Course Notes for 431

Thomas E. Love, Ph.D.

2023-08-14

Table of contents

II Part B. Comparing Summaries   372

18 Hypothesis Testing: What is it good for?   373

19 Confidence Intervals for a Mean   381

Working with These Notes

1. This document 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 document, using an automated index.
3. Any of the code provided in the document can be copied to the clipboard using the Copy icon at the top right of the code block.
4. If you find a typo or other problem, please tell us about it through Campuswire.
5. This is a **draft** document. The publication date is shown above. Later drafts (including, eventually, all three parts) **will appear** irregularly between now and December, so you'll need to **stay current**.
6. If, for some awful reason, you want to print or save this document, we recommend waiting until the semester is over to do so, when a more "final" version will be available.

What You'll Find Here

These Notes provide a series of examples using R to work through issues that are likely to come up in PQHS/CRSP/MPHP 431. What you will mostly find are brief explanations of a key idea or summary, accompanied (most of the time) by R code and a demonstration of the results of applying that code.

While these Notes share some of the features of a textbook, they are neither comprehensive nor completely original. 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,

but what we don't do is follow these notes very precisely. We assume instead that you will read the materials and try to learn from 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 2023 is

https://thomaselove.github.io/431-2023/.

Go there for all information related to the course.

All of the code and text in these Notes is posted online as HTML, and it is also possible to download PDF and ePub versions of the document from the down arrow next to the title (Notes for 431) at the top left of this screen. All data and R code related to these notes are also available to you through our course web site.

By the end of the semester, you will also have access to the Quarto files which generate everything in the document, including all of the R results. You will also use Quarto during the semester to complete your assignments, and so we will demonstrate the use of Quarto and RStudio (the "program" we use to interface with the R language) in class.

Setting Up R

These Notes make extensive use of

- the statistical software language R, and
- the development environment RStudio,

both of which are free, and you'll need to install them on your machine. Instructions for doing so will be found on the course website.

If you need a gentle introduction, or if you're just new to R and RStudio and need to learn about them, we encourage you to take a look at https://moderndive.com/, which provides an introduction to statistical and data sciences via R at Ismay and Kim (2022).

Quarto

These notes were written using Quarto, which we'll learn to use in 431. Quarto, like R and RStudio, is free and open source.

Quarto is described as an *scientific and technical publishing system* for data science, which lets you (among many other things):

- save and execute R code
- generate high-quality reports that can be shared with an audience

The Quarto Get Started page which provides an overview and quick tour of what's possible with Quarto.

Another excellent resource to learn more about Quarto is the Communicate section (especially the [Quarto chapter]https://r4ds.hadley.nz/quarto.html)) of Hadley Wickham and Grolemund (2023).

R Packages

At the start of each chapter that involves R code, I'll present a series of commands I run to set up R to use several packages (libraries) of functions that expand its capabilities, make a specific change to how I want R output to be displayed (that's the `comment = NA` piece) and sets the theme for most graphs to `theme_bw()`. A chunk of code like this will occur near the top of any Quarto work.

For example, this is the setup for one of our early chapters that loads four packages.

```r
knitr::opts_chunk$set(comment = NA)

library(palmerpenguins)
library(janitor)
library(knitr)
library(tidyverse)

theme_set(theme_bw())
```

You only need to install a package once, but you need to reload it (using the `library()` function) every time you start a new session. I always load the package called `tidyverse` last, since doing so avoids some annoying problems.

The `Love-boost.R` script

In October, when we start Part B of the course, we'll use some special R functions I've gathered for you in a script called `Love-boost`. I'll tell R about that code using the following command…

```r
source("data/Love-boost.R")
```

The `Love-boost.R` script includes four functions:

- `bootdif`
- `saifs.ci`
- `twobytwo`
- `retrodesign`

Packages Used in these Notes

A list of all R packages we want you to install this semester (which includes some packages not included in these Notes) is maintained at our course web site.

Package	Parts	Key functions in the Package
arm	C	–
boot	B	–
broom	A, B, C	tidy, glance, augment (part of tidymodels)
car	A, C	boxCox, powerTransform
Epi	B	twoby2
fivethirtyeight	Appendix	source of data
GGally	A, C	ggpairs
ggrepel	C	–
ggridges	A, B	–
ggstance	A	–
gt	A	for presenting tables
gtsummary	A	tbl_summary
Hmisc	A, B, C	describe and others
janitor	A, B, C	tabyl and others
kableExtra	A	kbl, kable_stylings
knitr	A, B, C	kable
lvplot	A	geom_lv
mice	C	–
modelsummary	A, C	modelsummary
mosaic	A, B, C	favstats, inspect
naniar	A	n_miss, miss_case_table, gg_miss_var
NHANES	A	source of data
palmerpenguins	A	source of data
patchwork	A, B, C	for combining/annotating plots
psych	A, B	describe
pwr	B	–
rms	C	–
sessioninfo	Appendix	–
simputation	A	various imputation functions
summarytools	A	descr, dfSummary
tidyverse	A, B, C, Appendix	dozens of functions
vcd	B	–
visdat	A	vis_dat, vis_miss

The `tidyverse`

The `tidyverse` package is actually a meta-package which includes the following core packages:

- `ggplot2` for creating graphics
- `dplyr` for data manipulation
- `tidyr` for creating tidy data
- `readr` for reading in rectangular data
- `purrr` for working with functions and vectors
- `tibble` for creating tibbles - lazy, surly data frames
- `stringr` for working with data strings
- `forcats` for solving problems with factors

Loading the tidyverse with `library(tidyverse)` loads those eight packages.

Installing the tidyverse also installs several other useful packages on your machine, like `glue` and `lubridate`, for example. Read more about the `tidyverse` at https://www.tidyverse.org/

1 Data Science and 431

The definition of **data science** can be a little slippery. One current view of data science, is exemplified by Steven Geringer's 2014 Venn diagram.
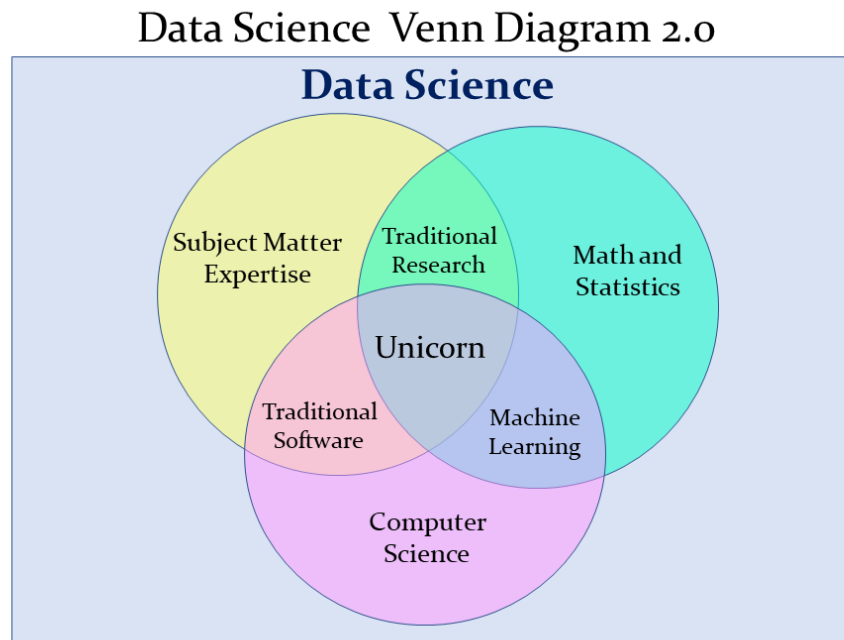
Figure 1.1: Data Science Venn Diagram from Steven Geringer

- 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**, by Garrett Grolemund and Hadley Wickham, which is a key text for this course (Hadley Wickham and Grolemund 2023).
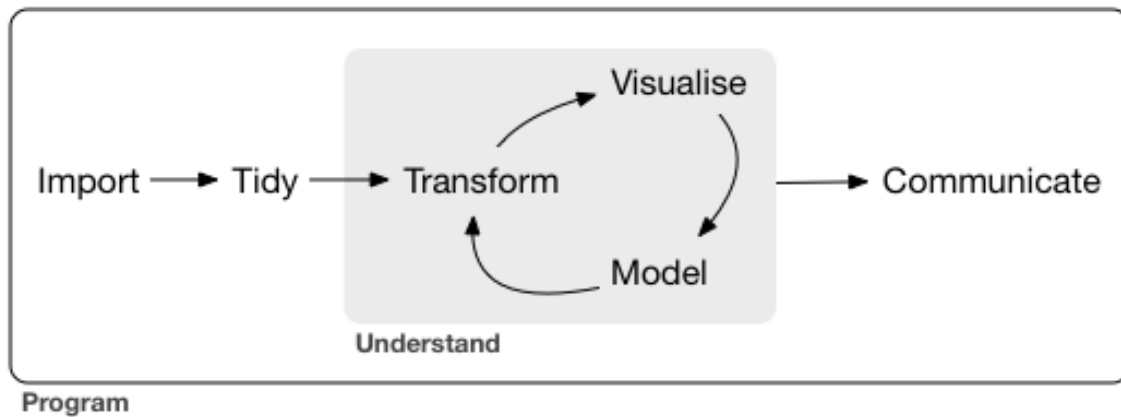
Figure 1.2: Source: R for Data Science: Introduction

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 Hadley Wickham and Grolemund (2023).

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 (2022)

- We learn how to use the `tidyverse` (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 Hadley Wickham and Grolemund (2023). 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.

- 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

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

There's very little of the mathematical underpinnings here:

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

Instead, these notes (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. Our next Chapter provides a first example.

Part I

Part A. Exploring Data

2 The Palmer Penguins

The data in the `palmerpenguins` package in R includes information on 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 it available[1]. The data describe three species of penguins, called Adelie, Chinstrap and Gentoo.

For more on the `palmerpenguins` package, visit https://allisonhorst.github.io/palmerpenguins/.

2.1 Setup: Packages Used Here

We will use the `palmerpenguins` package to supply us with data for this chapter. The `janitor` packages includes several useful functions, including `tabyl`. The `knitr` package includes the `kable()` function we'll use. Finally, the `tidyverse` package will provide the bulk of the functions we'll use in our work throughout the semester.

I always load the `tidyverse` last, because it solves some problems to do so.

```
knitr::opts_chunk$set(comment = NA)

library(palmerpenguins)
library(janitor)
library(knitr)
library(kableExtra)
library(gt)
library(tidyverse)

theme_set(theme_bw())
```

---

[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.

2.2 Viewing a Data Set

The `penguins` data from the `palmerpenguins` package contains 344 rows and 8 columns. Each row contains data for a different penguin, and each column describes a variable contained in the data set.

```
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 instance, the first penguin in the data is of the species Adelie, and was observed on the island called Torgeson. The remaining data for that penguin include measures of its bill length and depth, its flipper length and body mass, its sex and the year in which it was observed.

Note that though there are 344 rows in the tibble of data called `penguins`, only the first ten rows (penguins) are shown in the table above. Note also that the symbol `NA` or `<NA>` is used to indicate a missing (not available) value.

2.3 Create `new_penguins`: Eliminating Missing Data

Next, let's take the `penguins` data from the `palmerpenguins` package, and identify those observations which have complete data (so, no missing values) in four variables of interest. We'll store that result in a new tibble (data set) called `new_penguins` and then take a look at that result using the following code.

Note that the code below:

- uses the "pipe" |> to send the penguins tibble to the `filter()` function
- uses <- to assign the result of our work to the `new_penguins` tibble
- uses the `complete.cases()` function to remove cases within `penguins` that have missing data on any of the four variables (`flipper_length_mm`, `body_mass_g`, `species` or `sex`) that we identify

```
new_penguins <- penguins |>
    filter(complete.cases(flipper_length_mm, body_mass_g, species, sex))

new_penguins
```

```
# A tibble: 333 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           36.7          19.3               193        3450
 5 Adelie  Torgersen           39.3          20.6               190        3650
 6 Adelie  Torgersen           38.9          17.8               181        3625
 7 Adelie  Torgersen           39.2          19.6               195        4675
 8 Adelie  Torgersen           41.1          17.6               182        3200
 9 Adelie  Torgersen           38.6          21.2               191        3800
10 Adelie  Torgersen           34.6          21.1               198        4400
# i 323 more rows
# i 2 more variables: sex <fct>, year <int>
```

2.4 Counting Things and Making Tables

So, how many penguins are in our `new_penguins` data? When we printed out the result, we got an answer, but (as with many things in R) there are many ways to get the same result.

```
nrow(new_penguins)
```

```
[1] 333
```

How do our `new_penguins` data break down by sex and species? We'll use the `tabyl()` function from the `janitor` package to look at this.

sex/species	Adelie	Chinstrap	Gentoo	Total
female	73	34	58	165
male	73	34	61	168
Total	146	68	119	333

```
new_penguins |>
    tabyl(sex, species)
```

```
   sex Adelie Chinstrap Gentoo
female     73        34     58
  male     73        34     61
```

The output is reasonably clear (there are 73 female and 73 male Adelie penguins in the **newpenguins** tibble, for example) but could we make that table a little prettier, and while we're at it, can we add the row and column totals?

```
new_penguins |>
    tabyl(sex, species) |>
    adorn_totals(where = c("row", "col")) |> # add row, column totals
    kable()  # one convenient way to make the table prettier
```

sex	Adelie	Chinstrap	Gentoo	Total
female	73	34	58	165
male	73	34	61	168
Total	146	68	119	333

The **kable()** function comes from the **knitr** package we loaded earlier. Notice that we added some comments to the code here with the prefix **#**. These comments are ignored by R in processing the data.

Another approach we could have used here is aided by the **kableExtra** package's function called **kbl()**, which lets us set up the alignment of our columns. We'll also add the species name using **adorn_title()** from the **janitor** package.

```
new_penguins |>
    tabyl(sex, species) |>
    adorn_totals(where = c("row", "col")) |>
    adorn_title(placement = "combined") |>
    kbl(align = c('lcccr')) |>
    kable_styling(full_width = FALSE)
```

We can switch the rows and columns, and add some additional features, using the code below, which makes use of the `gt()` and `tab_header()` functions from the `gt` package, which is designed to help build complex tables. More on the incredibly versatile `gt()` package is available at https://gt.rstudio.com/.

```
new_penguins |>
    tabyl(species, sex) |>
    adorn_totals(where = c("row", "col")) |>
    gt() |>
    tab_header(
      title = md("Palmer Penguins in **newpenguins**"),
      subtitle = "Comparing sexes by species"
    )
```

<div align="center">

Palmer Penguins in **newpenguins**

Comparing sexes by species

species	female	male	Total
Adelie	73	73	146
Chinstrap	34	34	68
Gentoo	58	61	119
Total	165	168	333

</div>

2.5 Creating a Scatterplot

Now, let's look at the other two variables of interest. Let's create a graph showing the association of body mass with flipper length across the complete set of 333 penguins.

```
ggplot(new_penguins, aes(x = body_mass_g, y = flipper_length_mm)) +
    geom_point()
```
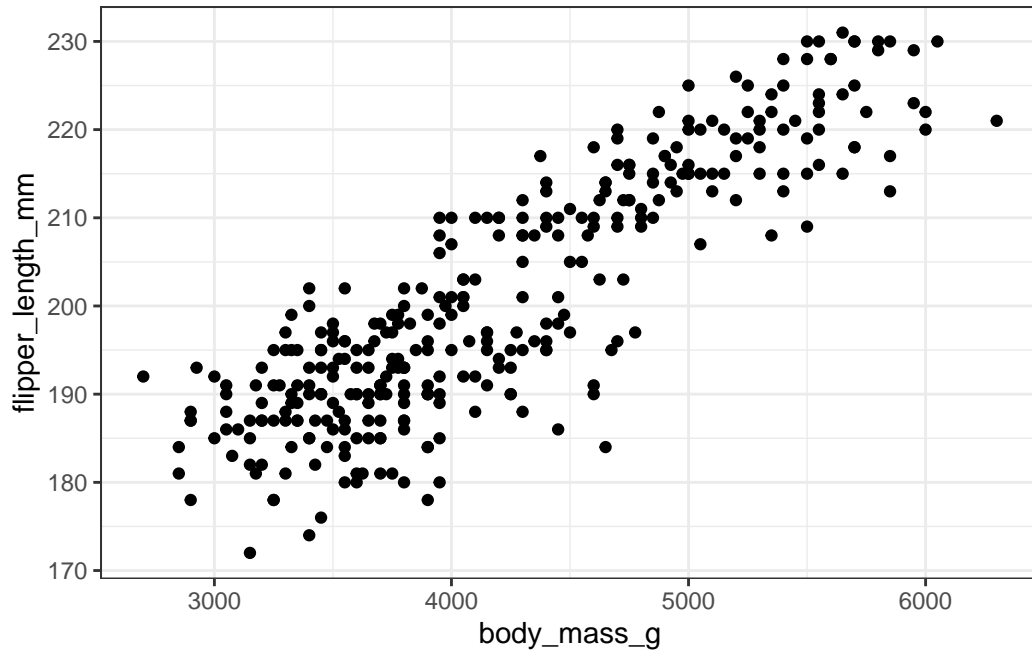
Some of you may want to include a straight-line model (fit by a classical linear regression) to this plot. One way to do that in R involves the addition of a single line of code, like this:

```r
ggplot(new_penguins, aes(x = body_mass_g, y = flipper_length_mm)) +
    geom_point() +
    geom_smooth(method = "lm", formula = y ~ x,
                col = "red", se = FALSE)
```
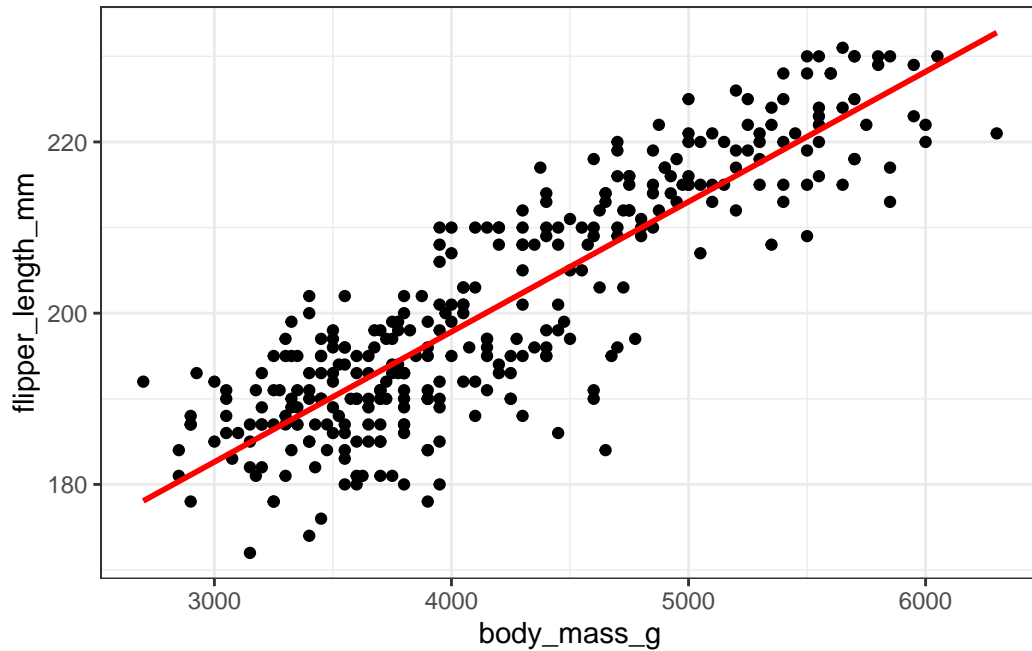
Whenever we build a graph for ourselves, these default choices may be sufficient. But I'd like to see a prettier version if I was going to show it to someone else. So, I might use a different color for each species, and I might add a title, like this.

```
ggplot(new_penguins, aes(x = body_mass_g, y = flipper_length_mm, col = species)) +
    geom_point() +
    labs(title = "Flipper Length and Body Mass for 333 of the Palmer Penguins")
```

Flipper Length and Body Mass for 333 of the Palmer Penguins
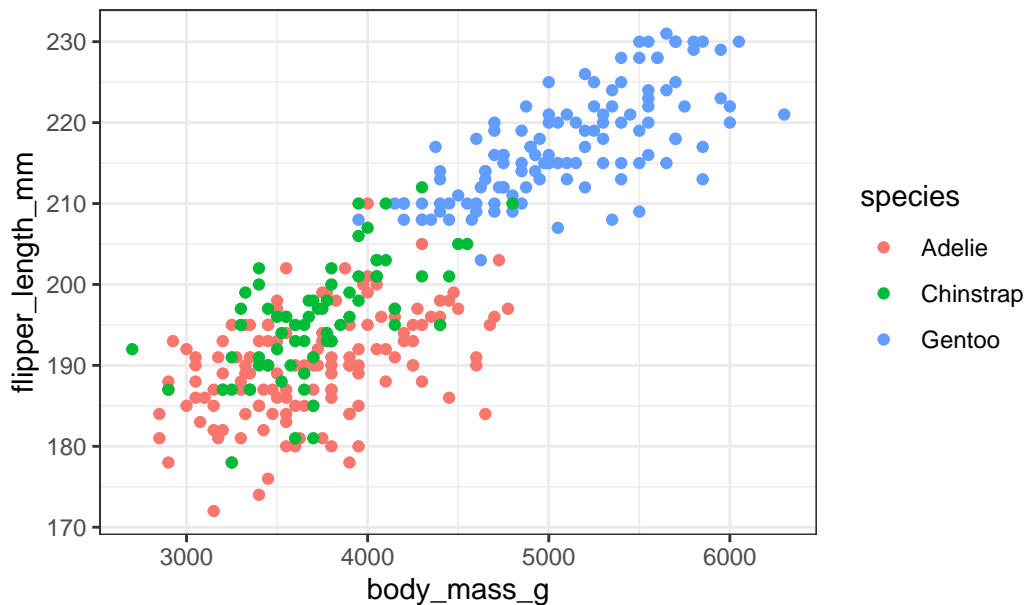
2.6 Six Ways To "Improve" This Graph

Now, let's build a new graph to incorporate some additional information and improve the appearance. Here, I want to:

1. plot the relationship between body mass and flipper length in light of both Sex and Species
2. increase the size of the points and add a little transparency so we can see if points overlap,
3. add some smooth curves to summarize the relationships between the two quantities (body mass and flipper length) within each combination of species and sex,
4. split the graph into two "facets" (one for each sex),
5. improve the axis labels,
6. improve the titles by adding a subtitle, and also adding in some code to count the penguins (rather than hard-coding in the total number.)

```
ggplot(new_penguins, aes(x = body_mass_g, y = flipper_length_mm,
                         col = species)) +
    geom_point(size = 2, alpha = 0.5) +
    geom_smooth(method = "loess", formula = y ~ x,
                se = FALSE, size = 1.5) +
    facet_grid(~ sex) +
```

```
    labs(title = "Flipper Length and Body Mass, by Sex & Species",
         subtitle = str_glue(nrow(new_penguins), " of the Palmer Penguins"),
         x = "Body Mass (g)",
         y = "Flipper Length (mm)")
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```
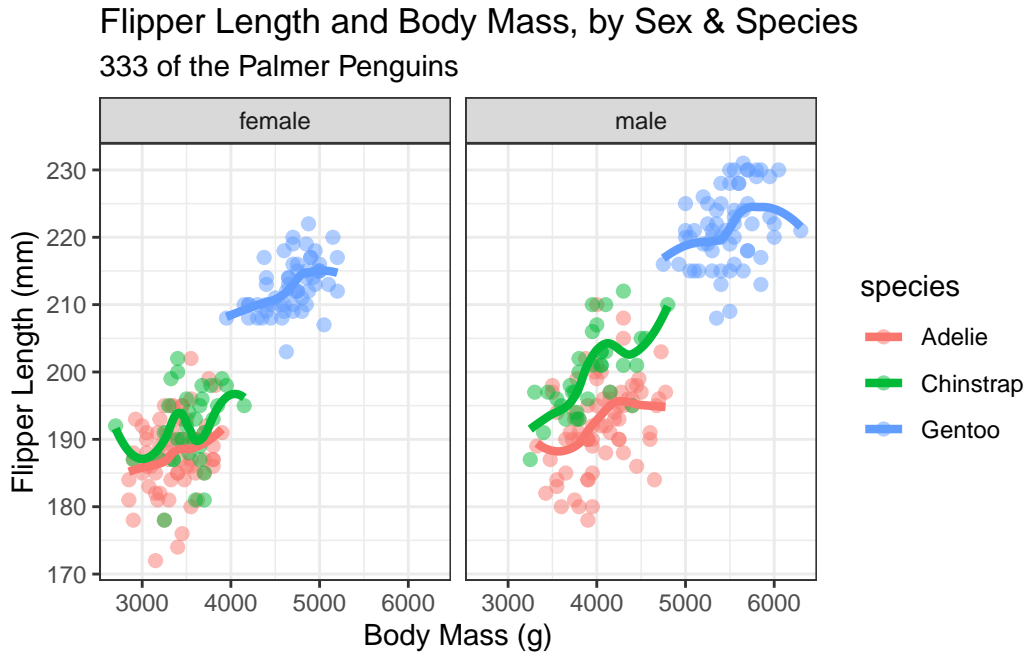

2.7 A Little Reflection

What can we learn from these plots and their construction? In particular,

- What do these plots suggest about the center of the distribution of each quantity (body mass and flipper length) overall, and within each combination of Sex and Species?
- What does the final plot suggest about the spread of the distribution of each of those quantities in each combination of Sex and Species?
- What do the plots suggest about the association of body mass and flipper length across the complete set of penguins?
- How does the shape and nature of this body mass - flipper length relationship change based on Sex and Species?

- Do you think it would be helpful to plot a straight-line relationship (rather than a smooth curve) within each combination of Sex and Species in the final plot? Why or why not? (Also, what would we have to do to the code to accomplish this?)
- How was the R code for the plot revised to accomplish each of the six "wants" specified above?

2.8 Coming Up

Next, we'll introduce and demonstrate some more of the many available tools for creating summaries (both graphical and numerical) in R, again working with the Palmer Penguins data.

3 Summarizing Penguins

We will again use the data contained in the `palmerpenguins` package in this chapter. Here, we present a few of the more appealing ways to obtain numerical and graphical summaries, without much explanation. We'll discuss these issues further in the rest of Part A of these Course Notes.

3.1 Setup: Packages Used Here

Here, we'll add several new packages to allow us to display some additional summaries, and present our tables and plots in different ways.

```r
knitr::opts_chunk$set(comment = NA)

library(palmerpenguins)
library(kableExtra)
library(gtsummary)
library(summarytools)
library(visdat)
library(lvplot)
library(tidyverse)

theme_set(theme_bw())
```

We will also use functions from the `mosaic` and `Hmisc` packages here, though I won't load them into our session at this time.

3.2 Our Data Set

Let's look again at the `penguins` data contained in the `palmerpenguins` package.

```r
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>
```

3.3 Numerical Summaries for a Tibble

Note that in this work, I sometimes don't explain all of the numerical summaries provided.
Some of that discussion is postponed to Chapter 7.

3.3.1 Using summary()

We have several ways to obtain useful summaries of all variables in the `penguins` data.

```
penguins |>
  summary()
```

```
     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
```

3.3.2 Using inspect() from `mosaic`

Some people like the `inspect()` function from the `mosaic` package.

```
penguins |>
  mosaic::inspect()
```

```
categorical variables:
    name  class levels   n missing
1 species factor      3 344       0
2  island factor      3 344       0
3     sex factor      2 333      11
                                distribution
1 Adelie (44.2%), Gentoo (36%) ...
2 Biscoe (48.8%), Dream (36%) ...
3 male (50.5%), female (49.5%)

quantitative variables:
             name    class    min      Q1  median      Q3     max        mean
1    bill_length_mm numeric   32.1  39.225   44.45    48.5    59.6    43.92193
2     bill_depth_mm numeric   13.1  15.600   17.30    18.7    21.5    17.15117
3 flipper_length_mm integer  172.0 190.000  197.00   213.0   231.0   200.91520
4       body_mass_g integer 2700.0 3550.000 4050.00 4750.0 6300.0  4201.75439
5              year integer 2007.0 2007.000 2008.00 2009.0 2009.0  2008.02907
         sd   n missing
1   5.4595837 342       2
2   1.9747932 342       2
3  14.0617137 342       2
4 801.9545357 342       2
5   0.8183559 344       0
```

Daniel Kaplan's Statistical Modeling, 2nd edition provides an entire course which coordinates nicely with the tools available in the `mosaic` package. In our course, we'll most often use this `inspect()` tool, and a related tool called `favstats`.

min	Q1	median	Q3	max	mean	sd	n	missing
32.1	39.225	44.45	48.5	59.6	43.92193	5.459584	342	2

species	min	Q1	median	Q3	max	mean	sd	n	missing
Adelie	32.1	36.75	38.80	40.750	46.0	38.79139	2.663405	151	1
Chinstrap	40.9	46.35	49.55	51.075	58.0	48.83382	3.339256	68	0
Gentoo	40.9	45.30	47.30	49.550	59.6	47.50488	3.081857	123	1

3.3.3 Using favstats() from mosaic.

The `favstats` function lets us look at some common summaries for a single variable, or for one variable divided into groups by another. We'll also return to this approach in Chapter 7.

```
mosaic::favstats(~ bill_length_mm, data = penguins) |>
  kbl() |>
  kable_styling()
```

```
mosaic::favstats(bill_length_mm ~ species, data = penguins) |>
  kbl() |>
  kable_styling()
```

3.3.4 Using describe() from `psych`

We can use the `describe()` function from the `psych` package to get some additional summaries, if we're interested, and here we also demonstrate the use of the `kbl()` and `kable_styling()` functions from the `kableExtra` package to make the table look appealing in HTML. More on the use of the `kableExtra` package is available here. We'll also return to this approach in Chapter 7.

```
penguins |>
  psych::describe() |>
  kbl() |>
  kable_styling()
```

3.3.5 Using describe() from `Hmisc`

One approach Frank Harrell has developed that I find helpful is the `describe()` function within his `Hmisc` package, which produces these results. We'll also return to this approach in Chapter 7.

	vars	n	mean	sd	median	trimmed	mad	min
species*	1	344	1.918605	0.8933198	2.00	1.898551	1.48260	1.0
island*	2	344	1.662791	0.7261940	2.00	1.579710	1.48260	1.0
bill_length_mm	3	342	43.921930	5.4595837	44.45	43.906934	7.04235	32.1
bill_depth_mm	4	342	17.151170	1.9747932	17.30	17.172628	2.22390	13.1
flipper_length_mm	5	342	200.915205	14.0617137	197.00	200.335766	16.30860	172.0
body_mass_g	6	342	4201.754386	801.9545357	4050.00	4154.014598	889.56000	2700.0
sex*	7	333	1.504504	0.5007321	2.00	1.505618	0.00000	1.0
year	8	344	2008.029070	0.8183559	2008.00	2008.036232	1.48260	2007.0

```
  penguins |>
    Hmisc::describe()
```

```
penguins

 8  Variables      344  Observations
--------------------------------------------------------------------------------
species
      n  missing distinct
    344        0        3

Value         Adelie Chinstrap    Gentoo
Frequency        152        68       124
Proportion     0.442     0.198     0.360
--------------------------------------------------------------------------------
island
      n  missing distinct
    344        0        3

Value          Biscoe     Dream Torgersen
Frequency         168       124        52
Proportion      0.488     0.360     0.151
--------------------------------------------------------------------------------
bill_length_mm
       n  missing distinct     Info     Mean      Gmd      .05      .10
     342        2      164        1    43.92    6.274    35.70    36.60
     .25      .50      .75      .90      .95
   39.23    44.45    48.50    50.80    51.99

lowest : 32.1 33.1 33.5 34   34.1, highest: 55.1 55.8 55.9 58   59.6
--------------------------------------------------------------------------------
```

```
bill_depth_mm
      n  missing distinct    Info    Mean     Gmd     .05     .10
    342        2      80       1   17.15   2.267    13.9    14.3
     .25      .50     .75     .90     .95
    15.6     17.3    18.7    19.5    20.0

lowest : 13.1 13.2 13.3 13.4 13.5, highest: 20.7 20.8 21.1 21.2 21.5
--------------------------------------------------------------------------------
flipper_length_mm
      n  missing distinct    Info    Mean     Gmd     .05     .10
    342        2      55   0.999   200.9   16.03   181.0   185.0
     .25      .50     .75     .90     .95
   190.0    197.0   213.0   220.9   225.0

lowest : 172 174 176 178 179, highest: 226 228 229 230 231
--------------------------------------------------------------------------------
body_mass_g
      n  missing distinct    Info    Mean     Gmd     .05     .10
    342        2      94       1    4202   911.8    3150    3300
     .25      .50     .75     .90     .95
    3550     4050    4750    5400    5650

lowest : 2700 2850 2900 2925 2975, highest: 5850 5950 6000 6050 6300
--------------------------------------------------------------------------------
sex
      n  missing distinct
    333       11       2

Value      female    male
Frequency     165     168
Proportion  0.495   0.505
--------------------------------------------------------------------------------
year
      n  missing distinct    Info    Mean     Gmd
    344        0       3   0.888    2008   0.8919

Value      2007  2008  2009
Frequency   110   114   120
Proportion 0.320 0.331 0.349

For the frequency table, variable is rounded to the nearest 0
--------------------------------------------------------------------------------
```

3.3.6 Using tbl_summary() from `gtsummary`

If you want to produce results which look like you might expect to see in a published paper, the `tbl_summary()` function from the `gtsummary` package has many nice features.

```
penguins |>
  tbl_summary()
```

```
Table printed with `knitr::kable()`, not {gt}. Learn why at
https://www.danieldsjoberg.com/gtsummary/articles/rmarkdown.html
To suppress this message, include `message = FALSE` in code chunk header.
```

Characteristic	**N = 344**
species	
Adelie	152 (44%)
Chinstrap	68 (20%)
Gentoo	124 (36%)
island	
Biscoe	168 (49%)
Dream	124 (36%)
Torgersen	52 (15%)
bill_length_mm	44.5 (39.2, 48.5)
Unknown	2
bill_depth_mm	17.30 (15.60, 18.70)
Unknown	2
flipper_length_mm	197 (190, 213)
Unknown	2
body_mass_g	4,050 (3,550, 4,750)
Unknown	2
sex	
female	165 (50%)
male	168 (50%)
Unknown	11
year	
2007	110 (32%)
2008	114 (33%)
2009	120 (35%)

A vignette explaining the use of the `gtsummary` package is available here. We'll also return to this approach in Chapter 7.

3.3.6.1 Using descr from `summarytools`

The `descr()` function from the `summarytools` package can also be used to provide numerical descriptions of all of the numerical variables contained within a tibble.

```
penguins |>
  descr(stats = "common" )
```

Non-numerical variable(s) ignored: species, island, sex

```
Descriptive Statistics
penguins
N: 344
```

	bill_depth_mm	bill_length_mm	body_mass_g	flipper_length_mm	year
Mean	17.15	43.92	4201.75	200.92	2008.03
Std.Dev	1.97	5.46	801.95	14.06	0.82
Min	13.10	32.10	2700.00	172.00	2007.00
Median	17.30	44.45	4050.00	197.00	2008.00
Max	21.50	59.60	6300.00	231.00	2009.00
N.Valid	342.00	342.00	342.00	342.00	344.00
Pct.Valid	99.42	99.42	99.42	99.42	100.00

An introduction to the summarytools package is available here, and illustrates some other ways to modify this output to suit your needs. We'll also return to this approach in Chapter 7.

3.3.7 dfSummary() from `summarytools`

The `dfSummary()` function from the `summarytools` package can be used to provide some additional descriptions of all variables within a tibble. You'll find more information about these numerical descriptions in Chapter 7.

```
dfSummary(penguins,
          plain.ascii  = FALSE,
          style        = "grid",
          graph.magnif = 0.75,
          valid.col    = FALSE)
```

Data Frame Summary
penguins
Dimensions: 344 x 8
Duplicates: 0

No	Variable	Stats / Values	Freqs (% of Valid)	Graph
1	species\ [factor]	1\. Adelie\ 2\. Chinstrap\ 3\. Gentoo	152 (44.2%)\ 68 (19.8%)\ 124 (36.0%)	IIIIIIII \ III \ IIIIIII
2	island\ [factor]	1\. Biscoe\ 2\. Dream\ 3\. Torgersen	168 (48.8%)\ 124 (36.0%)\ 52 (15.1%)	IIIIIIIII \ IIIIIII \ III
3	bill_length_mm\ [numeric]	Mean (sd) : 43.9 (5.5)\ min < med < max:\ 32.1 < 44.5 < 59.6\ IQR (CV) : 9.3 (0.1)	164 distinct values	\ \ \ \ . \ \ \ \ \ \ . : : : : \ \ \ : : : : : \ \ \ : : : : : : : : : : : :
4	bill_depth_mm\ [numeric]	Mean (sd) : 17.2 (2)\ min < med < max:\ 13.1 < 17.3 < 21.5\ IQR (CV) : 3.1 (0.1)	80 distinct values	\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ : . : : : . : : : : : :\ : : : : : : : :
5	flipper_length_mm\ [integer]	Mean (sd) : 200.9 (14.1)\ min < med < max:\ 172 < 197 < 231\ IQR (CV) : 23 (0.1)	55 distinct values	\ \ \ \ \ \ \ :\ \ \ \ \ . :\ \ \ \ \ : : \ \ \ . : : : \ \ \ \ \ : : : : :
6	body_mass_g\ [integer]	Mean (sd) : 4201.8 (802)\ min < med < max:\ 2700 < 4050 < 6300\ IQR (CV) : 1200 (0.2)	94 distinct values	\ \ \ \ \ :\ \ \ . :\ \ \ : : : :\ \ \ : : : : : . : : : : : : :
7	sex\ [factor]	1\. female\ 2\. male	165 (49.5%)\ 168 (50.5%)	IIIIIIIII \ IIIIIIIIII

```
| 8  | year\             | Mean (sd) : 2008 (0.8)\  | 2007 : 110 (32.0%)\ | IIIIII \
|    | [integer]         | min < med < max:\        | 2008 : 114 (33.1%)\ | IIIIII \
|    |                   | 2007 < 2008 < 2009\      | 2009 : 120 (34.9%)  | IIIIII
|    |                   | IQR (CV) : 2 (0)         |                     |
+----+-------------------+--------------------------+---------------------+----------------
```

3.3.8 Visualizing with `visdat` functions

The `vis_dat()` function from the `visdat` package shows something about the types of variables, providing visual clues about what's inside. The picture below identifies variables types, and missing values.

```
vis_dat(penguins)
```
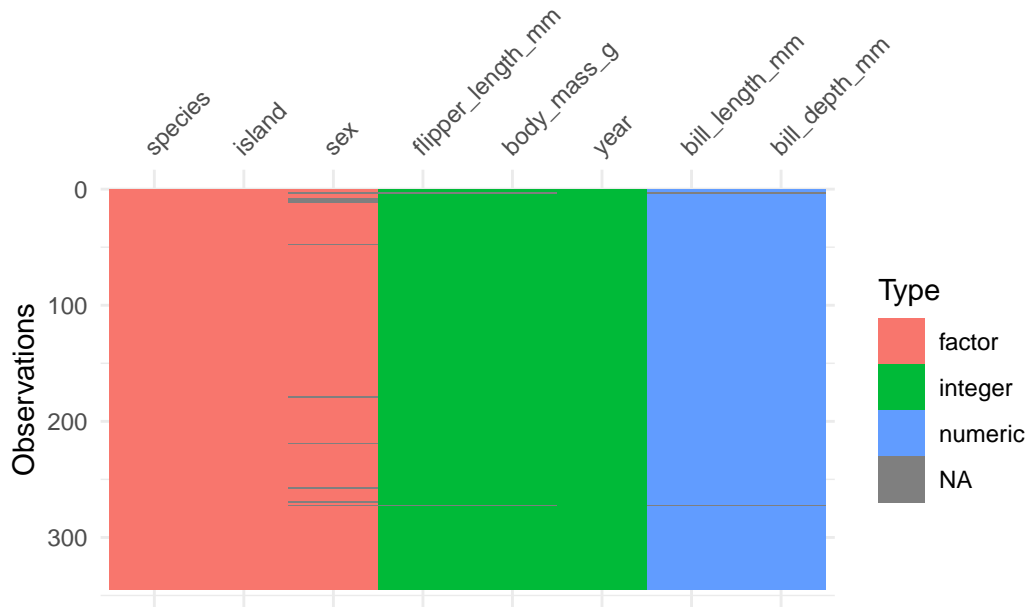

We can explore the missing data further using the `vis_miss` function.

```
vis_miss(penguins)
```
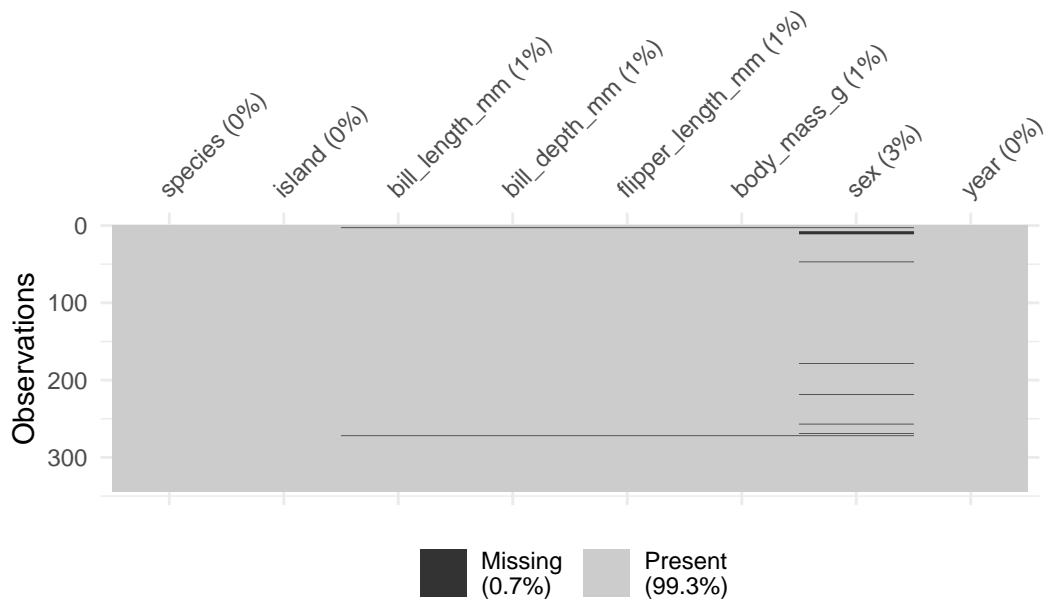
A vignette explaining the use of the `visdat` package is available here.

3.4 Histograms for a Variable

The most common tool we use in producing a graphical summary of a variable, like the penguin's flipper length, is a histogram. Here's one option.

```
ggplot(data = penguins, aes(x = flipper_length_mm)) +
  geom_histogram()
```

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

Warning: Removed 2 rows containing non-finite values (`stat_bin()`).
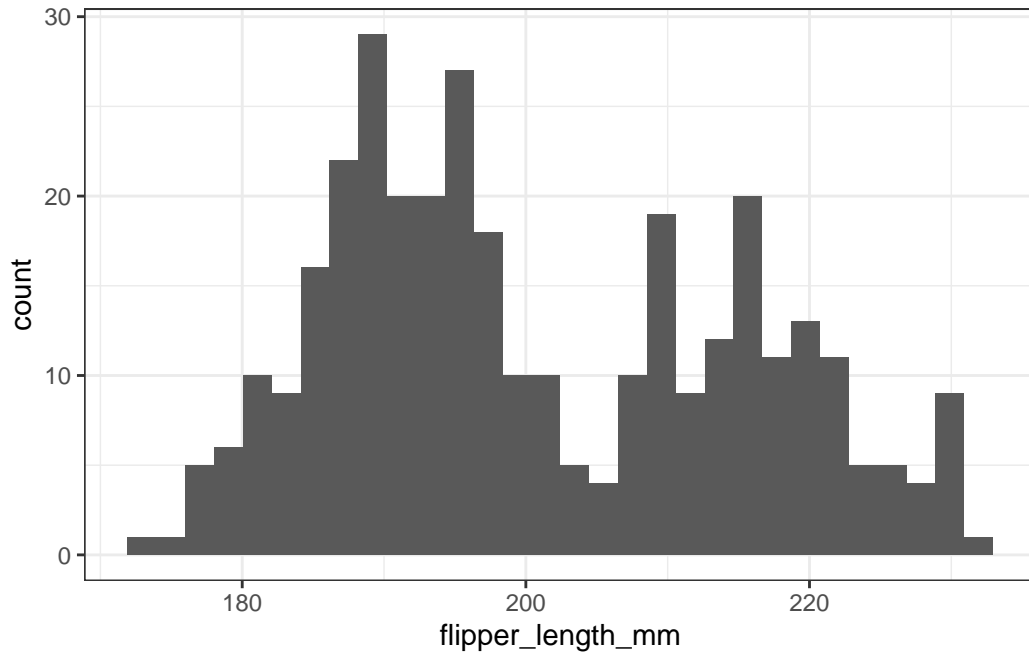
This approach produces two messages that alert us to potential concerns, and a fairly unattractive plot.

This time, we'll first exclude the two penguins without a measured flipper length, and then set the `binwidth` to be 10. How well does that work?

```
penguins2 <-
  penguins |>
  filter(complete.cases(flipper_length_mm))

ggplot(data = penguins2, aes(x = flipper_length_mm)) +
  geom_histogram(binwidth = 10)
```
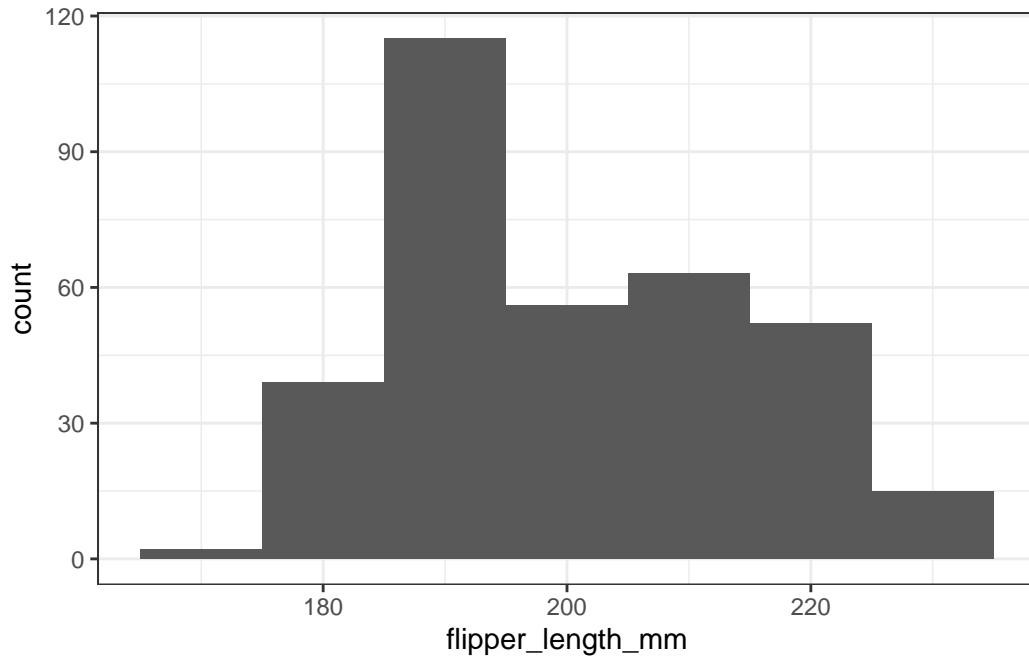
Now we've eliminated the messages, but it would be nice to have some more granularity in the bars (so we'd like a smaller binwidth) and I'd also like to make the bars more clearly separated with colors. I'd also like to add a title. Like this:

```
ggplot(data = penguins2, aes(x = flipper_length_mm)) +
  geom_histogram(binwidth = 5, fill = "orange", col = "navy") +
  labs(title = "Distribution of Flipper Length in 342 Palmer Penguins")
```

Distribution of Flipper Length in 342 Palmer Penguins
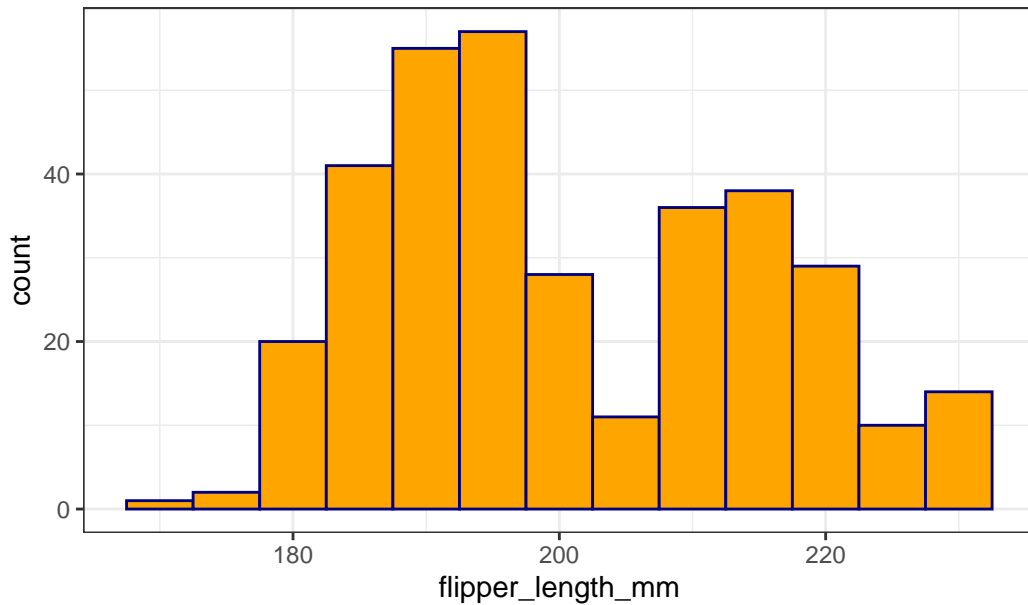
There are some other options for creating a graphical summary of a variable's distribution. For example, we might consider a density plot, as well as a rug plot along the horizontal (X) axis:

```
ggplot(data = penguins2, aes(x = flipper_length_mm)) +
  geom_density(col = "navy") +
  geom_rug(col = "red") +
  labs(title = "Density and Rug Plot of Flipper Length in 342 Palmer Penguins")
```

Density and Rug Plot of Flipper Length in 342 Palmer Penguin
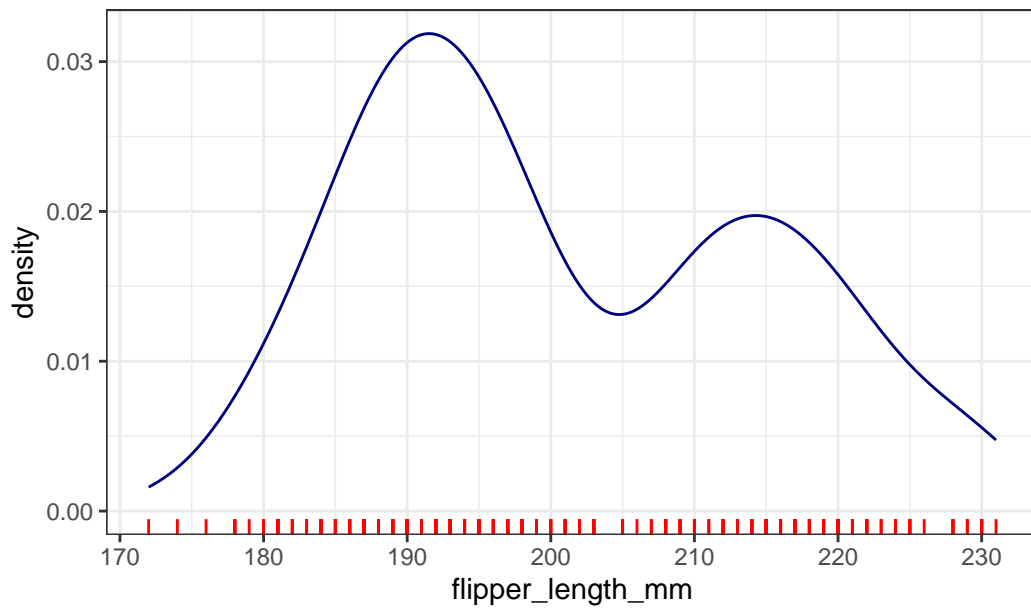
Or perhaps a dotplot would provide a useful look...

```
ggplot(data = penguins2, aes(x = flipper_length_mm)) +
  geom_dotplot(binwidth = 1, fill = "orange", col = "navy") +
  labs(title = "Dot Plot of Flipper Length in 342 Palmer Penguins")
```

Dot Plot of Flipper Length in 342 Palmer Penguins
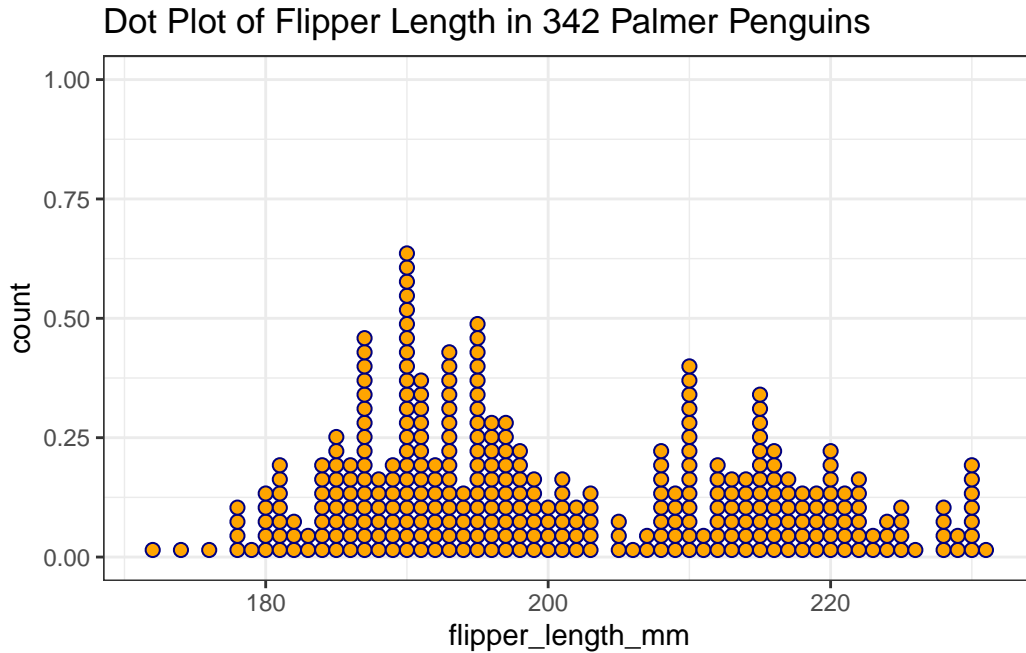
We'll learn about several other approaches to summarizing the distribution of a variable graphically later in the course.

3.5 Comparing Penguins by Species Numerically

We have data from three different species of penguin. Can we compare their flipper lengths numerically, perhaps by calculating the mean flipper length within each species?

```
penguins |>
  group_by(species) |>
  summarise(mean(flipper_length_mm))
```

```
# A tibble: 3 x 2
  species   `mean(flipper_length_mm)`
  <fct>                         <dbl>
1 Adelie                           NA
2 Chinstrap                      196.
3 Gentoo                           NA
```

Well, that's a problem. Looks like we have some missing values. Can we fix that, and also provide some additional summaries, like the sample size (n) and the median and standard

species	n	mean	sd	median
Adelie	151	189.9536	6.539457	190
Chinstrap	68	195.8235	7.131894	196
Gentoo	123	217.1870	6.484976	216

species	min	Q1	median	Q3	max	mean	sd	n	missing
Adelie	32.1	36.75	38.80	40.750	46.0	38.79139	2.663405	151	1
Chinstrap	40.9	46.35	49.55	51.075	58.0	48.83382	3.339256	68	0
Gentoo	40.9	45.30	47.30	49.550	59.6	47.50488	3.081857	123	1

deviation within each species? While we're at it, can we make it prettier, with `kbl()` and `kable_styling()`?

```
penguins |>
  filter(complete.cases(species, flipper_length_mm)) |>
  group_by(species) |>
  summarise(n = n(),
            mean = mean(flipper_length_mm),
            sd = sd(flipper_length_mm),
            median = median(flipper_length_mm)) |>
  kbl() |>
  kable_styling(bootstrap_options = "striped", full_width = FALSE)
```

3.6 Using favstats() from the mosaic package

As we noted previously, we can also use `favstats()` from the `mosaic` package to help us look at the results for a single variable, split into groups by another, like this:

```
mosaic::favstats(bill_length_mm ~ species, data = penguins) |>
  kbl() |>
  kable_styling()
```

One advantage of this approach is that (as you'll note) it handles the missing data in the way we'd probably expect, by restricting the summaries to the complete cases.

3.7 Using tbl_summary() to summarize the tibble

The `tbl_summary()` function from the `gtsummary` package can also do the job of summarizing all of the other variables in the tibble, broken down by species, very nicely.

```
penguins |>
  tbl_summary(by = species)
```

Table printed with `knitr::kable()`, not {gt}. Learn why at
https://www.danieldsjoberg.com/gtsummary/articles/rmarkdown.html
To suppress this message, include `message = FALSE` in code chunk header.

Characteristic	**Adelie**, N = 152	**Chinstrap**, N = 68	**Gentoo**, N = 124
island			
Biscoe	44 (29%)	0 (0%)	124 (100%)
Dream	56 (37%)	68 (100%)	0 (0%)
Torgersen	52 (34%)	0 (0%)	0 (0%)
bill_length_mm	38.8 (36.8, 40.8)	49.6 (46.4, 51.1)	47.3 (45.3, 49.6)
Unknown	1	0	1
bill_depth_mm	18.40 (17.50, 19.00)	18.45 (17.50, 19.40)	15.00 (14.20, 15.70)
Unknown	1	0	1
flipper_length_mm	190 (186, 195)	196 (191, 201)	216 (212, 221)
Unknown	1	0	1
body_mass_g	3,700 (3,350, 4,000)	3,700 (3,488, 3,950)	5,000 (4,700, 5,500)
Unknown	1	0	1
sex			
female	73 (50%)	34 (50%)	58 (49%)
male	73 (50%)	34 (50%)	61 (51%)
Unknown	6	0	5
year			
2007	50 (33%)	26 (38%)	34 (27%)
2008	50 (33%)	18 (26%)	46 (37%)
2009	52 (34%)	24 (35%)	44 (35%)

3.8 Comparing Penguins by Species Graphically

3.8.1 Faceting Histograms with facet_wrap()

We could compare the distributions of the flipper lengths across the three species, by creating
a set of faceted histograms, like so...

```
penguins3 <-
  penguins |>
  filter(complete.cases(flipper_length_mm, species))
```

```
ggplot(data = penguins3, aes(x = flipper_length_mm, fill = species)) +
  geom_histogram(binwidth = 5, col = "white") +
  facet_wrap(~ species) +
  labs(title = "Distribution of Flipper Length in Palmer Penguins, by Species")
```
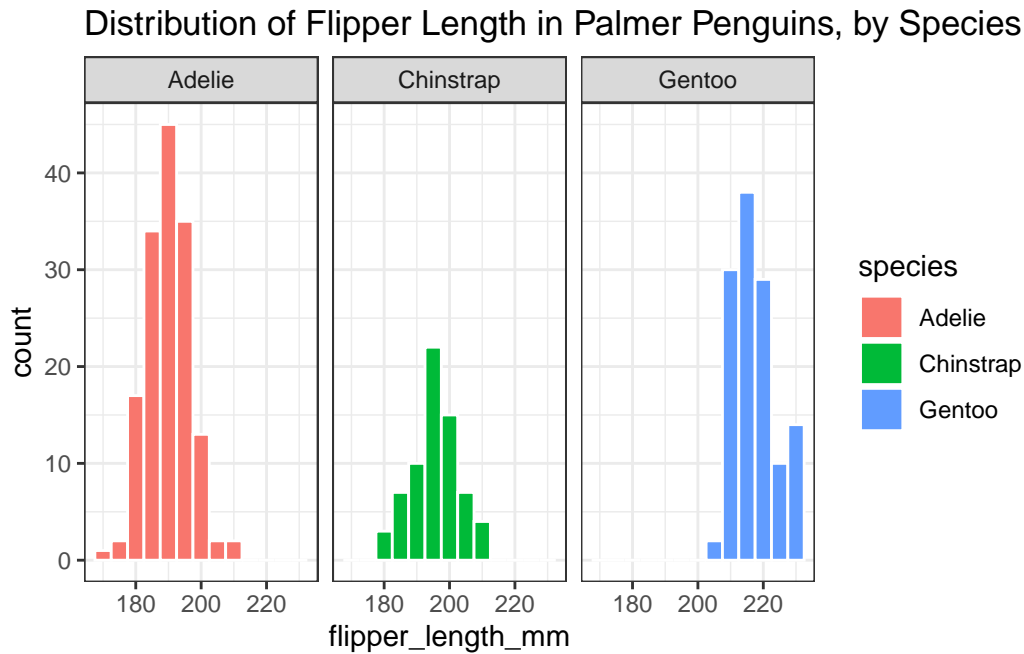

We might add in the command

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

to eliminate the redundant legend on the right-hand side of the plot.

3.8.2 Using facet_grid() instead

The `facet_wrap()` approach has created three histograms, spread horizontally. Alternatively, we could plot the species vertically using `facet_grid()`, which clearly shows which species produces the penguins with the larger flipper lengths, especially if we reduce the width of the bins a bit.

```
ggplot(data = penguins3, aes(x = flipper_length_mm, fill = species)) +
  geom_histogram(binwidth = 2, col = "white") +
  facet_grid(species ~ .) +
  guides(fill = "none") +
  labs(title = "Distribution of Flipper Length in Palmer Penguins, by Species")
```


Distribution of Flipper Length in Palmer Penguins, by Species
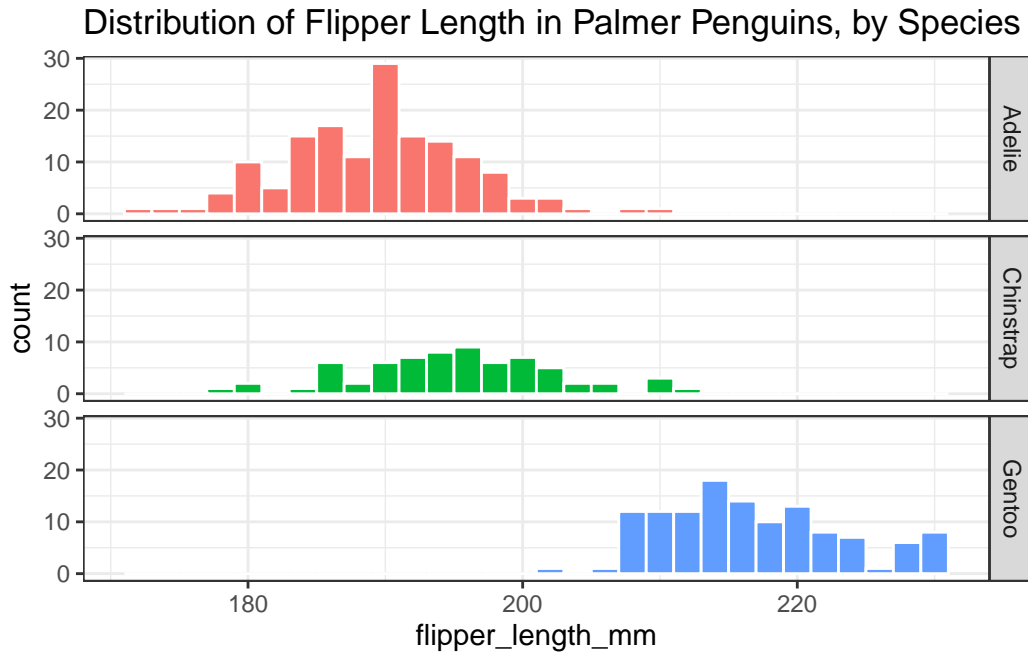
We'll use facets like this all the time in what follows.

3.8.3 Boxplots

Another very common tool we'll use for looking simultaneously at the distributions of a variable across two or more categories is a boxplot. More on this later, but here's one example of what this might look like.

```
ggplot(data = penguins3, aes(x = flipper_length_mm, y = species,
                             fill = species)) +
  geom_boxplot() +
  guides(fill = "none") +
  labs(title = "Distribution of Flipper Length in Palmer Penguins, by Species")
```

Distribution of Flipper Length in Palmer Penguins, by Spec
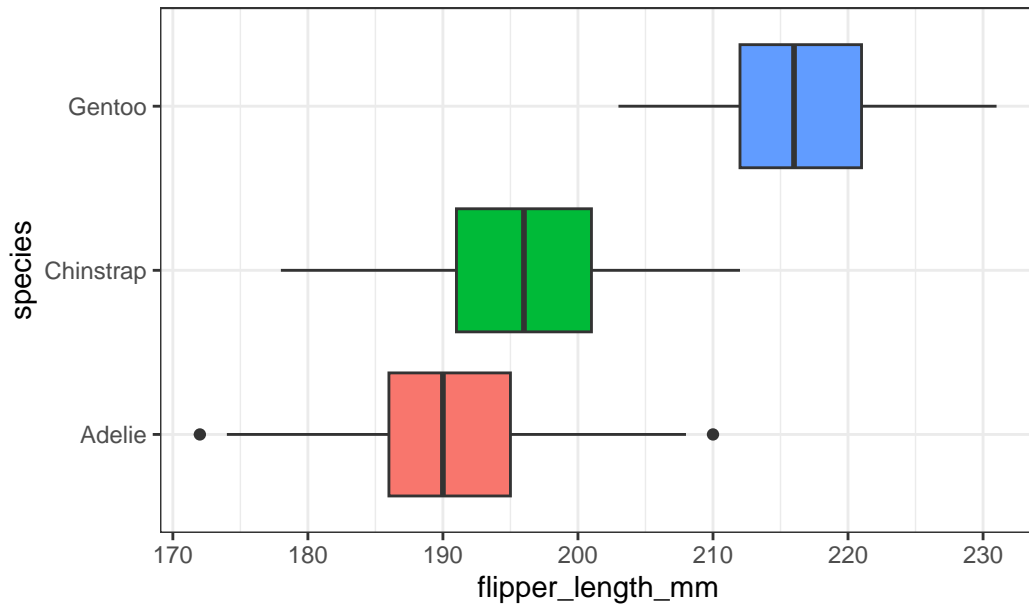
3.8.4 Adding Violins

And here's a somewhat fancier version, including a violin plot, and with the coordinates flipped so the plots are shown vertically rather than horizontally.

```
ggplot(data = penguins3, aes(x = flipper_length_mm, y = species)) +
  geom_violin(aes(col = species)) +
  geom_boxplot(aes(fill = species), width = 0.3) +
  guides(col = "none", fill = "none") +
  coord_flip() +
  labs(title = "Augmented Boxplot of Flipper Length in Penguins by Species")
```

Augmented Boxplot of Flipper Length in Penguins by Species
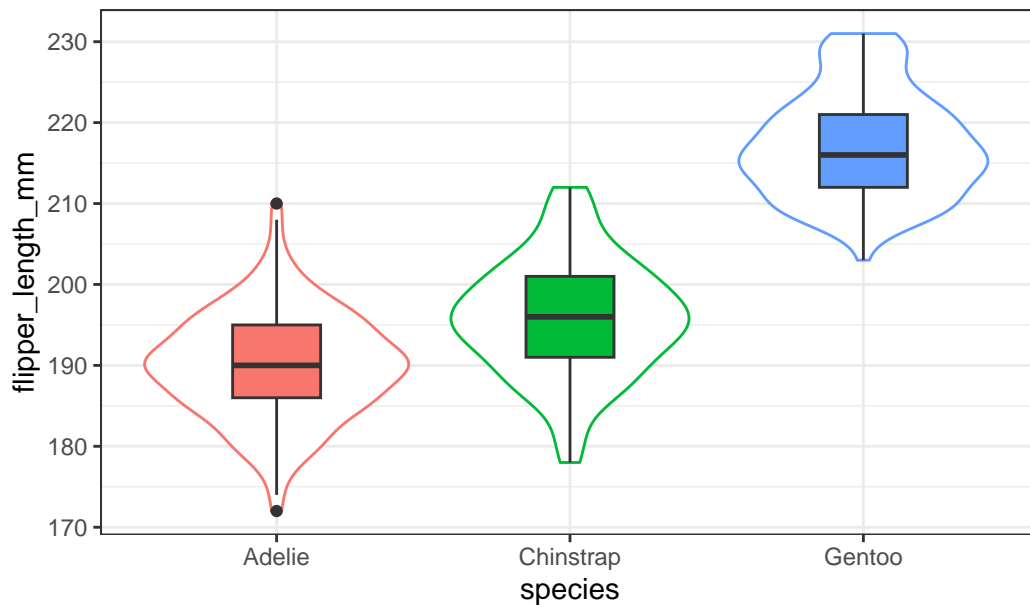
3.8.5 Letter-Value Plots (Boxplots for Large Data)

We might also consider a letter-value plot, using the `geom_lv()` function from the `lvplot` package in R, although I rarely use such a plot unless I have at least 1000 observations to work with.

```
ggplot(data = penguins3, aes(x = species, y = flipper_length_mm)) +
  geom_lv(aes(fill=..LV..)) + scale_fill_brewer() +
  labs(title = "Letter-Value Plot of Flipper Length in Penguins by Species")
```

```
Warning: The dot-dot notation (`..LV..`) was deprecated in ggplot2 3.4.0.
i Please use `after_stat(LV)` instead.
```

Letter–Value Plot of Flipper Length in Penguins by Species
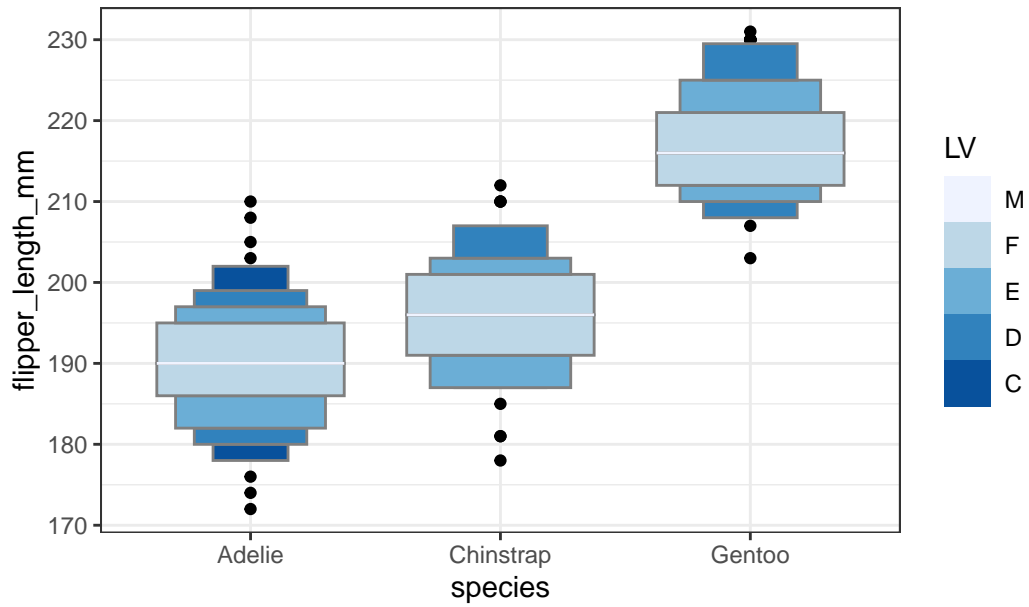
3.9 Coming Up

You're probably tiring of the penguins now. Next, we'll look at some data on people, taken from the National Health and Nutrition Examination Survey, or NHANES.

4 NHANES Data

Next, we'll explore some data from the US National Health and Nutrition Examination Survey, often referred to as NHANES.

4.1 Setup: Packages Used Here

```r
knitr::opts_chunk$set(comment = NA)

library(NHANES)
library(naniar)
library(kableExtra)
library(tidyverse)

theme_set(theme_bw())
```

4.2 The NHANES data: A First Sample

The `NHANES` package provides a sample of 10,000 NHANES responses from the 2009-10 and 2011-12 administrations, in a tibble also called `NHANES`. We can obtain the dimensions of this tibble with the `dim()` function.

```r
NHANES
```

```
# A tibble: 10,000 x 76
     ID SurveyYr Gender   Age AgeDecade AgeMonths Race1 Race3 Education
  <int> <fct>    <fct>  <int> <fct>         <int> <fct> <fct> <fct>
1 51624 2009_10  male      34 " 30-39"        409 White <NA>  High School
2 51624 2009_10  male      34 " 30-39"        409 White <NA>  High School
3 51624 2009_10  male      34 " 30-39"        409 White <NA>  High School
4 51625 2009_10  male       4 " 0-9"           49 Other <NA>  <NA>
5 51630 2009_10  female    49 " 40-49"        596 White <NA>  Some College
```

```
 6 51638 2009_10  male        9 " 0-9"        115 White <NA>  <NA>
 7 51646 2009_10  male        8 " 0-9"        101 White <NA>  <NA>
 8 51647 2009_10  female     45 " 40-49"      541 White <NA>  College Grad
 9 51647 2009_10  female     45 " 40-49"      541 White <NA>  College Grad
10 51647 2009_10  female     45 " 40-49"      541 White <NA>  College Grad
# i 9,990 more rows
# i 67 more variables: MaritalStatus <fct>, HHIncome <fct>, HHIncomeMid <int>,
#   Poverty <dbl>, HomeRooms <int>, HomeOwn <fct>, Work <fct>, Weight <dbl>,
#   Length <dbl>, HeadCirc <dbl>, Height <dbl>, BMI <dbl>,
#   BMICatUnder20yrs <fct>, BMI_WHO <fct>, Pulse <int>, BPSysAve <int>,
#   BPDiaAve <int>, BPSys1 <int>, BPDia1 <int>, BPSys2 <int>, BPDia2 <int>,
#   BPSys3 <int>, BPDia3 <int>, Testosterone <dbl>, DirectChol <dbl>, ...
```

We see that we have 10000 rows and 76 columns in the `NHANES` tibble. For more on what makes
a particular data frame into a tibble, and why we'd want such a thing, you might be interested
in the Tibbles section of Hadley Wickham and Grolemund (2023). Essentially, tibbles are data
frames that are easier and more predictable to work with.

