, but a detailed explanation is beyond the scope of this book.

```
nations_simp <-
  mice(nations, m = 1, seed = 431, print = FALSE) |>
  complete() |>
  tibble()
```

```
Warning: Number of logged events: 3
```

Here, `m` is the number of imputations I want to create (here just one), `seed` is the random seed I set so that I can replicate the results later, and `print` is set to FALSE to reduce the amount of unnecessary output this produces. I will ignore this warning about logged events in most cases, including here.

> **Tip**
>
> If you want to see the logged events, try:

```
ini <- mice(nations, m = 1, seed = 431, print = FALSE)
```

```
Warning: Number of logged events: 3
```

```
ini$loggedEvents
```

```
  it im dep     meth          out
1  0  0     constant        c_num
2  0  0     constant country_name
3  0  0     constant   iso_alpha3
```

> Here, our `nations` data include three variables which are not included in the imputation process (and which we don't want included) so all is well.

Here's what I wind up with:

```
nations_simp
```

```
# A tibble: 192 x 7
   c_num country_name       iso_alpha3 uhc_index univ_care gdp_percap who_region
   <chr> <chr>              <chr>          <dbl> <fct>          <dbl> <fct>
 1 1     Afghanistan        AFG               41 no             0.356 Eastern M~
 2 2     Albania            ALB               64 yes            6.81  European
 3 3     Algeria            DZA               74 yes            4.34  African
 4 4     Andorra            AND               79 no            42.0   European
 5 5     Angola             AGO               37 no             3     African
 6 6     Antigua and Barbu~ ATG               76 no            19.9   Americas
 7 7     Argentina          ARG               79 yes           13.7   Americas
 8 8     Armenia            ARM               68 no             7.02  European
 9 9     Australia          AUS               87 yes           65.1   Western P~
10 10    Austria            AUT               85 yes           52.1   European
# i 182 more rows
```

```
n_miss(nations_simp)
```

```
[1] 0
```

Now, I can do everything I did previously with this singly imputed data set.

```
fit6 <- lm(uhc_index ~ univ_care + gdp_percap,
           data = nations_simp)
```

```
summary(fit6)
```

```

Call:
lm(formula = uhc_index ~ univ_care + gdp_percap, data = nations_simp)

Residuals:
    Min      1Q  Median      3Q     Max
```

```
-36.216  -9.940   1.337   9.692  20.062

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  56.86773    1.19475  47.598  < 2e-16 ***
univ_careyes 11.16974    1.98286   5.633 6.34e-08 ***
gdp_percap    0.27131    0.03427   7.916 2.02e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 12.59 on 189 degrees of freedom
Multiple R-squared:  0.4085,    Adjusted R-squared:  0.4023
F-statistic: 65.27 on 2 and 189 DF,  p-value: < 2.2e-16
```

```r
model_parameters(fit6, ci = 0.95)
```

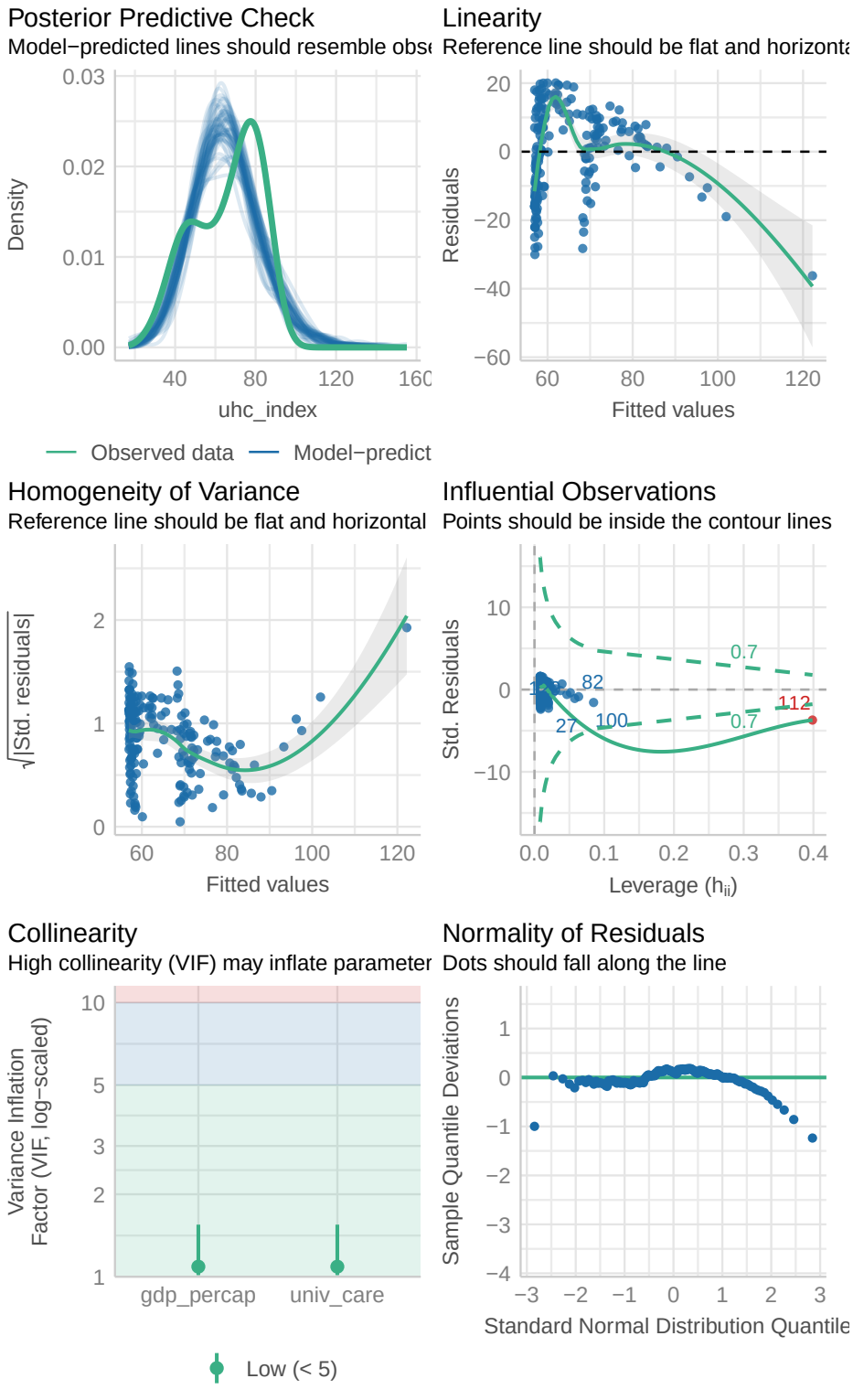
```
Parameter       | Coefficient |   SE |         95% CI | t(189) |      p
-------------------------------------------------------------------------
(Intercept)     |       56.87 | 1.19 | [54.51, 59.22] |  47.60 | < .001
univ care [yes] |       11.17 | 1.98 | [ 7.26, 15.08] |   5.63 | < .001
gdp percap      |        0.27 | 0.03 | [ 0.20,  0.34] |   7.92 | < .001


Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
  using a Wald t-distribution approximation.
```

```r
model_performance(fit6)
```

```
# Indices of model performance

AIC      |     AICc |      BIC |    R2 | R2 (adj.) |   RMSE |  Sigma
-----------------------------------------------------------------------
1522.564 | 1522.778 | 1535.594 | 0.409 |     0.402 | 12.494 | 12.593
```

And, again, we check the model…

```r
check_model(fit6)
```


Posterior Predictive Check
Model−predicted lines should resemble obs
Density
uhc_index
Observed data
Model−predict
Linearity
Reference line should be flat and horizonta
Residuals
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
Leverage (hii)
0.7
82
27
100
112
Collinearity
High collinearity (VIF) may inflate parameter
Variance Inflation Factor (VIF, log−scaled)
gdp_percap
univ_care
Low (< 5)
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
Standard Normal Distribution Quantile

17.9.3 Multiple Imputation

The first thing we might consider is how many multiple imputations we want to do.

```
nations |> prop_miss_case()
```

```
[1] 0.02083333
```

2% of the cases in our `nations` tibble contain missing values, so in selecting the number of imputations, I'd go for the most commonly used option. Let's try 5 imputations.

```
nations_imp5 <- mice(nations, m = 5, seed = 431, print = FALSE)
```

```
Warning: Number of logged events: 3
```

Here, `m` is the number of imputations I want to create (five, in our case), `seed` is the random seed I set so that I can replicate the results later, and `print` is set to FALSE to reduce the amount of unnecessary output this produces. I will ignore this warning about logged events in most cases, including here.

Here is how I would fit a set of five models, each using a different one of the five imputed data sets and obtain a summary of my results:

```
est7 <- nations |>
  mice(seed = 431, print = FALSE) |>
  with(lm(uhc_index ~ univ_care + gdp_percap)) |>
  pool()
```

```
Warning: Number of logged events: 3
```

```
model_parameters(est7, ci = 0.95)
```

```
Warning: Number of logged events: 3
Warning: Number of logged events: 3
Warning: Number of logged events: 3
Warning: Number of logged events: 3
```

```
# Fixed Effects

Parameter       | Coefficient |   SE |         95% CI |     t |     df |      p
-------------------------------------------------------------------------------
(Intercept)     |       56.81 | 1.19 | [54.46, 59.17] | 47.58 | 186.81 | < .001
univ care [yes] |       11.16 | 1.98 | [ 7.26, 15.06] |  5.64 | 187.01 | < .001
gdp percap      |        0.27 | 0.03 | [ 0.21,  0.34] |  7.98 | 186.80 | < .001


Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
  using a Wald t-distribution approximation.
```

```
glance(est7)
```

```
  nimp nobs r.squared adj.r.squared
1    5  192 0.4111089     0.4048772
```

17.10 For More Information

See https://library.virginia.edu/data/articles/getting-started-with-multiple-imputation-in-r

- Reference there to Rubin (1976)

  Rubin (1976) classified types of missing data in three categories: MCAR, MAR, MNAR

  MCAR: Missing Completely at Random - the reason for the missingness of data points are at random, meaning that the pattern of missing values is uncorrelated with the structure of the data. An example would be a random sample taken from the population: data on some people will be missing, but it will be at random since everyone had the same chance of being included in the sample.

  MAR: Missing at Random - the missingness is not completely random, but the propensity of missingness depends on the observed data, not the missing data. An example would be a survey respondent choosing not to answer a question on income because they believe the privacy of personal information. As seen in this case, the missing value for income can be predicted by looking at the answers for the personal information question.

MNAR: Missing Not at Random - the missing is not random, it correlates with unobservable characteristics unknown to a researcher. An example would be social desirability bias in survey - where respondents with certain characteristics we can't observe systematically shy away from answering questions on racial issues.

All multiple imputation techniques start with the MAR assumption. While MCAR is desirable, in general it is unrealistic for the data. Thus, researchers make the assumption that missing values can be replaced by predictions derived by the observable portion of the dataset. This is a fundamental assumption to make, otherwise we wouldn't be able to predict plausible values of missing data points from the observed data.

17.11 For More Information

- https://stefvanbuuren.name/fimd/

18 Multiple Regression

> ! This is a DRAFT version of this Chapter.
>
> This is a sketchy draft. I'll remove this notice when I post a version of this Chapter that is essentially finished.

18.1 R setup for this chapter

> i Note
>
> Appendix A lists all R packages used in this book, and also provides R session information. Appendix B describes the 431-Love.R script, and demonstrates its use.

```
library(broom)
library(car)
library(GGally)
library(knitr)
library(naniar)
library(rstanarm)
library(tidytuesdayR)

library(easystats)
library(tidyverse)

source("data/Love-431.R")
theme_set(theme_bw())
```

18.2 Child Care Costs from Tidy Tuesday

> **Note**
>
> Appendix C provides further guidance on pulling data from other systems into R, while Appendix D gives more information (including download links) for all data sets used in this book.

The data for this example is a subset of data described at the Tidy Tuesday repository for 2023-05-09, on Child Care Costs.

The data source is the National Database of Childcare Prices.

> The National Database of Childcare Prices (NDCP) is the most comprehensive federal source of childcare prices at the county level. The database offers childcare price data by childcare provider type, age of children, and county characteristics. Data are available from 2008 to 2018.

We will focus on the 2018 data at the county level for the US, and we will read in the data using the tidytuesdayR package.

The data we're interested in comes in two separate files, one called `childcare_costs`, and the other called `counties`.

```
tuesdata <- tidytuesdayR::tt_load("2023-05-09")
```

```
--- Compiling #TidyTuesday Information for 2023-05-09 ----

--- There are 2 files available ---

--- Starting Download ---


    Downloading file 1 of 2: `childcare_costs.csv`
    Downloading file 2 of 2: `counties.csv`

--- Download complete ---
```

```
childcare_costs <- tuesdata$childcare_costs
counties <- tuesdata$counties
```

```
head(childcare_costs)
```

```
# A tibble: 6 x 61
  county_fips_code study_year unr_16 funr_16 munr_16 unr_20to64 funr_20to64
             <dbl>      <dbl>  <dbl>   <dbl>   <dbl>      <dbl>       <dbl>
1             1001       2008   5.42    4.41    6.32        4.6         3.5
2             1001       2009   5.93    5.72    6.11        4.8         4.6
3             1001       2010   6.21    5.57    6.78        5.1         4.6
4             1001       2011   7.55    8.13    7.03        6.2         6.3
5             1001       2012   8.6     8.88    8.29        6.7         6.4
6             1001       2013   9.39   10.3     8.56        7.3         7.6
# i 54 more variables: munr_20to64 <dbl>, flfpr_20to64 <dbl>,
#   flfpr_20to64_under6 <dbl>, flfpr_20to64_6to17 <dbl>,
#   flfpr_20to64_under6_6to17 <dbl>, mlfpr_20to64 <dbl>, pr_f <dbl>,
#   pr_p <dbl>, mhi_2018 <dbl>, me_2018 <dbl>, fme_2018 <dbl>, mme_2018 <dbl>,
#   total_pop <dbl>, one_race <dbl>, one_race_w <dbl>, one_race_b <dbl>,
#   one_race_i <dbl>, one_race_a <dbl>, one_race_h <dbl>, one_race_other <dbl>,
#   two_races <dbl>, hispanic <dbl>, households <dbl>, ...
```

First, we work with the `childcare_costs` data to select the `study_year` of 2018, then collect the following variables (besides the `county_fips_code` we'll use to link to the `counties` data and the `study_year`):

- Our outcome will be `mfcc_preschool` (Aggregated weekly, full-time median price charged for Family Childcare for preschoolers (i.e. aged 36 through 54 months).
- Our first predictor is `mhi_2018` (Median household income expressed in 2018 dollars.) but recast as `mhi_18` (mhi_2018 / 1000) to express income in thousands of 2018 dollars,
- Our other predictor is `emp_service` (Percent of civilians employed in service occupations aged 16 years old and older in the county.)

In terms of dealing with missingness, we will drop all counties with missing values of the outcome, which is the only missing element .

```
costs <- childcare_costs |>
  filter(study_year == 2018) |>
  mutate(mhi_18 = mhi_2018/1000) |>
  select(county_fips_code, mfcc_preschool, mhi_18, emp_service) |>
  filter(complete.cases(mfcc_preschool))

prop_miss_case(costs)
```

```
[1] 0
```

Next, we use `left_join()` to merge the childcare costs and the county information by the `county_fips_code`.

```
care_costs <- left_join(costs, counties, by = "county_fips_code") |>
  rename(fips = county_fips_code,
         county = county_name,
         state = state_abbreviation) |>
  mutate(fips = as.character(fips))

care_costs
```

```
# A tibble: 2,348 x 7
   fips  mfcc_preschool mhi_18 emp_service county          state_name state
   <chr>          <dbl>  <dbl>       <dbl> <chr>           <chr>      <chr>
 1 1001           106.    58.8        15.9 Autauga County  Alabama    AL
 2 1003           109.    56.0        18.1 Baldwin County  Alabama    AL
 3 1005            77.1   34.2        14.6 Barbour County  Alabama    AL
 4 1007            87.1   45.3        18.7 Bibb County     Alabama    AL
 5 1009           112.    48.7        13   Blount County   Alabama    AL
 6 1011           106.    32.2        17.1 Bullock County  Alabama    AL
 7 1013           107.    39.1        16.0 Butler County   Alabama    AL
 8 1015            85.7   45.2        16.8 Calhoun County  Alabama    AL
 9 1017           116.    39.9        16.0 Chambers County Alabama    AL
10 1019            64.6   41.0        12.2 Cherokee County Alabama    AL
# i 2,338 more rows
```

If we like, we can then partition the data into separate training and testing samples. Here, we'll put 70% of the rows (counties) in our training sample, and the remaining 30% into the test sample.

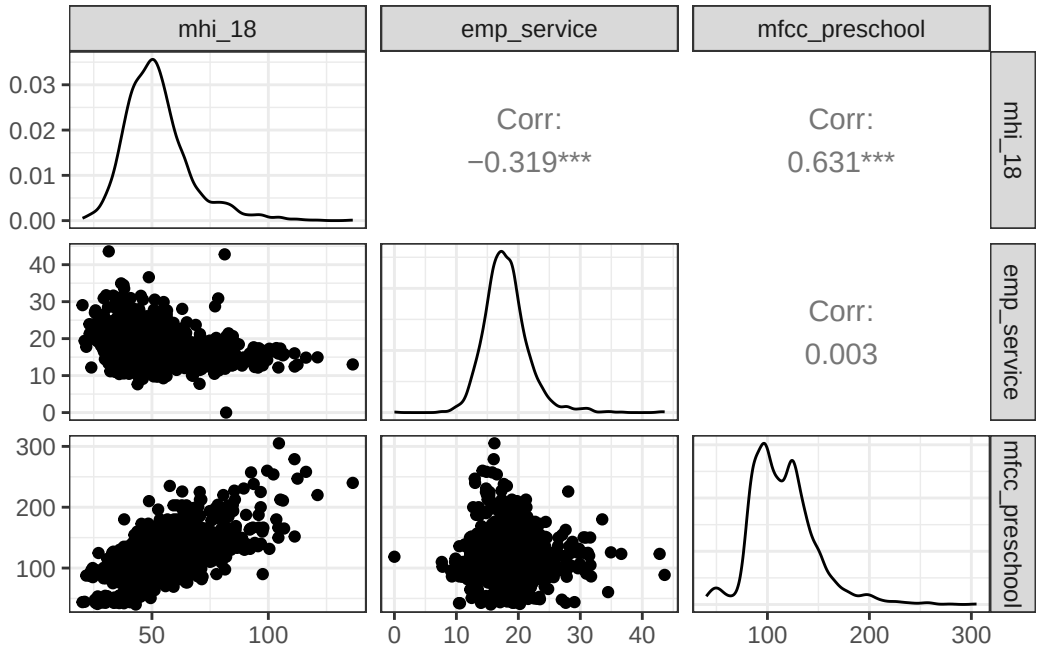
```
set.seed(431)
ccosts_train <- slice_sample(care_costs, prop = 0.7, replace = FALSE)
ccosts_test <- anti_join(care_costs, ccosts_train, by = "fips")

c(nrow(care_costs), nrow(ccosts_train), nrow(ccosts_test))
```

```
[1] 2348 1643  705
```

18.3 Fit model in training sample with `lm`

First, draw a picture with `ggpairs` from `GGally` of the training sample, so we can see the individual associations.

```
ggpairs(ccosts_train |>
          select(mhi_18, emp_service, mfcc_preschool))
```



Next, we'll fit the model, and obtain some basic summaries of the model's parameters and its performance.

```
fit1 <- lm(mfcc_preschool ~ mhi_18 + emp_service, data = ccosts_train)
```

```
model_parameters(fit1, ci = 0.95)
```

```
Parameter   | Coefficient |   SE |         95% CI | t(1640) |      p
-------------------------------------------------------------------
(Intercept) |       -3.63 | 4.42 | [-12.30, 5.03] |   -0.82 | 0.411
mhi 18      |        1.59 | 0.04 | [  1.51, 1.68] |   36.22 | < .001
emp service |        1.99 | 0.17 | [  1.66, 2.33] |   11.69 | < .001
```

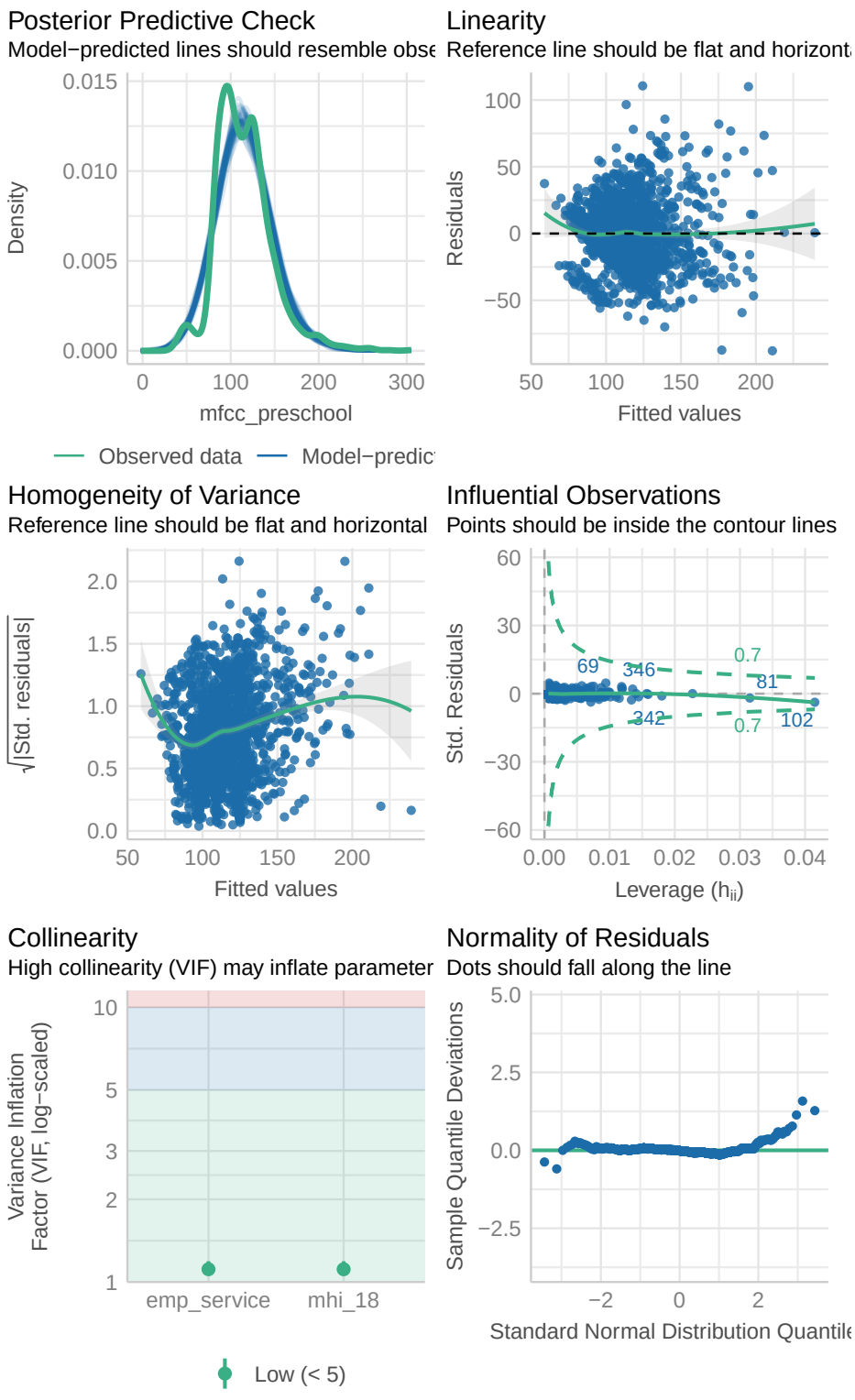
```
Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
  using a Wald t-distribution approximation.
```

```
model_performance(fit1)
```

```
# Indices of model performance

AIC       |      AICc |       BIC |    R2 | R2 (adj.) |   RMSE |  Sigma
-----------------------------------------------------------------------
15063.765 | 15063.789 | 15085.382 | 0.444 |     0.444 | 23.638 | 23.660
```

Next, we'll check a set of regression diagnostics to evaluate whether assumptions of linear regression appear to hold sufficiently well.

```
check_model(fit1)
```


Posterior Predictive Check
Model−predicted lines should resemble obse
Density
mfcc_preschool
Observed data
Model−predic
Linearity
Reference line should be flat and horizont
Residuals
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
Leverage (hii)
69
346
342
0.7
81
0.7
102
Collinearity
High collinearity (VIF) may inflate parameter
Variance Inflation Factor (VIF, log−scaled)
emp_service
mhi_18
Low (< 5)
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
Standard Normal Distribution Quantile

```
check_heteroscedasticity(fit1)
```

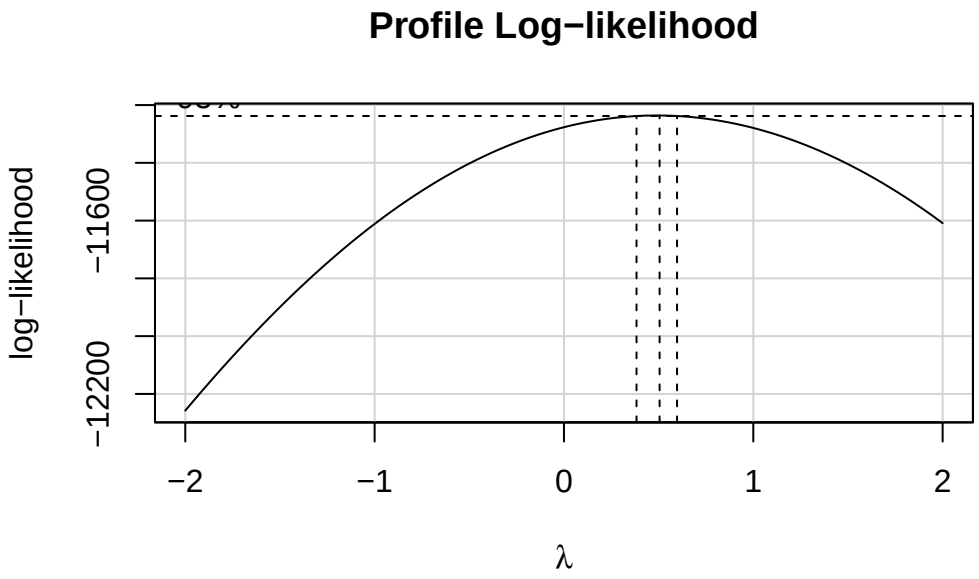
```
Warning: Heteroscedasticity (non-constant error variance) detected (p < .001).
```

```
check_normality(fit1)
```

```
Warning: Non-normality of residuals detected (p < .001).
```

18.4 Transformation needed?

```
fit1 <- lm(mfcc_preschool ~ mhi_18 + emp_service, data = ccosts_train)

boxCox(fit1)
```



18.5 Fitting an alternative model

Here, we'll also consider fitting a model with only one predictor (the `mhi_18`) variable, to see whether the addition of `emp_service` leads to a meaningful improvement. I don't expect this to be a better model than the model (fit1) that also includes `emp_service` in this case, but we'll use this comparison to demonstrate an appealing approach to using training and testing samples.

```
fit2 <- lm(mfcc_preschool ~ mhi_18, data = ccosts_train)
```

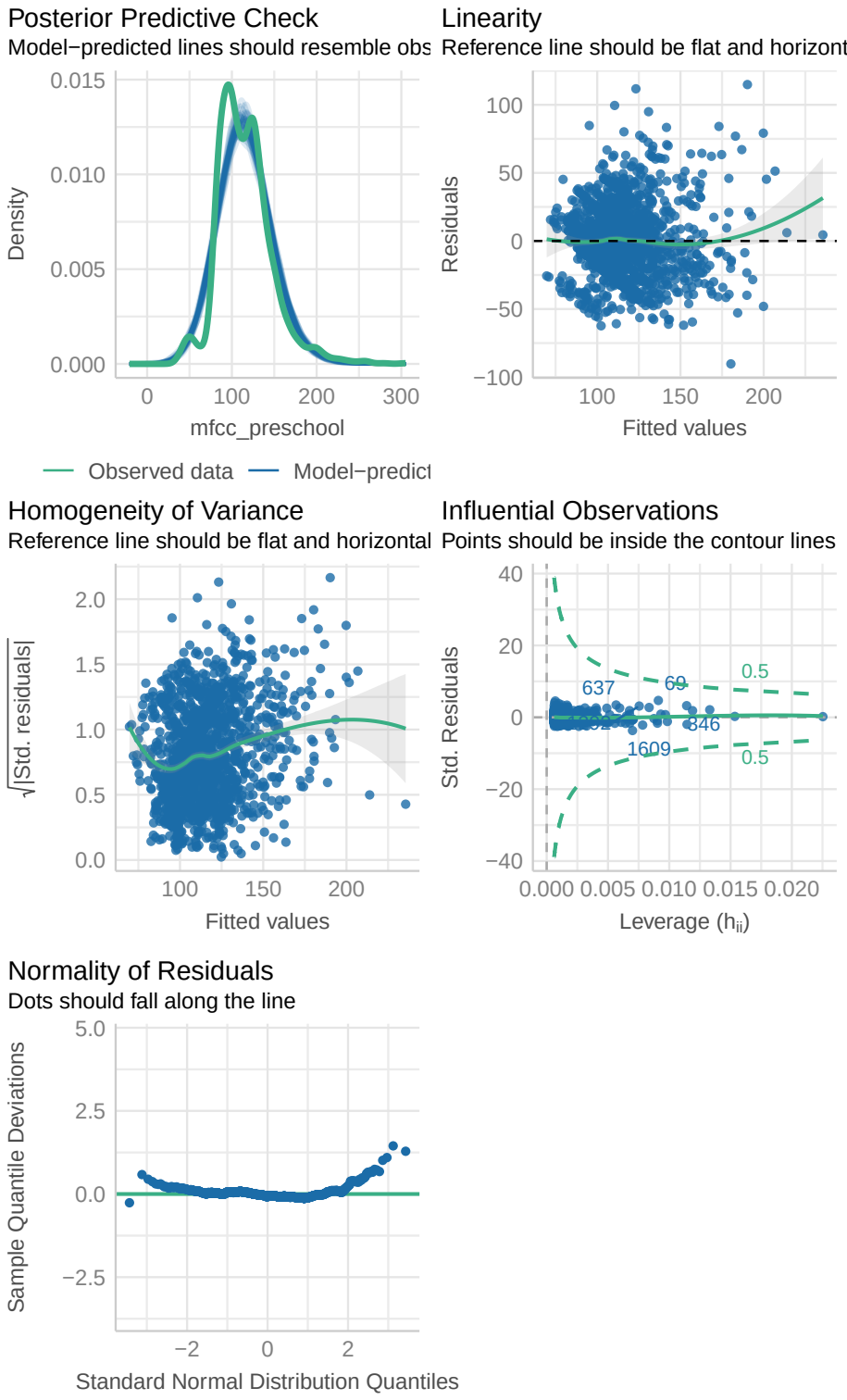
```
model_parameters(fit2, ci = 0.95)
```

```
Parameter   | Coefficient |   SE |         95% CI | t(1641) |      p
-------------------------------------------------------------------
(Intercept) |       40.81 | 2.35 | [36.21, 45.41] |   17.40 | < .001
mhi 18      |        1.43 | 0.04 | [ 1.34,  1.51] |   32.95 | < .001


Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
  using a Wald t-distribution approximation.
```

```
model_performance(fit2)
```

```
# Indices of model performance

AIC       |      AICc |       BIC |    R2 | R2 (adj.) |   RMSE |  Sigma
-------------------------------------------------------------------------
15193.355 | 15193.370 | 15209.568 | 0.398 |     0.398 | 24.604 | 24.619
```

```
check_model(fit2)
```


Posterior Predictive Check
Model−predicted lines should resemble obs
Density
mfcc_preschool
Observed data
Model−predict
Linearity
Reference line should be flat and horizonta
Residuals
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
Leverage (hii)
637
69
846
1609
0.5
0.5
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
Standard Normal Distribution Quantiles

```
check_heteroscedasticity(fit2)
```

```
Warning: Heteroscedasticity (non-constant error variance) detected (p < .001).
```

```
check_normality(fit2)
```

```
Warning: Non-normality of residuals detected (p < .001).
```

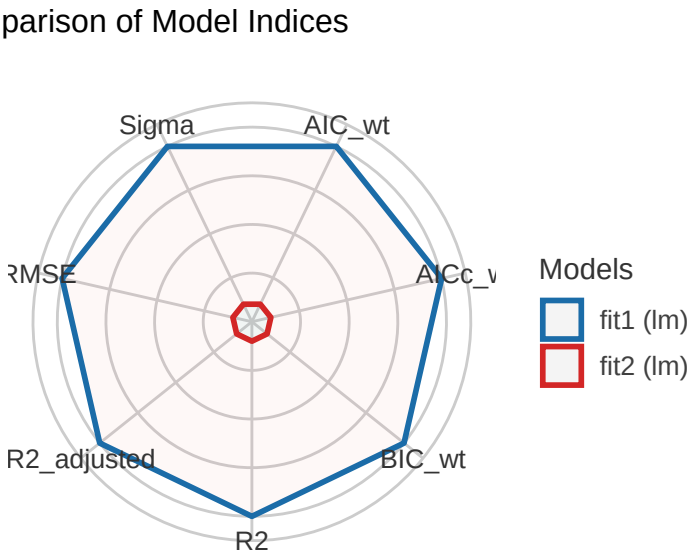
18.6 Comparing the models (training sample)

We can compare the models within our training sample by comparing the performance statistics, as well as the quality of the diagnostic checks.

```
compare_performance(fit1, fit2)
```

```
# Comparison of Model Performance Indices

Name | Model |   AIC (weights) |  AICc (weights) |   BIC (weights) |    R2 | R2 (adj.) |   RM
-------------------------------------------------------------------------------------------
fit1 |    lm | 15063.8 (>.999) | 15063.8 (>.999) | 15085.4 (>.999) | 0.444 |     0.444 | 23.6
fit2 |    lm | 15193.4 (<.001) | 15193.4 (<.001) | 15209.6 (<.001) | 0.398 |     0.398 | 24.6
```

```
plot(compare_performance(fit1, fit2))
```



For this "spiderweb" plot, the different indices are normalized and larger values indicate better model performance. So `fit2`, with points closer to the center displays worse fit indices than `fit1` within our training sample. Of course, this isn't a surprise to us. For more on this, see the documentation online.