4.3 Sampling NHANES Adults

Suppose we want to take this NHANES tibble, and use it to generate a sample describing 750
unique (distinct) adult subjects who completed the 2011-12 version of the survey when they
were between the ages of 21 and 64.

4.3.1 Creating a Temporary, Cleaner Tibble

I'll start by describing the plan we will use to create a new tibble called `nh_temp` from which
we will eventually build our final sample. In particular, let me lay out the steps I will use to
create the `nh_temp` frame from the original `NHANES` tibble available in the R package called
`NHANES`.

1. We'll **filter** the original NHANES tibble to include only the responses from the 2011-12
   administration of the survey. This will cut the sample in half, from 10,000 rows to 5,000.

2. We'll then **filter** again to restrict the sample to adults whose age is at least 21 and also
   less than 65. I'll do this because I want to avoid problems with including both children
   and adults in my sample, and because I also want to focus on the population of people
   in the US who are usually covered by private insurance from their job, or by Medicaid
   insurance from the government, rather than those covered by Medicare.

3. What is listed in the NHANES tibble as `Gender` should be more correctly referred to as `Sex`. `Sex` is a biological feature of an individual, while `Gender` is a social construct. This is an important distinction, so I'll change the name of the variable.

4. We'll also rename three other variables, specifically we'll use `Race` to describe the `Race3` variable in the original NHANES tibble, as well as `SBP` to refer to the average systolic blood pressure, which is specified as `BPSysAve`, and `DBP` to refer to the average diastolic blood pressure, which is specified as `BPDiaAve`.

5. Having accomplished the previous four steps, we'll then *select* the variables we want to keep in the sample. (We use *select* for choosing variables or columns in the tibble, and *filter* for selecting subjects or rows.) The sixteen variables we will select are: ID, Sex, Age, Height, Weight, Race, Education, BMI, SBP, DBP, Pulse, PhysActive, Smoke100, SleepTrouble, MaritalStatus and HealthGen.

6. The original NHANES tibble includes some subjects (rows) multiple times in an effort to incorporate some of the sampling weights used in most NHANES analyses. For our purposes, though, we'd like to only include each subject one time. We use the `distinct()` function to limit the tibble to completely unique subjects (so that, for example, we don't wind up with two or more rows that have the same ID number.)

Here is the code I used to complete the six steps listed above and create the `nh_temp` tibble.

```
nh_temp <- NHANES |>
    filter(SurveyYr == "2011_12") |>
    filter(Age >= 21 & Age < 65) |>
    rename(Sex = Gender, Race = Race3, SBP = BPSysAve, DBP = BPDiaAve) |>
    select(ID, Sex, Age, Height, Weight, Race, Education, BMI, SBP, DBP,
            Pulse, PhysActive, Smoke100, SleepTrouble,
            MaritalStatus, HealthGen) |>
    distinct()
```

The resulting `nh_temp` tibble has 1700 rows and 16 columns.

```
nh_temp
```

```
# A tibble: 1,700 x 16
      ID Sex      Age Height Weight Race   Education      BMI   SBP   DBP Pulse
   <int> <fct>  <int>  <dbl>  <dbl> <fct>  <fct>        <dbl> <int> <int> <int>
 1 62172 female    43   172    98.6 Black  High School   33.3   103    72    80
 2 62176 female    34   172.   68.7 White  College Gr~   23.3   107    69    92
 3 62180 male      35   179.   89   White  College Gr~   27.9   107    66    66
 4 62199 male      57   186    96.9 White  College Gr~   28     110    65    84
 5 62205 male      28   171.   84.8 White  College Gr~   28.9   122    87    70
```

```
 6 62206 female     35  167.   81.5 White    Some Colle~  29.1   106    50    58
 7 62208 male       38  169.   63.2 Hispanic Some Colle~  22.2   105    59    52
 8 62209 female     62  143.   53.5 Mexican  8th Grade    26     108    57    72
 9 62220 female     31  167.  113.  Black    College Gr~  40.4   120    71    62
10 62222 male       32  179    80.1 White    College Gr~  25     104    73    78
# i 1,690 more rows
# i 5 more variables: PhysActive <fct>, Smoke100 <fct>, SleepTrouble <fct>,
#   MaritalStatus <fct>, HealthGen <fct>
```

4.3.2 Sampling `nh_temp` to obtain our `nh_adult750` sample

Having established the `nh_temp` tibble, we now select a random sample of 750 adults from the 1700 available responses.

- We will use the `set.seed()` function in R to set a random numerical seed to ensure that if you redo this work, you will obtain the same sample.

  - Setting a seed is an important part of being able to replicate the work later when sampling is involved. If you and I use the same seed, we should get the same sample.

- Then we will use the `slice_sample()` function to actually draw the random sample, without replacement.

  - "Without replacement" means that once we've selected a particular subject, we won't select them again.

```
set.seed(431002)
# use set.seed to ensure that we all get the same random sample

nh_adult750 <- slice_sample(nh_temp, n = 750, replace = F)

nh_adult750
```

```
# A tibble: 750 x 16
      ID Sex      Age Height Weight Race     Education     BMI   SBP   DBP Pulse
   <int> <fct>  <int>  <dbl>  <dbl> <fct>    <fct>       <dbl> <int> <int> <int>
 1 68648 female     30  181.   67.1 White    College Gr~  20.4   103    59    78
 2 67200 male       30  180.   86.6 White    College Gr~  26.7   113    68    70
 3 66404 female     35  160.   71.1 White    College Gr~  27.8   116    80    68
 4 70535 male       40  177.   82   White    College Gr~  26.3   130    79    68
 5 65308 female     54  151.   60.6 Mexican  8th Grade    26.6   130    64    48
 6 67392 male       41  171.   90.7 Hispanic College Gr~  31.2   124    82    68
 7 63218 male       35  163.   81   Mexican  8th Grade    30.3   128    96    82
```

```
 8 65879 female    32   160.    66.4 Mexican   College Gr~  25.9    104    70     78
 9 63617 male      29   189.    83.3 White     College Gr~  23.2    105    72     76
10 64720 male      29   174.    62.3 Black     College Gr~  20.6    127    60     84
# i 740 more rows
# i 5 more variables: PhysActive <fct>, Smoke100 <fct>, SleepTrouble <fct>,
#   MaritalStatus <fct>, HealthGen <fct>
```

The `nh_adult750` tibble now includes 750 rows (observations) on 16 variables (columns). Essentially, we have 16 pieces of information on each of 750 adult NHANES subjects who were included in the 2011-12 panel.

4.3.3 Summarizing the Data's Structure

We can identify the number of rows and columns in a data frame or tibble with the `dim` function.

```
dim(nh_adult750)
```

```
[1] 750  16
```

The `str` function provides a lot of information about the structure of a data frame or tibble.

```
str(nh_adult750)
```

```
tibble [750 x 16] (S3: tbl_df/tbl/data.frame)
 $ ID           : int [1:750] 68648 67200 66404 70535 65308 67392 63218 65879 63617 64720 ..
 $ Sex          : Factor w/ 2 levels "female","male": 1 2 1 2 1 2 2 1 2 2 ...
 $ Age          : int [1:750] 30 30 35 40 54 41 35 32 29 29 ...
 $ Height       : num [1:750] 181 180 160 177 151 ...
 $ Weight       : num [1:750] 67.1 86.6 71.1 82 60.6 90.7 81 66.4 83.3 62.3 ...
 $ Race         : Factor w/ 6 levels "Asian","Black",..: 5 5 5 5 4 3 4 4 5 2 ...
 $ Education    : Factor w/ 5 levels "8th Grade","9 - 11th Grade",..: 5 5 5 5 1 5 1 5 5 5 ..
 $ BMI          : num [1:750] 20.4 26.7 27.8 26.3 26.6 31.2 30.3 25.9 23.2 20.6 ...
 $ SBP          : int [1:750] 103 113 116 130 130 124 128 104 105 127 ...
 $ DBP          : int [1:750] 59 68 80 79 64 82 96 70 72 60 ...
 $ Pulse        : int [1:750] 78 70 68 68 48 68 82 78 76 84 ...
 $ PhysActive   : Factor w/ 2 levels "No","Yes": 1 2 2 1 1 2 1 2 2 2 ...
 $ Smoke100     : Factor w/ 2 levels "No","Yes": 1 2 1 2 2 1 2 1 2 2 ...
 $ SleepTrouble : Factor w/ 2 levels "No","Yes": 2 1 1 1 1 1 1 1 2 1 ...
 $ MaritalStatus: Factor w/ 6 levels "Divorced","LivePartner",..: 3 4 3 3 2 3 3 3 3 2 ...
 $ HealthGen    : Factor w/ 5 levels "Excellent","Vgood",..: 1 1 1 2 4 3 NA 1 2 4 ...
```

To see the first few observations, use `head`, and to see the last few, try `tail`…

```
tail(nh_adult750, 5) # shows the last five observations in the data set
```

```
# A tibble: 5 x 16
     ID Sex        Age Height Weight Race  Education      BMI   SBP   DBP Pulse
  <int> <fct>    <int>  <dbl>  <dbl> <fct> <fct>        <dbl> <int> <int> <int>
1 63924 female      29  165.   113.  Black High School   41.9    98    56    74
2 69825 female      43  164.    63.3 White College Grad  23.7   122    83    88
3 68109 male        45  170.    78.7 Black High School   27.1   140    79   102
4 64598 female      60  158     74.5 White Some College  29.8   137    80    78
5 64048 female      54  161.    67.5 White Some College  26.2   121    87    72
# i 5 more variables: PhysActive <fct>, Smoke100 <fct>, SleepTrouble <fct>,
#   MaritalStatus <fct>, HealthGen <fct>
```

4.3.4 What are the variables?

We can use the `glimpse` function to get a short preview of the data.

```
glimpse(nh_adult750)
```

```
Rows: 750
Columns: 16
$ ID            <int> 68648, 67200, 66404, 70535, 65308, 67392, 63218, 65879, ~
$ Sex           <fct> female, male, female, male, female, male, male, female, ~
$ Age           <int> 30, 30, 35, 40, 54, 41, 35, 32, 29, 29, 64, 28, 31, 59, ~
$ Height        <dbl> 181.3, 180.2, 159.8, 176.6, 150.9, 170.6, 163.4, 160.2, ~
$ Weight        <dbl> 67.1, 86.6, 71.1, 82.0, 60.6, 90.7, 81.0, 66.4, 83.3, 62~
$ Race          <fct> White, White, White, White, Mexican, Hispanic, Mexican, ~
$ Education     <fct> College Grad, College Grad, College Grad, College Grad, ~
$ BMI           <dbl> 20.4, 26.7, 27.8, 26.3, 26.6, 31.2, 30.3, 25.9, 23.2, 20~
$ SBP           <int> 103, 113, 116, 130, 130, 124, 128, 104, 105, 127, 128, 1~
$ DBP           <int> 59, 68, 80, 79, 64, 82, 96, 70, 72, 60, 74, 76, 82, 66, ~
$ Pulse         <int> 78, 70, 68, 68, 48, 68, 82, 78, 76, 84, 62, 56, 78, 66, ~
$ PhysActive    <fct> No, Yes, Yes, No, No, Yes, No, Yes, Yes, Yes, Yes, No, N~
$ Smoke100      <fct> No, Yes, No, Yes, Yes, No, Yes, No, Yes, Yes, No, No, Ye~
$ SleepTrouble  <fct> Yes, No, No, No, No, No, No, No, Yes, No, No, Yes, No, Y~
$ MaritalStatus <fct> Married, NeverMarried, Married, Married, LivePartner, Ma~
$ HealthGen     <fct> Excellent, Excellent, Excellent, Vgood, Fair, Good, NA, ~
```

The variables we have collected are described in the brief table below[1].

Variable	Description	Sample Values
ID	a numerical code identifying the subject	68648, 67200
Sex	sex of subject (2 levels)	female, male
Age	age (years) at screening of subject	30, 35
Height	height (in cm) at screening of subject	181.3, 180.2
Weight	weight (in kg) at screening of subject	67.1, 86.6
Race	reported race of subject (6 levels)	White, Black
Education	educational level of subject (5 levels)	College Grad, High School
BMI	body-mass index, in kg/m^2	20.4, 26.7
SBP	systolic blood pressure in mm Hg	103, 113
DBP	diastolic blood pressure in mm Hg	59, 68
Pulse	60 second pulse rate in beats per minute	78, 70
PhysActive	Moderate or vigorous-intensity sports?	Yes, No
Smoke100	Smoked at least 100 cigarettes lifetime?	Yes, No
SleepTrouble	Told a doctor they have trouble sleeping?	Yes, No
MaritalStatus	Marital Status	Married, Divorced
HealthGen	Self-report general health rating (5 levels)	Vgood, Fair

The levels for the multi-categorical variables are:

- **Race**: Mexican, Hispanic, White, Black, Asian, or Other.
- **Education**: 8th Grade, 9 - 11th Grade, High School, Some College, or College Grad.
- **MaritalStatus**: Married, Widowed, Divorced, Separated, NeverMarried or LivePartner (living with partner).
- **HealthGen**: Excellent, Vgood, Good, Fair or Poor.

Some details can be obtained using the `summary` function, or any of the other approaches we saw used with the `penguins` data earlier.

```
summary(nh_adult750)
```

```
      ID             Sex          Age           Height          Weight
 Min.   :62206   female:388   Min.   :21.00   Min.   :142.4   Min.   : 39.30
 1st Qu.:64277   male  :362   1st Qu.:30.00   1st Qu.:161.8   1st Qu.: 67.40
```

---

[1]Descriptions are adapted from the ?NHANES help file. Remember that what NHANES lists as Gender is captured here as Sex, and similarly Race3, BPSysAve and BPDiaAve from NHANES are here listed as Race, SBP and DBP.

```
Median :66925                   Median :40.00   Median :168.9   Median : 80.00
Mean   :66936                   Mean   :40.82   Mean   :168.9   Mean   : 83.16
3rd Qu.:69414                   3rd Qu.:51.00   3rd Qu.:175.7   3rd Qu.: 95.30
Max.   :71911                   Max.   :64.00   Max.   :200.4   Max.   :198.70
                                                NA's   :5       NA's   :5
      Race              Education          BMI            SBP
Asian   : 70    8th Grade     : 50   Min.   :16.70   Min.   : 83.0
Black   :128    9 - 11th Grade: 76   1st Qu.:24.20   1st Qu.:108.0
Hispanic: 63    High School   :143   Median :27.90   Median :118.0
Mexican : 80    Some College  :241   Mean   :29.08   Mean   :118.8
White   :393    College Grad  :240   3rd Qu.:32.10   3rd Qu.:127.0
Other   : 16                         Max.   :80.60   Max.   :209.0
                                     NA's   :5       NA's   :33
      DBP              Pulse          PhysActive Smoke100   SleepTrouble
Min.   :  0.00   Min.   : 40.00   No :326    No :453    No :555
1st Qu.: 66.00   1st Qu.: 66.00   Yes:424    Yes:297    Yes:195
Median : 73.00   Median : 72.00
Mean   : 72.69   Mean   : 73.53
3rd Qu.: 80.00   3rd Qu.: 80.00
Max.   :108.00   Max.   :124.00
NA's   :33       NA's   :32
    MaritalStatus        HealthGen
Divorced    : 78    Excellent: 84
LivePartner : 70    Vgood    :197
Married     :388    Good     :252
NeverMarried:179    Fair     :104
Separated   : 19    Poor     : 14
Widowed     : 16    NA's     : 99
```

Note the appearance of `NA's` (indicating missing values) in some columns, and that some variables are summarized by a list of their (categorical) values (with counts) and some (quantitative/numeric) variables are summarized with a minimum, quartiles and means.

4.4 Counting Missing Values

The `summary()` command counts the number of missing observations in each variable, but sometimes you want considerably more information.

We can use some functions from the `naniar` package to learn useful things about the missing data in our `nh_adult750` sample. (Recall that we could also use the `vis_miss()` function from

the `visdat` package, as we saw earlier with the `penguins` to get some of this information, but the `naniar` approach provides more exploratory tools.)

The `miss_var_table` command provides a table of the number of variables with 0, 1, 2, up to n, missing values and the percentage of the total number of variables those variables make up.

```
miss_var_table(nh_adult750)
```

```
# A tibble: 5 x 3
  n_miss_in_var n_vars pct_vars
          <int>  <int>    <dbl>
1             0      9    56.2
2             5      3    18.8
3            32      1     6.25
4            33      2    12.5
5            99      1     6.25
```

So, for instance, we have 9 variables with no missing data, and that constitutes 56.25% of the 16 variables in our `nh_adult750` data.

The `miss_var_summary()` function tabulates the number, percent missing, and cumulative sum of missing of each variable in our tibble, in order of most to least missing values.

```
miss_var_summary(nh_adult750) |>
  kbl() |>
  kable_styling(full_width = FALSE, position = "center")
```

So, for example, the HealthGen variable is the one missing more of our data than anything else within the `nh_adult750` tibble.

A graph of this information is available, as well.

```
gg_miss_var(nh_adult750)
```

variable	n_miss	pct_miss
HealthGen	99	13.2000000
SBP	33	4.4000000
DBP	33	4.4000000
Pulse	32	4.2666667
Height	5	0.6666667
Weight	5	0.6666667
BMI	5	0.6666667
ID	0	0.0000000
Sex	0	0.0000000
Age	0	0.0000000
Race	0	0.0000000
Education	0	0.0000000
PhysActive	0	0.0000000
Smoke100	0	0.0000000
SleepTrouble	0	0.0000000
MaritalStatus	0	0.0000000
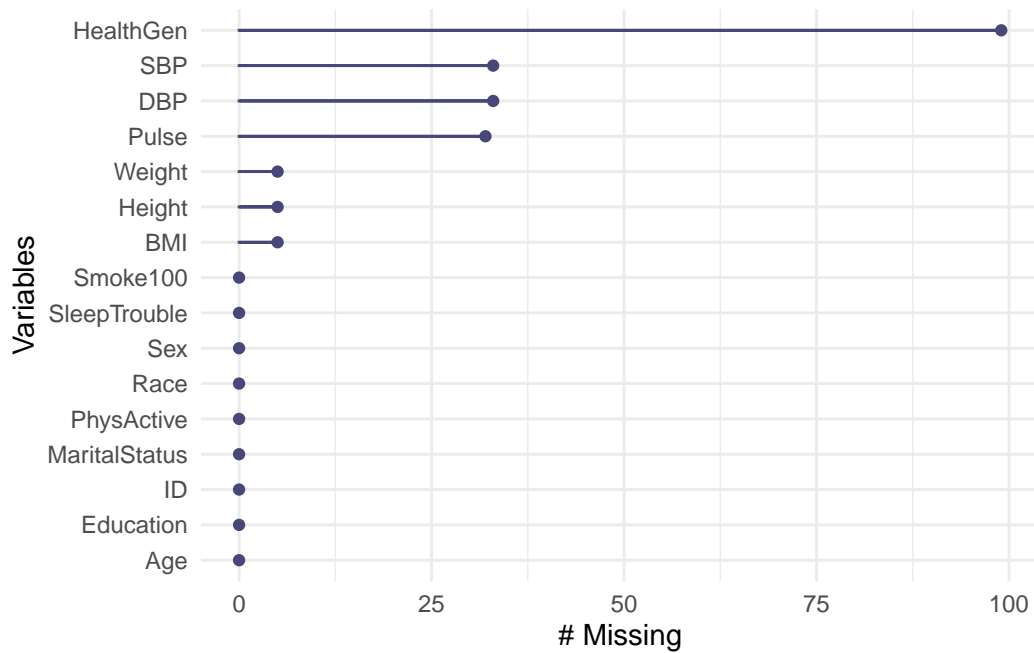
I'll note that there are also functions to count the number of missing observations by case (observation) rather than variable. For example, we can use `miss_case_table`.

```
miss_case_table(nh_adult750)
```

```
# A tibble: 6 x 3
  n_miss_in_case n_cases pct_cases
          <int>   <int>    <dbl>
1              0     636    84.8
2              1      78    10.4
3              3      15     2
4              4      19     2.53
5              6       1     0.133
6              7       1     0.133
```

Now we see that 636 observations, or 84.8% of all cases have no missing data.

We can use `miss_case_summary()` to identify cases with missing data, as well.

```
miss_case_summary(nh_adult750)
```

```
# A tibble: 750 x 3
     case n_miss pct_miss
    <int>  <int>    <dbl>
 1    342      7     43.8
 2    606      6     37.5
 3    157      4     25
 4    169      4     25
 5    204      4     25
 6    234      4     25
 7    323      4     25
 8    415      4     25
 9    478      4     25
10    483      4     25
# i 740 more rows
```

4.5 Sampling 500 Complete Cases

If we wanted a sample of exactly 750 subjects with complete data, we would have needed to add a step in the development of our `nh_temp` tibble to filter for complete cases.

```
nh_temp2 <- NHANES |>
    filter(SurveyYr == "2011_12") |>
    filter(Age >= 21 & Age < 65) |>
    rename(Sex = Gender, Race = Race3, SBP = BPSysAve, DBP = BPDiaAve) |>
    select(ID, Sex, Age, Height, Weight, Race, Education, BMI, SBP, DBP,
           Pulse, PhysActive, Smoke100, SleepTrouble,
           MaritalStatus, HealthGen) |>
    distinct() |>
    na.omit()
```

Let's check that this new tibble has no missing data.

```
miss_var_table(nh_temp2)
```

```
# A tibble: 1 x 3
  n_miss_in_var n_vars pct_vars
          <int>  <int>    <dbl>
1             0     16      100
```

OK. Now, let's create a second sample, called `nh_adult500cc`, where now, we will select 500 adults with complete data on all of the variables of interest, and using a different random seed. The `cc` here stands for complete cases.

```
set.seed(431003)
# use set.seed to ensure that we all get the same random sample

nh_adult500cc <- slice_sample(nh_temp2, n = 500, replace = F)

nh_adult500cc
```

```
# A tibble: 500 x 16
      ID Sex      Age Height Weight Race     Education     BMI   SBP   DBP Pulse
   <int> <fct>  <int>  <dbl>  <dbl> <fct>    <fct>       <dbl> <int> <int> <int>
 1 64079 female    25   159.   86.2 Hispanic Some Colle~  34.2   120    67    84
 2 64374 female    52   169    65.5 Asian    College Gr~  22.9    92    58    60
 3 71875 male      42   182.   94.1 Black    College Gr~  28.5   102    63    76
 4 66396 female    46   161.  107.  Asian    8th Grade    41.2   111    61    70
 5 64315 female    52   161.   64.5 White    9 - 11th G~  24.9   130    69    68
 6 64015 male      32   168.   82.3 Mexican  Some Colle~  29      119    79    70
 7 63590 male      21   181.   98.3 Black    Some Colle~  29.9   121    67    58
 8 70893 female    30   171.   65.7 White    9 - 11th G~  22.5   104    75    74
```

```
 9 70828 male       26   178.  100.  White    Some Colle~  31.5   119    77    66
10 67930 male       59   172.   91.7 Mexican  College Gr~  31     127    85    66
# i 490 more rows
# i 5 more variables: PhysActive <fct>, Smoke100 <fct>, SleepTrouble <fct>,
#   MaritalStatus <fct>, HealthGen <fct>
```

4.6 Saving our Samples in `.Rds` files

We'll save the `nh_adult750` and `nh_adult500cc` samples to use in later parts of the notes. To do this, we'll save them as `.Rds` files, which are files we can read directly into R with the `read_rds` command, and which will have some advantages for us later on.

```
write_rds(nh_adult750, file = "data/nh_adult750.Rds")
write_rds(nh_adult500cc, file = "data/nh_adult500cc.Rds")
```

You will also find these `.Rds` files as part of the 431-data repository for the course.

4.7 Coming Up

Next, we'll introduce some new ways of thinking about data and variables as we load, explore and learn about some of the variables in our two NHANES samples.

5 Types of Data

5.1 Setup: Packages Used Here

```r
knitr::opts_chunk$set(comment = NA)

library(janitor)
library(gtsummary)
library(kableExtra)
library(tidyverse)

theme_set(theme_bw())
```

We'll also use the `describe()` function from the `psych` package in what follows, but I won't load the whole `psych` package here.

5.2 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.

5.3 Reading in the "Complete Cases" Sample

Let's begin by loading into the `nh_500cc` tibble the information from the `nh_adult500cc.Rds` file we created in Section 4.5. Notice that I am simplifying the name of the tibble, to save me some typing.

```
nh_500cc <- read_rds("data/nh_adult500cc.Rds")
```

One obvious hurdle we'll avoid for the moment is what to do about missing data, since the `nh_500cc` data are specifically drawn from complete responses. Working with complete cases only can introduce bias to our estimates and visualizations, so it will be necessary in time to address what we should do when a complete-case analysis isn't a good choice. We'll return to this issue later.

5.4 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

79

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.

5.5 Quantitative Variables in `nh_500cc`

Here's a list of the variables contained in our `nh_500cc` tibble.

```
names(nh_500cc)
```

```
 [1] "ID"          "Sex"           "Age"            "Height"
 [5] "Weight"      "Race"          "Education"      "BMI"
 [9] "SBP"         "DBP"           "Pulse"          "PhysActive"
[13] "Smoke100"    "SleepTrouble"  "MaritalStatus"  "HealthGen"
```

The `nh_500cc` data includes seven quantitative variables, including `Age`, `Height`, `Weight`, BMI, SBP, DBP and `Pulse`.

- We know these are quantitative variables because they have units:
    - `Age` in years, `Height` in centimeters, `Weight` in kilograms,
    - BMI in kg/m^2, the BP measurements in mm Hg, and `Pulse` in beats per minute.

Let's summarize them with the `describe()` function from the `psych` package.

```
nh_500cc |>
    select(Age, Height, Weight, BMI, SBP, DBP, Pulse) |>
    psych::describe() |>
    kbl() |>
    kable_styling()
```

As an alternative, we could use `tbl_summary()` from the `gtsummary` package, as well. The approach below works nicely for producing a mean, standard deviation, and five-number summary for each of the variables we've identified as quantitative.

- Quantitative variables lend themselves to many of the summaries we will discuss, like means, quantiles, and our various measures of spread, like the standard deviation or inter-quartile range. They also have at least a chance to follow the Normal distribution.

80

	vars	n	mean	sd	median	trimmed	mad	min	max	range	s
Age	1	500	41.6060	12.803853	42.00	41.48000	16.30860	21.0	64.0	43.0	0.0469
Height	2	500	169.3726	9.935372	169.05	169.19025	10.52646	144.8	200.4	55.6	0.1618
Weight	3	500	82.7094	20.884652	80.00	81.05650	19.94097	41.9	184.5	142.6	0.9835
BMI	4	500	28.7574	6.567995	27.70	28.11625	6.07866	17.0	63.3	46.3	1.1522
SBP	5	500	119.3320	15.049344	119.00	118.41250	13.34340	84.0	221.0	137.0	1.1872
DBP	6	500	72.1580	11.245709	72.00	72.24500	8.89560	0.0	110.0	110.0	-0.5339
Pulse	7	500	74.1640	11.500505	74.00	73.76500	11.86080	48.0	114.0	66.0	0.3414

```
nh_500cc |>
  select(Age, Height, Weight, BMI, SBP, DBP, Pulse) |>
  tbl_summary( statistic = list(all_continuous() ~ "{mean} ({sd}):
                      [ {min}, {p25}, {median}, {p75}, {max} ]"))
```

Table printed with `knitr::kable()`, not {gt}. Learn why at
https://www.danieldsjoberg.com/gtsummary/articles/rmarkdown.html
To suppress this message, include `message = FALSE` in code chunk header.

Characteristic	**N = 500**
Age	42 (13): [21, 30, 42, 52, 64]
Height	169 (10): [145, 162, 169, 176, 200]
Weight	83 (21): [42, 68, 80, 94, 185]
BMI	29 (7): [17, 24, 28, 32, 63]
SBP	119 (15): [84, 110, 119, 127, 221]
DBP	72 (11): [0, 66, 72, 79, 110]
Pulse	74 (12): [48, 66, 74, 82, 114]

5.5.1 A look at BMI (Body-Mass Index)

The definition of BMI (*body-mass index*) for adult subjects (which is expressed in units of kg/m^2) is:

$$\text{Body Mass Index} = \frac{\text{weight in kg}}{(\text{height in meters})^2} = 703 \times \frac{\text{weight in pounds}}{(\text{height in inches})^2}$$

[BMI is essentially] ... a measure of a person's *thinness* or *thickness*... BMI was designed for use as a simple means of classifying average sedentary (physically inactive) populations, with an average body composition. For these individuals, the current value recommendations are as follow: a BMI from 18.5 up to 25 may indicate optimal weight, a BMI lower than 18.5 suggests the person is underweight,

a number from 25 up to 30 may indicate the person is overweight, and a number from 30 upwards suggests the person is obese.

Wikipedia, https://en.wikipedia.org/wiki/Body_mass_index

5.5.2 Types of Quantitative Variables

Depending on the context, we would likely treat most of these quantitative variables as *discrete* given that are measurements are fairly crude (this is certainly true for `Age`, measured in years) although BMI is probably *continuous* in most settings, even though it is a function of two other measures (`Height` and `Weight`) which are rounded off to integer numbers of centimeters and kilograms, respectively.

It is also possible to separate out quantitative variables into **ratio** variables or **interval** variables.

- An interval variable has equal distances between values, but the zero point is arbitrary.
- A ratio variable has equal intervals between values, and a meaningful zero point.

For example, weight is an example of a ratio variable, while IQ is an example of an interval variable. We all know what zero weight is. An intelligence score like IQ is a different matter. We say that the average IQ is 100, but that's only by convention. We could just as easily have decided to add 400 to every IQ value and make the average 500 instead. Because IQ's intervals are equal, the difference between and IQ of 70 and an IQ of 80 is the same as the difference between 120 and 130. However, an IQ of 100 is not twice as high as an IQ of 50. The point is that if the zero point is artificial and movable, then the differences between numbers are meaningful but the ratios between them are not.

On the other hand, most lab test values are ratio variables, as are physical characteristics like height and weight. Each of the quantitative variables in our `nh_500cc` data can be thought of as a ratio variable.A person who weighs 100 kg is twice as heavy as one who weighs 50 kg; even when we convert kg to pounds, this is still true. For the most part, we can treat and analyze interval or ratio variables the same way.

5.6 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.

In the `nh_500cc` data, we have eight categorical variables, four binary and four with multiple categories.

```
nh_500cc |>
  select(Sex, PhysActive, Smoke100, SleepTrouble,
         Race, Education, MaritalStatus, HealthGen) |>
  summary()
```

```
     Sex        PhysActive Smoke100  SleepTrouble        Race
 female:236    No :216    No :291   No :380       Asian   : 51
 male  :264    Yes:284    Yes:209   Yes:120       Black   : 81
                                                  Hispanic: 37
                                                  Mexican : 48
                                                  White   :262
                                                  Other   : 21

         Education          MaritalStatus      HealthGen
 8th Grade      : 26    Divorced   : 47   Excellent: 52
 9 - 11th Grade: 59     LivePartner : 46  Vgood    :167
 High School   : 89     Married    :256   Good     :204
 Some College  :153     NeverMarried:125  Fair     : 65
 College Grad  :173     Separated  : 17   Poor     : 12
                        Widowed    :  9
```

5.6.1 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 `nh_500cc` data, `Race` is a nominal variable with multiple unordered categories. So is `MaritalStatus`.

- 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.

Examples of ordinal multi-categorical variables in the `nh_500cc` data include the Education and HealthGen variables.

- 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 may be treated as either ordinal, or nominal.

Binary variables in the `nh_500cc` data include `Sex`, `PhysActive`, `Smoke100`, `SleepTrouble`. Each can be thought of 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.

5.7 Tabulating Binary Variables

Note how the `tbl_summary()` approach works with binary variables, and in particular with variables coded Yes and No, like `PhysActive`, `Smoke100` and `SleepTrouble`.

```
nh_500cc |>
    select(Sex, PhysActive, Smoke100, SleepTrouble) |>
    tbl_summary()
```

Table printed with `knitr::kable()`, not {gt}. Learn why at
https://www.danieldsjoberg.com/gtsummary/articles/rmarkdown.html
To suppress this message, include `message = FALSE` in code chunk header.

Characteristic	**N = 500**
Sex	
female	236 (47%)
male	264 (53%)
PhysActive	284 (57%)
Smoke100	209 (42%)
SleepTrouble	120 (24%)

We can also summarize any particular variable with the `tabyl()` function from the `janitor` package.

```
nh_500cc |>
  tabyl(Sex)
```

```
   Sex   n percent
female 236   0.472
  male 264   0.528
```

Or, we can make a basic cross-tabulation of two binary variables, like this:

```
nh_500cc |>
  tabyl(PhysActive, Smoke100) |>
  adorn_title()
```

```
          Smoke100
PhysActive      No Yes
        No     111 105
       Yes     180 104
```

5.8 Tabulating Multi-Categorical Variables

```
nh_500cc |>
  select(Race, Education, MaritalStatus, HealthGen) |>
  tbl_summary()
```

```
Table printed with `knitr::kable()`, not {gt}. Learn why at
https://www.danieldsjoberg.com/gtsummary/articles/rmarkdown.html
To suppress this message, include `message = FALSE` in code chunk header.
```

Characteristic	**N = 500**
Race	
Asian	51 (10%)
Black	81 (16%)
Hispanic	37 (7.4%)
Mexican	48 (9.6%)
White	262 (52%)
Other	21 (4.2%)
Education	
8th Grade	26 (5.2%)
9 - 11th Grade	59 (12%)
High School	89 (18%)
Some College	153 (31%)
College Grad	173 (35%)
MaritalStatus	
Divorced	47 (9.4%)
LivePartner	46 (9.2%)
Married	256 (51%)
NeverMarried	125 (25%)
Separated	17 (3.4%)
Widowed	9 (1.8%)
HealthGen	
Excellent	52 (10%)
Vgood	167 (33%)
Good	204 (41%)
Fair	65 (13%)
Poor	12 (2.4%)

We can also use `tabyl()` to look at combinations of multi-categorical variables, whether they are ordinal or nominal.

```
nh_500cc |>
  tabyl(Education, HealthGen) |>
  adorn_totals(where = c("row", "col")) |>
  adorn_title() |>
  kbl(align = 'lrrrrrc') |>
  kable_styling(full_width = FALSE)
```

Education	HealthGen Excellent	Vgood	Good	Fair	Poor	Total
8th Grade	2	3	9	9	3	26
9 - 11th Grade	4	12	28	12	3	59
High School	9	26	36	18	0	89
Some College	11	46	75	17	4	153
College Grad	26	80	56	9	2	173
Total	52	167	204	65	12	500

5.9 Coming Up

Next, we'll look at several additional approaches to building tabular and graphical summaries for these data, beyond the ideas provided here.

6 More NHANES Summaries

6.1 Setup: Packages Used Here

```
knitr::opts_chunk$set(comment = NA)

library(janitor)
library(kableExtra)
library(patchwork)
library(tidyverse)

theme_set(theme_bw())
```

6.2 Re-Loading the "Complete Cases" Sample

In this chapter, we'll build on the summaries we've illustrated previously. Let's begin by again loading into the nh_500cc tibble the information from the nh_adult500cc.Rds file we created in Section 4.5.

```
nh_500cc <- read_rds("data/nh_adult500cc.Rds")
```

6.3 Distribution of Heights

What is the distribution of height in this new sample?

```
ggplot(data = nh_500cc, aes(x = Height)) +
    geom_histogram()
```

`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
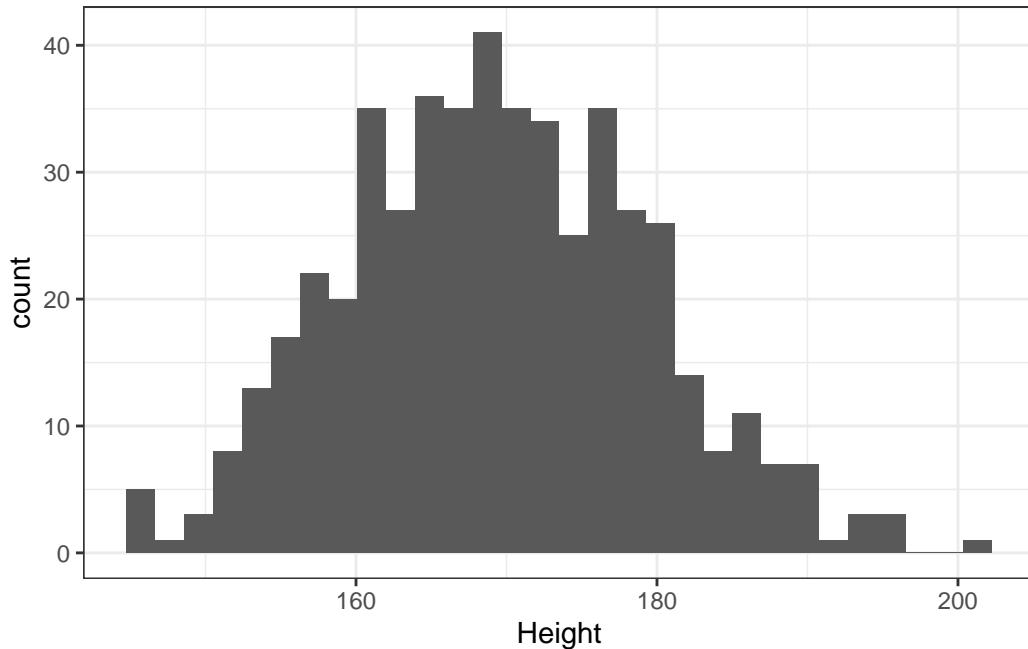
We can do several things to clean this up.

1. We'll change the color of the lines for each bar of the histogram.
2. We'll change the fill inside each bar to make them stand out a bit more.
3. We'll add a title and relabel the horizontal (x) axis to include the units of measurement.
4. We'll avoid the warning by selecting a number of bins (we'll use 25 here) into which we'll group the heights before drawing the histogram.

```
ggplot(data = nh_500cc, aes(x = Height)) +
    geom_histogram(bins = 25, col = "yellow", fill = "blue") +
    labs(title = "Height of 500 NHANES Adults",
         x = "Height in cm.")
```

Height of 500 NHANES Adults
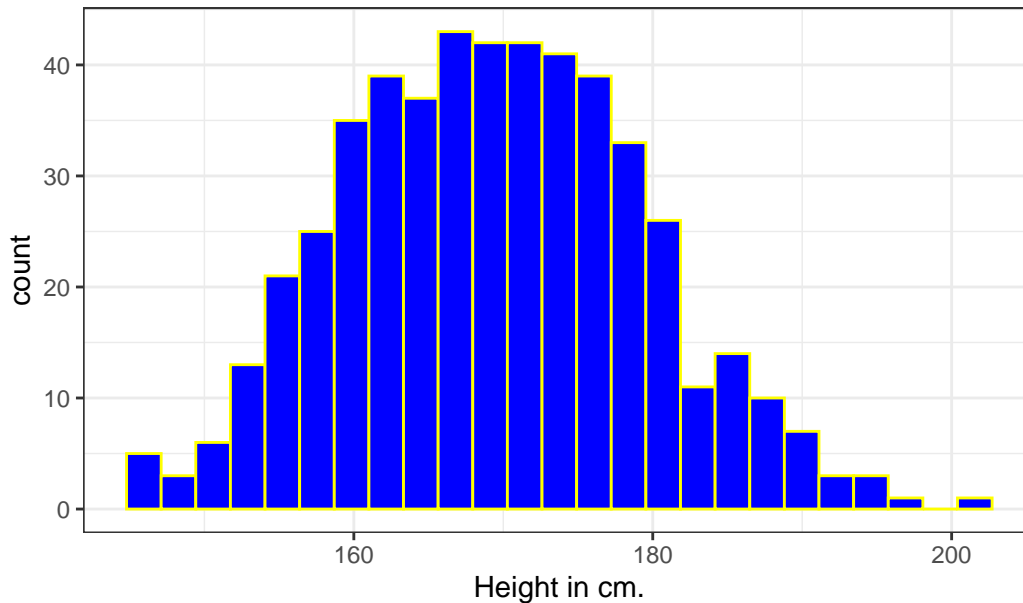
6.3.1 Changing a Histogram's Fill and Color

The CWRU color guide (https://case.edu/umc/our-brand/visual-guidelines/) lists the HTML color schemes for CWRU blue and CWRU gray. Let's match that color scheme. We will also change the bins for the histogram, to gather observations into groups of 2 cm. each, by specifying the width of the bins, rather than the number of bins.

```
cwru.blue <- '#0a304e'
cwru.gray <- '#626262'

ggplot(data = nh_500cc, aes(x = Height)) +
    geom_histogram(binwidth = 2,
                   col = cwru.gray, fill = cwru.blue) +
    labs(title = "Height of 500 NHANES Adults",
         x = "Height in cm.")
```

Height of 500 NHANES Adults
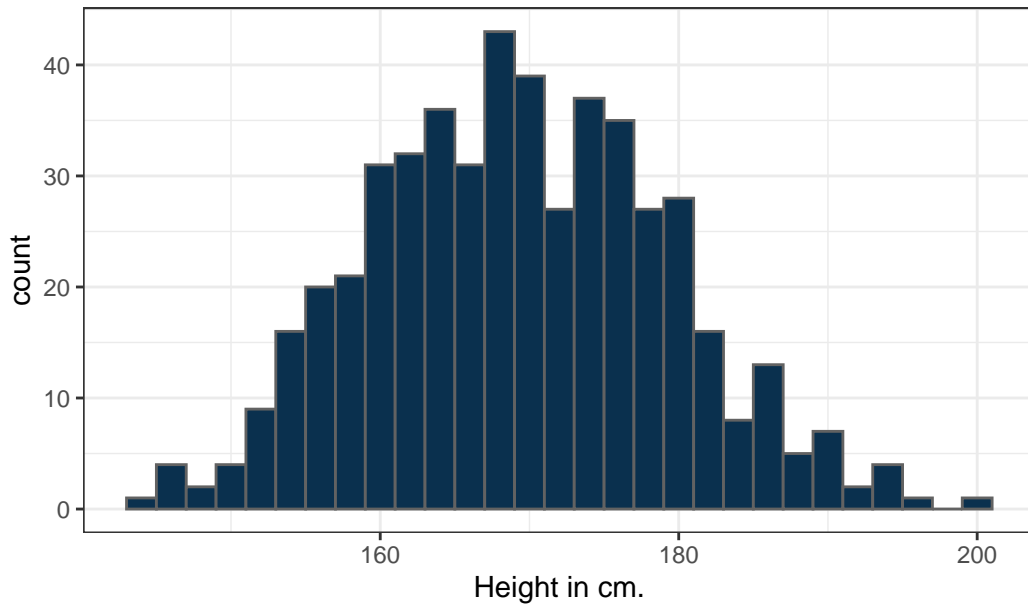
6.3.2 Using a frequency polygon

A frequency polygon essentially smooths out the top of the histogram, and can also be used to show the distribution of `Height`.

```
ggplot(data = nh_500cc, aes(x = Height)) +
    geom_freqpoly(bins = 20) +
    labs(title = "Height of 500 NHANES Adults",
         x = "Height in cm.")
```

Height of 500 NHANES Adults
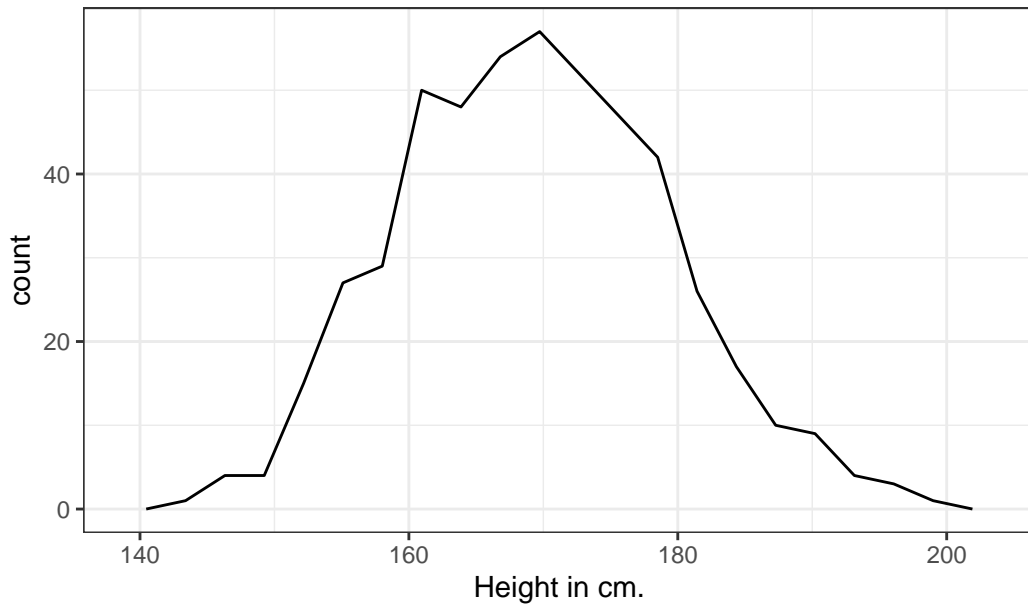
6.3.3 Using a dotplot

A dotplot can also be used to show the distribution of a variable like `Height`, and produces a somewhat more granular histogram, depending on the settings for `binwidth` and `dotsize`.

```
ggplot(data = nh_500cc, aes(x = Height)) +
    geom_dotplot(dotsize = 0.75, binwidth = 1) +
    scale_y_continuous(NULL, breaks = NULL) + # hide y axis
    labs(title = "Height of 500 NHANES Adults",
        x = "Height in cm.")
```

Height of 500 NHANES Adults
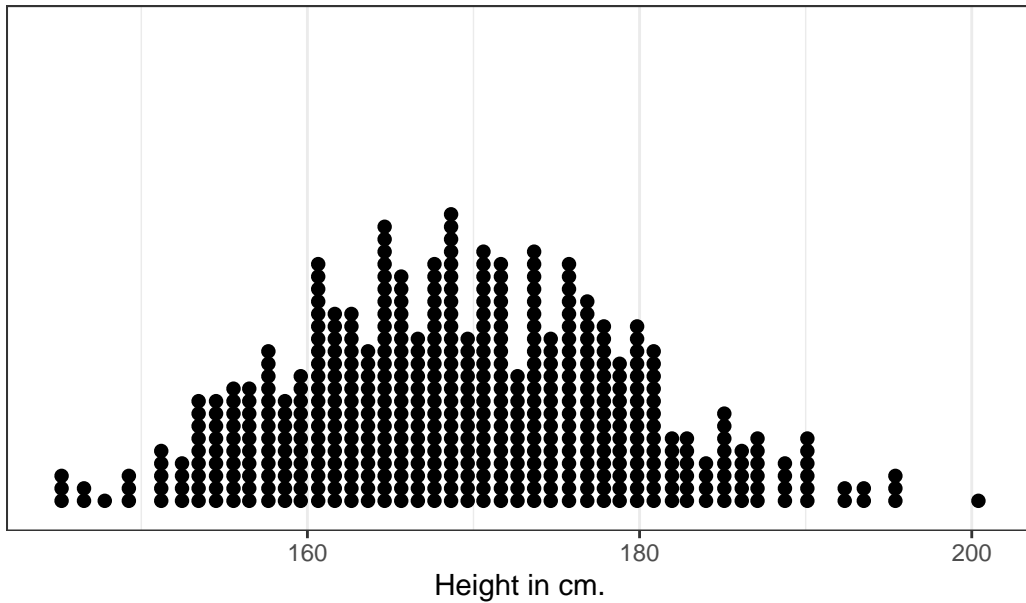
Height in cm.

6.4 Height and Sex

Let's look again at the impact of a respondent's sex on their height, but now within our sample of adults.

```
ggplot(data = nh_500cc,
       aes(x = Sex, y = Height, color = Sex)) +
  geom_point() +
  labs(title = "Height by Sex for 500 NHANES Adults",
       y = "Height in cm.")
```

Height by Sex for 500 NHANES Adults
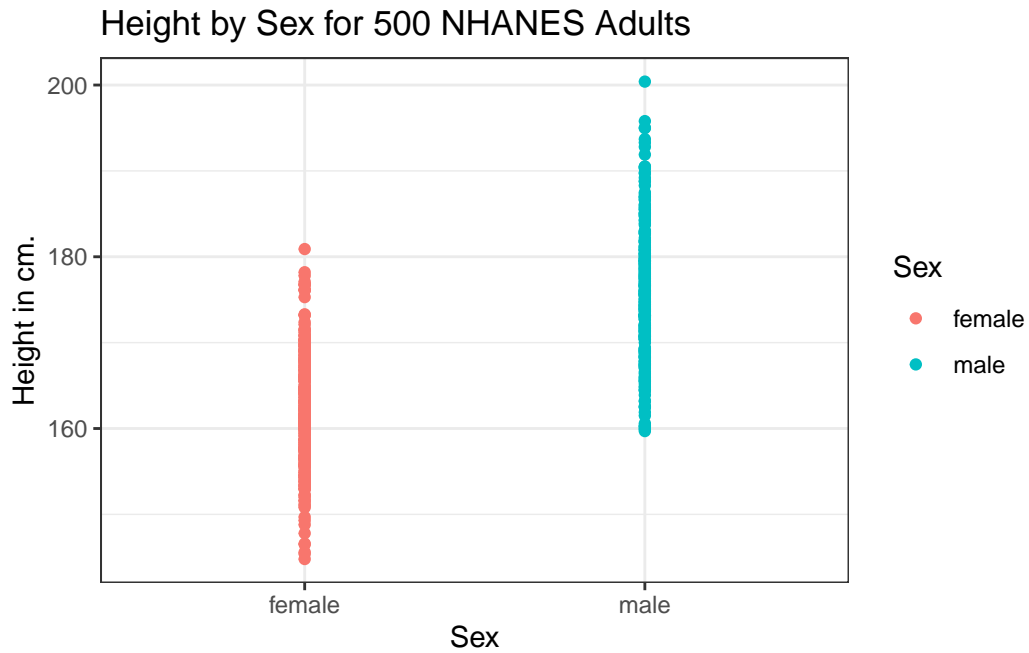
This plot isn't so useful. We can improve things a little by jittering the points horizontally, so that the overlap is reduced.

```
ggplot(data = nh_500cc, aes(x = Sex, y = Height, color = Sex)) +
    geom_jitter(width = 0.2) +
    labs(title = "Height by Sex (jittered) for 500 NHANES Adults",
        y = "Height in cm.")
```

Height by Sex (jittered) for 500 NHANES Adults
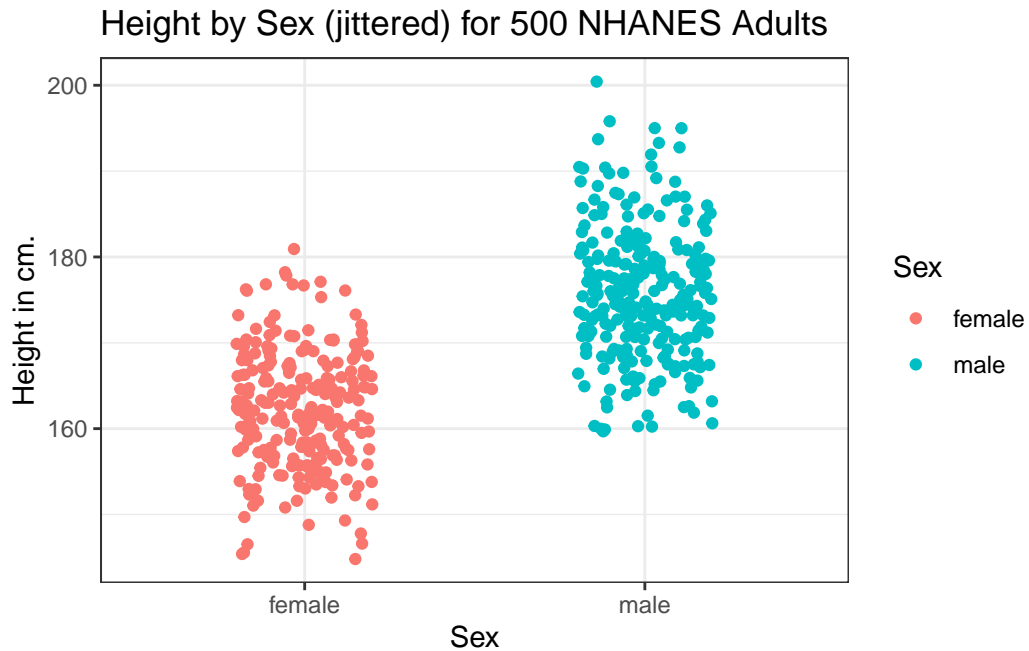
Perhaps it might be better to summarize the distribution in a different way. We might consider a boxplot of the data.

6.4.1 A Boxplot of Height by Sex

```
ggplot(data = nh_500cc, aes(x = Sex, y = Height, fill = Sex)) +
    geom_boxplot() +
    labs(title = "Boxplot of Height by Sex for 500 NHANES Adults",
         y = "Height in cm.")
```

Boxplot of Height by Sex for 500 NHANES Adults
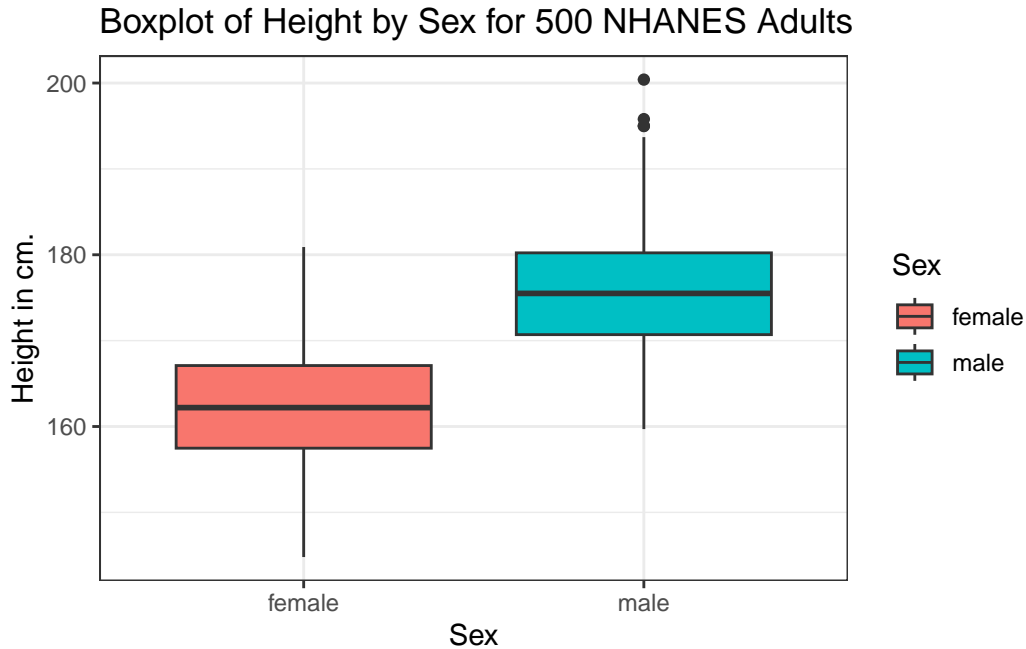
The boxplot shows some summary statistics based on percentiles. The boxes in the middle of the line show the data values that include the middle half of the data once its been sorted. The 25th percentile (value that exceeds 1/4 of the data) is indicated by the bottom of the box, while the top of the box is located at the 75th percentile. The solid line inside the box indicates the median (also called the 50th percentile) of the Heights for that Sex.

6.4.2 Adding a violin plot

A boxplot is often supplemented with a *violin plot* to better show the shape of the distribution.

```
ggplot(data = nh_500cc, aes(x = Sex, y = Height, fill = Sex)) +
    geom_violin() +
    geom_boxplot(width = 0.3) +
    labs(title = "Boxplot of Height by Sex for 500 NHANES Adults",
         y = "Height in cm.")
```

Boxplot of Height by Sex for 500 NHANES Adults
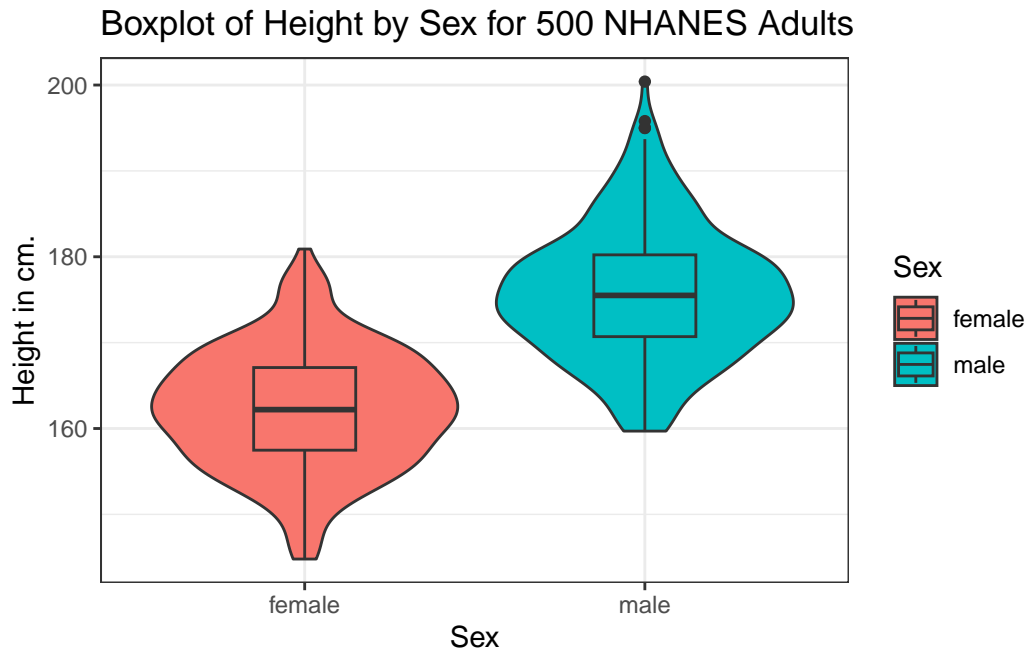
This usually works better if the boxes are given a different fill than the violins, as shown in the following figure.

```
ggplot(data = nh_500cc, aes(x = Sex, y = Height)) +
    geom_violin(aes(fill = Sex)) +
    geom_boxplot(width = 0.3) +
    labs(title = "Boxplot of Height by Sex for 500 NHANES Adults",
         y = "Height in cm.")
```

Boxplot of Height by Sex for 500 NHANES Adults
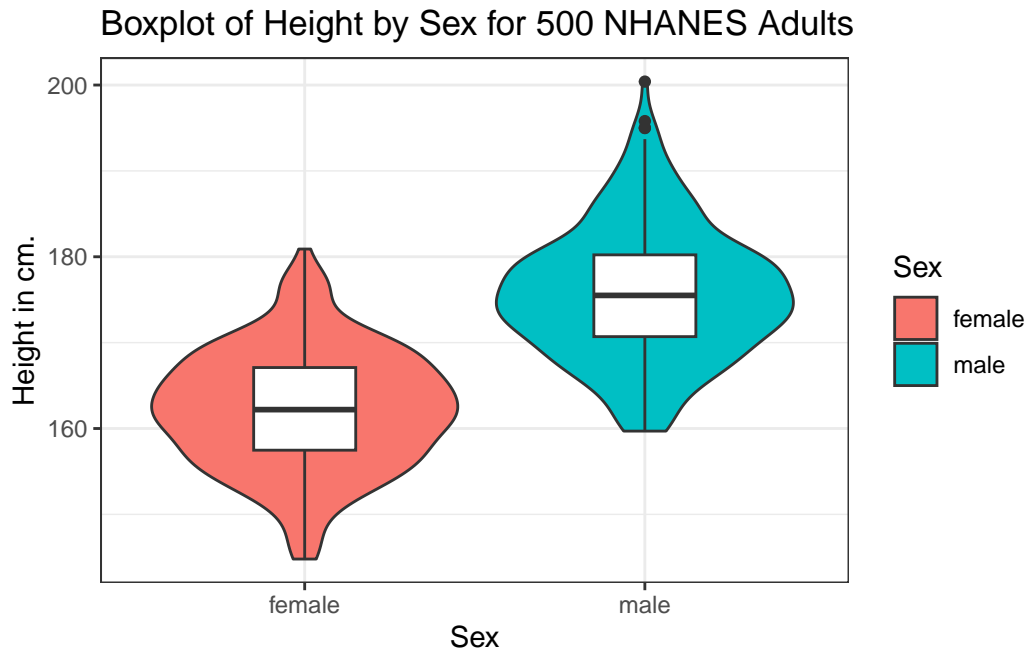
We can also flip the boxplots on their side, using `coord_flip()`.

```
ggplot(data = nh_500cc, aes(x = Sex, y = Height)) +
    geom_violin() +
    geom_boxplot(aes(fill = Sex), width = 0.3) +
    labs(title = "Boxplot of Height by Sex for 500 NHANES Adults",
         y = "Height in cm.") +
    coord_flip()
```

Boxplot of Height by Sex for 500 NHANES Adults
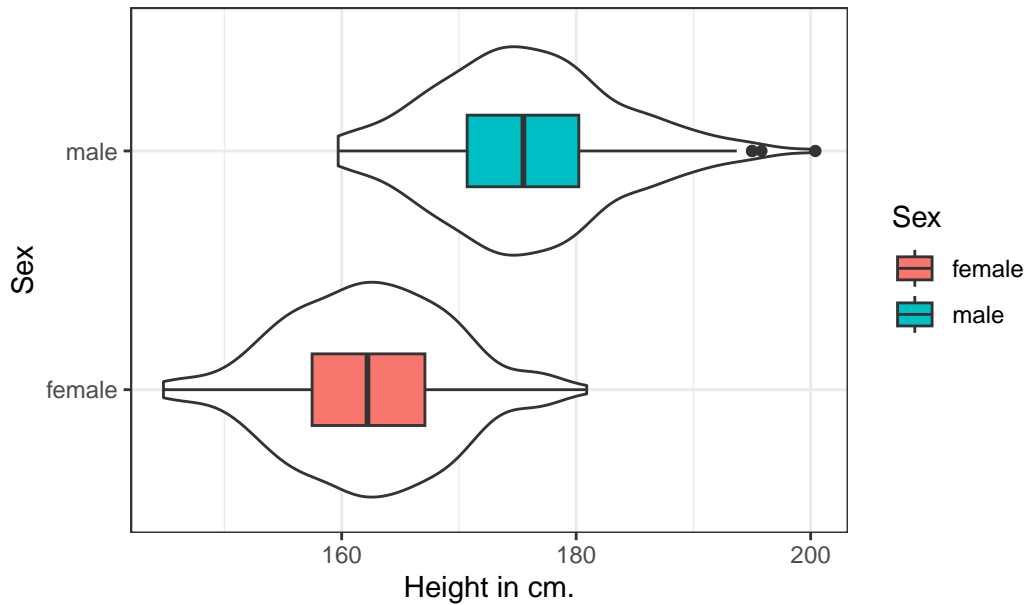
6.4.3 Histograms of Height by Sex

Or perhaps we'd like to see a pair of faceted histograms?

```
ggplot(data = nh_500cc, aes(x = Height, fill = Sex)) +
    geom_histogram(color = "white", bins = 20) +
    labs(title = "Histogram of Height by Sex for 500 NHANES Adults",
         x = "Height in cm.") +
    facet_wrap(~ Sex)
```

Histogram of Height by Sex for 500 NHANES Adults
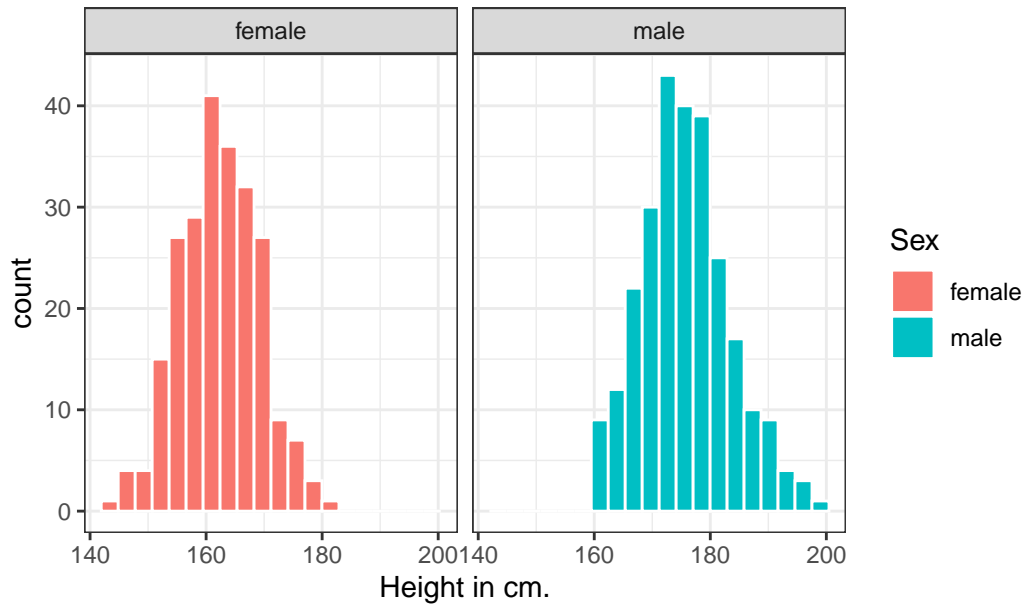
Can we redraw these histograms so that they are a little more comparable, and to get rid of the unnecessary legend?

```
ggplot(data = nh_500cc, aes(x = Height, fill = Sex)) +
    geom_histogram(color = "white", bins = 20) +
    labs(title = "Histogram of Height by Sex for 500 NHANES Adults",
         x = "Height in cm.") +
    guides(fill = "none") +
    facet_grid(Sex ~ .)
```

Histogram of Height by Sex for 500 NHANES Adults
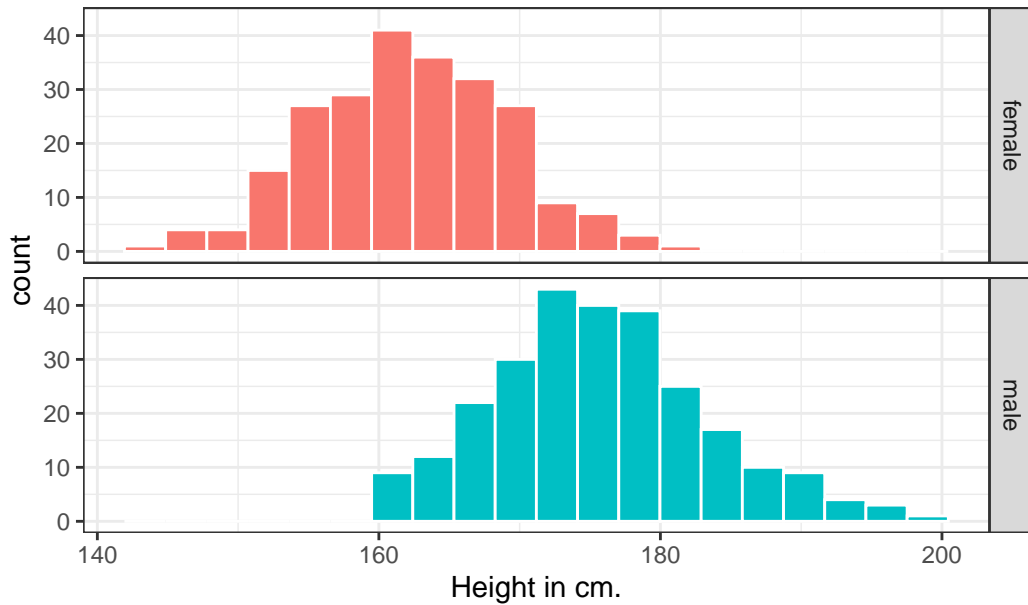
6.5 Visualizing Age and Height's Relationship, by Sex

We can start with a simple scatterplot of the Height (y-axis) and Age (x-axis) relationship across the subjects in our `nh_500cc` tibble.

```
ggplot(data = nh_500cc, aes(x = Age, y = Height)) +
  geom_point() +
  labs(title = "Height-Age Relationship in 500 NHANES adults")
```

Height–Age Relationship in 500 NHANES adults
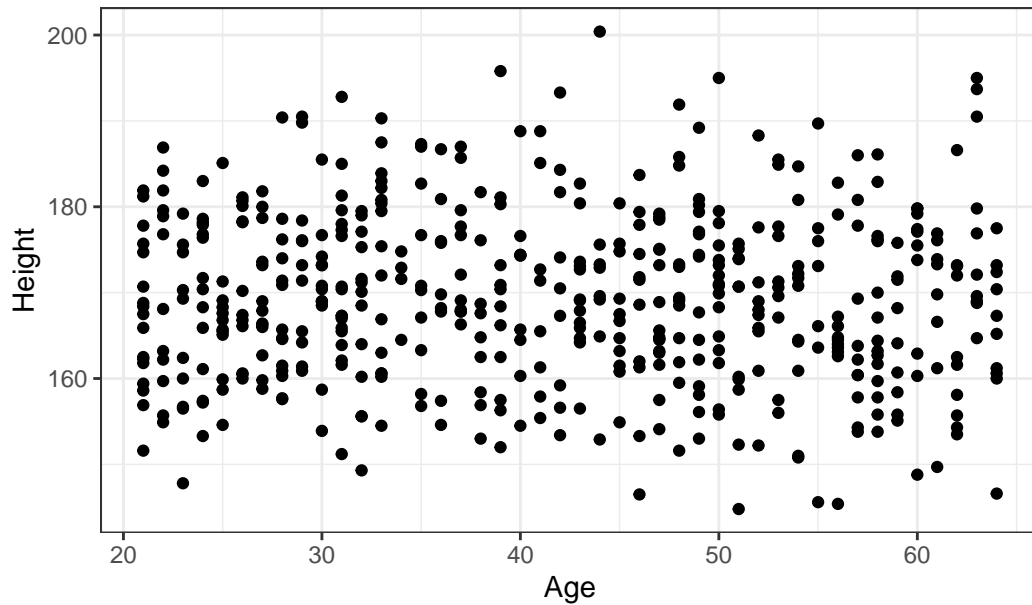
Is there a meaningful difference in what this relationship looks like, depending on Sex?

6.5.1 Adding Color to the plot

Let's add Sex to the plot using color, and also adjust the y axis label to incorporate the units of measurement.

```
ggplot(data = nh_500cc, aes(x = Age, y = Height, color = Sex)) +
    geom_point() +
    labs(title = "Height-Age Relationship in 500 NHANES adults, by Sex",
         y = "Height in cm.")
```

Height–Age Relationship in 500 NHANES adults, by Sex
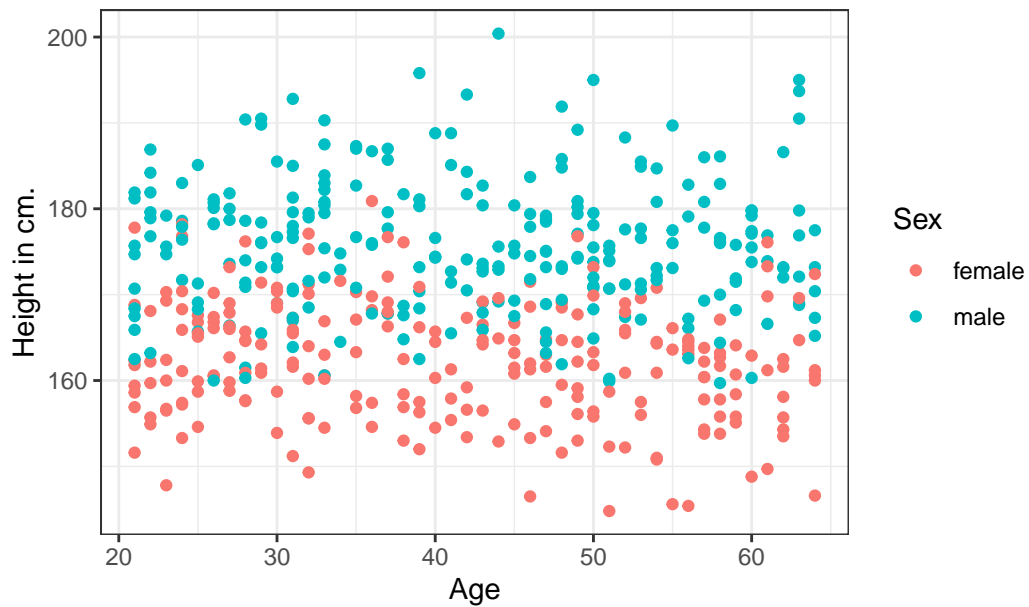
6.5.2 Can we show the Female and Male relationships in separate panels?