18.7 Comparing linear models (test sample)

We'll use the `augment()` function from the `broom` package in this effort.

```
test1 <- augment(fit1, newdata = ccosts_test) |>
  mutate(mod_n = "Model 1")
test2 <- augment(fit2, newdata = ccosts_test) |>
  mutate(mod_n = "Model 2")

test_res <- bind_rows(test1, test2) |>
  select(mod_n, fips, mfcc_preschool, .fitted, .resid,
         everything()) |>
  arrange(fips, mod_n)

test_res |> head()
```

```
# A tibble: 6 x 10
```

```
  mod_n fips  mfcc_preschool .fitted .resid mhi_18 emp_service county state_name
  <chr> <chr>          <dbl>   <dbl>  <dbl>  <dbl>       <dbl> <chr>  <chr>
1 Mode~ 10003           155.    142.  12.7    71.0        16.5 New C~ Delaware
2 Mode~ 10003           155.    142.  12.8    71.0        16.5 New C~ Delaware
3 Mode~ 10005           122.    130.  -8.38   60.9        18.4 Susse~ Delaware
4 Mode~ 10005           122.    128.  -6.11   60.9        18.4 Susse~ Delaware
5 Mode~ 1003            109.    122. -13.1    56.0        18.1 Baldw~ Alabama
6 Mode~ 1003            109.    121. -12.3    56.0        18.1 Baldw~ Alabama
# i 1 more variable: state <chr>
```

Now, we can summarize the quality of the predictions across these two models with four summary statistics calculated across the observations in the test sample.

- the mean absolute prediction error, or MAPE
- the median absolute prediction error, or medAPE
- the maximum absolute prediction error, or maxAPE
- the square root of the mean squared prediction error, or RMSPE

No one of these dominates the other, but we might be interested in which model gives us the best (smallest) result for each of these summaries. Let's run them.

```
test_res |>
    group_by(mod_n) |>
    summarise(MAPE = mean(abs(.resid)),
              medAPE = median(abs(.resid)),
              maxAPE = max(abs(.resid)),
              RMSPE = sqrt(mean(.resid^2)),
              rsqr = cor(mfcc_preschool, .fitted)^2) |>
  kable()
```

mod_n	MAPE	medAPE	maxAPE	RMSPE	rsqr
Model 1	17.34671	13.78572	151.2994	23.93467	0.5388358
Model 2	18.16049	13.86123	170.7277	25.37279	0.4738406

Model 1 (including both predictors) performs better within the test sample on each of these four summaries. It has the smaller mean, median and maximum absolute prediction error, and the smaller root mean squared prediction error.

18.8 Bayesian linear model

Here, we'll demonstrate a Bayesian fit including each of our predictors, with a weakly informative prior, and using the training data.

```
fit3 <- stan_glm(mfcc_preschool ~ mhi_18 + emp_service,
                 data = ccosts_train, refresh = 0)
```

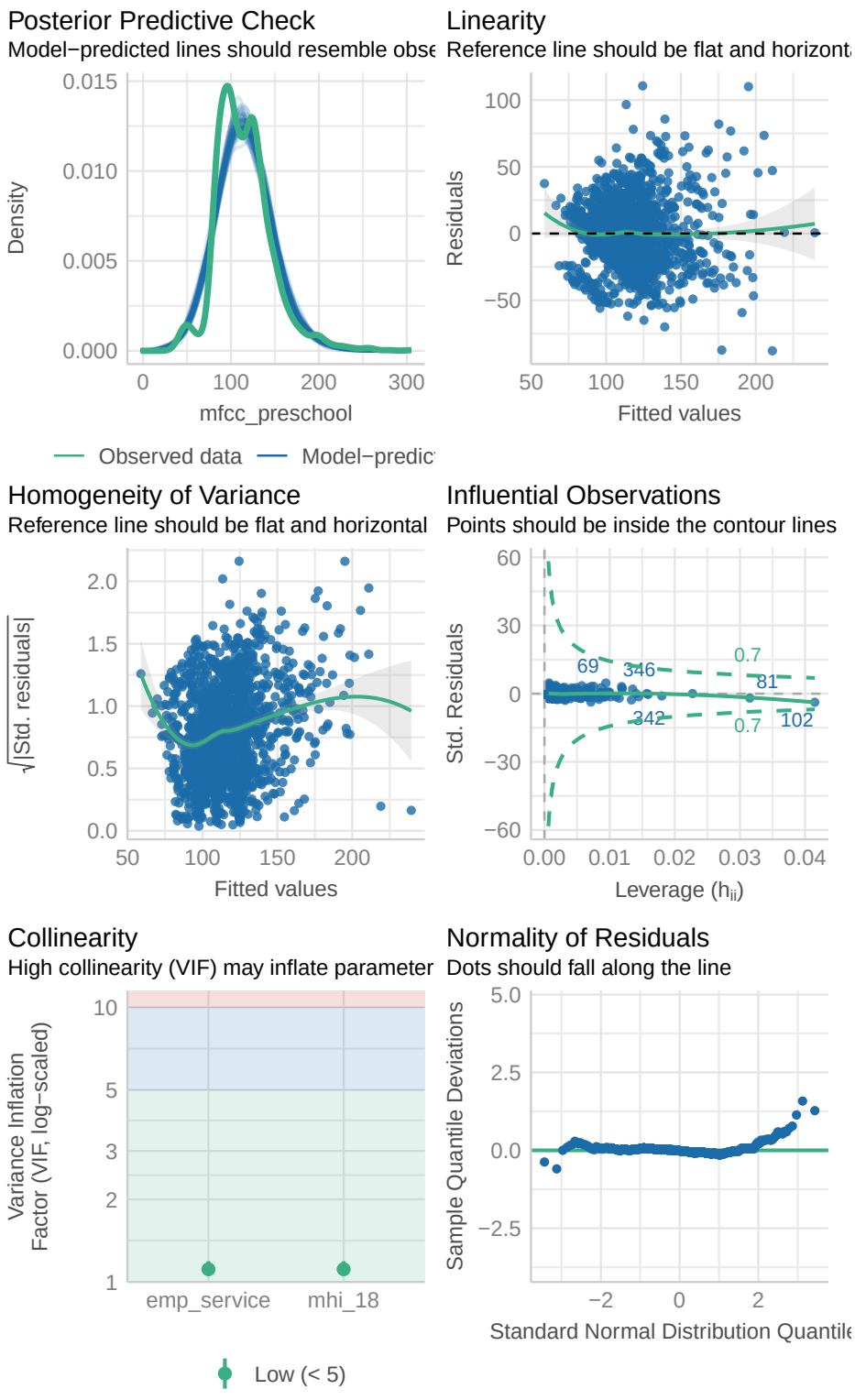
```
model_parameters(fit3, ci = 0.95)
```

```
Parameter   | Median |         95% CI |     pd |  Rhat |     ESS |                    Prior
-------------------------------------------------------------------------------------------
(Intercept) |  -3.55 | [-12.14, 4.66] | 79.15% | 0.999 | 4179.00 | Normal (115.45 +- 79.31)
mhi_18      |   1.59 | [  1.51, 1.68] |   100% | 0.999 | 4659.00 |    Normal (0.00 +- 5.66)
emp_service |   1.99 | [  1.65, 2.32] |   100% | 0.999 | 4568.00 |   Normal (0.00 +- 21.93)


Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
  using a MCMC distribution approximation.
```

```
model_performance(fit3)
```

```
# Indices of model performance

ELPD      | ELPD_SE |     LOOIC | LOOIC_SE |      WAIC |    R2 | R2 (adj.) |   RMSE |  Sigma
-----------------------------------------------------------------------------------------------
-7532.963 |  35.580 | 15065.927 |   71.160 | 15065.919 | 0.444 |     0.441 | 23.638 | 23.670
```

```
check_model(fit3)
```


Posterior Predictive Check
Model−predicted lines should resemble obse
Density
mfcc_preschool
Observed data
Model−predic
Linearity
Reference line should be flat and horizont
Residuals
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
Leverage (hii)
Collinearity
High collinearity (VIF) may inflate parameter
Variance Inflation Factor (VIF, log−scaled)
emp_service
mhi_18
Low (< 5)
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
Standard Normal Distribution Quantile

18.9 For More Information

431-book Chapter 18 reference OpenStats https://www.openintro.org/book/os/Section 9 - multiple regression (except we don't do 9.5 in 431)

19 Two Factor ANOVA

! This is a DRAFT version of this Chapter.

This is a sketchy draft. I'll remove this notice when I post a version of this Chapter that is essentially finished.

19.1 R setup for this chapter

i Note

Appendix A lists all R packages used in this book, and also provides R session information. Appendix B describes the 431-Love.R script, and demonstrates its use.

```
library(knitr)
library(naniar)
library(rstanarm)

library(easystats)
library(tidyverse)

source("data/Love-431.R")
theme_set(theme_bw())
```

19.2 Data are Rosner's FEV Data

i Note

Appendix C provides further guidance on pulling data from other systems into R, while Appendix D gives more information (including download links) for all data sets used in this book.

Rosner's FEV data in https://hbiostat.org/data/ FEV (quant) as a function of age, height, sex, smoker should be useful for a two-way anova perhaps adjusting for two additional continuous covariates

```
fev_ros <- read_csv("data/fev_ros.csv", show_col_types = FALSE) |>
  mutate(across(where(is.character), as_factor)) |>
  mutate(id = as.character(id))

fev_ros
```

```
# A tibble: 654 x 6
   id      age   fev height sex    smoke
   <chr> <dbl> <dbl>  <dbl> <fct>  <fct>
 1 301       9  1.71   57   female non-current smoker
 2 451       8  1.72   67.5 female non-current smoker
 3 501       7  1.72   54.5 female non-current smoker
 4 642       9  1.56   53   male   non-current smoker
 5 901       9  1.90   57   male   non-current smoker
 6 1701      8  2.34   61   female non-current smoker
 7 1752      6  1.92   58   female non-current smoker
 8 1753      6  1.42   56   female non-current smoker
 9 1901      8  1.99   58.5 female non-current smoker
10 1951      9  1.94   60   female non-current smoker
# i 644 more rows
```

```
n_miss(fev_ros)
```

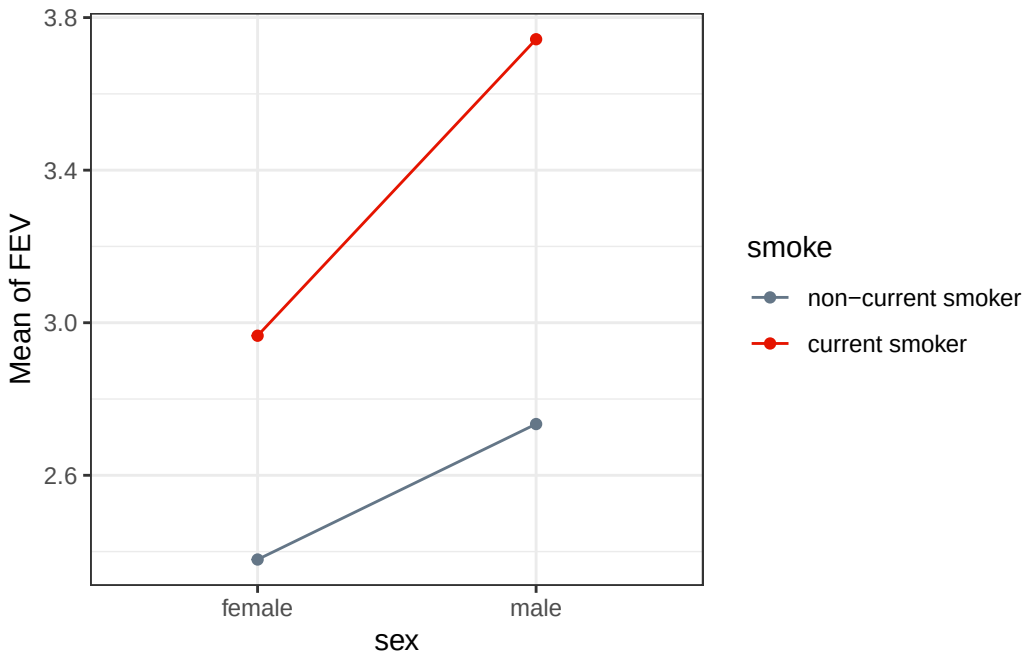
```
[1] 0
```

19.3 Interaction Plot

19.3.1 Simple Plot of Means

```
fev_ros |> group_by(sex, smoke) |>
  summarise(mean_fev = mean(fev)) |>
  ggplot(aes(x = sex, y = mean_fev, group = smoke, col = smoke)) +
  geom_point() +
  geom_line() +
  scale_color_metro_d() +
  labs(x = "sex", y = "Mean of FEV")
```

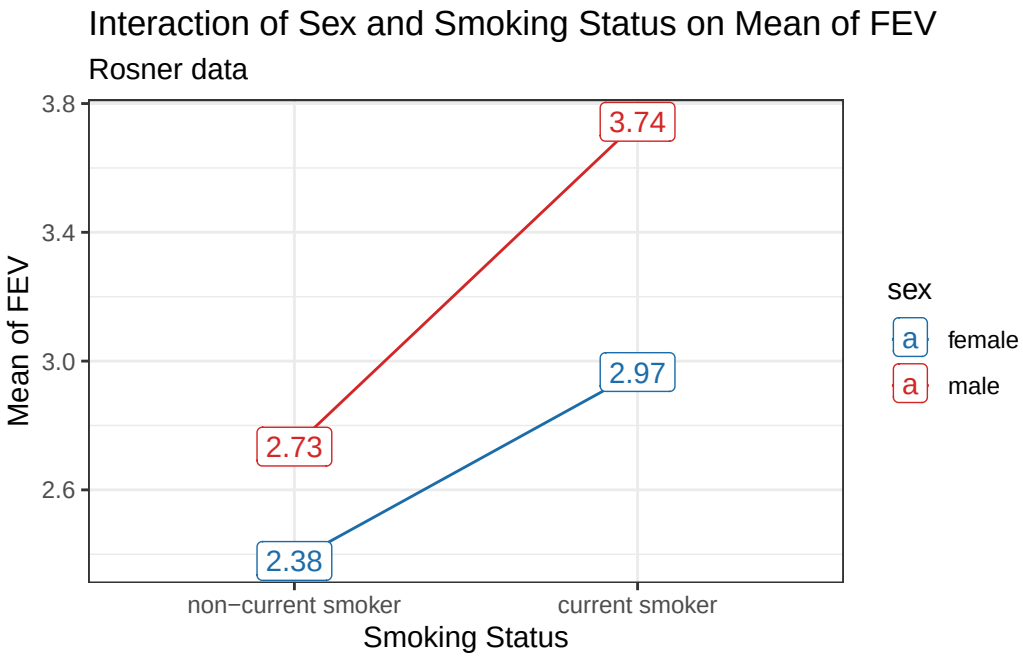
`summarise()` has grouped output by 'sex'. You can override using the `.groups` argument.



19.3.2 Labeled Interaction Plot

```
fev_ros |> group_by(sex, smoke) |>
  summarise(mean_fev = mean(fev)) |>
  ggplot(aes(x = smoke, y = mean_fev, group = sex, col = sex)) +
  geom_line() +
  geom_label(aes(label = round(mean_fev,2))) +
  scale_color_see_d() +
  labs(x = "Smoking Status", y = "Mean of FEV",
       title = "Interaction of Sex and Smoking Status on Mean of FEV",
       subtitle = "Rosner data")
```

`summarise()` has grouped output by 'sex'. You can override using the `.groups` argument.



19.4 ANOVA models

```
fit1 <- aov(fev ~ sex * smoke, data = fev_ros)

model_parameters(fit1, es_type = "eta", ci = 0.95)
```

```
Parameter | Sum_Squares |  df | Mean_Square |     F |      p | Eta2 (partial) |  Eta2 95% CI
-----------------------------------------------------------------------------------------------
sex       |       21.32 |   1 |       21.32 | 31.98 | < .001 |           0.05 | [0.02, 1.00]
smoke     |       33.68 |   1 |       33.68 | 50.51 | < .001 |           0.07 | [0.04, 1.00]
sex:smoke |        2.51 |   1 |        2.51 |  3.77 | 0.053  |       5.76e-03 | [0.00, 1.00]
Residuals |      433.40 | 650 |        0.67 |       |        |                |

Anova Table (Type 1 tests)
```

```
fit2 <- lm(fev ~ sex * smoke, data = fev_ros)

model_parameters(fit2, ci = 0.95)
```

```
Parameter                           | Coefficient |   SE |        95% CI | t(650) |      p
-------------------------------------------------------------------------------------------
(Intercept)                         |        2.38 | 0.05 | [ 2.28, 2.48] |  48.67 | < .001
sex [male]                          |        0.36 | 0.07 | [ 0.22, 0.49] |   5.27 | < .001
smoke [current smoker]              |        0.59 | 0.14 | [ 0.31, 0.86] |   4.20 | < .001
sex [male] × smoke [current smoker] |        0.42 | 0.22 | [ 0.00, 0.85] |   1.94 | 0.053


Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
  using a Wald t-distribution approximation.
```

19.4.1 Bootstrapping CIs for estimates

```
set.seed(431)
model_parameters(fit2,
           ci = 0.95, ci_method = "quantile",
           bootstrap = TRUE, iteractions = 1000)
```

```
Parameter                           | Coefficient |        95% CI |      p
-------------------------------------------------------------------------
(Intercept)                         |        2.38 | [ 2.31, 2.45] | < .001
sex [male]                          |        0.35 | [ 0.22, 0.49] | < .001
smoke [current smoker]              |        0.58 | [ 0.44, 0.73] | < .001
sex [male] × smoke [current smoker] |        0.43 | [-0.01, 0.81] | 0.052


Uncertainty intervals (equal-tailed) are naıve bootstrap intervals.
```

19.5 ANOVA without interaction

```
fit3 <- aov(fev ~ sex + smoke, data = fev_ros)

model_parameters(fit3, es_type = "eta", ci = 0.95)
```

```
Parameter | Sum_Squares |  df | Mean_Square |     F |      p | Eta2 (partial) |  Eta2 95% CI
-----------------------------------------------------------------------------------------------
sex       |       21.32 |   1 |       21.32 | 31.85 | < .001 |           0.05 | [0.02, 1.00]
```

```
smoke     |       33.68 |   1 |       33.68 | 50.30 | < .001 |           0.07 | [0.04, 1.00]
Residuals |      435.91 | 651 |        0.67 |       |        |                |
```

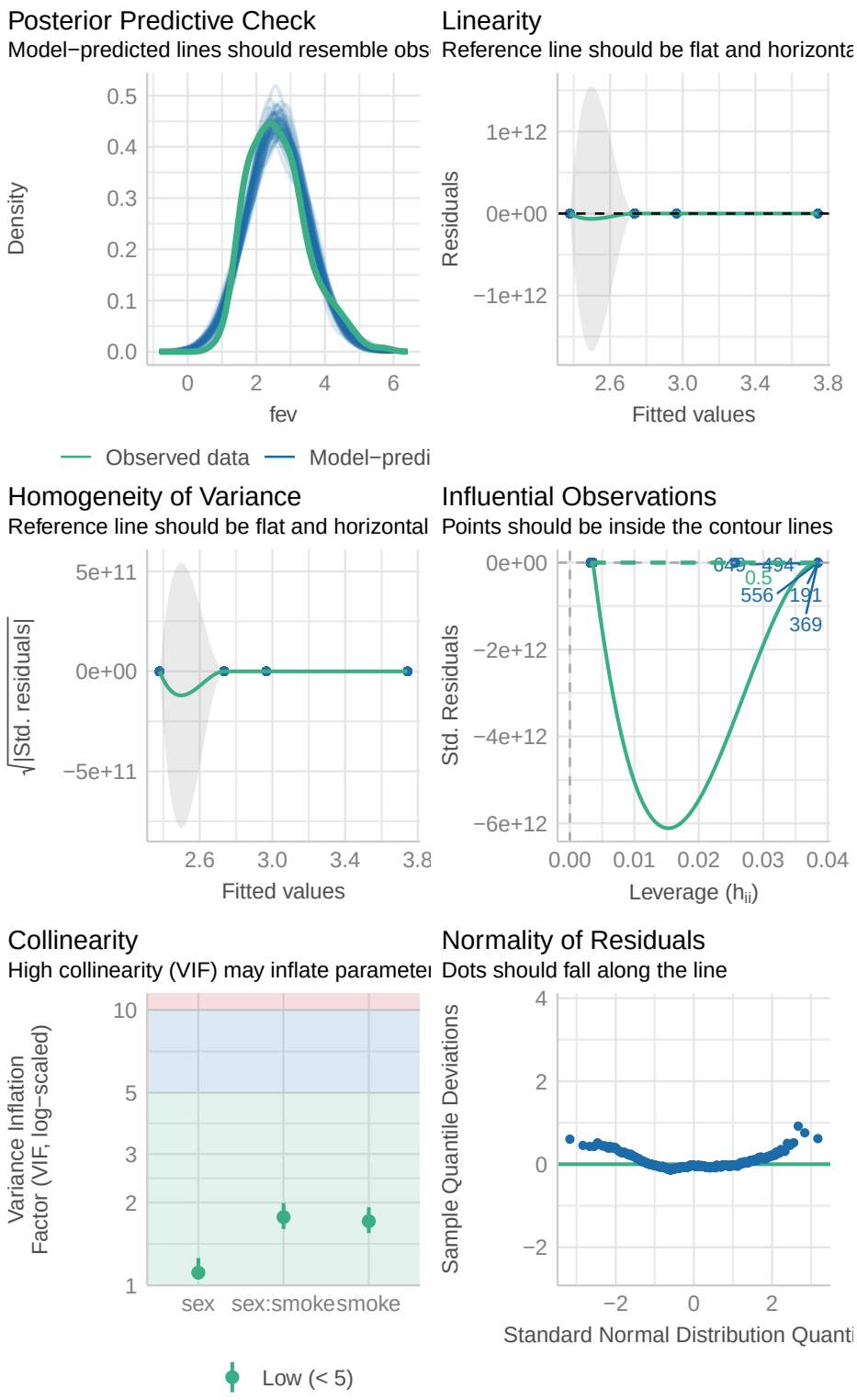
Anova Table (Type 1 tests)

```
anova(fit3, fit1)
```

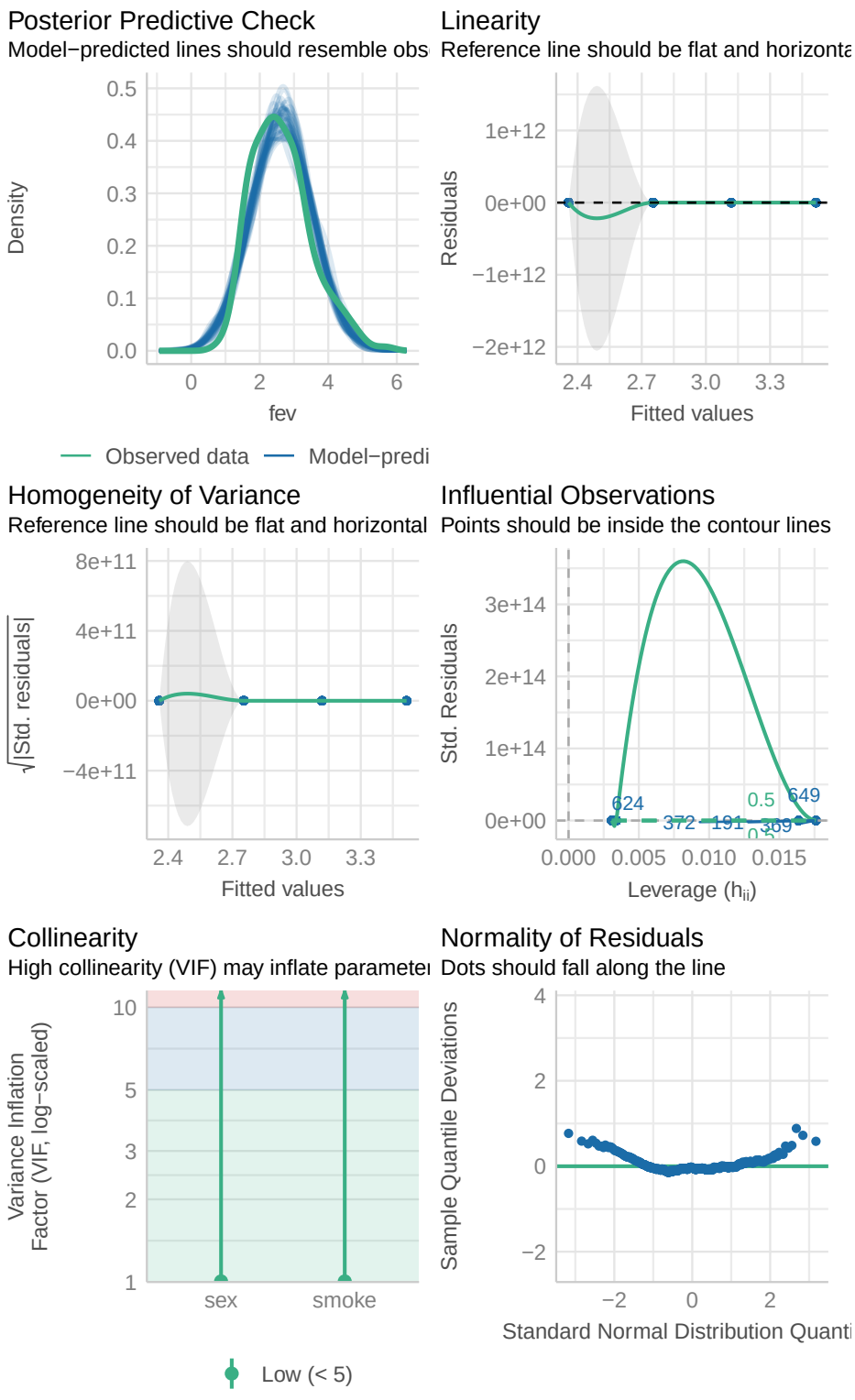
```
Analysis of Variance Table

Model 1: fev ~ sex + smoke
Model 2: fev ~ sex * smoke
  Res.Df    RSS Df Sum of Sq      F  Pr(>F)
1    651 435.91
2    650 433.40  1    2.5127 3.7684 0.05266 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

19.6 Considering Model Diagnostics

```
check_model(fit1)
```


Posterior Predictive Check
Model−predicted lines should resemble obs
Density
fev
Observed data
Model−predi
Linearity
Reference line should be flat and horizonta
Residuals
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
Leverage (hii)
Collinearity
High collinearity (VIF) may inflate parameter
Variance Inflation Factor (VIF, log−scaled)
sex
sex:smoke
smoke
Low (< 5)
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
Standard Normal Distribution Quanti

```
check_model(fit3)
```

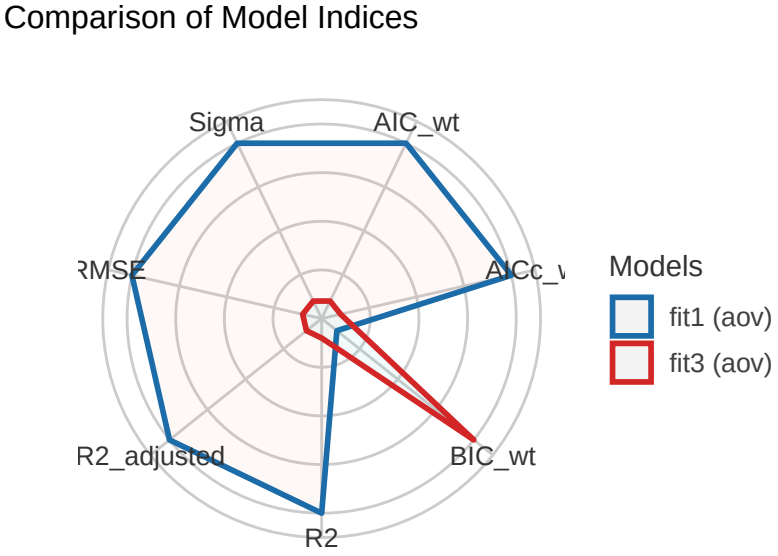

Posterior Predictive Check
Model−predicted lines should resemble obs
Density
fev
Observed data
Model−predi
Linearity
Reference line should be flat and horizonta
Residuals
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
Leverage (hii)
Collinearity
High collinearity (VIF) may inflate parameter
Variance Inflation Factor (VIF, log−scaled)
sex
smoke
Low (< 5)
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
Standard Normal Distribution Quanti

19.7 Comparing Model Performance

```
compare_performance(fit1, fit3)
```

```
# Comparison of Model Performance Indices

Name | Model |  AIC (weights) | AICc (weights) |  BIC (weights) |    R2 | R2 (adj.) |  RMSE
-----------------------------------------------------------------------------------------------
fit1 |   aov | 1596.9 (0.709) | 1597.0 (0.706) | 1619.3 (0.206) | 0.117 |     0.113 | 0.814
fit3 |   aov | 1598.7 (0.291) | 1598.7 (0.294) | 1616.6 (0.794) | 0.112 |     0.109 | 0.816
```

```
plot(compare_performance(fit1, fit3))
```



19.8 For More Information

20 Multiple Regression with many predictors

> **!** This is a DRAFT version of this Chapter.
>
> This is a sketchy draft. I'll remove this notice when I post a version of this Chapter that is essentially finished.

20.1 R setup for this chapter

> **i** Note
>
> Appendix A lists all R packages used in this book, and also provides R session information.

```
library(car)
library(janitor)
library(knitr)
library(naniar)
library(olsrr)
library(rstanarm)

library(easystats)
library(tidyverse)

theme_set(theme_bw())
```

Multiple regression y predicted by 3-15 variables

20.2 Data is bodyfat

> **Note**
>
> Appendix C provides further guidance on pulling data from other systems into R, while Appendix D gives more information (including download links) for all data sets used in this book.

My source for these data was Kaggle, although the data have been published in several other places. The data describe 252 men whose percentage of body fat was determined via underwater weighing and Siri's (1956) equation.

From Kaggle:

> These data are used to produce the predictive equations for lean body weight given in the abstract "Generalized body composition prediction equation for men using simple measurement techniques", K.W. Penrose, A.G. Nelson, A.G. Fisher, FACSM, Human Performance Research Center, Brigham Young University, Provo, Utah 84602 as listed in Medicine and Science in Sports and Exercise, vol. 17, no. 2, April 1985, p. 189.

```
bodyfat <- read_csv("data/bodyfat.csv", show_col_types = FALSE) |>
  janitor::clean_names()

bodyfat
```

```
# A tibble: 248 x 16
   kag_row density siri_bf   age weight height  neck chest abdomen   hip thigh
   <chr>     <dbl>   <dbl> <dbl>  <dbl>  <dbl> <dbl> <dbl>   <dbl> <dbl> <dbl>
 1 sub_001    1.07    12.3    23   154.   67.8  36.2  93.1    85.2  94.5  59
 2 sub_002    1.09     6.1    22   173.   72.2  38.5  93.6    83    98.7  58.7
 3 sub_003    1.04    25.3    22   154    66.2  34    95.8    87.9  99.2  59.6
 4 sub_004    1.08    10.4    26   185.   72.2  37.4 102.     86.4 101.   60.1
 5 sub_005    1.03    28.7    24   184.   71.2  34.4  97.3   100   102.   63.2
 6 sub_006    1.05    21.3    24   210.   74.8  39   104.     94.4 108.   66
 7 sub_007    1.05    19.2    26   181    69.8  36.4 105.     90.7 100.   58.4
 8 sub_008    1.07    12.4    25   176    72.5  37.8  99.6    88.5  97.1  60
 9 sub_009    1.09     4.1    25   191    74    38.1 101.     82.5  99.9  62.9
10 sub_010    1.07    11.7    23   198.   73.5  42.1  99.6    88.6 104.   63.1
# i 238 more rows
# i 5 more variables: knee <dbl>, ankle <dbl>, biceps <dbl>, forearm <dbl>,
#   wrist <dbl>
```

The variables included are:

Variable	Description[1]
kag_row	subject identification code
density	density determined from underwater weighing
siri_bf	percent body fat from Siri's (1956) equation[2]
age	age (years)
weight	weight (pounds)
height	height (inches)
neck	neck circumference (cm)
chest	chest circumference (cm)
abdomen	abdomen circumference (cm)
hip	hip circumference (cm)
thigh	thigh circumference (cm)
knee	knee circumference (cm)
ankle	ankle circumference (cm)
biceps	biceps (extended) circumference (cm)
forearm	forearm circumference (cm)
wrist	wrist circumference (cm)

As Kaggle motivates,

> Accurate measurement of body fat is inconvenient/costly and it is desirable to have easy methods of estimating body fat that are not inconvenient/costly.

So our job will be to predict the `siri_bf` measure using the 13 predictors listed above from `age` through `wrist`, or perhaps a well-chosen subset of those predictors.

Note that we have no missing data here to worry about.

```
n_miss(bodyfat)
```

```
[1] 0
```

Since our main goal is making accurate predictions, it's appealing to validate our models in new data. We might split into training and testing samples

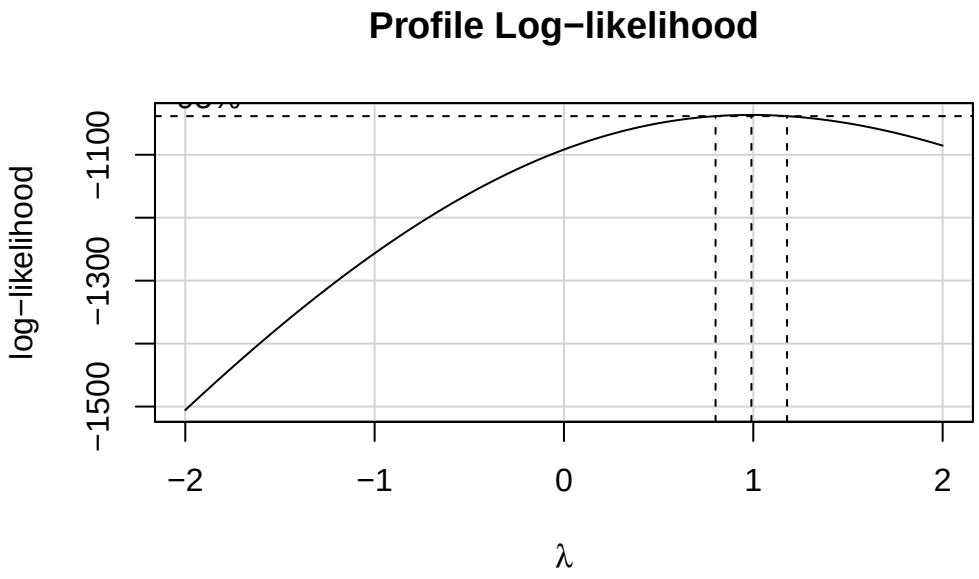
[2] Siri's equation is BF = 495/density - 450

[1] (Measurement standards are apparently those listed in Benhke and Wilmore (1974), pp. 45-48 where, for instance, the abdomen circumference is measured "laterally, at the level of the iliac crests, and anteriorly, at the umbilicus".)

20.3 Need for transformation?

We'll start by considering whether the Box-Cox approach suggests any particular transformation for our outcome variable when we fit a kitchen sink model (e.g., a model including main effects of all available predictors.)

```
fit1 <- lm(
  siri_bf ~ age + weight + height + neck + chest +
    abdomen + hip + thigh + knee + ankle + biceps +
    forearm + wrist,
  data = bodyfat
)

boxCox(fit1)
```



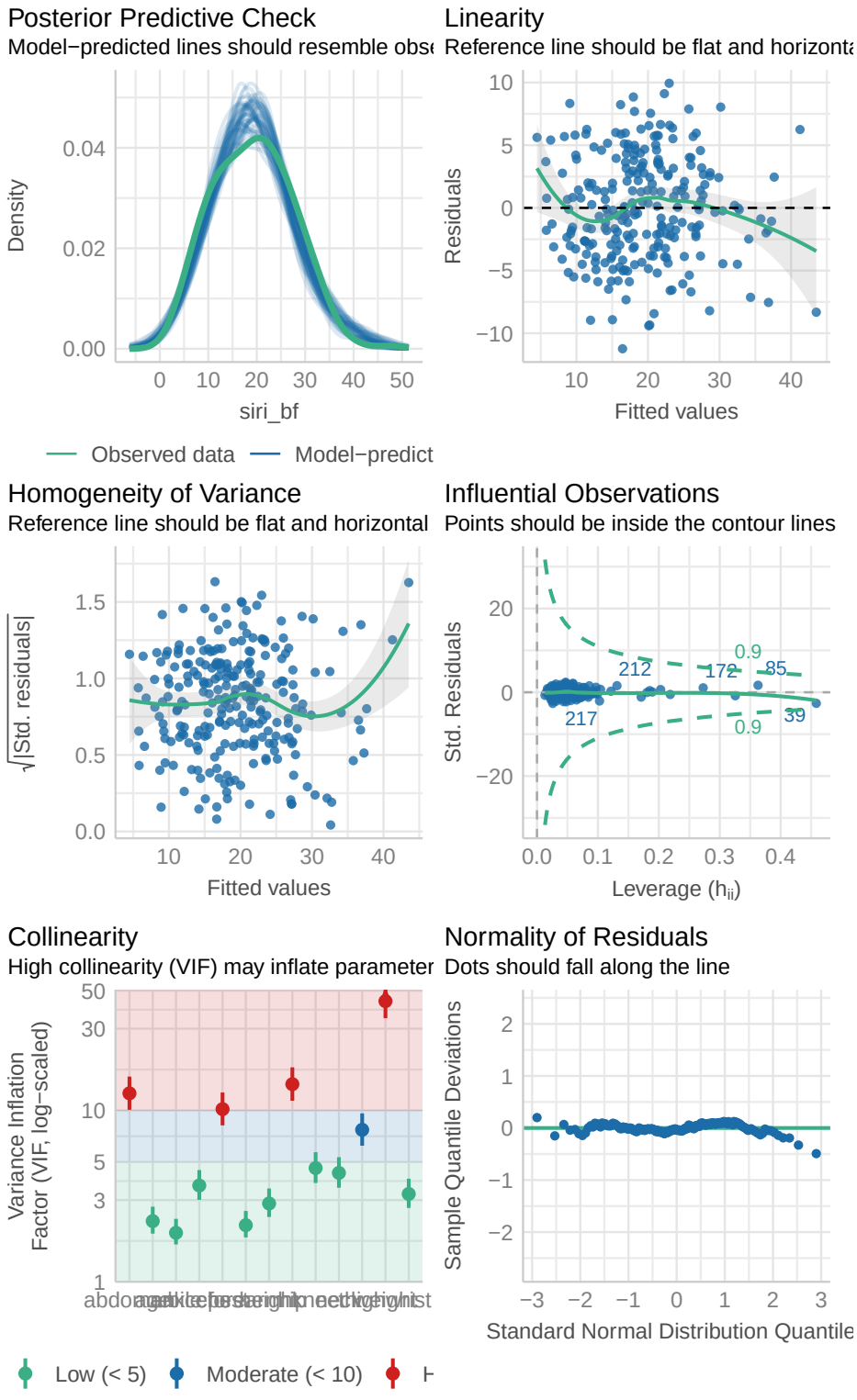
Since the recommended power transformation parameter λ is quite close to 1, we'll not be using any transformation in this chapter.

20.4 The Full OLS model

```
model_parameters(fit1, ci = 0.95)
```

```
Parameter   | Coefficient |    SE |          95% CI | t(234) |      p
-------------------------------------------------------------------
(Intercept) |      -14.73 | 22.35 | [-58.77, 29.30] |  -0.66 | 0.510
age         |        0.05 |  0.03 | [ -0.01,  0.11] |   1.53 | 0.128
weight      |       -0.09 |  0.06 | [ -0.21,  0.04] |  -1.40 | 0.162
height      |       -0.08 |  0.18 | [ -0.44,  0.27] |  -0.46 | 0.644
neck        |       -0.48 |  0.23 | [ -0.94, -0.02] |  -2.06 | 0.040
chest       |       -0.03 |  0.10 | [ -0.23,  0.18] |  -0.27 | 0.786
abdomen     |        0.97 |  0.09 | [  0.79,  1.14] |  10.69 | < .001
hip         |       -0.21 |  0.15 | [ -0.50,  0.08] |  -1.45 | 0.148
thigh       |        0.20 |  0.15 | [ -0.09,  0.49] |   1.36 | 0.176
knee        |        0.04 |  0.25 | [ -0.45,  0.52] |   0.14 | 0.886
ankle       |        0.17 |  0.22 | [ -0.27,  0.61] |   0.78 | 0.436
biceps      |        0.16 |  0.17 | [ -0.18,  0.50] |   0.93 | 0.354
forearm     |        0.44 |  0.20 | [  0.05,  0.83] |   2.21 | 0.028
wrist       |       -1.59 |  0.53 | [ -2.64, -0.54] |  -2.98 | 0.003


Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
  using a Wald t-distribution approximation.
```

```
check_model(fit1)
```


Posterior Predictive Check
Model−predicted lines should resemble obs
Density
siri_bf
Observed data
Model−predict
Linearity
Reference line should be flat and horizonta
Residuals
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
Leverage (hii)
Collinearity
High collinearity (VIF) may inflate parameter
Variance Inflation Factor (VIF, log−scaled)
Low (< 5)
Moderate (< 10)
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
Standard Normal Distribution Quantile

```
model_performance(fit1)
```

```
# Indices of model performance

AIC      |     AICc |      BIC |    R2 | R2 (adj.) |  RMSE | Sigma
-----------------------------------------------------------------
1440.004 | 1442.073 | 1492.706 | 0.742 |     0.728 | 4.153 | 4.275
```

20.5 Identify a predictor subset with backwards stepwise elimination

```
fit2 <- select_parameters(fit1)

model_parameters(fit2, ci = 0.95)
```

```
Parameter   | Coefficient |    SE |          95% CI | t(239) |      p
-------------------------------------------------------------------
(Intercept) |      -19.90 | 11.68 | [-42.91,  3.11] |  -1.70 | 0.090
age         |        0.05 |  0.03 | [ -0.01,  0.11] |   1.67 | 0.096
weight      |       -0.09 |  0.04 | [ -0.17, -0.01] |  -2.21 | 0.028
neck        |       -0.47 |  0.22 | [ -0.91, -0.03] |  -2.12 | 0.035
abdomen     |        0.96 |  0.07 | [  0.82,  1.10] |  13.32 | < .001
hip         |       -0.21 |  0.14 | [ -0.48,  0.07] |  -1.49 | 0.136
thigh       |        0.26 |  0.13 | [  0.00,  0.51] |   1.99 | 0.048
forearm     |        0.49 |  0.19 | [  0.13,  0.86] |   2.65 | 0.008
wrist       |       -1.46 |  0.51 | [ -2.47, -0.46] |  -2.88 | 0.004


Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
  using a Wald t-distribution approximation.
```

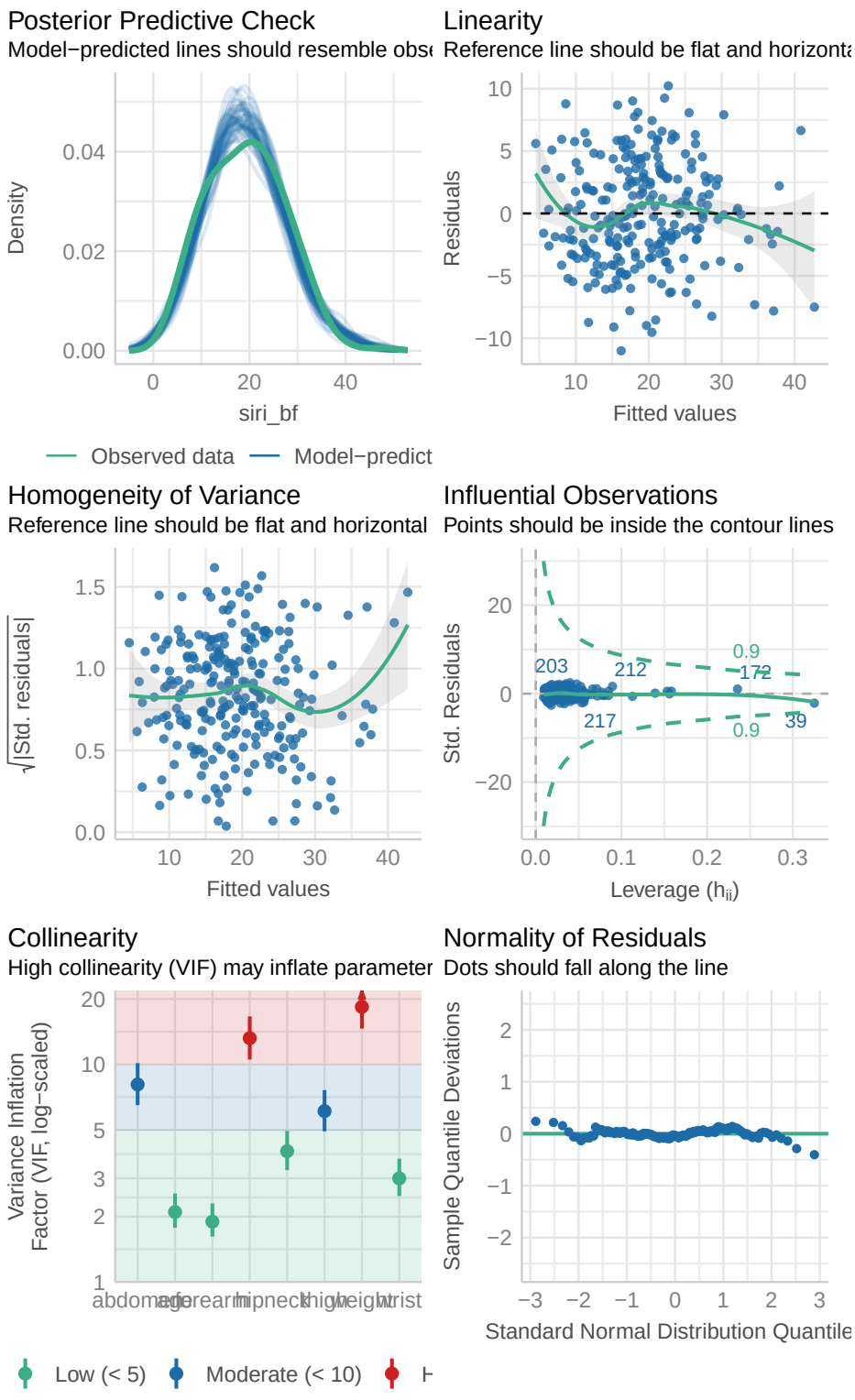
```
compare_models(fit1, fit2)
```

```
Parameter   |                  fit1 |                  fit2
-------------------------------------------------------------
(Intercept) | -14.73 (-58.77, 29.30) | -19.90 (-42.91,  3.11)
age         |   0.05 ( -0.01,  0.11) |   0.05 ( -0.01,  0.11)
```

```
weight       |  -0.09 ( -0.21,  0.04) |  -0.09 ( -0.17, -0.01)
neck         |  -0.48 ( -0.94, -0.02) |  -0.47 ( -0.91, -0.03)
abdomen      |   0.97 (  0.79,  1.14) |   0.96 (  0.82,  1.10)
hip          |  -0.21 ( -0.50,  0.08) |  -0.21 ( -0.48,  0.07)
thigh        |   0.20 ( -0.09,  0.49) |   0.26 (  0.00,  0.51)
forearm      |   0.44 (  0.05,  0.83) |   0.49 (  0.13,  0.86)
wrist        |  -1.59 ( -2.64, -0.54) |  -1.46 ( -2.47, -0.46)
chest        |  -0.03 ( -0.23,  0.18) |
ankle        |   0.17 ( -0.27,  0.61) |
height       |  -0.08 ( -0.44,  0.27) |
knee         |   0.04 ( -0.45,  0.52) |
biceps       |   0.16 ( -0.18,  0.50) |
-------------------------------------------------------------
Observations |                   248 |                   248
```

Note that `fit2` drops five of the 13 predictors from the `fit1` model.

```
check_model(fit2)
```


Posterior Predictive Check
Model−predicted lines should resemble obs
Density
siri_bf
Observed data
Model−predict
Linearity
Reference line should be flat and horizonta
Residuals
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
Leverage (hii)
203
212
217
39
0.9
0.9
Collinearity
High collinearity (VIF) may inflate parameter
Variance Inflation Factor (VIF, log−scaled)
abdomen
forearm
hip
neck
thigh
weight
wrist
Low (< 5)
Moderate (< 10)
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
Standard Normal Distribution Quantile

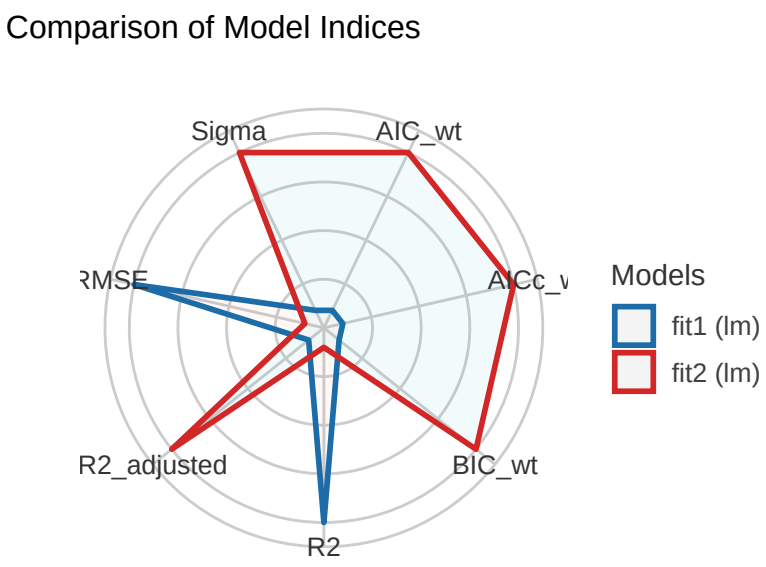
```
model_performance(fit2)
```

```
# Indices of model performance

AIC      |     AICc |      BIC |    R2 | R2 (adj.) |  RMSE | Sigma
-----------------------------------------------------------------
1431.933 | 1432.861 | 1467.067 | 0.740 |     0.732 | 4.169 | 4.247
```

```
compare_performance(fit1, fit2)
```

```
# Comparison of Model Performance Indices

Name | Model |  AIC (weights) | AICc (weights) |  BIC (weights) |    R2 | R2 (adj.) |  RMSE
-------------------------------------------------------------------------------------------
fit1 |    lm | 1440.0 (0.017) | 1442.1 (0.010) | 1492.7 (<.001) | 0.742 |     0.728 | 4.153
fit2 |    lm | 1431.9 (0.983) | 1432.9 (0.990) | 1467.1 (>.999) | 0.740 |     0.732 | 4.169
```

```
plot(compare_performance(fit1, fit2))
```



```
performance_mae(fit1)
```

```
[1] 3.418899
```

```
performance_mae(fit2)
```

```
[1] 3.412077
```

```
performance_rmse(fit1)
```

```
[1] 4.15292
```

```
performance_rmse(fit2)
```

```
[1] 4.169102
```

```
set.seed(431)
performance_accuracy(fit1, method = "cv", k = 5)
```

```
# Accuracy of Model Predictions

Accuracy (95% CI): 81.49% [73.12%, 92.09%]
Method: Correlation between observed and predicted
```

```
set.seed(431)
performance_accuracy(fit2, method = "cv", k = 5)
```

```
# Accuracy of Model Predictions

Accuracy (95% CI): 82.87% [74.24%, 93.11%]
Method: Correlation between observed and predicted
```

```
set.seed(431)
performance_cv(fit1)
```

```
# Cross-validation performance (30% holdout method)

MSE | RMSE |   R2
-----------------
16  |    4 | 0.74
```

```
set.seed(431)
performance_cv(fit2)
```

```
# Cross-validation performance (30% holdout method)

MSE | RMSE |   R2
-----------------
15  |  3.8 | 0.76
```

20.6 Best Subsets Regression and Mallows' C_p

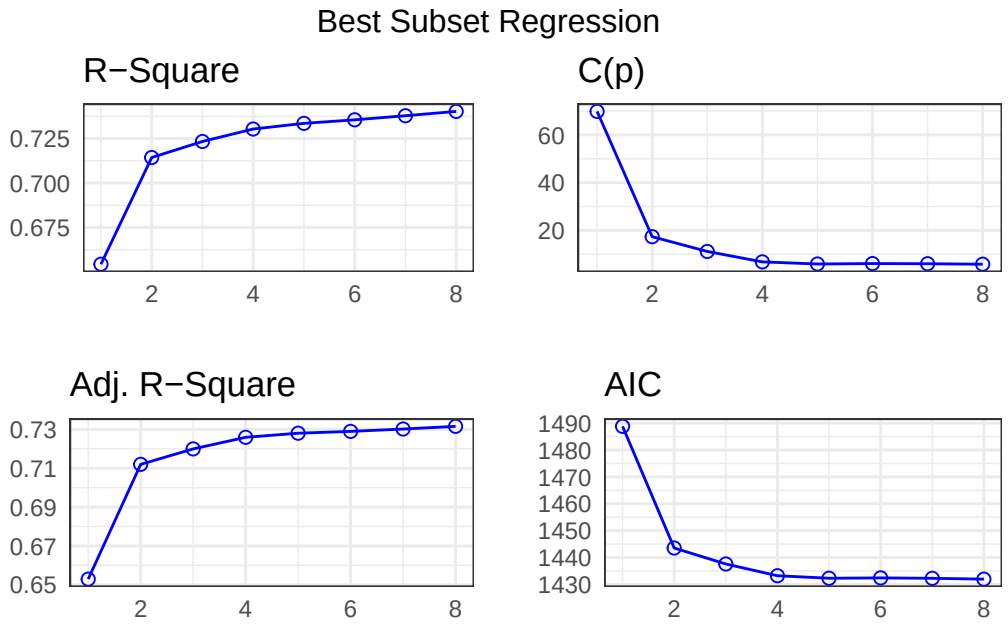
```
k <- ols_step_best_subset(fit1,
                          max_order = 8,
                          metric = "cp")
```

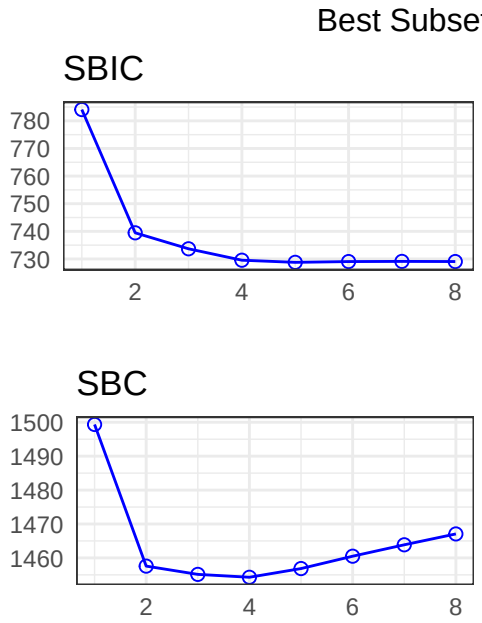
```
k
```

```
                    Best Subsets Regression
-------------------------------------------------------------------
Model Index    Predictors
-------------------------------------------------------------------
     1         abdomen
     2         weight abdomen
     3         weight abdomen wrist
     4         weight abdomen forearm wrist
     5         weight neck abdomen forearm wrist
     6         weight neck abdomen biceps forearm wrist
     7         age weight neck abdomen thigh forearm wrist
     8         age weight neck abdomen hip thigh forearm wrist
-------------------------------------------------------------------

                                                        Subsets Regression Summary
-----------------------------------------------------------------------------------------------------------
```

```
                     Adj.        Pred
Model    R-Square    R-Square    R-Square      C(p)          AIC          SBIC          SBC
-----------------------------------------------------------------------------------------------
  1        0.6543      0.6529      0.6438   69.8409    1488.8082     784.0875    1499.3485
  2        0.7143      0.7120      0.7047   17.3601    1443.5224     739.4622    1457.5761
  3        0.7233      0.7199      0.7104   11.1761    1437.5707     733.6757    1455.1379
  4        0.7304      0.7259      0.7159    6.8132    1433.2074     729.5437    1454.2879
  5        0.7335      0.7280      0.7175    5.9241    1432.2633     728.7699    1456.8573
  6        0.7355      0.7290      0.7176    6.1051    1432.3916     729.0562    1460.4990
  7        0.7378      0.7302      0.7175    6.0319    1432.2408     729.1139    1463.8617
  8        0.7403      0.7316      0.7186    5.8272    1431.9331     729.0634    1467.0674
-----------------------------------------------------------------------------------------------
AIC: Akaike Information Criteria
 SBIC: Sawa's Bayesian Information Criteria
 SBC: Schwarz Bayesian Criteria
 MSEP: Estimated error of prediction, assuming multivariate normality
 FPE: Final Prediction Error
 HSP: Hocking's Sp
 APC: Amemiya Prediction Criteria
```

```
plot(k)
```





20.7 Four-Predictor Model

```
fit4 <- lm(
  siri_bf ~ weight + abdomen + forearm + wrist,
  data = bodyfat
)
```

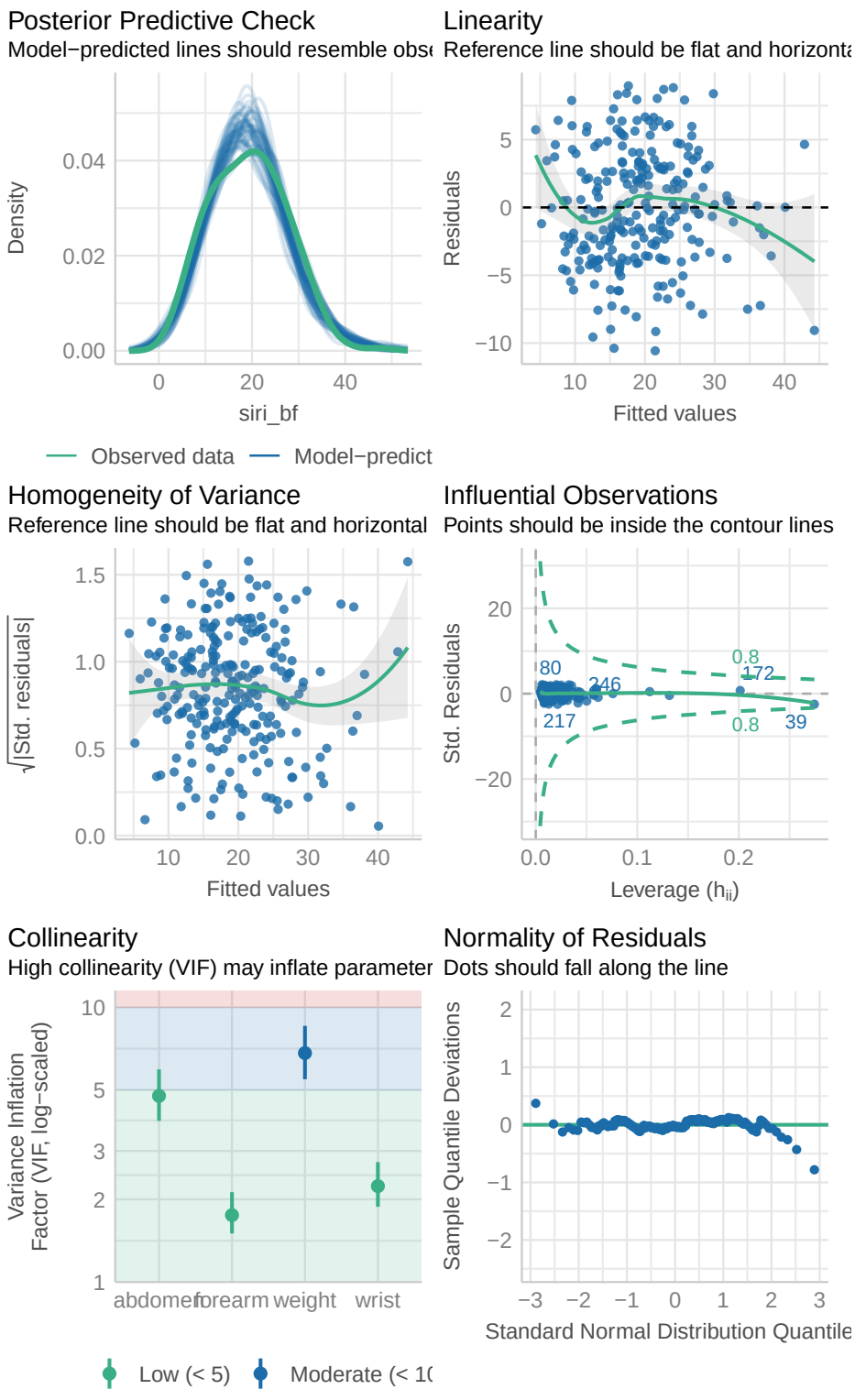
```
model_parameters(fit4, ci = 0.95)
```

```
Parameter   | Coefficient |   SE |           95% CI | t(243) |      p
-------------------------------------------------------------------
(Intercept) |      -33.61 | 7.26 | [-47.91, -19.32] |  -4.63 | < .001
weight      |       -0.14 | 0.02 | [ -0.19,  -0.09] |  -5.58 | < .001
abdomen     |        0.99 | 0.06 | [  0.88,   1.10] |  17.71 | < .001
forearm     |        0.45 | 0.18 | [  0.10,   0.81] |   2.51 | 0.013
wrist       |       -1.48 | 0.44 | [ -2.36,  -0.61] |  -3.34 | < .001


Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
  using a Wald t-distribution approximation.
```

```
model_performance(fit4)
```

```
# Indices of model performance

AIC      |     AICc |      BIC |    R2 | R2 (adj.) |  RMSE | Sigma
-----------------------------------------------------------------------
1433.207 | 1433.556 | 1454.288 | 0.730 |     0.726 | 4.248 | 4.291
```

```
check_model(fit4)
```


Posterior Predictive Check
Model−predicted lines should resemble obs
Density
0.04
0.02
0.00
0
20
40
siri_bf
Observed data
Model−predict
Linearity
Reference line should be flat and horizonta
Residuals
5
0
−5
−10
10
20
30
40
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
1.5
1.0
0.5
0.0
10
20
30
40
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
20
0
−20
80
246
217
0.8
172
0.8
39
0.0
0.1
0.2
Leverage (hii)
Collinearity
High collinearity (VIF) may inflate parameter
Variance Inflation Factor (VIF, log−scaled)
10
5
3
2
1
abdomen
forearm
weight
wrist
Low (< 5)
Moderate (< 10
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
2
1
0
−1
−2
−3
−2
−1
0
1
2
3
Standard Normal Distribution Quantile

20.8 Five-Predictor Model

```
fit5 <- lm(
  siri_bf ~ weight + neck + abdomen + forearm + wrist,
  data = bodyfat
)
```

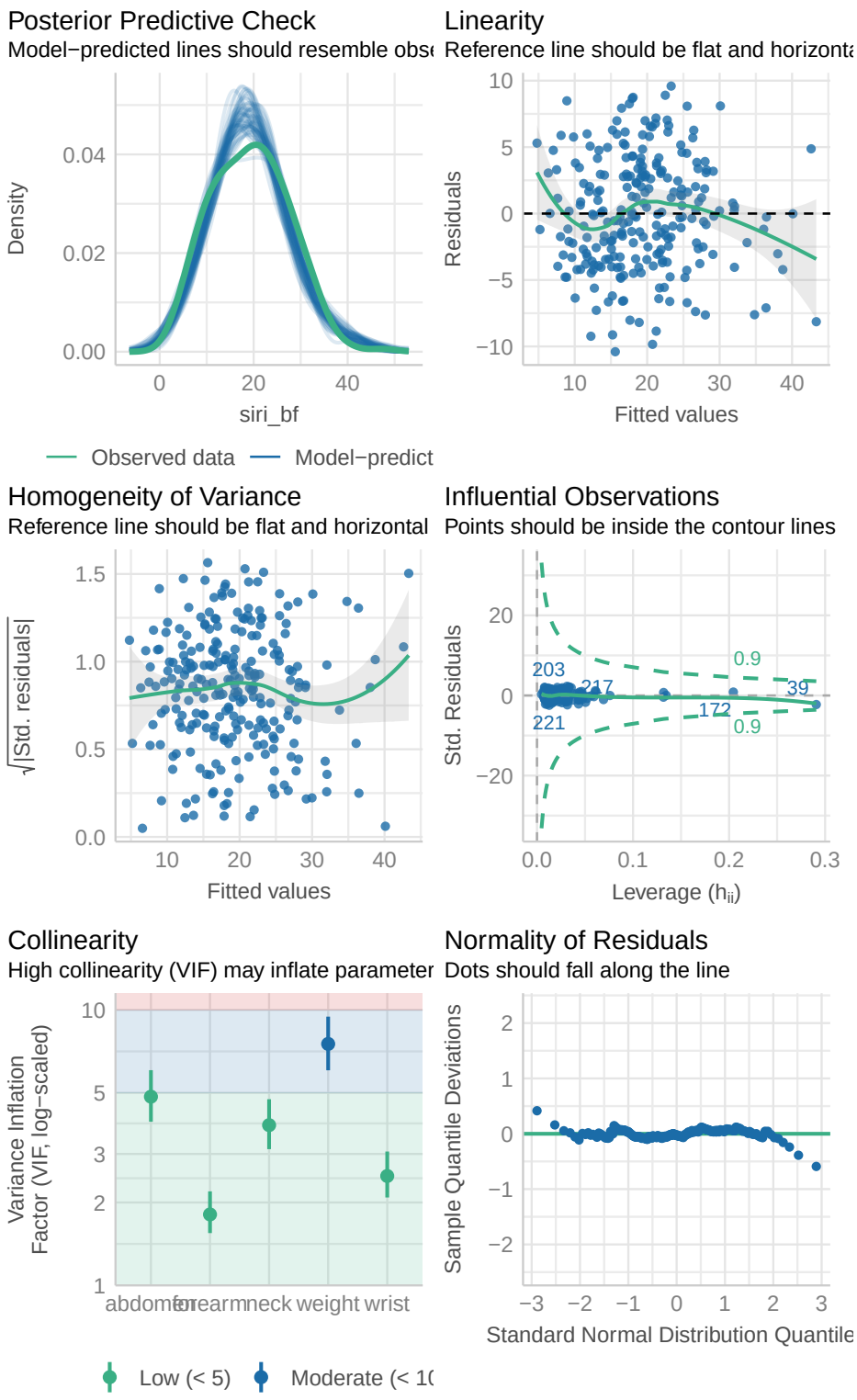
```
model_parameters(fit5, ci = 0.95)
```