Sure. We can facet the scatterplot into a panel for each Sex, as follows.

```
ggplot(data = nh_500cc, aes(x = Age, y = Height, color = Sex)) +
    geom_point() +
    labs(title = "Height-Age Relationship in 500 NHANES adults",
         y = "Height in cm.") +
    facet_wrap(~ Sex)
```
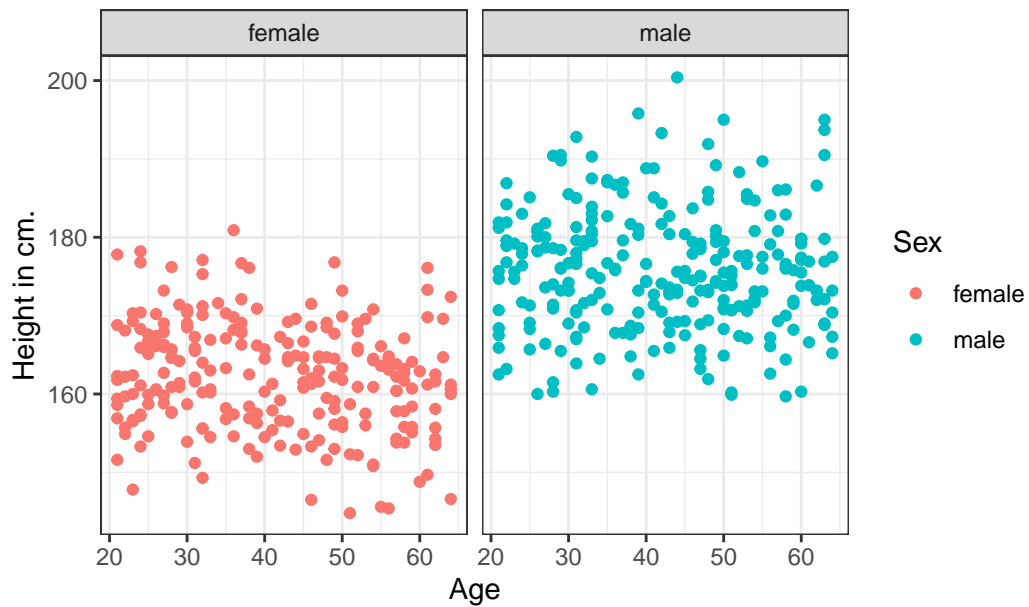
6.5.3 Can we add a smooth curve to show the relationship in each plot?

Yes, by adding a call to the `geom_smooth()` function. Is there any indication of a strong relationship between Age and Height in this sample?

```
ggplot(data = nh_500cc, aes(x = Age, y = Height, color = Sex)) +
    geom_point() +
    geom_smooth(method = "loess", formula = y ~ x) +
    labs(title = "Height-Age Relationship in NHANES sample",
         y = "Height in cm.") +
    facet_wrap(~ Sex)
```

Height−Age Relationship in NHANES sample
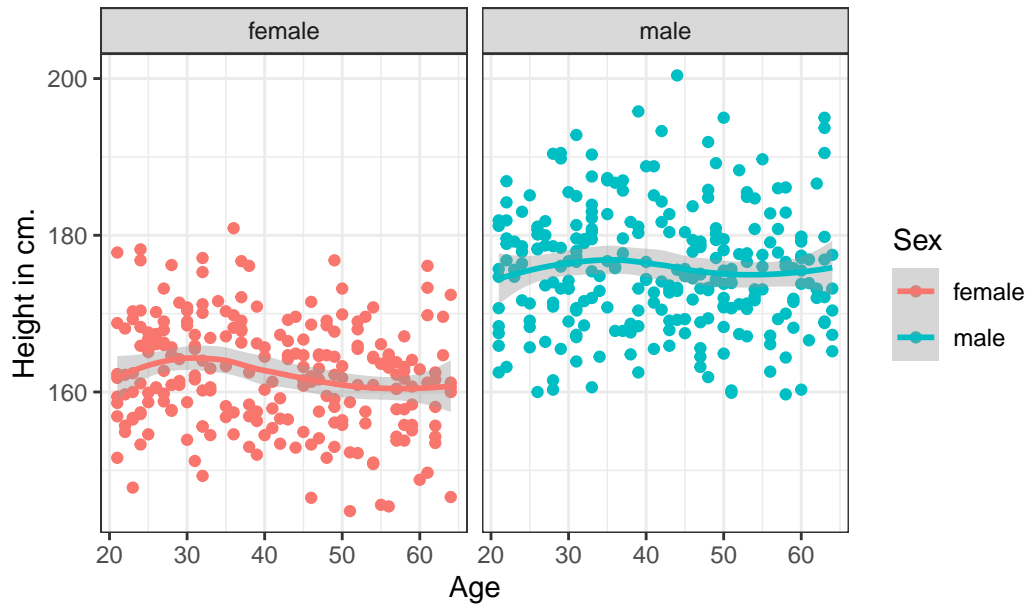
6.5.4 What if we want to assume straight line relationships?

We could look at a linear model in each part of the plot instead, and this time, we'll also get rid of the redundant legend, using the `guides()` command. Does assuming a straight line make much of a difference here?

```
ggplot(data = nh_500cc, aes(x = Age, y = Height, color = Sex)) +
    geom_point() +
    geom_smooth(method = "lm", formula = y ~ x) +
    guides(color = "none") +
    labs(title = "Height-Age Relationship in NHANES sample",
         y = "Height in cm.") +
    facet_wrap(~ Sex)
```

Height–Age Relationship in NHANES sample
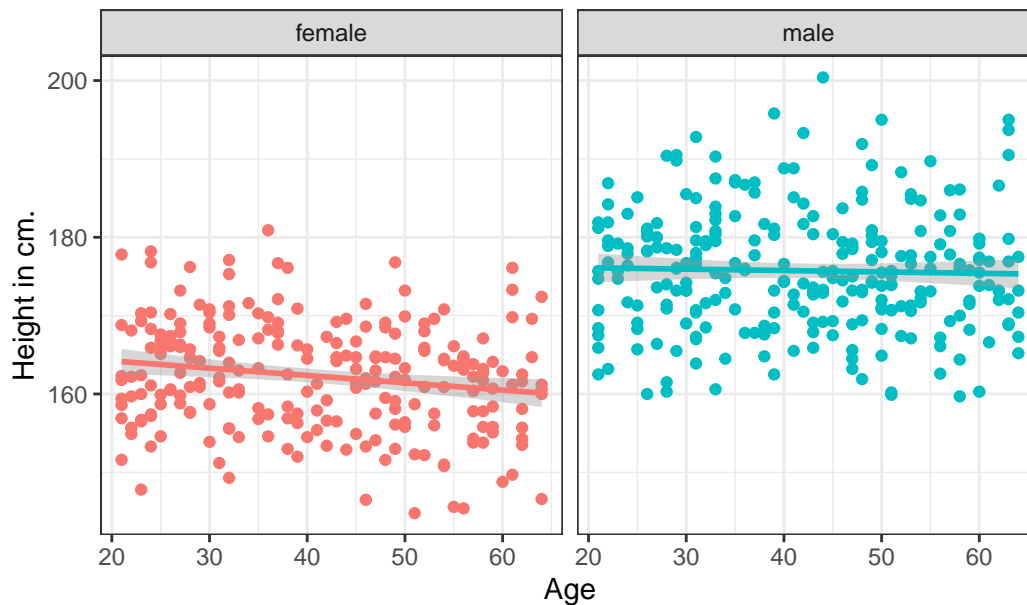
6.6 Combining Plots with `patchwork`

The `patchwork` package in R allows us to use some simple commands to put two plots together.

Suppose we create two separate plots, which we'll name `p1` and `p2`, as follows.

```
p1 <- ggplot(data = nh_500cc, aes(x = Age, y = Height)) +
    geom_point() +
    labs(title = "Height and Age")

p2 <- ggplot(data = nh_500cc, aes(x = Sex, y = Height)) +
    geom_boxplot() +
    labs(title = "Height, by Sex")
```

Now, suppose we want to put them together in a single figure. Thanks to `patchwork`, we can simply type in the following.

```
p1 / p2
```

Height and Age

Height, by Sex
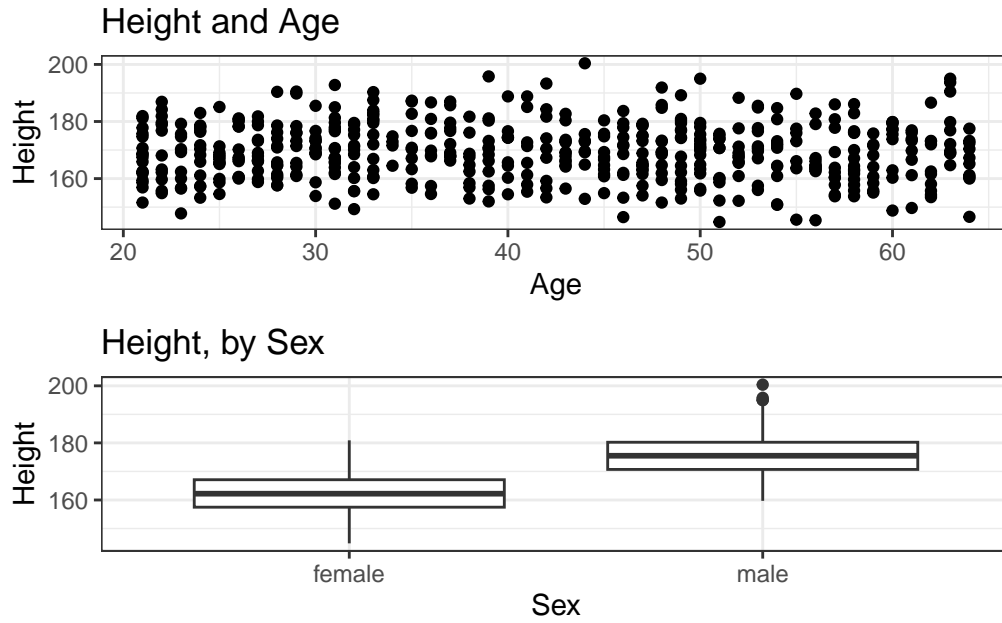
or we can place the images next to each other, and add an annotation, like this:

```
p1 + p2 +
    plot_annotation(title = "Our Combined Plots")
```

Our Combined Plots

Height and Age

Height, by Sex
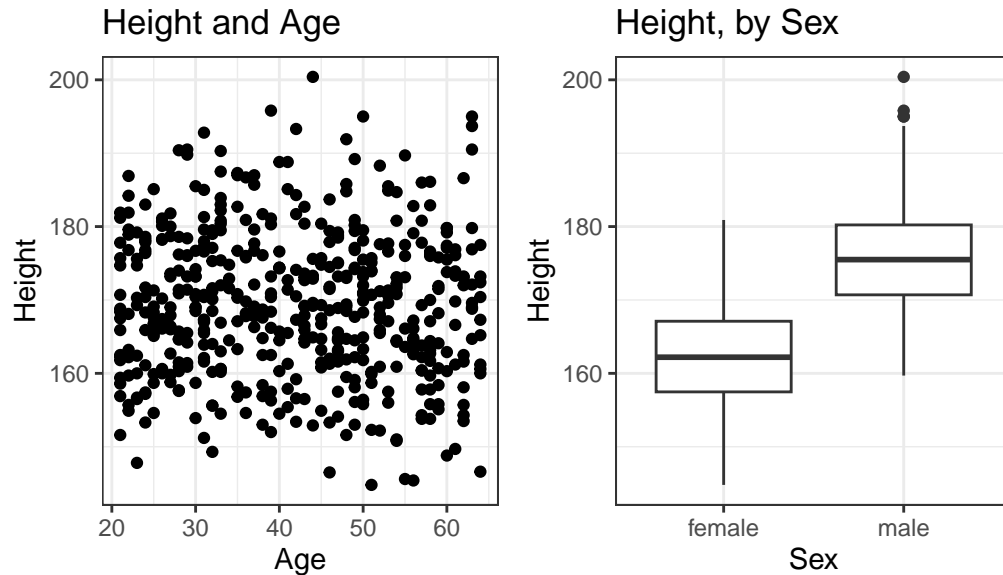
The `patchwork` package website provides lots of great examples and guides to make it very easy to combine separate ggplots into the same graphic. While there are other packages (`gridExtra` and `cowplot` are very nice, for instance) to do this task, I think `patchwork` is the most user-friendly, so that's the focus of these notes.

6.7 Looking at Pulse Rate

Let's look at a different outcome, the *pulse rate* for our subjects.

Here's a histogram, again with CWRU colors, for the pulse rates in our sample.

```
ggplot(data = nh_500cc, aes(x = Pulse)) +
    geom_histogram(binwidth = 1,
                   fill = cwru.blue, col = cwru.gray) +
    labs(title = "Histogram of Pulse Rate: NHANES Adults",
         x = "Pulse Rate (beats per minute)")
```

Histogram of Pulse Rate: NHANES Adults
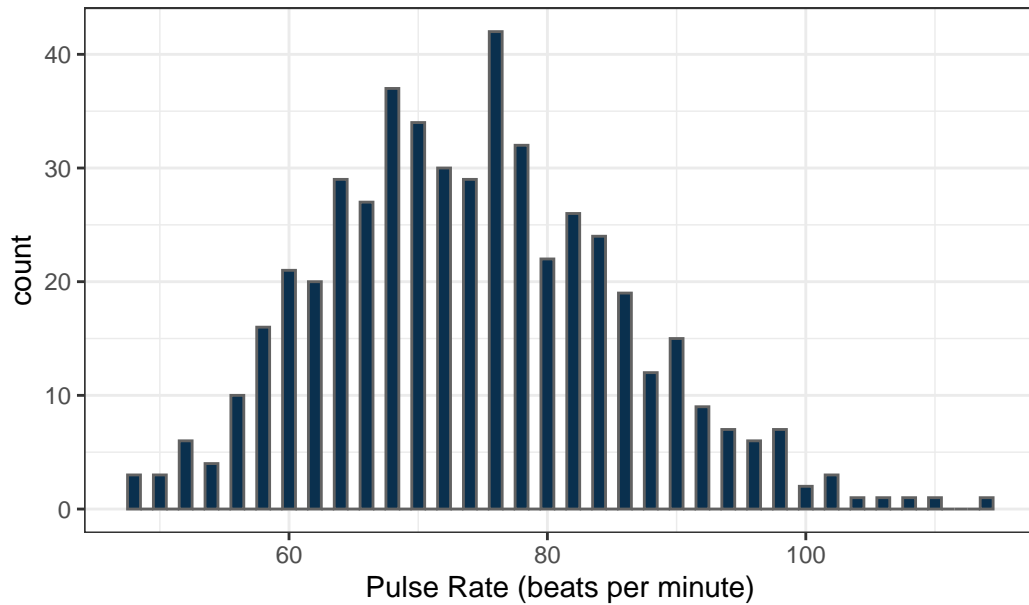
Suppose we instead bin up groups of 5 beats per minute together as we plot the Pulse rates.

```
ggplot(data = nh_500cc, aes(x = Pulse)) +
    geom_histogram(binwidth = 5,
                   fill = cwru.blue, col = cwru.gray) +
    labs(title = "Histogram of Pulse Rate: NHANES Adults",
         x = "Pulse Rate (beats per minute)")
```

Histogram of Pulse Rate: NHANES Adults
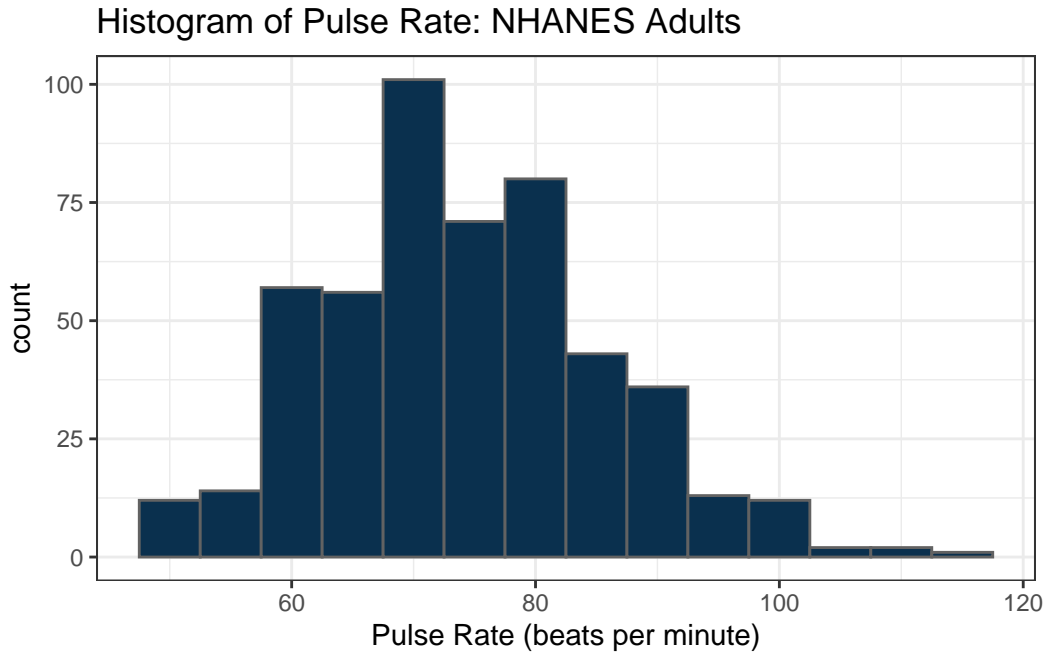
Which is the more useful representation will depend a lot on what questions you're trying to answer.

6.7.1 Pulse Rate and Physical Activity

We can also split up our data into groups based on whether the subjects are physically active. Let's try a boxplot.

```
ggplot(data = nh_500cc,
       aes(y = Pulse, x = PhysActive, fill = PhysActive)) +
    geom_boxplot() +
    guides(fill = "none") +
    labs(title = "Pulse Rate by Physical Activity Status in NHANES Adults")
```

PhysActive	count	mean(Pulse)	median(Pulse)
No	216	74.44	74
Yes	284	73.96	74

Pulse Rate by Physical Activity Status in NHANES Adults
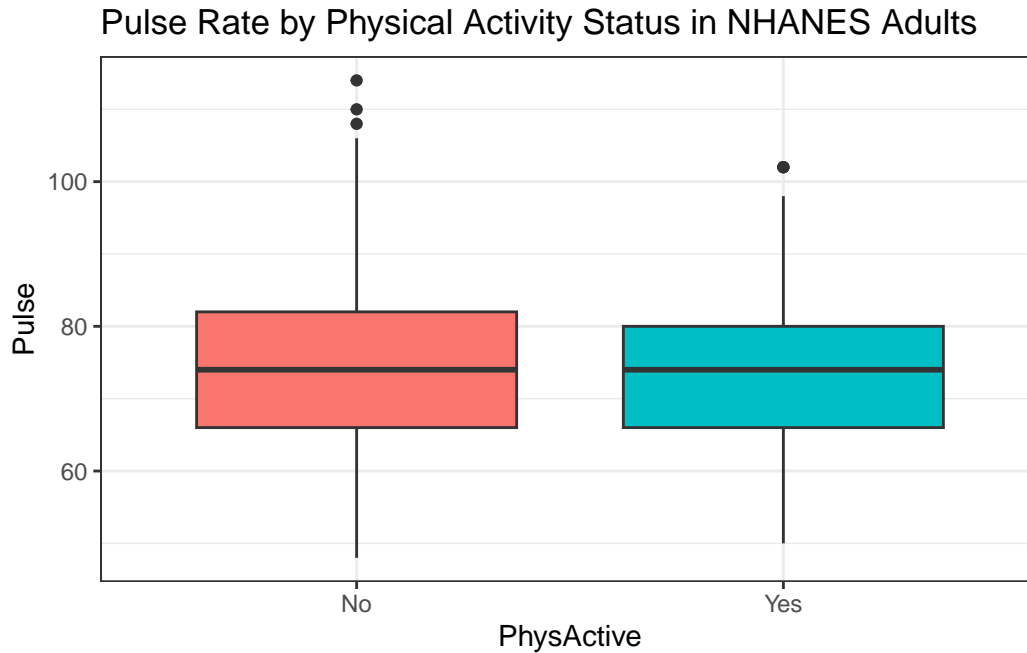
As an accompanying numerical summary, we might ask how many people fall into each of these `PhysActive` categories, and what is their "average" `Pulse` rate.

```
nh_500cc |>
    group_by(PhysActive) |>
    summarise(count = n(), mean(Pulse), median(Pulse)) |>
    kbl(digits = 2) |>
    kable_styling(full_width = FALSE)
```

The end of this chunk of code tells Quarto to generate a table with some attractive formatting, and rounding any decimals to two figures.

6.7.2 Pulse by Sleeping Trouble

```
ggplot(data = nh_500cc, aes(x = Pulse, fill = SleepTrouble)) +
    geom_histogram(color = "white", bins = 20) +
    labs(title = "Histogram of Pulse Rate by Sleep Trouble for NHANES Adults",
         x = "Pulse Rate (beats per minute)") +
    guides(fill = "none") +
    facet_grid(SleepTrouble ~ ., labeller = "label_both")
```
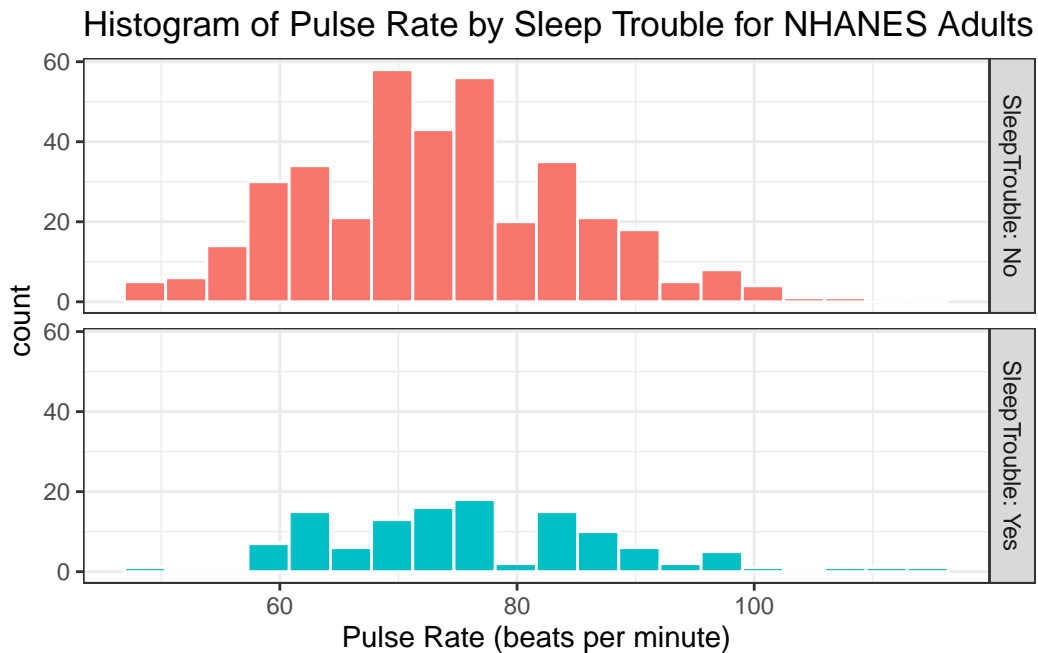

How many people fall into each of these `SleepTrouble` categories, and what is their "average" Pulse rate?

```
nh_500cc |>
    group_by(SleepTrouble) |>
    summarise(count = n(), mean(Pulse), median(Pulse)) |>
    kbl(digits = 2) |>
    kable_styling(full_width = F)
```

SleepTrouble	count	mean(Pulse)	median(Pulse)
No	380	73.45	73
Yes	120	76.43	76

6.7.3 Pulse and HealthGen

We can compare the distribution of Pulse rate across groups by the subject's self-reported overall health (`HealthGen`), as well.

```
ggplot(data = nh_500cc, aes(x = HealthGen, y = Pulse, fill = HealthGen)) +
    geom_boxplot() +
    labs(title = "Pulse by Self-Reported Overall Health for NHANES Adults",
         x = "Self-Reported Overall Health", y = "Pulse Rate") +
    guides(fill = "none")
```
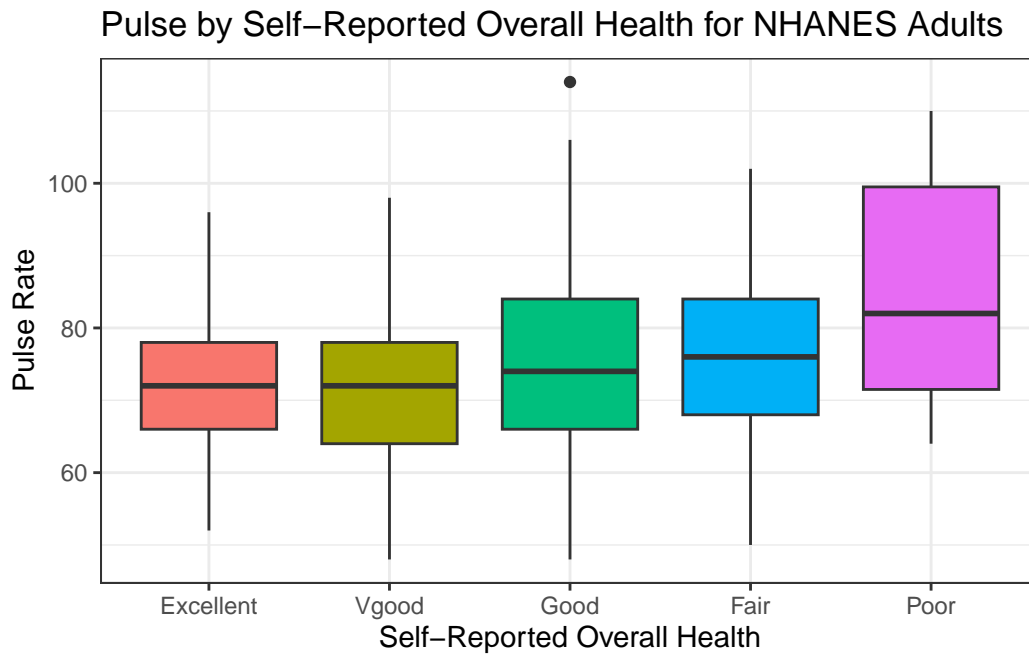

How many people fall into each of these `HealthGen` categories, and what is their "average" Pulse rate?

```
nh_500cc |>
    group_by(HealthGen) |>
    summarise(count = n(), mean(Pulse), median(Pulse)) |>
```

HealthGen	count	mean(Pulse)	median(Pulse)
Excellent	52	72.08	72
Vgood	167	71.78	72
Good	204	75.22	74
Fair	65	76.55	76
Poor	12	85.50	82

```
kbl(digits = 2) |>
kable_styling(full_width = F)
```

6.7.4 Pulse Rate and Systolic Blood Pressure

```
ggplot(data = nh_500cc, aes(x = SBP, y = Pulse)) +
    geom_point() +
    geom_smooth(method = "loess", formula = y ~ x) +
    labs(title = "Pulse Rate vs. SBP for NHANES Adults")
```
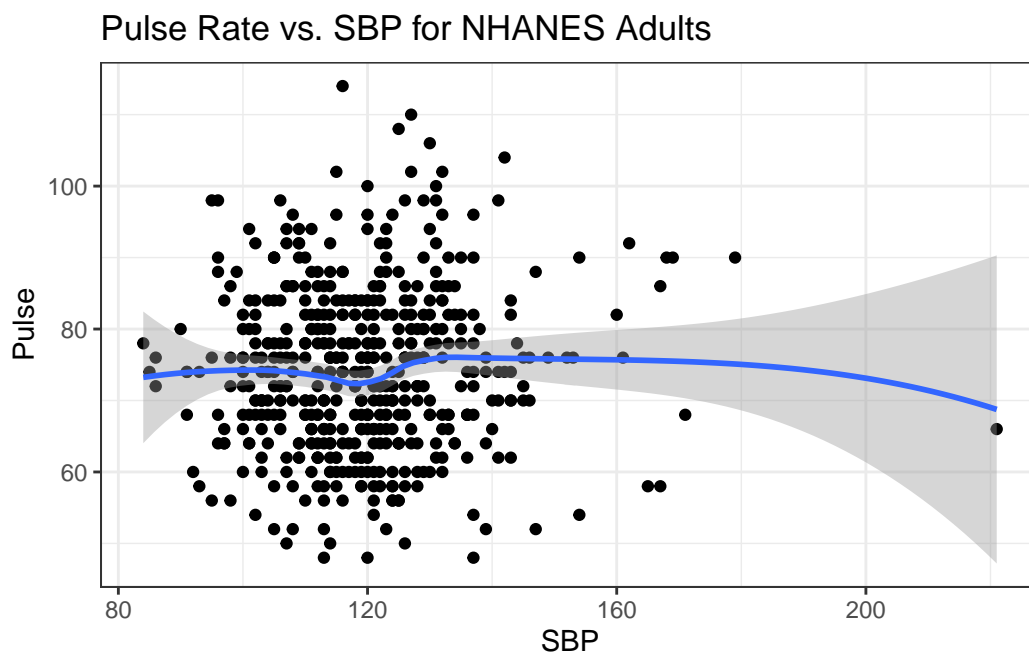
6.7.5 Sleep Trouble vs. No Sleep Trouble?

Could we see whether subjects who have described `SleepTrouble` show different SBP-pulse rate patterns than the subjects who haven't?

- Let's try doing this by changing the shape *and* the color of the points based on `SleepTrouble`.

```
ggplot(data = nh_500cc,
       aes(x = SBP, y = Pulse,
           color = SleepTrouble, shape = SleepTrouble)) +
    geom_point() +
    geom_smooth(method = "loess", formula = y ~ x) +
    labs(title = "Pulse Rate vs. SBP for NHANES Adults")
```
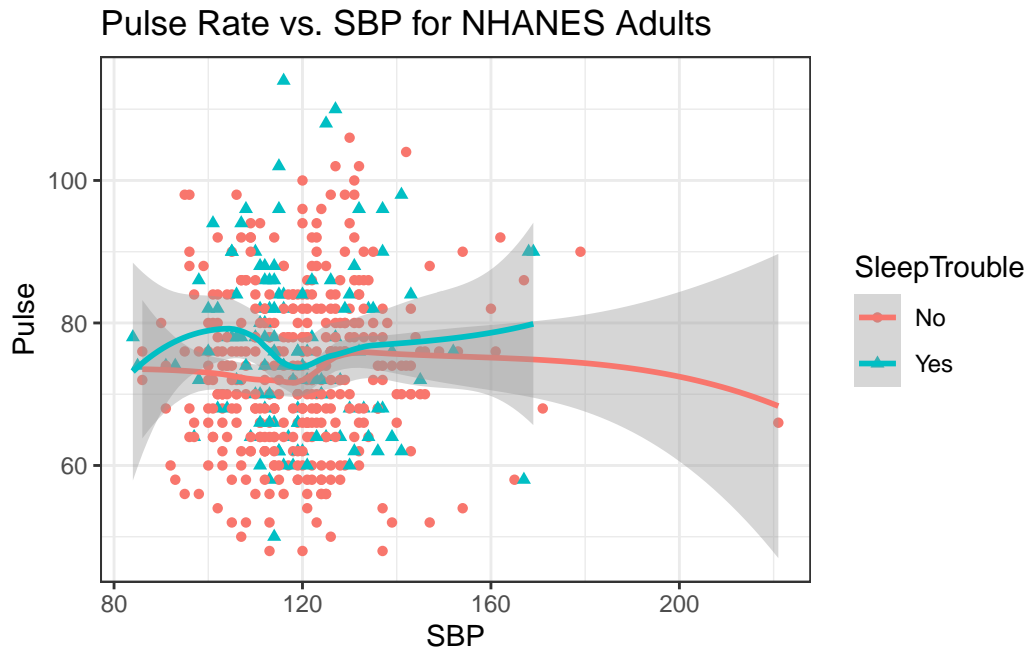

This plot might be easier to interpret if we faceted by `SleepTrouble`, as well.

```
ggplot(data = nh_500cc,
       aes(x = SBP, y = Pulse,
           color = SleepTrouble, shape = SleepTrouble)) +
    geom_point() +
    geom_smooth(method = "loess", formula = y ~ x) +
```

```
    labs(title = "Pulse Rate vs. SBP for NHANES Adults") +
    facet_wrap(~ SleepTrouble, labeller = "label_both")
```

Pulse Rate vs. SBP for NHANES Adults
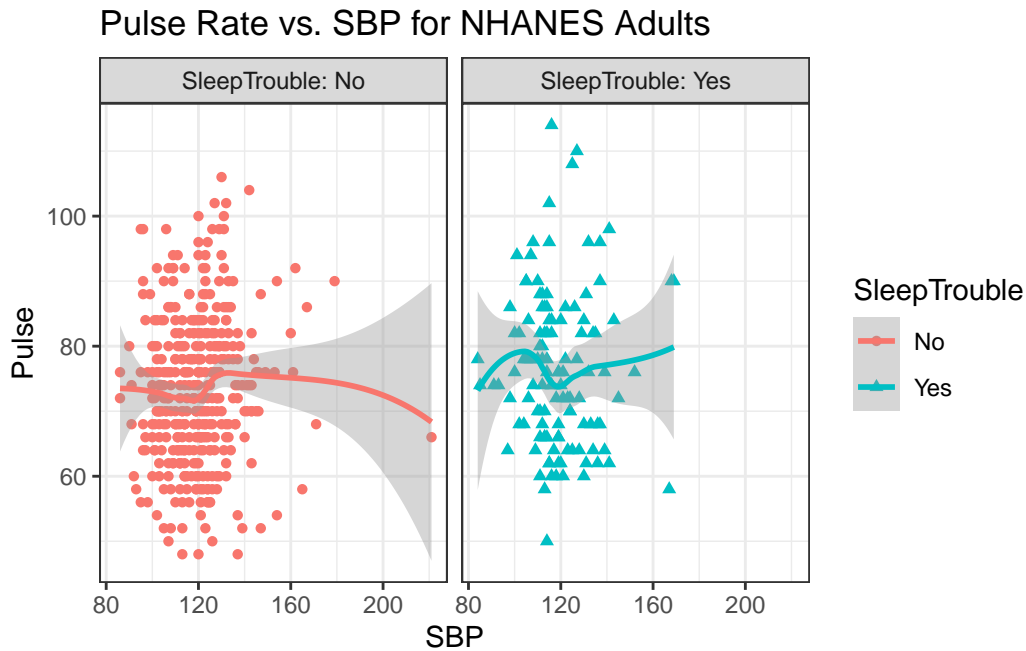
6.8 General Health Status

Here's a Table of the General Health Status results. Again, this is a self-reported rating of each subject's health on a five point scale (Excellent, Very Good, Good, Fair, Poor.)

```
  nh_500cc |>
      tabyl(HealthGen)
```

```
 HealthGen   n percent
 Excellent  52   0.104
     Vgood 167   0.334
      Good 204   0.408
      Fair  65   0.130
      Poor  12   0.024
```

The HealthGen data are categorical, which means that summarizing them with averages isn't

as appealing as looking at percentages, proportions and rates. The `tabyl` function comes from the `janitor` package in R.

- I don't actually like the title of `percent` here, as it's really a proportion, but that can be adjusted, and we can add a total.

```
nh_500cc |>
    tabyl(HealthGen) |>
    adorn_totals() |>
    adorn_pct_formatting()
```

```
HealthGen     n percent
Excellent    52   10.4%
    Vgood   167   33.4%
     Good   204   40.8%
     Fair    65   13.0%
     Poor    12    2.4%
    Total   500  100.0%
```

When working with an unordered categorical variable, like `MaritalStatus`, the same approach can work.

```
nh_500cc |>
    tabyl(MaritalStatus) |>
    adorn_totals() |>
    adorn_pct_formatting()
```

```
MaritalStatus     n percent
     Divorced    47    9.4%
  LivePartner    46    9.2%
      Married   256   51.2%
  NeverMarried  125   25.0%
    Separated    17    3.4%
      Widowed     9    1.8%
        Total   500  100.0%
```

6.8.1 Bar Chart for Categorical Data

Usually, a **bar chart** is the best choice for graphing a variable made up of categories.

```
ggplot(data = nh_500cc, aes(x = HealthGen)) +
    geom_bar()
```
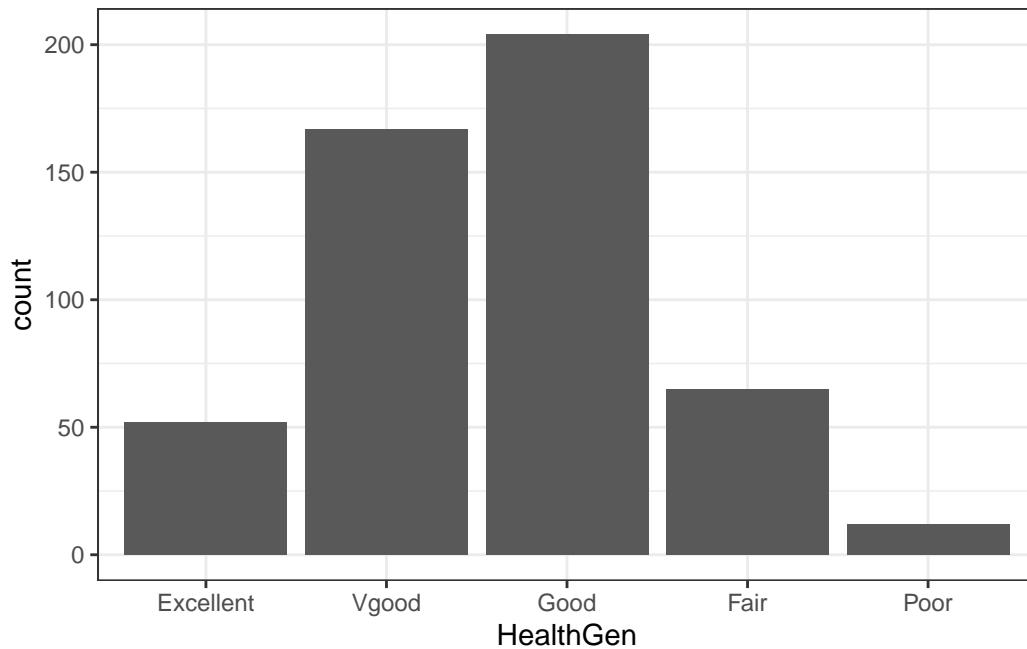

There are lots of things we can do to make this plot fancier.

```
ggplot(data = nh_500cc, aes(x = HealthGen, fill = HealthGen)) +
    geom_bar() +
    guides(fill = "none") +
    labs(x = "Self-Reported Health Status",
        y = "Number of NHANES subjects",
        title = "Self-Reported Health Status in NHANES Adults")
```

Self−Reported Health Status in NHANES Adults
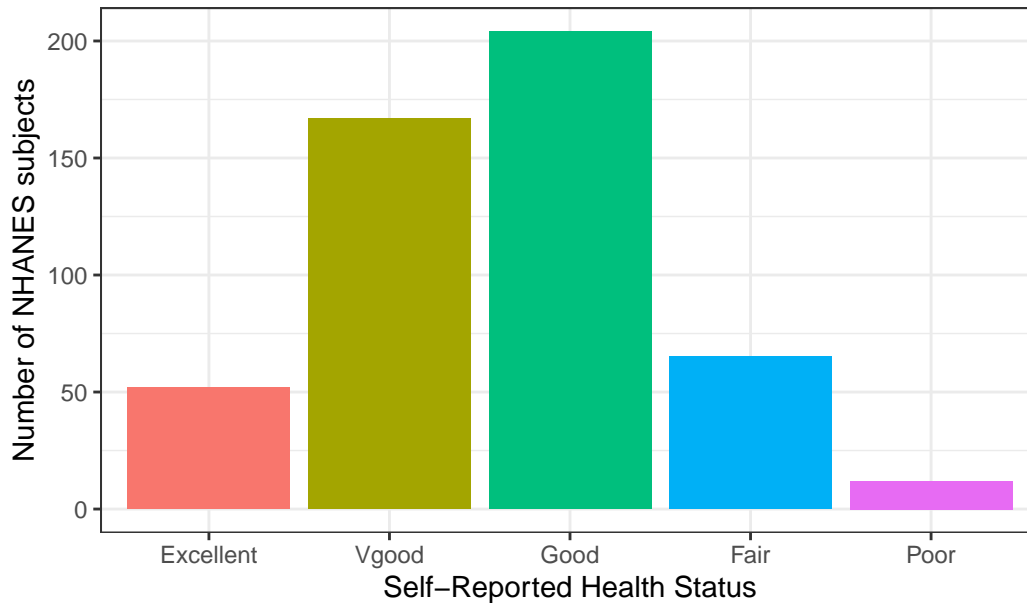
Or, we can really go crazy...

```
nh_500cc |>
    count(HealthGen) |>
    mutate(pct = round_half_up(prop.table(n) * 100, 1)) |>
    ggplot(aes(x = HealthGen, y = pct, fill = HealthGen)) +
    geom_bar(stat = "identity", position = "dodge") +
    scale_fill_viridis_d() +
    guides(fill = "none") +
    geom_text(aes(y = pct + 1,     # nudge above top of bar
                  label = paste0(pct, '%')),  # prettify
              position = position_dodge(width = .9),
              size = 4) +
    labs(x = "Self-Reported Health Status",
         y = "Percentage of Subjects",
         title = "Self-Reported Health Status in NHANES Adults")
```

Self−Reported Health Status in NHANES Adults
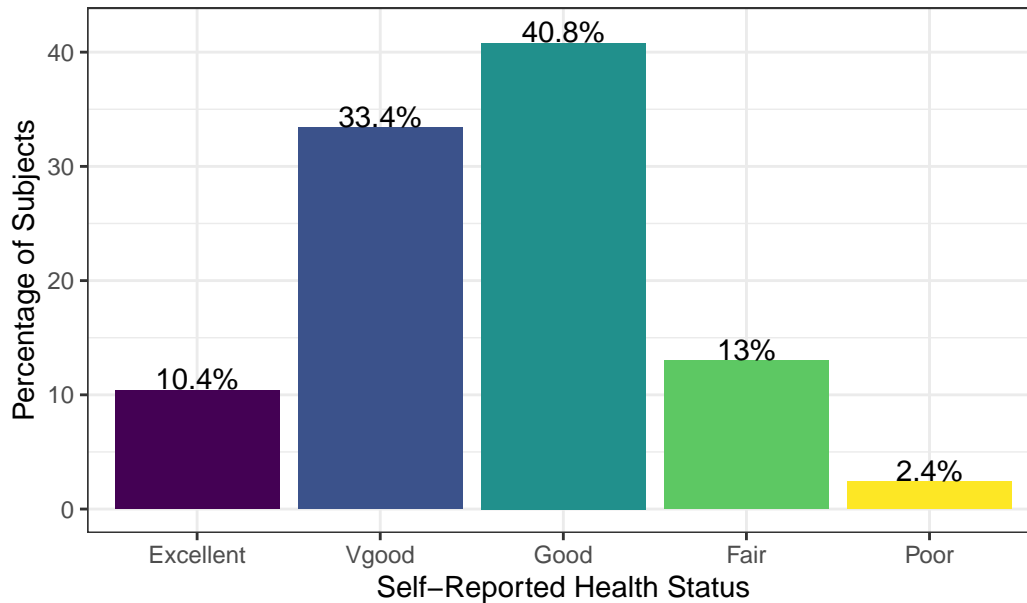
6.9 Two-Way Tables

We can create cross-classifications of two categorical variables (for example HealthGen and Smoke100), adding both row and column marginal totals, and compare subjects by Sex, as follows...

```
nh_500cc |>
    tabyl(Smoke100, HealthGen) |>
    adorn_totals(c("row", "col")) |>
    adorn_title()
```

```
          HealthGen
Smoke100 Excellent Vgood Good Fair Poor Total
      No        44   108  105   29    5   291
     Yes         8    59   99   36    7   209
   Total        52   167  204   65   12   500
```

If we like, we can make this look a little more polished with the `kbl()` and `kable_styling()` functions from the `kableExtra` package.

	HealthGen					
Smoke100	Excellent	Vgood	Good	Fair	Poor	Total
No	44	108	105	29	5	291
Yes	8	59	99	36	7	209
Total	52	167	204	65	12	500

	HealthGen				
Smoke100	Excellent	Vgood	Good	Fair	Poor
No	15.1% (44)	37.1% (108)	36.1% (105)	10.0% (29)	1.7% (5)
Yes	3.8% (8)	28.2% (59)	47.4% (99)	17.2% (36)	3.3% (7)
Total	10.4% (52)	33.4% (167)	40.8% (204)	13.0% (65)	2.4% (12)

```
nh_500cc |>
    tabyl(Smoke100, HealthGen) |>
    adorn_totals(c("row", "col")) |>
    adorn_title() |>
    kbl(align = 'crrrrr') |>
    kable_styling(position = "center", full_width = FALSE)
```

Or, we can get a complete cross-tabulation, including (in this case) the percentages of people within each of the two categories of `Smoke100` that fall in each `HealthGen` category (percentages within each row) like this.

```
nh_500cc |>
    tabyl(Smoke100, HealthGen) |>
    adorn_totals("row") |>
    adorn_percentages("row") |>
    adorn_pct_formatting() |>
    adorn_ns() |>
    adorn_title() |>
    kbl(align = 'crrrrr') |>
    kable_styling(position = "center", full_width = FALSE)
```

And, if we wanted the column percentages, to determine which sex had the higher rate of each HealthGen status level, we can get that by changing the adorn_percentages to describe results at the column level:

```
nh_500cc |>
    tabyl(Sex, HealthGen) |>
    adorn_totals("col") |>
    adorn_percentages("col") |>
```

Sex	HealthGen Excellent	Vgood	Good	Fair	Poor	Total
female	63.5% (33)	44.3% (74)	43.6% (89)	47.7% (31)	75.0% (9)	47.2% (236)
male	36.5% (19)	55.7% (93)	56.4% (115)	52.3% (34)	25.0% (3)	52.8% (264)

```
    adorn_pct_formatting() |>
    adorn_ns() |>
    adorn_title() |>
    kbl(align = 'crrrrr') |>
    kable_styling(position = "center", full_width = FALSE)
```

6.10 SBP by General Health Status

Let's consider now the relationship between self-reported overall health and systolic blood pressure.

```
ggplot(data = nh_500cc, aes(x = HealthGen, y = SBP,
                            fill = HealthGen)) +
    geom_boxplot() +
    labs(title = "SBP by Health Status, Overall Health for NHANES Adults",
         y = "Systolic Blood Pressure",
         x = "Self-Reported Overall Health") +
    guides(fill = "none")
```

HealthGen	count	mean(SBP)	median(SBP)
Excellent	52	113.9231	113
Vgood	167	117.5928	118
Good	204	121.5931	120
Fair	65	120.3846	118
Poor	12	122.8333	124

SBP by Health Status, Overall Health for NHANES Adults
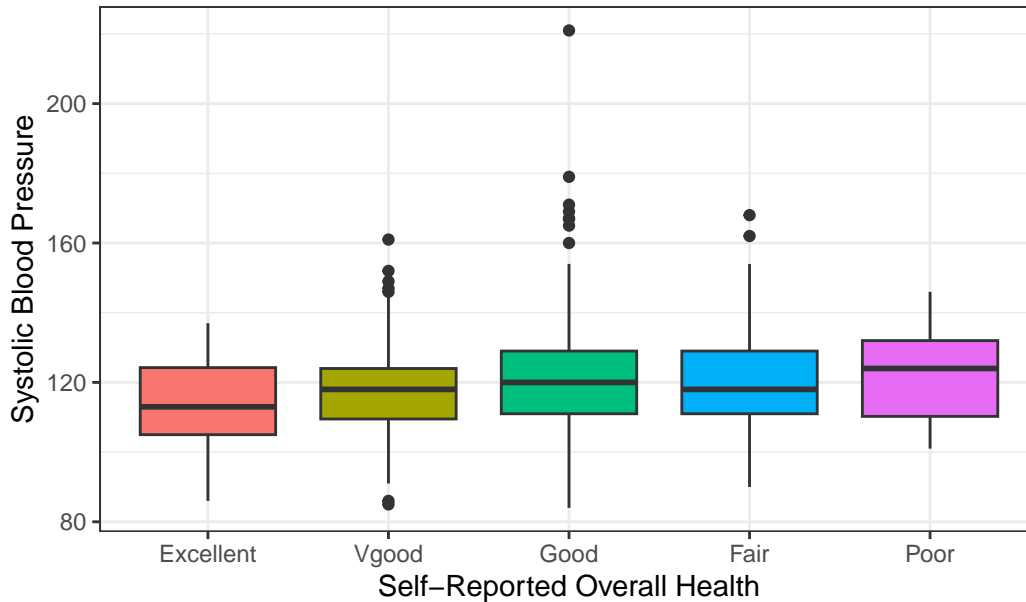
We can see that not too many people self-identify with the "Poor" health category.

```
nh_500cc |>
    group_by(HealthGen) |>
    summarise(count = n(), mean(SBP), median(SBP)) |>
    kbl() |>
    kable_styling(position = "center", full_width = FALSE)
```

6.10.1 SBP by Physical Activity and General Health Status

We'll build a panel of boxplots to try to understand the relationships between Systolic Blood Pressure, General Health Status and Physical Activity. Note the use of `coord_flip` to rotate the graph 90 degrees, and the use of `labeller` within `facet_wrap` to include both the name of the (Physical Activity) variable and its value.

```
ggplot(data = nh_500cc, aes(x = HealthGen, y = SBP, fill = HealthGen)) +
    geom_boxplot() +
    labs(title = "SBP by Health Status, Overall Health for NHANES Adults",
         y = "Systolic BP", x = "Self-Reported Overall Health") +
    guides(fill = "none") +
    facet_wrap(~ PhysActive, labeller = "label_both") +
    coord_flip()
```
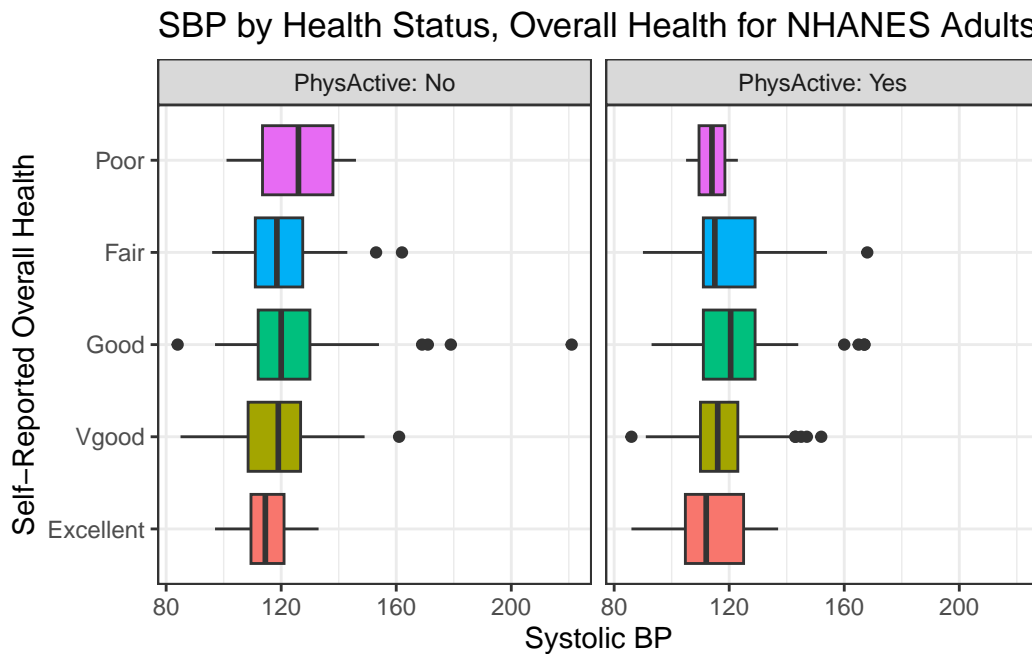

6.10.2 SBP by Sleep Trouble and General Health Status

Here's a plot of faceted histograms, which might be used to address similar questions related to the relationship between Overall Health, Systolic Blood Pressure and whether someone has trouble sleeping.

```
ggplot(data = nh_500cc, aes(x = SBP, fill = HealthGen)) +
    geom_histogram(color = "white", bins = 20) +
    labs(title = "SBP by Overall Health and Sleep Trouble for NHANES Adults",
         x = "Systolic BP") +
    guides(fill = "none") +
    facet_grid(SleepTrouble ~ HealthGen, labeller = "label_both")
```

SBP by Overall Health and Sleep Trouble for NHANES Adults
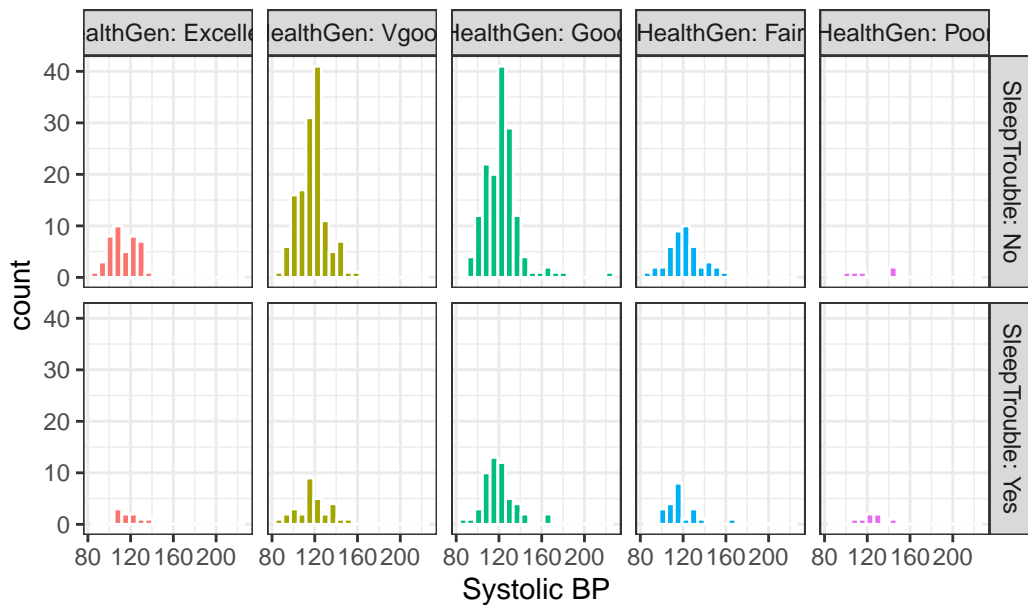
6.11 Conclusions

This is just a small piece of the toolbox for visualizations that we'll create in this class. Many additional tools are on the way, but the main idea won't change. Using the `ggplot2` package, we can accomplish several critical tasks in creating a visualization, including:

- Identifying (and labeling) the axes and titles
- Identifying a type of `geom` to use, like a point, bar or histogram
- Changing fill, color, shape, size to facilitate comparisons
- Building "small multiples" of plots with faceting

Good data visualizations make it easy to see the data, and `ggplot2`'s tools make it relatively difficult to make a really bad graph.

7 Summarizing Quantities

Most numerical summaries that might be new to you are applied most appropriately to quantitative variables. The measures that will interest us relate to:

- the **center** of our distribution,
- the **spread** of our distribution, and
- the **shape** of our distribution.

7.1 Setup: Packages Used Here

```r
knitr::opts_chunk$set(comment = NA)

library(gtsummary)
library(kableExtra)
library(patchwork)
library(summarytools)
library(tidyverse)

theme_set(theme_bw())
```

This chapter also requires that the `Hmisc`, `mosaic`, and `psych` packages are loaded on your machine, but these packages are not loaded with `library()` above.

7.2 Working with the `nh_750` data

To demonstrate key ideas in this Chapter, we will consider our sample of 750 adults ages 21-64 from NHANES 2011-12 which includes some missing values. We'll load into the `nh_750` data frame the information from the `nh_adult750.Rds` file we created in Section 4.3.2.

```r
nh_750 <- read_rds("data/nh_adult750.Rds")
```

126

7.3 The `summary` function for Quantitative data

R provides a small sampling of numerical summaries with the `summary` function, for instance.

```
nh_750 |>
  select(Age, BMI, SBP, DBP, Pulse) |>
  summary()
```

```
      Age              BMI              SBP              DBP
 Min.   :21.00   Min.   :16.70   Min.   : 83.0   Min.   :  0.00
 1st Qu.:30.00   1st Qu.:24.20   1st Qu.:108.0   1st Qu.: 66.00
 Median :40.00   Median :27.90   Median :118.0   Median : 73.00
 Mean   :40.82   Mean   :29.08   Mean   :118.8   Mean   : 72.69
 3rd Qu.:51.00   3rd Qu.:32.10   3rd Qu.:127.0   3rd Qu.: 80.00
 Max.   :64.00   Max.   :80.60   Max.   :209.0   Max.   :108.00
                 NA's   :5       NA's   :33      NA's   :33
     Pulse
 Min.   : 40.00
 1st Qu.: 66.00
 Median : 72.00
 Mean   : 73.53
 3rd Qu.: 80.00
 Max.   :124.00
 NA's   :32
```

This basic summary includes a set of five **quantiles**[1], plus the sample's **mean**.

- `Min.` = the **minimum** value for each variable, so, for example, the youngest subject's Age was 21.
- `1st Qu.` = the **first quartile** (25th percentile) for each variable - for example, 25% of the subjects were Age 30 or younger.
- `Median` = the **median** (50th percentile) - half of the subjects were Age 40 or younger.
- `Mean` = the **mean**, usually what one means by an *average* - the sum of the Ages divided by 750 is 40.8,
- `3rd Qu.` = the **third quartile** (75th percentile) - 25% of the subjects were Age 51 or older.
- `Max.` = the **maximum** value for each variable, so the oldest subject was Age 64.

The summary also specifies the number of missing values for each variable. Here, we are missing 5 of the BMI values, for example.

---

[1]The quantiles (sometimes referred to as percentiles) can also be summarized with a boxplot.

7.4 Measuring the Center of a Distribution

7.4.1 The Mean and The Median

The **mean** and **median** are the most commonly used measures of the center of a distribution for a quantitative variable. The median is the more generally useful value, as it is relevant even if the data have a shape that is not symmetric. We might also collect the **sum** of the observations, and the **count** of the number of observations, usually symbolized with n.

For variables without missing values, like `Age`, this is pretty straightforward.

```
nh_750 |>
    summarise(n = n(), Mean = mean(Age), Median = median(Age), Sum = sum(Age))
```

```
# A tibble: 1 x 4
      n  Mean Median   Sum
  <int> <dbl>  <dbl> <int>
1   750  40.8     40 30616
```

And again, the Mean is just the Sum (30616), divided by the number of non-missing values of Age (750), or 40.8213333.

The Median is the middle value when the data are sorted in order. When we have an odd number of values, this is sufficient. When we have an even number, as in this case, we take the mean of the two middle values. We could sort and list all 500 Ages, if we wanted to do so.

```
nh_750 |> select(Age) |>
    arrange(Age)
```

```
# A tibble: 750 x 1
    Age
  <int>
1    21
2    21
3    21
4    21
5    21
6    21
7    21
8    21
```

```
 9    21
10    21
# i 740 more rows
```

But this data set figures we don't want to output more than 10 observations to a table like this.

If we really want to see all of the data, we can use `View(nh_750)` to get a spreadsheet-style presentation, or use the `sort` command...

```
sort(nh_750$Age)
```

```
  [1]  21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 22 22 22
 [26]  22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 22 23 23 23 23 23 23
 [51]  23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 23 24 24 24 24 24 24 24 24 24
 [76]  24 24 24 24 24 24 24 24 24 24 24 24 24 24 25 25 25 25 25 25 25 25 25 25 25
[101]  25 25 25 26 26 26 26 26 26 26 26 26 26 26 26 26 26 27 27 27 27 27 27 27 27
[126]  27 27 27 27 27 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28 28
[151]  29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 29 30 30 30 30 30
[176]  30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 31 31 31 31 31 31
[201]  31 31 31 31 31 31 31 31 31 31 31 31 31 32 32 32 32 32 32 32 32 32 32 32 32
[226]  32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 33 33 33 33 33 33 33 33 33
[251]  33 33 33 33 33 33 33 33 33 33 33 33 33 33 33 34 34 34 34 34 34 34 34 34 34
[276]  34 34 34 34 35 35 35 35 35 35 35 35 35 35 35 36 36 36 36 36 36 36 36 36 36
[301]  36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 36 37 37 37 37 37 37 37 37 37
[326]  37 37 37 37 37 37 37 37 37 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38 38
[351]  39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 40 40 40 40 40 40
[376]  40 40 40 40 40 40 40 40 40 40 40 40 41 41 41 41 41 41 41 41 41 41 41 41 41
[401]  42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 42 43 43 43 43 43 43
[426]  43 43 43 43 43 43 43 43 43 43 43 43 44 44 44 44 44 44 44 44 44 44 44 44 44
[451]  44 44 44 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45 46 46 46 46 46
[476]  46 46 46 46 46 46 46 46 46 47 47 47 47 47 47 47 47 47 47 47 47 47 47 47 47
[501]  47 48 48 48 48 48 48 48 48 48 48 48 49 49 49 49 49 49 49 49 49 49 49 49 49
[526]  50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50 50
[551]  50 51 51 51 51 51 51 51 51 51 51 51 51 51 51 51 51 51 51 51 51 52 52 52 52
[576]  52 52 52 52 52 52 52 52 53 53 53 53 53 53 53 53 53 53 53 53 53 53 54 54
[601]  54 54 54 54 54 54 54 54 54 54 54 54 54 54 55 55 55 55 55 55 55 55 55 55 56
[626]  56 56 56 56 56 56 56 56 56 56 56 56 56 56 56 56 56 57 57 57 57 57 57 57
[651]  57 57 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 58 59 59 59 59 59
[676]  59 59 59 59 59 59 59 59 60 60 60 60 60 60 60 60 60 60 60 60 60 61 61 61 61
[701]  61 61 61 61 61 61 61 61 61 61 62 62 62 62 62 62 62 62 62 62 62 62 62 63 63
[726]  63 63 63 63 63 63 63 63 63 64 64 64 64 64 64 64 64 64 64 64 64 64 64
```

Again, to find the median, we would take the mean of the middle two observations in this sorted data set. That would be the 250th and 251st largest Ages.

```
sort(nh_750$Age)[250:251]
```

```
[1] 33 33
```

7.4.2 Dealing with Missingness

When calculating a mean, you may be tempted to try something like this...

```
nh_750 |>
    summarise(mean(Pulse), median(Pulse))
```

```
# A tibble: 1 x 2
  `mean(Pulse)` `median(Pulse)`
          <dbl>           <int>
1            NA              NA
```

This fails because we have some missing values in the Pulse data. We can address this by either omitting the data with missing values before we run the `summarise()` function, or tell the mean and median summary functions to remove missing values[2].

```
nh_750 |>
    filter(complete.cases(Pulse)) |>
    summarise(count = n(), mean(Pulse), median(Pulse))
```

```
# A tibble: 1 x 3
  count `mean(Pulse)` `median(Pulse)`
  <int>         <dbl>           <dbl>
1   718          73.5              72
```

Or, we could tell the summary functions themselves to remove NA values.

```
nh_750 |>
    summarise(mean(Pulse, na.rm=TRUE), median(Pulse, na.rm=TRUE))
```

---

[2]We could also use `!is.na` in place of `complete.cases` to accomplish the same thing.

```
# A tibble: 1 x 2
  `mean(Pulse, na.rm = TRUE)` `median(Pulse, na.rm = TRUE)`
                        <dbl>                         <dbl>
1                        73.5                            72
```

In Chapter 9, we will discuss various assumptions we can make about missing data, and the importance of **imputation** when dealing with it in modeling or making inferences. For now, we will limit our descriptive summaries to observed values, in what are called complete case or available case analyses.

7.4.3 The Mode of a Quantitative Variable

One other less common measure of the center of a quantitative variable's distribution is its most frequently observed value, referred to as the **mode**. This measure is only appropriate for discrete variables, be they quantitative or categorical. To find the mode, we usually tabulate the data, and then sort by the counts of the numbers of observations.

```
nh_750 |>
    group_by(Age) |>
    summarise(count = n()) |>
    arrange(desc(count))
```

```
# A tibble: 44 x 2
     Age count
   <int> <int>
 1    32    28
 2    36    26
 3    50    26
 4    30    24
 5    33    24
 6    24    23
 7    21    22
 8    22    22
 9    23    22
10    28    20
# i 34 more rows
```

The mode is just the most common Age observed in the data.

Note the use of three different "verbs" in our function there - for more explanation of this strategy, visit Hadley Wickham and Grolemund (2023). The `group_by` function here is very

useful. It converts the nh_750 data frame into a new grouped tibble where operations are performed on the groups. Here, this means that it groups the data by Age before counting observations, and then sorting the groups (the Ages) by their frequencies.

As an alternative, the `modeest` package's `mlv` function calculates the sample mode (or most frequent value) [3].

7.5 Measuring the Spread of a Distribution

Statistics is all about variation, so spread or dispersion is an important fundamental concept in statistics. Measures of spread like the inter-quartile range and range (maximum - minimum) can help us understand and compare data sets. If the values in the data are close to the center, the spread will be small. If many of the values in the data are scattered far away from the center, the spread will be large.

7.5.1 The Range and the Interquartile Range (IQR)

The **range** of a quantitative variable is sometimes interpreted as the difference between the maximum and the minimum, even though R presents the actual minimum and maximum values when you ask for a range...

```
nh_750 |>
    select(Age) |>
    range()
```

```
[1] 21 64
```

And, for a variable with missing values, we can use...

```
nh_750 |>
    select(BMI) |>
    filter(complete.cases(BMI)) |>
    range()
```

```
[1] 16.7 80.6
```

---

[3]See the documentation for the `modeest` package's `mlv` function to look at other definitions of the mode.

A more interesting and useful statistic is the **inter-quartile range**, or IQR, which is the range of the middle half of the distribution, calculated by subtracting the 25th percentile value from the 75th percentile value.

```
nh_750 |>
    summarise(IQR(Age), quantile(Age, 0.25), quantile(Age, 0.75))
```

```
# A tibble: 1 x 3
  `IQR(Age)` `quantile(Age, 0.25)` `quantile(Age, 0.75)`
      <dbl>                 <dbl>                 <dbl>
1        21                    30                    51
```

We can calculate the range and IQR nicely from the summary information on quantiles, of course:

```
nh_750 |>
    select(Age, BMI, SBP, DBP, Pulse) |>
    summary()
```

```
      Age             BMI             SBP             DBP
 Min.   :21.00   Min.   :16.70   Min.   : 83.0   Min.   :  0.00
 1st Qu.:30.00   1st Qu.:24.20   1st Qu.:108.0   1st Qu.: 66.00
 Median :40.00   Median :27.90   Median :118.0   Median : 73.00
 Mean   :40.82   Mean   :29.08   Mean   :118.8   Mean   : 72.69
 3rd Qu.:51.00   3rd Qu.:32.10   3rd Qu.:127.0   3rd Qu.: 80.00
 Max.   :64.00   Max.   :80.60   Max.   :209.0   Max.   :108.00
                 NA's   :5       NA's   :33      NA's   :33
     Pulse
 Min.   : 40.00
 1st Qu.: 66.00
 Median : 72.00
 Mean   : 73.53
 3rd Qu.: 80.00
 Max.   :124.00
 NA's   :32
```

7.5.2 The Variance and the Standard Deviation

The IQR is always a reasonable summary of spread, just as the median is always a reasonable summary of the center of a distribution. Yet, most people are inclined to summarize a batch

of data using two numbers: the **mean** and the **standard deviation**. This is really only a sensible thing to do if you are willing to assume the data follow a Normal distribution: a bell-shaped, symmetric distribution without substantial outliers.

But **most data do not (even approximately) follow a Normal distribution**. Summarizing by the median and quartiles (25th and 75th percentiles) is much more robust, explaining R's emphasis on them.

7.5.3 Obtaining the Variance and Standard Deviation in R

Here are the variances of the quantitative variables in the `nh_750` data. Note the need to include `na.rm = TRUE` to deal with the missing values in some variables.

```
nh_750 |>
    select(Age, BMI, SBP, DBP, Pulse) |>
    summarise_all(var, na.rm = TRUE)
```

```
# A tibble: 1 x 5
    Age   BMI   SBP   DBP Pulse
  <dbl> <dbl> <dbl> <dbl> <dbl>
1  157.  52.4  229.  128.  136.
```

And here are the standard deviations of those same variables.

```
nh_750 |>
    select(Age, BMI, SBP, DBP, Pulse) |>
    summarise_all(sd, na.rm = TRUE)
```

```
# A tibble: 1 x 5
    Age   BMI   SBP   DBP Pulse
  <dbl> <dbl> <dbl> <dbl> <dbl>
1  12.5  7.24  15.1  11.3  11.6
```

7.5.4 Defining the Variance and Standard Deviation

Bock, Velleman, and De Veaux (2004) have lots of useful thoughts here, which are lightly edited here.

In thinking about spread, we might consider how far each data value is from the mean. Such a difference is called a *deviation*. We could just average the deviations, but the positive and

negative differences always cancel out, leaving an average deviation of zero, so that's not helpful. Instead, we *square* each deviation to obtain non-negative values, and to emphasize larger differences. When we add up these squared deviations and find their mean (almost), this yields the **variance**.

$$\text{Variance} = s^2 = \frac{\Sigma(y - \bar{y})^2}{n - 1}$$

Why almost? It would be the mean of the squared deviations only if we divided the sum by n, but instead we divide by $n - 1$ because doing so produces an estimate of the true (population) variance that is unbiased[4]. If you're looking for a more intuitive explanation, this Stack Exchange link awaits your attention.

- To return to the original units of measurement, we take the square root of s^2, and instead work with s, the **standard deviation**, also abbreviated SD.

$$\text{Standard Deviation} = s = \sqrt{\frac{\Sigma(y - \bar{y})^2}{n - 1}}$$

7.5.5 Interpreting the SD when the data are Normally distributed

For a set of measurements that follow a Normal distribution, the interval:

- Mean $\pm$ Standard Deviation contains approximately 68% of the measurements;
- Mean $\pm$ 2(Standard Deviation) contains approximately 95% of the measurements;
- Mean $\pm$ 3(Standard Deviation) contains approximately all (99.7%) of the measurements.

We often refer to the population or process mean of a distribution with μ and the standard deviation with σ, leading to the Figure below.

But if the data are not from an approximately Normal distribution, then this Empirical Rule is less helpful.

---

[4] When we divide by n-1 as we calculate the sample variance, the average of the sample variances for all possible samples is equal to the population variance. If we instead divided by n, the average sample variance across all possible samples would be a little smaller than the population variance.
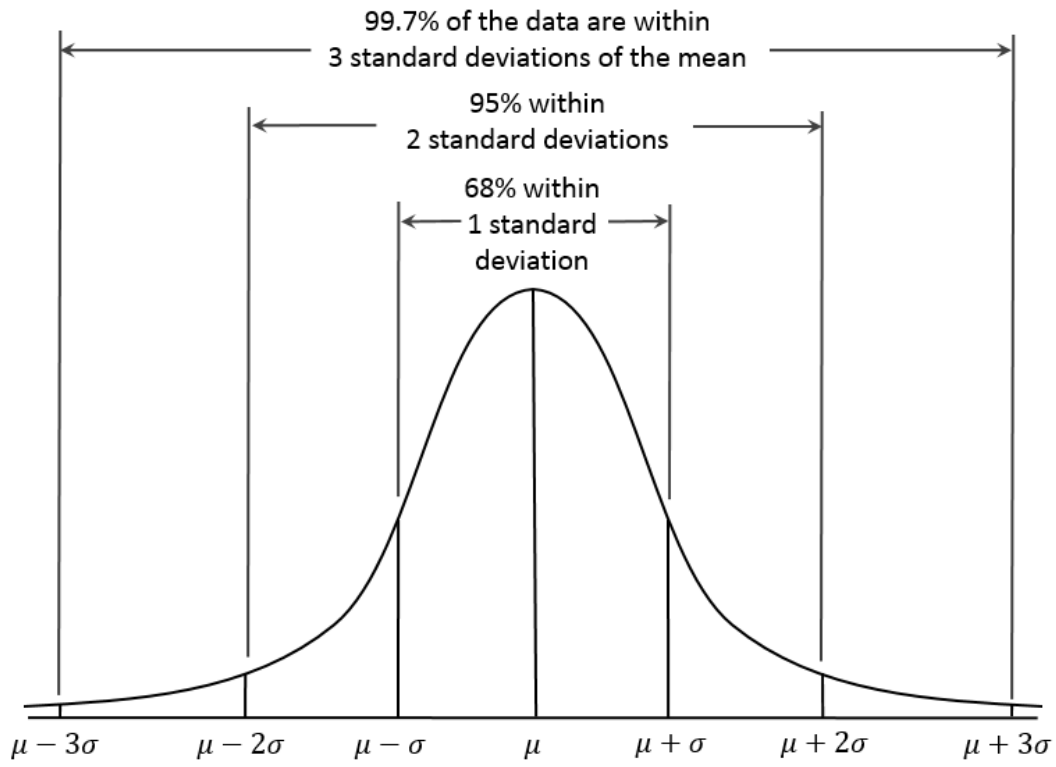
Figure 7.1: The Normal Distribution and the Empirical Rule

7.5.6 Chebyshev's Inequality: One Interpretation of the Standard Deviation

Chebyshev's Inequality tells us that for any distribution, regardless of its relationship to a Normal distribution, no more than $1/k^2$ of the distribution's values can lie more than k standard deviations from the mean. This implies, for instance, that for **any** distribution, at least 75% of the values must lie within two standard deviations of the mean, and at least 89% must lie within three standard deviations of the mean.