```
Parameter   | Coefficient |   SE |           95% CI | t(242) |      p
-------------------------------------------------------------------
(Intercept) |      -29.23 | 7.67 | [-44.35, -14.11] |  -3.81 | < .001
weight      |       -0.12 | 0.03 | [ -0.17,  -0.07] |  -4.81 | < .001
neck        |       -0.37 | 0.22 | [ -0.81,   0.06] |  -1.70 | 0.090
abdomen     |        1.00 | 0.06 | [  0.89,   1.11] |  17.85 | < .001
forearm     |        0.51 | 0.18 | [  0.15,   0.87] |   2.79 | 0.006
wrist       |       -1.23 | 0.47 | [ -2.15,  -0.31] |  -2.62 | 0.009


Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
  using a Wald t-distribution approximation.
```

```
model_performance(fit5)
```

```
# Indices of model performance

AIC      |     AICc |      BIC |    R2 | R2 (adj.) |  RMSE | Sigma
-----------------------------------------------------------------
1432.263 | 1432.730 | 1456.857 | 0.734 |     0.728 | 4.223 | 4.275
```

```
check_model(fit5)
```


Posterior Predictive Check
Model−predicted lines should resemble obs
Density
0.04
0.02
0.00
0
20
40
siri_bf
Observed data
Model−predict
Linearity
Reference line should be flat and horizonta
Residuals
10
5
0
−5
−10
10
20
30
40
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
1.5
1.0
0.5
0.0
10
20
30
40
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
20
0
−20
203
217
221
172
39
0.9
0.9
0.0
0.1
0.2
0.3
Leverage (hii)
Collinearity
High collinearity (VIF) may inflate parameter
Variance Inflation Factor (VIF, log−scaled)
10
5
3
2
1
abdomen
forearm
neck
weight
wrist
Low (< 5)
Moderate (< 10
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
2
1
0
−1
−2
−3
−2
−1
0
1
2
3
Standard Normal Distribution Quantile

```r
compare_performance(fit4, fit5)
```

```
# Comparison of Model Performance Indices

Name | Model |  AIC (weights) | AICc (weights) |  BIC (weights) |    R2 | R2 (adj.) |  RMSE
-----------------------------------------------------------------------------------------------
fit4 |    lm | 1433.2 (0.384) | 1433.6 (0.398) | 1454.3 (0.783) | 0.730 |     0.726 | 4.248
fit5 |    lm | 1432.3 (0.616) | 1432.7 (0.602) | 1456.9 (0.217) | 0.734 |     0.728 | 4.223
```

```r
set.seed(431)
performance_accuracy(fit1, method = "cv", k = 5)
```

```
# Accuracy of Model Predictions

Accuracy (95% CI): 81.49% [73.12%, 92.09%]
Method: Correlation between observed and predicted
```

```r
set.seed(431)
performance_accuracy(fit2, method = "cv", k = 5)
```

```
# Accuracy of Model Predictions

Accuracy (95% CI): 82.87% [74.24%, 93.11%]
Method: Correlation between observed and predicted
```

```r
set.seed(431)
performance_accuracy(fit4, method = "cv", k = 5)
```

```
# Accuracy of Model Predictions

Accuracy (95% CI): 83.10% [74.74%, 92.71%]
Method: Correlation between observed and predicted
```

```r
set.seed(431)
performance_accuracy(fit5, method = "cv", k = 5)
```

```
# Accuracy of Model Predictions

Accuracy (95% CI): 82.99% [74.96%, 93.04%]
Method: Correlation between observed and predicted
```

```r
set.seed(431)
performance_cv(fit1)
```

```
# Cross-validation performance (30% holdout method)

MSE | RMSE |   R2
-----------------
16  |    4 | 0.74
```

```r
set.seed(431)
performance_cv(fit2)
```

```
# Cross-validation performance (30% holdout method)

MSE | RMSE |   R2
-----------------
15  |  3.8 | 0.76
```

```r
set.seed(431)
performance_cv(fit4)
```

```
# Cross-validation performance (30% holdout method)

MSE | RMSE |   R2
-----------------
14  |  3.8 | 0.76
```

```r
set.seed(431)
performance_cv(fit5)
```

```
# Cross-validation performance (30% holdout method)

MSE | RMSE |   R2
-----------------
15  |  3.9 | 0.75
```

20.9 Bayesian linear model

We'll now refit the four-predictor model using `stan_glm()` and a weakly informative prior, as usual.

```
fit4b <- stan_glm(
  siri_bf ~ weight + abdomen + forearm + wrist,
  data = bodyfat, refresh = 0
)
```

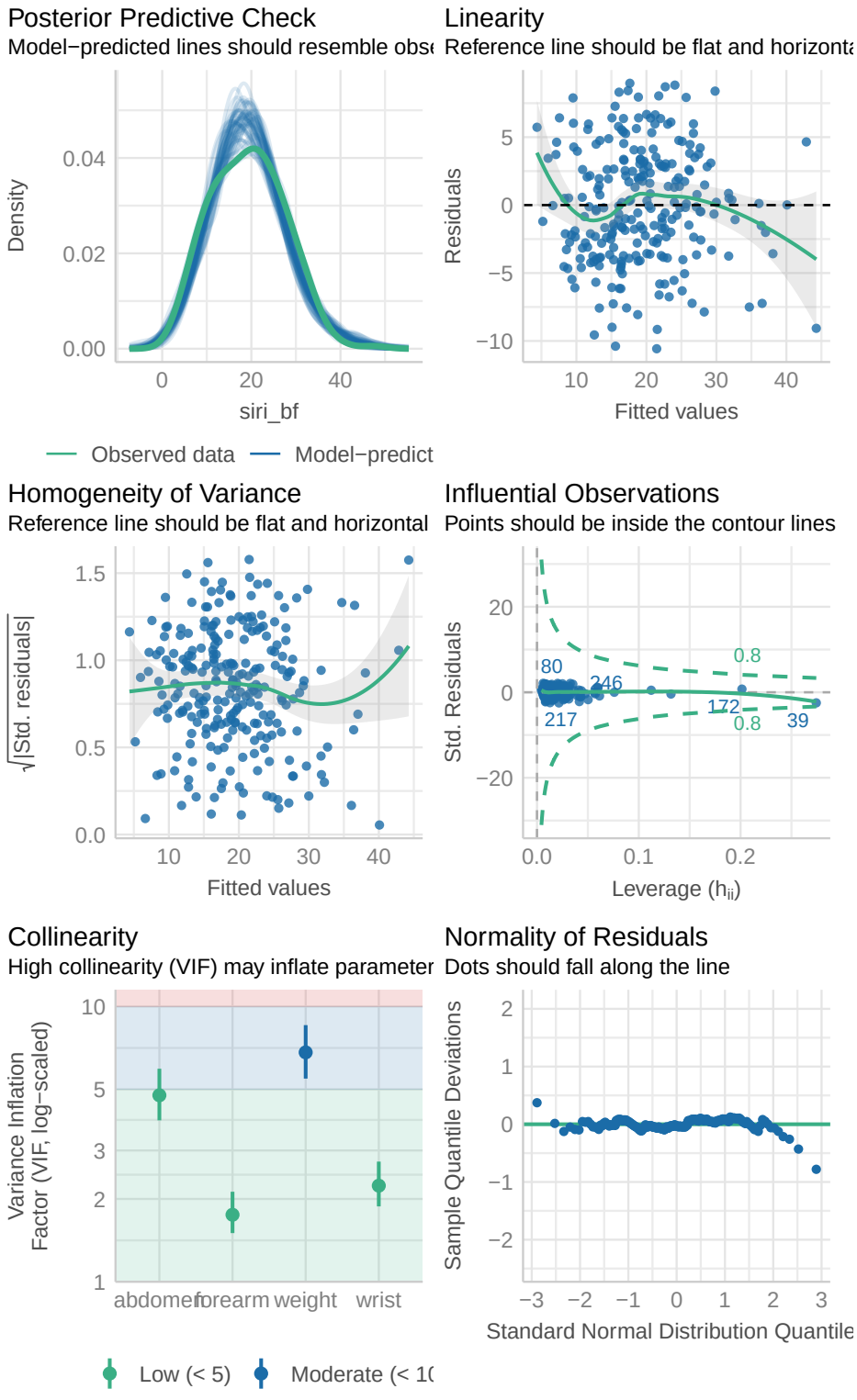
```
model_parameters(fit4b, ci = 0.95)
```

```
Parameter   | Median |           95% CI |     pd |  Rhat |     ESS |                   Prior
-------------------------------------------------------------------------------------------
(Intercept) | -33.40 | [-48.11, -19.22] |   100% | 1.003 | 2889.00 | Normal (19.28 +- 20.49)
weight      |  -0.14 | [ -0.19,  -0.09] |   100% | 1.004 | 2119.00 |   Normal (0.00 +- 0.71)
abdomen     |   0.99 | [  0.87,   1.10] |   100% | 1.002 | 2402.00 |   Normal (0.00 +- 1.92)
forearm     |   0.45 | [  0.09,   0.81] | 99.40% | 1.001 | 3820.00 |  Normal (0.00 +- 10.20)
wrist       |  -1.49 | [ -2.35,  -0.59] | 99.95% | 1.002 | 3160.00 |  Normal (0.00 +- 22.30)


Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
  using a MCMC distribution approximation.
```

```
model_performance(fit4b)
```

```
# Indices of model performance

ELPD     | ELPD_SE |    LOOIC | LOOIC_SE |     WAIC |    R2 | R2 (adj.) |  RMSE | Sigma
-------------------------------------------------------------------------------------------
-717.386 |   9.726 | 1434.773 |   19.452 | 1434.619 | 0.727 |     0.716 | 4.248 | 4.300
```

```
check_model(fit4b)
```


Posterior Predictive Check
Model−predicted lines should resemble obs
Density
0.04
0.02
0.00
0
20
40
siri_bf
Observed data
Model−predict
Linearity
Reference line should be flat and horizonta
Residuals
5
0
−5
−10
10
20
30
40
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
1.5
1.0
0.5
0.0
10
20
30
40
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
20
0
−20
0.8
80
246
172
217
0.8
39
0.0
0.1
0.2
Leverage (hii)
Collinearity
High collinearity (VIF) may inflate parameter
Variance Inflation Factor (VIF, log−scaled)
10
5
3
2
1
abdomen
forearm
weight
wrist
Low (< 5)
Moderate (< 10
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
2
1
0
−1
−2
−3
−2
−1
0
1
2
3
Standard Normal Distribution Quantile

```r
set.seed(431)
performance_accuracy(fit4b, method = "cv", k = 5)
```

```
# Accuracy of Model Predictions

Accuracy (95% CI): 83.11% [74.77%, 92.71%]
Method: Correlation between observed and predicted
```

```r
set.seed(431)
performance_cv(fit4b)
```

```
# Cross-validation performance (30% holdout method)

MSE | RMSE |   R2
-----------------
14  |  3.8 | 0.76
```

20.10 For More Information

21 Multiple Regression and Transformations

> **!** This is a DRAFT version of this Chapter.
>
> This is a sketchy draft. I'll remove this notice when I post a version of this Chapter that is essentially finished.

21.1 R setup for this chapter

> **i** Note
>
> Appendix A lists all R packages used in this book, and also provides R session information. Appendix B describes the 431-Love.R script, and demonstrates its use.

```
library(car)
library(glmnet)
library(haven)
library(janitor)
library(knitr)
library(naniar)
library(olsrr)
library(patchwork)
library(rstanarm)

library(easystats)
library(tidyverse)

source("data/Love-431.R")
theme_set(theme_bw())
```

21.2 Plasma Retinol and Beta-Carotene Dataset

> **Note**
>
> Appendix C provides further guidance on pulling data from other systems into R, while Appendix D gives more information (including download links) for all data sets used in this book.

The data in this study come from an unpublished study related to Nierenberg et al. (1989). From Thérèse Stukel, one of the authors of that study, and posted to Statlib as well as Frank Harrell's repository...

> Observational studies have suggested that low dietary intake or low plasma concentrations of retinol, beta-carotene, or other carotenoids might be associated with increased risk of developing certain types of cancer. However, relatively few studies have investigated the determinants of plasma concentrations of these micronutrients. We designed a cross-sectional study to investigate the relationship between personal characteristics and dietary factors, and plasma concentrations of retinol, beta-carotene and other carotenoids. Study subjects (N = 315) were patients who had an elective surgical procedure during a three-year period to biopsy or remove a lesion of the lung, colon, breast, skin, ovary or uterus that was found to be non-cancerous. Plasma concentrations of the micronutrients varied widely from subject to subject.

21.2.1 Data Ingest

```
plasma <- read_csv("data/plasma.csv", show_col_types = FALSE) |>
  mutate(across(where(is.character), as_factor)) |>
  mutate(P_ID = as.character(P_ID)) |>
  janitor::clean_names()

plasma
```

```
# A tibble: 315 x 15
   p_id    age sex    smokstat   bmi vituse       calories   fat fiber alcohol
   <chr> <dbl> <fct>  <fct>    <dbl> <fct>           <dbl> <dbl> <dbl>   <dbl>
 1 1        64 Female Former    21.5 Fairly_Often    1299.  57     6.3     0
 2 2        76 Female Never     23.9 Fairly_Often    1032.  50.1  15.8     0
 3 3        38 Female Former    20.0 Not_Often       2372.  83.6  19.1    14.1
 4 4        40 Female Former    25.1 Never           2450.  97.5  26.5     0.5
```

```
 5 5        72 Female Never     21.0 Fairly_Often    1952.  82.6  16.2     0
 6 6        40 Female Former    27.5 Never           1367.  56     9.6     1.3
 7 7        65 Female Never     22.0 Not_Often       2214.  52    28.7     0
 8 8        58 Female Never     28.8 Fairly_Often    1596.  63.4  10.9     0
 9 9        35 Female Never     23.1 Never           1800.  57.8  20.3     0.6
10 10       55 Female Former    35.0 Never           1264.  39.6  15.5     0
# i 305 more rows
# i 5 more variables: cholesterol <dbl>, betadiet <dbl>, retdiet <dbl>,
#   betaplasma <dbl>, retplasma <dbl>
```

21.2.2 Variable Descriptions

Variable	Description
p_id	Subject identification code
age	Age (years)
sex	Sex (Male, Female)
smokstat	Smoking status (Never, Former, Current Smoker)
bmi	Body mass index (weight/(height^2))
vituse	Vitamin Use (Yes, fairly often; Yes, not often; No)
calories	Number of calories consumed per day.
fat	Grams of fat consumed per day.
fiber	Grams of fiber consumed per day.
alcohol	Number of alcoholic drinks consumed per week.
cholesterol	Cholesterol consumed (mg per day).
betadiet	Dietary beta-carotene consumed (mcg per day).
retdiet	Dietary retinol consumed (mcg per day)
betaplasma	Plasma beta-carotene (ng/ml)
retplasma	Plasma Retinol (ng/ml)

```
n_miss(plasma)
```

```
[1] 0
```

21.2.3 Data Summary; Outlier Removal

```
describe_distribution(plasma)
```

Variable	Mean	SD	IQR	Range	Skewness	Kurtosis	n	m
age	50.15	14.58	24.00	[19.00, 83.00]	0.30	-0.87	315	
bmi	26.16	6.01	7.16	[16.33, 50.40]	1.38	2.01	315	
calories	1796.65	680.35	772.60	[445.20, 6662.20]	1.75	8.13	315	
fat	77.03	33.83	41.40	[14.40, 235.90]	1.10	2.02	315	
fiber	12.79	5.33	6.50	[3.10, 36.80]	1.15	2.48	315	
alcohol	3.28	12.32	3.20	[0.00, 203.00]	13.82	221.33	315	
cholesterol	242.46	131.99	154.00	[37.70, 900.70]	1.48	3.41	315	
betadiet	2185.60	1473.89	1749.00	[214.00, 9642.00]	1.61	3.47	315	
retdiet	832.71	589.29	568.00	[30.00, 6901.00]	4.47	38.07	315	
betaplasma	189.89	183.00	142.00	[0.00, 1415.00]	3.56	17.21	315	
retplasma	602.79	208.90	253.00	[179.00, 1727.00]	1.31	4.02	315	

That `alcohol` value of 203 drinks per week is concerning.

```
plasma |> count(alcohol) |> tail()
```

```
# A tibble: 6 x 2
  alcohol     n
    <dbl> <int>
1    18.2     1
2    20       1
3    21       1
4    22       1
5    35       2
6   203       1
```

The next highest value is 35 drinks per week. I'm going to make the decision to remove this observation from the data, for the remainder of this chapter.

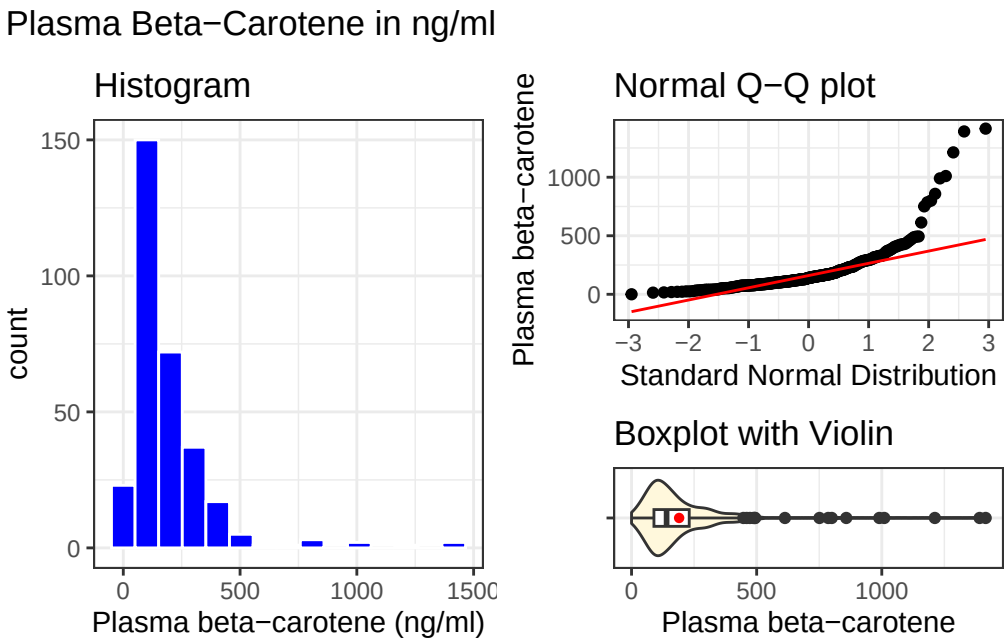
```
plasma <- plasma |>
  filter(alcohol < 200)

plasma |> reframe(lovedist(alcohol)) |> kable(digits = 2)
```

n	miss	mean	sd	med	mad	min	q25	q75	max
314	0	2.64	4.95	0.3	0.44	0	0	3.2	35

21.3 Predicting Plasma Beta-Carotene

Having removed the subject with an unrealistic `alcohol` value, our first main task is to predict `betaplasma` using (a subset of) most of the other variables in the data. Let's look at that outcome.

```
p1 <- ggplot(plasma, aes(x = betaplasma)) +
  geom_histogram(bins = 15, fill = "blue", col = "white") +
  labs(
    x = "Plasma beta-carotene (ng/ml)",
    title = "Histogram"
  )

p2 <- ggplot(plasma, aes(sample = betaplasma)) +
  geom_qq() +
  geom_qq_line(col = "red") +
  labs(
    y = "Plasma beta-carotene",
    x = "Standard Normal Distribution",
    title = "Normal Q-Q plot"
  )

p3 <- ggplot(plasma, aes(x = betaplasma, y = "")) +
  geom_violin(fill = "cornsilk") +
  geom_boxplot(width = 0.2) +
  stat_summary(
    fun = mean, geom = "point",
    shape = 16, col = "red"
  ) +
  labs(
    y = "", x = "Plasma beta-carotene",
    title = "Boxplot with Violin"
  )

p1 + (p2 / p3 + plot_layout(heights = c(2, 1))) +
  plot_annotation(
    title = "Plasma Beta-Carotene in ng/ml"
  )
```



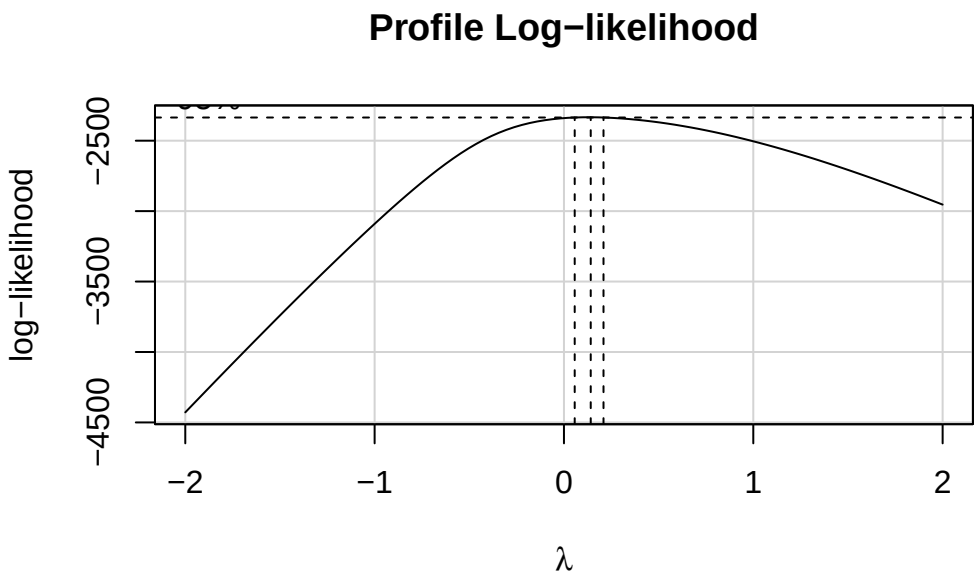
We see a fairly pronounced skew to the right here.

```
plasma |> reframe(lovedist(betaplasma))
```

```
# A tibble: 1 x 10
      n  miss  mean    sd   med   mad   min   q25   q75   max
  <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1   314     0  190.  183.   140  88.2     0  89.5  230.  1415
```

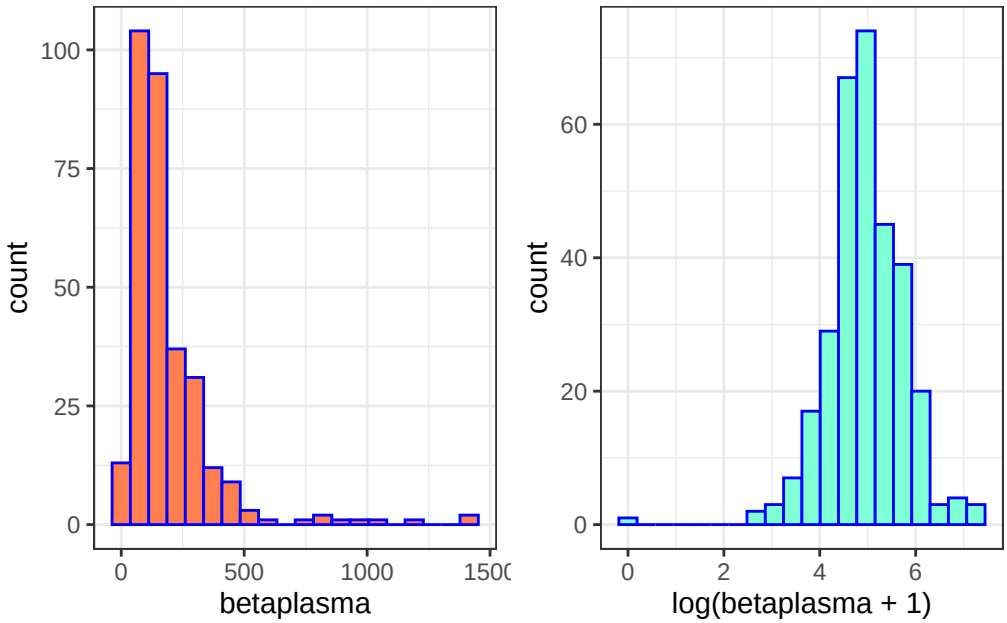
Note that we have a minimum value of 0 here. To use the Box-Cox method, we need to have values of our outcome that are strictly positive. The simplest solution to this problem (of seeing one or more zero values but no negative values) is to simply add one to each value of `betaplasma` before considering transformations.

```
fit0 <- lm((betaplasma + 1) ~ age + sex + smokstat + bmi + vituse +
  calories + fat + fiber + alcohol + cholesterol +
  betadiet, data = plasma)

boxCox(fit0)
```



We'll apply a logarithmic transformation to (`betaplasma` + 1) as our revised outcome for modeling.

```
p1 <- ggplot(plasma, aes(x = betaplasma)) +
  geom_histogram(bins = 20, fill = "coral", col = "blue")

p2 <- ggplot(plasma, aes(x = log(betaplasma + 1))) +
  geom_histogram(bins = 20, fill = "aquamarine", col = "blue")

p1 + p2
```



Most of our modeling of this transformed outcome is going to be affected, to some degree, by the outlying 0 value. Let's take a closer look at that:

```
plasma |> count(betaplasma) |> head()
```

```
# A tibble: 6 x 2
  betaplasma     n
       <dbl> <int>
1          0     1
2         14     1
3         16     1
4         19     1
5         21     1
6         22     1
```

A value of 0 ng/ml for plasma beta-carotene also seems at best implausible. I'm going to go ahead and remove that subject from our study now, as well, and reframe our decision about a transformation accordingly.

```
plasma <- plasma |> filter(betaplasma > 0)
```

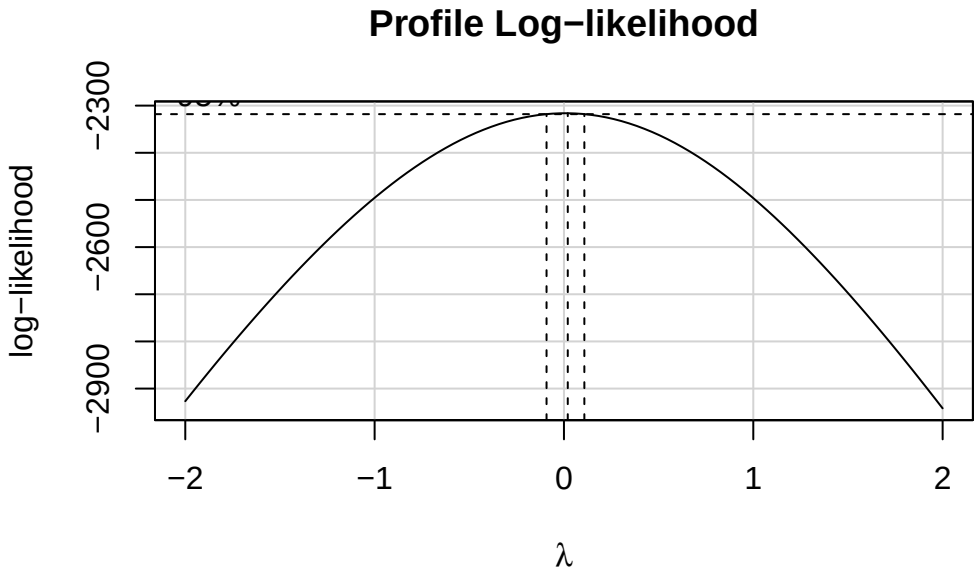
21.3.1 Final Analytic Data (- 2 observations)

So now, having removed two observations:

- one for having an implausible level of `betaplasma` of 0 ng/ml, and
- one for having an implaubible level of `alcohol` of 203 drinks per week,

we consider again the issue of transforming our outcome, and a Box-Cox plot, but now, thanks to dropping one of our outliers, we have a strictly positive set of `betaplasma` values to work with, so we no longer need to add 1 to that outcome.