Again, most data sets do not follow a Normal distribution. We'll return to this notion soon. But first, let's try to draw some pictures that let us get a better understanding of the distribution of our data.

7.6 Measuring the Shape of a Distribution

When considering the shape of a distribution, one is often interested in three key points.

- The number of modes in the distribution, which I always assess through plotting the data.
- The **skewness**, or symmetry that is present, which I typically assess by looking at a plot of the distribution of the data, but if required to, will summarize with a non-parametric measure of **skewness**, that we will discuss in Section 10.13.
- The **kurtosis**, or heavy-tailedness (outlier-proneness) that is present, usually in comparison to a Normal distribution. Again, this is something I nearly inevitably assess graphically, but there are measures.

A Normal distribution has a single mode, is symmetric and, naturally, is neither heavy-tailed nor light-tailed as compared to a Normal distribution (we call this mesokurtic).

7.6.1 Multimodal vs. Unimodal distributions

A unimodal distribution, on some level, is straightforward. It is a distribution with a single mode, or "peak" in the distribution. Such a distribution may be skewed or symmetric, light-tailed or heavy-tailed. We usually describe as multimodal distributions like the two on the right below, which have multiple local maxima, even though they have just a single global maximum peak.

Truly multimodal distributions are usually described that way in terms of shape. For unimodal distributions, skewness and kurtosis become useful ideas.
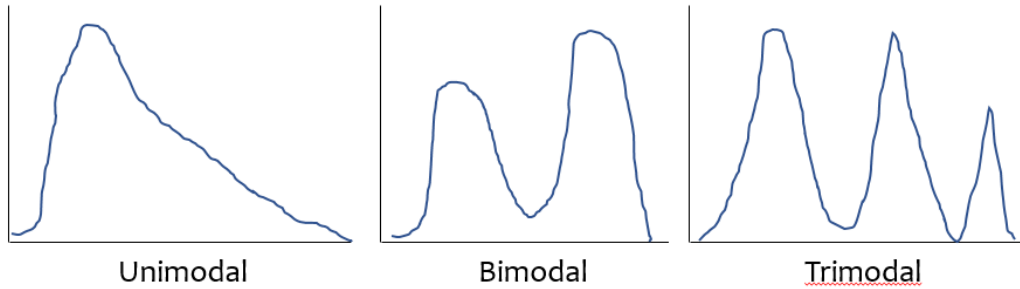
Figure 7.2: Unimodal and Multimodal Sketches

7.6.2 Skew

Whether or not a distribution is approximately symmetric is an important consideration in describing its shape. Graphical assessments are always most useful in this setting, particularly for unimodal data. My favorite measure of skew, or skewness if the data have a single mode, is:

$$skew_1 = \frac{\text{mean} - \text{median}}{\text{standard deviation}}$$

- Symmetric distributions generally show values of $skew_1$ near zero. If the distribution is actually symmetric, the mean should be equal to the median.
- Distributions with $skew_1$ values above 0.2 in absolute value generally indicate meaningful skew.
- Positive skew (mean > median if the data are unimodal) is also referred to as *right skew*.
- Negative skew (mean < median if the data are unimodal) is referred to as *left skew*.
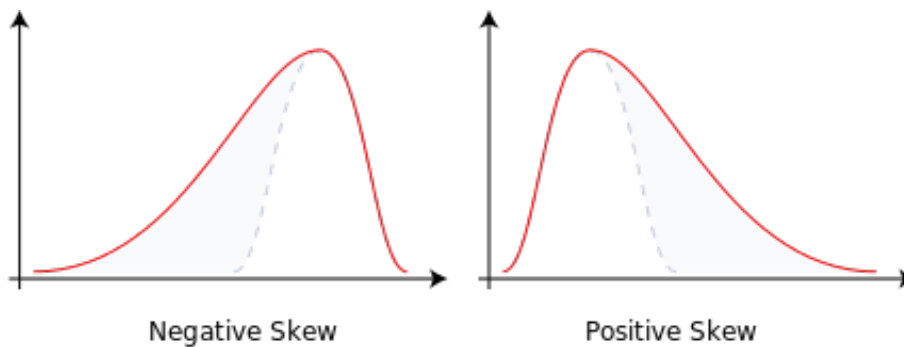
Figure 7.3: Negative (Left) Skew and Positive (Right) Skew

138

7.6.3 Kurtosis

When we have a unimodal distribution that is symmetric, we will often be interested in the behavior of the tails of the distribution, as compared to a Normal distribution with the same mean and standard deviation. High values of kurtosis measures (and there are several) indicate data which has extreme outliers, or is heavy-tailed.

- A mesokurtic distribution has similar tail behavior to what we would expect from a Normal distribution.
- A leptokurtic distribution is a thinner, more slender distribution, with heavier tails than we'd expect from a Normal distribution. One example is the t distribution.
- A platykurtic distribution is a broader, flatter distribution, with thinner tails than we'd expect from a Normal distribution. One example is a uniform distribution.

The visualization below (shown after the code) displays these three types of distributions.

```
set.seed(431)
sims_kurt <- tibble(meso = rnorm(n = 300, mean = 0, sd = 1),
                    lepto = rt(n = 300, df = 4),
                    platy = runif(n = 300, min = -2, max = 2))

p1 <- ggplot(sims_kurt, aes(x = meso)) +
  geom_histogram(aes(y = stat(density)),
                 bins = 25, fill = "royalblue", col = "white") +
  stat_function(fun = dnorm,
                args = list(mean = mean(sims_kurt$meso),
                            sd = sd(sims_kurt$meso)),
                col = "red") +
  labs(title = "Normal (mesokurtic)")

p1a <- ggplot(sims_kurt, aes(x = meso, y = "")) +
  geom_violin() +
  geom_boxplot(fill = "royalblue", outlier.color = "royalblue", width = 0.3) +
  labs(y = "", x = "Normal (mesokurtic)")

p2 <- ggplot(sims_kurt, aes(x = lepto)) +
  geom_histogram(aes(y = stat(density)),
                 bins = 25, fill = "tomato", col = "white") +
  stat_function(fun = dnorm,
                args = list(mean = mean(sims_kurt$lepto),
                            sd = sd(sims_kurt$lepto)),
                col = "royalblue") +
```

```
  labs(title = "t (leptokurtic)")

p2a <- ggplot(sims_kurt, aes(x = lepto, y = "")) +
  geom_violin() +
  geom_boxplot(fill = "tomato", outlier.color = "tomato", width = 0.3) +
  labs(y = "", x = "t (slender with heavy tails)")

p3 <- ggplot(sims_kurt, aes(x = platy)) +
  geom_histogram(aes(y = stat(density)),
                 bins = 25, fill = "yellow", col = "black") +
  stat_function(fun = dnorm,
                args = list(mean = mean(sims_kurt$platy),
                            sd = sd(sims_kurt$platy)),
                col = "royalblue", lwd = 1.5) +
  xlim(-3, 3) +
  labs(title = "Uniform (platykurtic)")

p3a <- ggplot(sims_kurt, aes(x = platy, y = "")) +
  geom_violin() +
  geom_boxplot(fill = "yellow", width = 0.3) +
  xlim(-3, 3) +
  labs(y = "", x = "Uniform (broad with thin tails)")


(p1 + p2 + p3) / (p1a + p2a + p3a) +
  plot_layout(heights = c(3, 1))
```
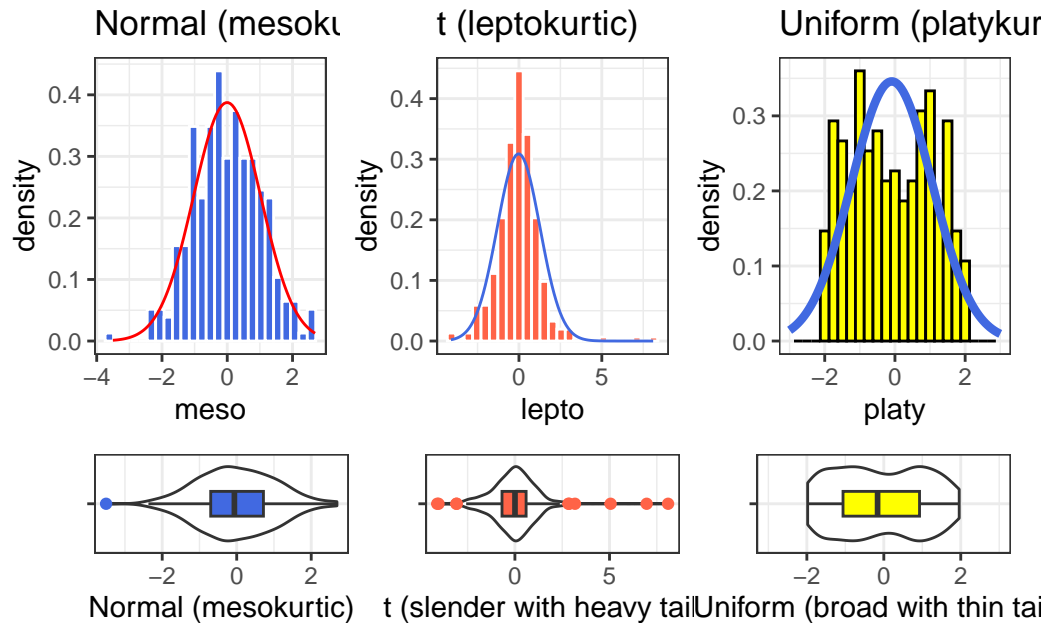
Graphical tools are in most cases the best way to identify issues related to kurtosis, and we usually only focus on kurtosis if we are willing to assume symmetry (or at least lack of meaningful skew) in our data.

7.7 Multiple Summaries at once

7.7.1 `favstats()` from the mosaic package

The `favstats` function adds the standard deviation, and counts of overall and missing observations to our usual `summary` for a continuous variable. Let's look at systolic blood pressure, because we haven't yet.

```
mosaic::favstats(~ SBP, data = nh_750)
```

```
 min  Q1 median  Q3 max     mean       sd   n missing
  83 108     118 127 209 118.7908 15.14329 717      33
```

We could, of course, duplicate these results with several `summarise()` pieces…

```
nh_750 |>
    filter(complete.cases(SBP)) |>
    summarise(min = min(SBP), Q1 = quantile(SBP, 0.25),
              median = median(SBP), Q3 = quantile(SBP, 0.75),
              max = max(SBP),  mean = mean(SBP),
              sd = sd(SBP), n = n(), miss = sum(is.na(SBP))) |>
    kbl(digits = 2)
```

min	Q1	median	Q3	max	mean	sd	n	miss
83	108	118	127	209	118.79	15.14	717	0

The somewhat unusual structure of `favstats` (complete with an easy to forget ~) is actually helpful. It allows you to look at some interesting grouping approaches, like this:

```
mosaic::favstats(SBP ~ Education, data = nh_750)
```

```
         Education min     Q1 median     Q3 max     mean       sd   n missing
1        8th Grade  96 110.25  119.5 129.75 167 122.4565 16.34993  46       4
2 9 - 11th Grade    85 107.75  116.0 127.00 191 118.8026 15.79453  76       0
3      High School  84 111.50  120.5 129.00 209 121.0882 16.52853 136       7
4    Some College   85 108.00  117.0 126.00 186 118.6293 14.32736 232       9
5    College Grad   83 107.00  117.0 125.00 171 116.8326 14.41202 227      13
```

Of course, we could accomplish the same comparison with `dplyr` commands, too, but the `favstats` approach has much to offer.

```
nh_750 |>
    filter(complete.cases(SBP, Education)) |>
    group_by(Education) |>
    summarise(min = min(SBP), Q1 = quantile(SBP, 0.25),
              median = median(SBP), Q3 = quantile(SBP, 0.75),
              max = max(SBP), mean = mean(SBP),
              sd = sd(SBP), n = n(), miss = sum(is.na(SBP))) |>
    kbl(digits = 2)
```

Education	min	Q1	median	Q3	max	mean	sd	n	miss
8th Grade	96	110.25	119.5	129.75	167	122.46	16.35	46	0
9 - 11th Grade	85	107.75	116.0	127.00	191	118.80	15.79	76	0
High School	84	111.50	120.5	129.00	209	121.09	16.53	136	0
Some College	85	108.00	117.0	126.00	186	118.63	14.33	232	0
College Grad	83	107.00	117.0	125.00	171	116.83	14.41	227	0

	Age	BMI	DBP	Pulse	SBP
Mean	40.82	29.08	72.69	73.53	118.79
Std.Dev	12.54	7.24	11.34	11.65	15.14
Min	21.00	16.70	0.00	40.00	83.00
Q1	30.00	24.20	66.00	66.00	108.00
Median	40.00	27.90	73.00	72.00	118.00
Q3	51.00	32.10	80.00	80.00	127.00
Max	64.00	80.60	108.00	124.00	209.00
MAD	14.83	5.93	10.38	11.86	13.34
IQR	21.00	7.90	14.00	14.00	19.00
CV	0.31	0.25	0.16	0.16	0.13
Skewness	0.16	1.72	-0.28	0.48	0.96
SE.Skewness	0.09	0.09	0.09	0.09	0.09
Kurtosis	-1.15	6.16	2.59	0.73	3.10
N.Valid	750.00	745.00	717.00	718.00	717.00
Pct.Valid	100.00	99.33	95.60	95.73	95.60

7.7.2 Using `descr()` from `summarytools`

The `descr()` function from the `summarytools` package produces numerous numerical summaries for quantities.

```
nh_750 |>
    select(Age, BMI, SBP, DBP, Pulse) |>
    descr() |>
    kbl(digits = 2) |>
    kable_styling(full_width = F)
```

The additional statistics presented here are:

- `MAD` = the median absolute deviation (from the median), which can be used in a manner similar to the standard deviation or IQR to measure spread.

    - If the data are $Y_1, Y_2, ..., Y_n$, then the `MAD` is defined as $median(|Y_i - median(Y_i)|)$.
    - To find the `MAD` for a set of numbers, find the median, subtract the median from each value and find the absolute value of that difference, and then find the median of those absolute differences.
    - For non-normal data with a skewed shape but tails well approximated by the Normal, the `MAD` is likely to be a better (more robust) estimate of the spread than is the standard deviation.

- `CV` = the coefficient of variation, or the standard deviation divided by the mean

	Age	BMI	DBP	Pulse	SBP
Mean	40.82	29.08	72.69	73.53	118.79
Std.Dev	12.54	7.24	11.34	11.65	15.14
Min	21.00	16.70	0.00	40.00	83.00
Median	40.00	27.90	73.00	72.00	118.00
Max	64.00	80.60	108.00	124.00	209.00
N.Valid	750.00	745.00	717.00	718.00	717.00
Pct.Valid	100.00	99.33	95.60	95.73	95.60

- a measure of `Skewness` and its standard error. This Skewness measure refers to how much asymmetry is present in the shape of the distribution. The measure is not the same as the *nonparametric skew* measure that we will usually prefer and that we discuss further in Section 10.13.
- Our nonparametric skew measure is just the difference between the mean and the median, divided by the standard deviation.
- The Wikipedia page on skewness is very detailed.
- a measure of `Kurtosis`, which refers to how outlier-prone, or heavy-tailed the shape of the distribution is, as compared to a Normal distribution.
- the number of valid (non-missing) observations in each variable.

Recall that in Chapter 3, we saw the use of an adjustment to show only "common" summaries. We can pick and choose from the entire list of available summaries. See the summarytools package vignette for more details.

```
nh_750 |>
  select(Age, BMI, SBP, DBP, Pulse) |>
  descr(stats = "common") |>
  kbl(digits = 2) |>
  kable_styling(full_width = F)
```

7.7.3 `describe` in the `psych` package

The `psych` package has an even more detailed list of numerical summaries for quantitative variables that lets us look at a group of observations at once.

```
psych::describe(nh_750 |> select(Age, BMI, SBP, DBP, Pulse))
```

```
      vars   n   mean    sd median trimmed   mad  min   max range  skew
Age      1 750  40.82 12.54   40.0   40.53 14.83 21.0  64.0  43.0  0.16
BMI      2 745  29.08  7.24   27.9   28.31  5.93 16.7  80.6  63.9  1.72
```

```
SBP      3 717 118.79 15.14  118.0  117.88 13.34 83.0 209.0 126.0  0.96
DBP      4 717  72.69 11.34   73.0   72.65 10.38  0.0 108.0 108.0 -0.28
Pulse    5 718  73.53 11.65   72.0   73.11 11.86 40.0 124.0  84.0  0.48
        kurtosis   se
Age       -1.15 0.46
BMI        6.16 0.27
SBP        3.10 0.57
DBP        2.59 0.42
Pulse      0.73 0.43
```

The additional statistics presented here, beyond those discussed previously, are:

- `trimmed` = a trimmed mean (by default in this function, this removes the top and bottom 10% from the data, then computes the mean of the remaining values - the middle 80% of the full data set.)
- `se` = the standard error of the sample mean, equal to the sample sd divided by the square root of the sample size.

7.7.4 The `Hmisc` package's version of `describe`

```
Hmisc::describe(nh_750 |>
                   select(Age, BMI, SBP, DBP, Pulse))
```

```
select(nh_750, Age, BMI, SBP, DBP, Pulse)

 5  Variables      750  Observations
--------------------------------------------------------------------------------
Age
       n  missing distinct    Info    Mean     Gmd     .05     .10
     750        0       44   0.999   40.82   14.46      22      24
     .25      .50      .75     .90     .95
      30       40       51      59      62

lowest : 21 22 23 24 25, highest: 60 61 62 63 64
--------------------------------------------------------------------------------
BMI
       n  missing distinct    Info    Mean     Gmd     .05     .10
     745        5      250       1   29.08   7.538   20.22   21.30
     .25      .50      .75     .90     .95
   24.20    27.90    32.10   37.60   41.28
```

145

```
lowest : 16.7 17.6 17.8 17.9 18  , highest: 59.1 62.8 63.3 69   80.6
--------------------------------------------------------------------------------
SBP
        n  missing distinct     Info     Mean      Gmd      .05      .10
      717       33       81    0.999    118.8    16.36     98.0    102.0
      .25      .50      .75      .90      .95
    108.0    118.0    127.0    137.0    144.2

lowest :  83  84  85  86  89, highest: 171 179 186 191 209
--------------------------------------------------------------------------------
DBP
        n  missing distinct     Info     Mean      Gmd      .05      .10
      717       33       66    0.999    72.69    12.43       55       59
      .25      .50      .75      .90      .95
       66       73       80       86       91

lowest :   0  25  41  42  44, highest: 104 105 106 107 108
--------------------------------------------------------------------------------
Pulse
        n  missing distinct     Info     Mean      Gmd      .05      .10
      718       32       37    0.997    73.53    12.95       56       60
      .25      .50      .75      .90      .95
       66       72       80       88       94

lowest :  40  44  46  48  50, highest: 108 112 114 118 124
--------------------------------------------------------------------------------
```

The `Hmisc` package's version of `describe` for a distribution of data presents three new ideas, in addition to a more comprehensive list of quartiles (the 5th, 10th, 25th, 50th, 75th, 90th and 95th are shown) and the lowest and highest few observations. These are:

- `distinct` - the number of different values observed in the data.
- `Info` - a measure of how "continuous" the variable is, related to how many "ties" there are in the data, with Info taking a higher value (closer to its maximum of one) if the data are more continuous.
- `Gmd` - the Gini mean difference - a robust measure of spread that is calculated as the mean absolute difference between any pairs of observations. Larger values of Gmd indicate more spread-out distributions. (Gini is usually pronounced as "Genie".)

146

7.7.5 Using tbl_summary() from `gtsummary`

If you want to produce results which look like you might expect to see in a published paper, the `tbl_summary()` function from the `gtsummary` package has many nice features. Here, we'll just show the medians, and 25th and 75th percentiles. Recall that we looked at some other options for this tool back in Chapter 3.

```
nh_750 |>
  select(Age, BMI, SBP, DBP, Pulse) |>
  tbl_summary()
```

```
Table printed with `knitr::kable()`, not {gt}. Learn why at
https://www.danieldsjoberg.com/gtsummary/articles/rmarkdown.html
To suppress this message, include `message = FALSE` in code chunk header.
```

Characteristic	**N = 750**
Age	40 (30, 51)
BMI	28 (24, 32)
Unknown	5
SBP	118 (108, 127)
Unknown	33
DBP	73 (66, 80)
Unknown	33
Pulse	72 (66, 80)
Unknown	32

7.7.6 Some Other Options

You'll recall that in our discussion of the Palmer Penguins, back in Chapter 3, we used several other strategies to develop graphical and numerical summaries of quantities, and all of those approaches could be used here, too.

- The DataExplorer package can be used for more automated exploratory data analyses.
- Some people also like skimr which has some similar goals.

Next, we'll focus on a few tools for summarizing categorical information.

8 Summarizing Categories

8.1 Setup: Packages Used Here

```r
knitr::opts_chunk$set(comment = NA)

library(janitor)
library(kableExtra)
library(gt)
library(tidyverse)

theme_set(theme_bw())
```

8.2 Using the `nh_adult750` data again

To demonstrate key ideas in this Chapter, we will again consider our sample of 750 adults ages 21-64 from NHANES 2011-12 which includes some missing values. We'll load into the `nh_750` data frame the information from the `nh_adult750.Rds` file we created in Section @ref(newNHANES).

```r
nh_750 <- read_rds("data/nh_adult750.Rds")
```

Summarizing categorical variables numerically is mostly about building tables, and calculating percentages or proportions. We'll save our discussion of modeling categorical data for later. Recall that in the `nh_750` data set we built in Section @ref(newNHANES) we had the following categorical variables. The number of levels indicates the number of possible categories for each categorical variable.

Variable	Description	Levels	Type
Sex	sex of subject	2	binary
Race	subject's race	6	nominal
Education	subject's educational level	5	ordinal
PhysActive	Participates in sports?	2	binary

Variable	Description	Levels	Type
Smoke100	Smoked 100+ cigarettes?	2	binary
SleepTrouble	Trouble sleeping?	2	binary
HealthGen	Self-report health	5	ordinal

8.3 The `summary` function for Categorical data

When R recognizes a variable as categorical, it stores it as a *factor*. Such variables get special treatment from the `summary` function, in particular a table of available values (so long as there aren't too many.)

```
nh_750 |>
  select(Sex, Race, Education, PhysActive, Smoke100,
         SleepTrouble, HealthGen, MaritalStatus) |>
  summary()
```

```
     Sex              Race                 Education    PhysActive Smoke100
 female:388    Asian   : 70    8th Grade     : 50    No :326    No :453
 male  :362    Black   :128    9 - 11th Grade: 76    Yes:424    Yes:297
               Hispanic: 63    High School   :143
               Mexican : 80    Some College  :241
               White   :393    College Grad  :240
               Other   : 16
 SleepTrouble      HealthGen          MaritalStatus
 No :555       Excellent: 84    Divorced    : 78
 Yes:195       Vgood    :197    LivePartner : 70
               Good     :252    Married     :388
               Fair     :104    NeverMarried:179
               Poor     : 14    Separated   : 19
               NA's     : 99    Widowed     : 16
```

8.4 Tables to describe One Categorical Variable

Suppose we build a table (using the `tabyl` function from the `janitor` package) to describe the `HealthGen` distribution.

```
nh_750 |>
    tabyl(HealthGen) |>
    adorn_pct_formatting()
```

```
HealthGen    n percent valid_percent
Excellent   84   11.2%          12.9%
    Vgood  197   26.3%          30.3%
     Good  252   33.6%          38.7%
     Fair  104   13.9%          16.0%
     Poor   14    1.9%           2.2%
     <NA>   99   13.2%              –
```

Note how the missing (`<NA>`) values are not included in the `valid_percent` calculation, but are in the `percent` calculation. Note also the use of percentage formatting.

What if we want to add a total count, sometimes called the *marginal* total?

```
nh_750 |>
    tabyl(HealthGen) |>
    adorn_totals() |>
    adorn_pct_formatting()
```

```
HealthGen    n percent valid_percent
Excellent   84   11.2%          12.9%
    Vgood  197   26.3%          30.3%
     Good  252   33.6%          38.7%
     Fair  104   13.9%          16.0%
     Poor   14    1.9%           2.2%
     <NA>   99   13.2%              –
    Total  750  100.0%         100.0%
```

What about marital status, which has no missing data in our sample?

```
nh_750 |>
    tabyl(MaritalStatus) |>
    adorn_totals() |>
    adorn_pct_formatting()
```

```
MaritalStatus    n percent
     Divorced   78   10.4%
```

```
   LivePartner  70    9.3%
      Married 388   51.7%
 NeverMarried 179   23.9%
    Separated  19    2.5%
      Widowed  16    2.1%
        Total 750  100.0%
```

8.5 Constructing Tables Well

The prolific Howard Wainer is responsible for many interesting books on visualization and related issues, including Wainer (2005) and Wainer (2013). These rules come from Chapter 10 of Wainer (1997).

1. Order the rows and columns in a way that makes sense.
2. Round, a lot!
3. ALL is different and important

8.5.1 Alabama First!

Which of these Tables is more useful to you?

2013 Percent of Students in grades 9-12 who are obese

State	% Obese	95% CI	Sample Size
Alabama	17.1	(14.6 - 19.9)	1,499
Alaska	12.4	(10.5-14.6)	1,167
Arizona	10.7	(8.3-13.6)	1,520
Arkansas	17.8	(15.7-20.1)	1,470
Connecticut	12.3	(10.2-14.7)	2,270
Delaware	14.2	(12.9-15.6)	2,475
Florida	11.6	(10.5-12.8)	5,491
…			
Wisconsin	11.6	(9.7-13.9)	2,771
Wyoming	10.7	(9.4-12.2)	2,910

or …

State	% Obese	95% CI	Sample Size
Kentucky	18.0	(15.7 - 20.6)	1,537
Arkansas	17.8	(15.7 - 20.1)	1,470

State	% Obese	95% CI	Sample Size
Alabama	17.1	(14.6 - 19.9)	1,499
Tennessee	16.9	(15.1 - 18.8)	1,831
Texas	15.7	(13.9 - 17.6)	3,039
...			
Massachusetts	10.2	(8.5 - 12.1)	2,547
Idaho	9.6	(8.2 - 11.1)	1,841
Montana	9.4	(8.4 - 10.5)	4,679
New Jersey	8.7	(6.8 - 11.2)	1,644
Utah	6.4	(4.8 - 8.5)	2,136

It is a rare event when Alabama first is the best choice.

8.5.2 ALL is different and important

Summaries of rows and columns provide a measure of what is typical or usual. Sometimes a sum is helpful, at other times, consider presenting a median or other summary. The ALL category, as Wainer (1997) suggests, should be both visually different from the individual entries and set spatially apart.

On the whole, it's *far* easier to fall into a good graph in R (at least if you have some ggplot2 skills) than to produce a good table.

8.6 The Mode of a Categorical Variable

A common measure applied to a categorical variable is to identify the mode, the most frequently observed value. To find the mode for variables with lots of categories (so that the `summary` may not be sufficient), we usually tabulate the data, and then sort by the counts of the numbers of observations, as we did with discrete quantitative variables.

```
nh_750 |>
    group_by(HealthGen) |>
    summarise(count = n()) |>
    arrange(desc(count))
```

```
# A tibble: 6 x 2
  HealthGen count
  <fct>     <int>
1 Good        252
```

```
2 Vgood          197
3 Fair           104
4 <NA>            99
5 Excellent       84
6 Poor            14
```

8.7 `describe` in the `Hmisc` package

```
Hmisc::describe(nh_750 |>
                select(Sex, Race, Education, PhysActive,
                       Smoke100, SleepTrouble,
                       HealthGen, MaritalStatus))
```

```
select(nh_750, Sex, Race, Education, PhysActive, Smoke100, SleepTrouble, HealthGen, MaritalSt

 8  Variables      750  Observations
--------------------------------------------------------------------------------
Sex
       n  missing distinct
     750        0        2

Value        female    male
Frequency       388     362
Proportion    0.517   0.483
--------------------------------------------------------------------------------
Race
       n  missing distinct
     750        0        6

Value         Asian    Black Hispanic  Mexican    White    Other
Frequency        70      128       63       80      393       16
Proportion    0.093    0.171    0.084    0.107    0.524    0.021
--------------------------------------------------------------------------------
Education
       n  missing distinct
     750        0        5

Value       8th Grade 9 - 11th Grade  High School  Some College
Frequency          50             76          143           241
Proportion      0.067          0.101        0.191         0.321
```

```
Value        College Grad
Frequency             240
Proportion          0.320
-------------------------------------------------------------------------------
PhysActive
       n  missing distinct
     750        0        2

Value         No   Yes
Frequency    326   424
Proportion 0.435 0.565
-------------------------------------------------------------------------------
Smoke100
       n  missing distinct
     750        0        2

Value         No   Yes
Frequency    453   297
Proportion 0.604 0.396
-------------------------------------------------------------------------------
SleepTrouble
       n  missing distinct
     750        0        2

Value         No  Yes
Frequency    555  195
Proportion 0.74 0.26
-------------------------------------------------------------------------------
HealthGen
       n  missing distinct
     651       99        5

Value      Excellent      Vgood       Good       Fair       Poor
Frequency         84        197        252        104         14
Proportion     0.129      0.303      0.387      0.160      0.022
-------------------------------------------------------------------------------
MaritalStatus
       n  missing distinct
     750        0        6

Value          Divorced LivePartner       Married NeverMarried     Separated
Frequency            78          70           388          179            19
```

```
Proportion           0.104           0.093           0.517           0.239           0.025

Value           Widowed
Frequency            16
Proportion     0.021
--------------------------------------------------------------------------------
```

8.8 Cross-Tabulations of Two Variables

It is very common for us to want to describe the association of one categorical variable with another. For instance, is there a relationship between Education and SleepTrouble in these data?

```
nh_750 |>
    tabyl(Education, SleepTrouble) |>
    adorn_totals(where = c("row", "col"))
```

```
     Education  No Yes Total
     8th Grade  40  10    50
 9 - 11th Grade  52  24    76
   High School 102  41   143
  Some College 173  68   241
  College Grad 188  52   240
         Total 555 195   750
```

Note the use of `adorn_totals` to get the marginal counts, and how we specify that we want both the row and column totals. We can add a title for the columns with...

```
nh_750 |>
    tabyl(Education, SleepTrouble) |>
    adorn_totals(where = c("row", "col")) |>
    adorn_title(placement = "combined")
```

```
 Education/SleepTrouble  No Yes Total
             8th Grade  40  10    50
         9 - 11th Grade  52  24    76
           High School 102  41   143
          Some College 173  68   241
          College Grad 188  52   240
                 Total 555 195   750
```

Often, we'll want to show percentages in a cross-tabulation like this. To get row percentages so that we can directly see the probability of `SleepTrouble = Yes` for each level of `Education`, we can use:

```
nh_750 |>
    tabyl(Education, SleepTrouble) |>
    adorn_totals(where = "row") |>
    adorn_percentages(denominator = "row") |>
    adorn_pct_formatting() |>
    adorn_title(placement = "combined")
```

```
Education/SleepTrouble    No    Yes
            8th Grade 80.0% 20.0%
        9 - 11th Grade 68.4% 31.6%
          High School 71.3% 28.7%
         Some College 71.8% 28.2%
         College Grad 78.3% 21.7%
                Total 74.0% 26.0%
```

If we want to compare the distribution of `Education` between the two levels of `SleepTrouble` with column percentages, we can use the following…

```
nh_750 |>
    tabyl(Education, SleepTrouble) |>
    adorn_totals(where = "col") |>
    adorn_percentages(denominator = "col") |>
    adorn_pct_formatting() |>
    adorn_title(placement = "combined")
```

```
Education/SleepTrouble    No    Yes Total
            8th Grade  7.2%  5.1%  6.7%
        9 - 11th Grade  9.4% 12.3% 10.1%
          High School 18.4% 21.0% 19.1%
         Some College 31.2% 34.9% 32.1%
         College Grad 33.9% 26.7% 32.0%
```

If we want overall percentages in the cells of the table, so that the total across all combinations of `Education` and `SleepTrouble` is 100%, we can use:

MaritalStatus/HealthGen	Excellent	Vgood	Good	Fair	Poor	NA_
Divorced	7 (9.0%)	19 (24.4%)	29 (37.2%)	11 (14.1%)	3 (3.8%)	9 (11.5%)
LivePartner	4 (5.7%)	19 (27.1%)	25 (35.7%)	18 (25.7%)	0 (0.0%)	4 (5.7%)
Married	46 (11.9%)	101 (26.0%)	130 (33.5%)	41 (10.6%)	6 (1.5%)	64 (16.5%)
NeverMarried	25 (14.0%)	52 (29.1%)	56 (31.3%)	24 (13.4%)	3 (1.7%)	19 (10.6%)
Separated	2 (10.5%)	3 (15.8%)	4 (21.1%)	8 (42.1%)	0 (0.0%)	2 (10.5%)
Widowed	0 (0.0%)	3 (18.8%)	8 (50.0%)	2 (12.5%)	2 (12.5%)	1 (6.2%)
Total	84 (11.2%)	197 (26.3%)	252 (33.6%)	104 (13.9%)	14 (1.9%)	99 (13.2%)

```
nh_750 |>
    tabyl(Education, SleepTrouble) |>
    adorn_totals(where = c("row", "col")) |>
    adorn_percentages(denominator = "all") |>
    adorn_pct_formatting() |>
    adorn_title(placement = "combined") |>
    kbl(align = 'lrrrrrr')
```

Education/SleepTrouble	No	Yes	Total
8th Grade	5.3%	1.3%	6.7%
9 - 11th Grade	6.9%	3.2%	10.1%
High School	13.6%	5.5%	19.1%
Some College	23.1%	9.1%	32.1%
College Grad	25.1%	6.9%	32.0%
Total	74.0%	26.0%	100.0%

Another common approach is to include both counts and percentages in a cross-tabulation. Let's look at the breakdown of `HealthGen` by `MaritalStatus`.

```
nh_750 |>
    tabyl(MaritalStatus, HealthGen) |>
    adorn_totals(where = c("row")) |>
    adorn_percentages(denominator = "row") |>
    adorn_pct_formatting() |>
    adorn_ns(position = "front") |>
    adorn_title(placement = "combined") |>
    kbl(align = 'lrrrrrr') |>
    kable_styling(full_width = FALSE)
```

What if we wanted to ignore the missing `HealthGen` values? Most often, I filter down to the complete observations.

MaritalStatus/HealthGen	Excellent	Vgood	Good	Fair	Poor
Divorced	7 (10.1%)	19 (27.5%)	29 (42.0%)	11 (15.9%)	3 (4.3%)
LivePartner	4 (6.1%)	19 (28.8%)	25 (37.9%)	18 (27.3%)	0 (0.0%)
Married	46 (14.2%)	101 (31.2%)	130 (40.1%)	41 (12.7%)	6 (1.9%)
NeverMarried	25 (15.6%)	52 (32.5%)	56 (35.0%)	24 (15.0%)	3 (1.9%)
Separated	2 (11.8%)	3 (17.6%)	4 (23.5%)	8 (47.1%)	0 (0.0%)
Widowed	0 (0.0%)	3 (20.0%)	8 (53.3%)	2 (13.3%)	2 (13.3%)
Total	84 (12.9%)	197 (30.3%)	252 (38.7%)	104 (16.0%)	14 (2.2%)

```
nh_750 |>
    filter(complete.cases(MaritalStatus, HealthGen)) |>
    tabyl(MaritalStatus, HealthGen) |>
    adorn_totals(where = c("row")) |>
    adorn_percentages(denominator = "row") |>
    adorn_pct_formatting() |>
    adorn_ns(position = "front") |>
    adorn_title(placement = "combined") |>
    kbl(align = 'lrrrrr') |>
    kable_styling(full_width = FALSE)
```

For more on working with `tabyls`, see this overview of janitor functions. There you'll find a complete list of all the `adorn` functions, for example.

Here's another approach, to look at the cross-classification of Race and HealthGen:

```
xtabs(~ Race + HealthGen, data = nh_750)
```

```
          HealthGen
Race       Excellent Vgood Good Fair Poor
  Asian           10    17   24    6    1
  Black           15    28   40   24    4
  Hispanic         4     9   24   13    2
  Mexican          6    12   25   21    2
  White           48   128  131   37    5
  Other            1     3    8    3    0
```

8.9 Cross-Classifying Three Categorical Variables

Suppose we are interested in `Smoke100` and its relationship to `PhysActive` and `SleepTrouble`.

```
nh_750 |>
    tabyl(Smoke100, PhysActive, SleepTrouble) |>
    adorn_title(placement = "top")
```

```
$No
         PhysActive
 Smoke100        No Yes
       No      137 219
      Yes       93 106


$Yes
         PhysActive
 Smoke100        No Yes
       No       41  56
      Yes       55  43
```

The result here is a tabyl of `Smoke100` (rows) by `PhysActive` (columns), split into a list by `SleepTrouble`.

There are several alternative approaches for doing this, although I expect us to stick with `tabyl` for our work in 431. These alternatives include the use of the `xtabs` function:

```
xtabs(~ Smoke100 + PhysActive + SleepTrouble, data = nh_750)
```

```
, , SleepTrouble = No

        PhysActive
Smoke100  No Yes
     No  137 219
     Yes  93 106

, , SleepTrouble = Yes

        PhysActive
Smoke100  No Yes
     No   41  56
     Yes  55  43
```

We can also build a **flat** version of this table, as follows:

```
ftable(Smoke100 ~ PhysActive + SleepTrouble, data = nh_750)
```

```
                       Smoke100  No Yes
PhysActive SleepTrouble
No         No                   137  93
           Yes                   41  55
Yes        No                   219 106
           Yes                   56  43
```

And we can do this with **dplyr** functions and the `table()` function, as well, for example…

```
nh_750 |>
    select(Smoke100, PhysActive, SleepTrouble) |>
    table()
```

```
, , SleepTrouble = No

        PhysActive
Smoke100  No Yes
     No  137 219
     Yes  93 106

, , SleepTrouble = Yes

        PhysActive
Smoke100  No Yes
     No   41  56
     Yes  55  43
```

8.10 Gaining Control over Tables in R: the gt package

With the **gt** package, anyone can make wonderful-looking tables using the R programming language. The **gt** package allows you to start with a tibble or data frame, and use it to make very detailed tables that look professional, and includes tools that enable you to include titles and subtitles, all sorts of labels, as well as footnotes and source notes.

Here's a fairly simple example of a cross-tabulation of part of the `nh_750` data built using a few tools from the **gt** package.

```
temp_tbl <- nh_750 |> filter(complete.cases(PhysActive, HealthGen)) |>
  tabyl(PhysActive, HealthGen) |>
  tibble()

gt(temp_tbl) |>
  tab_header(title = md("**Cross-Tabulation from nh_750**"),
             subtitle = md("Physical Activity vs. Overall Health"))
```

Cross-Tabulation from nh_750

Physical Activity vs. Overall Health

PhysActive	Excellent	Vgood	Good	Fair	Poor
No	24	66	126	59	10
Yes	60	131	126	45	4

The `gt` package and its usage is described in detail at https://gt.rstudio.com/.

8.11 Coming Up

Next, we'll make some early attempts at describing missingness in our data.

9 Missing Data and Single Imputation

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`.

9.1 Setup: Packages Used Here

```
knitr::opts_chunk$set(comment = NA)

library(janitor)
library(naniar)
library(simputation)
library(tidyverse)

theme_set(theme_bw())
```

We'll focus on tools from the **naniar** and **simputation** packages in this chapter. This chapter also requires that the **mosaic** package is loaded on your machine so we can use the `favstats()` function, but I won't load that here.

9.2 A Toy Example (n = 15)

In the following tiny data set called `sbp_example`, we have four variables for a set of 15 subjects. In addition to a subject id, we have:

- the treatment this subject received (A, B or C are the treatments),
- an indicator (1 = yes, 0 = no) of whether the subject has diabetes,
- the subject's systolic blood pressure at baseline
- the subject's systolic blood pressure after the application of the treatment

```
# create some temporary variables
subject <- 101:115
x1 <- c("A", "B", "C", "A", "C", "A", "A", NA, "B", "C", "A", "B", "C", "A", "B")
```

```
x2 <- c(1, 0, 0, 1, NA, 1, 0, 1, NA, 1, 0, 0, 1, 1, NA)
x3 <- c(120, 145, 150, NA, 155, NA, 135, NA, 115, 170, 150, 145, 140, 160, 135)
x4 <- c(105, 135, 150, 120, 135, 115, 160, 150, 130, 155, 140, 140, 150, 135, 120)

sbp_example <-
  tibble(subject, treat = factor(x1), diabetes = x2,
         sbp.before = x3, sbp.after = x4)

rm(subject, x1, x2, x3, x4) # just cleaning up

sbp_example
```

```
# A tibble: 15 x 5
   subject treat diabetes sbp.before sbp.after
     <int> <fct>    <dbl>      <dbl>     <dbl>
 1     101 A            1        120       105
 2     102 B            0        145       135
 3     103 C            0        150       150
 4     104 A            1         NA       120
 5     105 C           NA        155       135
 6     106 A            1         NA       115
 7     107 A            0        135       160
 8     108 <NA>         1         NA       150
 9     109 B           NA        115       130
10     110 C            1        170       155
11     111 A            0        150       140
12     112 B            0        145       140
13     113 C            1        140       150
14     114 A            1        160       135
15     115 B           NA        135       120
```

9.3 Identifying missingness with `naniar` functions

The **naniar** package has many useful functions.

1. How many missing values do we have, overall?

```
n_miss(sbp_example)
```

```
[1] 7
```

2. How many variables have missing values, overall?

```
n_var_miss(sbp_example)
```

[1] 3

3. Which variables contain missing values?

```
miss_var_which(sbp_example)
```

[1] "treat"   "diabetes"   "sbp.before"

4. How many missing values do we have in each variable?

```
miss_var_summary(sbp_example)
```

```
# A tibble: 5 x 3
  variable    n_miss pct_miss
  <chr>        <int>    <dbl>
1 diabetes         3    20
2 sbp.before       3    20
3 treat            1     6.67
4 subject          0     0
5 sbp.after        0     0
```

We are missing one `treat`, 3 `diabetes` and 3 `sbp.before` values.

5. Can we plot missingness, by variable?

```
gg_miss_var(sbp_example)
```
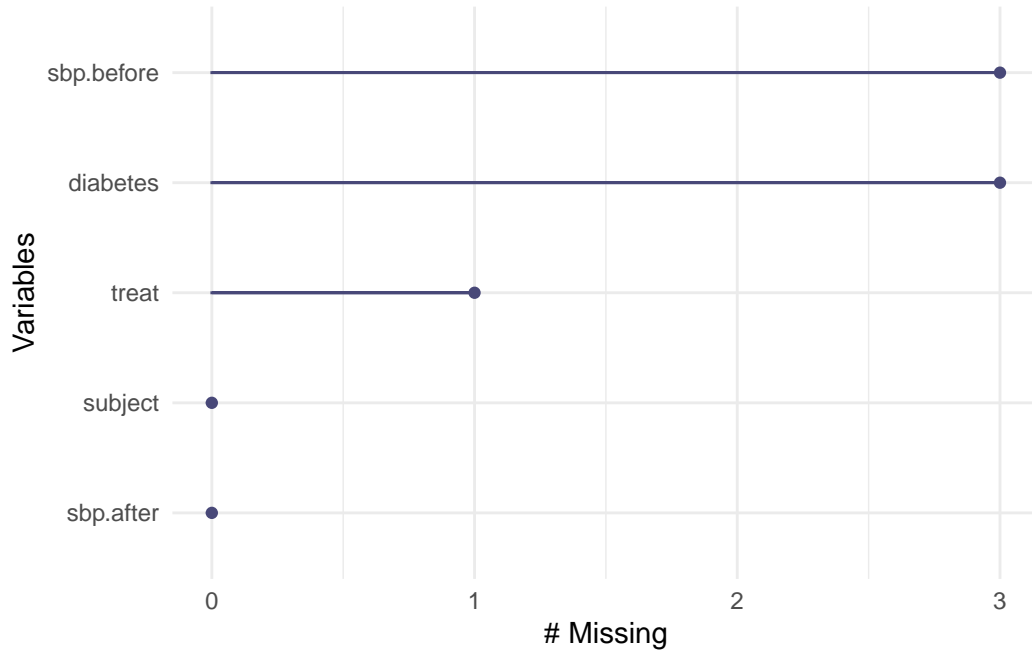
6. How many of the cases (rows) have missing values?

```
n_case_miss(sbp_example)
```

```
[1] 6
```

7. How many cases have complete data, with no missing values?

```
n_case_complete(sbp_example)
```

```
[1] 9
```

8. Can we tabulate missingness by case?

```
miss_case_table(sbp_example)
```

```
# A tibble: 3 x 3
  n_miss_in_case n_cases pct_cases
           <int>   <int>     <dbl>
1              0       9        60
2              1       5      33.3
3              2       1      6.67
```

9. Which cases have missing values?

```
miss_case_summary(sbp_example)
```

```
# A tibble: 15 x 3
    case n_miss pct_miss
   <int>  <int>    <dbl>
 1     8      2       40
 2     4      1       20
 3     5      1       20
 4     6      1       20
 5     9      1       20
 6    15      1       20
 7     1      0        0
 8     2      0        0
 9     3      0        0
10     7      0        0
11    10      0        0
12    11      0        0
13    12      0        0
14    13      0        0
15    14      0        0
```

10. How can we identify the subjects with missing data?

```
sbp_example |> filter(!complete.cases(sbp_example))
```

```
# A tibble: 6 x 5
  subject treat diabetes sbp.before sbp.after
    <int> <fct>    <dbl>      <dbl>     <dbl>
1     104 A            1         NA       120
2     105 C           NA        155       135
3     106 A            1         NA       115
4     108 <NA>         1         NA       150
5     109 B           NA        115       130
6     115 B           NA        135       120
```

We have nine subjects with complete data, three subjects with missing `diabetes` (only), two subjects with missing `sbp.before` (only), and 1 subject who is missing both `treat` and `sbp.before`.

9.4 Missing-data mechanisms

My source for this 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 the `diabetes` status 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.

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 nonresponse to `diabetes` 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 controls for all 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.

9.5 Dealing with Missingness: Three Approaches

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

9.6 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.

9.7 Building a 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 - a new data set with 9 rows (after dropping the six subjects with any NA values) and 5 columns.

```
cc.1 <- na.omit(sbp_example)
cc.2 <- sbp_example |> drop_na()
cc.3 <- sbp_example |> filter(complete.cases(sbp_example))
```

9.8 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.
- The `simputation` package can help us execute single imputations using a wide variety of techniques, within the pipe approach used by the `tidyverse`. Another approach I have used in the past is the `mice` package, which can also perform single imputations.

9.9 Single Imputation with the Mean or Mode

The most straightforward approach to single imputation is to impute a single summary of the variable, such as the mean, median or mode.

```
mosaic::favstats(~ sbp.before, data = sbp_example)
```

```
 min  Q1 median     Q3 max      mean       sd  n missing
 115 135     145 151.25 170 143.3333 15.71527 12        3
```

```
sbp_example |> tabyl(diabetes, treat) |>
  adorn_totals(where = c("row", "col"))
```

```
 diabetes A B C NA_ Total
        0 2 2 1   0     5
        1 4 0 2   1     7
     <NA> 0 2 1   0     3
    Total 6 4 4   1    15
```

Here, suppose we decide to impute

- `sbp.before` with the mean (143.3) among non-missing values,
- `diabetes` with its more common value, 1, and
- `treat` with its more common value, or mode (A)

```
si.1 <- sbp_example |>
    replace_na(list(sbp.before = 143.33,
                    diabetes = 1,
                    treat = "A"))
si.1
```

```
# A tibble: 15 x 5
   subject treat diabetes sbp.before sbp.after
     <int> <fct>    <dbl>      <dbl>     <dbl>
 1     101 A            1        120       105
 2     102 B            0        145       135
 3     103 C            0        150       150
 4     104 A            1        143.      120
 5     105 C            1        155       135
 6     106 A            1        143.      115
 7     107 A            0        135       160
 8     108 A            1        143.      150
 9     109 B            1        115       130
10     110 C            1        170       155
11     111 A            0        150       140
12     112 B            0        145       140
13     113 C            1        140       150
14     114 A            1        160       135
15     115 B            1        135       120
```

9.10 Doing Single Imputation with `simputation`

Single imputation is a potentially appropriate method when missingness can be assumed to
be either completely at random (MCAR) or dependent only on observed predictors (MAR).
We'll use the `simputation` package to accomplish it.

- The `simputation` vignette is available at https://cran.r-project.org/web/packages/simputation/vignettes/
- The `simputation` reference manual is available at https://cran.r-project.org/web/packages/simputation/si

Suppose we wanted to use:

- a robust linear model to predict `sbp.before` missing values, on the basis of `sbp.after`
  and `diabetes` status, and
- a predictive mean matching approach (which, unlike the robust linear model, will ensure
  that only values of `diabetes` that we've seen before will be imputed) to predict `diabetes`
  status, on the basis of `sbp.after`, and
- a decision tree approach to predict `treat` status, using all other variables in the data

```
set.seed(432009)

imp.2 <- sbp_example |>
    impute_rlm(sbp.before ~ sbp.after + diabetes) |>
```

```
    impute_pmm(diabetes ~ sbp.after) |>
    impute_cart(treat ~ .)

imp.2
```

```
# A tibble: 15 x 5
   subject treat diabetes sbp.before sbp.after
 *   <int> <fct>    <dbl>      <dbl>     <dbl>
 1     101 A            1        120       105
 2     102 B            0        145       135
 3     103 C            0        150       150
 4     104 A            1        139.      120
 5     105 C            1        155       135
 6     106 A            1        136.      115
 7     107 A            0        135       160
 8     108 A            1        155.      150
 9     109 B            1        115       130
10     110 C            1        170       155
11     111 A            0        150       140
12     112 B            0        145       140
13     113 C            1        140       150
14     114 A            1        160       135
15     115 B            1        135       120
```

Details on the many available methods in `simputation` are provided in its manual. These include:

- `impute_cart` uses a Classification and Regression Tree approach for numerical or categorical data. There is also an `impute_rf` command which uses Random Forests for imputation.
- `impute_pmm` is one of several "hot deck" options for imputation, this one is predictive mean matching, which can be used with numeric data (only). Missing values are first imputed using a predictive model. Next, these predictions are replaced with the observed values which are nearest to the prediction. Other imputation options in this group include random hot deck, sequential hot deck and k-nearest neighbor imputation.
- `impute_rlm` is one of several regression imputation methods, including linear models, robust linear models (which use what is called M-estimation to impute numerical variables) and lasso/elastic net/ridge regression models.

The `simputation` package can also do EM-based multivariate imputation, and multivariate random forest imputation, and several other approaches.

9.11 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.

- We'll postpone further discussion of multiple imputation to later in the semester, when we're building models for relationships in our data.

9.12 Coming Up

Next, we'll look an even more detailed look at how we might summarize results from another large data set, this time from the National Youth Fitness Survey.

10 National Youth Fitness Survey

10.1 Setup: Packages Used Here

```r
knitr::opts_chunk$set(comment = NA)

library(ggridges)
library(janitor)
library(kableExtra)
library(patchwork)
library(tidyverse)

theme_set(theme_bw())
```

We also use functions from the `Hmisc` and `mosaic` packages in this chapter, but do not load the whole packages.

10.2 What is the NHANES NYFS?

The `nnyfs.csv` and the `nnyfs.Rds` data files were built by Professor Love using data from the 2012 National Youth Fitness Survey.

> The NHANES National Youth Fitness Survey (NNYFS) was conducted in 2012 to collect data on physical activity and fitness levels in order to provide an evaluation of the health and fitness of children in the U.S. ages 3 to 15. The NNYFS collected data on physical activity and fitness levels of our youth through interviews and fitness tests.

In the `nnyfs` data file (either `.csv` or `.Rds`), I'm only providing a modest fraction of the available information. More on the NNYFS (including information I'm not using) is available at https://wwwn.cdc.gov/nchs/nhanes/search/nnyfs12.aspx.

The data elements I'm using fall into four main groups, or components:

- Demographics
- Dietary

- [Examination](#) and
- [Questionnaire](#)

What I did was merge a few elements from each of the available components of the NHANES National Youth Fitness Survey, reformulated (and in some cases simplified) some variables, and restricted the sample to kids who had completed elements of each of the four components.

10.3 The Variables included in `nnyfs`

This section tells you where the data come from, and briefly describe what is collected.

10.3.1 From the NNYFS Demographic Component

All of these come from the `Y_DEMO` file.

In `nnyfs`	In `Y_DEMO`	Description
SEQN	SEQN	Subject ID, connects all of the files
sex	RIAGENDR	Really, this is sex, not gender
age_child	RIDAGEYR	Age in years at screening
race_eth	RIDRETH1	Race/Hispanic origin (collapsed to 4 levels)
educ_child	DMDEDUC3	Education Level (for children ages 6-15). 0 = Kindergarten, 9 = Ninth grade or higher
language	SIALANG	Language in which the interview was conducted
sampling_wt	WTMEC	Full-sample MEC exam weight (for inference)
income_pov	INDFMPIR	Ratio of family income to poverty (ceiling is 5.0)
age_adult	DMDHRAGE	Age of adult who brought child to interview
educ_adult	DMDHREDU	Education level of adult who brought child

10.3.2 From the NNYFS Dietary Component

From the `Y_DR1TOT` file, we have a number of variables related to the child's diet, with the following summaries mostly describing consumption "yesterday" in a dietary recall questionnaire.

In `nnyfs`	In `Y_DR1TOT`	Description
respondent	DR1MNRSP	who responded to interview (child, Mom, someone else)
salt_used	DBQ095Z	uses salt, lite salt or salt substitute at the table
energy	DR1TKCAL	energy consumed (kcal)

In `nnyfs`	In `Y_DR1TOT`	Description
protein	DR1TPROT	protein consumed (g)
sugar	DR1TSUGR	total sugar consumed (g)
fat	DR1TTFAT	total fat consumed (g)
diet_yesterday	DR1_300	compare food consumed yesterday to usual amount
water	DR1_320Z	total plain water drank (g)

10.3.3 From the NNYFS Examination Component

From the `Y_BMX` file of Body Measures:

In `nnyfs`	In `Y_BMX`	Description
height	BMXHT	standing height (cm)
weight	BMXWT	weight (kg)
bmi	BMXBMI	body mass index (kg/m^2)
bmi_cat	BMDBMIC	BMI category (4 levels)
arm_length	BMXARML	Upper arm length (cm)
waist	BMXWAIST	Waist circumference (cm)
arm_circ	BMXARMC	Arm circumference (cm)
calf_circ	BMXCALF	Maximal calf circumference (cm)
calf_skinfold	BMXCALFF	Calf skinfold (mm)
triceps_skinfold	BMXTRI	Triceps skinfold (mm)
subscapular_skinfold	BMXSUB	Subscapular skinfold (mm)

From the `Y_PLX` file of Plank test results:

In `nnyfs`	In `Y_PLX`	Description
plank_time	MPXPLANK	# of seconds plank position is held

10.3.4 From the NNYFS Questionnaire Component

From the `Y_PAQ` file of Physical Activity questions:

In `nnyfs`	In `Y_PAQ`	Description
active_days	PAQ706	Days physically active (≥ 60 min.) in past week
tv_hours	PAQ710	Average hours watching TV/videos past 30d
computer_hours	PAQ715	Average hours on computer past 30d

In `nnyfs`	In `Y_PAQ`	Description
physical_last_week	PAQ722	Any physical activity outside of school past week
enjoy_recess	PAQ750	Enjoy participating in PE/recess

From the `Y_DBQ` file of Diet Behavior and Nutrition questions:

In `nnyfs`	In `Y_DBQ`	Description
meals_out	DBD895	# meals not home-prepared in past 7 days

From the `Y_HIQ` file of Health Insurance questions:

In `nnyfs`	In `Y_HIQ`	Description
insured	HIQ011	Covered by Health Insurance?
insurance	HIQ031	Type of Health Insurance coverage

From the `Y_HUQ` file of Access to Care questions:

In `nnyfs`	In `Y_HUQ`	Description
phys_health	HUQ010	General health condition (Excellent - Poor)
access_to_care	HUQ030	Routine place to get care?
care_source	HUQ040	Type of place most often goes to for care

From the `Y_MCQ` file of Medical Conditions questions:

In `nnyfs`	In `Y_MCQ`	Description
asthma_ever	MCQ010	Ever told you have asthma?
asthma_now	MCQ035	Still have asthma?

From the `Y_RXQ_RX` file of Prescription Medication questions:

In `nnyfs`	In `Y_RXQ_RX`	Description
med_use	RXDUSE	Taken prescription medication in last month?
med_count	RXDCOUNT	# of prescription meds taken in past month

10.4 Reading in the Data

Now, I'll take a look at the `nnyfs` data, which I've made available in a comma-separated version (`nnyfs.csv`), if you prefer, as well as in an R data set (`nnyfs.Rds`) which loads a bit faster. After loading the file, let's get a handle on its size and contents. In my R Project for these notes, the data are contained in a separate `data` subdirectory.

```
nnyfs <- readRDS("data/nnyfs.Rds")

## size of the tibble
dim(nnyfs)
```

```
[1] 1518    45
```

There are 1518 rows (subjects) and 45 columns (variables), by which I mean that there are 1518 kids in the `nnyfs` data frame, and we have 45 pieces of information on each subject. So, what do we have, exactly?

```
nnyfs
```

```
# A tibble: 1,518 x 45
     SEQN sex    age_child race_eth    educ_child language sampling_wt income_pov
    <dbl> <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>, ...
```

We can learn something about the structure of the tibble from such functions as `str` or `glimpse`.

```
str(nnyfs)
```

```
tibble [1,518 x 45] (S3: tbl_df/tbl/data.frame)
 $ SEQN                : num [1:1518] 71917 71918 71919 71920 71921 ...
 $ sex                 : Factor w/ 2 levels "Female","Male": 1 1 1 1 2 2 2 1 2 2 ...
 $ age_child           : num [1:1518] 15 8 14 15 3 12 12 8 7 8 ...
 $ race_eth            : Factor w/ 4 levels "1_Hispanic","2_White Non-Hispanic",..: 3 3 2 2 2
 $ educ_child          : num [1:1518] 9 2 8 8 NA 6 5 2 0 2 ...
 $ language            : Factor w/ 2 levels "English","Spanish": 1 1 1 1 1 1 1 1 1 1 ...
 $ sampling_wt         : num [1:1518] 28299 15127 29977 80652 55592 ...
 $ income_pov          : num [1:1518] 0.21 5 5 0.87 4.34 5 5 2.74 0.46 1.57 ...
 $ age_adult           : num [1:1518] 46 46 42 53 31 42 39 31 45 56 ...
 $ educ_adult          : Factor w/ 5 levels "1_Less than 9th Grade",..: 2 3 5 3 3 4 2 3 2 3
 $ respondent          : Factor w/ 3 levels "Child","Mom",..: 1 2 1 1 2 1 1 1 2 1 ...
 $ salt_used           : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 2 2 1 2 ...
 $ energy              : num [1:1518] 2844 1725 2304 1114 1655 ...
 $ protein             : num [1:1518] 169.1 55.2 199.3 14 50.6 ...
 $ sugar               : num [1:1518] 128.2 118.7 81.4 119.2 90.3 ...
 $ fat                 : num [1:1518] 127.9 63.7 86.1 36 53.3 ...
 $ diet_yesterday      : Factor w/ 3 levels "1_Much more than usual",..: 2 2 2 2 2 2 1 2 2 3
 $ water               : num [1:1518] 607 178 503 859 148 ...
 $ plank_time          : num [1:1518] NA 45 121 45 11 107 127 44 184 58 ...
 $ height              : num [1:1518] NA 131.6 172 167.1 90.2 ...
 $ weight              : num [1:1518] NA 38.6 58.7 92.5 12.4 66.4 56.7 22.2 20.9 28.3 ...
 $ bmi                 : num [1:1518] NA 22.3 19.8 33.1 15.2 25.9 22.5 14.4 15.9 17 ...
 $ bmi_cat             : Factor w/ 4 levels "1_Underweight",..: NA 4 2 4 2 4 3 2 2 2 ...
 $ arm_length          : num [1:1518] NA 27.7 38.4 35.9 18.3 34.2 33 26.5 24.2 26 ...
 $ waist               : num [1:1518] NA 71.9 79.4 96.4 46.8 90 72.3 56.1 54.5 59.7 ...
 $ arm_circ            : num [1:1518] NA 25.4 26 37.9 15.1 29.5 27.9 17.6 17.7 19.9 ...
 $ calf_circ           : num [1:1518] NA 32.3 35.3 46.8 19.4 36.9 36.8 24 24.3 27.3 ...
 $ calf_skinfold       : num [1:1518] NA 22 18.4 NA 8.4 22 18.3 7 7.2 8.2 ...
 $ triceps_skinfold    : num [1:1518] NA 19.9 15 20.6 8.6 22.8 20.5 12.9 6.9 8.8 ...
 $ subscapular_skinfold: num [1:1518] NA 17.4 9.8 22.8 5.7 24.4 12.6 6.8 4.8 6.1 ...
 $ active_days         : num [1:1518] 3 5 3 3 7 2 5 3 7 7 ...
 $ tv_hours            : num [1:1518] 2 2 1 3 2 3 0 4 2 2 ...
 $ computer_hours      : num [1:1518] 1 2 3 3 0 1 0 3 1 1 ...
 $ physical_last_week  : Factor w/ 2 levels "No","Yes": 1 1 2 2 2 2 2 2 2 2 ...
 $ enjoy_recess        : Factor w/ 5 levels "1_Strongly Agree",..: 1 1 3 2 NA 2 2 NA 1 1 ...
 $ meals_out           : num [1:1518] 0 2 3 2 1 1 2 1 0 2 ...
```

```
 $ insured            : Factor w/ 2 levels "Has Insurance",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ phys_health        : Factor w/ 5 levels "1_Excellent",..: 1 3 1 3 1 1 3 1 2 1 ...
 $ access_to_care     : Factor w/ 2 levels "Has Usual Care Source",..: 1 1 1 1 1 1 1 1 1 1
 $ care_source        : Factor w/ 6 levels "Clinic or Health Center",..: 1 2 2 2 2 2 2 2 2
 $ asthma_ever        : Factor w/ 2 levels "History of Asthma",..: 2 1 2 1 2 2 2 2 2 2 ...
 $ asthma_now         : Factor w/ 2 levels "Asthma Now","No Asthma Now": 2 1 2 1 2 2 2 2 2
 $ med_use            : Factor w/ 2 levels "Had Medication",..: 2 1 2 1 2 2 2 2 2 2 ...
 $ med_count          : num [1:1518] 0 1 0 2 0 0 0 0 0 0 ...
 $ insurance          : Factor w/ 10 levels "Medicaid","Medicare",..: 8 8 5 8 5 5 5 5 8 1 .
```

There are a lot of variables here. Let's run through the first few in a little detail.

10.4.1 `SEQN`

The first variable, `SEQN` is just a (numerical) identifying code attributable to a given subject of the survey. This is *nominal* data, which will be of little interest down the line. On some occasions, as in this case, the ID numbers are sequential, in the sense that subject 71919 was included in the data base after subject 71918, but this fact isn't particularly interesting here, because the protocol remained unchanged throughout the study.

10.4.2 `sex`

The second variable, `sex`, is listed as a factor variable (R uses **factor** and **character** to refer to categorical, especially non-numeric information). Here, as we can see below, we have two levels, *Female* and *Male*.

```
  nnyfs |>
    tabyl(sex) |>
    adorn_totals() |>
    adorn_pct_formatting()
```

```
    sex     n percent
 Female   760   50.1%
   Male   758   49.9%
  Total  1518  100.0%
```

179

10.4.3 `age_child`

The third variable, `age_child`, is the age of the child at the time of their screening to be in the study, measured in years. Note that age is a continuous concept, but the measure used here (number of full years alive) is a common discrete approach to measurement. Age, of course, has a meaningful zero point, so this can be thought of as a ratio variable; a child who is 6 is half as old as one who is 12. We can tabulate the observed values, since there are only a dozen or so.

```
nnyfs |>
  tabyl(age_child) |>
  adorn_pct_formatting()
```

```
age_child    n percent
        3 110    7.2%
        4 112    7.4%
        5 114    7.5%
        6 129    8.5%
        7 123    8.1%
        8 112    7.4%
        9  99    6.5%
       10 124    8.2%
       11 111    7.3%
       12 137    9.0%
       13 119    7.8%
       14 130    8.6%
       15  98    6.5%
```

At the time of initial screening, these children should have been between 3 and 15 years of age, so things look reasonable. Since this is a meaningful quantitative variable, we may be interested in a more descriptive summary.

```
nnyfs |>
  select(age_child) |>
  summary()
```

```
  age_child
 Min.   : 3.000
 1st Qu.: 6.000
 Median : 9.000
 Mean   : 9.033
```

```
3rd Qu.:12.000
Max.   :15.000
```

These six numbers provide a nice, if incomplete, look at the ages.

- `Min.` = the minimum, or youngest age at the examination was 3 years old.
- `1st Qu.` = the first quartile (25th percentile) of the ages was 6. This means that 25 percent of the subjects were age 6 or less.
- `Median` = the second quartile (50th percentile) of the ages was 9. This is often used to describe the center of the data. Half of the subjects were age 9 or less.
- `3rd Qu.` = the third quartile (75th percentile) of the ages was 12
- `Max.` = the maximum, or oldest age at the examination was 15 years.

We could get the standard deviation and a count of missing and non-missing observations with `favstats` from the `mosaic` package, among many other options (see Chapter 3 and @#sec-summ_quant for examples.)

```
mosaic::favstats(~ age_child, data = nnyfs) |>
  kbl(digits = 1)
```

```
Registered S3 method overwritten by 'mosaic':
  method                          from
  fortify.SpatialPolygonsDataFrame ggplot2
```

min	Q1	median	Q3	max	mean	sd	n	missing
3	6	9	12	15	9	3.7	1518	0

10.4.4 `race_eth`

The fourth variable in the data set is `race_eth`, which is a multi-categorical variable describing the child's race and ethnicity.

```
nnyfs |> tabyl(race_eth) |>
  adorn_pct_formatting() |>
  kbl()
```

race_eth	n	percent
1_Hispanic	450	29.6%
2_White Non-Hispanic	610	40.2%
3_Black Non-Hispanic	338	22.3%
4_Other Race/Ethnicity	120	7.9%

And now, we get the idea of looking at whether our numerical summaries of the children's ages varies by their race/ethnicity...

```
mosaic::favstats(age_child ~ race_eth, data = nnyfs)
```

```
           race_eth min   Q1 median Q3 max     mean       sd  n missing
1          1_Hispanic  3 5.25    9.0 12  15 8.793333 3.733846 450       0
2   2_White Non-Hispanic  3 6.00    9.0 12  15 9.137705 3.804421 610       0
3   3_Black Non-Hispanic  3 6.00    9.0 12  15 9.038462 3.576423 338       0
4 4_Other Race/Ethnicity  3 7.00    9.5 12  15 9.383333 3.427970 120       0
```

10.4.5 `income_pov`

Skipping down a bit, let's look at the family income as a multiple of the poverty level. Here's the summary.

```
nnyfs |>
  select(income_pov) |>
  summary()
```

```
   income_pov
 Min.   :0.000
 1st Qu.:0.870
 Median :1.740
 Mean   :2.242
 3rd Qu.:3.520
 Max.   :5.000
 NA's   :89
```

We see there is some missing data here. Let's ignore that for the moment and concentrate on interpreting the results for the children with actual data. We should start with a picture.

```
ggplot(nnyfs, aes(x = income_pov)) +
  geom_histogram(bins = 30, fill = "white", col = "blue")
```

```
Warning: Removed 89 rows containing non-finite values (`stat_bin()`).
```

race_eth	min	Q1	median	Q3	max	mean	sd	n	missing
1_Hispanic	0	0.6	1.0	1.7	5	1.3	1.1	409	41
2_White Non-Hispanic	0	1.5	3.0	4.5	5	2.9	1.6	588	22
3_Black Non-Hispanic	0	0.8	1.6	2.8	5	2.0	1.5	328	10
4_Other Race/Ethnicity	0	1.2	2.7	4.6	5	2.8	1.7	104	16
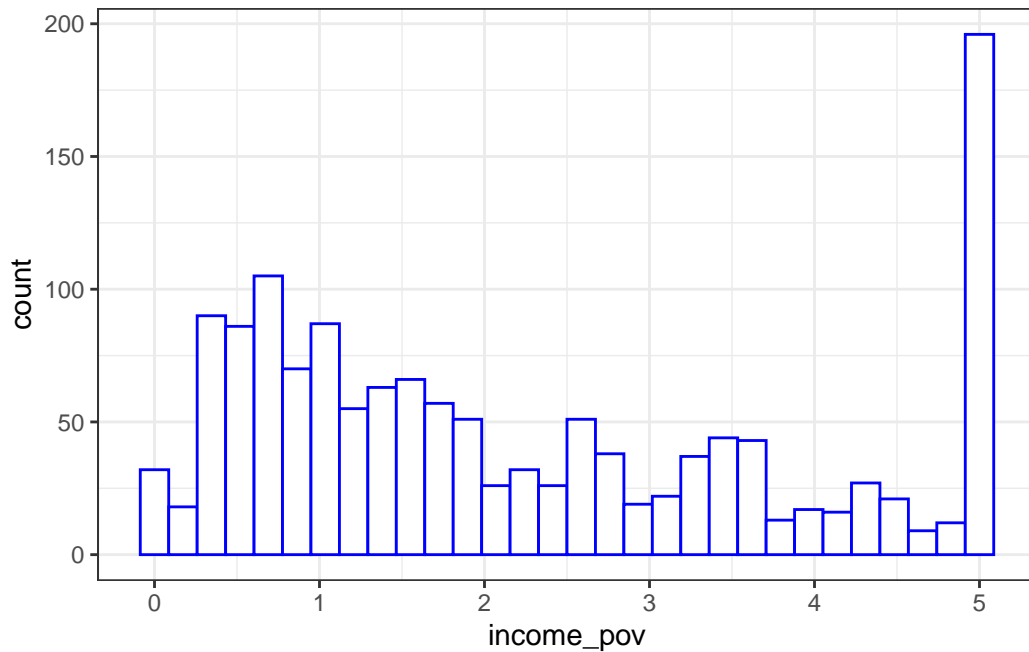
The histogram shows us that the values are truncated at 5, so that children whose actual family income is above 5 times the poverty line are listed as 5. We also see a message reminding us that some of the data are missing for this variable.

Is there a relationship between `income_pov` and `race_eth` in these data?

```
mosaic::favstats(income_pov ~ race_eth, data = nnyfs) |>
  kbl(digits = 1) |>
  kable_styling(full_width = FALSE)
```

This deserves a picture. Let's try a boxplot.

```
ggplot(nnyfs, aes(x = race_eth, y = income_pov)) +
  geom_boxplot()
```

```
Warning: Removed 89 rows containing non-finite values (`stat_boxplot()`).
```
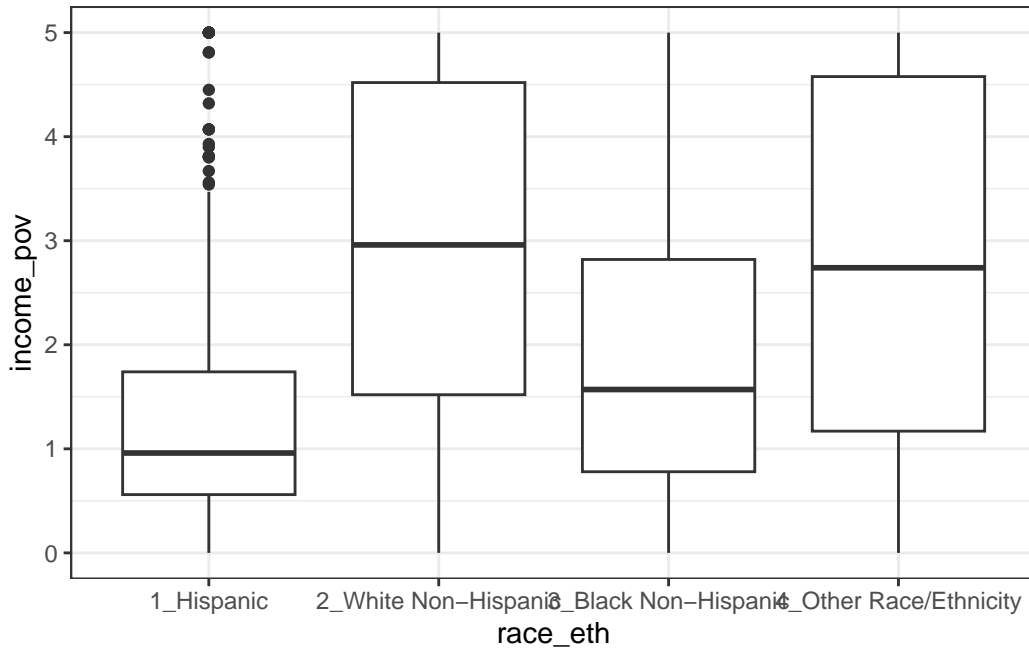
We're reminded here that several of our subjects are not included in this plot, since they have some missing information.