```
fit00 <- lm(betaplasma ~
              age + sex + smokstat + bmi + vituse +
              calories + fat + fiber + alcohol +
              cholesterol + betadiet,
            data = plasma)

boxCox(fit00)
```



Again, we're strongly encouraged to fit a logarithm of the outcome, by virtue of the estimated $\lambda = 0$.

21.3.2 Initial OLS Fit

```
plasma <- plasma |>
  mutate(logbeta = log(betaplasma))

fit1 <- lm(logbeta ~ age + sex + smokstat + bmi +
             vituse + calories + fat + fiber + alcohol +
             cholesterol + betadiet,
           data = plasma)
```

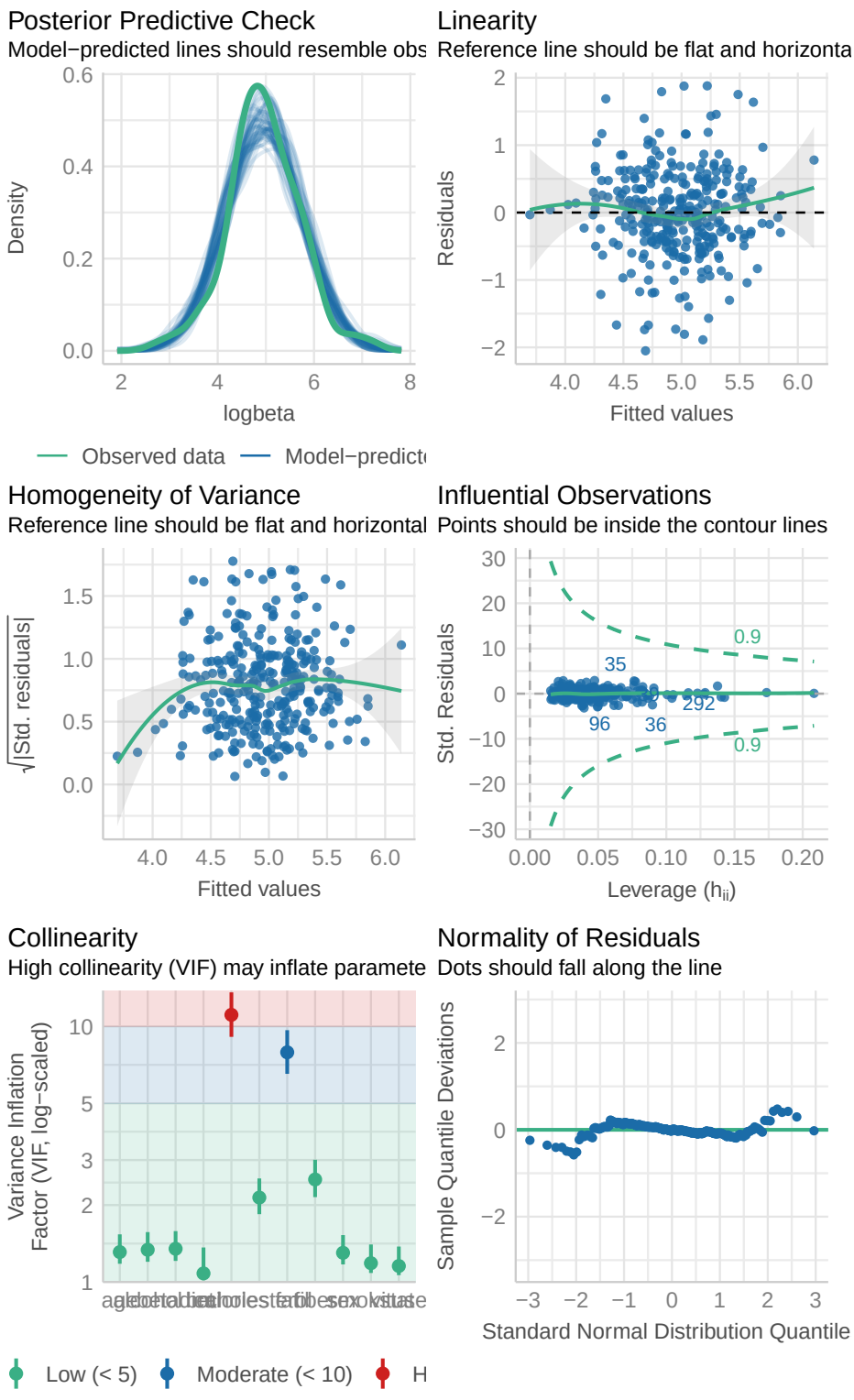
```
model_parameters(fit1)
```

```
Parameter          | Coefficient |       SE |         95% CI | t(299) |      p
-------------------------------------------------------------------------------
(Intercept)        |        5.48 |     0.28 | [ 4.93,  6.04] |  19.57 | < .001
age                |    5.24e-03 | 2.94e-03 | [ 0.00,  0.01] |   1.78 | 0.076
sex [Male]         |       -0.23 |     0.13 | [-0.48,  0.02] |  -1.85 | 0.066
smokstat [Never]   |        0.09 |     0.08 | [-0.08,  0.25] |   1.02 | 0.308
smokstat [Current] |       -0.21 |     0.12 | [-0.45,  0.04] |  -1.66 | 0.098
bmi                |       -0.03 | 6.48e-03 | [-0.04, -0.02] |  -4.87 | < .001
vituse [Not_Often] |       -0.03 |     0.10 | [-0.22,  0.17] |  -0.26 | 0.797
vituse [Never]     |       -0.30 |     0.09 | [-0.48, -0.12] |  -3.25 | 0.001
calories           |   -2.19e-04 | 2.01e-04 | [ 0.00,  0.00] |  -1.09 | 0.278
fat                |    1.40e-03 | 3.18e-03 | [ 0.00,  0.01] |   0.44 | 0.659
fiber              |        0.03 |     0.01 | [ 0.01,  0.05] |   2.78 | 0.006
alcohol            |    5.25e-03 | 8.76e-03 | [-0.01,  0.02] |   0.60 | 0.549
cholesterol        |   -2.70e-04 | 4.38e-04 | [ 0.00,  0.00] |  -0.62 | 0.538
betadiet           |    4.69e-05 | 2.95e-05 | [ 0.00,  0.00] |   1.59 | 0.113


Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
  using a Wald t-distribution approximation.
```

```
model_performance(fit1)
```

```
# Indices of model performance

AIC     |    AICc |     BIC |    R2 | R2 (adj.) |  RMSE | Sigma
-----------------------------------------------------------------
646.464 | 648.080 | 702.657 | 0.249 |     0.216 | 0.648 | 0.663
```

```
check_model(fit1)
```


Posterior Predictive Check
Model−predicted lines should resemble obs
Density
logbeta
Observed data
Model−predict
Linearity
Reference line should be flat and horizonta
Residuals
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
Leverage (hii)
0.9
35
292
96
36
0.9
Collinearity
High collinearity (VIF) may inflate paramete
Variance Inflation Factor (VIF, log−scaled)
Low (< 5)
Moderate (< 10)
H
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
Standard Normal Distribution Quantile

21.3.3 Best Subsets Suggestion?

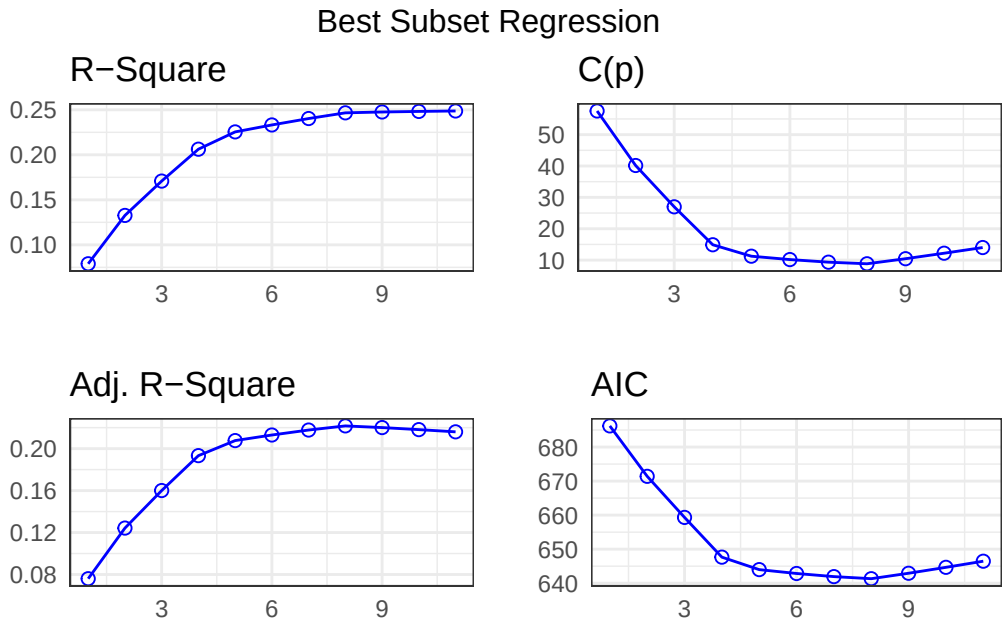
```
k <- ols_step_best_subset(fit1,
                          max_order = 11,
                          metric = "cp")
```

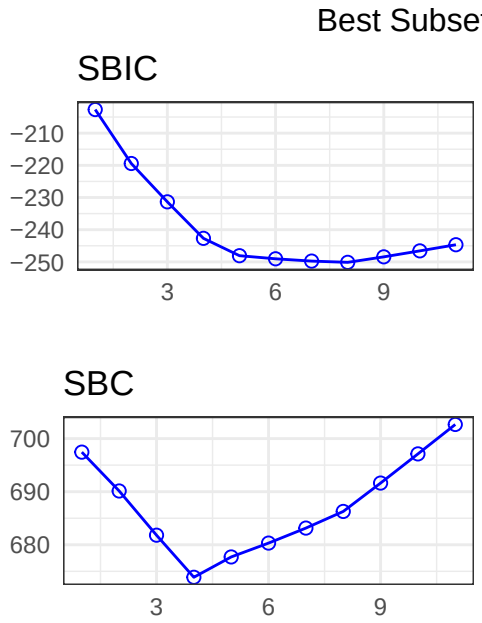
```
k
```

```
                                Best Subsets Regression
-------------------------------------------------------------------------------------------
Model Index    Predictors
-------------------------------------------------------------------------------------------
     1         bmi
     2         bmi vituse
     3         bmi vituse fiber
     4         bmi vituse calories fiber
     5         smokstat bmi vituse calories fiber
     6         smokstat bmi vituse calories fiber betadiet
     7         age sex smokstat bmi vituse calories fiber
     8         age sex smokstat bmi vituse calories fiber betadiet
     9         age sex smokstat bmi vituse calories fiber cholesterol betadiet
    10         age sex smokstat bmi vituse calories fiber alcohol cholesterol betadiet
    11         age sex smokstat bmi vituse calories fat fiber alcohol cholesterol betadiet
-------------------------------------------------------------------------------------------

                                                    Subsets Regression Summary
-------------------------------------------------------------------------------------------------------
                      Adj.        Pred
Model    R-Square    R-Square    R-Square     C(p)        AIC         SBIC         SBC
-------------------------------------------------------------------------------------------------------
  1        0.0790      0.0760      0.0672    57.5387    686.2096    -202.6728    697.4483
  2        0.1327      0.1243       0.111    40.1581    671.3979    -219.4340    690.1289
  3        0.1708      0.1600      0.1438    26.9872    659.3289    -231.3437    681.8061
  4        0.2062      0.1933       0.175    14.8858    647.6592    -242.6879    673.8826
  5        0.2254      0.2077      0.1839    11.2438    643.9939    -248.0769    677.7098
  6        0.2332      0.2130      0.1862    10.1707    642.8577    -249.0400    680.3197
  7        0.2403      0.2177      0.1883     9.3375    641.9383    -249.7554    683.1465
  8        0.2466      0.2217       0.189     8.8230    641.3242    -250.1441    686.2786
  9        0.2476      0.2201      0.1847    10.4380    642.9220    -248.4318    691.6227
 10        0.2482      0.2181        0.18    12.1951    644.6680    -246.5767    697.1149
 11        0.2487      0.2160      0.1738    14.0000    646.4639    -244.6723    702.6569
```

```
-------------------------------------------------------------------------------------------
AIC: Akaike Information Criteria
 SBIC: Sawa's Bayesian Information Criteria
 SBC: Schwarz Bayesian Criteria
 MSEP: Estimated error of prediction, assuming multivariate normality
 FPE: Final Prediction Error
 HSP: Hocking's Sp
 APC: Amemiya Prediction Criteria
```

```
plot(k)
```





I'm going to look more closely at the four-predictor (suggested by SBC) and eight-predictor (suggested by C_p) subsets[1]

21.3.4 Four-Predictor Model

```
fit4 <- lm(logbeta ~ bmi + vituse + calories + fiber,
           data = plasma)
```

```
compare_models(fit1, fit4)
```

```
Parameter          |                   fit1 |                   fit4
------------------------------------------------------------------------
(Intercept)        |     5.48 ( 4.93,  6.04) |     5.79 ( 5.37,  6.21)
bmi                |    -0.03 (-0.04, -0.02) |    -0.03 (-0.04, -0.02)
vituse (Not_Often) |    -0.03 (-0.22,  0.17) |    -0.07 (-0.26,  0.12)
vituse (Never)     |    -0.30 (-0.48, -0.12) |    -0.36 (-0.53, -0.18)
calories           | -2.19e-04 ( 0.00,  0.00) | -2.66e-04 ( 0.00,  0.00)
fiber              |     0.03 ( 0.01,  0.05) |     0.04 ( 0.03,  0.06)
```

[1] If we run `select_parameters(fit1)`, that stepwise procedure suggests the 8-predictor subset identified by best subsets, as it turns out.

```
smokstat (Current) |     -0.21 (-0.45,  0.04) |
age                |  5.24e-03 ( 0.00,  0.01) |
sex (Male)         |     -0.23 (-0.48,  0.02) |
smokstat (Never)   |      0.09 (-0.08,  0.25) |
cholesterol        | -2.70e-04 ( 0.00,  0.00) |
fat                |  1.40e-03 ( 0.00,  0.01) |
alcohol            |  5.25e-03 (-0.01,  0.02) |
betadiet           |  4.69e-05 ( 0.00,  0.00) |
-------------------------------------------------------------------------
Observations       |                       313 |                       313
```

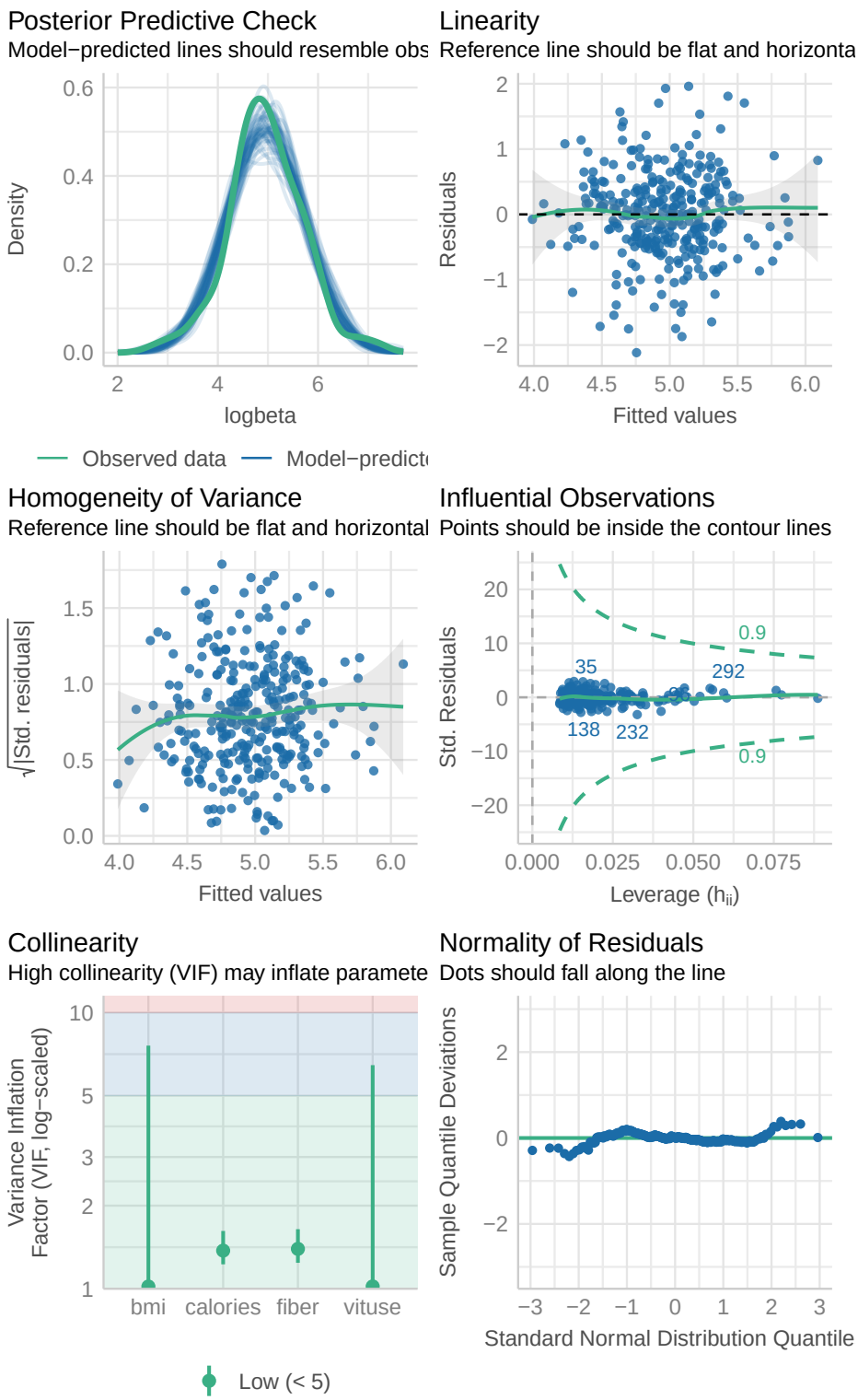
```
model_parameters(fit4, ci = 0.95)
```

```
Parameter          | Coefficient |       SE |         95% CI | t(307) |      p
-------------------------------------------------------------------------------
(Intercept)        |        5.79 |     0.21 | [ 5.37,  6.21] |  27.17 | < .001
bmi                |       -0.03 | 6.38e-03 | [-0.04, -0.02] |  -4.63 | < .001
vituse [Not_Often] |       -0.07 |     0.10 | [-0.26,  0.12] |  -0.69 | 0.493
vituse [Never]     |       -0.36 |     0.09 | [-0.53, -0.18] |  -3.99 | < .001
calories           |   -2.66e-04 | 7.19e-05 | [ 0.00,  0.00] |  -3.70 | < .001
fiber              |        0.04 | 8.41e-03 | [ 0.03,  0.06] |   5.21 | < .001


Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
  using a Wald t-distribution approximation.
```

```
model_performance(fit4)
```

```
# Indices of model performance

AIC     |    AICc |     BIC |    R2 | R2 (adj.) |  RMSE | Sigma
-----------------------------------------------------------------
647.659 | 648.026 | 673.883 | 0.206 |     0.193 | 0.666 | 0.672
```

```
check_model(fit4)
```


Posterior Predictive Check
Model−predicted lines should resemble obs
Density
logbeta
Observed data
Model−predict
Linearity
Reference line should be flat and horizonta
Residuals
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
Leverage (h_ii)
0.9
0.9
35
292
138
232
Collinearity
High collinearity (VIF) may inflate paramete
Variance Inflation Factor (VIF, log−scaled)
bmi
calories
fiber
vituse
Low (< 5)
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
Standard Normal Distribution Quantile

21.3.5 Eight-Predictor Model

```
fit8 <- lm(logbeta ~ age + sex + smokstat + bmi +
             vituse + calories + fiber + betadiet,
           data = plasma)
```

```
compare_models(fit1, fit8)
```

Parameter	fit1	fit8
(Intercept)	5.48 (4.93, 6.04)	5.49 (4.95, 6.04)
age	5.24e-03 (0.00, 0.01)	5.40e-03 (0.00, 0.01)
sex (Male)	-0.23 (-0.48, 0.02)	-0.23 (-0.48, 0.01)
smokstat (Never)	0.09 (-0.08, 0.25)	0.08 (-0.09, 0.24)
smokstat (Current)	-0.21 (-0.45, 0.04)	-0.21 (-0.46, 0.03)
bmi	-0.03 (-0.04, -0.02)	-0.03 (-0.05, -0.02)
vituse (Not_Often)	-0.03 (-0.22, 0.17)	-0.02 (-0.21, 0.17)
vituse (Never)	-0.30 (-0.48, -0.12)	-0.29 (-0.46, -0.11)
calories	-2.19e-04 (0.00, 0.00)	-1.70e-04 (0.00, 0.00)
fiber	0.03 (0.01, 0.05)	0.03 (0.01, 0.05)
betadiet	4.69e-05 (0.00, 0.00)	4.65e-05 (0.00, 0.00)
cholesterol	-2.70e-04 (0.00, 0.00)	
fat	1.40e-03 (0.00, 0.01)	
alcohol	5.25e-03 (-0.01, 0.02)	
Observations	313	313

```
model_parameters(fit8)
```

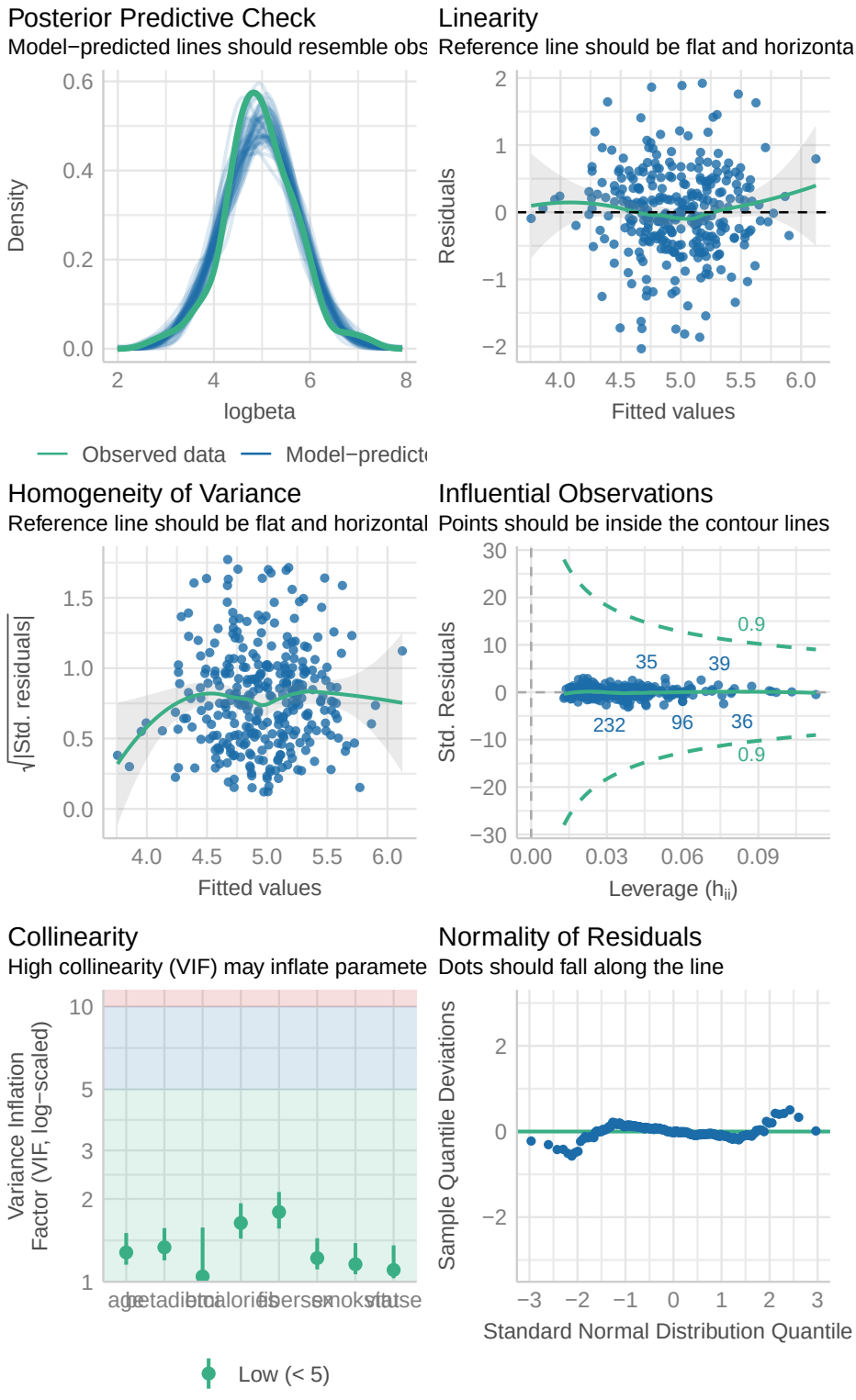
Parameter	Coefficient	SE	95% CI	t(302)	p
(Intercept)	5.49	0.28	[4.95, 6.04]	19.90	< .001
age	5.40e-03	2.90e-03	[0.00, 0.01]	1.86	0.063
sex [Male]	-0.23	0.12	[-0.48, 0.01]	-1.92	0.056
smokstat [Never]	0.08	0.08	[-0.09, 0.24]	0.94	0.350
smokstat [Current]	-0.21	0.12	[-0.46, 0.03]	-1.75	0.081
bmi	-0.03	6.35e-03	[-0.05, -0.02]	-5.13	< .001
vituse [Not_Often]	-0.02	0.10	[-0.21, 0.17]	-0.24	0.814
vituse [Never]	-0.29	0.09	[-0.46, -0.11]	-3.21	0.001
calories	-1.70e-04	7.71e-05	[0.00, 0.00]	-2.21	0.028

```
fiber              |        0.03 | 9.36e-03 | [ 0.01,  0.05] |   3.13 | 0.002
betadiet           |    4.65e-05 | 2.92e-05 | [ 0.00,  0.00] |   1.59 | 0.113


Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
  using a Wald t-distribution approximation.
```

```
model_performance(fit8)
```

```
# Indices of model performance

AIC     |    AICc |     BIC |    R2 | R2 (adj.) |  RMSE | Sigma
-----------------------------------------------------------------
641.324 | 642.364 | 686.279 | 0.247 |     0.222 | 0.649 | 0.660
```

```
check_model(fit8)
```

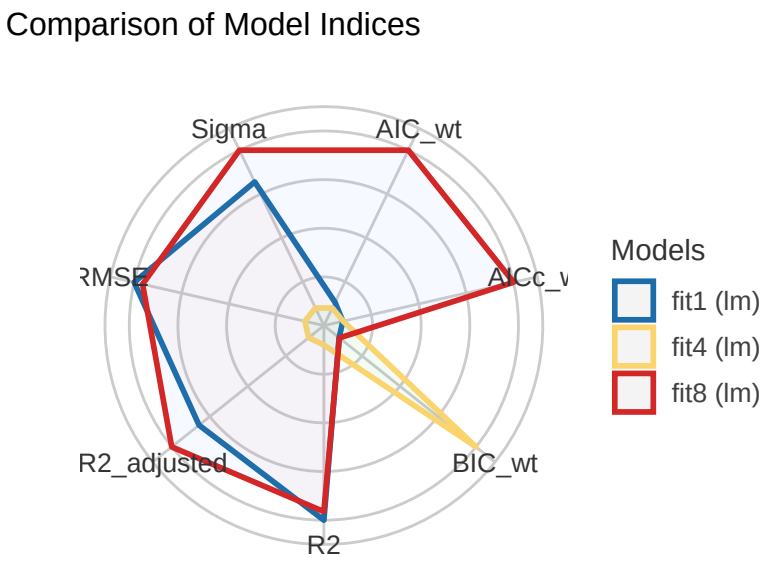

Posterior Predictive Check
Model−predicted lines should resemble obs
Density
logbeta
Observed data
Model−predict
Linearity
Reference line should be flat and horizonta
Residuals
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
Leverage (h_ii)
0.9
35
39
232
96
36
0.9
Collinearity
High collinearity (VIF) may inflate paramete
Variance Inflation Factor (VIF, log−scaled)
Low (< 5)
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
Standard Normal Distribution Quantile

21.3.6 Our Three Models So Far

```
compare_performance(fit1, fit4, fit8)
```

```
# Comparison of Model Performance Indices

Name | Model | AIC (weights) | AICc (weights) | BIC (weights) |    R2 | R2 (adj.) |  RMSE | S
-------------------------------------------------------------------------------------------
fit1 |    lm | 646.5 (0.068) |  648.1 (0.051) | 702.7 (<.001) | 0.249 |     0.216 | 0.648 | 0
fit4 |    lm | 647.7 (0.038) |  648.0 (0.053) | 673.9 (0.998) | 0.206 |     0.193 | 0.666 | 0
fit8 |    lm | 641.3 (0.894) |  642.4 (0.896) | 686.3 (0.002) | 0.247 |     0.222 | 0.649 | 0
```

```
plot(compare_performance(fit1, fit4, fit8))
```




```
set.seed(431)
performance_accuracy(fit1, method = "cv", k = 5)
```

```
# Accuracy of Model Predictions

Accuracy (95% CI): 42.42% [32.42%, 52.44%]
Method: Correlation between observed and predicted
```

```r
set.seed(431)
performance_accuracy(fit4, method = "cv", k = 5)
```

```
# Accuracy of Model Predictions

Accuracy (95% CI): 42.98% [37.50%, 52.45%]
Method: Correlation between observed and predicted
```

```r
set.seed(431)
performance_accuracy(fit8, method = "cv", k = 5)
```

```
# Accuracy of Model Predictions

Accuracy (95% CI): 44.05% [32.92%, 57.60%]
Method: Correlation between observed and predicted
```

```r
set.seed(431)
performance_cv(fit1)
```

```
# Cross-validation performance (30% holdout method)

MSE  | RMSE |   R2
------------------
0.56 | 0.75 | 0.19
```

```r
set.seed(431)
performance_cv(fit4)
```

```
# Cross-validation performance (30% holdout method)

MSE  | RMSE |   R2
------------------
0.56 | 0.75 | 0.19
```

```r
set.seed(431)
performance_cv(fit8)
```

```
# Cross-validation performance (30% holdout method)

MSE  | RMSE |   R2
------------------
0.55 | 0.74 | 0.21
```

21.4 Using the Lasso

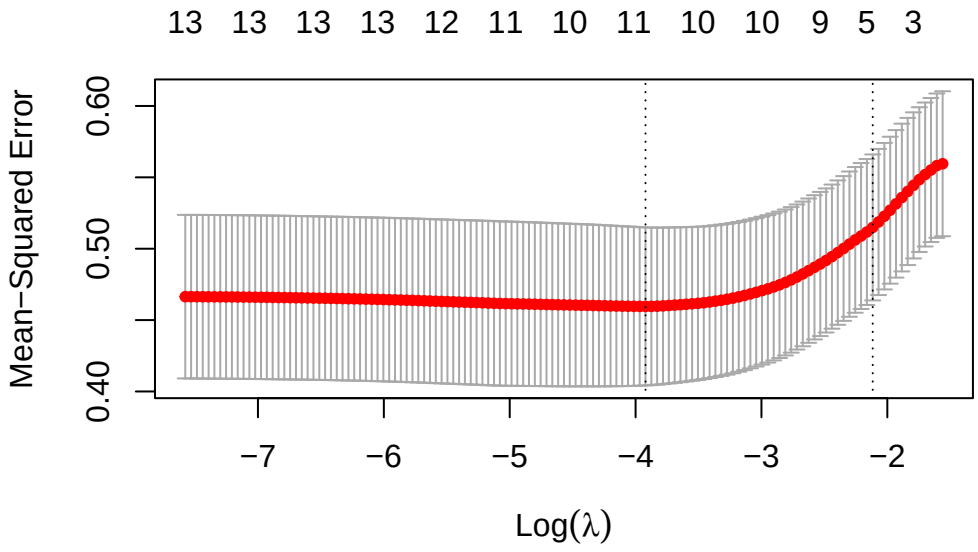
Need to discuss glmnet package, and point to:

- https://rpubs.com/jmkelly91/881590
- https://glmnet.stanford.edu/articles/glmnet.html#linear-regression-family-gaussian-default

and add some of the comments I used in https://thomaselove.github.io/432-notes/lasso.html#using-the-lasso-to-suggest-a-smaller-model

```
pred_x <- model.matrix(fit1)
out_y <- plasma |> select(logbeta) |> as.matrix()

set.seed(431)
lasso <- cv.glmnet(x = pred_x, y = out_y,
                   type.measure = "mse",
                   alpha = 1, family = "gaussian",
                   nlambda = 200, nfolds = 10)
```

```
plot(lasso)
```




```
log(lasso$lambda.min)
```

```
[1] -3.921033
```

```
predict(lasso, type = "coef", s = "lambda.min")
```

```
15 x 1 sparse Matrix of class "dgCMatrix"
                    lambda.min
(Intercept)      5.423141e+00
(Intercept)      .
age              4.601321e-03
sexMale         -1.703956e-01
smokstatNever    6.368383e-02
smokstatCurrent -1.874663e-01
bmi             -2.918528e-02
vituseNot_Often  .
vituseNever     -2.486722e-01
calories        -5.125006e-05
fat             -4.158782e-04
fiber            2.216516e-02
alcohol          .
```

```
cholesterol     -3.158420e-04
betadiet         4.038103e-05
```

We can use the lasso to identify a set of predictors to drop. Here, the lasso only drops the `alcohol` variable from our initial set of predictors, since we aren't willing to drop just one level of a multi-categorical predictor like `vituse`.

21.4.1 No "alcohol" model

```
fit10 <- lm(logbeta ~ age + sex + smokstat + bmi +
              vituse + calories + fat + fiber +
              cholesterol + betadiet,
            data = plasma)
```

```
compare_models(fit1, fit10)
```

```
Parameter          |                    fit1 |                   fit10
-------------------------------------------------------------------------
(Intercept)        |      5.48 ( 4.93,  6.04) |      5.49 ( 4.94,  6.04)
age                |  5.24e-03 ( 0.00,  0.01) |  5.35e-03 ( 0.00,  0.01)
sex (Male)         |     -0.23 (-0.48,  0.02) |     -0.22 (-0.46,  0.02)
smokstat (Never)   |      0.09 (-0.08,  0.25) |      0.08 (-0.09,  0.24)
smokstat (Current) |     -0.21 (-0.45,  0.04) |     -0.21 (-0.45,  0.03)
bmi                |     -0.03 (-0.04, -0.02) |     -0.03 (-0.04, -0.02)
vituse (Not_Often) |     -0.03 (-0.22,  0.17) |     -0.02 (-0.21,  0.17)
vituse (Never)     |     -0.30 (-0.48, -0.12) |     -0.29 (-0.47, -0.11)
calories           | -2.19e-04 ( 0.00,  0.00) | -1.73e-04 ( 0.00,  0.00)
fat                |  1.40e-03 ( 0.00,  0.01) |  8.53e-04 (-0.01,  0.01)
fiber              |      0.03 ( 0.01,  0.05) |      0.03 ( 0.01,  0.05)
cholesterol        | -2.70e-04 ( 0.00,  0.00) | -2.90e-04 ( 0.00,  0.00)
betadiet           |  4.69e-05 ( 0.00,  0.00) |  4.81e-05 ( 0.00,  0.00)
alcohol            |  5.25e-03 (-0.01,  0.02) |
-------------------------------------------------------------------------
Observations       |                     313 |                     313
```