10.4.6 `bmi`

Moving into the body measurement data, `bmi` is the body-mass index of the child. The BMI is a person's weight in kilograms divided by his or her height in meters squared. Symbolically, BMI = weight in kg / (height in m)^2. This is a continuous concept, measured to as many decimal places as you like, and it has a meaningful zero point, so it's a ratio variable.

```
nnyfs |>
  select(bmi) |>
  summary()
```

```
      bmi
 Min.   :11.90
 1st Qu.:15.90
 Median :18.10
 Mean   :19.63
 3rd Qu.:21.90
 Max.   :48.30
```

184

```
NA's   :4
```

Why would a table of these BMI values not be a great idea, for these data? A hint is that R represents this variable as `num` or numeric in its depiction of the data structure, and this implies that R has some decimal values stored. Here, I'll use the `head()` function and the `tail()` function to show the first few and the last few values of what would prove to be a very long table of `bmi` values.

```
nnyfs |>
  tabyl(bmi) |>
  adorn_pct_formatting() |>
  head()
```

```
 bmi n percent valid_percent
11.9 1    0.1%          0.1%
12.6 1    0.1%          0.1%
12.7 1    0.1%          0.1%
12.9 1    0.1%          0.1%
13.0 2    0.1%          0.1%
13.1 1    0.1%          0.1%
```

```
nnyfs |>
  tabyl(bmi) |>
  adorn_pct_formatting() |>
  tail()
```

```
 bmi n percent valid_percent
42.8 1    0.1%          0.1%
43.0 1    0.1%          0.1%
46.9 1    0.1%          0.1%
48.2 1    0.1%          0.1%
48.3 1    0.1%          0.1%
  NA 4    0.3%             -
```

10.4.7 `bmi_cat`

Next I'll look at the `bmi_cat` information. This is a four-category ordinal variable, which divides the sample according to BMI into four groups. The BMI categories use sex-specific 2000 BMI-for-age (in months) growth charts prepared by the Centers for Disease Control for the US. We can get the breakdown from a table of the variable's values.

bmi_cat	min	Q1	median	Q3	max	mean	sd	n	missing
1_Underweight	11.9	13.4	13.7	15.0	16.5	14.1	1.1	41	0
2_Normal	13.5	15.4	16.5	18.7	24.0	17.2	2.3	920	0
3_Overweight	16.9	18.3	21.4	23.4	27.9	21.2	2.9	258	0
4_Obese	17.9	22.3	26.2	30.2	48.3	26.7	5.7	295	0

```
nnyfs |>
  tabyl(bmi_cat) |>
  adorn_pct_formatting()
```

```
      bmi_cat    n percent valid_percent
1_Underweight   41    2.7%          2.7%
    2_Normal  920   60.6%         60.8%
3_Overweight  258   17.0%         17.0%
    4_Obese  295   19.4%         19.5%
        <NA>    4    0.3%             -
```

In terms of percentiles by age and sex from the growth charts, the meanings of the categories are:

- Underweight (BMI < 5th percentile)
- Normal weight (BMI 5th to < 85th percentile)
- Overweight (BMI 85th to < 95th percentile)
- Obese (BMI ≥ 95th percentile)

Note how I've used labels in the `bmi_cat` variable that include a number at the start so that the table results are sorted in a rational way. R sorts tables alphabetically, in general. We'll use the `forcats` package to work with categorical variables that we store as *factors* eventually, but for now, we'll keep things relatively simple.

Note that the `bmi_cat` data don't completely separate out the raw `bmi` data, because the calculation of percentiles requires different tables for each combination of `age` and `sex`.

```
mosaic::favstats(bmi ~ bmi_cat, data = nnyfs) |>
  kbl(digits = 1) |>
  kable_styling(full_width = FALSE)
```

10.4.8 `waist`

Let's also look briefly at `waist`, which is the circumference of the child's waist, in centimeters. Again, this is a numeric variable, so perhaps we'll stick to the simple summary, rather than

obtaining a table of observed values.

```
mosaic::favstats(~ waist, data = nnyfs)
```

```
 min   Q1 median   Q3   max     mean        sd     n missing
42.5 55.6   64.8 76.6 144.7 67.70536 15.19809 1512        6
```

Here's a histogram of the waist circumference data.

```
ggplot(nnyfs, aes(x = waist)) +
  geom_histogram(bins = 25, fill = "tomato", color = "cyan")
```

Warning: Removed 6 rows containing non-finite values (`stat_bin()`).
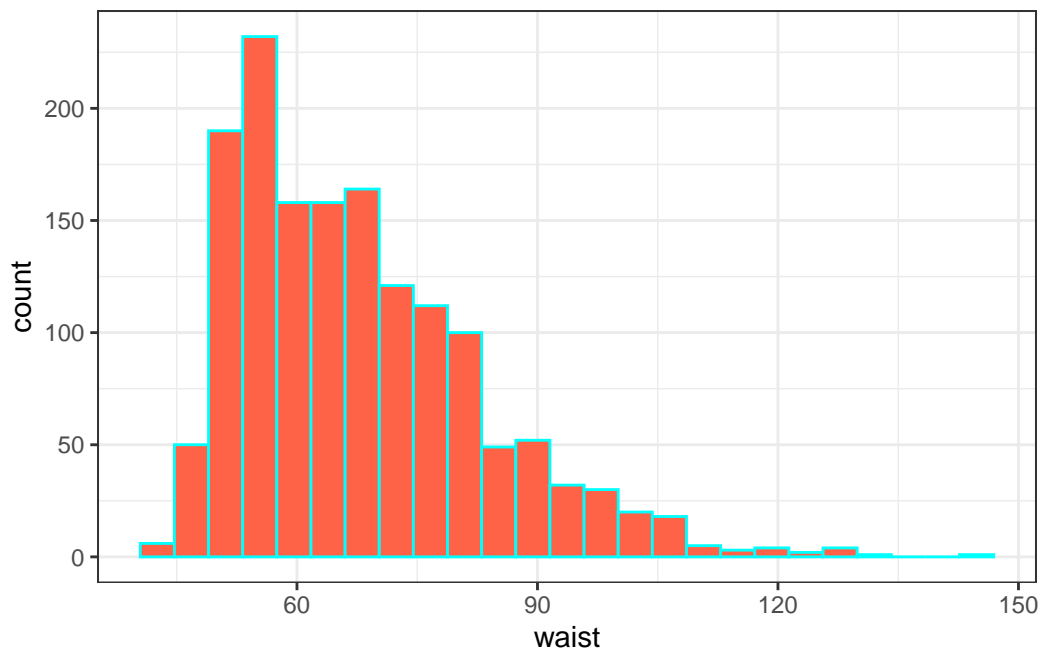
10.4.9 `triceps_skinfold`

The last variable I'll look at for now is `triceps_skinfold`, which is measured in millimeters. This is one of several common locations used for the assessment of body fat using skinfold calipers, and is a frequent part of growth assessments in children. Again, this is a numeric variable according to R.

```
mosaic::favstats(~ triceps_skinfold, data = nnyfs)
```

```
 min  Q1 median Q3  max      mean        sd    n missing
   4 9.1   12.4 18 38.8 14.35725 6.758825 1497      21
```

And here's a histogram of the triceps skinfold data, with the fill and color flipped from what we saw in the plot of the waist circumference data a moment ago.

```
ggplot(nnyfs, aes(x = triceps_skinfold)) +
  geom_histogram(bins = 25, fill = "cyan", color = "tomato")
```

```
Warning: Removed 21 rows containing non-finite values (`stat_bin()`).
```
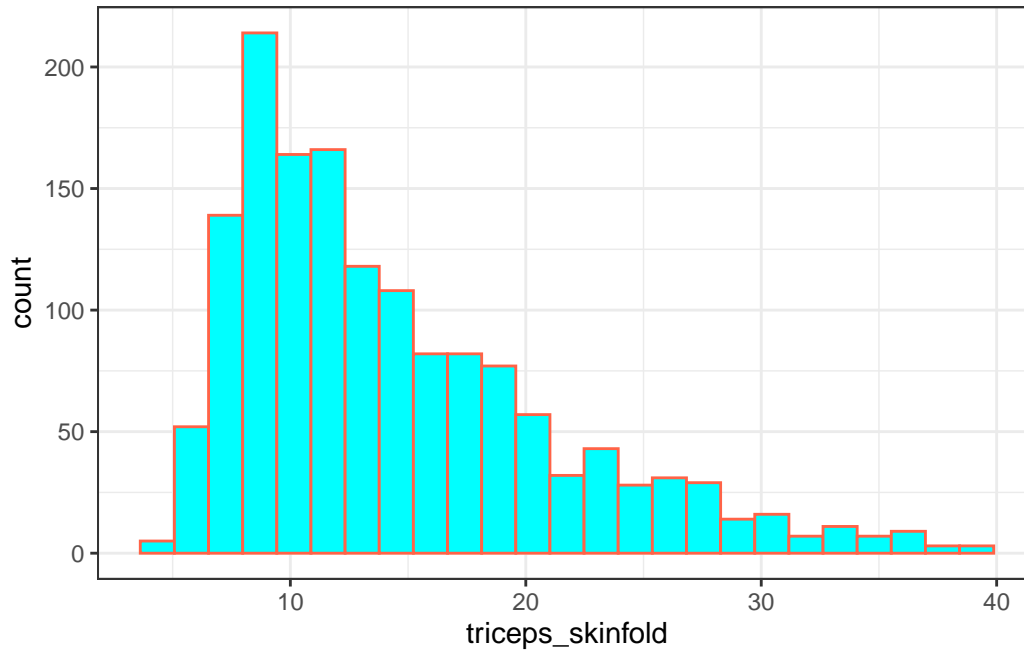

OK. We've seen a few variables, and we'll move on now to look more seriously at the data.

10.5 Additional Numeric Summaries

10.5.1 The Five Number Summary, Quantiles and IQR

The **five number summary** is most famous when used to form a box plot - it's the minimum, 25th percentile, median, 75th percentile and maximum. For numerical and integer variables, the `summary` function produces the five number summary, plus the mean, and a count of any missing values (NA's).

```
nnyfs |>
  select(waist, energy, sugar) |>
  summary()
```

```
    waist             energy          sugar
 Min.   : 42.50   Min.   : 257   Min.   :  1.00
 1st Qu.: 55.60   1st Qu.:1368   1st Qu.: 82.66
 Median : 64.80   Median :1794   Median :116.92
 Mean   : 67.71   Mean   :1877   Mean   :124.32
 3rd Qu.: 76.60   3rd Qu.:2306   3rd Qu.:157.05
 Max.   :144.70   Max.   :5265   Max.   :405.49
 NA's   :6
```

As an alternative, we can use the `$` notation to indicate the variable we wish to study inside a data set, and we can use the `fivenum` function to get the five numbers used in developing a box plot. We'll focus for a little while on the number of kilocalories consumed by each child, according to the dietary recall questionnaire. That's the `energy` variable.

```
fivenum(nnyfs$energy)
```

```
[1]  257.0 1367.0 1794.5 2306.0 5265.0
```

- As mentioned in @ref(rangeandiqr), the **inter-quartile range**, or IQR, is sometimes used as a competitor for the standard deviation. It's the difference between the 75th percentile and the 25th percentile. The 25th percentile, median, and 75th percentile are referred to as the quartiles of the data set, because, together, they split the data into quarters.

```
IQR(nnyfs$energy)
```

```
[1] 938.5
```

We can obtain **quantiles** (percentiles) as we like - here, I'm asking for the 1st and 99th:

```
quantile(nnyfs$energy, probs=c(0.01, 0.99))
```

```
    1%      99%
566.85 4051.75
```

10.6 Additional Summaries from `favstats`

If we're focusing on a single variable, the `favstats` function in the `mosaic` package can be very helpful. Rather than calling up the entire `mosaic` library here, I'll just specify the function within the library.

```
mosaic::favstats(~ energy, data = nnyfs)
```

```
 min     Q1 median   Q3  max     mean       sd    n missing
 257 1367.5 1794.5 2306 5265 1877.157 722.3537 1518       0
```

This adds three useful results to the base summary - the standard deviation, the sample size and the number of missing observations.

10.7 The Histogram

Obtaining a basic **histogram** of, for example, the energy (kilocalories consumed) in the `nnyfs` data is pretty straightforward.

```
ggplot(data = nnyfs, aes(x = energy)) +
    geom_histogram(binwidth = 100, col = "white")
```
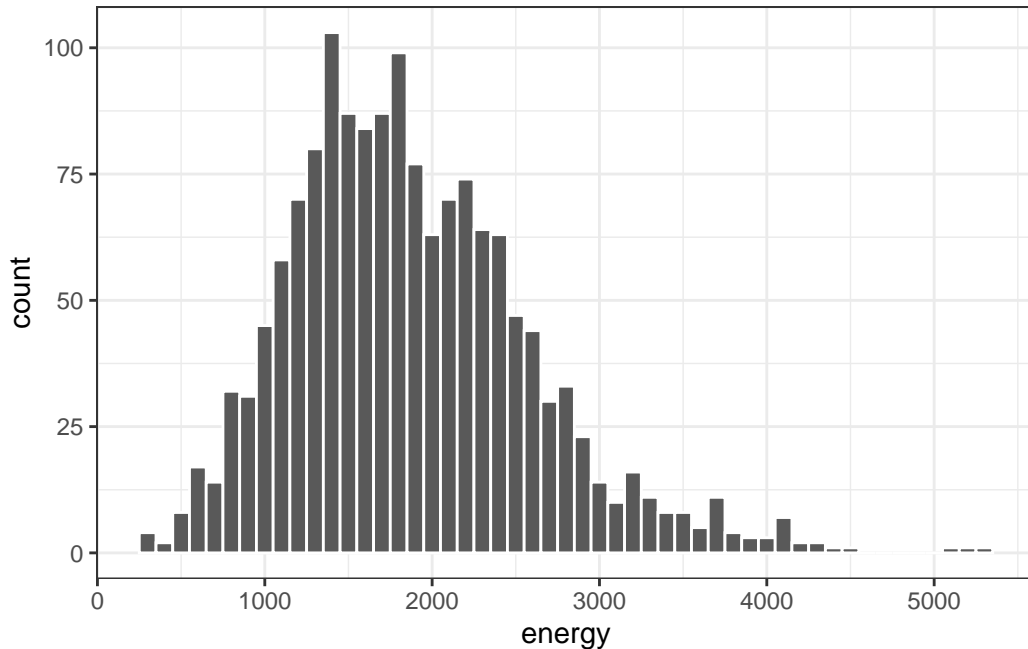
10.7.1 Freedman-Diaconis Rule to select bin width

If we like, we can suggest a particular number of cells for the histogram, instead of accepting the defaults. In this case, we have $n = 1518$ observations. The **Freedman-Diaconis rule** can be helpful here. That rule suggests that we set the bin-width to

$$h = \frac{2 * IQR}{n^{1/3}}$$

so that the number of bins is equal to the range of the data set (maximum - minimum) divided by h.

For the `energy` data in the `nnyfs` tibble, we have

- IQR of 938.5, $n = 1518$ and range = 5008
- Thus, by the Freedman-Diaconis rule, the optimal binwidth h is 163.3203676, or, realistically, 163.
- And so the number of bins would be 30.6636586, or, realistically 31.

Here, we'll draw the graph again, using the Freedman-Diaconis rule to identify the number of bins, and also play around a bit with the fill and color of the bars.

```
bw <- 2 * IQR(nnyfs$energy) / length(nnyfs$energy)^(1/3)
ggplot(data = nnyfs, aes(x = energy)) +
    geom_histogram(binwidth=bw, color = "white", fill = "black")
```
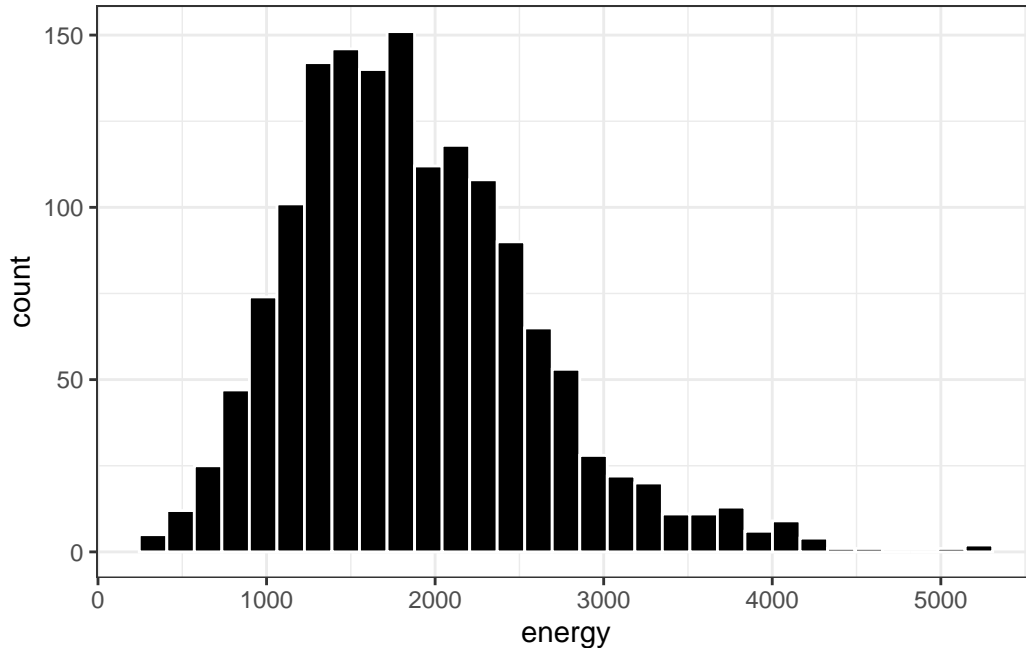

This is a nice start, but it is by no means a finished graph.

Let's improve the axis labels, add a title, and fill in the bars with a distinctive blue and use a black outline around each bar. I'll just use 25 bars, because I like how that looks in this case, and optimizing the number of bins is rarely important.

```
ggplot(data = nnyfs, aes(x = energy)) +
    geom_histogram(bins=25, color = "black", fill = "dodgerblue") +
    labs(title = "Histogram of Body-Mass Index Results in the nnyfs data",
         x = "Energy Consumed (kcal)", y = "# of Subjects")
```

Histogram of Body–Mass Index Results in the nnyfs data
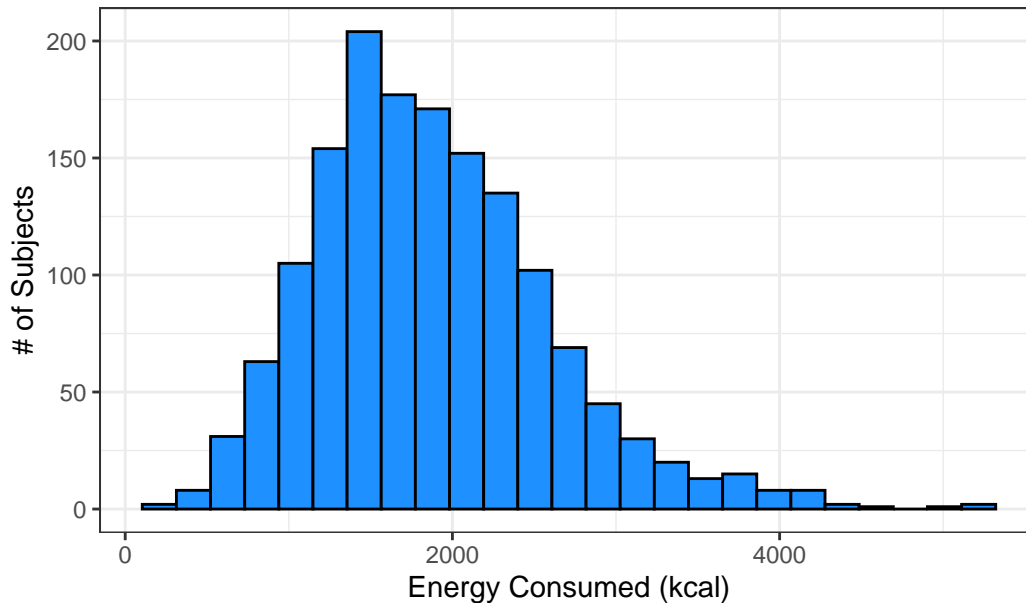
10.7.2 A Note on Colors

The simplest way to specify a color is with its name, enclosed in parentheses. My favorite list of R colors is http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf. In a pinch, you can usually find it by googling **Colors in R**. You can also type `colors()` in the R console to obtain a list of the names of the same 657 colors.

When using colors to make comparisons, you may be interested in using a scale that has some nice properties. The viridis package vignette describes four color scales (viridis, magma, plasma and inferno) that are designed to be colorful, robust to colorblindness and gray scale printing, and perceptually uniform, which means (as the package authors describe it) that values close to each other have similar-appearing colors and values far away from each other have more different-appearing colors, consistently across the range of values. We can apply these colors with special functions within `ggplot`.

Here's a comparison of several histograms, looking at `energy` consumed as a function of whether yesterday was typical in terms of food consumption.

```
ggplot(data = nnyfs, aes(x = energy, fill = diet_yesterday)) +
  geom_histogram(bins = 20, col = "white") +
  scale_fill_viridis_d() +
  facet_wrap(~ diet_yesterday)
```
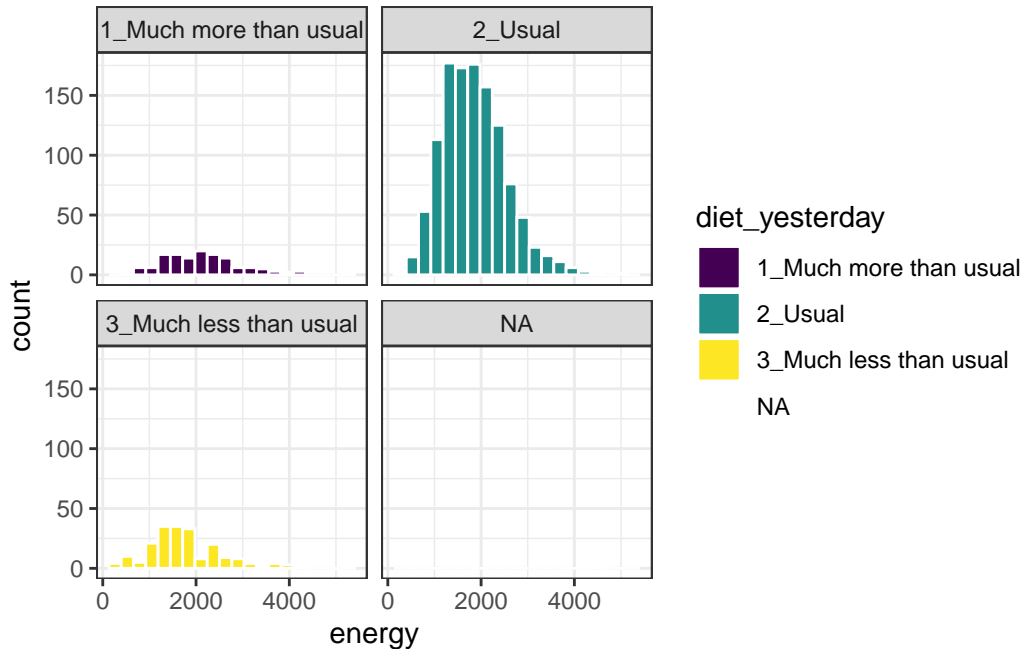
We don't really need the legend here, and perhaps we should restrict the plot to participants who responded to the `diet_yesterday` question, and put in a title and better axis labels?

```
nnyfs_temp <- nnyfs |>
  filter(!is.na(energy), !is.na(diet_yesterday))

ggplot(data = nnyfs_temp, aes(x = energy, fill = diet_yesterday)) +
  geom_histogram(bins = 20, col = "white") +
  scale_fill_viridis_d() +
  guides(fill = "none") +
  facet_wrap(~ diet_yesterday) +
  labs(x = "Energy consumed, in kcal",
       title = "Energy Consumption and How Typical Was Yesterday's Eating",
       subtitle = "NHANES National Youth Fitness Survey, no survey weighting")
```

Energy Consumption and How Typical Was Yesterday's Eating
NHANES National Youth Fitness Survey, no survey weighting
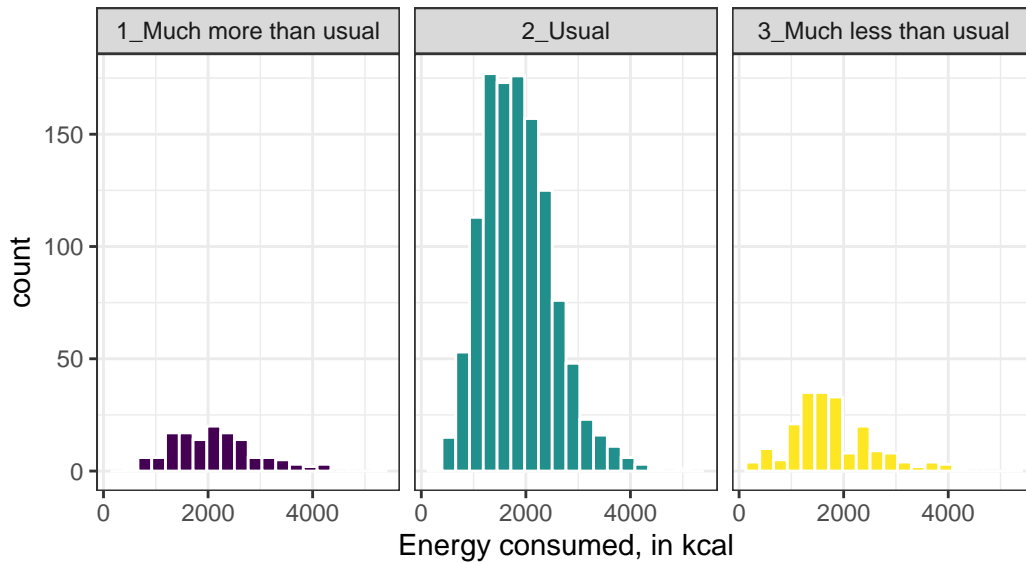
10.8 The Frequency Polygon with Rug Plot

As we've seen, we can also plot the distribution of a single continuous variable using the
`freqpoly` geom. We can also add a *rug plot*, which places a small vertical line on the horizontal
axis everywhere where an observation appears in the data.

```
ggplot(data = nnyfs, aes(x = energy)) +
    geom_freqpoly(binwidth = 150, color = "dodgerblue") +
    geom_rug(color = "red") +
    labs(title = "Frequency Polygon of nnyfs Energy data",
         x = "Energy (kcal)", y = "# of Patients")
```

Frequency Polygon of nnyfs Energy data
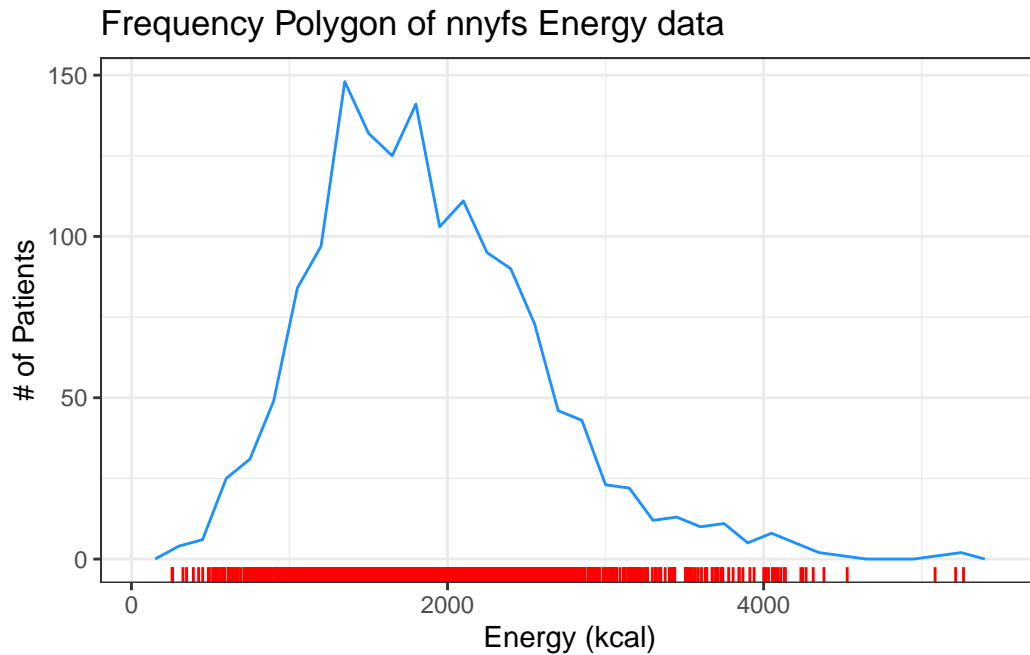
10.9 Plotting the Probability Density Function

We can also produce a density function, which has the effect of smoothing out the bumps in a histogram or frequency polygon, while also changing what is plotted on the y-axis.

```
ggplot(data = nnyfs, aes(x = energy)) +
    geom_density(kernel = "gaussian", color = "dodgerblue") +
    labs(title = "Density of nnyfs Energy data",
         x = "Energy (kcal)", y = "Probability Density function")
```

Density of nnyfs Energy data
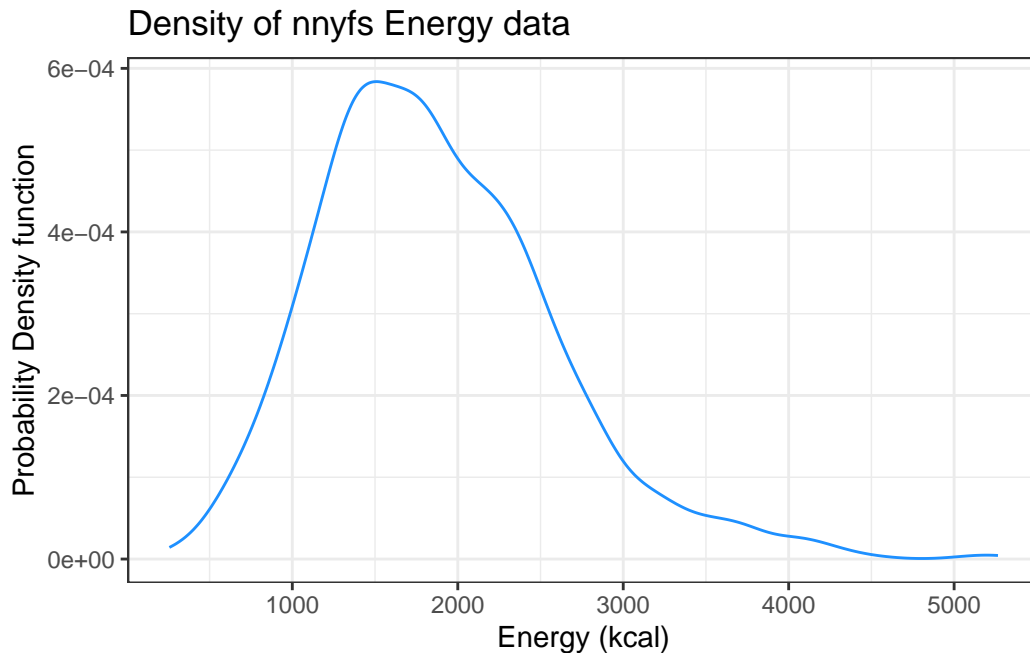
So, what's a density function?

- A probability density function is a function of a continuous variable, x, that represents the probability of x falling within a given range. Specifically, the integral over the interval (a,b) of the density function gives the probability that the value of x is within (a,b).
- If you're interested in exploring more on the notion of density functions for continuous (and discrete) random variables, some nice elementary material is available at Khan Academy.

10.10 The Boxplot

Sometimes, it's helpful to picture the five-number summary of the data in such a way as to get a general sense of the distribution. One approach is a **boxplot**, sometimes called a box-and-whisker plot.

10.10.1 Drawing a Boxplot for One Variable in `ggplot2`

The `ggplot2` library easily handles comparison boxplots for multiple distributions, as we'll see in a moment. However, building a boxplot for a single distribution requires a little trickiness.

```
ggplot(nnyfs, aes(x = 1, y = energy)) +
    geom_boxplot(fill = "deepskyblue") +
    coord_flip() +
    labs(title = "Boxplot of Energy for kids in the NNYFS",
        y = "Energy (kcal)",
        x = "") +
    theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())
```

Boxplot of Energy for kids in the NNYFS
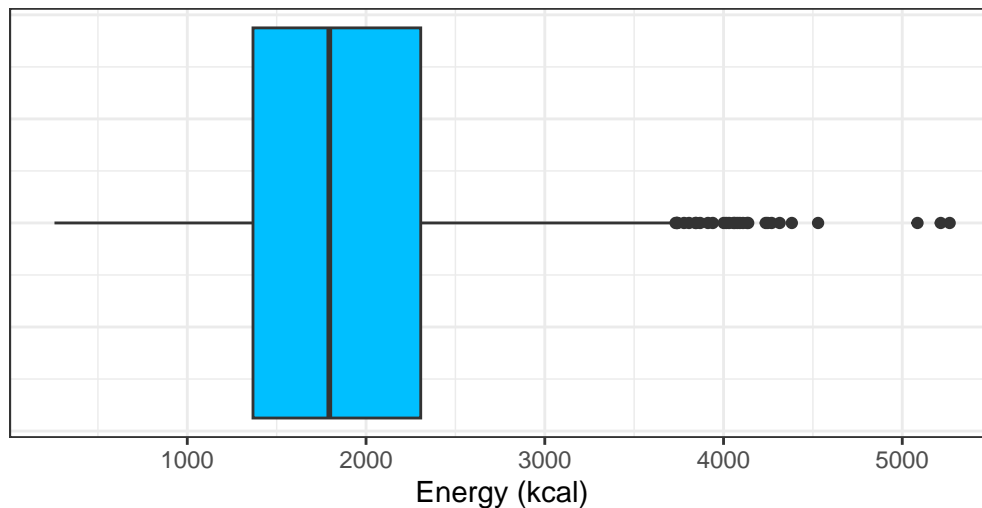
10.10.2 About the Boxplot

The boxplot is another John Tukey invention.

- R draws the box (here in yellow) so that its edges of the box fall at the 25th and 75th percentiles of the data, and the thick line inside the box falls at the median (50th percentile).
- The whiskers then extend out to the largest and smallest values that are not classified by the plot as candidate *outliers*.
- An outlier is an unusual point, far from the center of a distribution.
- Note that I've used the `horizontal` option to show this boxplot in this direction. Most comparison boxplots, as we'll see below, are oriented vertically.

The boxplot's **whiskers** that are drawn from the first and third quartiles (i.e. the 25th and 75th percentiles) out to the most extreme points in the data that do not meet the standard

of "candidate outliers.'' An outlier is simply a point that is far away from the center of the data - which may be due to any number of reasons, and generally indicates a need for further investigation.

Most software, including R, uses a standard proposed by Tukey which describes a "candidate outlier'' as any point above the **upper fence** or below the **lower fence**. The definitions of the fences are based on the inter-quartile range (IQR).

If IQR = 75th percentile - 25th percentile, then the upper fence is 75th percentile + 1.5 IQR, and the lower fence is 25th percentile - 1.5 IQR.

So for these `energy` data,

- the upper fence is located at 2306 + 1.5(938.5) = 3713.75
- the lower fence is located at 1367 - 1.5(938.5) = -40.75

In this case, we see no points identified as outliers in the low part of the distribution, but quite a few identified that way on the high side. This tends to identify about 5% of the data as a candidate outlier, *if* the data follow a Normal distribution.

- This plot is indicating clearly that there is some asymmetry (skew) in the data, specifically right skew.
- The standard R uses is to indicate as outliers any points that are more than 1.5 inter-quartile ranges away from the edges of the box.

The horizontal orientation I've chosen here clarifies the relationship of direction of skew to the plot. A plot like this, with multiple outliers on the right side is indicative of a long right tail in the distribution, and hence, positive or right skew - with the mean being larger than the median. Other indications of skew include having one side of the box being substantially wider than the other, or one side of the whiskers being substantially longer than the other. More on skew later.

10.11 A Simple Comparison Boxplot

Boxplots are most often used for comparison, as we've seen (for example) in Chapter 3 and @#sec-summ_quant. We can build boxplots using `ggplot2`, as well, and we'll discuss that in detail later. For now, here's a boxplot built to compare the `energy` results by the subject's race/ethnicity.

```
ggplot(nnyfs, aes(x = factor(race_eth), y = energy, fill=factor(race_eth))) +
  geom_boxplot() +
  guides(fill = "none") +
  labs(y = "Energy consumed (kcal)", x = "Race/Ethnicity")
```
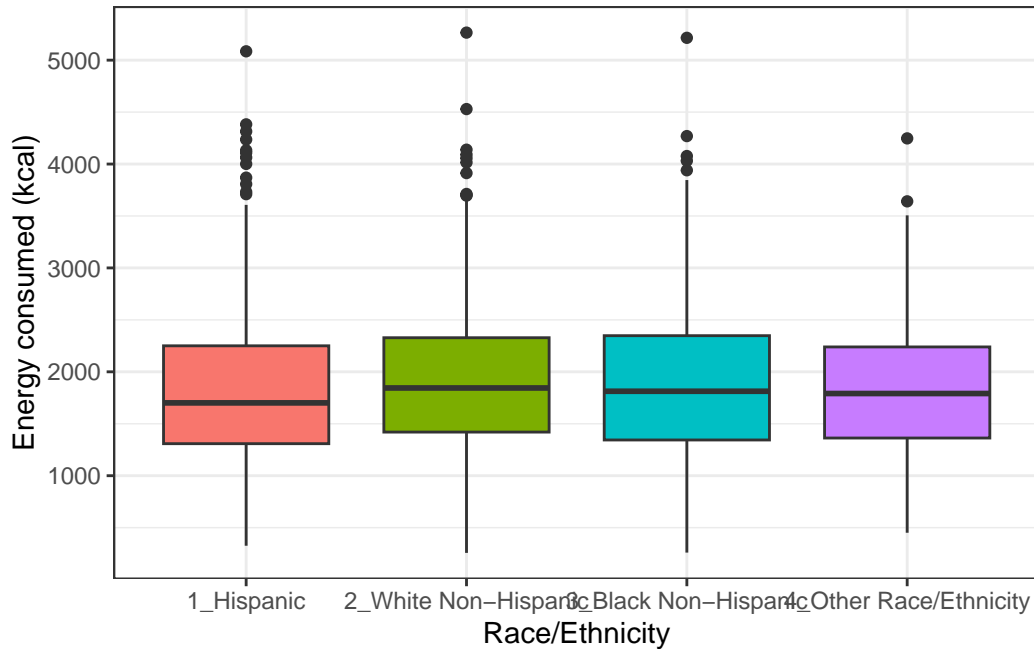
Let's look at the comparison of observed `energy` levels across the five categories in our `phys_health` variable, now making use of the `viridis` color scheme.

```
ggplot(nnyfs, aes(x = factor(phys_health), y = energy, fill = factor(phys_health))) +
  geom_boxplot() +
  scale_fill_viridis_d() +
  labs(title = "Energy by Self-Reported Physical Health, in nnyfs data")
```

Energy by Self–Reported Physical Health, in nnyfs data
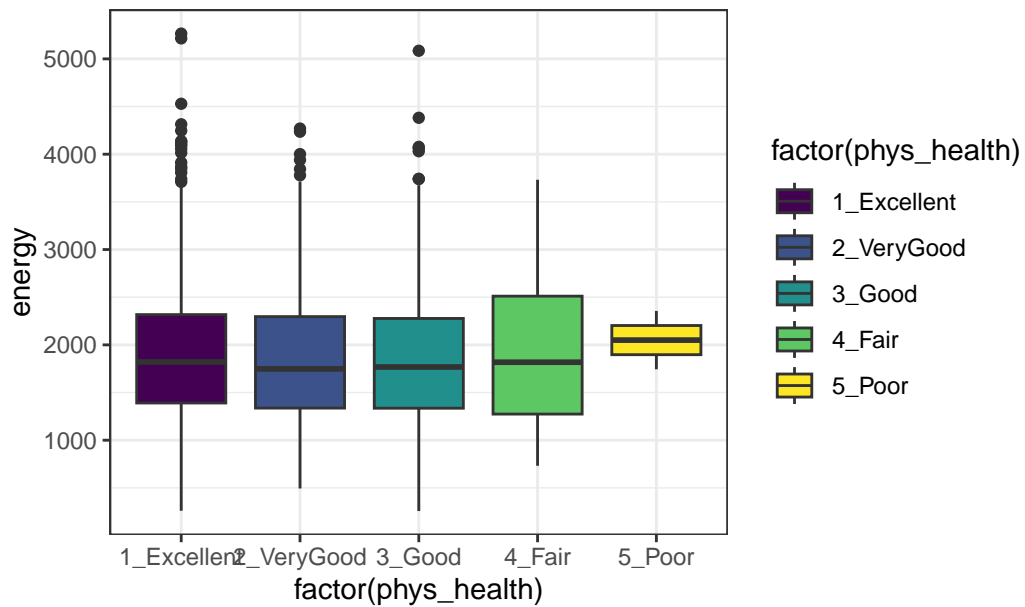
As a graph, that's not bad, but what if we want to improve it further?

Let's turn the boxes in the horizontal direction, and get rid of the perhaps unnecessary `phys_health` labels.

```
ggplot(nnyfs, aes(x = factor(phys_health), y = energy, fill = factor(phys_health))) +
    geom_boxplot() +
    scale_fill_viridis_d() +
    coord_flip() +
    guides(fill = "none") +
    labs(title = "Energy Consumed by Self-Reported Physical Health",
         subtitle = "NHANES National Youth Fitness Survey, unweighted",
         x = "")
```

Energy Consumed by Self−Reported Physical Health
NHANES National Youth Fitness Survey, unweighted
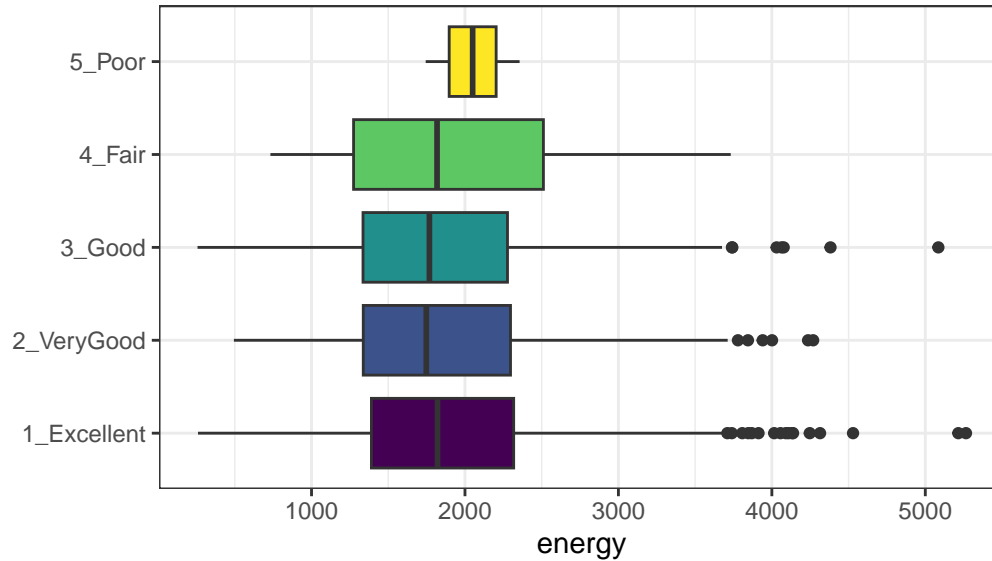
10.12 Using `describe` in the `psych` library

For additional numerical summaries, one option would be to consider using the `describe` function from the `psych` library.

```
psych::describe(nnyfs$energy)
```

```
   vars    n    mean     sd median trimmed    mad min  max range skew kurtosis
X1    1 1518 1877.16 722.35 1794.5  1827.1 678.29 257 5265  5008  0.8     1.13
      se
X1 18.54
```

This package provides, in order, the following…

- `n` = the sample size
- `mean` = the sample mean
- `sd` = the sample standard deviation
- `median` = the median, or 50th percentile
- `trimmed` = mean of the middle 80% of the data
- `mad` = median absolute deviation
- `min` = minimum value in the sample

- `max` = maximum value in the sample
- `range` = max - min
- `skew` = skewness measure, described below (indicates degree of asymmetry)
- `kurtosis` = kurtosis measure, described below (indicates heaviness of tails, degree of outlier-proneness)
- `se` = standard error of the sample mean = sd / square root of sample size, useful in inference

10.12.1 The Trimmed Mean

The **trimmed mean** trim value in R indicates proportion of observations to be trimmed from each end of the outcome distribution before the mean is calculated. The `trimmed` value provided by the `psych::describe` package describes what this particular package calls a 20% trimmed mean (bottom and top 10% of `energy` values are removed before taking the mean - it's the mean of the middle 80% of the data.) I might call that a 80% trimmed mean sometimes, but that's just me.

```
mean(nnyfs$energy, trim=.1)
```

```
[1] 1827.1
```

10.12.2 The Median Absolute Deviation

An alternative to the IQR that is fancier, and a bit more robust, is the **median absolute deviation**, which, in large sample sizes, for data that follow a Normal distribution, will be (in expectation) equal to the standard deviation. The MAD is the median of the absolute deviations from the median, multiplied by a constant (1.4826) to yield asymptotically normal consistency.

```
mad(nnyfs$energy)
```

```
[1] 678.2895
```

10.13 Assessing Skew

A relatively common idea is to assess **skewness**, several measures of which are available. Many models assume a Normal distribution, where, among other things, the data are symmetric around the mean.

Skewness measures asymmetry in the distribution, where left skew (mean < median) is indicated by negative skewness values, while right skew (mean > median) is indicated by positive values. The skew value will be near zero for data that follow a symmetric distribution.

10.13.1 Non-parametric Skewness

A simpler measure of skew, sometimes called the **nonparametric skew** and closely related to Pearson's notion of median skewness, falls between -1 and +1 for any distribution. It is just the difference between the mean and the median, divided by the standard deviation.

- Values greater than +0.2 are sometimes taken to indicate fairly substantial right skew, while values below -0.2 indicate fairly substantial left skew.

```
(mean(nnyfs$energy) - median(nnyfs$energy))/sd(nnyfs$energy)
```

```
[1] 0.114427
```

The Wikipedia page on skewness, from which some of this material is derived, provides definitions for several other skewness measures.

10.14 Assessing Kurtosis (Heavy-Tailedness)

Another measure of a distribution's shape that can be found in the `psych` library is the **kurtosis**. Kurtosis is an indicator of whether the distribution is heavy-tailed or light-tailed as compared to a Normal distribution. Positive kurtosis means more of the variance is due to outliers - unusual points far away from the mean relative to what we might expect from a Normally distributed data set with the same standard deviation.

- A Normal distribution will have a kurtosis value near 0, a distribution with similar tail behavior to what we would expect from a Normal is said to be *mesokurtic*
- Higher kurtosis values (meaningfully higher than 0) indicate that, as compared to a Normal distribution, the observed variance is more the result of extreme outliers (i.e. heavy tails) as opposed to being the result of more modest sized deviations from the mean. These heavy-tailed, or outlier prone, distributions are sometimes called *leptokurtic*.
- Kurtosis values meaningfully lower than 0 indicate light-tailed data, with fewer outliers than we'd expect in a Normal distribution. Such distributions are sometimes referred to as *platykurtic*, and include distributions without outliers, like the Uniform distribution.

Here's a table:

Fewer outliers than a Normal	Approximately Normal	More outliers than a Normal
Light-tailed *platykurtic* (kurtosis < 0)	"Normalish" *mesokurtic* (kurtosis $= 0$)	Heavy-tailed *leptokurtic* (kurtosis > 0)

```
psych::kurtosi(nnyfs$energy)
```

```
[1] 1.130539
```

Note that the `kurtosi()` function is strangely named, and is part of the `psych` package.

10.14.1 The Standard Error of the Sample Mean

The **standard error** of the sample mean, which is the standard deviation divided by the square root of the sample size:

```
sd(nnyfs$energy)/sqrt(length(nnyfs$energy))
```

```
[1] 18.54018
```

10.15 The `describe` function in the `Hmisc` package

The `Hmisc` package has lots of useful functions. It's named for its main developer, Frank Harrell. The `describe` function in `Hmisc` knows enough to separate numerical from categorical variables, and give you separate (and detailed) summaries for each.

- For a categorical variable, it provides counts of total observations (n), the number of missing values, and the number of unique categories, along with counts and percentages falling in each category.
- For a numerical variable, it provides:
- counts of total observations (n), the number of missing values, and the number of unique values
- an Info value for the data, which indicates how continuous the variable is (a score of 1 is generally indicative of a completely continuous variable with no ties, while scores near 0 indicate lots of ties, and very few unique values)
- the sample Mean

- Gini's mean difference, which is a robust measure of spread, with larger values indicating greater dispersion in the data. It is defined as the mean absolute difference between any pairs of observations.
- many sample percentiles (quantiles) of the data, specifically (5, 10, 25, 50, 75, 90, 95, 99)
- either a complete table of all observed values, with counts and percentages (if there are a modest number of unique values), or
- a table of the five smallest and five largest values in the data set, which is useful for range checking

```
nnyfs |>
  select(waist, energy, bmi) |>
  Hmisc::describe()
```

```
select(nnyfs, waist, energy, bmi)

 3  Variables      1518  Observations
--------------------------------------------------------------------------------
waist
      n  missing distinct      Info     Mean      Gmd      .05      .10
   1512        6      510         1    67.71     16.6    49.40    51.40
    .25      .50      .75      .90      .95
  55.60    64.80    76.60    88.70    96.84

lowest : 42.5  43.4  44.1  44.4  44.5 , highest: 125.8 126    127   132.3 144.7
--------------------------------------------------------------------------------
energy
      n  missing distinct      Info     Mean      Gmd      .05      .10
   1518        0     1137         1     1877    796.1      849     1047
    .25      .50      .75      .90      .95
   1368     1794     2306     2795     3195

lowest :   257   260   326   349   392, highest: 4382 4529 5085 5215 5265
--------------------------------------------------------------------------------
bmi
      n  missing distinct      Info     Mean      Gmd      .05      .10
   1514        4      225         1    19.63    5.269    14.30    14.90
    .25      .50      .75      .90      .95
  15.90    18.10    21.90    26.27    30.20

lowest : 11.9 12.6 12.7 12.9 13   , highest: 42.8 43    46.9 48.2 48.3
--------------------------------------------------------------------------------
```

More on the `Info` value in Hmisc::describe is [available here](#)

10.16 Summarizing data within subgroups

Suppose we want to understand how the subjects whose diet involved consuming much more than usual yesterday compare to those who consumer their usual amount, or to those who consumed much less than usual, in terms of the energy they consumed, as well as the protein. We might start by looking at the medians and means.

```
nnyfs |>
    group_by(diet_yesterday) |>
    select(diet_yesterday, energy, protein) |>
    summarise_all(list(median = median, mean = mean))
```

```
# A tibble: 4 x 5
  diet_yesterday        energy_median protein_median energy_mean protein_mean
  <fct>                         <dbl>          <dbl>       <dbl>        <dbl>
1 1_Much more than usual         2098           69.4       2150.         75.1
2 2_Usual                        1794           61.3       1858.         67.0
3 3_Much less than usual         1643           53.9       1779.         60.1
4 <NA>                           4348          155.        4348         155.
```

Perhaps we should restrict ourselves to the people who were not missing the `diet_yesterday` category, and look now at their `sugar` and `water` consumption.

```
nnyfs |>
    filter(complete.cases(diet_yesterday)) |>
    group_by(diet_yesterday) |>
    select(diet_yesterday, energy, protein, sugar, water) |>
    summarise_all(list(median))
```

```
# A tibble: 3 x 5
  diet_yesterday        energy protein sugar water
  <fct>                  <dbl>   <dbl> <dbl> <dbl>
1 1_Much more than usual   2098    69.4  137.  500
2 2_Usual                  1794    61.3  114.  385.
3 3_Much less than usual   1643    53.9  115.  311.
```

It looks like the children in the "Much more than usual" category consumed more energy, protein, sugar and water than the children in the other two categories. Let's draw a picture of this.

```
temp_dat <- nnyfs |>
    filter(complete.cases(diet_yesterday)) |>
    mutate(diet_yesterday = fct_recode(diet_yesterday,
        "Much more" = "1_Much more than usual",
        "Usual diet" = "2_Usual",
        "Much less" = "3_Much less than usual"))

p1 <- ggplot(temp_dat, aes(x = diet_yesterday, y = energy)) +
    geom_violin() +
    geom_boxplot(aes(fill = diet_yesterday), width = 0.2) +
    theme_light() +
    scale_fill_viridis_d() +
    guides(fill = "none") +
    labs(title = "Energy Comparison")

p2 <- ggplot(temp_dat, aes(x = diet_yesterday, y = protein)) +
    geom_violin() +
    geom_boxplot(aes(fill = diet_yesterday), width = 0.2) +
    theme_light() +
    scale_fill_viridis_d() +
    guides(fill = "none") +
    labs(title = "Protein Comparison")

p3 <- ggplot(temp_dat, aes(x = diet_yesterday, y = sugar)) +
    geom_violin() +
    geom_boxplot(aes(fill = diet_yesterday), width = 0.2) +
    theme_light() +
    scale_fill_viridis_d() +
    guides(fill = "none") +
    labs(title = "Sugar Comparison")

p4 <- ggplot(temp_dat, aes(x = diet_yesterday, y = water)) +
    geom_violin() +
    geom_boxplot(aes(fill = diet_yesterday), width = 0.2) +
    theme_light() +
    scale_fill_viridis_d() +
    guides(fill = "none") +
    labs(title = "Water Comparison")
```

```
p1 + p2 + p3 + p4
```
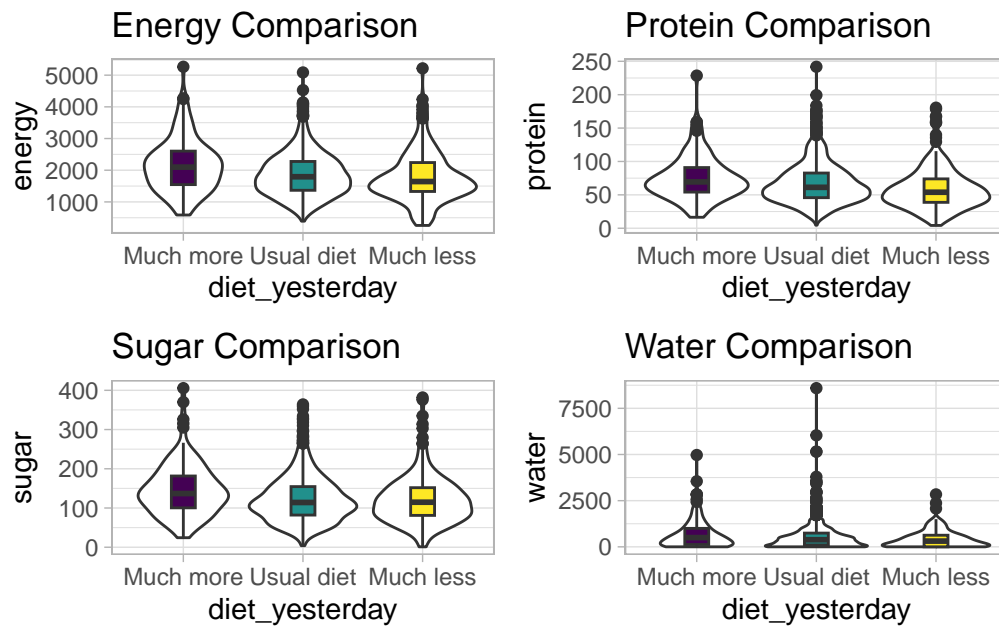

We can see that there is considerable overlap in these distributions, regardless of what we're measuring.

10.17 Another Example

Suppose now that we ask a different question. Do kids in larger categories of BMI have larger waist circumferences?

```
nnyfs |>
    group_by(bmi_cat) |>
    summarise(mean = mean(waist), sd = sd(waist),
              median = median(waist),
              skew_1 = round((mean(waist) - median(waist)) /
                                 sd(waist),2))
```

```
# A tibble: 5 x 5
  bmi_cat        mean      sd median skew_1
```

bmi_cat	min	Q1	median	Q3	max	mean	sd	n	missing
1_Underweight	42.5	49.3	54.5	62.4	68.5	55.2	7.6	41	0
2_Normal	44.1	53.9	59.5	68.4	89.2	61.2	9.4	917	3
3_Overweight	49.3	62.3	74.0	81.2	105.3	72.3	11.9	258	0
4_Obese	52.1	72.7	86.8	96.8	144.7	85.6	17.1	294	1

```
  <fct>          <dbl> <dbl>  <dbl>  <dbl>
1 1_Underweight  55.2  7.58    54.5  0.09
2 2_Normal        NA    NA      NA    NA
3 3_Overweight   72.3 11.9     74   -0.14
4 4_Obese         NA    NA      NA    NA
5 <NA>            NA    NA      NA    NA
```

Oops. Looks like we need to filter for cases with complete data on both BMI category and waist circumference in order to get meaningful results. We should add a count, too.

```
nnyfs |>
    filter(complete.cases(bmi_cat, waist)) |>
    group_by(bmi_cat) |>
    summarise(count = n(), mean = mean(waist),
              sd = sd(waist), median = median(waist),
        skew_1 =
          round((mean(waist) - median(waist)) / sd(waist),2))
```

```
# A tibble: 4 x 6
  bmi_cat       count  mean    sd median skew_1
  <fct>         <int> <dbl> <dbl>  <dbl>  <dbl>
1 1_Underweight    41  55.2  7.58   54.5   0.09
2 2_Normal        917  61.2  9.35   59.5   0.19
3 3_Overweight    258  72.3 11.9    74    -0.14
4 4_Obese         294  85.6 17.1    86.8  -0.07
```

Or, we could use something like `favstats` from the `mosaic` package, which automatically accounts for missing data, and omits it when calculating summary statistics within each group.

```
mosaic::favstats(waist ~ bmi_cat, data = nnyfs) |>
    kbl(digits = 1) |>
    kable_styling(full_width = FALSE)
```

While patients in the heavier groups generally had higher waist circumferences, the standard deviations suggest there may be some meaningful overlap. Let's draw the picture, in this case a comparison boxplot accompanying a violin plot.

```
nnyfs_temp2 <- nnyfs |>
    filter(complete.cases(bmi_cat, waist))

ggplot(nnyfs_temp2, aes(x = bmi_cat, y = waist)) +
    geom_violin() +
    geom_boxplot(aes(fill = bmi_cat), width = 0.2) +
    theme_light() +
    scale_fill_viridis_d() +
    guides(fill = "none") +
    labs(title = "Waist Circumference by BMI Category")
```
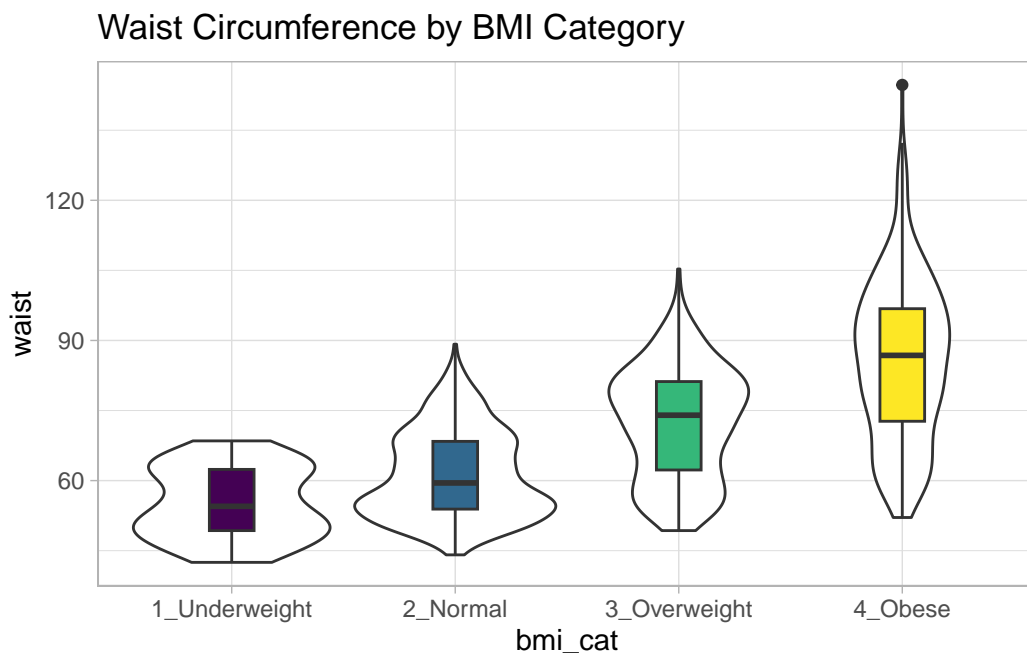

The data transformation with dplyr cheat sheet found under the Help menu in RStudio is a great resource. And, of course, for more details, visit Hadley Wickham and Grolemund (2023).

10.18 Boxplots to Relate an Outcome to a Categorical Predictor

Boxplots are much more useful when comparing samples of data. For instance, consider this comparison boxplot describing the triceps skinfold results across the four levels of BMI category.

```
ggplot(nnyfs, aes(x = bmi_cat, y = triceps_skinfold,
                  fill = bmi_cat)) +
    geom_boxplot() +
    scale_fill_viridis_d() +
    theme_light()
```

Warning: Removed 21 rows containing non-finite values (`stat_boxplot()`).
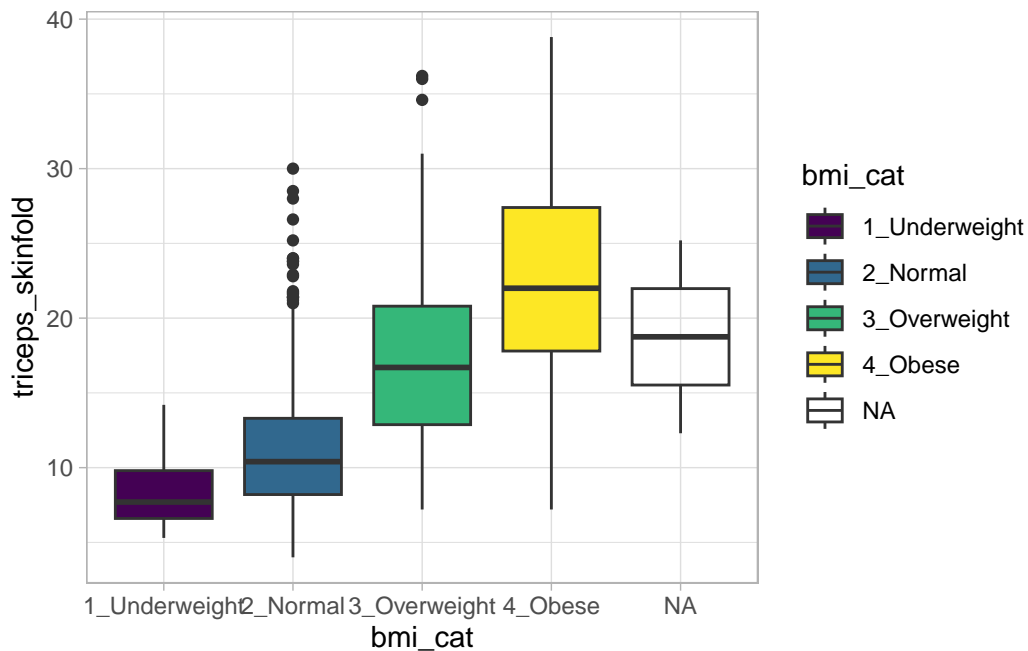
Again, we probably want to omit those missing values (both in `bmi_cat` and `triceps_skinfold`) and also eliminate the repetitive legend (guides) on the right.

```
nnyfs_temp3 <- nnyfs |>
    filter(complete.cases(bmi_cat, triceps_skinfold))

ggplot(nnyfs_temp3,
```

```
       aes(x = bmi_cat, y = triceps_skinfold, fill = bmi_cat)) +
  geom_boxplot() +
  scale_fill_viridis_d() +
  guides(fill = "none") +
  theme_light() +
  labs(x = "BMI Category", y = "Triceps Skinfold in mm",
       title = "Triceps Skinfold increases with BMI category",
       subtitle = "NNYFS children")
```


Triceps Skinfold increases with BMI category
NNYFS children
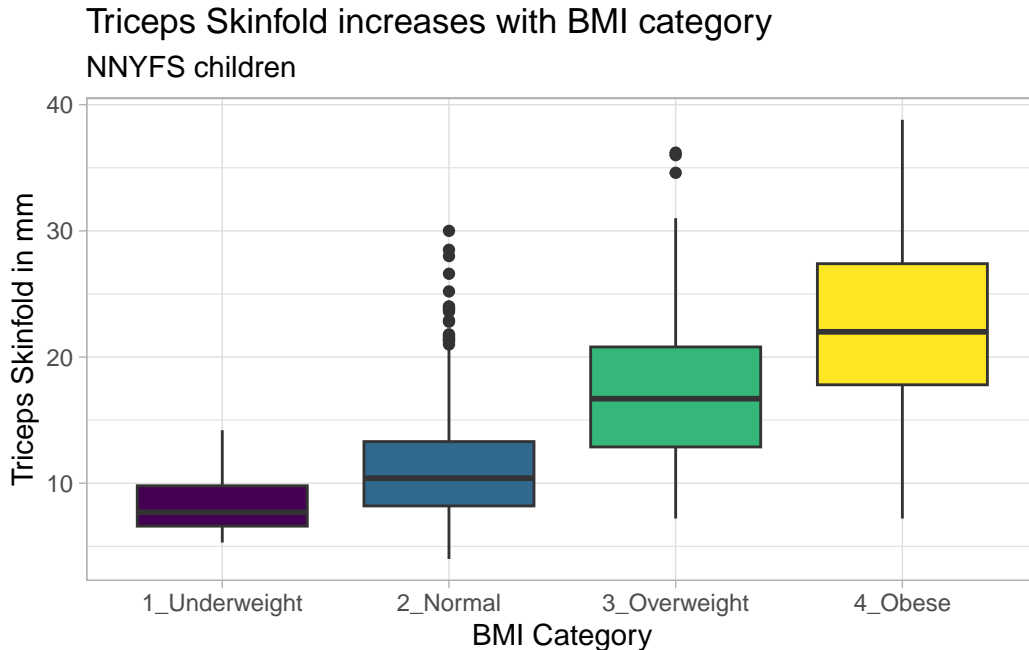
As always, the boxplot shows the five-number summary (minimum, 25th percentile, median, 75th percentile and maximum) in addition to highlighting candidate outliers.

10.18.1 Augmenting the Boxplot with the Sample Mean

Often, we want to augment such a plot, perhaps by adding a little diamond to show the **sample mean** within each category, so as to highlight skew (in terms of whether the mean is meaningfully different from the median.)

```
nnyfs_temp3 <- nnyfs |>
  filter(complete.cases(bmi_cat, triceps_skinfold))
```

```
ggplot(nnyfs_temp3,
       aes(x = bmi_cat, y = triceps_skinfold, fill = bmi_cat)) +
    geom_boxplot() +
    stat_summary(fun="mean", geom="point",
                 shape=23, size=3, fill="white") +
    scale_fill_viridis_d() +
    guides(fill = "none") +
    theme_light() +
    labs(x = "BMI Category", y = "Triceps Skinfold in mm",
         title = "Triceps Skinfold increases with BMI category",
         subtitle = "NNYFS children")
```


Triceps Skinfold increases with BMI category
NNYFS children
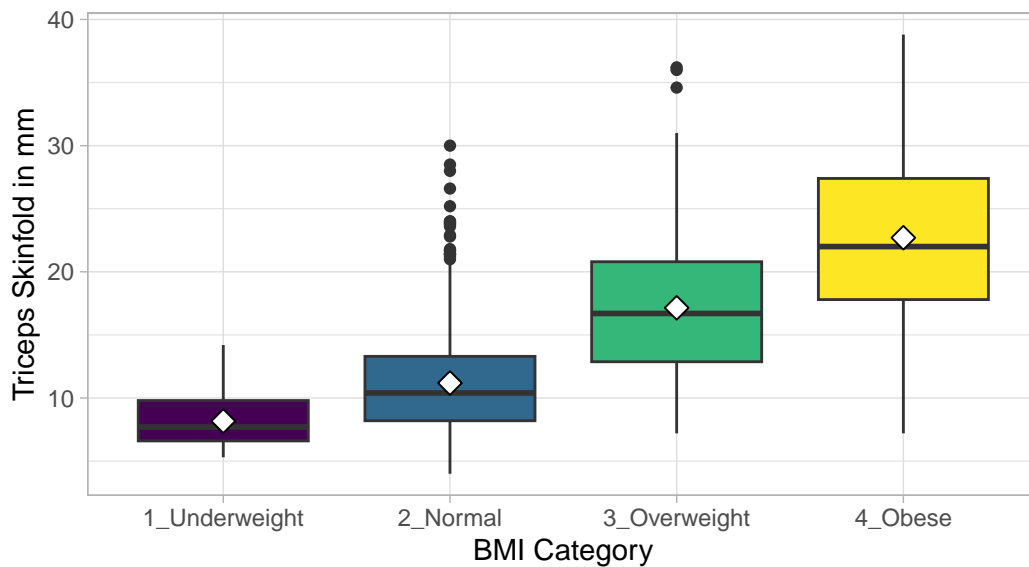
10.19 Building a Violin Plot

There are a number of other plots which compare distributions of data sets. An interesting one is called a **violin plot**. A violin plot is a kernel density estimate, mirrored to form a symmetrical shape.

```
nnyfs_temp3 <- nnyfs |>
    filter(complete.cases(bmi_cat, triceps_skinfold))
```

```
ggplot(nnyfs_temp3, aes(x=bmi_cat, y=triceps_skinfold, fill = bmi_cat)) +
    geom_violin(trim=FALSE) +
    scale_fill_viridis_d() +
    guides(fill = "none") +
    labs(title = "Triceps Skinfold by BMI Category")
```


Triceps Skinfold by BMI Category
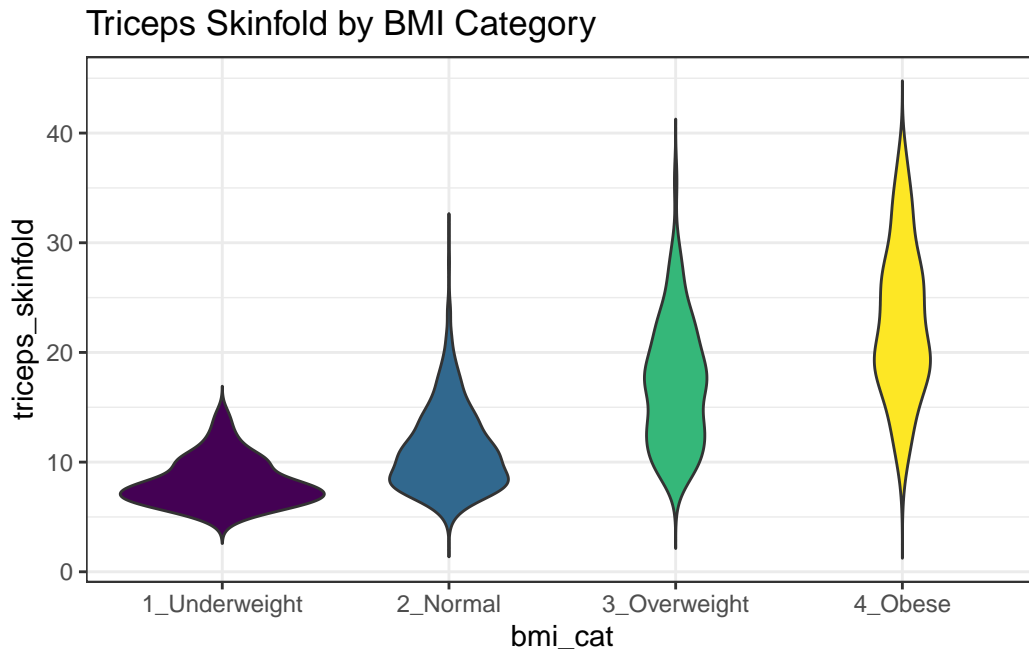
Traditionally, these plots are shown with overlaid boxplots and a white dot at the median, like this example, now looking at waist circumference again.

```
nnyfs_temp2 <- nnyfs |>
    filter(complete.cases(bmi_cat, waist))

ggplot(nnyfs_temp2, aes(x = bmi_cat, y = waist, fill = bmi_cat)) +
    geom_violin(trim=FALSE) +
    geom_boxplot(width=.1, outlier.colour=NA,
                 color = c(rep("white",2), rep("black",2))) +
    stat_summary(fun=median, geom="point",
                 fill="white", shape=21, size=3) +
    scale_fill_viridis_d() +
    guides(fill = "none") +
```

```
labs(title = "Waist Circumference by BMI Category")
```

Waist Circumference by BMI Category
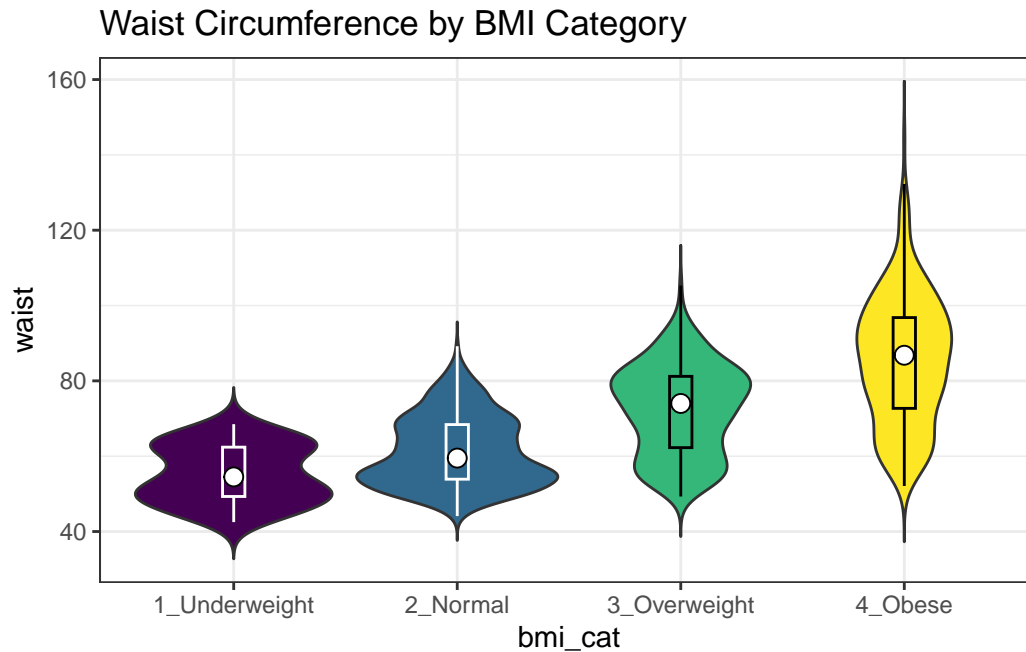
10.19.1 Adding Notches to a Boxplot

Notches are used in boxplots to help visually assess whether the medians of the distributions across the various groups actually differ to a statistically detectable extent. Think of them as confidence regions around the medians. If the notches do not overlap, as in this situation, this provides some evidence that the medians in the populations represented by these samples may be different.

```
nnyfs_temp3 <- nnyfs |>
    filter(complete.cases(bmi_cat, triceps_skinfold))

ggplot(nnyfs_temp3, aes(x = bmi_cat, y = triceps_skinfold)) +
    geom_violin() +
    geom_boxplot(aes(fill = bmi_cat), width = 0.3, notch = TRUE) +
    scale_fill_viridis_d() +
    guides(fill = "none") +
    theme_light() +
    labs(x = "BMI Category", y = "Triceps Skinfold in mm",
```

```
        title = "Triceps Skinfold increases with BMI category",
        subtitle = "NNYFS children")
```

Triceps Skinfold increases with BMI category
NNYFS children
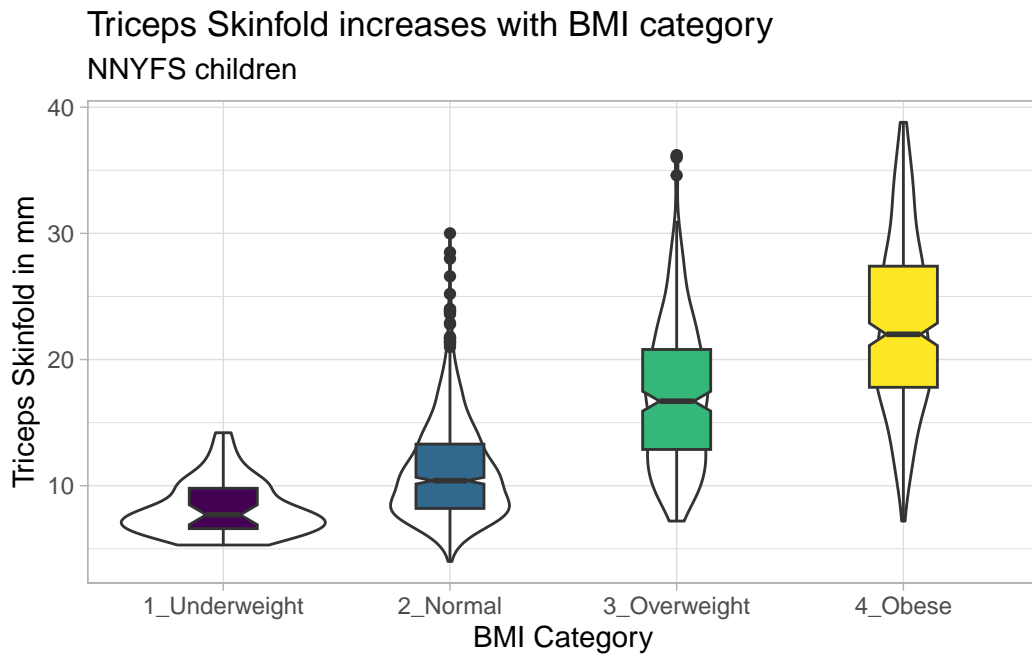
There is no overlap between the notches for each of the four categories, so we might reasonably conclude that the true median triceps skinfold values across the four categories are statistically significantly different.