```
model_parameters(fit10)
```

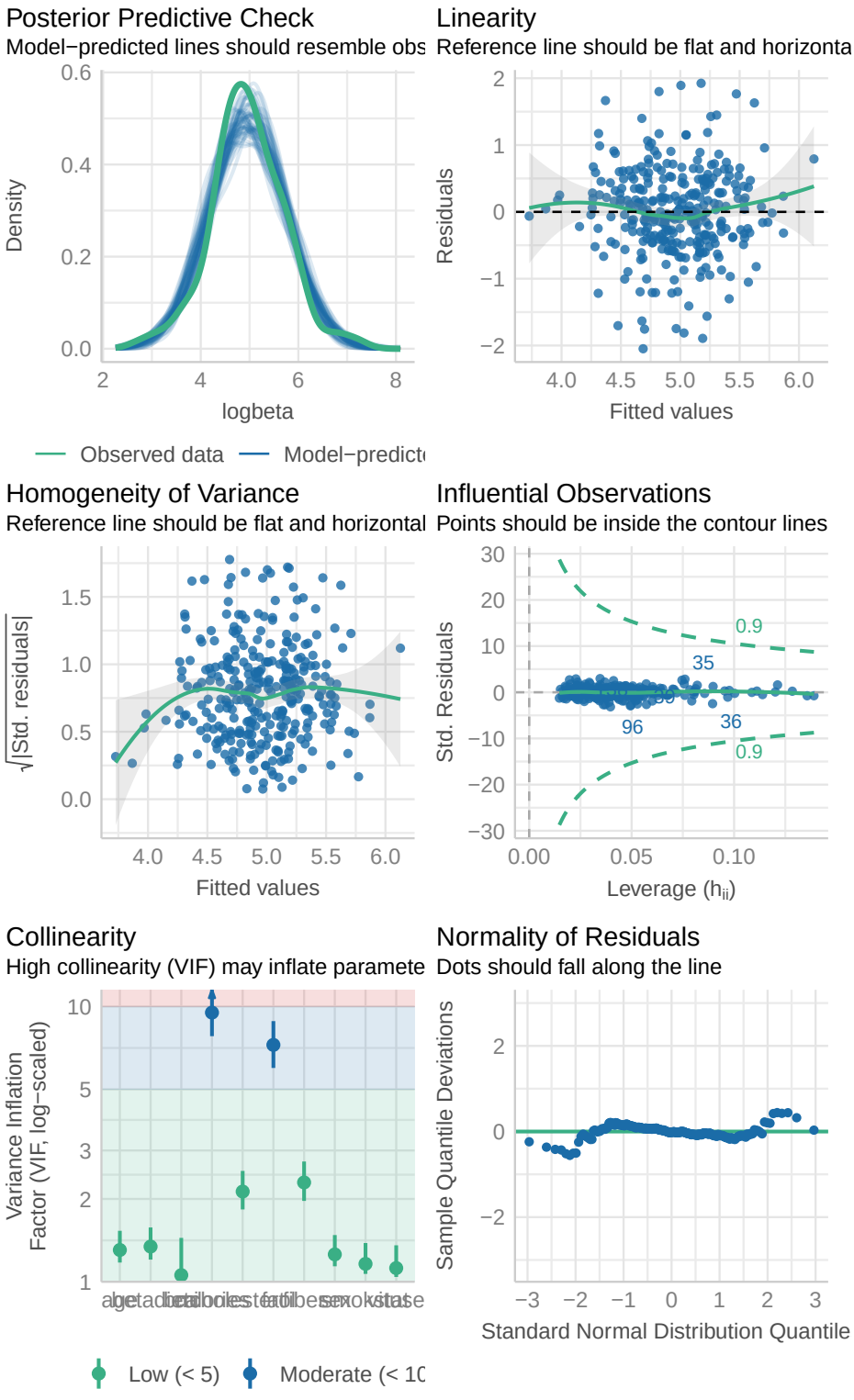
```
Parameter          | Coefficient |       SE |         95% CI | t(300) |      p
-------------------------------------------------------------------------------
```

```
(Intercept)        |        5.49 |     0.28 | [ 4.94,  6.04] |  19.66 | < .001
age                |    5.35e-03 | 2.94e-03 | [ 0.00,  0.01] |   1.82 | 0.069
sex [Male]         |       -0.22 |     0.12 | [-0.46,  0.02] |  -1.77 | 0.078
smokstat [Never]   |        0.08 |     0.08 | [-0.09,  0.24] |   0.95 | 0.343
smokstat [Current] |       -0.21 |     0.12 | [-0.45,  0.03] |  -1.72 | 0.087
bmi                |       -0.03 | 6.40e-03 | [-0.04, -0.02] |  -5.02 | < .001
vituse [Not_Often] |       -0.02 |     0.10 | [-0.21,  0.17] |  -0.21 | 0.834
vituse [Never]     |       -0.29 |     0.09 | [-0.47, -0.11] |  -3.20 | 0.002
calories           |   -1.73e-04 | 1.86e-04 | [ 0.00,  0.00] |  -0.93 | 0.353
fat                |    8.53e-04 | 3.04e-03 | [-0.01,  0.01] |   0.28 | 0.779
fiber              |        0.03 |     0.01 | [ 0.01,  0.05] |   2.72 | 0.007
cholesterol        |   -2.90e-04 | 4.36e-04 | [ 0.00,  0.00] |  -0.66 | 0.507
betadiet           |    4.81e-05 | 2.94e-05 | [ 0.00,  0.00] |   1.64 | 0.103


Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
  using a Wald t-distribution approximation.
```

```
model_performance(fit10)
```

```
# Indices of model performance

AIC     |    AICc |     BIC |    R2 | R2 (adj.) |  RMSE | Sigma
-----------------------------------------------------------------
644.840 | 646.249 | 697.287 | 0.248 |     0.218 | 0.648 | 0.662
```

```
check_model(fit10)
```

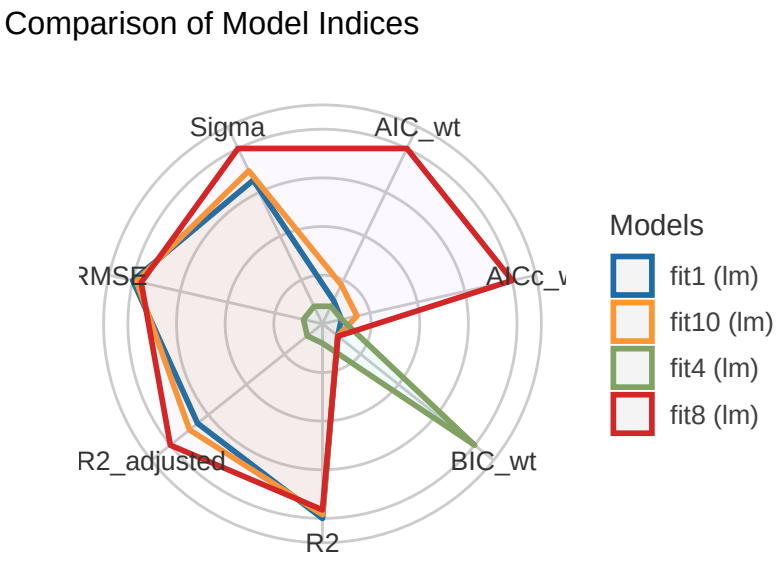

Posterior Predictive Check
Model−predicted lines should resemble obs
Density
logbeta
Observed data
Model−predict
Linearity
Reference line should be flat and horizonta
Residuals
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
Leverage (h_ii)
Collinearity
High collinearity (VIF) may inflate paramete
Variance Inflation Factor (VIF, log−scaled)
Low (< 5)
Moderate (< 10
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
Standard Normal Distribution Quantile

21.4.2 Comparing the Four Models

```
compare_performance(fit1, fit4, fit8, fit10)
```

```
# Comparison of Model Performance Indices

Name  | Model | AIC (weights) | AICc (weights) | BIC (weights) |    R2 | R2 (adj.) |  RMSE |
-----------------------------------------------------------------------------------------------
fit1  |    lm | 646.5 (0.059) |  648.1 (0.046) | 702.7 (<.001) | 0.249 |     0.216 | 0.648 |
fit4  |    lm | 647.7 (0.033) |  648.0 (0.047) | 673.9 (0.998) | 0.206 |     0.193 | 0.666 |
fit8  |    lm | 641.3 (0.775) |  642.4 (0.794) | 686.3 (0.002) | 0.247 |     0.222 | 0.649 |
fit10 |    lm | 644.8 (0.134) |  646.2 (0.114) | 697.3 (<.001) | 0.248 |     0.218 | 0.648 |
```

```
plot(compare_performance(fit1, fit4, fit8, fit10))
```



Within our sample, it looks like `fit8` is the strongest of these models.

How about its cross-validation performance?

```
set.seed(431)
performance_accuracy(fit10, method = "cv", k = 5)
```

```
# Accuracy of Model Predictions

Accuracy (95% CI): 42.65% [32.47%, 52.20%]
Method: Correlation between observed and predicted
```

```
set.seed(431)
performance_cv(fit10)
```

```
# Cross-validation performance (30% holdout method)

MSE  | RMSE |  R2
-----------------
0.55 | 0.74 | 0.2
```

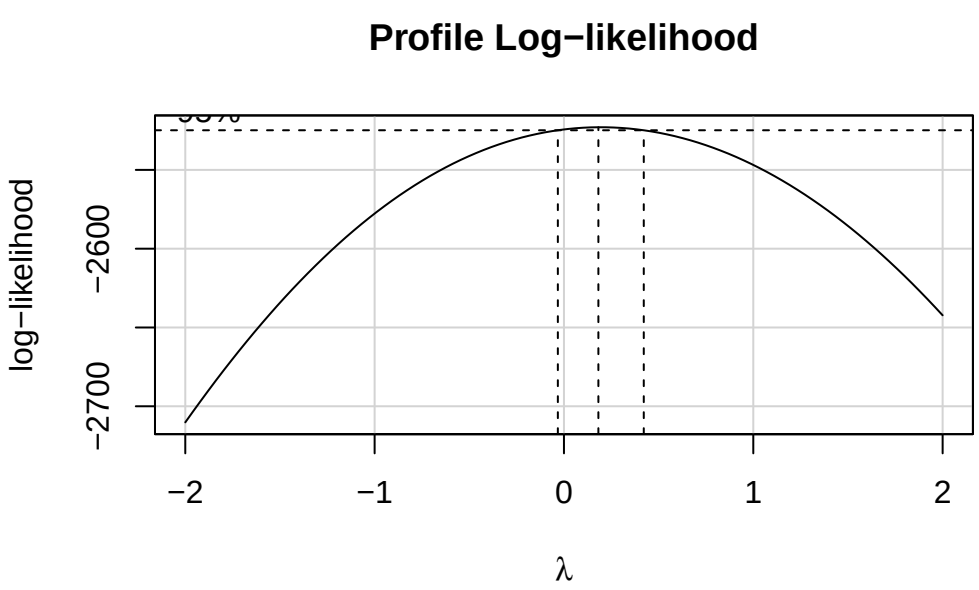
21.4.3 Table of Cross-validated Performance

Model	Accuracy and 95% CI	MSE_{30}	$RMSE_{30}$	R^2_{30}
fit1	.4242 (.3242, .5244)	0.56	0.75	0.19
fit10	.4265 (.3247, .5220)	0.55	0.74	0.20
fit8	.4405 (.3292, .5760)	0.56	0.75	0.19
fit4	.4298 (.3750, .5245)	0.55	0.74	0.21

It appears that `fit8` also performs well in cross-validation, in terms of accuracy. Model `fit4` may be a bit better in the measures assessed here using the 30% holdout sample approach. Still, given the strong in-sample performance of `fit8`, I think I would be inclined to select that model.

21.5 Predicting Plasma Retinol

```
fit_ret0 <- lm(retplasma ~ age + sex + smokstat + bmi + vituse +
  calories + fat + fiber + alcohol + cholesterol +
  retdiet, data = plasma)

boxCox(fit_ret0)
```



It appears we might want to be working with the logarithm of `retplasma` as our outcome.

```
plasma <- plasma |>
  mutate(logret = log(retplasma))

fit_ret1 <- lm(logret ~ age + sex + smokstat + bmi +
                 vituse + calories + fat + fiber +
                 alcohol + cholesterol + retdiet,
               data = plasma)
```

```
model_parameters(fit_ret1, ci = 0.95)
```

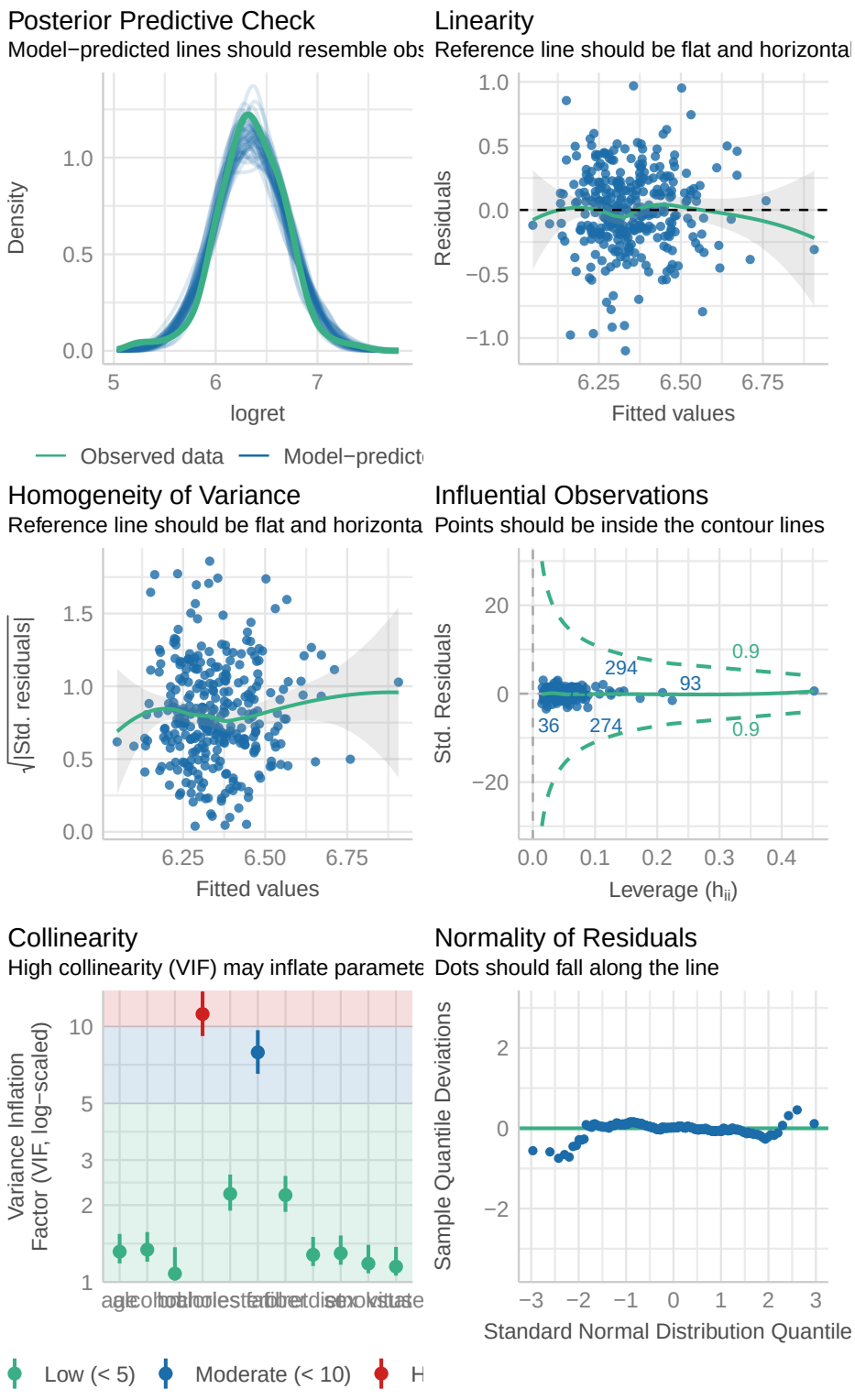
```
Parameter          | Coefficient |       SE |        95% CI | t(299) |      p
-------------------------------------------------------------------------------
(Intercept)        |        6.17 |     0.14 | [ 5.90, 6.44] |  45.37 | < .001
age                |    4.98e-03 | 1.43e-03 | [ 0.00, 0.01] |   3.48 | < .001
sex [Male]         |        0.06 |     0.06 | [-0.06, 0.18] |   1.00 | 0.316
smokstat [Never]   |       -0.07 |     0.04 | [-0.15, 0.01] |  -1.74 | 0.083
smokstat [Current] |       -0.09 |     0.06 | [-0.20, 0.03] |  -1.43 | 0.153
bmi                |    1.38e-03 | 3.14e-03 | [ 0.00, 0.01] |   0.44 | 0.659
vituse [Not_Often] |    2.27e-03 |     0.05 | [-0.09, 0.09] |   0.05 | 0.962
vituse [Never]     |       -0.04 |     0.04 | [-0.13, 0.05] |  -0.86 | 0.390
```

```
calories           |    1.51e-04 | 9.81e-05 | [ 0.00, 0.00] |   1.54 | 0.124
fat                |   -2.93e-03 | 1.54e-03 | [-0.01, 0.00] |  -1.90 | 0.058
fiber              |   -8.58e-03 | 5.04e-03 | [-0.02, 0.00] |  -1.70 | 0.090
alcohol            |        0.01 | 4.25e-03 | [ 0.00, 0.02] |   2.54 | 0.012
cholesterol        |   -7.67e-05 | 2.16e-04 | [ 0.00, 0.00] |  -0.36 | 0.723
retdiet            |    1.99e-06 | 3.57e-05 | [ 0.00, 0.00] |   0.06 | 0.956


Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
  using a Wald t-distribution approximation.
```

```
model_performance(fit_ret1)
```

```
# Indices of model performance

AIC     |    AICc |     BIC |    R2 | R2 (adj.) |  RMSE | Sigma
-----------------------------------------------------------------
193.645 | 195.262 | 249.839 | 0.125 |     0.087 | 0.314 | 0.322
```

```
check_model(fit_ret1)
```


Posterior Predictive Check
Model−predicted lines should resemble obs
Density
logret
Observed data
Model−predict
Linearity
Reference line should be flat and horizontal
Residuals
Fitted values
Homogeneity of Variance
Reference line should be flat and horizonta
√|Std. residuals|
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
Leverage (hii)
0.9
294
93
36
274
0.9
Collinearity
High collinearity (VIF) may inflate paramete
Variance Inflation Factor (VIF, log−scaled)
Low (< 5)
Moderate (< 10)
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
Standard Normal Distribution Quantile

Here, the use of a stepwise approach suggests a 6-predictor subset, including `age`, `smokstat`, `calories`, `fat`, `fiber` and `alcohol`:

```
fit_ret6 <- select_parameters(fit_ret1)

compare_models(fit_ret1, fit_ret6)
```

Parameter	fit_ret1	fit_ret6
(Intercept)	6.17 (5.90, 6.44)	6.17 (5.97, 6.36)
age	4.98e-03 (0.00, 0.01)	5.36e-03 (0.00, 0.01)
smokstat (Never)	-0.07 (-0.15, 0.01)	-0.07 (-0.15, 0.01)
smokstat (Current)	-0.09 (-0.20, 0.03)	-0.09 (-0.21, 0.02)
calories	1.51e-04 (0.00, 0.00)	1.65e-04 (0.00, 0.00)
fat	-2.93e-03 (-0.01, 0.00)	-3.18e-03 (-0.01, 0.00)
fiber	-8.58e-03 (-0.02, 0.00)	-8.95e-03 (-0.02, 0.00)
alcohol	0.01 (0.00, 0.02)	0.01 (0.00, 0.02)
cholesterol	-7.67e-05 (0.00, 0.00)	
bmi	1.38e-03 (0.00, 0.01)	
sex (Male)	0.06 (-0.06, 0.18)	
vituse (Never)	-0.04 (-0.13, 0.05)	
retdiet	1.99e-06 (0.00, 0.00)	
vituse (Not_Often)	2.27e-03 (-0.09, 0.09)	
Observations	313	313

I'll leave for the reader the exercise of evaluating this particular subset, and perhaps developing other potential choices before making additional modeling decisions. One might also consider returning to the data the observation with an unreasably low `betaplasma`, since that variable isn't involved in our prediction of `retplasma`.

21.6 For More Information

22 Multiple Regression and Imputation

> ! This is a DRAFT version of this Chapter.
>
> This is a sketchy draft. I'll remove this notice when I post a version of this Chapter that is essentially finished.

22.1 R setup for this chapter

> i Note
>
> Appendix A lists all R packages used in this book, and also provides R session information. Appendix B describes the 431-Love.R script, and demonstrates its use.

```
library(broom)
library(car)
library(ggmice)
library(glmnet)
library(janitor)
library(knitr)
library(mice)
library(naniar)
library(olsrr)
library(patchwork)
library(rstanarm)

library(easystats)
library(tidyverse)

source("data/Love-431.R")
theme_set(theme_bw())
```

22.2 Data will be Countries of the World version 2

> **Note**
>
> Appendix C provides further guidance on pulling data from other systems into R, while Appendix D gives more information (including download links) for all data sets used in this book.

```
ctry <- read_csv("data/countries.csv", show_col_types = FALSE) |>
  mutate(
    UNIV_CARE = factor(UNIV_CARE),
    WHO_REGION = factor(WHO_REGION),
    WBANK_INCOME = factor(WBANK_INCOME),
    C_NUM = as.character(C_NUM)
  ) |>
  janitor::clean_names()

ctry
```

```
# A tibble: 192 x 14
   c_num country_name    hale_all univ_care wbank_income hale_2000 births_attend
   <chr> <chr>              <dbl> <fct>     <fct>            <dbl>         <dbl>
 1 1     Afghanistan         50.4 no        Low               46.8            68
 2 2     Albania             66.7 yes       Upper-middle      65.2           100
 3 3     Algeria             65.5 yes       Lower-middle      62.7            99
 4 4     Andorra             NA   no        High              NA             100
 5 5     Angola              53.8 no        Lower-middle      42.9            50
 6 6     Antigua and Ba~     66.5 no        High              65.5            99
 7 7     Argentina           64.8 yes       Upper-middle      65.1            99
 8 8     Armenia             64   no        Upper-middle      63.5           100
 9 9     Australia           70.6 yes       High              68.6            99
10 10    Austria             69.8 yes       High              68.2            98
# i 182 more rows
# i 7 more variables: ihr_core_cap <dbl>, avg_tempc <dbl>, female_22 <dbl>,
#   rural_22 <dbl>, covid_vax100 <dbl>, who_region <fct>, iso_alpha3 <chr>
```

```
miss_var_summary(ctry)
```

```
# A tibble: 14 x 3
   variable      n_miss pct_miss
```

```
   <chr>          <int>    <num>
 1 births_attend     20   10.4
 2 hale_all           9    4.69
 3 hale_2000          9    4.69
 4 covid_vax100       6    3.12
 5 ihr_core_cap       2    1.04
 6 wbank_income       1    0.521
 7 c_num              0    0
 8 country_name       0    0
 9 univ_care          0    0
10 avg_tempc          0    0
11 female_22          0    0
12 rural_22           0    0
13 who_region         0    0
14 iso_alpha3         0    0
```

```
miss_case_table(ctry)
```

```
# A tibble: 4 x 3
  n_miss_in_case n_cases pct_cases
           <int>   <int>     <dbl>
1              0     164     85.4
2              1      16      8.33
3              2       5      2.60
4              3       7      3.65
```

- Outcome HALE_ALL
- 3 categorical variables (UNIV_CARE, WHO_REGION, WBANK_INCOME)
- 8 quantitative predictors (HALE_2000, BIRTHS_ATTEND, IHR_CORE_CAP, AVG_TEMPC, FEMALE_22, RURAL_22, COVID_VAX100)

and some missing values (at least in the predictors) to deal with…

22.3 Build single imputation

22.3.1 Filter out countries without information on our outcome, `hale_all`.

```
ctry_cc <- ctry |>
  filter(complete.cases(hale_all))

pct_miss_case(ctry_cc)
```

```
[1] 10.38251
```

Later, when we do multiple imputation, we'll run 15 imputations.

22.3.2 Single Imputation

```
ctry_si <-
  mice(ctry_cc, m = 1, seed = 431, print = FALSE) |>
  complete() |>
  tibble()
```

```
Warning: Number of logged events: 3
```

```
dim(ctry_si)
```

```
[1] 183  14
```

```
n_miss(ctry_si)
```

```
[1] 0
```

22.3.3 Partition data after single imputation

We've seen other ways to partition the data. Here, let's use `data_partition()` from the `datawizard` package, part of the `easystats` ecosystem.

```
out <- data_partition(ctry_si, proportion = 0.7, seed = 431)

ctry_si_train <- out$p_0.7
ctry_si_test <- out$test

dim(ctry_si_train)
```

```
[1] 128  15
```

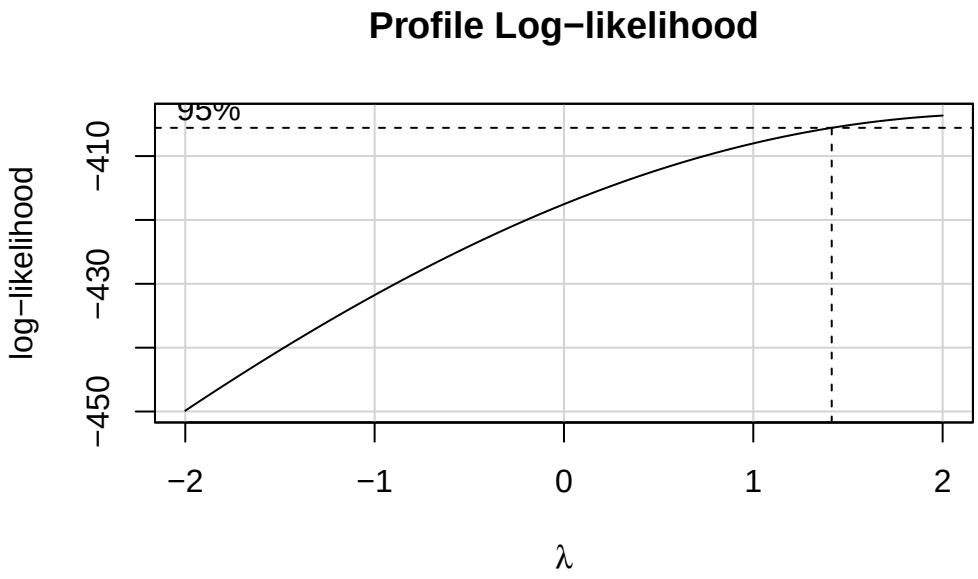
```
dim(ctry_si_test)
```

```
[1] 55 15
```

22.4 Transforming the outcome?

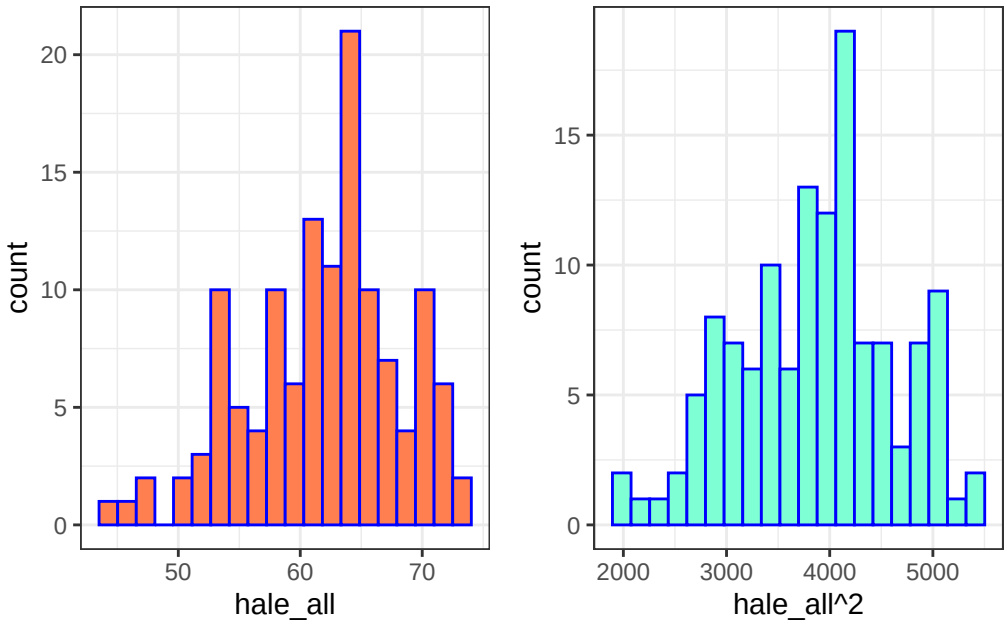
```
fit0 <- lm(
  hale_all ~ univ_care + wbank_income +
    hale_2000 + births_attend + ihr_core_cap +
    avg_tempc + female_22 + rural_22 + covid_vax100,
  data = ctry_si_train
)
```

```
boxCox(fit0)
```



Let's try the square of `hale_all` as our outcome.

```
p1 <- ggplot(ctry_si_train, aes(x = hale_all)) +
  geom_histogram(bins = 20, fill = "coral", col = "blue")

p2 <- ggplot(ctry_si_train, aes(x = hale_all^2)) +
  geom_histogram(bins = 20, fill = "aquamarine", col = "blue")

p1 + p2
```



Not much of a difference, but perhaps we will see more meaningful improvements in our regression residual plots.

```
ctry_si_train <- ctry_si_train |>
  mutate(hale_sqr = hale_all^2)

ctry_si_test <- ctry_si_test |>
  mutate(hale_sqr = hale_all^2)
```

22.5 Fitting and Evaluating in Training Data

22.5.1 Kitchen Sink Model

```
fit1 <- lm( hale_sqr ~
              univ_care + wbank_income + hale_2000 + births_attend +
              ihr_core_cap + avg_tempc + female_22 + rural_22 +
              covid_vax100,
  data = ctry_si_train
)
```

```
model_parameters(fit1)
```

```
Parameter                   | Coefficient |     SE |              95% CI | t(116) |      p
-------------------------------------------------------------------------------------------
(Intercept)                 |     -291.03 | 598.75 | [-1476.93,  894.86] |  -0.49 | 0.628
univ care [yes]             |       44.22 |  61.01 | [  -76.62,  165.05] |   0.72 | 0.470
wbank income [Low]          |      -28.88 | 140.59 | [ -307.33,  249.58] |  -0.21 | 0.838
wbank income [Lower-middle] |     -186.50 |  98.98 | [ -382.54,    9.54] |  -1.88 | 0.062
wbank income [Upper-middle] |     -284.94 |  74.59 | [ -432.67, -137.21] |  -3.82 | < .001
hale 2000                   |       59.94 |   5.05 | [   49.94,   69.94] |  11.87 | < .001
births attend               |        4.81 |   2.11 | [    0.64,    8.99] |   2.28 | 0.024
ihr core cap                |        4.65 |   1.87 | [    0.96,    8.35] |   2.49 | 0.014
avg tempc                   |       -5.48 |   3.75 | [  -12.91,    1.95] |  -1.46 | 0.147
female 22                   |       -2.05 |   9.07 | [  -20.01,   15.91] |  -0.23 | 0.822
rural 22                    |        1.67 |   1.62 | [   -1.54,    4.89] |   1.03 | 0.305
covid vax100                |        3.02 |   1.28 | [    0.49,    5.56] |   2.36 | 0.020


Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
  using a Wald t-distribution approximation.
```

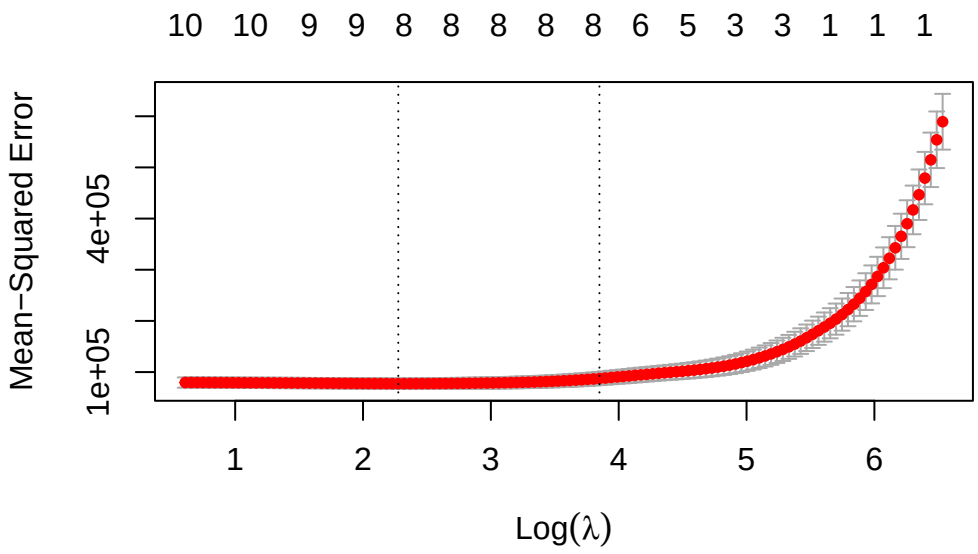
```
model_performance(fit1)
```

```
# Indices of model performance

AIC      |     AICc |      BIC |    R2 | R2 (adj.) |    RMSE |   Sigma
-----------------------------------------------------------------------
1807.365 | 1810.558 | 1844.441 | 0.888 |     0.878 | 254.555 | 267.398
```

22.5.2 LASSO to identify a subset

```
pred_x <- model.matrix(fit1)
out_y <- ctry_si_train |> select(hale_sqr) |> as.matrix()

set.seed(431)
lasso <- cv.glmnet(x = pred_x, y = out_y,
                   type.measure = "mse",
                   alpha = 1, family = "gaussian",
                   nlambda = 200, nfolds = 10)
```

```
plot(lasso)
```




```
log(lasso$lambda.min)
```

```
[1] 2.275831
```

```
predict(lasso, type = "coef", s = "lambda.min")
```

```
13 x 1 sparse Matrix of class "dgCMatrix"
                            lambda.min
(Intercept)              -222.093827
(Intercept)                 .
univ_careyes               29.607576
wbank_incomeLow             .
wbank_incomeLower-middle -129.930275
wbank_incomeUpper-middle -241.669217
hale_2000                  58.555358
births_attend               4.299195
ihr_core_cap                4.520508
avg_tempc                  -4.936356
female_22                   .
rural_22                    .
covid_vax100                2.959860
```

The LASSO suggestion is to drop two of the 9 variables in our original fit1 model, specifcally `female_22` and `rural_22`, since `wbank_income` is multi-categorical.

So that model would be:

```
fit7 <- lm( hale_sqr ~
              univ_care + wbank_income + hale_2000 + births_attend +
              ihr_core_cap + avg_tempc + covid_vax100,
  data = ctry_si_train
)
```

22.5.3 Stepwise Approach

```
fit6 <- select_parameters(fit1)

compare_models(fit1, fit6)
```

```
Parameter                   |                       fit1 |                       fit6
-------------------------------------------------------------------------------------
(Intercept)                 | -291.03 (-1476.93,  894.86) | -288.36 (-992.56,  415.84)
wbank income (Low)          |  -28.88 ( -307.33,  249.58) |    1.23 (-263.48,  265.94)
wbank income (Lower-middle) | -186.50 ( -382.54,    9.54) | -165.91 (-343.22,   11.40)
wbank income (Upper-middle) | -284.94 ( -432.67, -137.21) | -280.72 (-421.53, -139.91)
hale 2000                   |   59.94 (   49.94,   69.94) |   59.16 (  49.52,   68.80)
```

```
births attend               |    4.81 (    0.64,    8.99) |    4.96 (   0.83,    9.10)
ihr core cap                |    4.65 (    0.96,    8.35) |    4.58 (   1.08,    8.09)
avg tempc                   |   -5.48 (  -12.91,    1.95) |   -5.42 ( -11.86,    1.01)
covid vax100                |    3.02 (    0.49,    5.56) |    3.08 (   0.61,    5.55)
univ care (yes)             |   44.22 (  -76.62,  165.05) |
rural 22                    |    1.67 (   -1.54,    4.89) |
female 22                   |   -2.05 (  -20.01,   15.91) |
-------------------------------------------------------------------------------------
Observations                |                        128 |                       128
```

The stepwise approach drops three predictors, `univ_care`, `rural_22` and `female_22`.

22.5.4 Best Subsets

How about using the best subsets approach?

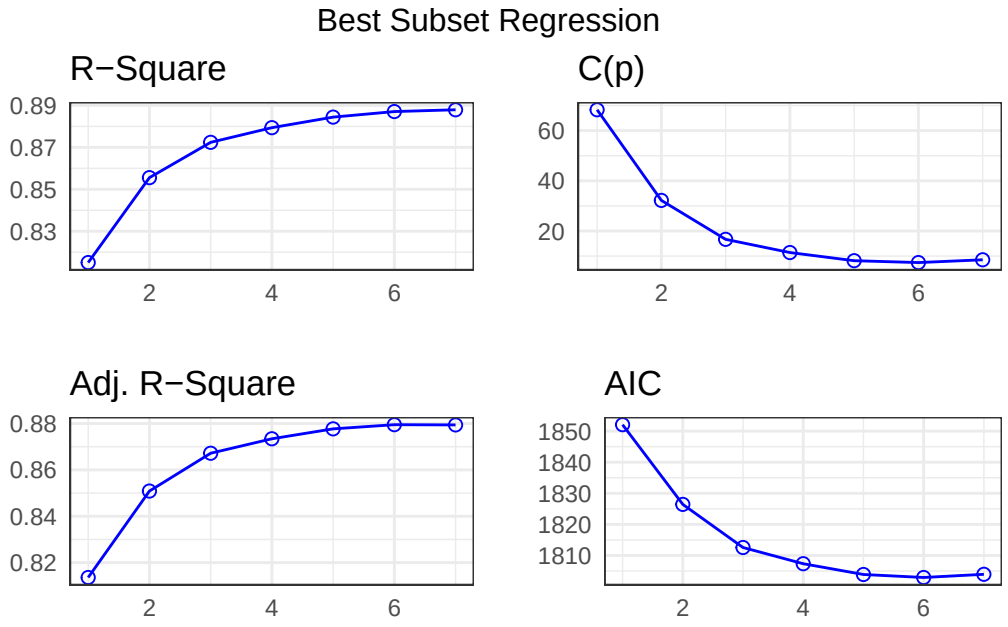
```
k <- ols_step_best_subset(fit1,
                          max_order = 7,
                          metric = "cp")
```

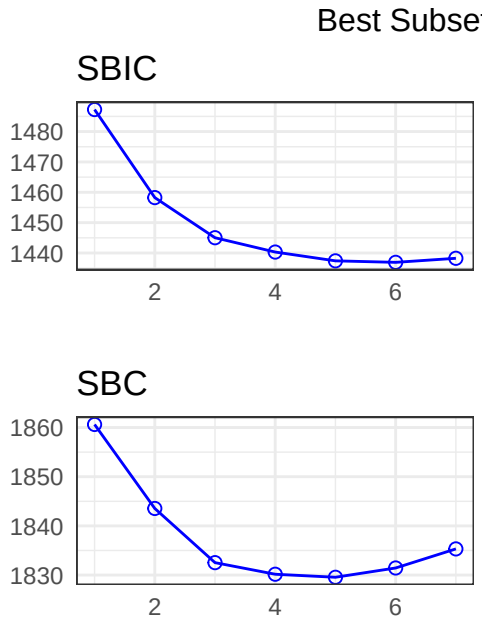
```
k
```

```
                                  Best Subsets Regression
-------------------------------------------------------------------------------------------
Model Index    Predictors
-------------------------------------------------------------------------------------------
     1         hale_2000
     2         wbank_income hale_2000
     3         wbank_income hale_2000 ihr_core_cap
     4         wbank_income hale_2000 births_attend ihr_core_cap
     5         wbank_income hale_2000 births_attend ihr_core_cap covid_vax100
     6         wbank_income hale_2000 births_attend ihr_core_cap avg_tempc covid_vax100
     7         wbank_income hale_2000 births_attend ihr_core_cap avg_tempc rural_22 covid_va
-------------------------------------------------------------------------------------------

                                                        Subsets Regression Summary
-------------------------------------------------------------------------------------------
                      Adj.        Pred
Model    R-Square    R-Square    R-Square     C(p)         AIC         SBIC          SBC
-------------------------------------------------------------------------------------------
  1        0.8151      0.8136      0.8073    68.3017    1852.0658    1487.2564    1860.6218
```

```
  2         0.8556       0.8509        0.842     32.1908     1826.4294     1458.2510     1843.5416
  3         0.8724       0.8672       0.8573     16.6786     1812.5604     1445.0276     1832.5246
  4         0.8794       0.8734       0.8632     11.4137     1807.3525     1440.3096     1830.1687
  5         0.8844       0.8777       0.8668      8.1680     1803.8834     1437.4087     1829.5516
  6         0.8871       0.8795       0.8675      7.4172     1802.9193     1436.9166     1831.4396
  7         0.8879       0.8794       0.8666      8.5288     1803.9471     1438.2545     1835.3194
--------------------------------------------------------------------------------------------------
AIC: Akaike Information Criteria
 SBIC: Sawa's Bayesian Information Criteria
 SBC: Schwarz Bayesian Criteria
 MSEP: Estimated error of prediction, assuming multivariate normality
 FPE: Final Prediction Error
 HSP: Hocking's Sp
 APC: Amemiya Prediction Criteria
```

```
plot(k)
```





The best subsets suggestions include

- the same six-predictor model we got with stepwise
- a five-predictor model with `avg_tempc` dropped from our 6-predictor model, and maybe
- a different seven-parameter model than we obtained from the LASSO (keeping `rural_22` but dropping `univ_care`)

Let's store the five-predictor model:

```
fit5 <- lm( hale_sqr ~
              wbank_income + hale_2000 + births_attend +
              ihr_core_cap + covid_vax100,
  data = ctry_si_train
)
```

22.5.5 Compare Candidate Models in Training Sample

So we have at least four candidate models: `fit1`, `fit5`, `fit6` and `fit7`.

```
compare_models(fit1, fit7, fit6, fit5)
```

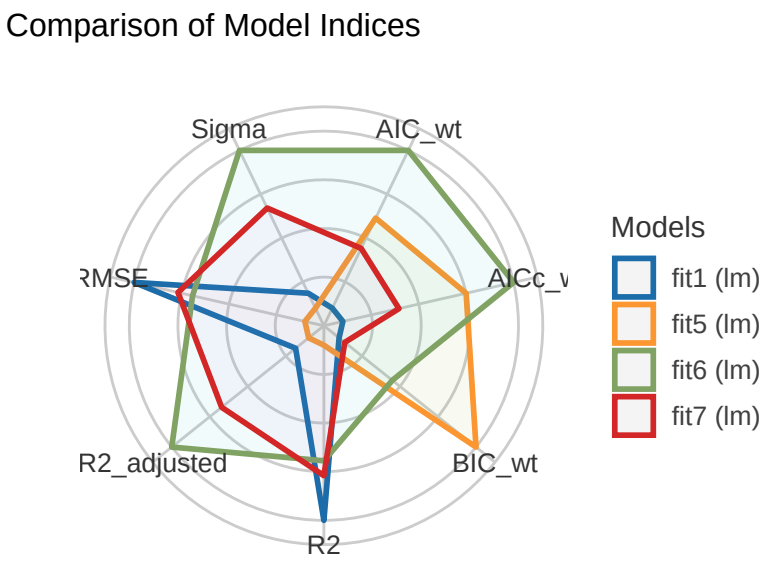
```
Parameter                   |                  fit1 |                  fit7 |
```

```
-------------------------------------------------------------------------------------------
(Intercept)                  | -291.03 (-1476.93,  894.86) | -263.05 (-974.18,  448.09) | -288
wbank income (Low)           |  -28.88 ( -307.33,  249.58) |    1.60 (-263.85,  267.06) |
wbank income (Lower-middle)  | -186.50 ( -382.54,    9.54) | -157.94 (-337.70,   21.81) | -165
wbank income (Upper-middle)  | -284.94 ( -432.67, -137.21) | -279.75 (-420.99, -138.52) | -280
hale 2000                    |   59.94 (   49.94,   69.94) |   58.86 (  49.13,   68.58) |   59
births attend                |    4.81 (    0.64,    8.99) |    4.87 (   0.71,    9.02) |
ihr core cap                 |    4.65 (    0.96,    8.35) |    4.42 (   0.87,    7.98) |
covid vax100                 |    3.02 (    0.49,    5.56) |    3.00 (   0.50,    5.49) |
univ care (yes)              |   44.22 (  -76.62,  165.05) |   35.15 ( -81.32,  151.62) |
female 22                    |   -2.05 (  -20.01,   15.91) |                            |
avg tempc                    |   -5.48 (  -12.91,    1.95) |   -5.35 ( -11.81,    1.10) |   -5
rural 22                     |    1.67 (   -1.54,    4.89) |                            |
-------------------------------------------------------------------------------------------
Observations                 |                        128 |                        128 |
```

```
compare_performance(fit1, fit7, fit6, fit5)
```

```
# Comparison of Model Performance Indices

Name | Model |  AIC (weights) | AICc (weights) |  BIC (weights) |    R2 | R2 (adj.) |    RMSE
-----------------------------------------------------------------------------------------------
fit1 |    lm | 1807.4 (0.050) | 1810.6 (0.026) | 1844.4 (<.001) | 0.888 |     0.878 | 254.555
fit7 |    lm | 1804.5 (0.206) | 1806.8 (0.170) | 1835.9 (0.029) | 0.887 |     0.879 | 255.719
fit6 |    lm | 1802.9 (0.460) | 1804.8 (0.463) | 1831.4 (0.272) | 0.887 |     0.879 | 256.106
fit5 |    lm | 1803.9 (0.284) | 1805.4 (0.341) | 1829.6 (0.699) | 0.884 |     0.878 | 259.088
```

```
plot(compare_performance(fit1, fit7, fit6, fit5))
```



It looks like the six-predictor model is the best choice (but just barely) using adjusted R^2, σ and AIC. Let's look at the test sample.

22.5.6 Compare Candidate Models in Test Sample

```
test1 <- augment(fit1, newdata = ctry_si_test) |>
  mutate(mod_n = "Model 1 (KS)")
test7 <- augment(fit7, newdata = ctry_si_test) |>
  mutate(mod_n = "Model 7")
test6 <- augment(fit6, newdata = ctry_si_test) |>
  mutate(mod_n = "Model 6")
test5 <- augment(fit5, newdata = ctry_si_test) |>
  mutate(mod_n = "Model 5")

test_res <- bind_rows(test1, test7, test6, test5) |>
  mutate(predicted_haleall = sqrt(.fitted),
         resid_haleall = hale_all - predicted_haleall) |>
  select(mod_n, country_name, hale_all,
         predicted_haleall, resid_haleall, everything()) |>
  arrange(c_num, mod_n)
```

```
head(test_res, 8)
```

```
# A tibble: 8 x 21
  mod_n    country_name hale_all predicted_haleall resid_haleall c_num univ_care
  <chr>    <chr>           <dbl>             <dbl>         <dbl> <chr> <fct>
1 Model 1~ Luxembourg       71.2              68.6          2.55 100   yes
2 Model 5  Luxembourg       71.2              68.5          2.68 100   yes
3 Model 6  Luxembourg       71.2              68.7          2.49 100   yes
4 Model 7  Luxembourg       71.2              68.8          2.39 100   yes
5 Model 1~ Maldives         66.7              63.8          2.90 104   yes
6 Model 5  Maldives         66.7              63.6          3.15 104   yes
7 Model 6  Maldives         66.7              63.2          3.53 104   yes
8 Model 7  Maldives         66.7              63.3          3.38 104   yes
# i 14 more variables: wbank_income <fct>, hale_2000 <dbl>,
#   births_attend <dbl>, ihr_core_cap <dbl>, avg_tempc <dbl>, female_22 <dbl>,
#   rural_22 <dbl>, covid_vax100 <dbl>, who_region <fct>, iso_alpha3 <chr>,
#   .row_id <int>, hale_sqr <dbl>, .fitted <dbl>, .resid <dbl>
```

```
test_res |>
    group_by(mod_n) |>
    summarise(MAPE = mean(abs(resid_haleall)),
              medAPE = median(abs(resid_haleall)),
              maxAPE = max(abs(resid_haleall)),
              RMSPE = sqrt(mean(resid_haleall^2)),
              rsqr = cor(hale_all, predicted_haleall)^2) |>
  kable()
```

mod_n	MAPE	medAPE	maxAPE	RMSPE	rsqr
Model 1 (KS)	1.399561	1.172160	4.242786	1.682548	0.9063445
Model 5	1.387650	1.091875	3.818286	1.700453	0.9009602
Model 6	1.357642	1.198238	4.019098	1.660509	0.9072666
Model 7	1.347411	1.168969	3.973297	1.637438	0.9092709

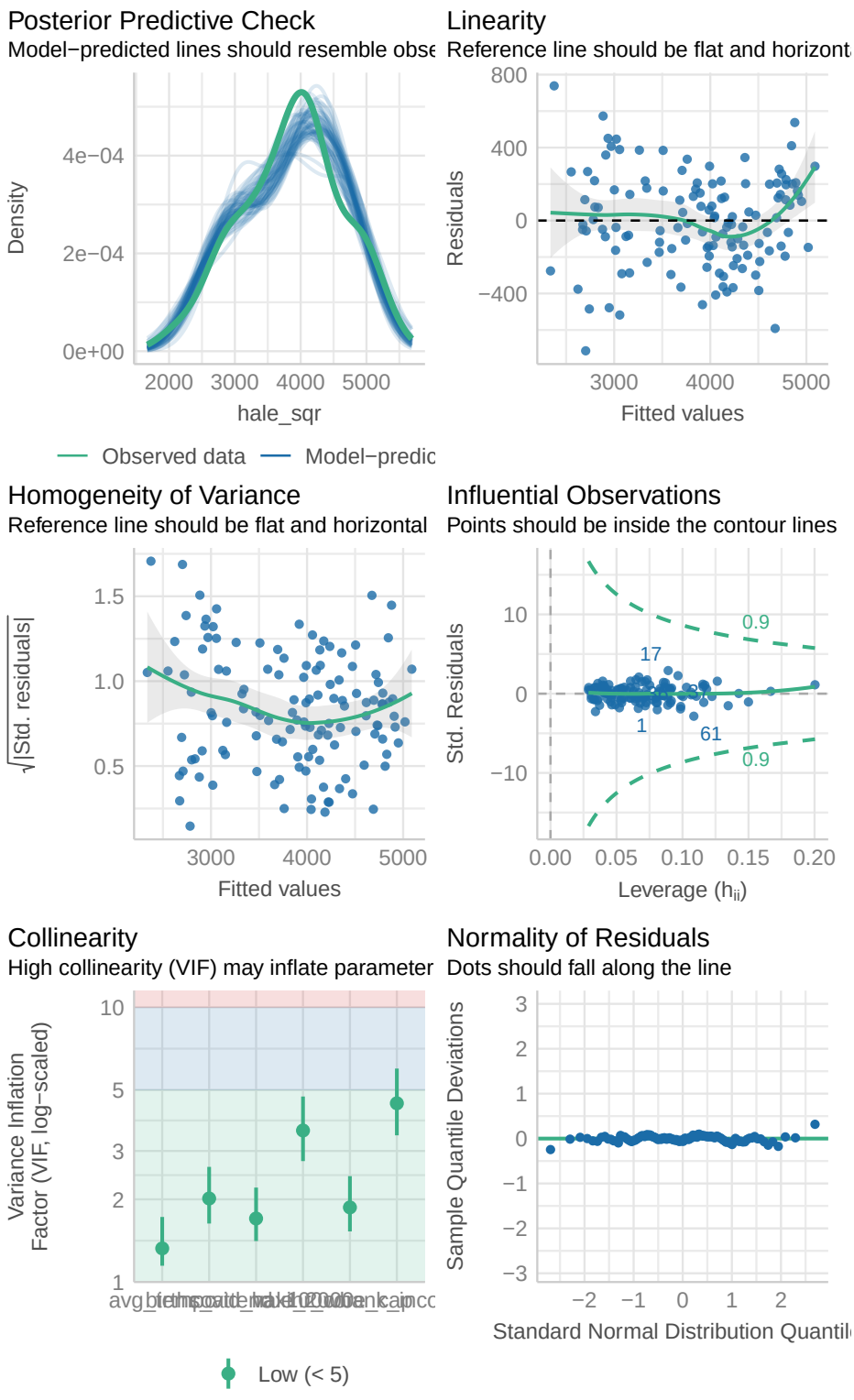
We see that:

- Model 6 has the smallest mean, median and maximum average prediction error, but
- Model 7 has the smallest RMSPE and largest validated R^2.

I'll select model 6, assuming that model assumptions aren't too badly violated in our training sample. Let's check that.

22.6 Check Assumptions in Winning Model

```
check_model(fit6)
```


Posterior Predictive Check
Model−predicted lines should resemble obse
Density
4e−04
2e−04
0e+00
2000
3000
4000
5000
hale_sqr
Observed data
Model−predic
Linearity
Reference line should be flat and horizont
Residuals
800
400
0
−400
3000
4000
5000
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
1.5
1.0
0.5
3000
4000
5000
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
10
0
−10
0.9
17
1
61
0.9
0.00
0.05
0.10
0.15
0.20
Leverage (hii)
Collinearity
High collinearity (VIF) may inflate parameter
Variance Inflation
Factor (VIF, log−scaled)
10
5
3
2
1
Low (< 5)
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
3
2
1
0
−1
−2
−3
−2
−1
0
1
2
Standard Normal Distribution Quantil

The collinearity is OK now, and outside of a couple of observations, I think Normality isn't a serious problem.

22.7 Estimates for the Winning Model

So now, I'll refit the six-predictor model, which includes:

- `wbank_income`
- `hale_2000`
- `births_attend`
- `ihr_core_cap`
- `avg_tempc`, and
- `covid_vax100`

to the entire set of data, with various assumptions about the missing data.

22.7.1 Complete Case Analysis

Here, we'll fit the model to all of the data except (as we'll do in all three cases) the countries without outcome (`hale_all`) information.

```
fit_cc <- lm((hale_all^2) ~ wbank_income + hale_2000 +
               births_attend + ihr_core_cap +
               avg_tempc + covid_vax100,
             data = ctry_cc)

model_parameters(fit_cc, ci = 0.95)
```

```
Parameter                   | Coefficient |     SE |             95% CI | t(155) |      p
-------------------------------------------------------------------------------------------
(Intercept)                 |      -93.51 | 302.42 | [-690.90,  503.89] |  -0.31 | 0.758
wbank income [Low]          |      -68.39 | 112.63 | [-290.88,  154.11] |  -0.61 | 0.545
wbank income [Lower-middle] |     -148.19 |  73.52 | [-293.42,   -2.97] |  -2.02 | 0.046
wbank income [Upper-middle] |     -267.91 |  60.30 | [-387.02, -148.81] |  -4.44 | < .001
hale 2000                   |       57.44 |   4.20 | [  49.14,   65.74] |  13.67 | < .001
births attend               |        4.84 |   1.79 | [   1.31,    8.37] |   2.71 | 0.008
ihr core cap                |        3.80 |   1.51 | [   0.81,    6.79] |   2.51 | 0.013
avg tempc                   |       -5.51 |   2.80 | [ -11.05,    0.03] |  -1.97 | 0.051
covid vax100                |        2.65 |   1.09 | [   0.49,    4.81] |   2.42 | 0.017
```

```
Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
  using a Wald t-distribution approximation.
```

```
model_performance(fit_cc)
```

```
# Indices of model performance

AIC       |      AICc |       BIC |    R2 | R2 (adj.) |    RMSE |   Sigma
-------------------------------------------------------------------------
44110.965 | 44112.403 | 44141.963 | 0.881 |     0.874 | 247.017 | 254.088
```

22.7.2 After Single Imputation

```
fit_si <- lm((hale_all^2) ~ wbank_income + hale_2000 +
               births_attend + ihr_core_cap +
               avg_tempc + covid_vax100,
             data = ctry_si)

model_parameters(fit_si, ci = 0.95)
```

```
Parameter                   | Coefficient |     SE |             95% CI | t(174) |      p
-------------------------------------------------------------------------------------------
(Intercept)                 |      -49.82 | 270.59 | [-583.88,  484.25] |  -0.18 | 0.854
wbank income [Low]          |      -79.91 |  97.08 | [-271.51,  111.69] |  -0.82 | 0.412
wbank income [Lower-middle] |     -178.92 |  65.89 | [-308.97,  -48.86] |  -2.72 | 0.007
wbank income [Upper-middle] |     -293.54 |  56.08 | [-404.22, -182.87] |  -5.23 | < .001
hale 2000                   |       57.96 |   3.67 | [  50.72,   65.20] |  15.80 | < .001
births attend               |        4.50 |   1.63 | [   1.29,    7.72] |   2.76 | 0.006
ihr core cap                |        3.75 |   1.41 | [   0.97,    6.53] |   2.66 | 0.008
avg tempc                   |       -6.47 |   2.60 | [ -11.60,   -1.35] |  -2.49 | 0.014
covid vax100                |        2.64 |   0.98 | [   0.70,    4.57] |   2.69 | 0.008

Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
  using a Wald t-distribution approximation.
```

```
model_performance(fit_si)
```

```
# Indices of model performance

AIC       |      AICc |       BIC |    R2 | R2 (adj.) |    RMSE |   Sigma
-------------------------------------------------------------------------
52919.799 | 52921.078 | 52951.894 | 0.892 |     0.887 | 239.530 | 245.647
```

22.7.3 Multiple Imputation

As mentioned earlier, I am dropping the observations with no information on my outcome (`hale_all`) which is the `ctry_cc` data. Then I am building 15 imputations, because I am missing data on just over 10% of cases within `ctry_cc`.

```
pct_miss_case(ctry_cc) # as a reminder
```

```
[1] 10.38251
```

Now, we fit our six-predictor model in each of these 15 imputed data sets, like this:

```
imp_ests <- ctry_cc |>
  mice(m = 15, seed = 431, print = FALSE) |>
  with(lm((hale_all)^2 ~ wbank_income + hale_2000 +
              births_attend + ihr_core_cap +
              avg_tempc + covid_vax100)) |>
  pool()
```

```
Warning: Number of logged events: 3
```

```
model_parameters(imp_ests, ci = 0.95)
```

```
Warning: Number of logged events: 3
Warning: Number of logged events: 3
Warning: Number of logged events: 3
Warning: Number of logged events: 3
```

```
# Fixed Effects

Parameter                    | Coefficient |     SE |             95% CI |     t |     df |
-------------------------------------------------------------------------------------------
(Intercept)                  |      -66.57 | 274.03 | [-607.55,  474.42] | -0.24 | 168.60 | 0
wbank income [Low]           |      -74.63 |  97.66 | [-267.42,  118.15] | -0.76 | 169.38 | 0
wbank income [Lower-middle]  |     -172.29 |  66.57 | [-303.70,  -40.88] | -2.59 | 170.49 | 0
wbank income [Upper-middle]  |     -294.49 |  56.44 | [-405.89, -183.09] | -5.22 | 171.96 | <
hale 2000                    |       58.20 |   3.69 | [  50.91,   65.49] | 15.76 | 169.20 | <
births attend                |        4.50 |   1.66 | [   1.23,    7.77] |  2.72 | 166.98 | 0
ihr core cap                 |        3.70 |   1.42 | [   0.91,    6.50] |  2.62 | 171.40 | 0
avg tempc                    |       -6.47 |   2.61 | [ -11.62,   -1.32] | -2.48 | 171.81 | 0
covid vax100                 |        2.70 |   1.01 | [   0.72,    4.68] |  2.69 | 171.28 | 0


Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
  using a Wald t-distribution approximation.
```

```
glance(imp_ests)
```

```
  nimp nobs r.squared adj.r.squared
1   15  183 0.8915267     0.8865394
```

and hopefully show them all to be fairly similar. Then describe that winning fit in some detail, probably in the multiple imputation case.

22.8 Bayesian Linear Fit

Show Bayesian fit for the winning model across the entire sample, and draw conclusions about its fit and form.

```
ctry_si <- ctry_si |>
  mutate(hale_sqr = hale_all^2)

fit_bay <- stan_glm(hale_sqr ~
                      wbank_income + hale_2000 +
                      births_attend + ihr_core_cap +
                      avg_tempc + covid_vax100,
                    data = ctry_si, refresh = 0)

model_parameters(fit_bay, ci = 0.95)
```

```
Parameter                |  Median |            95% CI |     pd |  Rhat |     ESS |
-------------------------------------------------------------------------------------
(Intercept)              |  -58.22 | [-593.32,  491.26] | 58.45% | 1.000 | 2586.00 | Normal
wbank_incomeLow          |  -79.75 | [-272.32,  116.58] | 78.38% | 1.001 | 2311.00 |    Norma
wbank_incomeLower-middle | -177.89 | [-309.74,  -47.08] | 99.50% | 1.000 | 2335.00 |    Norma
wbank_incomeUpper-middle | -292.57 | [-409.28, -179.50] |   100% | 1.001 | 3126.00 |    Norma
hale_2000                |   58.05 | [  50.54,   64.89] |   100% | 1.001 | 3029.00 |     Norm
births_attend            |    4.54 | [   1.29,    7.67] | 99.72% | 1.000 | 4459.00 |     Norm
ihr_core_cap             |    3.74 | [   0.99,    6.54] | 99.55% | 1.000 | 4377.00 |     Norm
avg_tempc                |   -6.43 | [ -11.52,   -1.41] | 99.33% | 1.000 | 4533.00 |     Norm
covid_vax100             |    2.60 | [   0.68,    4.50] | 99.62% | 1.000 | 4575.00 |      No

Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
  using a MCMC distribution approximation.
```

```
model_performance(fit_bay)
```

```
# Indices of model performance

ELPD      | ELPD_SE |    LOOIC | LOOIC_SE |     WAIC |    R2 | R2 (adj.) |    RMSE |   Sigma
-------------------------------------------------------------------------------------------
-1272.992 |  10.426 | 2545.984 |   20.853 | 2545.895 | 0.887 |     0.880 | 239.536 | 246.897
```

22.9 For More Information

point to https://r4ds.hadley.nz/missing-values

23 NNYFS Case Study

> ! This is a DRAFT version of this Chapter.
>
> This is a sketchy draft. I'll remove this notice when I post a version of this Chapter that is essentially finished.

23.1 R setup for this chapter

> i Note
>
> This section loads all needed R packages for this chapter. Appendix A lists all R packages and data sets used in the book, and also provides R session information. Appendix B describes the 431-Love.R script, and demonstrates its use.

```
library(knitr)
library(mice)
library(naniar)
library(rstanarm)

library(easystats)
library(tidyverse)

source("data/Love-431.R")
theme_set(theme_bw())
```

23.2 Data should be NNYFS

> **Note**
>
> Appendix C provides further guidance on pulling data from other systems into R, while Appendix D gives more information (including download links) for all data sets used in this book.