For an example where the notches do overlap, consider the comparison of plank times by BMI category.

```
nnyfs_temp4 <- nnyfs |>
  filter(complete.cases(bmi_cat, plank_time))

ggplot(nnyfs_temp4, aes(x=bmi_cat, y=plank_time)) +
    geom_violin(aes(fill = bmi_cat)) +
    geom_boxplot(width = 0.3, notch=TRUE) +
    scale_fill_viridis_d() +
    guides(fill = "none") +
    theme_light() +
    labs(title = "Plank Times by BMI category",
        x = "", y = "Plank Time (in seconds)")
```

Plank Times by BMI category
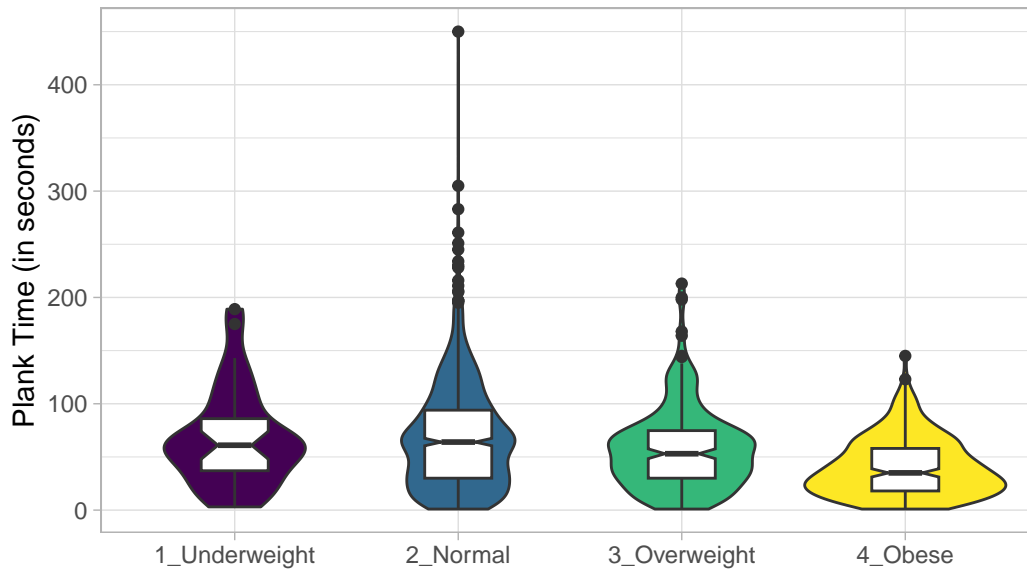
The overlap in the notches (for instance between Underweight and Normal) suggests that the median plank times in the population of interest don't necessarily differ in a meaningful way by BMI category, other than perhaps the Obese group which may have a shorter time.

These data are somewhat right skewed. Would a logarithmic transformation in the plot help us see the patterns more clearly?

```
nnyfs_temp4 <- nnyfs |>
  filter(complete.cases(bmi_cat, plank_time))

ggplot(nnyfs_temp4, aes(x=bmi_cat, y = log(plank_time))) +
    geom_violin() +
    geom_boxplot(aes(fill = bmi_cat), width = 0.3, notch=TRUE) +
    scale_fill_viridis_d() +
    guides(fill = "none") +
    theme_light() +
    labs(title = "log(Plank Times) by BMI category",
         x = "", y = "Natural Log of Plank Time")
```

log(Plank Times) by BMI category
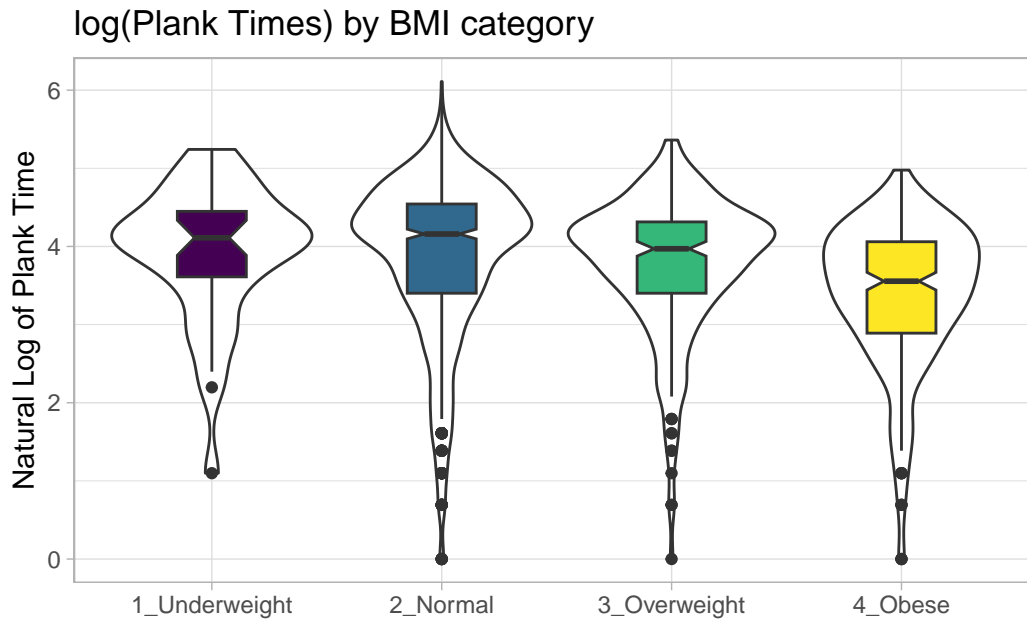
10.20 Using Multiple Histograms to Make Comparisons

We can make an array of histograms to describe multiple groups of data, using `ggplot2` and the notion of **faceting** our plot.

```
nnyfs_temp3 <- nnyfs |>
    filter(complete.cases(bmi_cat, triceps_skinfold))

ggplot(nnyfs_temp3, aes(x=triceps_skinfold, fill = bmi_cat)) +
    geom_histogram(binwidth = 2, color = "black") +
    facet_wrap(~ bmi_cat) +
    scale_fill_viridis_d() +
    guides(fill = "none") +
    labs(title = "Triceps Skinfold by BMI Category")
```

Triceps Skinfold by BMI Category
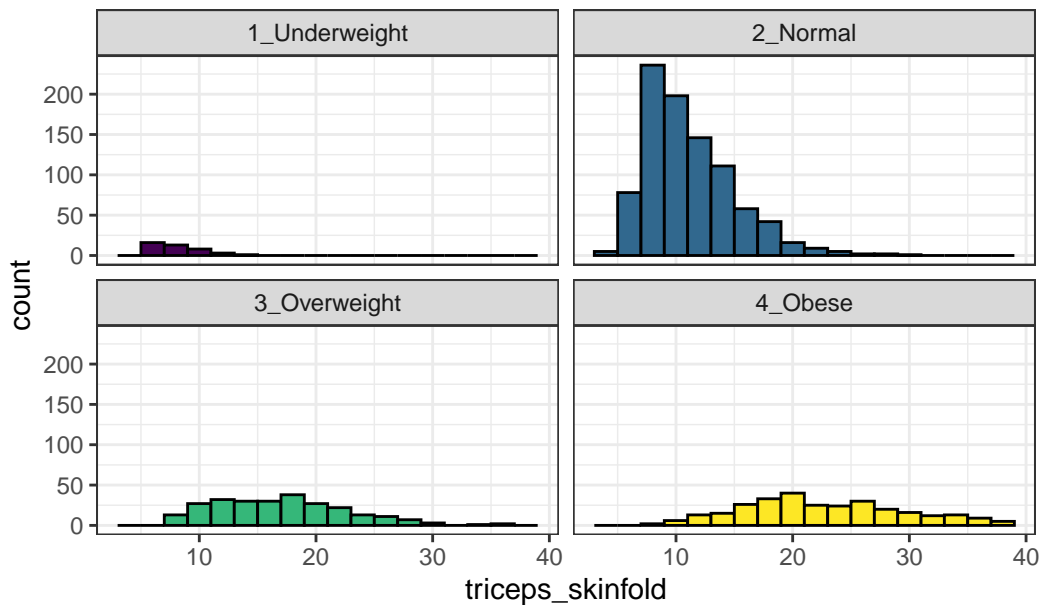
10.21 Using Multiple Density Plots to Make Comparisons

Or, we can make a series of density plots to describe multiple groups of data.

```
nnyfs_temp3 <- nnyfs |>
  filter(complete.cases(bmi_cat, triceps_skinfold))

ggplot(nnyfs_temp3, aes(x=triceps_skinfold, fill = bmi_cat)) +
    geom_density(color = "black") +
    facet_wrap(~ bmi_cat) +
    scale_fill_viridis_d() +
    guides(fill = "none") +
    labs(title = "Triceps Skinfold by BMI Category")
```

Triceps Skinfold by BMI Category
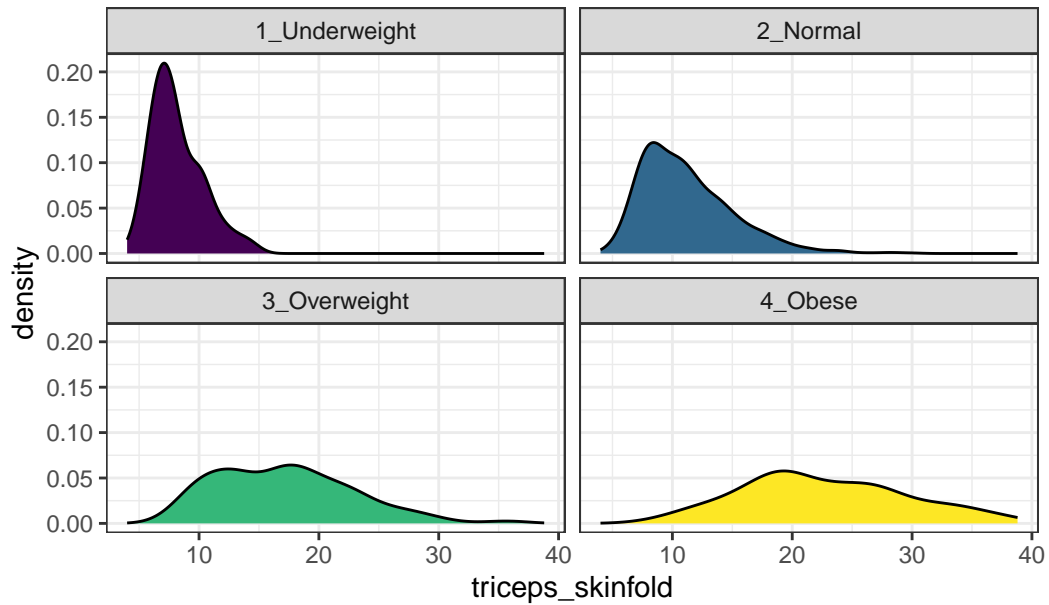
Or, we can plot all of the densities on top of each other with semi-transparent fills.

```
nnyfs_temp3 <- nnyfs |>
  filter(complete.cases(bmi_cat, triceps_skinfold))

ggplot(nnyfs_temp3, aes(x=triceps_skinfold, fill = bmi_cat)) +
    geom_density(alpha=0.3) +
    scale_fill_viridis_d() +
    labs(title = "Triceps Skinfold by BMI Category")
```

Triceps Skinfold by BMI Category
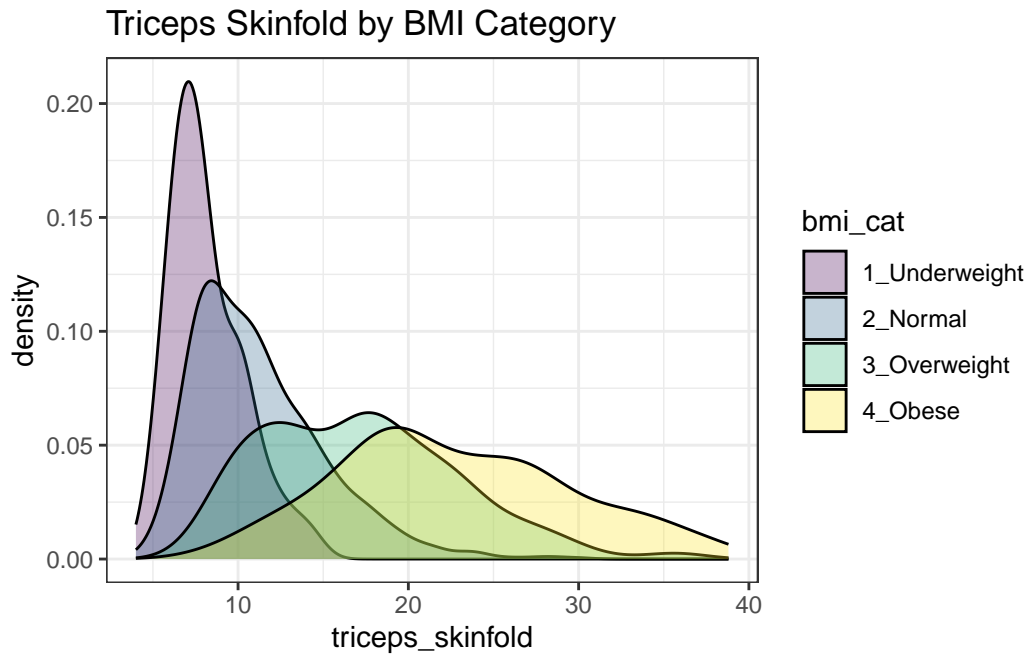
This really works better when we are comparing only two groups, like females to males.

```r
nnyfs_temp5 <- nnyfs |>
  filter(complete.cases(sex, triceps_skinfold))

ggplot(nnyfs_temp5, aes(x=triceps_skinfold, fill = sex)) +
    geom_density(alpha=0.5) +
    labs(title = "Triceps Skinfold by Sex")
```

Triceps Skinfold by Sex
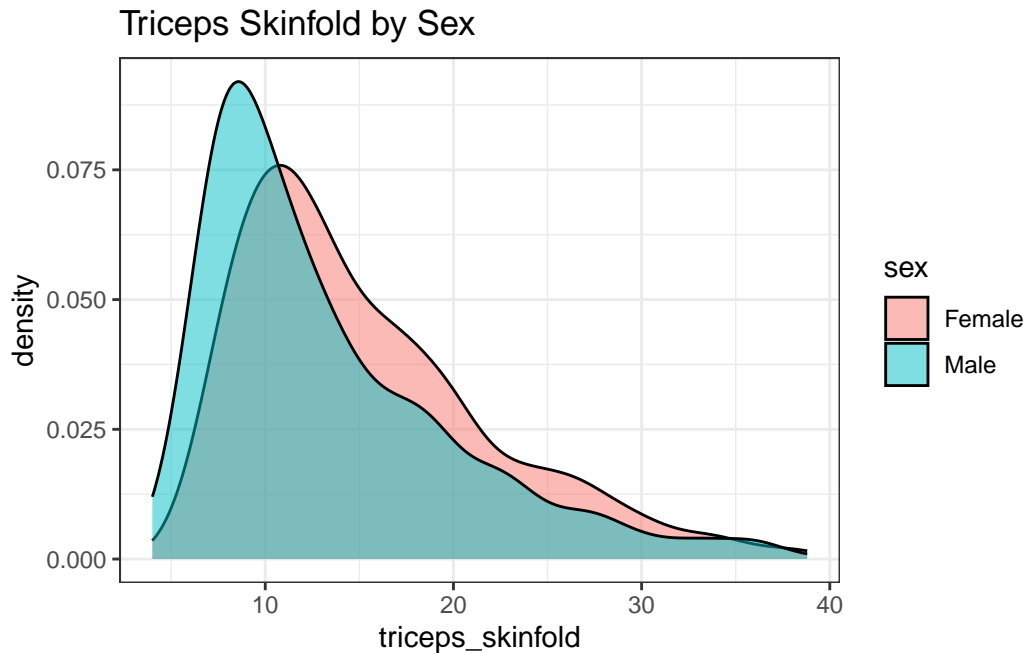
10.22 A Ridgeline Plot

Some people don't like violin plots - for example, see https://simplystatistics.org/2017/07/13/the-joy-of-no-more-violin-plots/. An alternative plot is available as part of the **ggridges** package. This shows the distribution of several groups simultaneously, especially when you have lots of subgroup categories, and is called a **ridgeline plot**.

```
nnyfs_temp6 <- nnyfs |>
  filter(complete.cases(waist, bmi_cat))

ggplot(nnyfs_temp6, aes(x = waist, y = bmi_cat, height = ..density..)) +
    geom_density_ridges(scale = 0.85) +
    theme_light() +
    labs(title = "Ridgeline Plot: Waist Circumference by BMI category (nnyfs)",
         x = "Waist Circumference", y = "BMI Category")
```

Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
i Please use `after_stat(density)` instead.

Picking joint bandwidth of 3.47

Ridgeline Plot: Waist Circumference by BMI category (
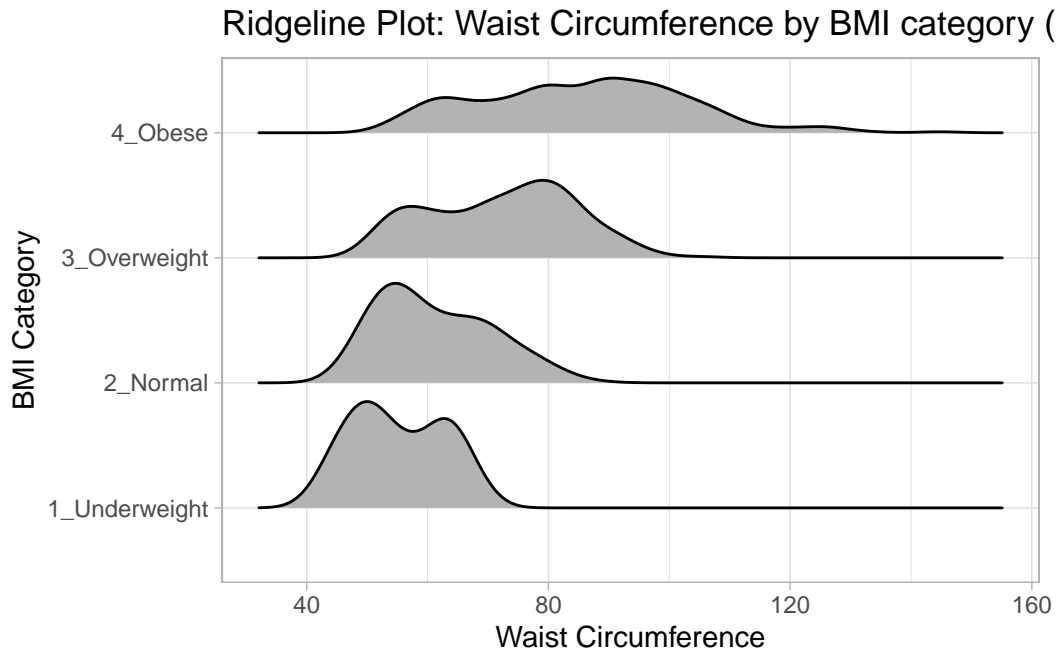
And here's a ridgeline plot for the triceps skinfolds. We'll start by sorting the subgroups by the median value of our outcome (triceps skinfold) in this case, though it turns out not to matter. We'll also add some color.

```
nnyfs_temp3 <- nnyfs |>
  filter(complete.cases(bmi_cat, triceps_skinfold)) |>
  mutate(bmi_cat = fct_reorder(bmi_cat,
                               triceps_skinfold,
                               .fun = median))

ggplot(nnyfs_temp3, aes(x = triceps_skinfold, y = bmi_cat,
                  fill = bmi_cat, height = ..density..)) +
    ggridges::geom_density_ridges(scale = 0.85) +
    scale_fill_viridis_d(option = "magma") +
    guides(fill = "none") +
    labs(title = "Ridgeline Plot of Triceps Skinfold by BMI Category (nnyfs)",
         x = "Triceps Skinfold", y = "BMI Category") +
    theme_light()
```

```
Picking joint bandwidth of 1.37
```

Ridgeline Plot of Triceps Skinfold by BMI Category (nn
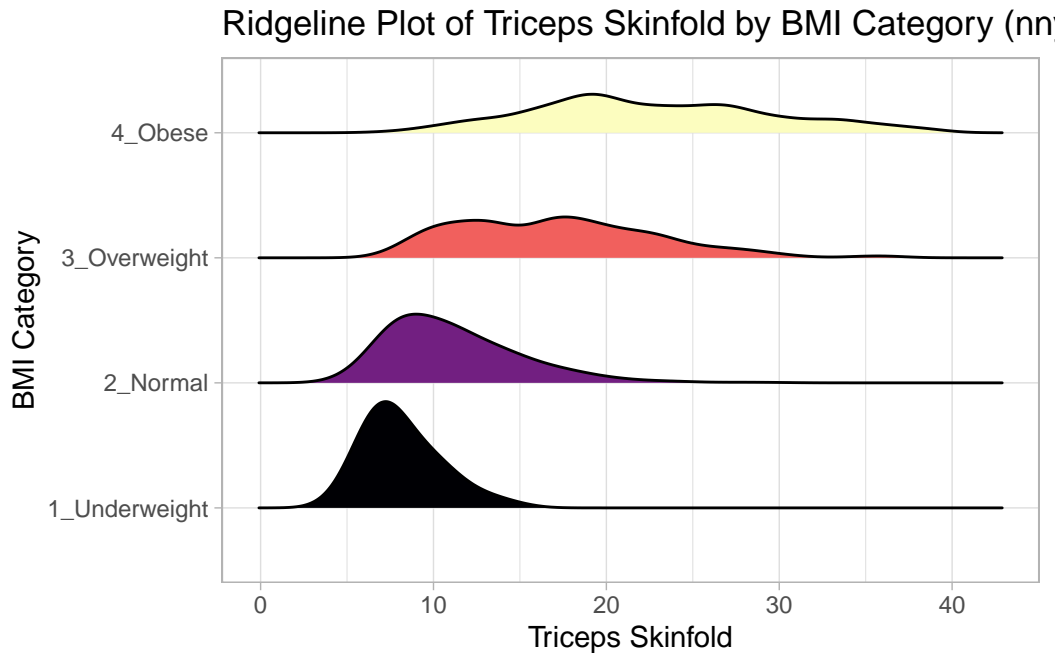
For one last example, we'll look at age by BMI category, so that sorting the BMI subgroups by the median matters, and we'll try an alternate color scheme, and a theme specially designed for the ridgeline plot.

```
nnyfs_temp7 <- nnyfs |>
  filter(complete.cases(bmi_cat, age_child)) |>
  mutate(bmi_cat = reorder(bmi_cat, age_child, median))

ggplot(nnyfs_temp7, aes(x = age_child, y = bmi_cat,
                        fill = bmi_cat, height = ..density..)) +
    geom_density_ridges(scale = 0.85) +
    scale_fill_brewer(palette = "YlOrRd") +
    guides(fill = "none") +
    labs(title = "Ridgeline Plot of Age at Exam by BMI category (nnyfs)",
         x = "Age of Child at Exam", y = "BMI Category") +
    theme_ridges()
```

```
Picking joint bandwidth of 1.15
```

Ridgeline Plot of Age at Exam by BMI cate
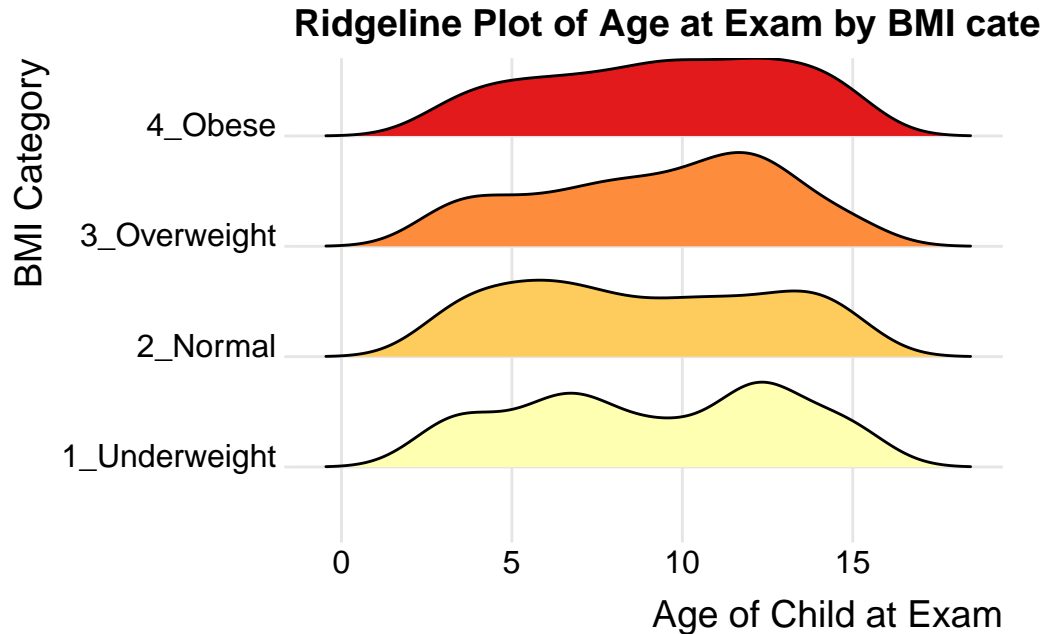
10.23 What Summaries to Report

It is usually helpful to focus on the shape, center and spread of a distribution. Bock, Velleman and DeVeaux provide some useful advice:

- If the data are skewed, report the median and IQR (or the three middle quantiles). You may want to include the mean and standard deviation, but you should point out why the mean and median differ. The fact that the mean and median do not agree is a sign that the distribution may be skewed. A histogram will help you make that point.
- If the data are symmetric, report the mean and standard deviation, and possibly the median and IQR as well.
- If there are clear outliers and you are reporting the mean and standard deviation, report them with the outliers present and with the outliers removed. The differences may be revealing. The median and IQR are not likely to be seriously affected by outliers.

10.24 Coming Up

Next, we'll look at the issue of Normalilty, and in particular how to assess whether a particular batch of data is well-approximated by the Normal distribution.

11 Assessing Normality

11.1 Setup: Packages Used Here

```r
knitr::opts_chunk$set(comment = NA)

library(patchwork)
library(tidyverse)

theme_set(theme_bw())
```

We also use the `favstat` function from the `mosaic` package in this chapter, but do not load the whole package.

11.2 Introduction

Data are well approximated by a Normal distribution if the shape of the data's distribution is a good match for a Normal distribution with mean and standard deviation equal to the sample statistics.

- the data are symmetrically distributed about a single peak, located at the sample mean
- the spread of the distribution is well characterized by a Normal distribution with standard deviation equal to the sample standard deviation
- the data show outlying values (both in number of candidate outliers, and size of the distance between the outliers and the center of the distribution) that are similar to what would be predicted by a Normal model.

We have several tools for assessing Normality of a single batch of data, including:

- a histogram with superimposed Normal distribution
- histogram variants (like the boxplot) which provide information on the center, spread and shape of a distribution
- the Empirical Rule for interpretation of a standard deviation

- a specialized *normal Q-Q plot* (also called a normal probability plot or normal quantile-quantile plot) designed to reveal differences between a sample distribution and what we might expect from a normal distribution of a similar number of values with the same mean and standard deviation

11.3 Empirical Rule Interpretation of the Standard Deviation

For a set of measurements that follows a Normal distribution, the interval:

- Mean $\pm$ Standard Deviation contains approximately 68% of the measurements;
- Mean $\pm$ 2(Standard Deviation) contains approximately 95% of the measurements;
- Mean $\pm$ 3(Standard Deviation) contains approximately all (99.7%) of the measurements.

Again, most data sets do not follow a Normal distribution. We will occasionally think about transforming or re-expressing our data to obtain results which are better approximated by a Normal distribution, in part so that a standard deviation can be more meaningful.

For the energy data we have been studying, here again are some summary statistics...

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

```
mosaic::favstats(nnyfs$energy)
```

```
 min    Q1 median   Q3  max    mean       sd    n missing
 257 1367.5 1794.5 2306 5265 1877.157 722.3537 1518       0
```

The mean is 1877 and the standard deviation is 722, so if the data really were Normally distributed, we'd expect to see:

- About 68% of the data in the range (1155, 2600). In fact, 1085 of the 1518 energy values are in this range, or 71.5%.
- About 95% of the data in the range (432, 3322). In fact, 1450 of the 1518 energy values are in this range, or 95.5%.
- About 99.7% of the data in the range (-290, 4044). In fact, 1502 of the 1518 energy values are in this range, or 98.9%.

So, based on this Empirical Rule approximation, do the energy data seem to be well approximated by a Normal distribution?

11.4 Describing Outlying Values with Z Scores

The maximum energy consumption value here is 5265. One way to gauge how extreme this is (or how much of an outlier it is) uses that observation's **Z score**, the number of standard deviations away from the mean that the observation falls.

Here, the maximum value, 5265 is 4.69 standard deviations above the mean, and thus has a Z score of 4.7.

A negative Z score would indicate a point below the mean, while a positive Z score indicates, as we've seen, a point above the mean. The minimum body-mass index, 257 is 2.24 standard deviations *below* the mean, so it has a Z score of -2.2.

Recall that the Empirical Rule suggests that if a variable follows a Normal distribution, it would have approximately 95% of its observations falling inside a Z score of (-2, 2), and 99.74% falling inside a Z score range of (-3, 3).

11.4.1 Fences and Z Scores

Note the relationship between the fences (Tukey's approach to identifying points which fall within the whiskers of a boxplot, as compared to candidate outliers) and the Z scores.

The upper inner fence in this case falls at 3713.75, which indicates a Z score of 2.5, while the lower inner fence falls at -40.25, which indicates a Z score of -2.7. It is neither unusual nor inevitable for the inner fences to fall at Z scores near -2.0 and +2.0.

11.5 Comparing a Histogram to a Normal Distribution

Most of the time, when we want to understand whether our data are well approximated by a Normal distribution, we will use a graph to aid in the decision.

One option is to build a histogram with a Normal density function (with the same mean and standard deviation as our data) superimposed. This is one way to help visualize deviations between our data and what might be expected from a Normal distribution.

```
res <- mosaic::favstats(~ energy, data = nnyfs)
bin_w <- 50 # specify binwidth

ggplot(nnyfs, aes(x=energy)) +
    geom_histogram(aes(y = ..density..), binwidth = bin_w,
                   fill = "papayawhip", color = "seagreen") +
    stat_function(fun = dnorm,
```

```
                  args = list(mean = res$mean, sd = res$sd),
                  lwd = 1.5, col = "blue") +
    geom_text(aes(label = paste("Mean", round(res$mean,1),
                           ", SD", round(res$sd,1))),
          x = 4000, y = 0.0006,
          color="blue", fontface = "italic") +
    labs(title = "nnyfs energy values with Normal Density Superimposed",
        x = "Energy (kcal)", y = "Probability Density Function")
```

Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
i Please use `after_stat(density)` instead.

nnyfs energy values with Normal Density Superimposed
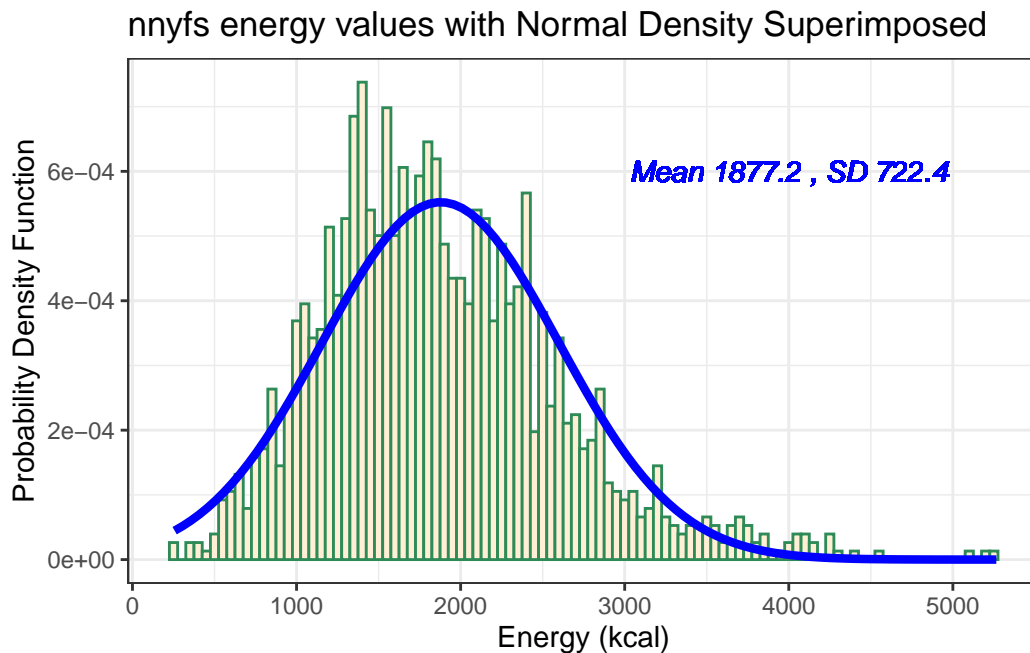
Does it seem as though the Normal model (as shown in the blue density curve) is an effective approximation to the observed distribution shown in the bars of the histogram?

We'll return shortly to the questions:

- Does a Normal distribution model fit our data well? *and*
- If the data aren't Normal, but we want to use a Normal model anyway, what should we do?

11.5.1 Histogram of `energy` with Normal model (with Counts)

But first, we'll demonstrate an approach to building a histogram of counts (rather than a probability density) and then superimposing a Normal model.

```
res <- mosaic::favstats(~ energy, data = nnyfs)
bin_w <- 50 # specify binwidth

ggplot(nnyfs, aes(x = energy)) +
  geom_histogram(binwidth = bin_w,
                 fill = "papayawhip",
                 col = "navy") +
  stat_function(
    fun = function(x) dnorm(x, mean = res$mean,
                            sd = res$sd) * res$n * bin_w,
    col = "blue", size = 2) +
    geom_text(aes(label = paste("Mean", round(res$mean,1),
                                ", SD", round(res$sd,1))),
              x = 4000, y = 50,
              color="blue", fontface = "italic") +
    labs(title = "Histogram of energy, with Normal Model",
         x = "Energy consumed (kcal)", y = "# of subjects")
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```

Histogram of energy, with Normal Model

Mean 1877.2 , SD 722.4
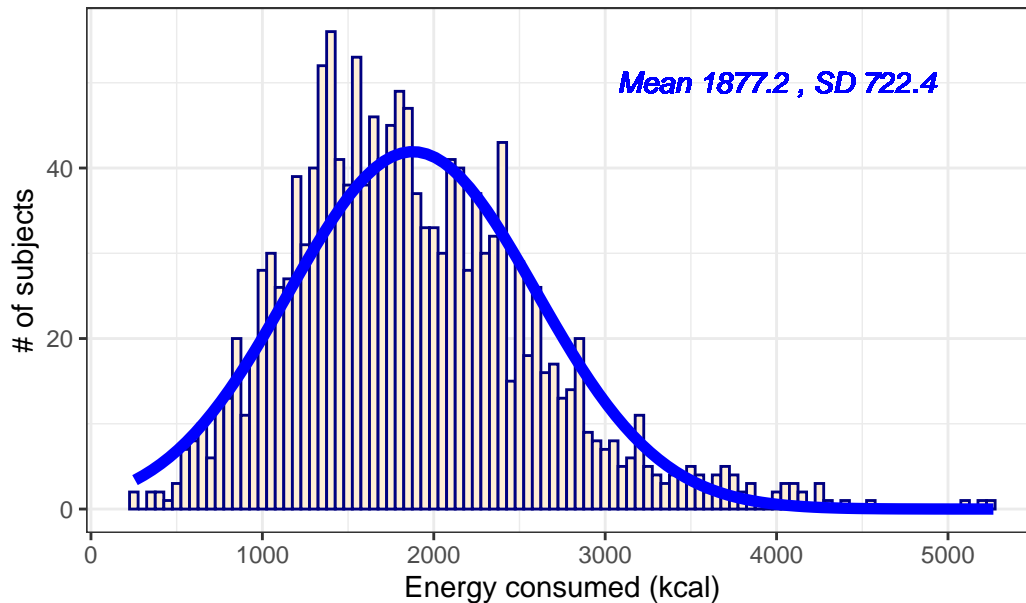
11.6 Does a Normal model work well for the `waist` circumference?

Now, suppose we instead look at the `waist` data, remembering to filter the data to the complete cases before plotting. Do these data appear to follow a Normal distribution?

```
res <- mosaic::favstats(~ waist, data = nnyfs)
bin_w <- 5 # specify binwidth

nnyfs_ccw <- nnyfs |>
  filter(complete.cases(waist))

ggplot(nnyfs_ccw, aes(x = waist)) +
    geom_histogram(binwidth = bin_w,
                   fill = "antiquewhite",
                   col = "navy") +
    stat_function(
        fun = function(x) dnorm(x, mean = res$mean,
                                sd = res$sd) *
            res$n * bin_w,
        col = "darkred", size = 2) +
    geom_text(aes(label = paste("Mean", round(res$mean,1),
```

```
                                  ", SD", round(res$sd,1))),
                   x = 100, y = 200,
                   color="darkred", fontface = "italic") +
     labs(title = "Histogram of waist, with Normal Model",
           x = "Waist Circumference (cm)", y = "# of subjects")
```

Histogram of waist, with Normal Model
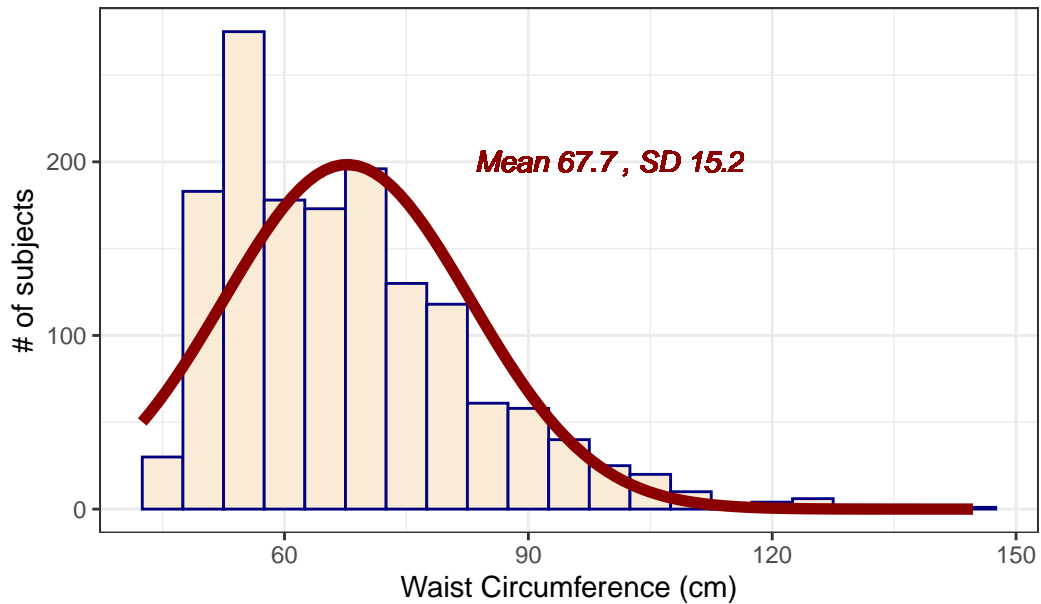

```
mosaic::favstats(~ waist, data = nnyfs)
```

```
  min    Q1 median   Q3   max     mean        sd    n missing
 42.5 55.6   64.8 76.6 144.7 67.70536 15.19809 1512       6
```

If we rerun this after excluding the missing values, only the last column changes.

```
mosaic::favstats(~ waist, data = nnyfs_ccw)
```

```
  min    Q1 median   Q3   max     mean        sd    n missing
 42.5 55.6   64.8 76.6 144.7 67.70536 15.19809 1512       0
```

The mean is 67.71 and the standard deviation is 15.2 so if the `waist` data really were Normally distributed, we'd expect to see:

- About 68% of the data in the range (52.51, 82.9). In fact, 1076 of the 1512 Age values are in this range, or 71.2%.

- About 95% of the data in the range (37.31, 98.1). In fact, 1443 of the 1512 Age values are in this range, or 95.4%.

- About 99.7% of the data in the range (22.11, 113.3). In fact, 1500 of the 1512 Age values are in this range, or 99.2%.

How does the Normal approximation work for waist circumference, according to the Empirical Rule?

11.7 The Normal Q-Q Plot

A normal probability plot (or normal quantile-quantile plot) of the energy results from the **nnyfs** data, developed using **ggplot2** is shown below. In this case, this is a picture of 1518 energy consumption assessments. The idea of a normal Q-Q plot is that it plots the observed sample values (on the vertical axis) and then, on the horizontal, the expected or theoretical quantiles that would be observed in a standard normal distribution (a Normal distribution with mean 0 and standard deviation 1) with the same number of observations.

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.

```
ggplot(nnyfs, aes(sample = energy)) +
    geom_qq() + geom_qq_line(col = "red") +
    theme_light() +
    labs(title = "Normal Q-Q plot for energy data")
```

Normal Q–Q plot for energy data
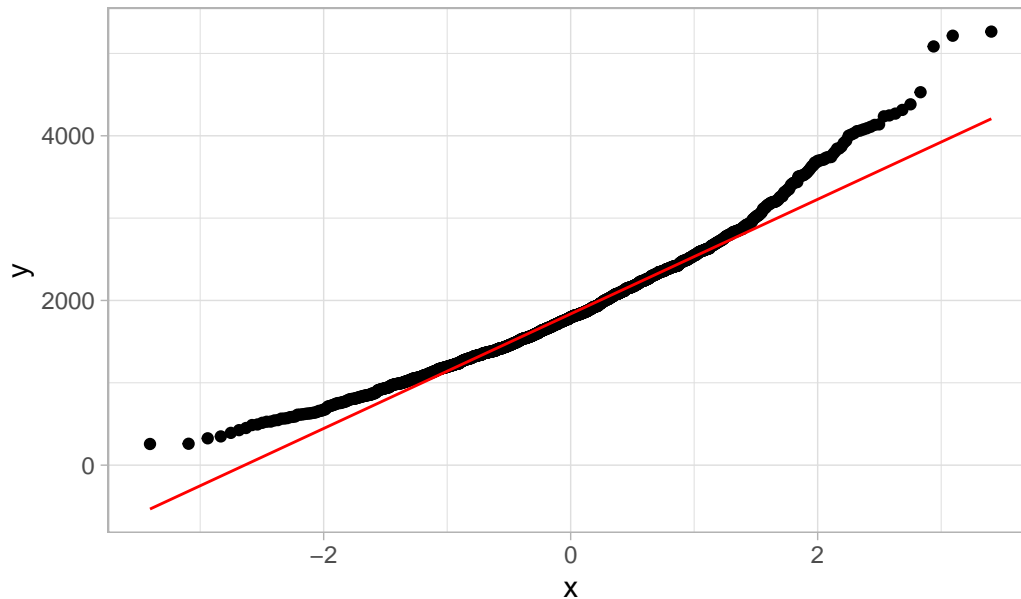
11.8 Interpreting the Normal Q-Q Plot

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)
- behavior in the tails (which could be heavy-tailed [more outliers than expected] or light-tailed)

11.8.1 Data from a Normal distribution shows up as a straight line in a Normal Q-Q plot

We'll demonstrate the looks that we can obtain from a Normal Q-Q plot in some simulations. First, here is an example of a Normal Q-Q plot, and its associated histogram, for a sample of 200 observations simulated from a Normal distribution.

```
set.seed(123431) # so the results can be replicated
```

```r
# simulate 200 observations from a Normal(20, 5) distribution and place them
# in the d variable within the temp_1 tibble
temp_1 <- tibble(d = rnorm(200, mean = 20, sd = 5))

# left plot - basic Normal Q-Q plot of simulated data
p1 <- ggplot(temp_1, aes(sample = d)) +
    geom_qq() + geom_qq_line(col = "red") +
    theme_light() +
    labs(y = "Ordered Simulated Sample Data")

# right plot - histogram with superimposed normal distribution
res <- mosaic::favstats(~ d, data = temp_1)
bin_w <- 2 # specify binwidth

p2 <- ggplot(temp_1, aes(x = d)) +
    geom_histogram(binwidth = bin_w,
                   fill = "papayawhip",
                   col = "seagreen") +
    stat_function(
        fun = function(x) dnorm(x, mean = res$mean,
                                sd = res$sd) *
            res$n * bin_w,
        col = "blue", size = 1.5) +
    geom_text(aes(label = paste("Mean", round(res$mean,1),
                                ", SD", round(res$sd,1))),
              x = 25, y = 35,
              color="blue", fontface = "italic") +
    labs(x = "Simulated Sample Data", y = "")

p1 + p2 +
  plot_annotation(title = "200 observations from a simulated Normal")
```

200 observations from a simulated Normal
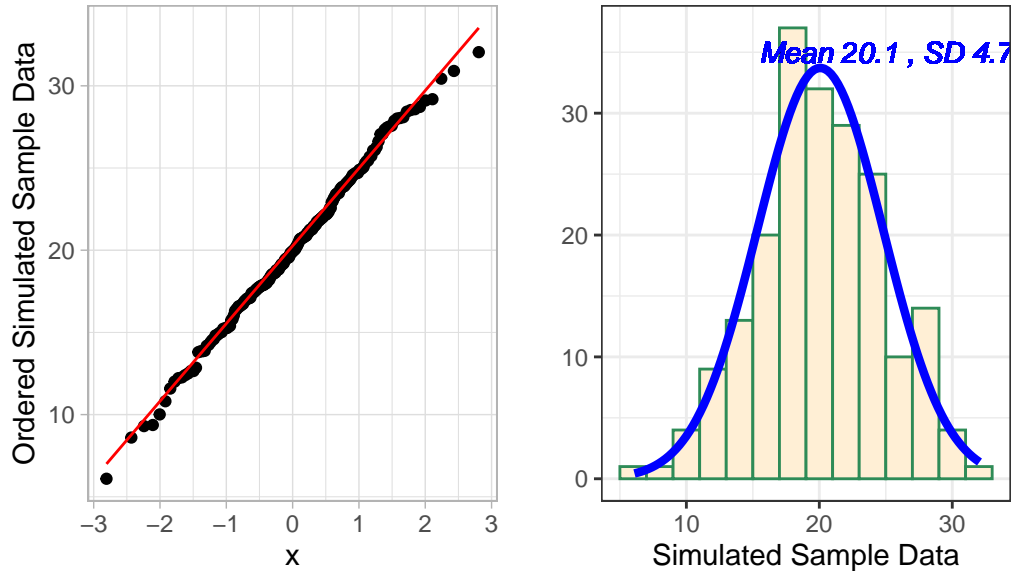

```
# uses patchwork package to combine plots
```

These simulated data appear to be well-modeled by the Normal distribution, because the points on the Normal Q-Q plot follow the diagonal reference line. In particular,

- there is no substantial curve (such as we'd see with data that were skewed)
- there is no particularly surprising behavior (curves away from the line) at either tail, so there's no obvious problem with outliers

11.8.2 Skew is indicated by monotonic curves in the Normal Q-Q plot

Data that come from a skewed distribution appear to curve away from a straight line in the Q-Q plot.

```
set.seed(123431) # so the results can be replicated

# simulate 200 observations from a beta(5, 2) distribution into the e1 variable
# simulate 200 observations from a beta(1, 5) distribution into the e2 variable
temp_2 <- tibble(e1 = rbeta(200, 5, 2), e2 = rbeta(200, 1, 5))

p1 <- ggplot(temp_2, aes(sample = e1)) +
```

```
    geom_qq(col = "orchid") + geom_qq_line(col = "blue") +
    theme_light() +
    labs(y = "Ordered Sample e1",
         title = "Beta(5, 2) sample: Left Skewed")

p2 <- ggplot(temp_2, aes(sample = e2)) +
    geom_qq(col = "darkorange") + geom_qq_line(col = "blue") +
    theme_light() +
    labs(y = "Ordered Sample e2",
         title = "Beta(1, 5) sample: Right Skewed")

p1 + p2 + plot_annotation(title = "200 observations from simulated Betas")
```


200 observations from simulated Betas
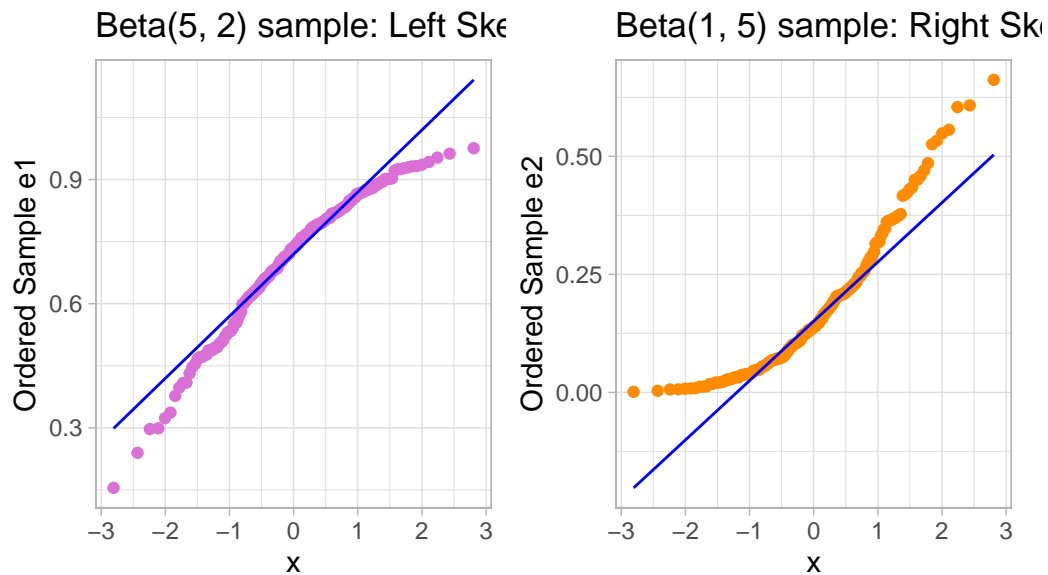
Note the bends away from a straight line in each sample. The non-Normality may be easier to see in a histogram.

```
res1 <- mosaic::favstats(~ e1, data = temp_2)
bin_w1 <- 0.025 # specify binwidth

p1 <- ggplot(temp_2, aes(x = e1)) +
    geom_histogram(binwidth = bin_w1,
```

```
                    fill = "orchid",
                    col = "black") +
    stat_function(
        fun = function(x) dnorm(x, mean = res1$mean,
                                sd = res1$sd) *
            res1$n * bin_w1,
        col = "blue", size = 1.5) +
labs(x = "Sample e1", y = "",
    title = "Beta(5,2) sample: Left Skew")

res2 <- mosaic::favstats(~ e2, data = temp_2)
bin_w2 <- 0.025 # specify binwidth

p2 <- ggplot(temp_2, aes(x = e2)) +
    geom_histogram(binwidth = bin_w2,
                    fill = "darkorange",
                    col = "black") +
    stat_function(
        fun = function(x) dnorm(x, mean = res2$mean,
                                sd = res2$sd) *
            res2$n * bin_w2,
        col = "blue", size = 1.5) +
labs(x = "Sample e1", y = "",
    title = "Beta(1,5) sample: Right Skew")

p1 + p2 + plot_annotation(caption = "Histograms with Normal curve superimposed")
```
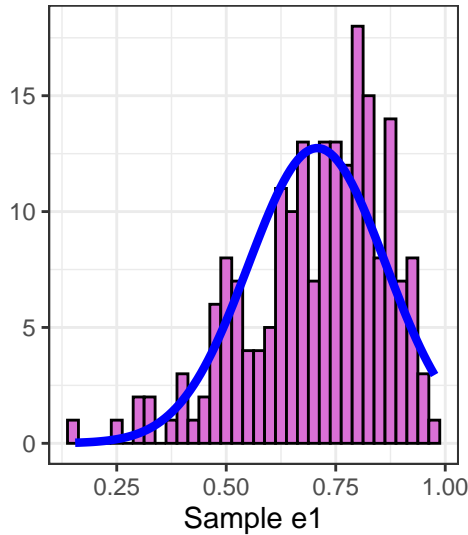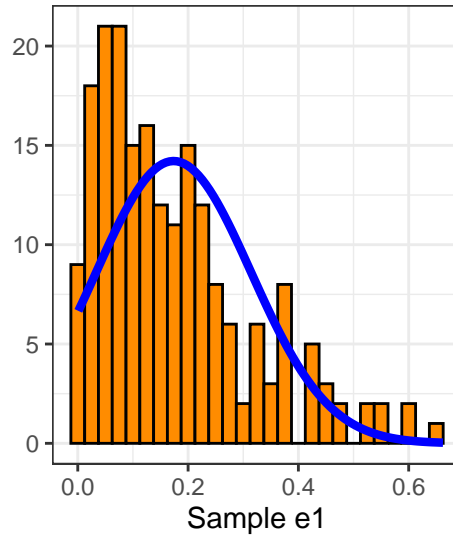
Histograms with Normal curve superimposed

11.8.3 Direction of Skew

In each of these pairs of plots, we see the same basic result.

- The left plot (for data e1) shows left skew, with a longer tail on the left hand side and more clustered data at the right end of the distribution.
- The right plot (for data e2) shows right skew, with a longer tail on the right hand side, the mean larger than the median, and more clustered data at the left end of the distribution.

11.8.4 Outlier-proneness is indicated by "s-shaped" curves in a Normal Q-Q plot

- Heavy-tailed but symmetric distributions are indicated by reverse "S"-shapes, as shown on the left below.
- Light-tailed but symmetric distributions are indicated by "S" shapes in the plot, as shown on the right below.

```
set.seed(4311) # so the results can be replicated

# sample 200 observations from each of two probability distributions
temp_3 <- tibble(s1 = rcauchy(200, location=10, scale = 1),
                 s2 = runif(200, -30, 30))
```

240

```
p1 <- ggplot(temp_3, aes(sample = s1)) +
    geom_qq(col = "slateblue") + geom_qq_line(col = "black") +
    theme_light() +
    labs(y = "Ordered Sample s1",
         title = "Heavy-Tailed Symmetric Sample s1")

p2 <- ggplot(temp_3, aes(sample = s2)) +
    geom_qq(col = "dodgerblue") + geom_qq_line(col = "black") +
    theme_light() +
    labs(y = "Ordered Sample s2",
         title = "Light-Tailed Symmetric Sample s2")

p1 + p2 + plot_annotation(title = "200 observations from simulated distributions")
```

200 observations from simulated distributions
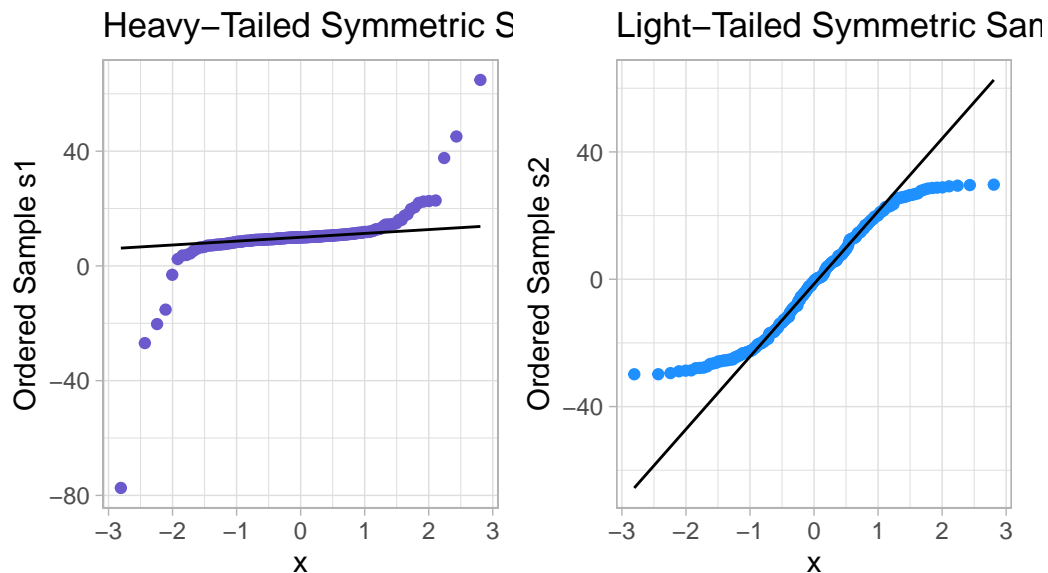
And, we can also visualize these simulations with histograms, although they're less helpful for understanding tail behavior than they are for skew.

```
res1 <- mosaic::favstats(~ s1, data = temp_3)
bin_w1 <- 20 # specify binwidth
```

```r
p1 <- ggplot(temp_3, aes(x = s1)) +
    geom_histogram(binwidth = bin_w1,
                   fill = "slateblue",
                   col = "white") +
    stat_function(
        fun = function(x) dnorm(x, mean = res1$mean,
                                   sd = res1$sd) *
            res1$n * bin_w1,
        col = "blue") +
labs(x = "Sample s1", y = "",
    title = "Cauchy sample: Heavy Tails")

res2 <- mosaic::favstats(~ s2, data = temp_3)
bin_w2 <- 2 # specify binwidth

p2 <- ggplot(temp_3, aes(x = s2)) +
    geom_histogram(binwidth = bin_w2,
                   fill = "dodgerblue",
                   col = "white") +
    stat_function(
        fun = function(x) dnorm(x, mean = res2$mean,
                                   sd = res2$sd) *
            res2$n * bin_w2,
        col = "blue") +
labs(x = "Sample s2", y = "",
    title = "Uniform sample: Light Tails")

p1 + p2 + plot_annotation(caption = "Histograms with Normal curve superimposed")
```
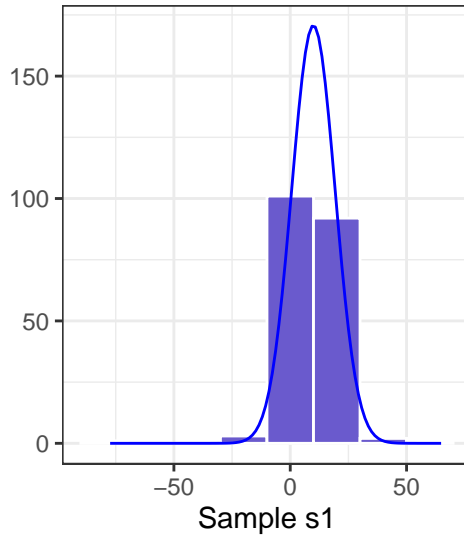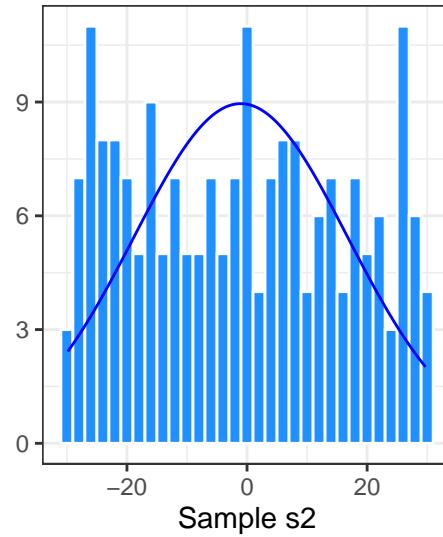
Histograms with Normal curve superimposed

Instead, boxplots (here augmented with violin plots) can be more helpful when thinking about light-tailed vs. heavy-tailed distributions.

```r
p1 <- ggplot(temp_3, aes(x = "s1", y = s1)) +
    geom_violin(col = "slateblue") +
    geom_boxplot(fill = "slateblue", width = 0.2) +
    theme_light() +
    coord_flip() +
    labs(y = "Ordered Sample s1", x = "",
         title = "Heavy-Tailed Symmetric Sample s1")

p2 <- ggplot(temp_3, aes(x = "s2", y = s2)) +
    geom_violin(col = "dodgerblue") +
    geom_boxplot(fill = "dodgerblue", width = 0.2) +
    theme_light() +
    coord_flip() +
    labs(y = "Ordered Sample s2", x = "",
         title = "Light-Tailed Symmetric Sample s2")

p1 + p2 + plot_annotation(title = "200 observations from simulated distributions")
```

200 observations from simulated distributions
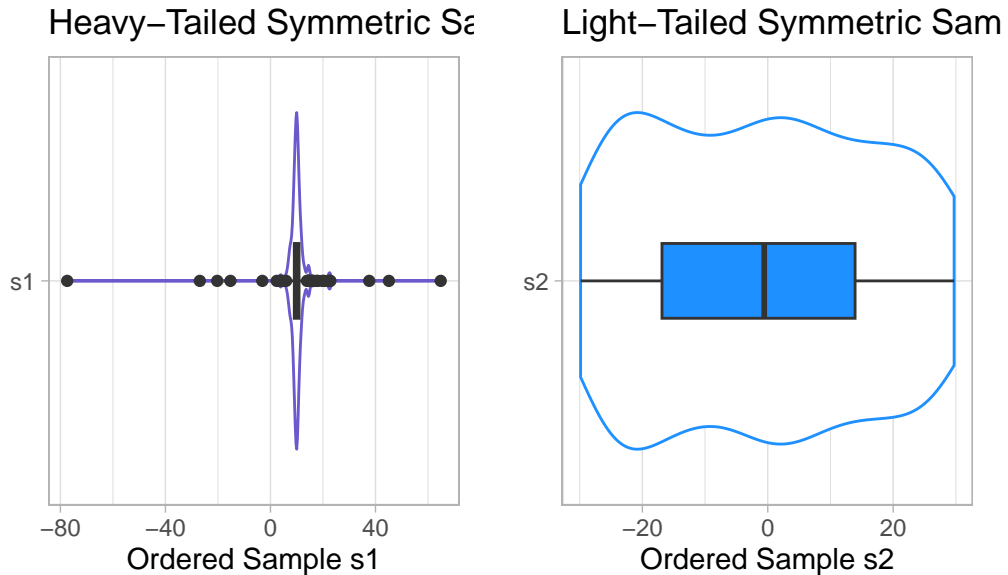
```
rm(temp_1, temp_2, temp_3, p1, p2,
   res, res1, res2, bin_w, bin_w1, bin_w2) # cleaning up
```

11.9 Can a Normal Distribution Fit the `nnyfs energy` data Well?

The `energy` data we've been studying shows meaningful signs of right skew.

```
p1 <- ggplot(nnyfs, aes(sample = energy)) +
    geom_qq(col = "coral", size = 2) +
    geom_qq_line(col = "blue") +
    theme_light() +
    labs(title = "Energy Consumed",
         y = "Sorted Energy data")

res <- mosaic::favstats(~ energy, data = nnyfs)
bin_w <- 250 # specify binwidth

p2 <- ggplot(nnyfs, aes(x = energy)) +
    geom_histogram(binwidth = bin_w,
                   fill = "coral",
```

```
                    col = "white") +
    stat_function(
        fun = function(x) dnorm(x, mean = res$mean,
                                sd = res$sd) *
            res$n * bin_w,
        col = "blue", size = 1.5) +
labs(x = "Energy (kcal consumed)", y = "",
    title = "Energy Consumed")


p1 + p2
```
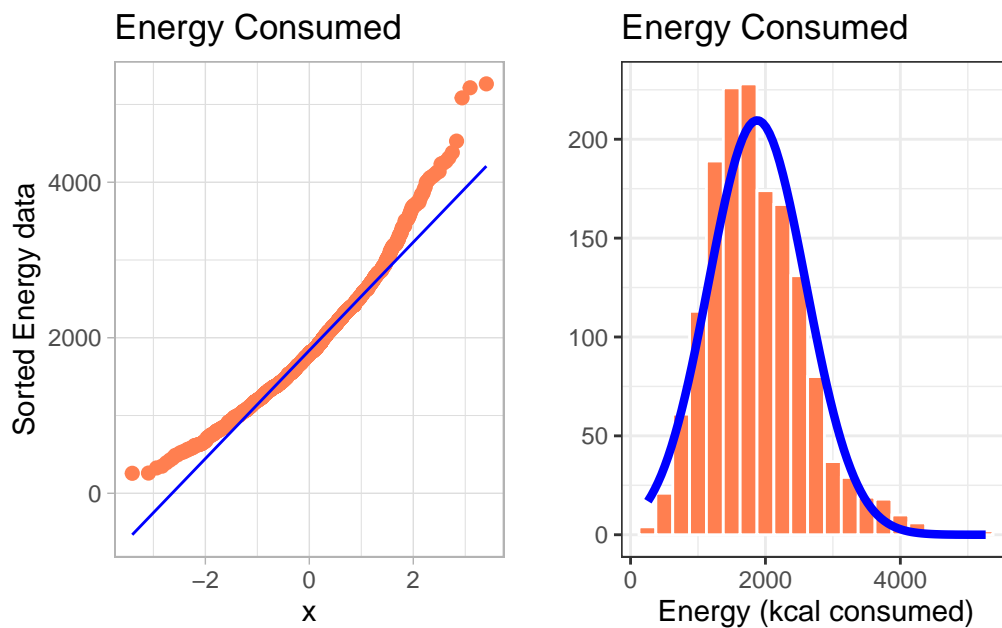

- Skewness is indicated by the curve in the Normal Q-Q plot. Curving up and away from the line in both tails suggests right skew, as does the histogram.

What if we plotted not the original `energy` values (all of which are positive) but instead plotted the square roots of the `energy` values?

- Compare these two plots - the left describes the distribution of the original energy data from the NNYFS data frame, and the right plot shows the distribution of the square root of those values.

```
p1 <- ggplot(nnyfs, aes(sample = energy)) +
    geom_qq(col = "coral", size = 2) +
    geom_qq_line(col = "blue") +
    theme_light() +
    labs(title = "Energy Consumed",
         y = "Sorted Energy data")

p2 <- ggplot(nnyfs, aes(sample = sqrt(energy))) +
    geom_qq(col = "darkcyan", size = 2) +
    geom_qq_line(col = "blue") +
    theme_light() +
    labs(title = "Square Root of Energy",
         y = "Sorted Square Root of Energy")

p1 + p2
```
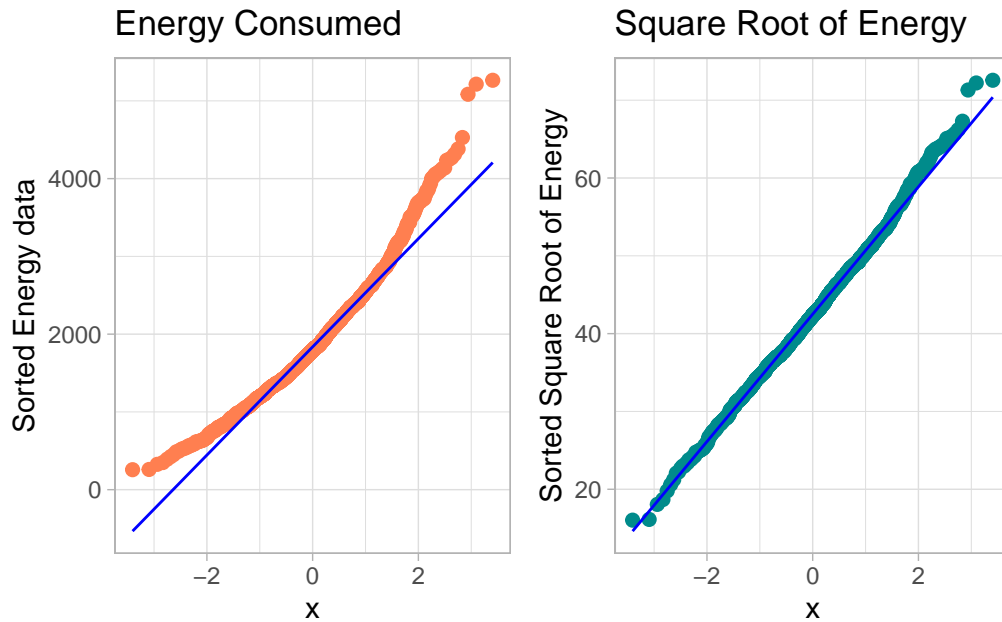

- The left plot shows substantial **right** or *positive* skew
- The right plot shows there's much less skew after the square root has been taken.

Our conclusion is that a Normal model is a far better fit to the square root of the energy values than it is to the raw energy values.

The effect of taking the square root may be clearer from the histograms below, with Normal

246

models superimposed.

```r
res <- mosaic::favstats(~ energy, data = nnyfs)
bin_w <- 250 # specify binwidth

p1 <- ggplot(nnyfs, aes(x = energy)) +
    geom_histogram(binwidth = bin_w,
                   fill = "coral",
                   col = "white") +
    stat_function(
        fun = function(x) dnorm(x, mean = res$mean,
                                   sd = res$sd) *
            res$n * bin_w,
        col = "black", size = 1.5) +
labs(x = "Energy (kcal consumed)", y = "",
     title = "Energy Consumed")


res2 <- mosaic::favstats(~ sqrt(energy), data = nnyfs)
bin_w2 <- 5 # specify binwidth

p2 <- ggplot(nnyfs, aes(x = sqrt(energy))) +
    geom_histogram(binwidth = bin_w2,
                   fill = "darkcyan",
                   col = "white") +
    stat_function(
        fun = function(x) dnorm(x, mean = res2$mean,
                                   sd = res2$sd) *
            res2$n * bin_w2,
        col = "black", size = 1.5) +
labs(x = "Square Root of Energy", y = "",
     title = "Square Root of Energy")


p1 + p2 + plot_annotation(title = "Comparing energy to sqrt(energy)")
```

Comparing energy to sqrt(energy)
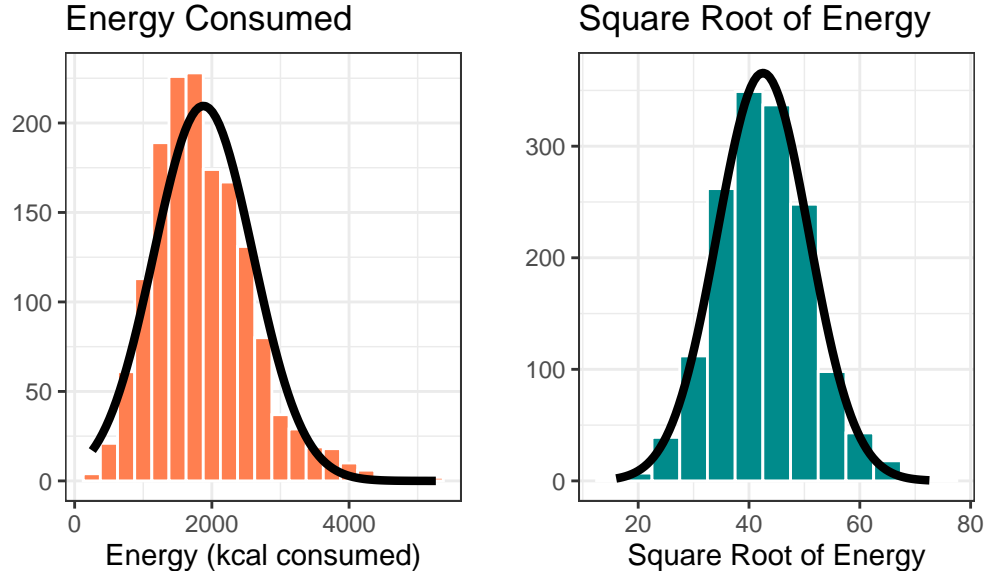
```
rm(p1, p2, bin_w, bin_w2, res, res2) # cleanup
```

When we are confronted with a variable that is not Normally distributed but that we wish was Normally distributed, it is sometimes useful to consider whether working with a **transformation** of the data will yield a more helpful result, as the square root does in this instance.

The rest of this Chapter provides some guidance about choosing from a class of power transformations that can reduce the impact of non-Normality in unimodal data.

- When we are confronted with a variable that is not Normally distributed but that we wish was Normally distributed, it is sometimes useful to consider whether working with a transformation of the data will yield a more helpful result.
- Many statistical methods, including t tests and analyses of variance, assume Normal distributions.
- We'll discuss using R to assess a range of what are called Box-Cox power transformations, via plots, mainly.

11.10 The Ladder of Power Transformations

The key notion in re-expression of a single variable to obtain a distribution better approximated by the Normal or re-expression of an outcome in a simple regression model is that of a **ladder of power transformations**, which applies to any unimodal data.

Power	Transformation
3	x^3
2	x^2
1	x (unchanged)
0.5	$x^{0.5} = \sqrt{x}$
0	ln x
-0.5	$x^{-0.5} = 1/\sqrt{x}$
-1	$x^{-1} = 1/x$
-2	$x^{-2} = 1/x^2$

11.11 Using the Ladder

As we move further away from the *identity* function (power = 1) we change the shape more and more in the same general direction.

- For instance, if we try a logarithm, and this seems like too much of a change, we might try a square root instead.
- Note that this ladder (which like many other things is due to John Tukey) uses the logarithm for the "power zero" transformation rather than the constant, which is what x^0 actually is.
- If the variable x can take on negative values, we might take a different approach. If x is a count of something that could be zero, we often simply add 1 to x before transformation.

The ladder of power transformations is particularly helpful when we are confronted with data that shows skew.

- To handle right skew (where the mean exceeds the median) we usually apply powers below 1.
- To handle left skew (where the median exceeds the mean) we usually apply powers greater than 1.

The most common transformations are the square (power 2), the square root (power 1/2), the logarithm (power 0) and the inverse (power -1), and I usually restrict myself to those options in practical work.

11.12 Protein Consumption in the NNYFS data

Here are the protein consumption (in grams) results from the NNYFS data.

```
mosaic::favstats(~ protein, data = nnyfs)
```

```
 min    Q1 median     Q3    max     mean       sd    n missing
4.18 45.33 61.255 82.565 241.84 66.90148 30.96319 1518       0
```

```
p1 <- ggplot(nnyfs, aes(x = "Protein", y = protein)) +
    geom_violin() +
    geom_boxplot(width = 0.2, fill = "salmon",
                outlier.color = "red") +

    labs(title = "NNYFS Protein consumption",
         x = "", y = "Protein Consumption (g)")

p2 <- ggplot(nnyfs, aes(sample = protein)) +
    geom_qq(col = "salmon") +
    geom_qq_line(col = "black") +

    labs(title = "NNYFS Protein Consumption",
         y = "Protein Consumption (g)")

p1 + p2
```
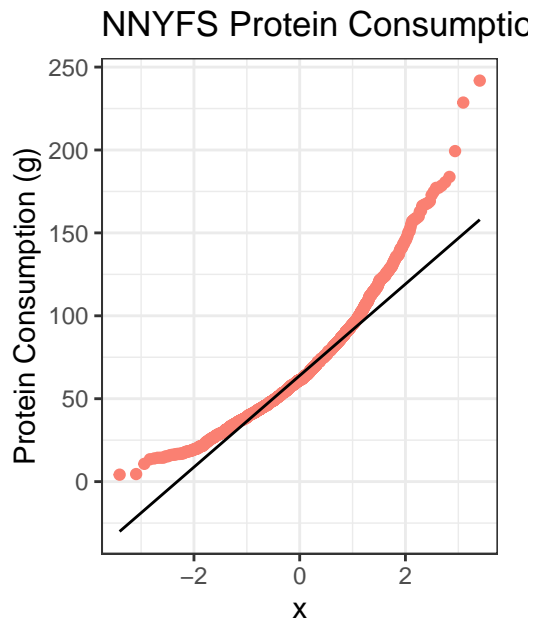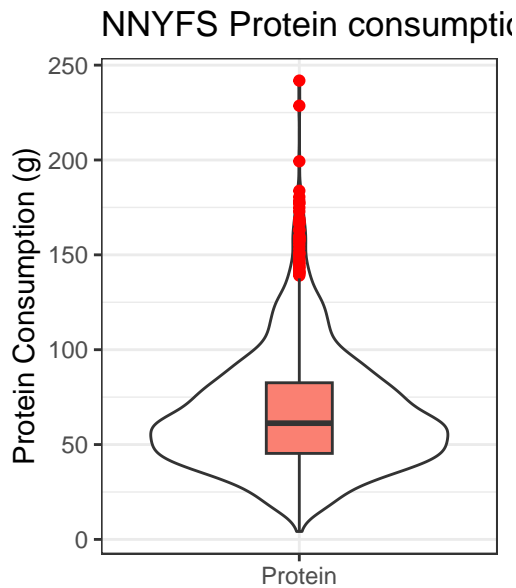
The key point here is that we see several signs of meaningful right skew, and we'll want to consider a transformation that might make a Normal model more plausible.

11.12.1 Using `patchwork` to compose plots

As we mentioned previously, I feel that the slickest approach to composing how a series of plots are placed together is available in the `patchwork` package. Here's another example.

```
res <- mosaic::favstats(~ protein, data = nnyfs)
bin_w <- 5 # specify binwidth

p1 <- ggplot(nnyfs, aes(x = protein)) +
    geom_histogram(binwidth = bin_w,
                   fill = "salmon",
                   col = "white") +

    stat_function(
        fun = function(x) dnorm(x, mean = res$mean,
                                sd = res$sd) *
            res$n * bin_w,
        col = "darkred", size = 2) +
    labs(title = "Histogram with Normal fit",
         x = "Protein Consumption (g)", y = "# of subjects")


p2 <- ggplot(nnyfs, aes(sample = protein)) +
    geom_qq(col = "salmon") +
    geom_qq_line(col = "black") +

    labs(title = "Normal Q-Q plot",
         y = "Protein Consumption (g)")

p3 <- ggplot(nnyfs, aes(x = "", y = protein)) +
    geom_violin() +
    geom_boxplot(width = 0.2, fill = "salmon",
                 outlier.color = "red") +

    coord_flip() +
    labs(title = "Boxplot with Violin",
         x = "", y = "Protein Consumption (g)")
```

```
p1 + p2 - p3 + plot_layout(ncol = 1, height = c(3, 1)) +
    plot_annotation(title = "NNYFS Protein Consumption")
```


NNYFS Protein Consumption
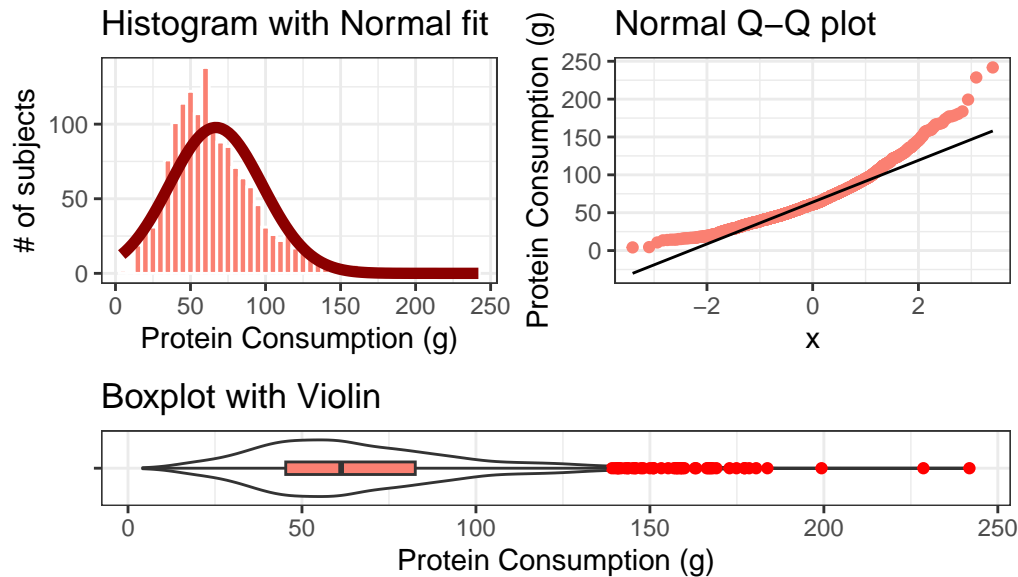
Again, the `patchwork` package repository at https://patchwork.data-imaginist.com/index.html has lots of nice examples to work from.

11.13 Can we transform the `protein` data?

As we've seen, the `protein` data are right skewed, and all of the values are strictly positive. If we want to use the tools of the Normal distribution to describe these data, we might try taking a step "down" our ladder from power 1 (raw data) to lower powers.

11.13.1 The Square Root

Would a square root applied to the protein data help alleviate that right skew?

```
res <- mosaic::favstats(~ sqrt(protein), data = nnyfs)
bin_w <- 1 # specify binwidth
```

```r
p1 <- ggplot(nnyfs, aes(x = sqrt(protein))) +
    geom_histogram(binwidth = bin_w,
                   fill = "salmon",
                   col = "white") +

    stat_function(
        fun = function(x) dnorm(x, mean = res$mean,
                                   sd = res$sd) *
            res$n * bin_w,
        col = "darkred", size = 2) +
    labs(title = "Histogram with Normal fit",
        x = "Square Root of Protein Consumption (g)", y = "# of subjects")


p2 <- ggplot(nnyfs, aes(sample = sqrt(protein))) +
    geom_qq(col = "salmon") +
    geom_qq_line(col = "black") +

    labs(title = "Normal Q-Q plot",
        y = "Square Root of Protein Consumption (g)")

p3 <- ggplot(nnyfs, aes(x = "", y = sqrt(protein))) +
    geom_violin() +
    geom_boxplot(width = 0.2, fill = "salmon",
                 outlier.color = "red") +

    coord_flip() +
    labs(title = "Boxplot with Violin",
        x = "", y = "Square Root of Protein Consumption (g)")

p1 + p2 - p3 + plot_layout(ncol = 1, height = c(3, 1)) +
    plot_annotation(title = "Square Root of NNYFS Protein Consumption")
```

Square Root of NNYFS Protein Consumption
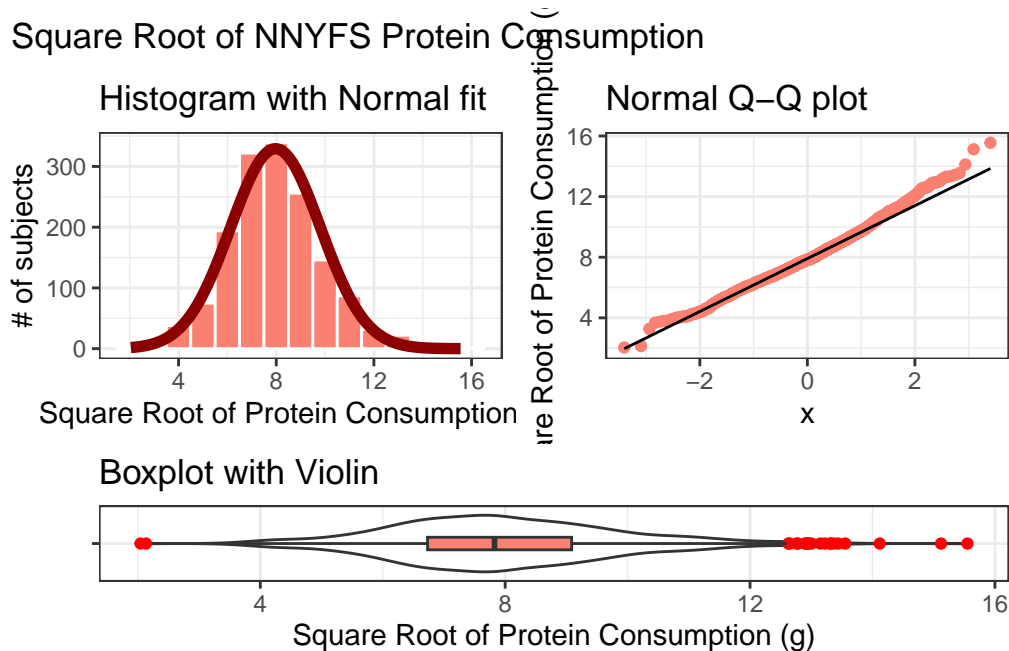
That looks like a more symmetric distribution, certainly, although we still have some outliers on the right side of the distribution. Should we take another step down the ladder?

11.13.2 The Logarithm

We might also try a logarithm of the energy circumference data. We can use either the natural logarithm (log, in R) or the base-10 logarithm (log10, in R) - either will have the same impact on skew.

```r
res <- mosaic::favstats(~ log(protein), data = nnyfs)
bin_w <- 0.5 # specify binwidth

p1 <- ggplot(nnyfs, aes(x = log(protein))) +
    geom_histogram(binwidth = bin_w,
                   fill = "salmon",
                   col = "white") +
    stat_function(
        fun = function(x) dnorm(x, mean = res$mean,
                                sd = res$sd) *
            res$n * bin_w,
        col = "darkred", size = 2) +
```

```
    labs(title = "Histogram with Normal fit",
         x = "Log of Protein Consumption (g)", y = "# of subjects")


p2 <- ggplot(nnyfs, aes(sample = log(protein))) +
    geom_qq(col = "salmon") +
    geom_qq_line(col = "black") +
    labs(title = "Normal Q-Q plot",
         y = "Log of Protein Consumption (g)")

p3 <- ggplot(nnyfs, aes(x = "", y = log(protein))) +
    geom_violin() +
    geom_boxplot(width = 0.2, fill = "salmon",
                 outlier.color = "red") +
    coord_flip() +
    labs(title = "Boxplot with Violin",
         x = "", y = "Log of Protein Consumption (g)")

p1 + p2 - p3 + plot_layout(ncol = 1, height = c(3, 1)) +
    plot_annotation(title = "Logarithm of NNYFS Protein Consumption")
```
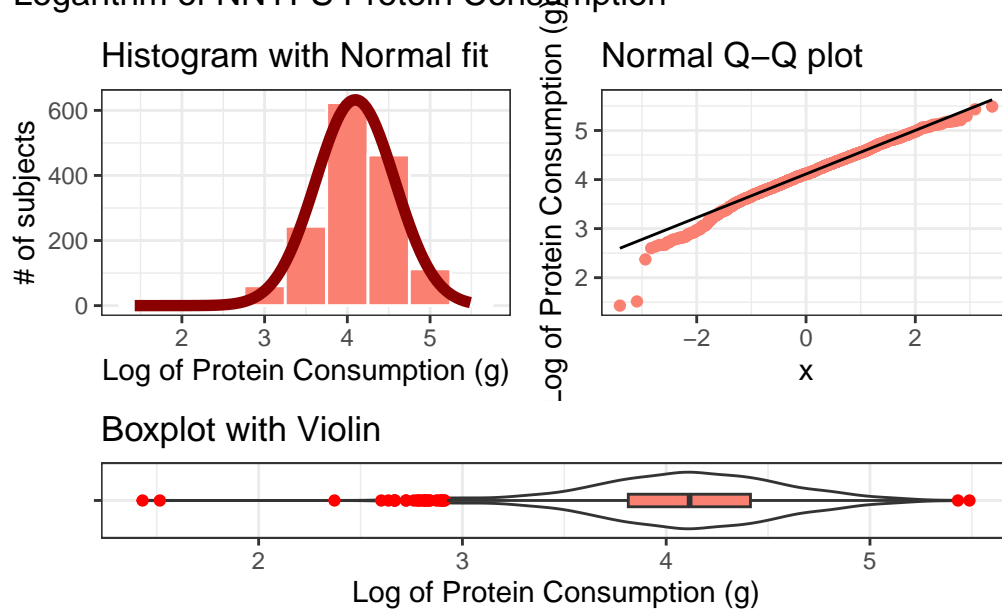

Now, it looks like we may have gone too far in the other direction. It looks like the square root

is a sensible choice to try to improve the fit of a Normal model to the protein consumption data.

11.13.3 This course uses Natural Logarithms, unless otherwise specified

In this course, we will assume the use of natural logarithms unless we specify otherwise. Following R's convention, we will use `log` for natural logarithms.

11.14 What if we considered all 9 available transformations?

```
p1 <- ggplot(nnyfs, aes(sample = protein^3)) +
    geom_qq(col = "salmon") +
    geom_qq_line(col = "black") +
    labs(title = "Cube (power 3)",
         y = "Protein, Cubed")

p2 <- ggplot(nnyfs, aes(sample = protein^2)) +
    geom_qq(col = "salmon") +
    geom_qq_line(col = "black") +
    labs(title = "Square (power 2)",
         y = "Protein, Squared")

p3 <- ggplot(nnyfs, aes(sample = protein)) +
    geom_qq(col = "salmon") +
    geom_qq_line(col = "black") +
    labs(title = "Original Data",
         y = "Protein (g)")


p4 <- ggplot(nnyfs, aes(sample = sqrt(protein))) +
    geom_qq(col = "salmon") +
    geom_qq_line(col = "black") +
    labs(title = "sqrt (power 0.5)",
         y = "Square Root of Protein")

p5 <- ggplot(nnyfs, aes(sample = log(protein))) +
    geom_qq(col = "salmon") +
    geom_qq_line(col = "black") +
    labs(title = "log (power 0)",
         y = "Natural Log of Protein")
```

```r
p6 <- ggplot(nnyfs, aes(sample = protein^(-0.5))) +
    geom_qq(col = "salmon") +
    geom_qq_line(col = "black") +
    labs(title = "1/sqrt (power -0.5)",
         y = "1/Square Root(Protein)")


p7 <- ggplot(nnyfs, aes(sample = 1/protein)) +
    geom_qq(col = "salmon") +
    geom_qq_line(col = "black") +
    labs(title = "Inverse (power -1)",
         y = "1/Protein")

p8 <- ggplot(nnyfs, aes(sample = 1/(protein^2))) +
    geom_qq(col = "salmon") +
    geom_qq_line(col = "black") +
    labs(title = "1/Square (power -2)",
         y = "1 /(Protein squared)")

p9 <- ggplot(nnyfs, aes(sample = 1/(protein^3))) +
    geom_qq(col = "salmon") +
    geom_qq_line(col = "black") +
    labs(title = "1/Cube (power -3)",
         y = "1/(Protein cubed)")


p1 + p2 + p3 + p4 + p5 + p6 + p7 + p8 + p9 +
    plot_layout(nrow = 3) +
    plot_annotation(title = "Transformations of NNYFS Protein Consumption")
```

Transformations of NNYFS Protein Consumption
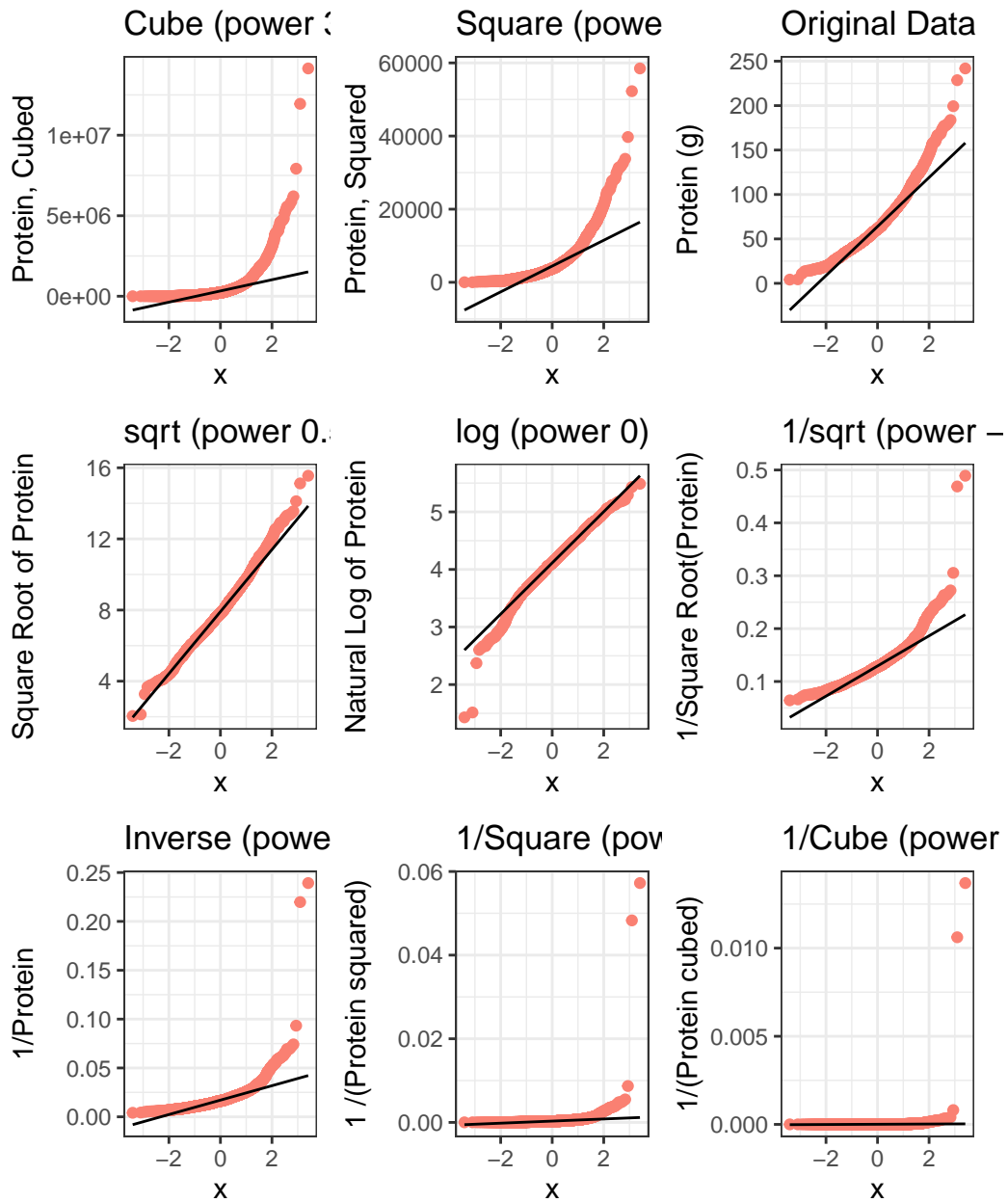
The square root still appears to be the best choice of transformation here, even after we consider all 8 transformation of the raw data.

11.15 A Simulated Data Set

```
set.seed(431);
sim_2 <-
    tibble(sample2 = 100*rbeta(n = 125, shape1 = 5, shape2 = 2))
```

If we'd like to transform these data so as to better approximate a Normal distribution, where should we start? What transformation do you suggest?

```
res <- mosaic::favstats(~ sample2, data = sim_2)
bin_w <- 4 # specify binwidth

p1 <- ggplot(sim_2, aes(x = sample2)) +
    geom_histogram(binwidth = bin_w,
                   fill = "royalblue",
                   col = "white") +
    stat_function(
        fun = function(x) dnorm(x, mean = res$mean,
                                sd = res$sd) *
            res$n * bin_w,
        col = "darkred", size = 2) +
    labs(title = "Histogram with Normal fit",
         x = "Simulated Data", y = "# of subjects")


p2 <- ggplot(sim_2, aes(sample = sample2)) +
    geom_qq(col = "royalblue") +
    geom_qq_line(col = "black") +
    labs(title = "Normal Q-Q plot",
         y = "Simulated Data")

p3 <- ggplot(sim_2, aes(x = "", y = sample2)) +
    geom_violin() +
    geom_boxplot(width = 0.3, fill = "royalblue",
                 outlier.color = "royalblue") +
    coord_flip() +
    labs(title = "Boxplot with Violin",
         x = "", y = "Simulated Data")

p1 + p2 - p3 + plot_layout(ncol = 1, height = c(3, 1)) +
    plot_annotation(title = "Simulated Data")
```

Simulated Data
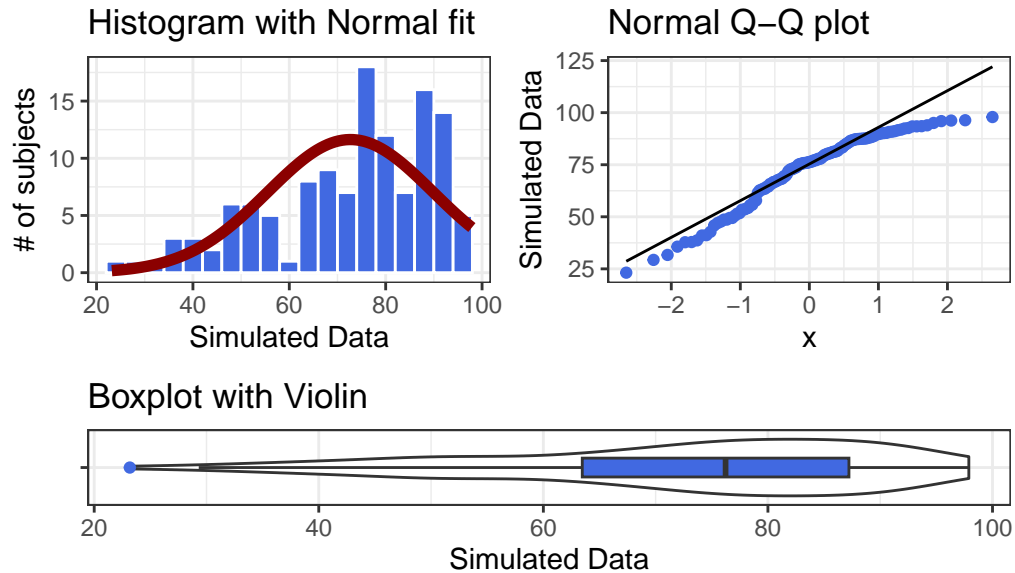
Given the left skew in the data, it looks like a step up in the ladder is warranted, perhaps by looking at the square of the data?

```r
res <- mosaic::favstats(~ sample2^2, data = sim_2)
bin_w <- 600 # specify binwidth

p1 <- ggplot(sim_2, aes(x = sample2^2)) +
    geom_histogram(binwidth = bin_w,
                   fill = "royalblue",
                   col = "white") +
    stat_function(
        fun = function(x) dnorm(x, mean = res$mean,
                                sd = res$sd) *
            res$n * bin_w,
        col = "darkred", size = 2) +
    labs(title = "Histogram with Normal fit",
         x = "Squared Simulated Data", y = "# of subjects")


p2 <- ggplot(sim_2, aes(sample = sample2^2)) +
    geom_qq(col = "royalblue") +
    geom_qq_line(col = "black") +
```

```
        labs(title = "Normal Q-Q plot",
             y = "Squared Simulated Data")

p3 <- ggplot(sim_2, aes(x = "", y = sample2^2)) +
    geom_violin() +
    geom_boxplot(width = 0.3, fill = "royalblue",
                 outlier.color = "royalblue") +
    coord_flip() +
    labs(title = "Boxplot with Violin",
         x = "", y = "Squared Simulated Data")

p1 + p2 - p3 + plot_layout(ncol = 1, height = c(3, 1)) +
    plot_annotation(title = "Squared Simulated Data")
```
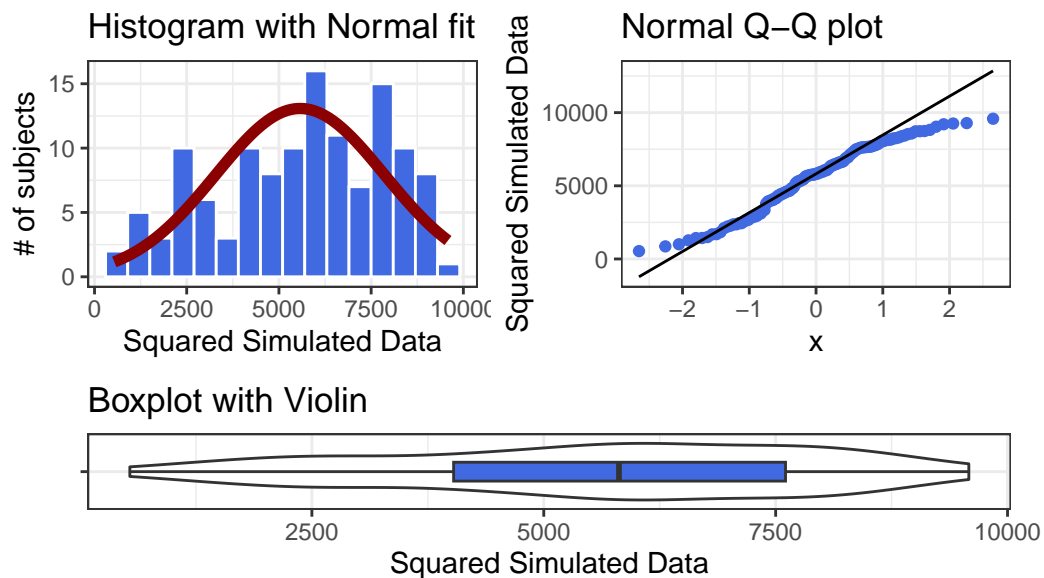

Looks like at best a modest improvement. How about cubing the data, instead?

```
res <- mosaic::favstats(~ sample2^3, data = sim_2)
bin_w <- 100000 # specify binwidth

p1 <- ggplot(sim_2, aes(x = sample2^3)) +
    geom_histogram(binwidth = bin_w,
```

```
                  fill = "royalblue",
                  col = "white") +
    stat_function(
        fun = function(x) dnorm(x, mean = res$mean,
                                sd = res$sd) *
            res$n * bin_w,
        col = "darkred", size = 2) +
    labs(title = "Histogram with Normal fit",
        x = "Cubed Simulated Data", y = "# of subjects")


p2 <- ggplot(sim_2, aes(sample = sample2^3)) +
    geom_qq(col = "royalblue") +
    geom_qq_line(col = "black") +
    labs(title = "Normal Q-Q plot",
        y = "Cubed Simulated Data")

p3 <- ggplot(sim_2, aes(x = "", y = sample2^3)) +
    geom_violin() +
    geom_boxplot(width = 0.3, fill = "royalblue",
                 outlier.color = "royalblue") +
    coord_flip() +
    labs(title = "Boxplot with Violin",
        x = "", y = "Cubed Simulated Data")

p1 + p2 - p3 + plot_layout(ncol = 1, height = c(3, 1)) +
    plot_annotation(title = "Cubed Simulated Data")
```

Cubed Simulated Data
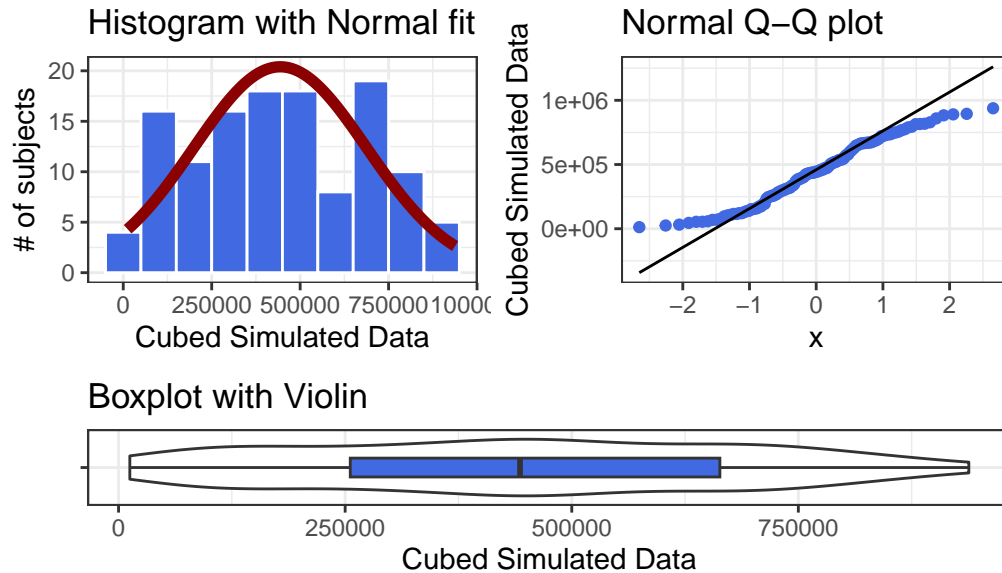
The newly transformed (cube of the) data appears more symmetric, although somewhat light-tailed. Perhaps a Normal model would be more appropriate now, although the standard deviation is likely to overstate the variation we see in the data due to the light tails. Again, I wouldn't be thrilled using a cube in practical work, as it is so hard to interpret, but it does look like a reasonable choice here.

11.16 What if we considered all 9 available transformations?

```
p1 <- ggplot(sim_2, aes(sample = sample2^3)) +
    geom_qq(col = "royalblue") +
    geom_qq_line(col = "black") +
    labs(title = "Cube (power 3)")

p2 <- ggplot(sim_2, aes(sample = sample2^2)) +
    geom_qq(col = "royalblue") +
    geom_qq_line(col = "black") +
    labs(title = "Square (power 2)")

p3 <- ggplot(sim_2, aes(sample = sample2)) +
    geom_qq(col = "royalblue") +
```

```r
    geom_qq_line(col = "black") +
    labs(title = "Original Data")

p4 <- ggplot(sim_2, aes(sample = sqrt(sample2))) +
    geom_qq(col = "royalblue") +
    geom_qq_line(col = "black") +
    labs(title = "sqrt (power 0.5)")

p5 <- ggplot(sim_2, aes(sample = log(sample2))) +
    geom_qq(col = "royalblue") +
    geom_qq_line(col = "black") +
    labs(title = "log (power 0)")

p6 <- ggplot(sim_2, aes(sample = sample2^(0.5))) +
    geom_qq(col = "royalblue") +
    geom_qq_line(col = "black") +
    labs(title = "1/sqrt (power -0.5)")

p7 <- ggplot(sim_2, aes(sample = 1/sample2)) +
    geom_qq(col = "royalblue") +
    geom_qq_line(col = "black") +
    labs(title = "Inverse (power -1)")

p8 <- ggplot(sim_2, aes(sample = 1/(sample2^2))) +
    geom_qq(col = "royalblue") +
    geom_qq_line(col = "black") +
    labs(title = "1/Square (power -2)")

p9 <- ggplot(sim_2, aes(sample = 1/(sample2^3))) +
    geom_qq(col = "royalblue") +
    geom_qq_line(col = "black") +
    labs(title = "1/Cube (power -3)")

p1 + p2 + p3 + p4 + p5 + p6 + p7 + p8 + p9 +
    plot_layout(nrow = 3) +
    plot_annotation(title = "Transformations of Simulated Sample")
```

Transformations of Simulated Sample
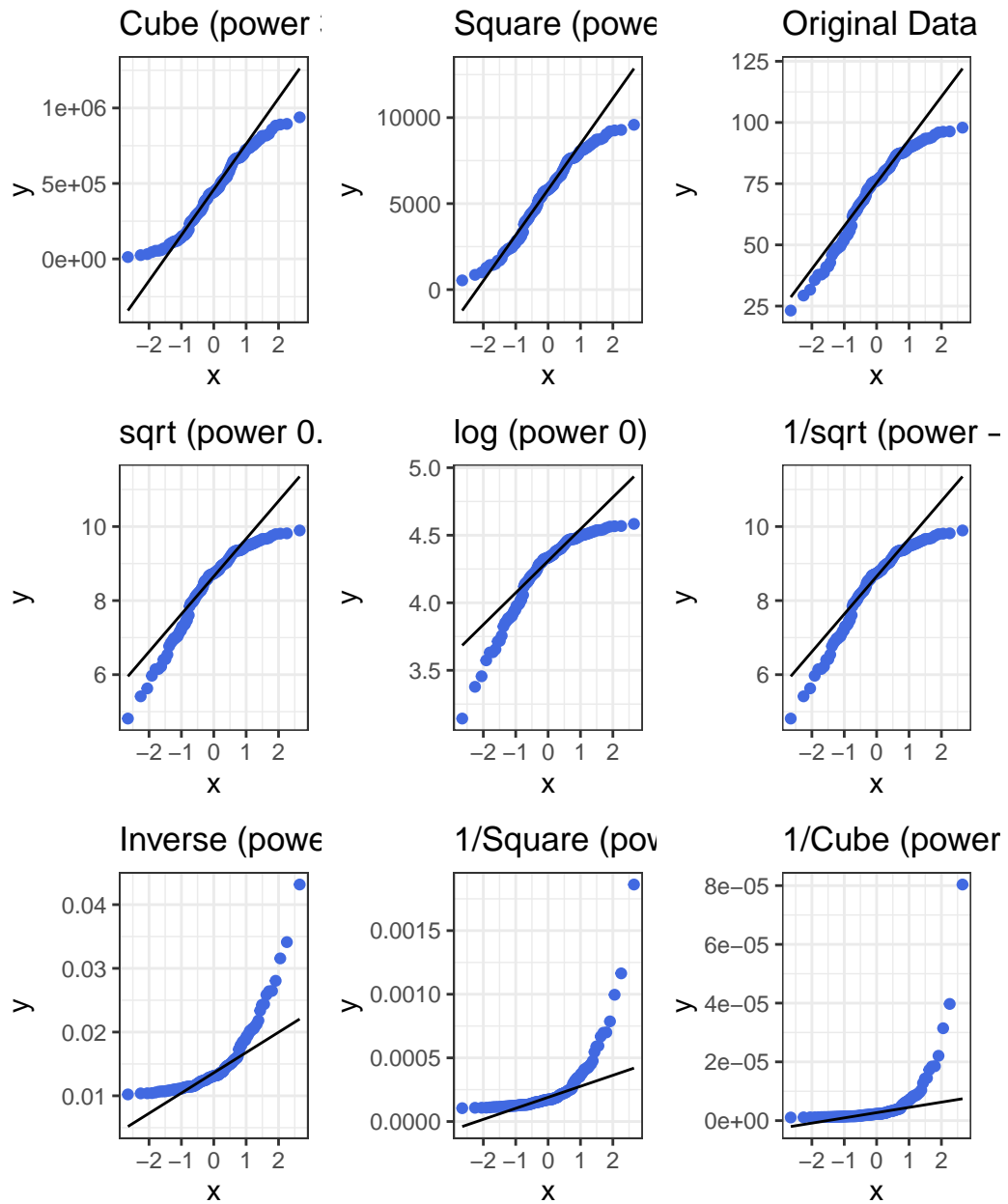
Again, either the cube or the square looks like best choice here, in terms of creating a more symmetric (albeit light-tailed) distribution.

11.17 Coming Up

Next, we'll spend some time thinking about scatterplots, correlation and straight line regression models.

12 Straight Line Models

12.1 Setup: Packages Used Here

```r
knitr::opts_chunk$set(comment = NA)

library(broom)
library(janitor)
library(kableExtra)
library(modelsummary)
library(patchwork)
library(tidyverse)

theme_set(theme_bw())
```

12.2 Assessing A Scatterplot

Let's consider the relationship of `protein` and `fat` consumption for children in the `nnyfs` data.

```r
nnyfs <- read_rds("data/nnyfs.Rds")
```

We'll begin our investigation, as we always should, by drawing a relevant picture. For the association of two quantitative variables, a **scatterplot** is usually the right start. Each subject in the `nnyfs` data is represented by one of the points below. To the plot, I've also used `geom_smooth` to add a straight line regression model, which we'll discuss later.

```r
ggplot(data = nnyfs, aes(x = protein, y = fat)) +
    geom_point(shape = 1, size = 2, col = "sienna") +
    geom_smooth(method = "lm", formula = y ~ x,
                se = FALSE, col = "steelblue") +
    labs(title = "Protein vs. Fat consumption in NNYFS data",
         subtitle = "with fitted straight line regression model",
```

```
x = "Protein (in g)", y = "Fat (in g)")
```

Protein vs. Fat consumption in NNYFS data
with fitted straight line regression model
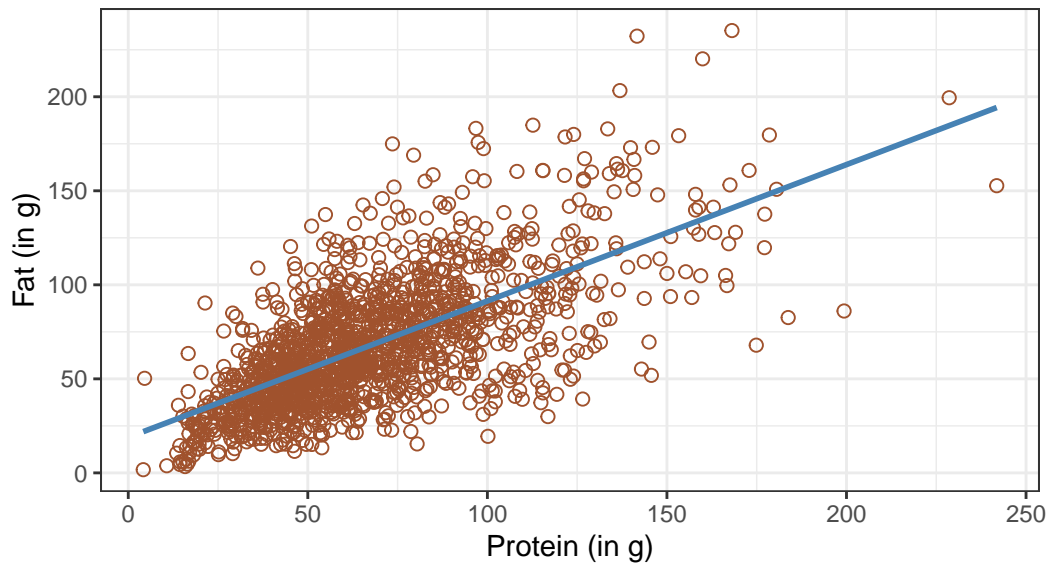
Here, I've arbitrarily decided to place `fat` on the vertical axis, and `protein` on the horizontal. Fitting a prediction model to this scatterplot will then require that we predict `fat` on the basis of `protein`.

In this case, the pattern appears to be:

1. **direct**, or positive, in that the values of the x variable (`protein`) increase, so do the values of the y variable (`fat`). Essentially, it appears that subjects who consumed more protein also consumed more fat, but we don't know cause and effect here.
2. fairly **linear** in that most of the points cluster around what appears to be a pattern which is well-fitted by a straight line.
3. moderately **strong** in that the range of values for `fat` associated with any particular value of `protein` is fairly tight. If we know someone's protein consumption, that should meaningfully improve our ability to predict their fat consumption, among the subjects in these data.
4. that we see some unusual or **outlier** values, further away from the general pattern of most subjects shown in the data.

12.3 Highlighting an unusual point

Consider the subject with protein consumption close to 200 g, whose fat consumption is below 100 g. That's well below the prediction of the linear model for example. We can identify the subject because it is the only person with `protein` > 190 and `fat` < 100 with BMI > 35 and waist.circ < 70. So I'll create a subset of the `nnyfs` data containing the point that meets that standard, and then add a red point and a label to the plot.

```
# identify outlier and place it in nnyfs_temp1 tibble
nnyfs_temp1 <- nnyfs |>
  filter(protein > 190 & fat < 100)

ggplot(data = nnyfs, aes(x = protein, y = fat)) +
    geom_point(shape = 1, size = 2, col = "sienna") +
    geom_smooth(method = "lm", se = FALSE, formula = y ~ x, col = "steelblue") +
    geom_point(data = nnyfs_temp1, size = 2, col = "red") +
    geom_text(data = nnyfs_temp1, label = nnyfs_temp1$SEQN,
              vjust = -1, col = "red") +
    labs(title = "Protein vs. Fat consumption in NNYFS",
         subtitle = "with regression line, and an outlier labeled by SEQN",
         x = "Protein (in g)", y = "Fat (in g)")
```
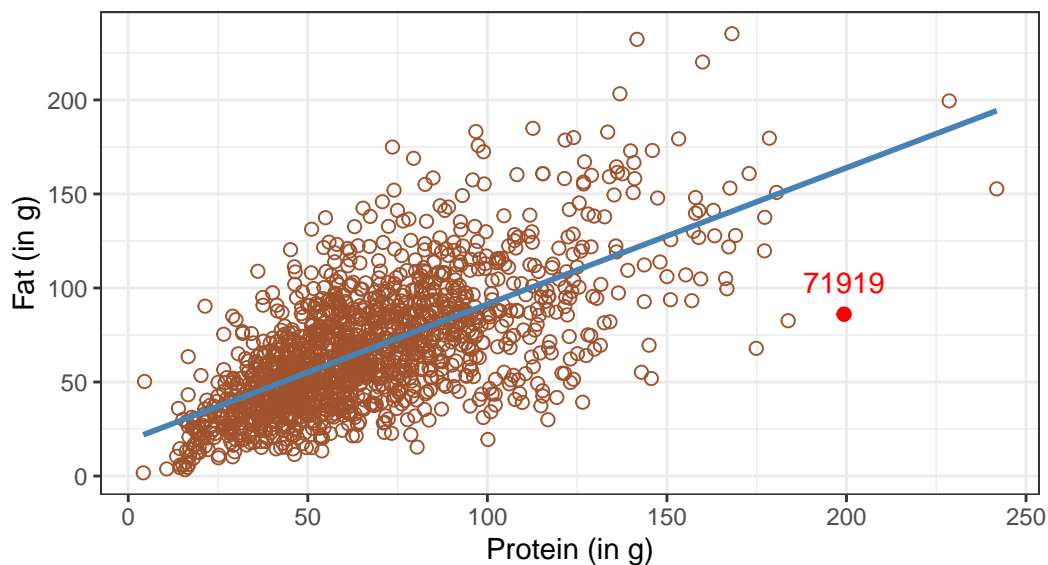
While this subject is hardly the only unusual point in the data set, it is one of the more unusual ones, in terms of its vertical distance from the regression line. We can identify the subject by printing (part of) the tibble we created.

```
nnyfs_temp1 |>
    select(SEQN, sex, race_eth, age_child, protein, fat) |> kable()
```

SEQN	sex	race_eth	age_child	protein	fat
71919	Female	2_White Non-Hispanic	14	199.33	86.08

Now, does it seem to you like a straight line model will describe this `protein-fat` relationship well?

12.4 Adding a Scatterplot Smooth using loess

Next, we'll use the **loess** procedure to fit a smooth curve to the data, which attempts to capture the general pattern.

```
ggplot(data = nnyfs, aes(x = protein, y = fat)) +
    geom_point(shape = 1, size = 2, col = "sienna") +
    geom_smooth(method = "loess", se = FALSE, formula = y ~ x, col = "red") +
    geom_smooth(method = "lm", se = FALSE, formula = y ~ x, col = "steelblue") +
    labs(title = "Protein vs. Fat consumption in NNYFS",
         subtitle = "with loess smooth (red) and linear fit (blue)",
         x = "Protein (in g)", y = "Fat (in g)")
```

Protein vs. Fat consumption in NNYFS
with loess smooth (red) and linear fit (blue)
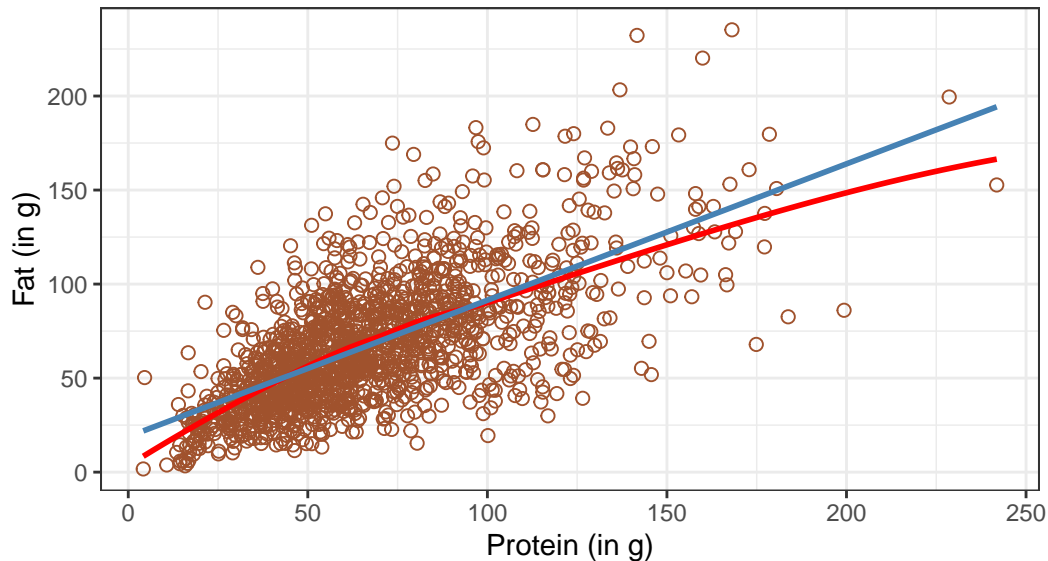
This "loess" smooth curve is fairly close to the straight line fit, indicating that perhaps a linear regression model might fit the data well.

A **loess smooth** is a method of fitting a local polynomial regression model that R uses as its default smooth for scatterplots with fewer than 1000 observations. Think of the loess as a way of fitting a curve to data by tracking (at point x) the points within a neighborhood of point x, with more emphasis given to points near x. It can be adjusted by tweaking two specific parameters, in particular:

- a `span` parameter (defaults to 0.75) which is also called α in the literature, that controls the degree of smoothing (essentially, how large the neighborhood should be), and
- a `degree` parameter (defaults to 2) which specifies the degree of polynomial to be used. Normally, this is either 1 or 2 - more complex functions are rarely needed for simple scatterplot smoothing.

In addition to the curve, smoothing procedures can also provide confidence intervals around their main fitted line. Consider the following plot, which adjusts the span and also adds in the confidence intervals.

```
p1 <- ggplot(data = nnyfs, aes(x = protein, y = fat)) +
    geom_point(shape = 1, size = 2, col = "sienna") +
    geom_smooth(method = "loess", span = 0.75, se = TRUE,
                col = "red", formula = y ~ x) +
```

```
    labs(title = "loess smooth (span = 0.75)",
         x = "Protein (in g)", y = "Fat (in g)")

p2 <- ggplot(data = nnyfs, aes(x = protein, y = fat)) +
    geom_point(shape = 1, size = 2, col = "sienna") +
    geom_smooth(method = "loess", span = 0.2, se = TRUE,
                col = "red", formula = y ~ x) +
    labs(title = "loess smooth (span = 0.2)",
         x = "Protein (in g)", y = "Fat (in g)")

p1 + p2 +
    plot_annotation(title = "Impact of adjusting loess smooth span: NNYFS")
```

Impact of adjusting loess smooth span: NNYFS
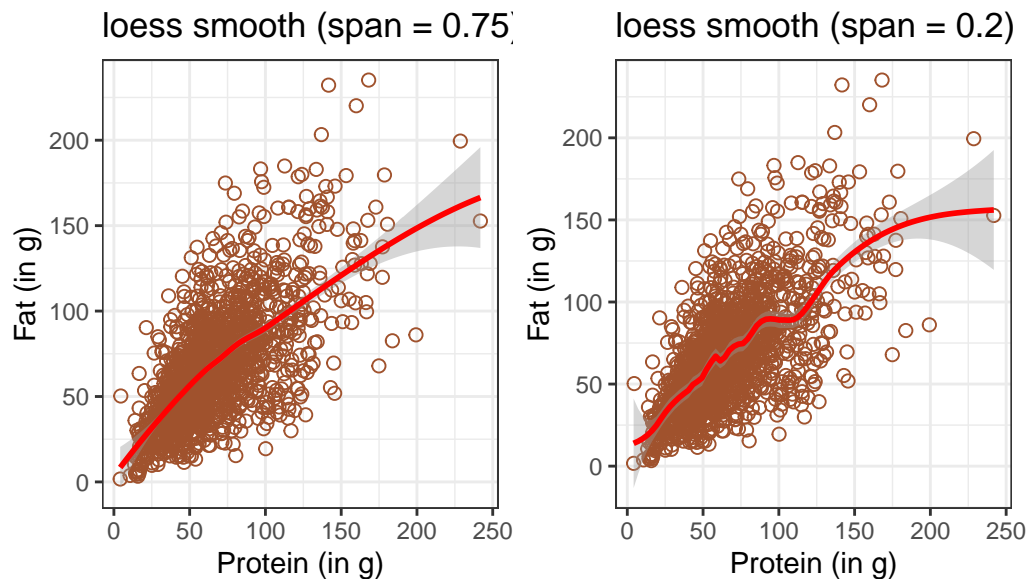
By reducing the size of the span, the plot on the right shows a somewhat less "smooth" function than the plot on the left.

12.5 Equation for a Linear Model

Returning to the linear regression model, how can we, mathematically, characterize that line? As with any straight line, our model equation requires us to specify two parameters: a slope

and an intercept (sometimes called the y-intercept.)

To identify the equation R used to fit this line (using the method of least squares), we use the `lm` command

```
m <- lm(fat ~ protein, data = nnyfs)
m
```

```
Call:
lm(formula = fat ~ protein, data = nnyfs)

Coefficients:
(Intercept)      protein
    18.8945       0.7251
```

So the fitted line contained in model `m` can be specified as

$$\hat{\text{fat}} = 18.89 + 0.73(\text{protein})$$

12.6 Summarizing the Fit of a Linear Model

A detailed summary of the fitted linear regression model is also available.

```
summary(m)
```

```
Call:
lm(formula = fat ~ protein, data = nnyfs)

Residuals:
    Min      1Q  Median      3Q     Max
-77.798 -14.841  -2.449  13.601 110.597

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  18.8945     1.5330   12.32   <2e-16 ***
protein       0.7251     0.0208   34.87   <2e-16 ***
---
```

	(1)
(Intercept)	18.895
	(1.533)
protein	0.725
	(0.021)
Num.Obs.	1518
R2	0.445
R2 Adj.	0.445
AIC	14 094.2
BIC	14 110.1
Log.Lik.	−7044.082
F	1215.779
RMSE	25.06

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 25.08 on 1516 degrees of freedom
Multiple R-squared:  0.4451,    Adjusted R-squared:  0.4447
F-statistic:  1216 on 1 and 1516 DF,  p-value: < 2.2e-16
```

12.6.1 Using `modelsummary()`

Another available approach to provide a summary is to use the `modelsummary()` function from the `modelsummary` package. Although this is mostly used for comparing multiple models, we can obtain good results for just one, like this.

```
modelsummary(m)
```

The modelsummary vignette provides a lot of information on how to amplify these results.

12.6.2 Plotting coefficients with `modelplot()`

Also from the `modelsummary` package, the `modelplot()` function can be of some help in understanding the size of our coefficients (and their standard errors).

```
modelplot(m) +
  labs(title = "Model m coefficients")
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	18.895	1.533	12.325	0	15.887	21.902
protein	0.725	0.021	34.868	0	0.684	0.766

Model m coefficients

Coefficient estimates and 95% confidence intervals
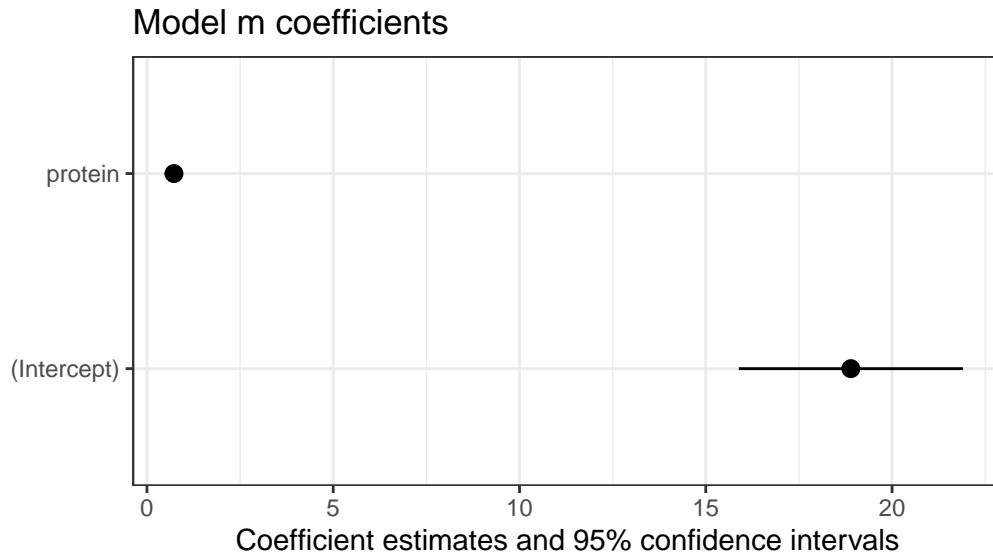
12.7 Summaries with the `broom` package

If we want to use them to do anything else, the way we'll usually summarize the estimated coefficients of a linear model is to use the `broom` package's `tidy` function to put the coefficient estimates into a tibble.

```
tidy(lm(fat ~ protein, data = nnyfs),
          conf.int = TRUE, conf.level = 0.95) |>
    kbl(digits = 3) |>
    kable_styling(full_width = FALSE)
```

We can also summarize the quality of fit in a linear model using the `broom` package's `glance` function. For now, we'll focus our attention on just one of the many summaries available for a linear model from `glance`: the R-squared value.

```
glance(lm(fat ~ protein, data = nnyfs)) |>
  select(r.squared) |>
  kbl(digits = 3) |>
  kable_styling(full_width = FALSE)
```

275

r.squared
0.445

We'll spend a lot of time working with these regression summaries in this course.

12.8 Key Takeaways from a Simple Regression

For now, it will suffice to understand the following:

- The outcome variable in this model is **fat**, and the predictor variable is **protein**.
- The straight line model for these data fitted by least squares is `fat = 18.895 + 0.725 protein`
- The slope of `protein` is positive, which indicates that as `protein` increases, we expect that `fat` will also increase. Specifically, we expect that for every additional gram of protein consumed, the fat consumption will be 0.725 gram larger.
- The multiple R-squared (squared correlation coefficient) is 0.445, which implies that 44.5% of the variation in `fat` is explained using this linear model with `protein`.
- This also implies that the Pearson correlation between `fat` and `protein` is the square root of 0.445, or 0.667. More on the Pearson correlation soon.

So, if we plan to use a simple (least squares) linear regression model to describe fat consumption as a function of protein consumption in the NNYFS data, does it look like a least squares (or linear regression) model will be an effective choice?

One way to study this is through looking at correlation: which is our next subject.

13 Correlation

13.1 Setup: Packages Used Here

```
knitr::opts_chunk$set(comment = NA)

library(janitor)
library(kableExtra)
library(tidyverse)

theme_set(theme_bw())
```

13.2 Our Data

In this chapter, we will make use of the `nnyfs` data, as well as several simulated examples stored on our 431-data page as `.csv` files.

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

correx1 <- read_csv("data/correx1.csv", show_col_types = FALSE)
correx2 <- read_csv("data/correx2.csv", show_col_types = FALSE)
correx3 <- read_csv("data/correx3.csv", show_col_types = FALSE)
```

13.3 Measuring Correlation

Two correlation measures are worth our immediate attention.

- The one most often used is called the *Pearson* correlation coefficient, and is symbolized with the letter r or sometimes the Greek letter rho (ρ).
- Another tool is the Spearman rank correlation coefficient, also occasionally symbolized by ρ.

For the `nnyfs` data, the Pearson correlation of `fat` and `protein` can be found using the `cor()` function.

```
nnyfs |>
  select(fat, protein) |>
  cor()
```

```
            fat    protein
fat     1.0000000 0.6671209
protein 0.6671209 1.0000000
```

Note that the correlation of any variable with itself is 1, and that the correlation of `fat` with `protein` is the same regardless of whether you enter `fat` first or `protein` first.

13.4 The Pearson Correlation Coefficient

Suppose we have n observations on two variables, called X and Y. The Pearson correlation coefficient assesses how well the relationship between X and Y can be described using a linear function.

- The Pearson correlation is **dimension-free**.
- It falls between -1 and +1, with the extremes corresponding to situations where all the points in a scatterplot fall exactly on a straight line with negative and positive slopes, respectively.
- A Pearson correlation of zero corresponds to the situation where there is no linear association.
- Unlike the estimated slope in a regression line, the sample correlation coefficient is symmetric in X and Y, so it does not depend on labeling one of them (Y) the response variable, and one of them (X) the predictor.

Suppose we have n observations on two variables, called X and Y, where $\bar{X}$ is the sample mean of X and s_x is the standard deviation of X. The **Pearson** correlation coefficient r_{XY} is:

$$r_{XY} = \frac{1}{n-1} \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{s_x}\right)\left(\frac{y_i - \bar{y}}{s_y}\right)$$

13.5 Studying Correlation through Six Examples

The `correx1` data file we read in at the start of this Chapter contains six different sets of (x,y) points, identified by the `set` variable.

```
summary(correx1)
```

```
     set                   x                 y
 Length:277        Min.   : 5.897    Min.   : 7.308
 Class :character  1st Qu.:29.487    1st Qu.:30.385
 Mode  :character  Median :46.154    Median :46.923
                   Mean   :46.529    Mean   :49.061
                   3rd Qu.:63.333    3rd Qu.:68.077
                   Max.   :98.205    Max.   :95.385
```

13.5.1 Data Set Alex

Let's start by working with the **Alex** data set.

```
ggplot(filter(correx1, set == "Alex"), aes(x = x, y = y)) +
    geom_point() +
    labs(title = "correx1: Data Set Alex")
```

correx1: Data Set Alex
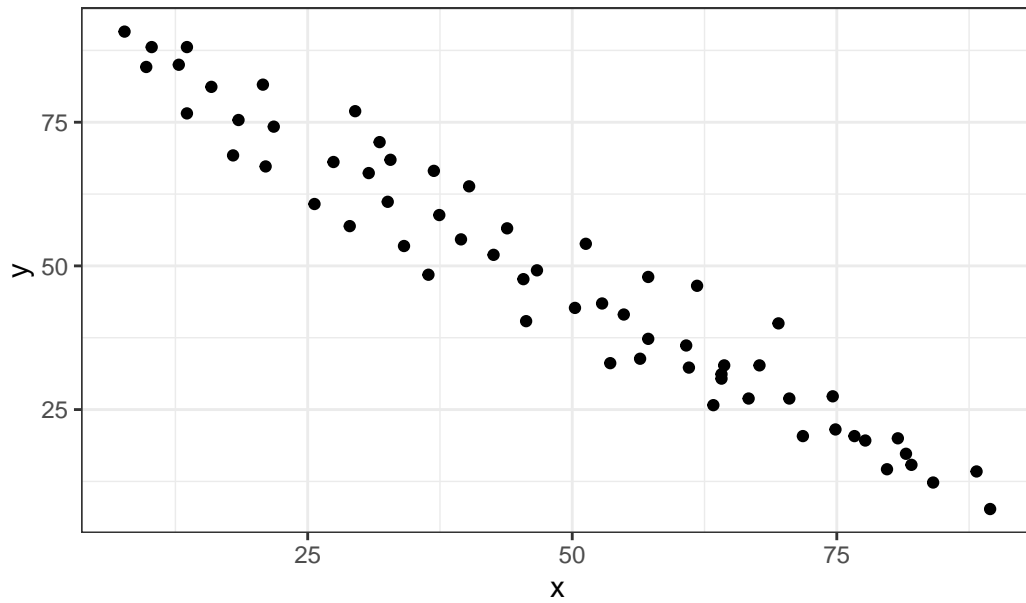

```
ggplot(filter(correx1, set == "Alex"), aes(x = x, y = y)) +
    geom_point() +
    geom_smooth(method = "loess", formula = y ~ x, col = "blue") +
    labs(title = "correx1: Alex, with loess smooth")
```

correx1: Alex, with loess smooth
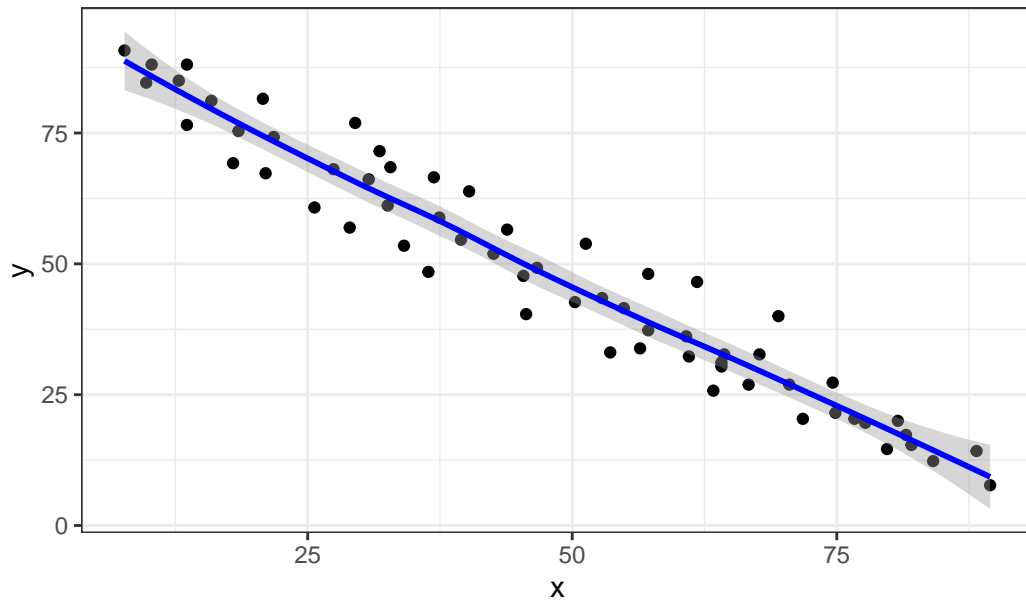

```
setA <- filter(correx1, set == "Alex")

ggplot(setA, aes(x = x, y = y)) +
    geom_point() +
    geom_smooth(method = "lm", formula = y ~ x, col = "red") +
    labs(title = "correx1: Alex, with Fitted Linear Model") +
    annotate("text", x = 75, y = 75, col = "purple", size = 6,
             label = paste("Pearson r = ", round_half_up(cor(setA$x, setA$y),3))) +
    annotate("text", x = 50, y = 15,  col = "red", size = 5,
             label = paste("y = ", round_half_up(coef(lm(setA$y ~ setA$x))[1],3),
                           round_half_up(coef(lm(setA$y ~ setA$x))[2],2), "x"))
```

correx1: Alex, with Fitted Linear Model

Pearson r = −0.97

y = 94.205 −0.96 x
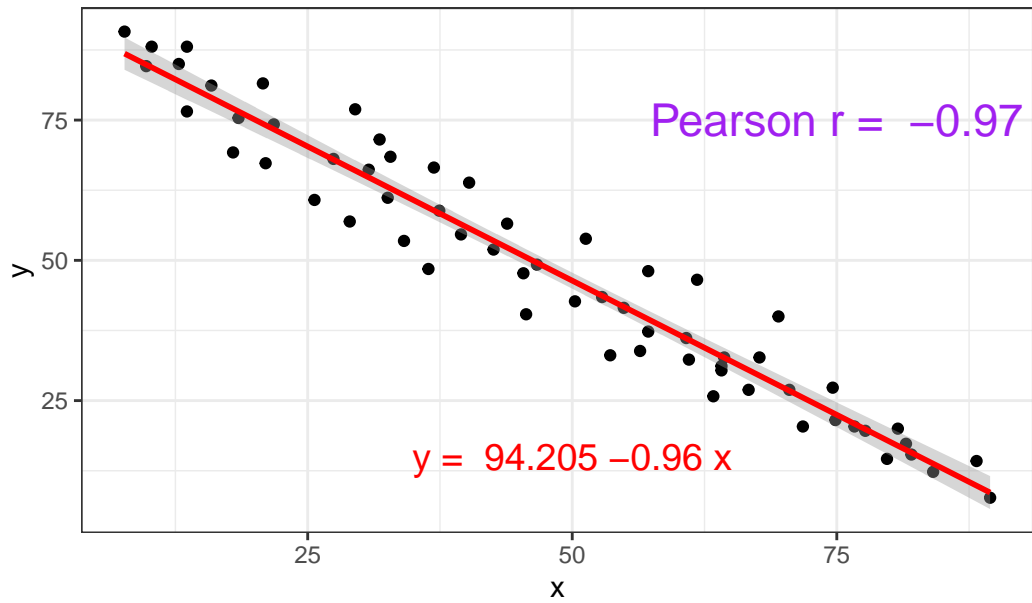
13.5.2 Data Set Bonnie

```
setB <- filter(correx1, set == "Bonnie")

ggplot(setB, aes(x = x, y = y)) +
    geom_point() +
    labs(title = "correx1: Data Set Bonnie")
```

correx1: Data Set Bonnie
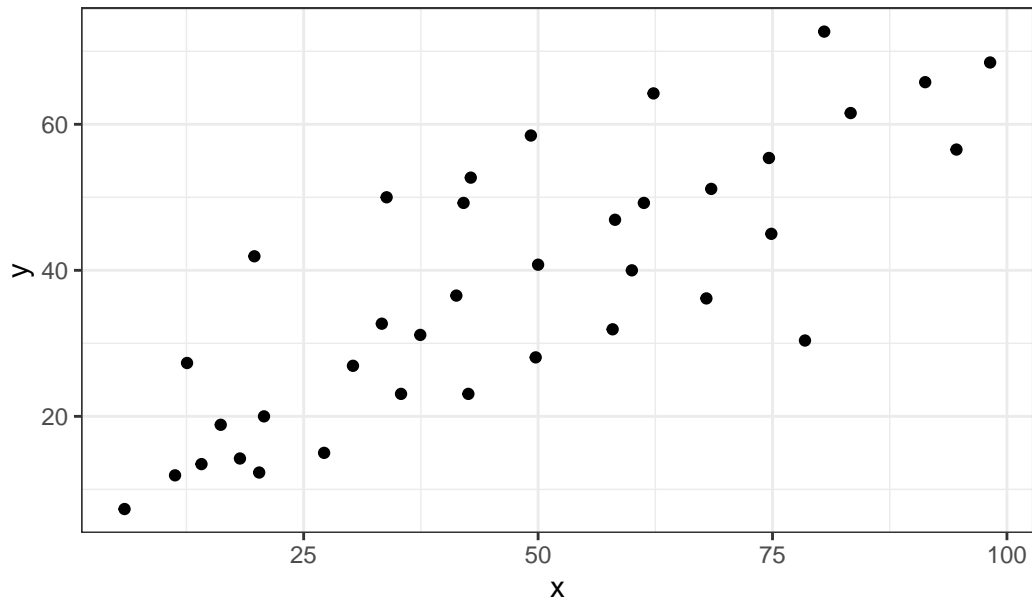

```
ggplot(setB, aes(x = x, y = y)) +
    geom_point() +
    geom_smooth(method = "lm", formula = y ~ x, col = "red") +
    labs(title = "correx1: Bonnie, with Fitted Linear Model") +
    annotate("text", x = 25, y = 65, col = "purple", size = 6,
             label = paste("Pearson r = ", round_half_up(cor(setB$x, setB$y),2))) +
    annotate("text", x = 50, y = 15,  col = "red", size = 5,
             label = paste("y = ", round_half_up(coef(lm(setB$y ~ setB$x))[1],3),
                           " + ",
                           round_half_up(coef(lm(setB$y ~ setB$x))[2],2), "x"))
```

set	Pearson r
Alex	-0.97
Bonnie	0.80
Colin	-0.80
Danielle	0.00
Earl	-0.01
Fiona	0.00

correx1: Bonnie, with Fitted Linear Model
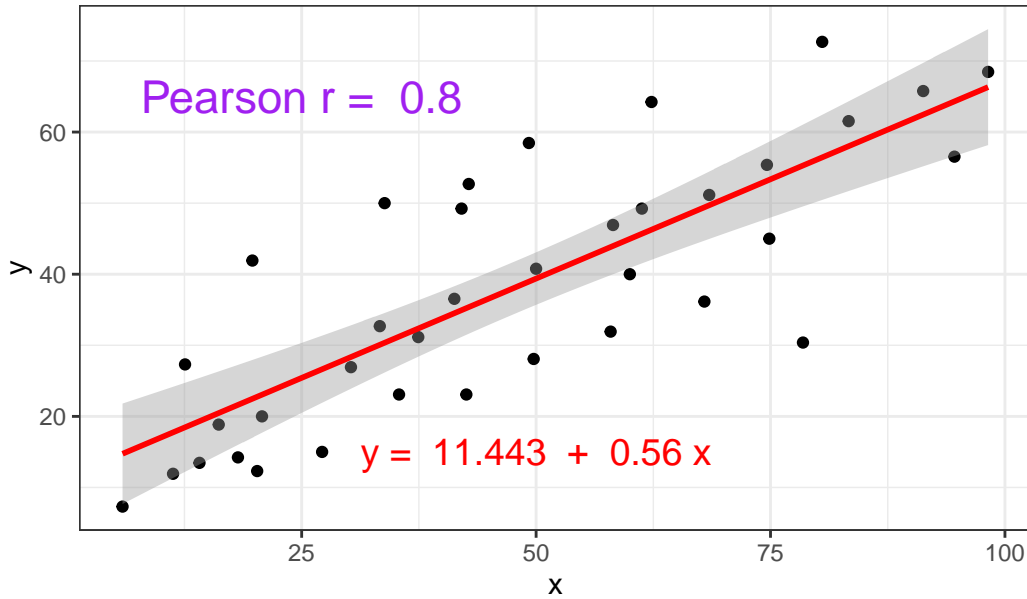
13.5.3 Correlations for All Six Data Sets in `correx1`

Let's look at the Pearson correlations associated with each of the six data sets contained in the `correx1` example.

```
tab1 <- correx1 |>
    group_by(set) |>
    summarise("Pearson r" = round_half_up(cor(x, y, use="complete"),2))

tab1 |>
  kbl() |>
  kable_styling(full_width = FALSE)
```

13.5.4 Data Set Colin

It looks like the picture for Colin should be very similar (in terms of scatter) to the picture for Bonnie, except that Colin will have a negative slope, rather than the positive one Bonnie has. Is that how this plays out?

```
setBC <- filter(correx1, set == "Bonnie" | set == "Colin")

ggplot(setBC, aes(x = x, y = y)) +
    geom_point() +
    geom_smooth(method = "lm", formula = y ~ x, col = "red") +
    facet_wrap(~ set)
```
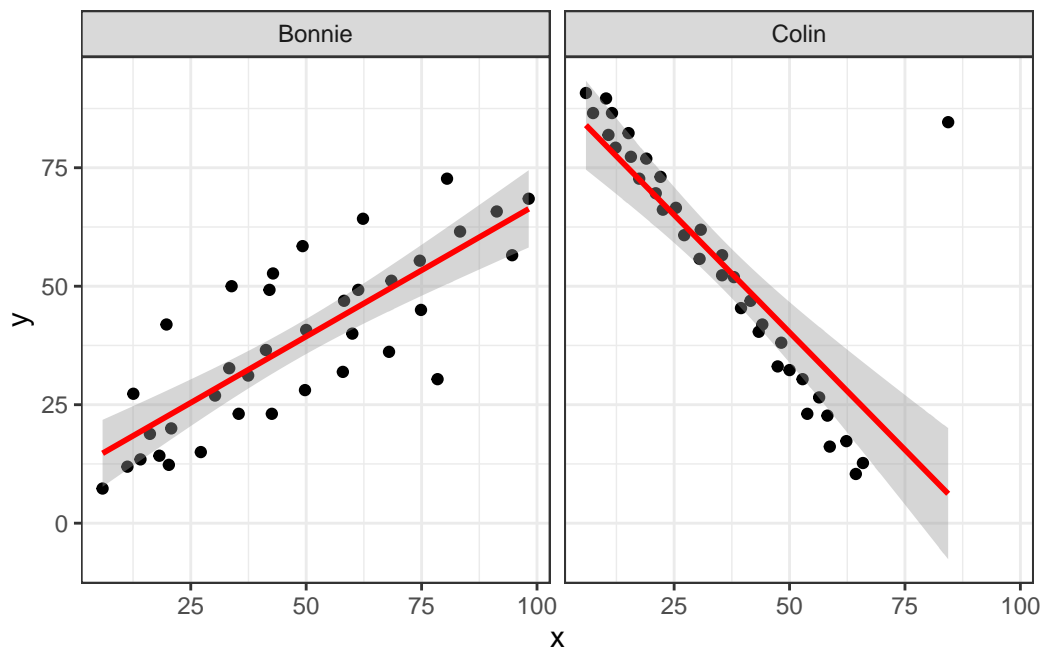

Uh, oh. It looks like the point in Colin at the top right is twisting what would otherwise be a very straight regression model with an extremely strong negative correlation. There's no better way to look for outliers than to examine the scatterplot.