```
nnyfs <- read_rds("data/nnyfs.Rds")

miss_var_summary(nnyfs)
```

```
# A tibble: 45 x 3
   variable             n_miss pct_miss
   <chr>                 <int>    <num>
 1 educ_child              337   22.2
 2 enjoy_recess            278   18.3
 3 plank_time              134    8.83
 4 calf_skinfold           128    8.43
 5 income_pov               89    5.86
 6 subscapular_skinfold     67    4.41
 7 educ_adult               22    1.45
 8 triceps_skinfold         21    1.38
 9 salt_used                13    0.856
10 respondent               12    0.791
# i 35 more rows
```

23.3 For More Information

A R Packages

A.1 R Packages used in this book

Here is a list of the R packages that are used in this book.

Chapters	Package	Description	Reference(s)
17-18, 22	**broom**	Summarize key information about models into tibbles	Website
7, 10, 17-18, 20-22	**car**	Box-Cox transformations, Companion to Applied Regression	PDF manual, related book
13	**Epi**	twoby2(), Epidemiological Analysis with R	Website
Appendix	**fivethirtyeight**	bechdel data set	Website
11, 18	**GGally**	scatterplot matrix through ggpairs()	Website
2, 6-7, 9-10	**ggdist**	Rain cloud plots, Visualizations of distributions and uncertainty	Website
2, 11	**ggpubr**	add fitted equation to scatterplot	Website
21-22	**glmnet**	LASSO and Elastic Net models	Website
2, 11-12	**glue**	working with strings and R results	Website
8, 12, 16, 21	**haven**	read and write various data formats	Website
5-7	**infer**	tidy statistical inference	Website
2-4, 7, 9-12, 14, 17, 20-22	**janitor**	clean_names(), tabyl(), for cleaning and exploring	Overview
2-14, 16-23	**knitr**	kable(), report generation	Website
13	**medicaldata**	data sets for teaching reproducible medical research	Website
17, 22-23	**mice**	Multivariate Imputation by Chained Equations	Website, van Buuren (2021)
6-7	**MKinfer**	bootstrap testing	Website
2-3, 5, 7-14, 16-23	**naniar**	Tidy ways to summarize missingness	Website
20-22	**olsrr**	building OLS models, including best subsets	Website

Chapters	Package	Description	Reference(s)
2-3	**palmerpenguins**	Palmer Penguins data	Website, Horst, Hill, and Gorman (2020)
2, 5-7, 10-12, 16-17, 21-22	**patchwork**	Combining separate ggplots into one graphic	Website
6	**readxl**	get data out of Excel and into R	Website
5-7, 9-12, 16-23	**rstanarm**	Bayesian Applied Regression Modeling via STAN	Website
all	**styler**	non-invasive code formatting for pretty printing	Website Tidyverse Style Guide
18	**tidytuesdayR**	access data from Tidy Tuesday repository	Website
Appendix	**xfun**	for session information, Section A.3	Website

A.1.1 Meta-packages

Chapters	Meta-Package	Description	Reference(s)
2-14, 16-23	**easystats**	An R Framework for Easy Statistical Modeling, Visualization and Reporting	Website, Lüdecke et al. (2022)
2-14, 16-23	**tidyverse**	R packages for Data Science	Website

A.2 Loading all of these R packages

```
knitr::opts_chunk$set(comment = NA)

library(broom)
library(car)
library(emmeans)
library(Epi)
library(fivethirtyeight)
library(GGally)
library(ggdist)
library(ggpubr)
```

```
library(glue)
library(haven)
library(infer)
library(janitor)
library(knitr)
library(medicaldata)
library(mice)
library(MKinfer)
library(naniar)
library(olsrr)
library(palmerpenguins)
library(patchwork)
library(readxl)
library(rstanarm)
library(styler)
library(tidytuesdayR)
library(xfun)

library(easystats)
library(tidyverse)
```

A.3 Session Information

```
xfun::session_info()
```

```
R version 4.4.1 (2024-06-14 ucrt)
Platform: x86_64-w64-mingw32/x64
Running under: Windows 11 x64 (build 22631)

Locale:
  LC_COLLATE=English_United States.utf8
  LC_CTYPE=English_United States.utf8
  LC_MONETARY=English_United States.utf8
  LC_NUMERIC=C
  LC_TIME=English_United States.utf8

Package version:
  abind_1.4-5            arrangements_1.1.9     askpass_1.2.0
  backports_1.5.0        base64enc_0.1-3        bayesplot_1.11.1
```

bayestestR_0.14.0 BH_1.84.0.0 bit_4.0.5
bit64_4.0.5 blob_1.2.4 boot_1.3-31
broom_1.0.6 broom.helpers_1.17.0 bslib_0.8.0
cachem_1.1.0 callr_3.7.6 car_3.1-2
carData_3.0-5 cards_0.2.2 cellranger_1.1.0
checkmate_2.3.2 cli_3.6.3 clipr_0.8.0
cmprsk_2.2-12 coda_0.19-4.1 codetools_0.2-20
colorspace_2.1-1 colourpicker_1.3.0 commonmark_1.9.1
compiler_4.4.1 conflicted_1.2.0 correlation_0.8.5
corrplot_0.94 cowplot_1.1.3 cpp11_0.5.0
crayon_1.5.3 credentials_2.0.1 crosstalk_1.2.1
curl_5.2.2 data.table_1.16.0 datasets_4.4.1
datawizard_0.12.3 DBI_1.2.3 dbplyr_2.5.0
Deriv_4.1.3 desc_1.4.3 digest_0.6.37
distributional_0.4.0 doBy_4.6.22 dplyr_1.1.4
DT_0.33 dtplyr_1.3.1 dygraphs_1.1.1.6
easystats_0.7.3 effectsize_0.8.9 emmeans_1.10.4
Epi_2.53 estimability_1.5.1 etm_1.1.1
evaluate_0.24.0 exactRankTests_0.8-35 fansi_1.0.6
farver_2.1.2 fastmap_1.2.0 fivethirtyeight_0.6.2
fontawesome_0.5.2 forcats_1.0.0 foreach_1.5.2
fs_1.6.4 gargle_1.5.2 generics_0.1.3
gert_2.1.1 GGally_2.2.1 ggdist_3.3.2
ggplot2_3.5.1 ggpubr_0.6.0 ggrepel_0.9.6
ggridges_0.5.6 ggsci_3.2.0 ggsignif_0.6.4
ggstats_0.6.0 gh_1.4.1 gitcreds_0.1.2
glmnet_4.1-8 glue_1.7.0 gmp_0.7-5
goftest_1.2-3 googledrive_2.1.1 googlesheets4_1.1.1
graphics_4.4.1 grDevices_4.4.1 grid_4.4.1
gridExtra_2.3 gtable_0.3.5 gtools_3.9.5
haven_2.5.4 highr_0.11 hms_1.1.3
htmltools_0.5.8.1 htmlwidgets_1.6.4 httpuv_1.6.15
httr_1.4.7 httr2_1.0.3 ids_1.0.1
igraph_2.0.3 infer_1.0.7 ini_0.3.1
inline_0.3.19 insight_0.20.4 isoband_0.2.7
iterators_1.0.14 janitor_2.2.0 jomo_2.7-6
jquerylib_0.1.4 jsonlite_1.8.8 knitr_1.48
labeling_0.4.3 labelled_2.13.0 later_1.3.2
lattice_0.22-6 lazyeval_0.2.2 lifecycle_1.0.4
lme4_1.1-35.5 loo_2.8.0 lubridate_1.9.3
magrittr_2.0.3 markdown_1.13 MASS_7.3-61
Matrix_1.7-0 MatrixModels_0.5.3 matrixStats_1.4.0
medicaldata_0.2.0 memoise_2.0.1 methods_4.4.1

mgcv_1.9-1	mice_3.16.0	miceadds_3.17-44
microbenchmark_1.5.0	mime_0.12	miniUI_0.1.1.1
minqa_1.2.8	mitml_0.4-5	mitools_2.4
MKdescr_0.8	MKinfer_1.2	modelbased_0.8.8
modelr_0.1.11	multcomp_1.4-26	munsell_0.5.1
mvtnorm_1.3-1	naniar_1.1.0	nlme_3.1-164
nloptr_2.1.1	nnet_7.3-19	norm_1.0.11.1
nortest_1.0-4	numDeriv_2016.8-1.1	olsrr_0.6.0
openssl_2.2.1	ordinal_2023.12.4.1	palmerpenguins_0.1.1
pan_1.9	parallel_4.4.1	parameters_0.22.2
patchwork_1.2.0	pbkrtest_0.5.3	performance_0.12.3
pillar_1.9.0	pkgbuild_1.4.4	pkgconfig_2.0.3
plyr_1.8.9	polynom_1.4.1	posterior_1.6.0
prettyunits_1.2.0	processx_3.8.4	progress_1.2.3
promises_1.3.0	ps_1.7.7	purrr_1.0.2
quadprog_1.5.8	quantreg_5.98	QuickJSR_1.3.1
R.cache_0.16.0	R.methodsS3_1.8.2	R.oo_1.26.0
R.utils_2.12.3	R6_2.5.1	ragg_1.3.2
rappdirs_0.3.3	RColorBrewer_1.1-3	Rcpp_1.0.13
RcppArmadillo_14.0.0.1	RcppEigen_0.3.4.0.2	RcppParallel_5.1.9
readr_2.1.5	readxl_1.4.3	rematch_2.0.0
rematch2_2.1.2	report_0.5.9	reprex_2.1.1
reshape2_1.4.4	rlang_1.1.4	rmarkdown_2.28
rpart_4.1.23	rprojroot_2.0.4	rstan_2.32.6
rstanarm_2.32.1	rstantools_2.4.0	rstatix_0.7.2
rstudioapi_0.16.0	rvest_1.0.4	sandwich_3.1-0
sass_0.4.9	scales_1.3.0	see_0.9.0
selectr_0.4.2	shape_1.4.6.1	shiny_1.9.1
shinyjs_2.1.0	shinystan_2.6.0	shinythemes_1.2.0
snakecase_0.11.1	sourcetools_0.1.7.1	SparseM_1.84.2
splines_4.4.1	StanHeaders_2.32.10	stats_4.4.1
stats4_4.4.1	stringi_1.8.4	stringr_1.5.1
styler_1.10.3	survival_3.7-0	sys_3.4.2
systemfonts_1.1.0	tensorA_0.36.2.1	textshaping_0.4.0
TH.data_1.1-2	threejs_0.3.3	tibble_3.2.1
tidyr_1.3.1	tidyselect_1.2.1	tidytuesdayR_1.1.2
tidyverse_2.0.0	timechange_0.3.0	tinytex_0.52
tools_4.4.1	tzdb_0.4.0	ucminf_1.2.2
UpSetR_1.4.0	usethis_3.0.0	utf8_1.2.4
utils_4.4.1	uuid_1.2.1	V8_5.0.0
vctrs_0.6.5	viridis_0.6.5	viridisLite_0.4.2
visdat_0.6.0	vroom_1.6.5	whisker_0.4.1
withr_3.0.1	xfun_0.47	xml2_1.3.6

```
xplorerr_0.1.2          xtable_1.8-4            xts_0.14.0
yaml_2.3.10             zip_2.3.1               zoo_1.8-12
```

B The Love-431.R script

B.1 R setup for this appendix

> **Note**
>
> Appendix A lists all R packages and data sets used in the book, and also provides R session information.

```
library(Epi)
library(palmerpenguins)

library(easystats)
library(tidyverse)
```

B.2 Contents of `Love-431.R` script

B.3 The `lovedist()` function

```
lovedist <- function(x) {
  tibble::tibble(
    n = length(x),
    miss = sum(is.na(x)),
    mean = mean(x, na.rm = TRUE),
    sd = sd(x, na.rm = TRUE),
    med = median(x, na.rm = TRUE),
    mad = mad(x, na.rm = TRUE),
    min = min(x, na.rm = TRUE),
    q25 = quantile(x, 0.25, na.rm = TRUE),
    q75 = quantile(x, 0.75, na.rm = TRUE),
    max = max(x, na.rm = TRUE),
  )
}
```

B.3.1 Sample Use

```
palmerpenguins::penguins |>
  reframe(lovedist(bill_depth_mm))
```

```
# A tibble: 1 x 10
      n  miss  mean    sd   med   mad   min   q25   q75   max
  <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1   344     2  17.2  1.97  17.3  2.22  13.1  15.6  18.7  21.5
```

```
palmerpenguins::penguins |>
  reframe(lovedist(bill_depth_mm), .by = species)
```

```
# A tibble: 3 x 11
  species       n  miss  mean    sd   med   mad   min   q25   q75   max
  <fct>     <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 Adelie      152     1  18.3 1.22   18.4  1.19  15.5  17.5  19    21.5
2 Gentoo      124     1  15.0 0.981  15    1.19  13.1  14.2  15.7  17.3
3 Chinstrap    68     0  18.4 1.14   18.4  1.41  16.4  17.5  19.4  20.8
```

B.4 The `twobytwo` function

```
twobytwo <-
  function(a, b, c, d,
           namer1 = "Row1", namer2 = "Row2",
           namec1 = "Col1", namec2 = "Col2",
           conf.level = 0.95)
           # build 2 by 2 table and run Epi library's twoby2 command to summarize
           # from the row-by-row counts in a cross-tab
           # upper left cell is a, upper right is b,
           # lower left is c, lower right is d
           # names are then given in order down the rows then across the columns
           # use standard epidemiological format:
  # outcomes in columns, treatments in rows
  {
    .Table <- matrix(c(a, b, c, d), 2, 2,
      byrow = T,
      dimnames = list(
```

```
        c(namer1, namer2),
        c(namec1, namec2)
      )
    )
    Epi::twoby2(.Table, alpha = 1 - conf.level)
  }
```

B.4.1 Sample Use

```
twobytwo(23, 14, 12, 9, "Treatment 1", "Treatment 2", "Out 1", "Out 2",
  conf.level = 0.90
)
```

```
2 by 2 table analysis:
------------------------------------------------------
Outcome   : Out 1
Comparing : Treatment 1 vs. Treatment 2

            Out 1 Out 2    P(Out 1) 90% conf. interval
Treatment 1    23    14      0.6216    0.4847   0.7415
Treatment 2    12     9      0.5714    0.3923   0.7336

                                    90% conf. interval
             Relative Risk: 1.0878    0.7472   1.5839
         Sample Odds Ratio: 1.2321    0.4936   3.0759
Conditional MLE Odds Ratio: 1.2277    0.4265   3.5045
    Probability difference: 0.0502   -0.1587   0.2620

             Exact P-value: 0.7834
        Asymptotic P-value: 0.7074
------------------------------------------------------
```

B.5 The `saifs_ci()` function

```
saifs_ci <-
  function(x, n, conf.level = 0.95, dig = 3) {
    p.sample <- round(x / n, digits = dig)
```

```
    p1 <- x / (n + 1)
    p2 <- (x + 1) / (n + 1)

    var1 <- (p1 * (1 - p1)) / n
    se1 <- sqrt(var1)
    var2 <- (p2 * (1 - p2)) / n
    se2 <- sqrt(var2)

    lowq <- (1 - conf.level) / 2
    tcut <- qt(lowq, df = n - 1, lower.tail = FALSE)

    lower.bound <- round(p1 - tcut * se1, digits = dig)
    upper.bound <- round(p2 + tcut * se2, digits = dig)
    tibble(
      sample_x = x,
      sample_n = n,
      sample_p = p.sample,
      lower = lower.bound,
      upper = upper.bound,
      conf_level = conf.level
    )
  }
```

B.5.1 Sample Usage

```
saifs_ci(x = 19, n = 25, conf.level = 0.95)
```

```
# A tibble: 1 x 6
  sample_x sample_n sample_p lower upper conf_level
     <dbl>    <dbl>    <dbl> <dbl> <dbl>      <dbl>
1       19       25     0.76 0.548 0.943       0.95
```

C Getting Data Into R

C.1 Using data from an R package

To use data from an R package, for instance, the `bechdel` data from the `fivethirtyeight` package, you can simply load the relevant package with `library` and then the data frame will be available

```
library(fivethirtyeight)
library(tidyverse)

bechdel
```

```
# A tibble: 1,794 x 15
    year imdb      title  test  clean_test binary budget domgross intgross code
   <int> <chr>     <chr>  <chr> <ord>      <chr>   <int>    <dbl>    <dbl> <chr>
 1  2013 tt1711425 21 & ~ nota~ notalk     FAIL   1.3 e7 25682380   4.22e7 2013~
 2  2012 tt1343727 Dredd~ ok-d~ ok         PASS   4.50e7 13414714   4.09e7 2012~
 3  2013 tt2024544 12 Ye~ nota~ notalk     FAIL   2   e7 53107035   1.59e8 2013~
 4  2013 tt1272878 2 Guns nota~ notalk     FAIL   6.1 e7 75612460   1.32e8 2013~
 5  2013 tt0453562 42     men   men        FAIL   4   e7 95020213   9.50e7 2013~
 6  2013 tt1335975 47 Ro~ men   men        FAIL   2.25e8 38362475   1.46e8 2013~
 7  2013 tt1606378 A Goo~ nota~ notalk     FAIL   9.2 e7 67349198   3.04e8 2013~
 8  2013 tt2194499 About~ ok-d~ ok         PASS   1.20e7 15323921   8.73e7 2013~
 9  2013 tt1814621 Admis~ ok    ok         PASS   1.3 e7 18007317   1.80e7 2013~
10  2013 tt1815862 After~ nota~ notalk     FAIL   1.3 e8 60522097   2.44e8 2013~
# i 1,784 more rows
# i 5 more variables: budget_2013 <int>, domgross_2013 <dbl>,
#   intgross_2013 <dbl>, period_code <int>, decade_code <int>
```

For more on this example, visit Bechdel analysis using the tidyverse.

C.2 Using `read_rds` to read in an R data set

We have provided the `nnyfs.Rds` data file on the course data page.

Suppose you have downloaded this data file into a directory on your computer called `data` which is a sub-directory of the directory where you plan to do your work, perhaps called `431-nnyfs`.

Open RStudio and create a new project into the `431-nnyfs` directory on your computer. You should see a `data` subdirectory in the Files window in RStudio after the project is created.

Now, read in the `nnyfs.Rds` file to a new tibble in R called `nnyfs` with the following command:

```
nnyfs <- read_rds("data/nnyfs.Rds")
```

Here are the results…

```
nnyfs
```

```
# A tibble: 1,518 x 45
   SEQN  sex    age_child race_eth    educ_child language sampling_wt income_pov
   <chr> <fct>      <dbl> <fct>            <dbl> <fct>          <dbl>      <dbl>
 1 71917 Female        15 3_Black No~          9 English       28299.       0.21
 2 71918 Female         8 3_Black No~          2 English       15127.       5
 3 71919 Female        14 2_White No~          8 English       29977.       5
 4 71920 Female        15 2_White No~          8 English       80652.       0.87
 5 71921 Male           3 2_White No~         NA English       55592.       4.34
 6 71922 Male          12 1_Hispanic           6 English       27365.       5
 7 71923 Male          12 2_White No~          5 English       86673.       5
 8 71924 Female         8 4_Other Ra~          2 English       39549.       2.74
 9 71925 Male           7 1_Hispanic           0 English       42333.       0.46
10 71926 Male           8 3_Black No~          2 English       15307.       1.57
# i 1,508 more rows
# i 37 more variables: age_adult <dbl>, educ_adult <fct>, respondent <fct>,
#   salt_used <fct>, energy <dbl>, protein <dbl>, sugar <dbl>, fat <dbl>,
#   diet_yesterday <fct>, water <dbl>, plank_time <dbl>, height <dbl>,
#   weight <dbl>, bmi <dbl>, bmi_cat <fct>, arm_length <dbl>, waist <dbl>,
#   arm_circ <dbl>, calf_circ <dbl>, calf_skinfold <dbl>,
#   triceps_skinfold <dbl>, subscapular_skinfold <dbl>, active_days <dbl>, ...
```

C.3 Using `read_csv` to read in a comma-separated version of a data file

We have provided the `fev_ros.csv` data file on the course data page.

Suppose you have downloaded this data file into a directory on your computer called `data`.

Now, read in the `fev_ros.csv` file to a new tibble in R called `fev_ros` with the following command, assuming you also want to convert the `character` variables to `factors`, as you will often want to do before analyzing the results.

```
fev_ros <- read_csv("data/fev_ros.csv") |>
  mutate(across(where(is.character), as_factor))
```

```
Rows: 654 Columns: 6
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (2): sex, smoke
dbl (4): id, age, fev, height

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
fev_ros
```

```
# A tibble: 654 x 6
      id   age   fev height sex    smoke
   <dbl> <dbl> <dbl>  <dbl> <fct>  <fct>
 1   301     9  1.71   57   female non-current smoker
 2   451     8  1.72   67.5 female non-current smoker
 3   501     7  1.72   54.5 female non-current smoker
 4   642     9  1.56   53   male   non-current smoker
 5   901     9  1.90   57   male   non-current smoker
 6  1701     8  2.34   61   female non-current smoker
 7  1752     6  1.92   58   female non-current smoker
 8  1753     6  1.42   56   female non-current smoker
 9  1901     8  1.99   58.5 female non-current smoker
10  1951     9  1.94   60   female non-current smoker
# i 644 more rows
```

Note that, for example, `sex` and `smoke` are now listed as factor (`fctr`) variables.

For more on factors, visit https://r4ds.had.co.nz/factors.html.

Converting Data Frames to Tibbles

Use `as_tibble()` or simply `tibble()` to assign the attributes of a tibble to a data frame. Note that `read_rds` and `read_csv` automatically create tibbles.

For more on tibbles, visit https://r4ds.had.co.nz/tibbles.html.

For more advice

- The R Graphics Cookbook has an excellent chapter on Getting Your Data into Shape which is well worth your time.

D Data Sets Used in this Book

D.1 Data Sets Provided on our Web Site

See the repository at https://github.com/THOMASELOVE/431-data.

Data Set	File	Type	Loaded	Source
bloodlead	`.csv`	Comma-separated text	Section 5.2	J. Statistics Education
bodyfat	`.csv`	Comma-separated text	Section 20.2	Kaggle
BPX_I	`.xpt`	SAS transport file	Section 8.2	NHANES 2015-16
cbaths	`.txt`	Tab-delimited text	Section 9.2	Data and Story Library
cle_nbd	`.csv`	Comma-separated text	Section 4.2	Census Reporter & NEOCANDO
coasters	`.csv`	Comma-separated text	Section 11.2	Roller Coaster Data Base
countries	`.csv`	Comma-separated text	Section 22.2	WHO and others
craters	`.sav`	SPSS data set	Section 12.2	Data and Story Library
darwin	`.Rds`	R data set	Section 7.2	UC Irvine Repository
DEMO_I	`.xpt`	SAS transport file	Section 8.2	NHANES 2015-16
fev_ros	`.csv`	Comma-separated text	Section 19.2	Vanderbilt Data
nations	`.csv`	Comma-separated text	Section 17.2	WHO and others
nnyfs	`.Rds`	R data set	Section 23.2	NNYFS at CDC
park_rct	`.xlsx`	Excel worksheet	Section 6.3	NEJM article
plasma	`.csv`	Comma-separated text	Section 21.2	Vanderbilt Data
storage	`.Rds`	R data set	Section 10.2	Cleveland Clinic

Data Set	File	Type	Loaded	Source
supraclav	`.dta`	Stata data set	Section 16.2	Cleveland Clinic
tattoos	`.txt`	Tab-delimited text	Section 14.2	Data and Story Library

D.2 Data Sets imported from R Packages

Data Set	R Package	Loaded	HTML Link
bechdel	`fivethirtyeight`	Section C.1	Analysis using the Tidyverse
childcare_costs	`tidytuesdayR`	Section 18.2	Github for Tidy Tuesday 2023-05-09
counties	`tidytuesdayR`	Section 18.2	Github for Tidy Tuesday 2023-05-09
penguins	`palmerpenguins`	Section 2.2	palmerpenguins
strep_tb	`medicaldata`	Section 13.2	medicaldata

E Statistical Summaries

E.1 Summarizing a Quantity

E.1.1 The sample mean, $\bar{x}$

The sample mean, $\bar{x}$, of a set of n observations is the sum of the observations divided by the count, n.

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

- Wikipedia on the mean

E.1.2 The sample variance and standard deviation

The sample variance, s^2 of a set of n observations is the sum of the squared deviance of each of the observations from the sample mean, divided by one less than the count, n minus 1.

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

The sample standard deviation, s is the square root of the sample variance.

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

- Wikipedia on the standard deviation

E.1.3 The range

The range of a data set can be expressed either as the two-element vector consisting of the minimum and maximum values of the data, or as the difference (maximum - minimum) depending on context. Larger range is an indication of greater variation.

- Wikipedia on the range

E.1.4 The median

The median of a set of n observations separates the upper half of the observations from the lower half. It is the 50th percentile (or quantile) of the data, calculated as follows:

- if n is odd, the sample median is the middle value when the data are sorted in order from lowest to highest
- if n is even, the sample median is the mean of the two middle values when the data are sorted in order from lowest to highest
- Wikipedia on the median

E.1.5 Quantiles / Percentiles

Quantiles or percentiles (the terms are used equivalently here) divide the observations of a sample into groups with equal probabilities within the sample. For example, the 42nd percentile is the cut point which splits the sample data so that 42% of the observations are below that cut point, with the remaining 58% above that cut point.

We often describe the 25th percentile as the first quartile (Q25) of the data, and the 75th percentile as the third quartile (Q75). Some regard the zeroth percentile as the minimum value in the data, while the 100th percentile describes the maximum value.

- Wikipedia on percentiles

E.1.6 The IQR (inter-quartile range)

The IQR of a data set is the difference between the third quartile (Q75) and the first quartile (Q25). Thus, it provides a measure of the range of the "middle half" of the data. It also describes the length of the box in a boxplot.

- Wikipedia on the inter-quartile range

E.1.7 The median absolute deviation

The median absolute deviation (MAD) used in this book as a robust measure of variation, is the median of the absolute deviations from the sample median, so if x_{MED} is the sample median of a set of n observations, multiplied by the constant 1.4826:

$$MAD = median(|x_i - x_{MED}) \times 1.4826$$

The purpose of multiplying by the constant is so that if data come from a Normal distribution, the MAD (with this multiplication) and the standard deviation will have the same value.

- Wikipedia on the median absolute deviation

E.1.8 The standard error of the sample mean

The standard error of the sample mean for a set of n observations is:

$$SE = \frac{s}{\sqrt{n}}$$

- Wikipedia on the standard error generally, and the standard error of the sample mean, specifically

E.1.9 The coefficient of variation

The coefficient of variation of a set of n observations is the sample standard deviation divided by the sample mean:

$$CV = \frac{s}{\bar{x}}$$

- Wikipedia on the coefficient of variation

E.1.10 The mode

The mode of a set of observations is simply the most common value. A batch of data can have more than one mode, if there are multiple observations which tie for the most common value.

- Wikipedia on the mode

E.1.11 Skewness

Skew measures the degree of asymmetry in our data. The `skewness` function in R that is used by `describe_distribution()` is Type II, often used by SAS and SPSS, for instance. For a sample of n observations with sample mean $\bar{x}$,

- The Type I or "classical" method produces

$$skew_I = \left(\frac{\sum_{i=1}^{n}(x_i - \bar{x})^3}{n} \right) / \left(\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n} \right)^{1.5}$$

- The Type II method then adjusts the result of the Type I method as follows:

$$skew_{II} = skew_I \times \sqrt{\frac{n(n-1)}{n-2}}$$

and this is what `describe_distribution()` returns.

Each of these skewness measures will have a value of zero for symmetric data, including the Normal distribution, with negative values indicating left skew and positive values indicating right skew.

- See the skewness and kurtosis page in easystats for more details.
- Wikipedia on skewness.

E.1.12 Simple Skewness ($skew_0$)

A (perhaps overly) simple description of skew of a set of n observations that I occasionally look at is the sample mean minus the sample median, divided by the sample standard deviation:

$$skew_0 = \frac{\bar{x} - median}{s}$$

- Values of $skew_0$ above +0.2 indicate right skew worthy of additional consideration
- Values of $skew_0$ below -0.2 indicate left skew worthy of additional consideration
- Values of $skew_0$ near 0 indicate fairly symmetric data

E.1.13 Kurtosis

The sample kurtosis measures the "tailedness" of a distribution - whether it is light or heavy tailed as compared to a Normal distribution.

The `kurtosis` function in R that is used by `describe_distribution()`is also the Type II approach.

For a sample of n observations with sample mean $\bar{x}$,

- The Type I or "classical" method produces

$$kurt_I = n \times \frac{\sum_{i=1}^{n}(x_i - \bar{x})^4}{(\sum_{i=1}^{n}(x_i - \bar{x})^2)^2}$$

- The Type II method then adjusts the result of the Type I method as follows:

$$kurt_{II} = ((n+1) \times kurt_I + 6) \times \frac{n-1}{(n-2) \times (n-3)}$$

and this is what `describe_distribution()` returns.

Each of these kurtosis estimates measures tail behavior and can help in characterizing the sample distribution as either:

- mesokurtic (distribution has a kurtosis value near 0) which indicates similar tail behavior to a Normal distribution
- leptokurtic ("fatter tails") as indicated by a kurtosis value well above 0, or
- platykurtic ("thinner tails") as indicated by a kurtosis value well below 0.

That said, I cannot remember the last time I used a kurtosis calculation in practical work.

- See the skewness and kurtosis page in easystats for more details.
- Wikipedia on kurtosis.

E.2 Summarizing an Association

E.2.1 The Pearson Correlation Coefficient

When applied to a sample, the Pearson correlation is represented by r_{xy}, and can be calculated using the formula below, assuming we have n observations on both x and y, and that $\bar{x}$ is the sample mean of the x values, and $\bar{y}$ is the sample mean of the y values.

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

An equivalent formula requires that we specify s_x as the standard deviation of x, and s_y as the standard deviation of y. Then we have:

$$r_{xy} = \frac{1}{n-1}\sum_{i=1}^{n}\left(\frac{x_i - \bar{x}}{s_x}\right)\left(\frac{y_i - \bar{y}}{s_y}\right)$$

Yet another formula for the Pearson correlation is:

$$r_{xy} = \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{(n-1)s_x s_y}$$

- Wikipedia on the Pearson correlation coefficient

E.2.2 Intercept and Slope of a Least Squares Fit

Suppose we have n observations on two variables, x and y, where the mean of x is $\bar{x}$ and the mean of y is $\bar{y}$. We want to estimate the slope (b) and y-intercept (a) of the least squares line $y = a + bx$.

The least squares estimate of the slope is:

$$\hat{b} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})y}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

and the least squares estimate of the intercept is:

$$\hat{a} = \bar{y} - \hat{b}\bar{x}$$

In addition to writing the equation as $y = a + bx$, we could also write it as:

$$y_i = \bar{y} + \hat{b}(x_i - \bar{x}) + r_i$$

where the residual r_i is

$$r_i = y_i - (\hat{a} + \hat{b}x_i)$$

and this should make it clear that the least squares regression line must pass through $(\bar{x}, \bar{y})$, the means of the two variables.

- Wikipedia on the method of least squares

F References

Bock, David E., Paul F. Velleman, and Richard D. De Veaux. 2004. *Stats: Modelling the World*. Boston MA: Pearson Addison-Wesley.

Cata, Juan P, Eric A. Klein, and Gerald A. Hoeltge et al. 2011. "Blood Storage Duration and Biochemical Recurrence of Cancer After Radical Prostatectomy." *Mayo Clinic Proceedings* 86(2): 120–27. https://doi.org/10.4065/mcp.2010.0313.

Çetinkaya-Rundel, Mine, and Johanna Hardin. 2024. *Introduction to Modern Statistics*. Second Edition. OpenIntro Project. https://openintro-ims.netlify.app/.

Cilia, Nicole D., Giuseppe De Gregorio, Claudio De Stefano, Francesco Fontanella, Angelo Marcelli, and Antonio Parziale. 2022. "Diagnosing Alzheimer's Disease from on-Line Handwriting: A Novel Dataset and Performance Benchmarking." *Engineering Applications of Artificial Intelligence* 111: 104822. https://doi.org/10.1016/j.engappai.2022.104822.

Cohen, Jacob. 1988. *Statistical Power Analysis for the Behavioral Sciences*. 2nd ed. New York: Routledge. https://www.utstat.toronto.edu/~brunner/oldclass/378f16/readings/CohenPower.pdf.

Gelman, Andrew, and Jennifer Hill. 2007. *Data Analysis Using Regression and Multilevel-Hierarchical Models*. New York: Cambridge University Press. http://www.stat.columbia.edu/~gelman/arm/.

Gelman, Andrew, Jennifer Hill, and Aki Vehtari. 2021. *Regression and Other Stories*. New York: Cambridge University Press. https://avehtari.github.io/ROS-Examples/.

Gelman, Andrew, and Deborah Nolan. 2017. *Teaching Statistics: A Bag of Tricks*. Second Edition. Oxford, UK: Oxford University Press.

Goetz, Christopher G., et al. 2008. "MDS-Unified Parkinson's Disease Rating Scale (MDS-UPDRS)." https://www.movementdisorders.org/MDS/MDS-Rating-Scales/MDS-Unified-Parkinsons-Disease-Rating-Scale-MDS-UPDRS.htm.

Higgins, Peter. 2023. "Medicaldata: Data Package for Medical Datasets." https://higgi13425.github.io/medicaldata/.

Horst, Allison Marie, Alison Presmanes Hill, and Kristen B Gorman. 2020. "Palmerpenguins: Palmer Archipelago (Antarctica) Penguin Data." https://allisonhorst.github.io/palmerpenguins/.

Ismay, Chester, and Albert Y. Kim. 2024. *ModernDive: Statistical Inference via Data Science*. CRC Press. http://moderndive.com/.

Lüdecke, Daniel, Mattan S. Ben-Shachar, Indrajeet Patil, Brenton M. Wiernik, Etienne Bacher, Rémi Thériault, and Dominique Makowski. 2022. "Easystats: Framework for Easy Statistical Modeling, Visualization, and Reporting." *CRAN*. https://doi.org/10.32614/CRAN.package.easystats.

Makowski, Dominique, Mattan S. Ben-Shachar, and Daniel Lüdecke. 2019. "bayestestR: Describing Effects and Their Uncertainty, Existence and Significance Within the Bayesian Framework." *Journaal of Open Source Software* 4 (40): 1541. https://doi.org/doi.org/10.21105/joss.01541.

Meissner, Wassilios G., Philippe Remy, Caroline Giordana, David Maltête, Pascal Derkinderen, Jean-Luc Houéto, and Mathieu Anheim for the LIXIPARK Study Group. 2024. "Trial of Lixisenatide in Early Parkinson's Disease." *The New England Journal of Medicine* 390 (13): 1176–85. https://doi.org/10.1056/NEJMoa2312323.

Morton, D., A. Saah, S. Silberg, W. Owens, M. Roberts, and M. Saah. 1982. "Lead Absorption in Children of Employees in a Lead Related Industry." *American Journal of Epidemiology* 115: 549–55.

NEOCANDO. 2024. "NEOCANDO: Northeast Ohio Community and Neighborhood Data for Organizing." https://neocando.case.edu/neocando/index.jsp.

Nierenberg, D. W., T. A. Stukel, J. A. Baron, B. J. Dain, and E. R. Greenberg. 1989. "Determinants of Plasma Levels of Beta-Carotene and Retinol." *American Journal of Epidemiology* 130(3): 511–21. https://doi.org/10.1093/oxfordjournals.aje.a115365.

Norman, Geoffrey R., and David L. Streiner. 2014. *Biostatistics: The Bare Essentials*. Fourth Edition. People's Medical Publishing House.

Pruzek, Robert M., and James E. Helmreich. 2009. "Enhancing Dependent Sample Analyses with Graphics." *Journal of Statistics Education* 17(1). http://ww2.amstat.org/publications/jse/v17n1/helmreich.html.

Roberman, Dmitry, Harendra Arora, Daniel I. Sessler, Michael Ritchey, Jing You, and Priya Kumar. 2011. "Combined Versus Sequential Injection of Mepivacaine and Ropivacaine for Supraclavicular Nerve Blocks." *Reg Anesth Pain Med* 36: 145–50.

Tukey, John W. 1977. *Exploratory Data Analysis*. Boston: Addison-Wesley. https://archive.org/details/exploratorydataa0000tuke_7616.

U.S. Census Bureau. 2020. "Cleveland, OH City Neighborhoods (Statistical Planning Areas)." https://censusreporter.org/user_geo/70623cc2a11b0ad0e223a2eaa330cf8d/.

van Buuren, Stef. 2021. *Flexible Imputation of Missing Data*. 2nd ed. New York: Chapman & Hall / CRC. https://stefvanbuuren.name/fimd/.

Vittinghoff, Eric, David V. Glidden, Stephen C. Shiboski, and Charles E. McCulloch. 2012. *Regression Methods in Biostatistics: Linear, Logistic, Survival, and Repeated Measures Models*. Second. Springer-Verlag, Inc. http://www.biostat.ucsf.edu/vgsm/.

Wickham, Hadley, Mine Çetinkaya-Rundel, and Garrett Grolemund. 2024. "R for Data Science, 2e." https://r4ds.hadley.nz/.