13.5.5 Draw the Picture!

We've seen that Danielle, Earl and Fiona all show Pearson correlations of essentially zero. However, the three data sets look very different in a scatterplot.

```
ggplot(correx1, aes(x = x, y = y)) +
    geom_point() +
    geom_smooth(method = "loess", formula = y ~ x) +
    facet_wrap(~ set)
```
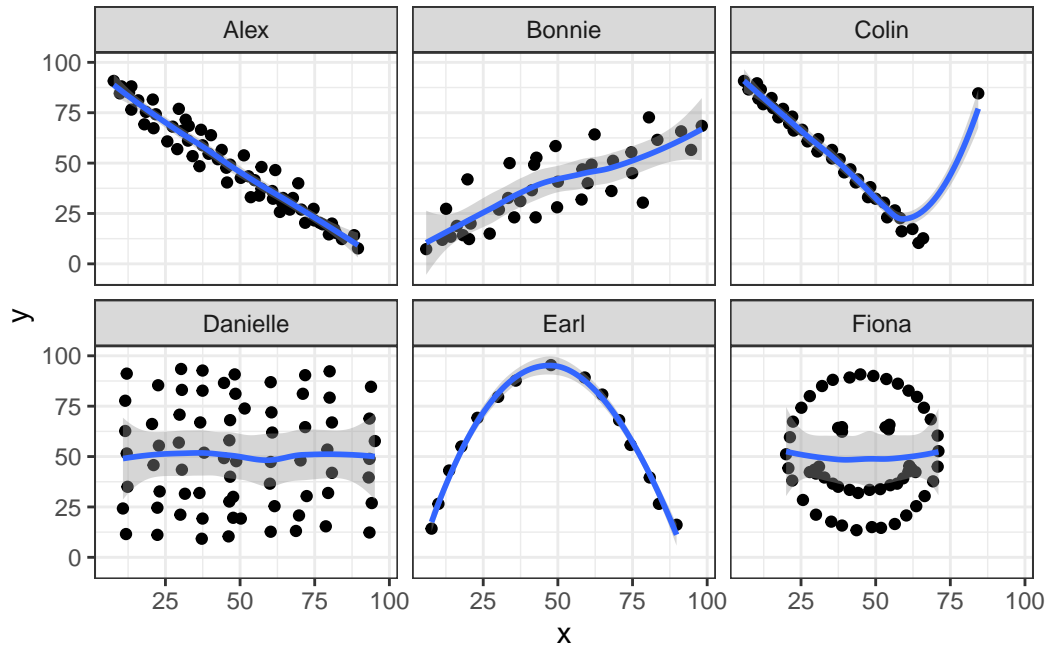

When we learn that the correlation is zero, we tend to assume we have a picture like the Danielle data set. If Danielle were our real data, we might well think that x would be of little use in predicting y.

- But what if our data looked like Earl? In the Earl data set, x is incredibly helpful in predicting y, but we can't use a straight line model - instead, we need a non-linear modeling approach.
- You'll recall that the Fiona data set also had a Pearson correlation of zero. But here, the picture is rather more interesting.

So, remember, draw the appropriate scatterplot whenever you make use of a correlation coefficient.

13.6 Estimating Correlation from Scatterplots

The correx2 data set is designed to help you calibrate yourself a bit in terms of estimating a correlation from a scatterplot. There are 11 data sets buried within the correx2 example, and they are labeled by their Pearson correlation coefficients, ranging from r = 0.01 to r = 0.999

```
correx2 |>
    group_by(set) |>
    summarise(cor = round(cor(x, y, use="complete"),3))
```

```
# A tibble: 11 x 2
   set       cor
   <chr>    <dbl>
 1 Set 01   0.01
 2 Set 10   0.102
 3 Set 20   0.202
 4 Set 30   0.301
 5 Set 40   0.403
 6 Set 50   0.499
 7 Set 60   0.603
 8 Set 70   0.702
 9 Set 80   0.799
10 Set 90   0.902
11 Set 999 0.999
```

Here is a plot of the 11 data sets, showing the increase in correlation from 0.01 (in Set 01) to 0.999 (in Set 999).

```
ggplot(correx2, aes(x = x, y = y)) +
    geom_point() +
    facet_wrap(~ set) +
    labs(title = "Pearson Correlations from 0.01 to 0.999")
```

Pearson Correlations from 0.01 to 0.999
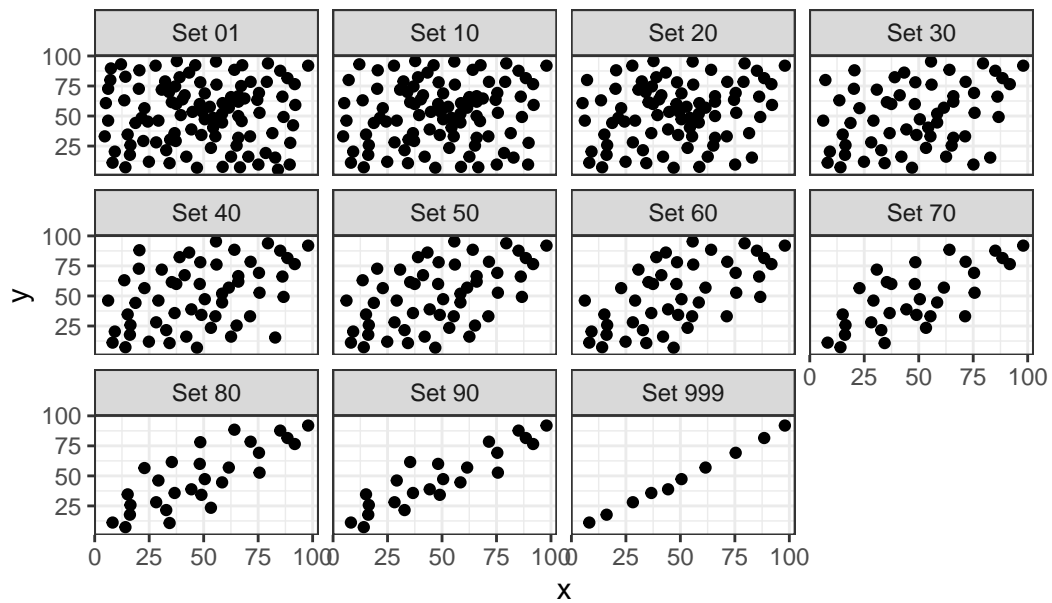
Note that R will allow you to fit a straight line model to any of these relationships, no matter how appropriate it might be to do so.

```
ggplot(correx2, aes(x = x, y = y)) +
    geom_point() +
    geom_smooth(method = "lm", formula = y ~ x, col = "red") +
    facet_wrap(~ set) +
    labs(title = "R will fit a straight line to anything.")
```

R will fit a straight line to anything.
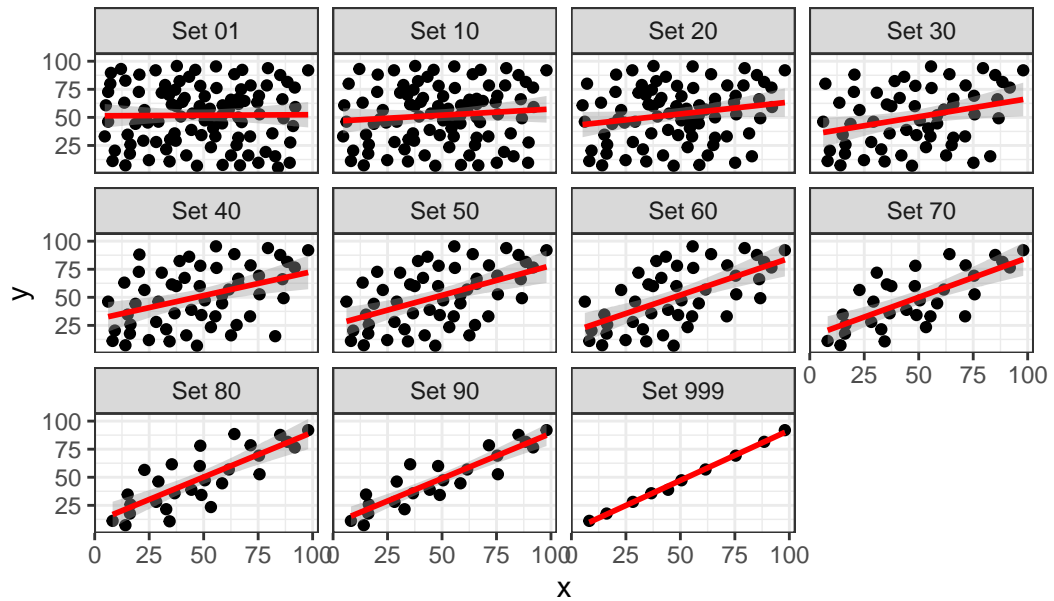

```r
ggplot(correx2, aes(x = x, y = y)) +
    geom_point() +
    geom_smooth(col = "blue") +
    facet_wrap(~ set) +
    labs(title = "Even if a loess smooth suggests non-linearity.")
```

Even if a loess smooth suggests non−linearity.
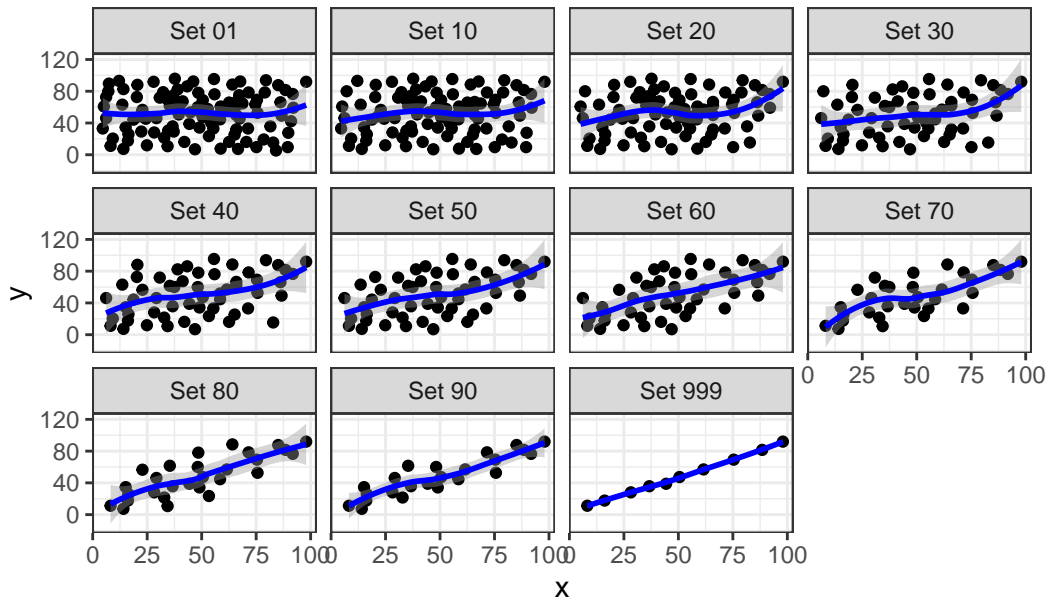

```
ggplot(correx2, aes(x = x, y = y, color = factor(group))) +
    geom_point() +
    guides(color = "none") +
    facet_wrap(~ set) +
    labs(title = "Note: The same 10 points (in red) are in each plot.")
```

Note: The same 10 points (in red) are in each plot.
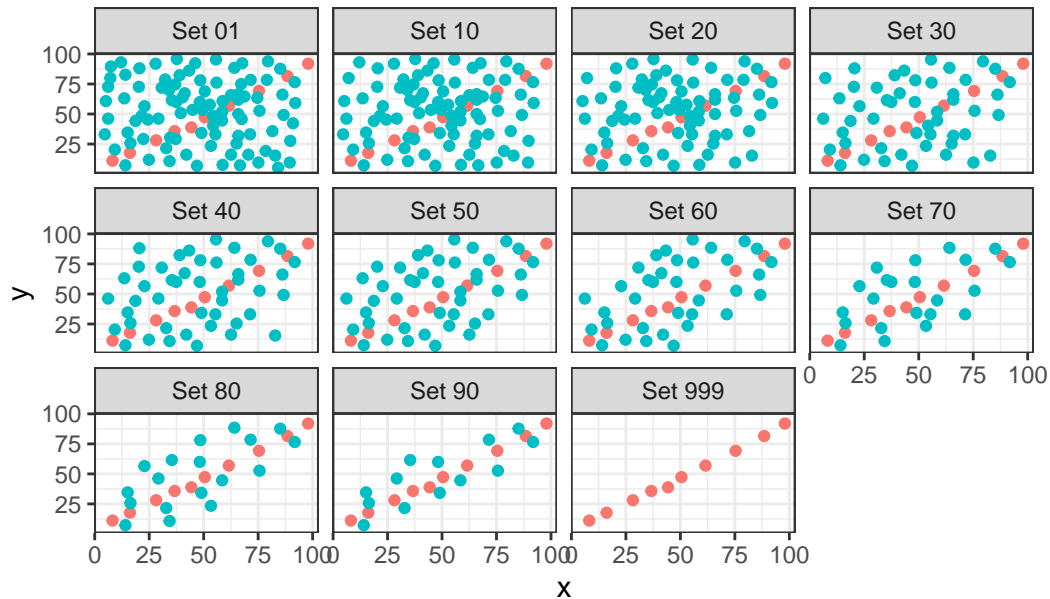
Note that the same 10 points are used in each of the data sets. It's always possible that a lurking subgroup of the data within a scatterplot follows a very strong linear relationship. This is why it's so important (and difficult) not to go searching for such a thing without a strong foundation of logic, theory and prior empirical evidence.

13.7 The Spearman Rank Correlation

The Spearman rank correlation coefficient is a rank-based measure of statistical dependence that assesses how well the relationship between X and Y can be described using a **monotone function** even if that relationship is not linear.

- A monotone function preserves order, that is, Y must either be strictly increasing as X increases, or strictly decreasing as X increases.
- A Spearman correlation of 1.0 indicates simply that as X increases, Y always increases.
- Like the Pearson correlation, the Spearman correlation is dimension-free, and falls between -1 and +1.
- A positive Spearman correlation corresponds to an increasing (but not necessarily linear) association between X and Y, while a negative Spearman correlation corresponds to a decreasing (but again not necessarily linear) association.

13.7.1 Spearman Formula

To calculate the Spearman rank correlation, we take the ranks of the X and Y data, and then apply the usual Pearson correlation. To find the ranks, sort X and Y into ascending order, and then number them from 1 (smallest) to n (largest). In the event of a tie, assign the average rank to the tied subjects.

13.7.2 Comparing Pearson and Spearman Correlations

Let's look at the `nnyfs` data again.

```
nnyfs |> select(fat, protein) |> cor()
```

```
              fat    protein
fat     1.0000000 0.6671209
protein 0.6671209 1.0000000
```

```
cor_sp <- nnyfs |>
  select(fat, protein)

cor_sp |> cor(method = "spearman")
```

```
              fat    protein
fat     1.0000000 0.6577489
protein 0.6577489 1.0000000
```

The Spearman and Pearson correlations are not especially different in this case.

13.7.3 Spearman vs. Pearson Example 1

The next few plots describe relationships where we anticipate the Pearson and Spearman correlations might differ in their conclusions.

Example 1 shows a function where the Pearson correlation is 0.925 (a strong but not perfect linear relation), but the Spearman correlation is 1 because the relationship is monotone, even though it is not perfectly linear.

```
ggplot(correx3, aes(x = x1, y = y1)) +
    geom_point() +
```

```
        labs(title = "Spearman vs. Pearson, Example 1") +
        annotate("text", x = -10, y = 20,
                 label = paste("Pearson r = ",
                     round_half_up(cor(correx3$x1, correx3$y1,
                                      use = "complete.obs"),3),
                     ", Spearman r = ",
                     round_half_up(cor(correx3$x1, correx3$y1, method = "spearman",
                                      use = "complete.obs"),3)))
```

Spearman vs. Pearson, Example 1
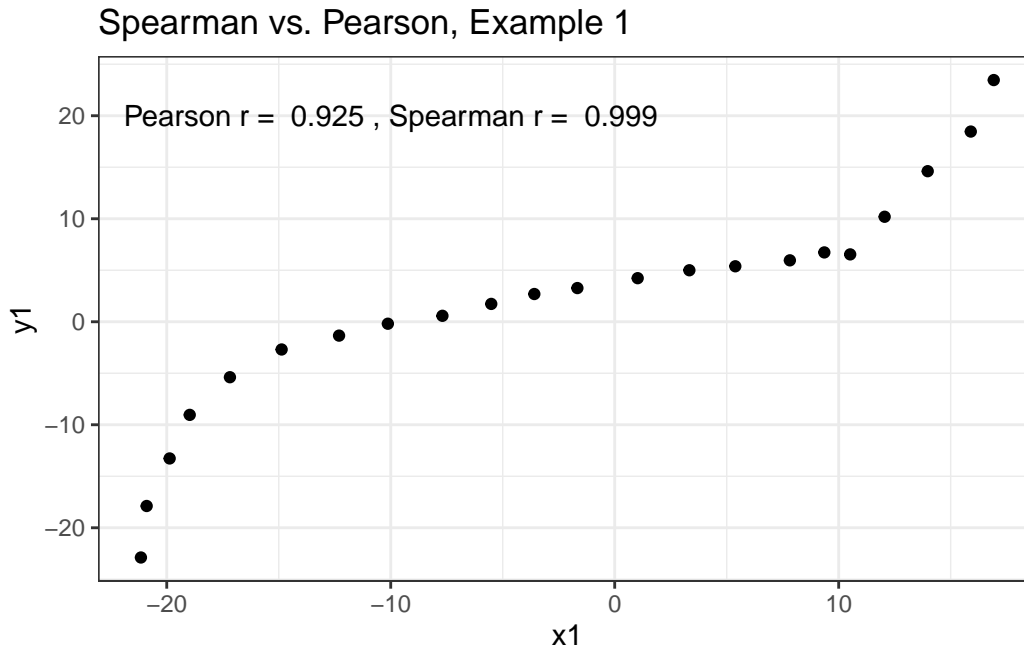
So, a positive Spearman correlation corresponds to an increasing (but not necessarily linear) association between x and y.

13.7.4 Spearman vs. Pearson Example 2

Example 2 shows that a negative Spearman correlation corresponds to a decreasing (but, again, not necessarily linear) association between x and y.

```
ggplot(correx3, aes(x = x2, y = y2)) +
    geom_point(col = "purple") +
    geom_smooth(method = "loess", formula = y ~ x, se = FALSE) +
```

```
        labs(title = "Spearman vs. Pearson, Example 2") +
        annotate("text", x = 0, y = 20, col = "purple",
                 label = paste("Pearson r = ",
                      round_half_up(cor(correx3$x2, correx3$y2,
                                        use = "complete.obs"),2),
                      ", Spearman r = ",
                      round_half_up(cor(correx3$x2, correx3$y2, method = "spearman",
                                        use = "complete.obs"),2)))
```

Spearman vs. Pearson, Example 2
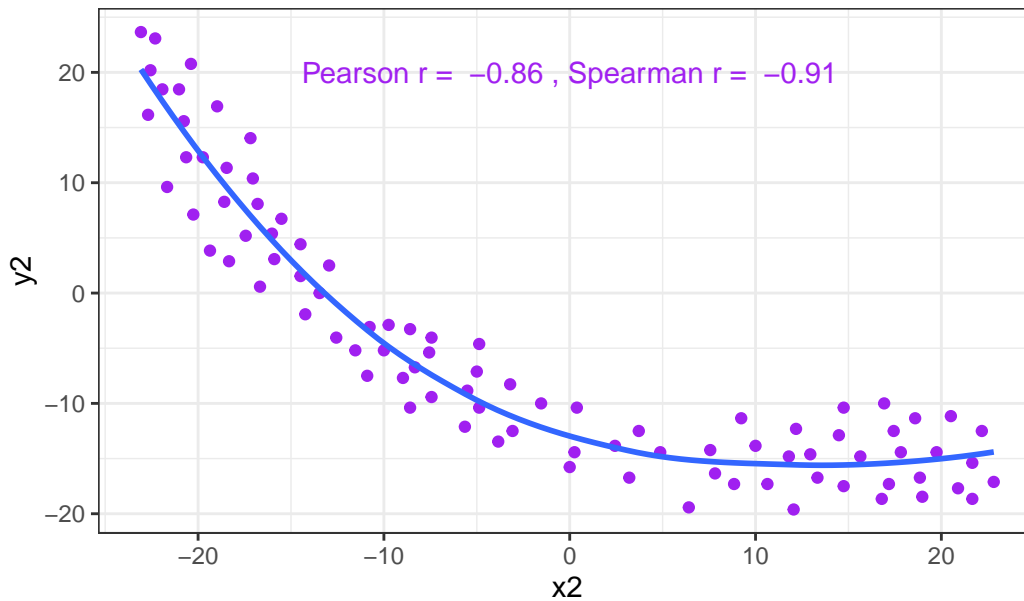
13.7.5 Spearman vs. Pearson Example 3

The Spearman correlation is less sensitive than the Pearson correlation is to strong outliers that are unusual on either the X or Y axis, or both. That is because the Spearman rank coefficient limits the outlier to the value of its rank.

In Example 3, for instance, the Spearman correlation reacts much less to the outliers around X = 12 than does the Pearson correlation.

```
ggplot(correx3, aes(x = x3, y = y3)) +
    geom_point(col = "blue") +
    labs(title = "Spearman vs. Pearson, Example 3") +
```

```
      annotate("text", x = 5, y = -15, col = "blue",
             label = paste("Pearson r = ",
                  round_half_up(cor(correx3$x3, correx3$y3,
                                   use = "complete.obs"),2),
                  ", Spearman r = ",
                  round_half_up(cor(correx3$x3, correx3$y3, method = "spearman",
                                   use = "complete.obs"),2)))
```

Spearman vs. Pearson, Example 3
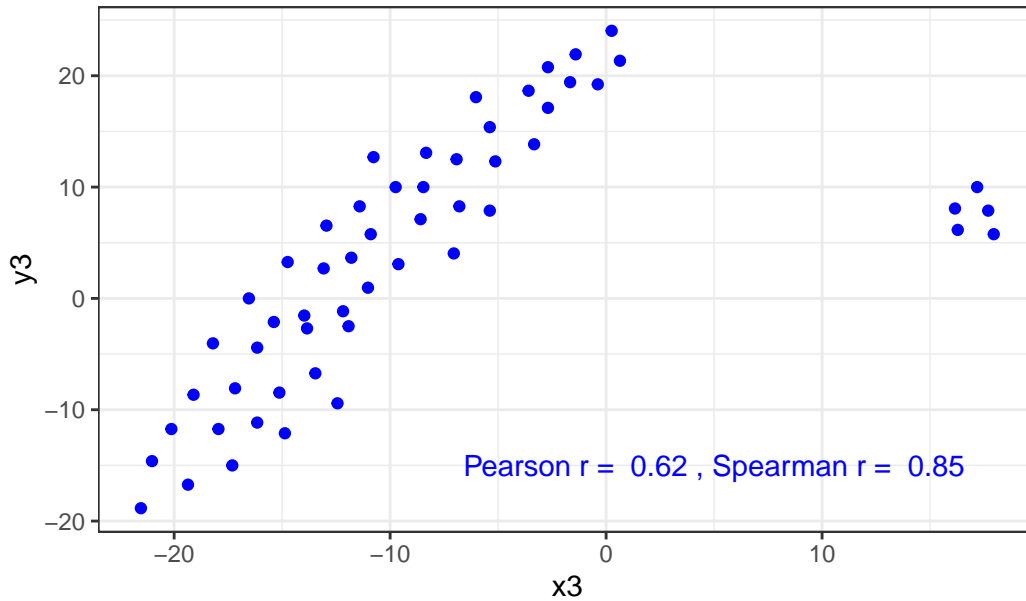
13.7.6 Spearman vs. Pearson Example 4

The use of a Spearman correlation is no substitute for looking at the data. For non-monotone data like what we see in Example 4, neither the Spearman nor the Pearson correlation alone provides much guidance, and just because they are (essentially) telling you the same thing, that doesn't mean what they're telling you is all that helpful.

```
ggplot(correx3, aes(x = x4, y = y4)) +
    geom_point(col = "purple") +
    geom_smooth(method = "loess", formula = y ~ x, se = FALSE) +
    labs(title = "Spearman vs. Pearson, Example 4") +
    annotate("text", x = 10, y = 20, col = "purple",
             label = paste("Pearson r = ",
```

```
                    round_half_up(cor(correx3$x4, correx3$y4,
                                      use = "complete.obs"),2),
                ", Spearman r = ",
                round_half_up(cor(correx3$x4, correx3$y4, method = "spearman",
                                  use = "complete.obs"),2)))
```

Spearman vs. Pearson, Example 4
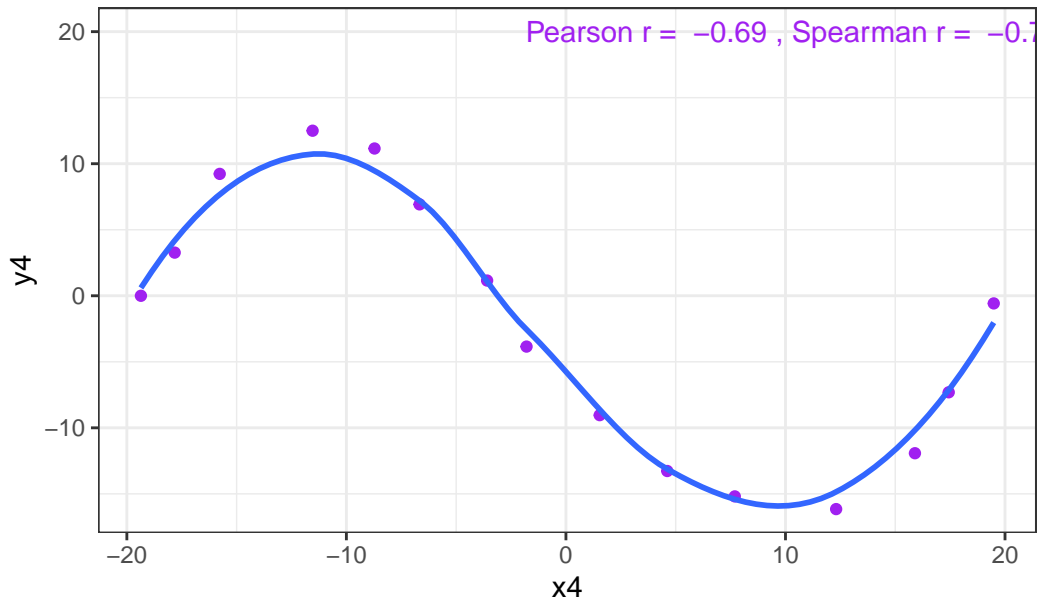
13.8 Coming Up

Next, we'll look at some options for improving the fit of a linear model to a scatterplot that shows a strong, but not a linear association.

14 Linearizing Transformations

14.1 Linearize The Association between Quantitative Variables

Confronted with a scatterplot describing a monotone association between two quantitative variables, we may decide the data are not well approximated by a straight line, and thus, that a least squares regression may not be not sufficiently useful. In these circumstances, we have at least two options, which are not mutually exclusive:

a. Let the data be as they may, and summarize the scatterplot using tools like loess curves, polynomial functions, or cubic splines to model the relationship.
b. Consider re-expressing the data (often we start with re-expressions of the outcome data [the Y variable]) using a transformation so that the transformed data may be modeled effectively using a straight line.

14.2 Setup: Packages Used Here

```r
knitr::opts_chunk$set(comment = NA)

library(broom)
library(car)
library(patchwork)
library(tidyverse)

theme_set(theme_bw())
```

14.3 The Box-Cox Plot

As before, 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**, from the `boxCox` function in the `car` package, sifts through the ladder of options to suggest a transformation (for Y) to best linearize the outcome-predictor(s) relationship.

14.3.1 A Few Caveats

1. These methods work well with *monotone* data, where a smooth function of Y is either strictly increasing, or strictly decreasing, as X increases.
2. Some of these transformations 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.
3. 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).
4. 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.
5. 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.

14.4 A Simulated Example

```
set.seed(999);
  x.rand <- rbeta(80, 2, 5) * 20 + 3
set.seed(1000);
  y.rand <- abs(50 + 0.75*x.rand^(3)
               - 0.65*x.rand + rnorm(80, 0, 200))

scatter1 <- tibble(x = x.rand, y = y.rand)
rm(x.rand, y.rand)

ggplot(scatter1, aes(x = x, y = y)) +
    geom_point(shape = 1, size = 3) +
    ## add loess smooth
    geom_smooth(method = "loess", se = FALSE,
                col = "dodgerblue", formula = y ~ x) +
    ## then add linear fit
    geom_smooth(method = "lm", se = FALSE,
                col = "red", formula = y ~ x, linetype = "dashed") +
```

```
labs(title = "Simulated scatter1 example: Y vs. X")
```

Simulated scatter1 example: Y vs. X
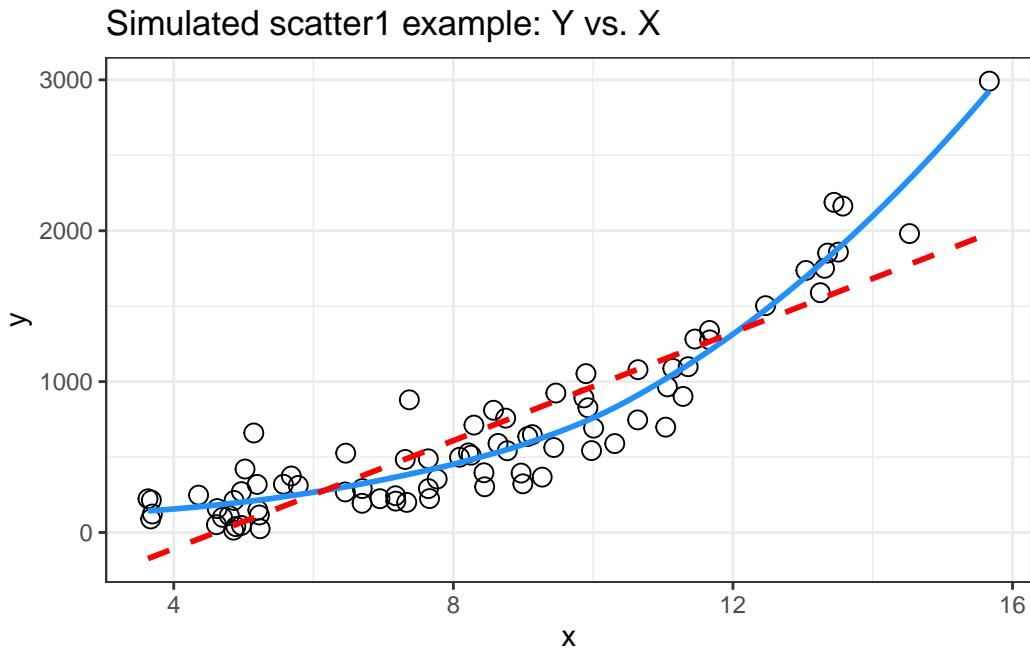
Having simulated data that produces a curved scatterplot, I will now use the Box-Cox plot to lead my choice of an appropriate power transformation for Y in order to "linearize" the association of Y and X.

```
boxCox(scatter1$y ~ scatter1$x)
```

Profile Log–likelihood
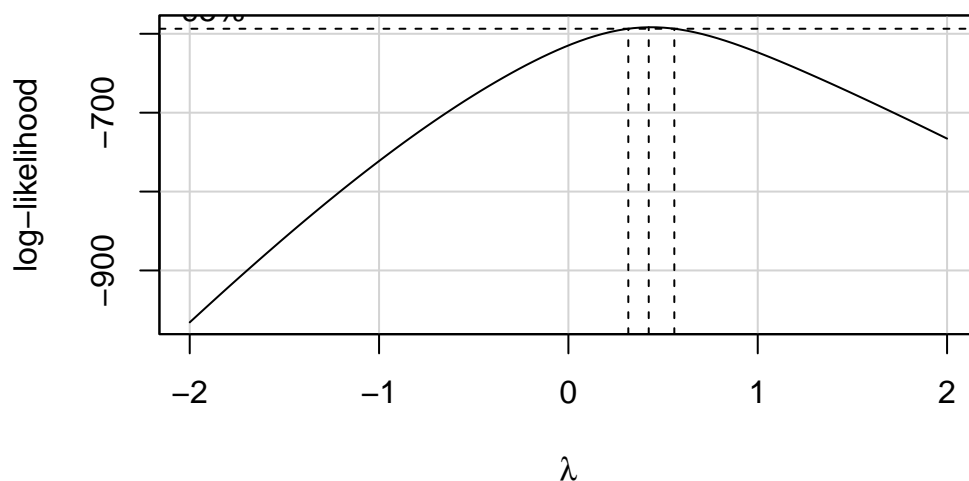

```
powerTransform(scatter1$y ~ scatter1$x)
```

```
Estimated transformation parameter
       Y1
0.4368753
```

The Box-Cox plot peaks at the value $\lambda = 0.44$, which is pretty close to $\lambda = 0.5$. Now, 0.44 isn't on Tukey's ladder, but 0.5 is.

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

If we use $\lambda = 0.5$, on Tukey's ladder of power transformations, it suggests we look at the relationship between the square root of Y and X, as shown next.

```
p1 <- ggplot(scatter1, aes(x = x, y = y)) +
    geom_point(size = 2) +
    geom_smooth(method = "loess", se = FALSE,
                formula = y ~ x, col = "dodgerblue") +
    geom_smooth(method = "lm", se = FALSE,
```

```
               formula = y ~ x, col = "red", linetype = "dashed") +
   labs(title = "scatter1: Y vs. X")

p2 <- ggplot(scatter1, aes(x = x, y = sqrt(y))) +
   geom_point(size = 2) +
   geom_smooth(method = "loess", se = FALSE,
               formula = y ~ x, col = "dodgerblue") +
   geom_smooth(method = "lm", se = FALSE,
               formula = y ~ x, col = "red", linetype = "dashed") +
   labs(title = "scatter1: Square Root of Y vs. X")

p1 + p2
```
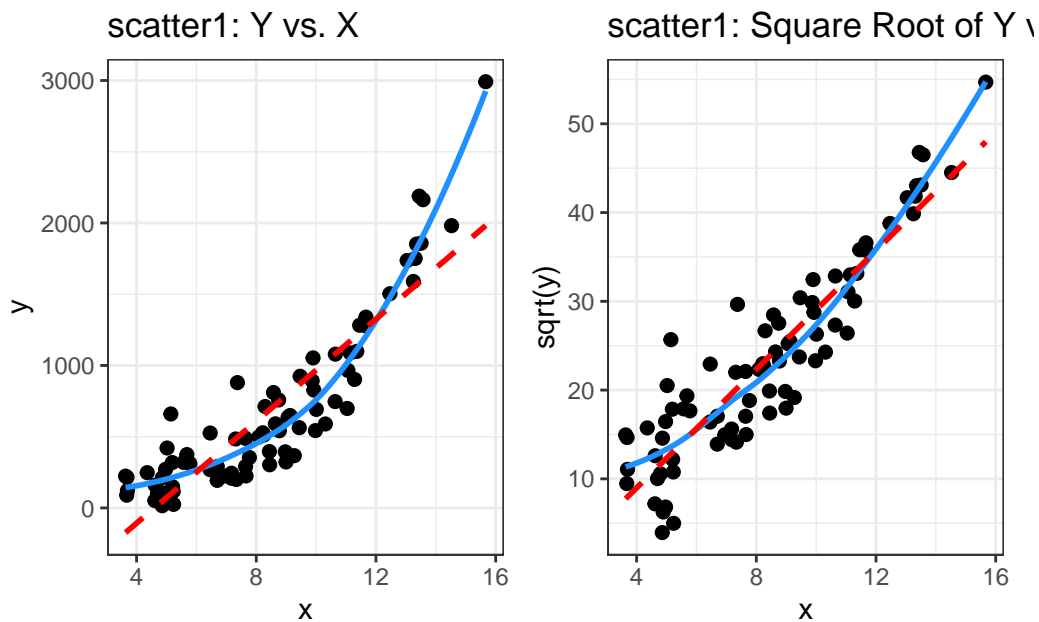

By eye, I think the square root plot better matches the linear fit.

14.5 Checking on a Transformation or Re-Expression

In addition to plotting the impact of the transformation, we can do at least two other things to check on our transformation.

- We can calculate the correlation of our original and re-expressed associations.

- We can go ahead and fit the regression models using each approach and compare the plots of studentized residuals against fitted values from the data to see if the re-expression reduces the curve in that residual plot, as well.

The last of these options is by far the most important in practice, and it's the one we'll focus on going forward, but we'll demonstrate both of these new approaches here.

14.5.1 Checking the Correlation Coefficients

Here, we calculate the correlation of original and re-expressed associations.

```
cor(scatter1$y, scatter1$x)
```

```
[1] 0.891198
```

```
cor(sqrt(scatter1$y), scatter1$x)
```

```
[1] 0.9144307
```

The Pearson correlation is a little stronger after the transformation. as we'd expect.

14.5.2 Comparing the Residual Plots

We can fit the regression models, obtain plots of residuals against fitted values, and compare them to see which one has less indication of a curve in the residuals.

```
model.orig <- lm(scatter1$y ~ scatter1$x)
model.sqrt <- lm(sqrt(scatter1$y) ~ scatter1$x)

p1 <- ggplot(augment(model.orig), aes(x = scatter1$x, y = .resid)) +
    geom_point() +
    geom_smooth(method = "loess", formula = y ~ x, se = FALSE) +
    geom_smooth(method = "lm", formula = y ~ x, se = FALSE, col = "red") +
    labs(title = "Y vs X Residual Plot")

p2 <- ggplot(augment(model.sqrt), aes(x = scatter1$x, y = .resid)) +
    geom_point() +
    geom_smooth(method = "loess", formula = y ~ x, se = FALSE) +
    geom_smooth(method = "lm", formula = y ~ x, se = FALSE, col = "red") +
```

```
    labs(title = "Square Root Model Residuals")

p1 + p2
```
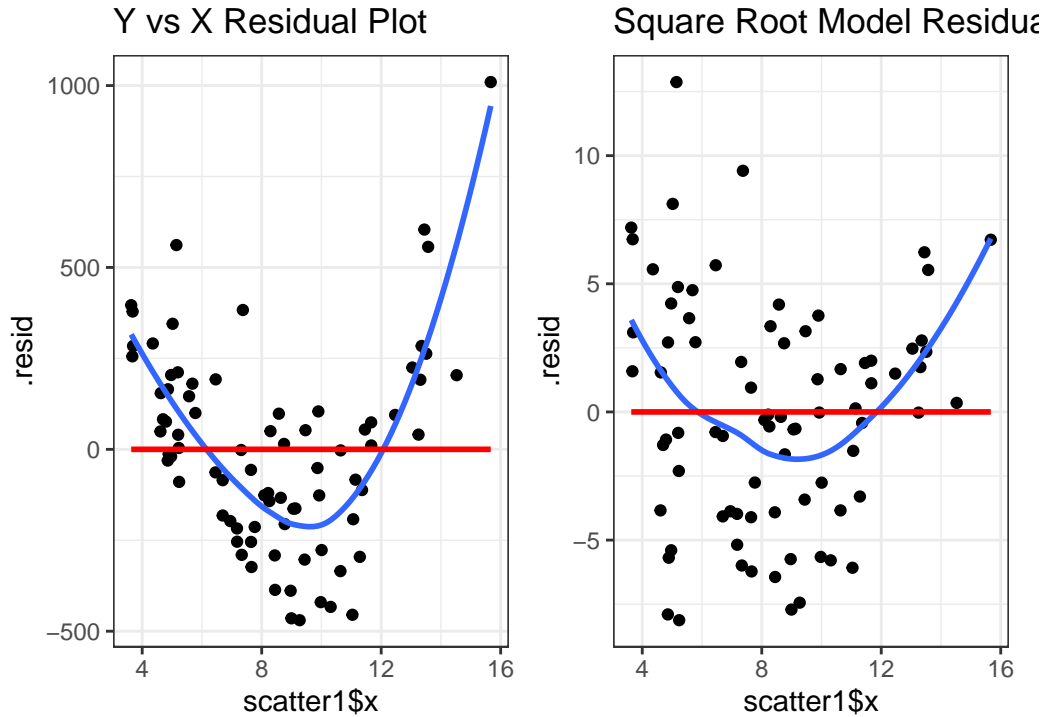

What we're looking for in such a plot is the absence of a curve, among other things, we want to see "fuzzy football" shapes.

As compared to the original residual plot, the square root version, is a modest improvement in this regard. It does look a bit less curved, and a bit more like a random cluster of points, so that's nice. Usually, we can do a little better in real data, as shown in the next example from the NNYFS data we introduced in Chapter 10.

14.6 An Example from the NNYFS data

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

Using the subjects in the **nnyfs** data with complete data on the two variables of interest, let's look at the relationship between arm circumference (the outcome, shown on the Y axis) and arm length (the predictor, shown on the X axis.)

```
nnyfs_c <- nnyfs |>
    filter(complete.cases(arm_circ, arm_length)) |>
    select(SEQN, arm_circ, arm_length)
```

14.6.1 Pearson correlation and scatterplot

Here is the Pearson correlation between these two variables.

```
nnyfs_c |> select(arm_length, arm_circ) |> cor()
```

```
          arm_length  arm_circ
arm_length  1.0000000 0.8120242
arm_circ    0.8120242 1.0000000
```

Here's the resulting scatterplot.

```
ggplot(nnyfs_c, aes(x = arm_length, y = arm_circ)) +
    geom_point(alpha = 0.2) +
    geom_smooth(method = "loess", formula = y ~ x,
                se = FALSE, color = "blue") +
    geom_smooth(method = "lm", formula = y ~ x,
                se = FALSE, color = "red")
```
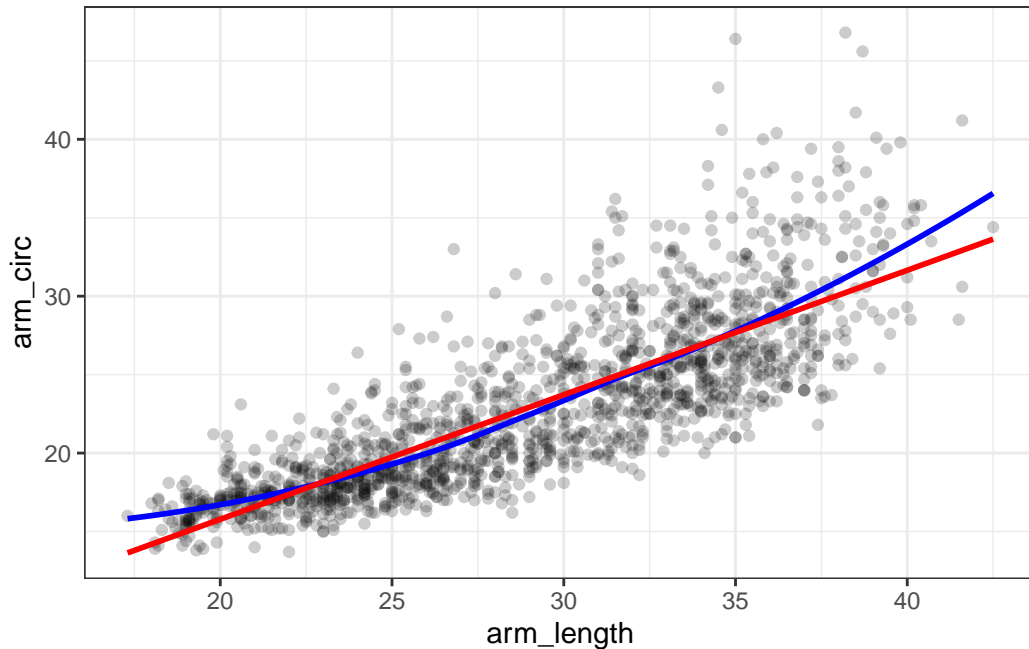
While the Pearson correlation is still quite strong, note that the loess smooth (shown in blue) bends up from the straight line model (shown in red) at both the low and high end of arm length.

Note also the use of `alpha = 0.2` to show the points with greater transparency than they would be shown normally (the default setting is no transparency with `alpha = 1`.)

14.6.2 Plotting the Residuals

Now, let's build a plot of residuals from the straight line model plotted against the arm length. We can obtain these residuals using the `augment()` function from the `broom` package.

```
m1 <- lm(arm_circ ~ arm_length, data = nnyfs_c)

nnyfs_c_aug1 <- augment(m1, data = nnyfs_c)

nnyfs_c_aug1
```

```
# A tibble: 1,511 x 9
    SEQN arm_circ arm_length .fitted .resid    .hat .sigma   .cooksd .std.resid
   <dbl>    <dbl>      <dbl>   <dbl>  <dbl> <dbl>   <dbl> <dbl>     <dbl>      <dbl>
```

```
 1 71918      25.4          27.7        21.9  3.51  0.000695    3.21 0.000416         1.09
 2 71919      26            38.4        30.4 -4.38  0.00253     3.21 0.00237         -1.37
 3 71920      37.9          35.9        28.4  9.50  0.00167     3.20 0.00735          2.96
 4 71921      15.1          18.3        14.4  0.669 0.00304     3.21 0.0000663        0.209
 5 71922      29.5          34.2        27.0  2.45  0.00124     3.21 0.000362         0.764
 6 71923      27.9          33          26.1  1.80  0.00100     3.21 0.000159         0.562
 7 71924      17.6          26.5        20.9 -3.34  0.000788    3.21 0.000427        -1.04
 8 71925      17.7          24.2        19.1 -1.41  0.00113     3.21 0.000110        -0.441
 9 71926      19.9          26          20.5 -0.642 0.000844    3.21 0.0000169       -0.200
10 71927      17.3          20          15.8  1.52  0.00234     3.21 0.000263         0.474
# i 1,501 more rows
```

OK. So the residuals are now stored in the `.resid` variable. We can create a residual plot, as
follows.

```
ggplot(nnyfs_c_aug1, aes(x = arm_length, y = .resid)) +
    geom_point(alpha = 0.2) +
    geom_smooth(method = "loess", col = "purple",
                formula = y ~ x, se = FALSE) +
    labs(title = "Residuals show a curve.")
```
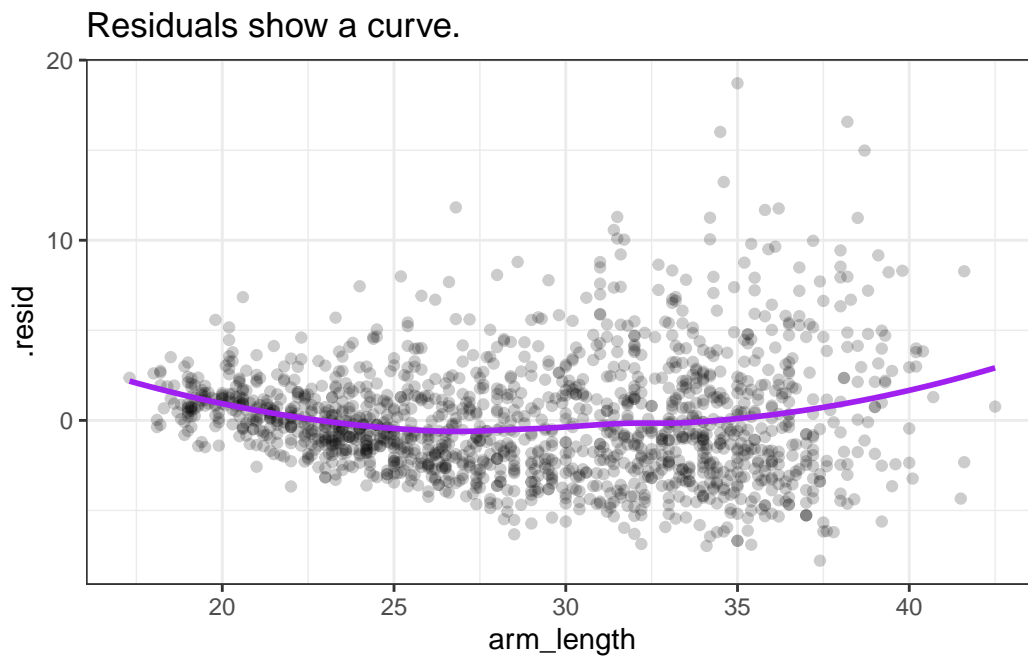
14.6.3 Using the Box-Cox approach to identify a transformation

```
boxCox(nnyfs_c$arm_circ ~ nnyfs_c$arm_length)
```


Profile Log–likelihood
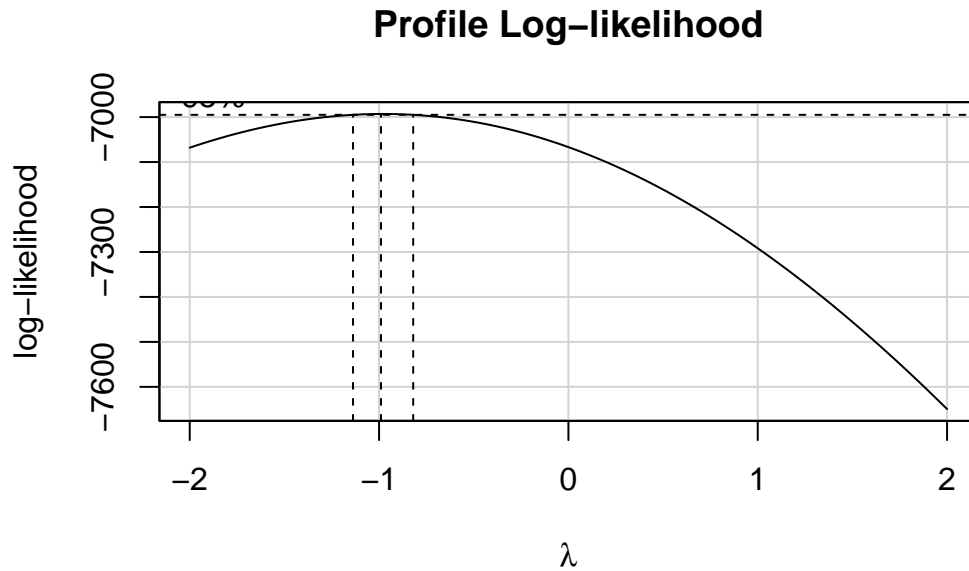
```
powerTransform(nnyfs_c$arm_circ ~ nnyfs_c$arm_length)
```

```
Estimated transformation parameter
        Y1
-0.9783135
```

This suggests that we should transform the `arm_circ` data by taking its inverse (power = -1.) Let's take a look at that result.

14.6.4 Plots after Inverse Transformation

Let's build (on the left) the revised scatterplot and (on the right) the revised residual plot after transforming the outcome (`arm_circ`) by taking its inverse.

```
nnyfs_c <- nnyfs_c |>
    mutate(inv_arm_circ = 1/arm_circ)

p1 <- ggplot(nnyfs_c, aes(x = arm_length, y = inv_arm_circ)) +
    geom_point(alpha = 0.2) +
    geom_smooth(method = "loess", formula = y ~ x,
                se = FALSE, color = "blue") +
    geom_smooth(method = "lm", formula = y ~ x,
                se = FALSE, color = "red") +
    labs(title = "Transformation reduces curve")

m2 <- lm(inv_arm_circ ~ arm_length, data = nnyfs_c)

nnyfs_c_aug2 <- augment(m2, data = nnyfs_c)

p2 <- ggplot(nnyfs_c_aug2, aes(x = arm_length, y = .resid)) +
    geom_point(alpha = 0.2) +
    geom_smooth(method = "loess", col = "purple",
                formula = y ~ x, se = FALSE) +
    labs(title = "Residuals much improved")

p1 + p2 +
    plot_annotation(title = "Evaluating the Inverse Transformation")
```

Evaluating the Inverse Transformation
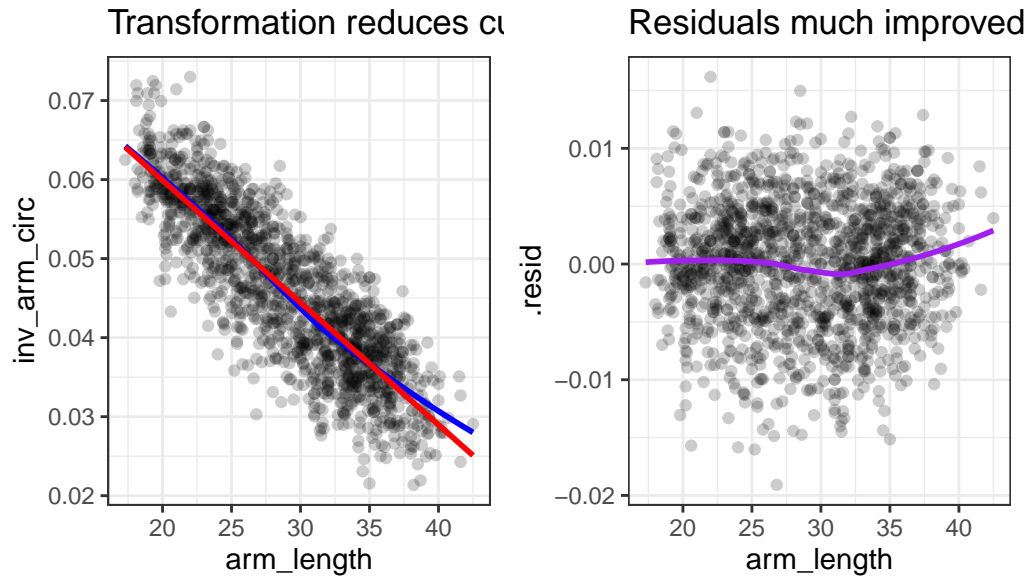
14.7 Coming Up

The rest of Part A of these Course Notes walk through additional case studies, showing some new ideas, but primarily providing context for some tools we've already seen.

15 Studying Crab Claws

For our next example, we'll consider a study from zoology, specifically carcinology - the study of crustaceans. My source for these data is Chapter 7 in Ramsey and Schafer (2002) which drew the data from a figure in Yamada and Boulding (1998).

15.1 Setup: Packages Used Here

```
knitr::opts_chunk$set(comment = NA)

library(janitor)
library(broom)
library(knitr)
library(tidyverse)

theme_set(theme_bw())
```

We will also use the `describe` function from the `psych` package.

15.2 The Data

The available data are the mean closing forces (in Newtons) and the propodus heights (mm) of the claws on 38 crabs that came from three different species. The *propodus* is the segment of the crab's clawed leg with an immovable finger and palm.
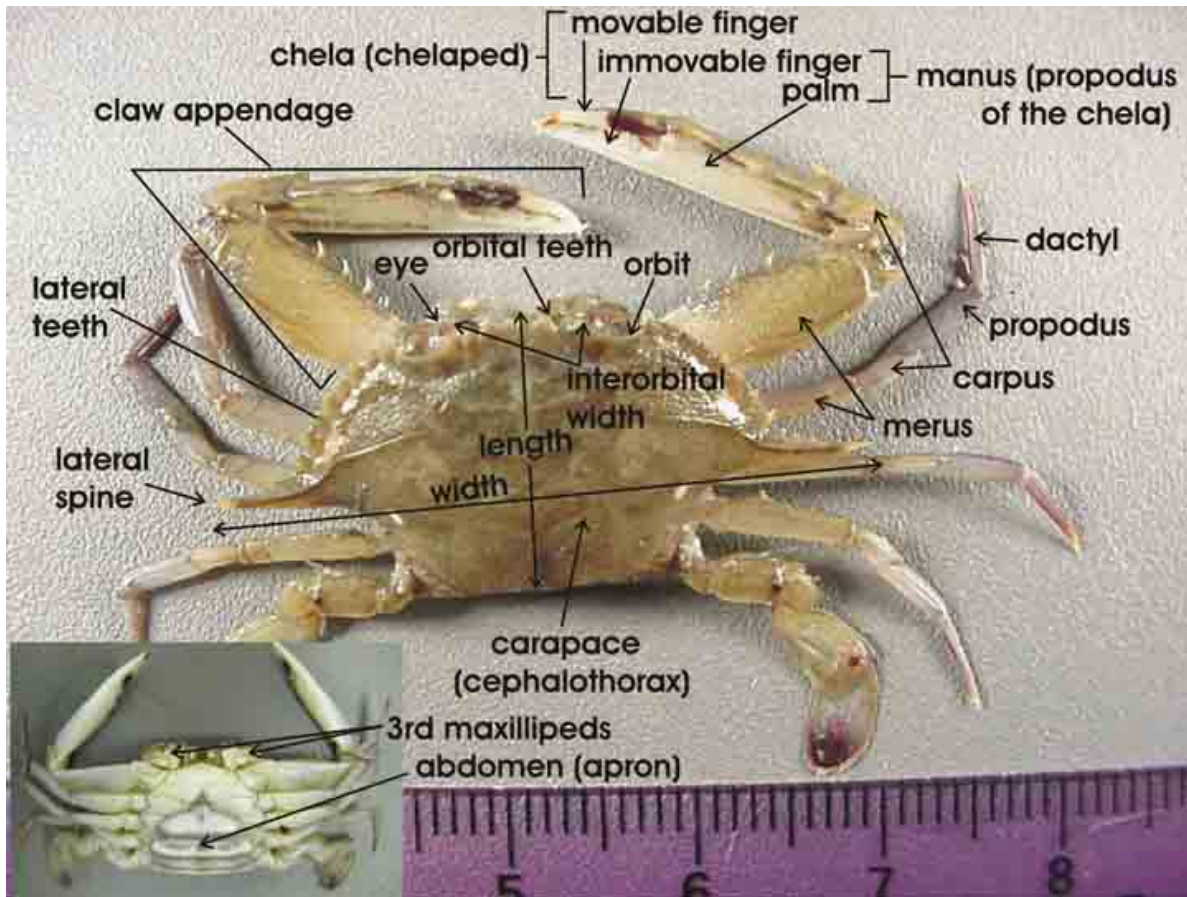
This was part of a study of the effects that predatory intertidal crab species have on populations of snails. The three crab species under study are:

- 14 Hemigraspus nudus, also called the purple shore crab (14 crabs)
- 12 Lophopanopeus bellus, also called the black-clawed pebble crab, and
- 12 Cancer productus, one of several species of red rock crabs (12)

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

crabs
```

```
# A tibble: 38 x 4
   crab species            force height
  <dbl> <chr>              <dbl>  <dbl>
1     1 Hemigraspus nudus      4    8
2     2 Lophopanopeus bellus 15.1    7.9
3     3 Cancer productus       5    6.7
```

```
 4       4 Lophopanopeus bellus   2.9    6.6
 5       5 Hemigraspus nudus      3.2    5
 6       6 Hemigraspus nudus      9.5    7.9
 7       7 Cancer productus      22.5    9.4
 8       8 Hemigraspus nudus      7.4    8.3
 9       9 Cancer productus      14.6   11.2
10      10 Lophopanopeus bellus   8.7    8.6
# i 28 more rows
```

The `species` information is stored here as a character variable. How many different crabs are we talking about in each `species`?

```
crabs |> tabyl(species)
```

```
            species   n    percent
    Cancer productus  12  0.3157895
   Hemigraspus nudus  14  0.3684211
Lophopanopeus bellus  12  0.3157895
```

As it turns out, we're going to want to treat the `species` information as a **factor** with three levels, rather than as a character variable.

```
crabs <- crabs |>
    mutate(species = factor(species))
```

Here's a quick summary of the data. Take care to note the useless results for the first two variables. At least the function flags with a * those variables it thinks are non-numeric.

```
psych::describe(crabs)
```

```
         vars  n  mean     sd median trimmed   mad min   max range skew kurtosis
crab        1 38 19.50 11.11  19.50   19.50 14.08   1  38.0  37.0 0.00    -1.30
species*    2 38  2.00  0.81   2.00    2.00  1.48   1   3.0   2.0 0.00    -1.50
force       3 38 12.13  8.98   8.70   11.53  9.04   2  29.4  27.4 0.47    -1.25
height      4 38  8.81  2.23   8.25    8.78  2.52   5  13.1   8.1 0.19    -1.14
           se
crab     1.80
species* 0.13
force    1.46
height   0.36
```

Actually, we're more interested in these results after grouping by species.

```
crabs |>
    group_by(species) |>
    summarise(n = n(), median(force), median(height))
```

```
# A tibble: 3 x 4
  species                n `median(force)` `median(height)`
  <fct>              <int>           <dbl>            <dbl>
1 Cancer productus      12            19.7             11.0
2 Hemigraspus nudus     14             3.7              7.9
3 Lophopanopeus bellus  12            14.8             8.15
```

15.3 Association of Size and Force

Suppose we want to describe force on the basis of height, across all 38 crabs. We'll add titles and identify the three species of crab, using shape and color.

```
ggplot(crabs, aes(x = height, y = force, color = species, shape = species)) +
    geom_point(size = 3) +
    labs(title = "Crab Claw Force by Size",
         x = "Claw's propodus height (mm)", y = "Mean closing force (N)")
```

Crab Claw Force by Size
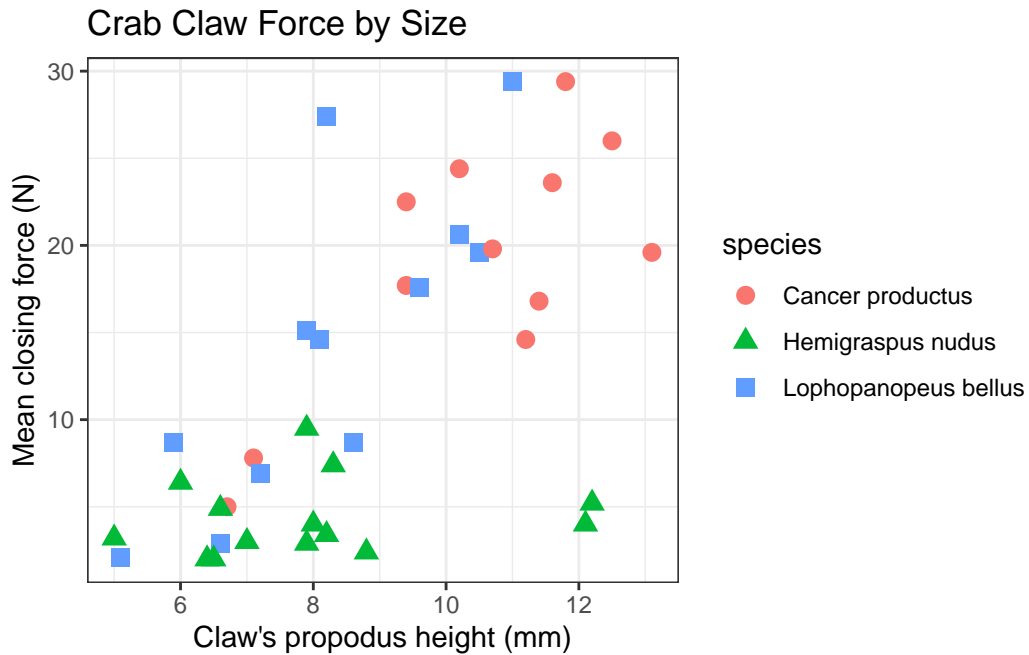
A faceted plot for each species really highlights the difference in force between the *Hemigraspus nudus* and the other two species of crab.

```
ggplot(crabs, aes(x = height, y = force, color = species)) +
    geom_point(size = 3) +
    facet_wrap(~ species) +
    guides(color = "none") +
    labs(title = "Crab Claw Force by Size",
        x = "Claw's propodus height (mm)", y = "Mean closing force (N)")
```

Crab Claw Force by Size
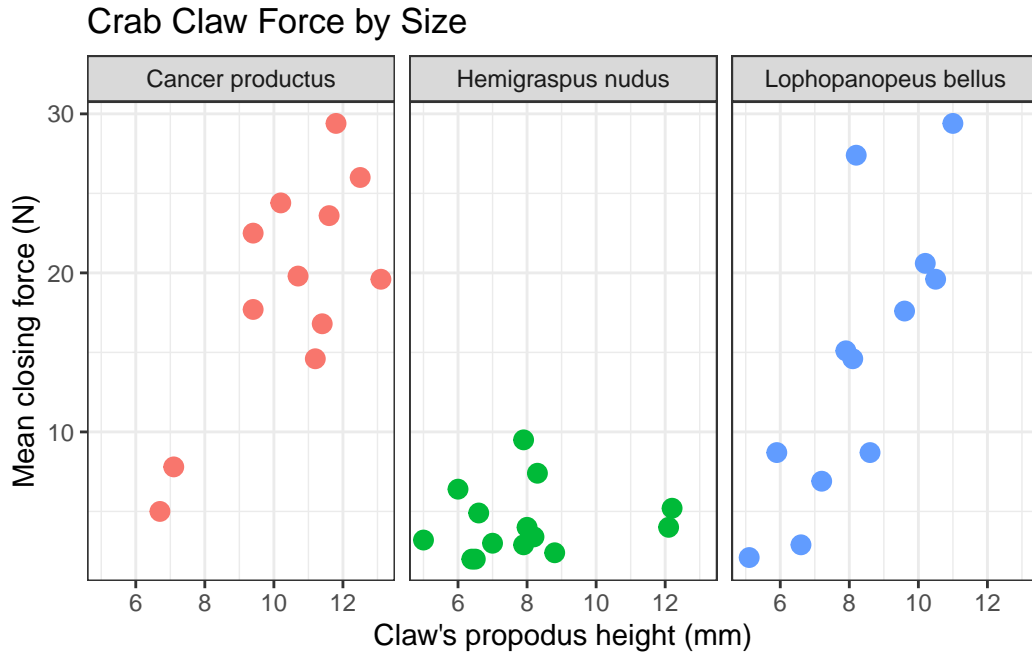
15.4 The `loess` smooth

We can obtain a smoothed curve (using several different approaches) to summarize the pattern presented by the data in any scatterplot. For instance, we might build such a plot for the complete set of 38 crabs, adding in a non-linear smooth function (called a loess smooth.)

```
ggplot(crabs, aes(x = height, y = force)) +
    geom_point() +
    geom_smooth(method = "loess", se = FALSE, formula = y ~ x) +
    labs(title = "Crab Claw Force by Size",
        x = "Claw's propodus height (mm)", y = "Mean closing force (N)")
```

Crab Claw Force by Size
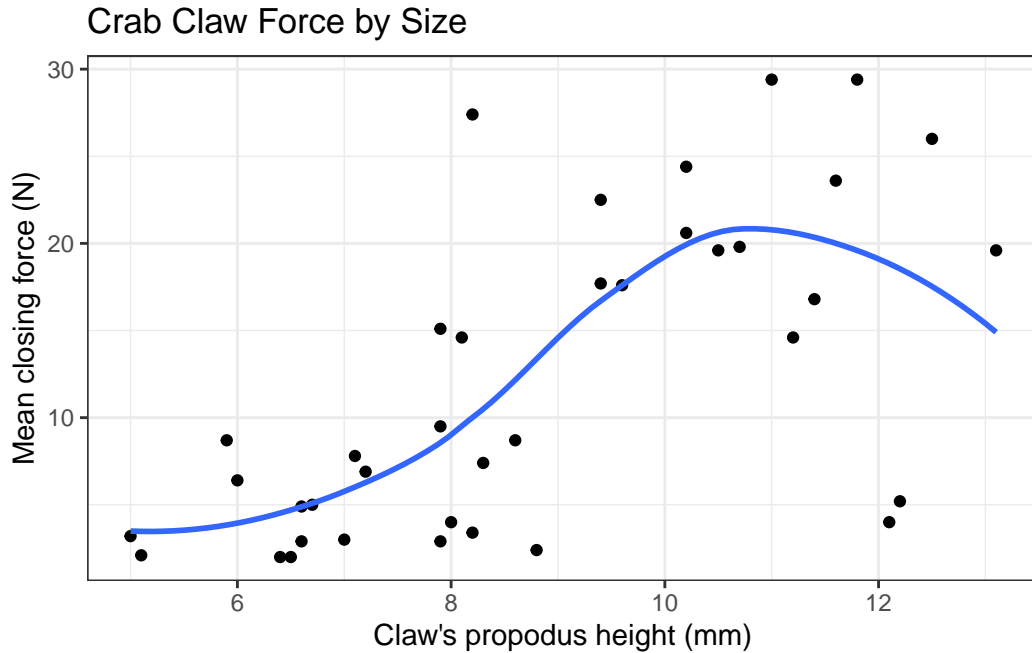
As we have discussed previously, a **loess smooth** fits a curve to data by tracking (at point x) the points within a neighborhood of point x, with more emphasis given to points near x. It can be adjusted by tweaking the `span` and `degree` parameters.

In addition to the curve, smoothing procedures can also provide confidence intervals around their main fitted line. Consider the following plot of the `crabs` information, which adjusts the span (from its default of 0.75) and also adds in the confidence intervals.

```
ggplot(crabs, aes(x = height, y = force)) +
    geom_point() +
    geom_smooth(method = "loess", formula = y ~ x, span = 0.5, se = TRUE) +
    labs(title = "Crab Claw Force by Size",
        x = "Claw's propodus height (mm)", y = "Mean closing force (N)")
```

Crab Claw Force by Size
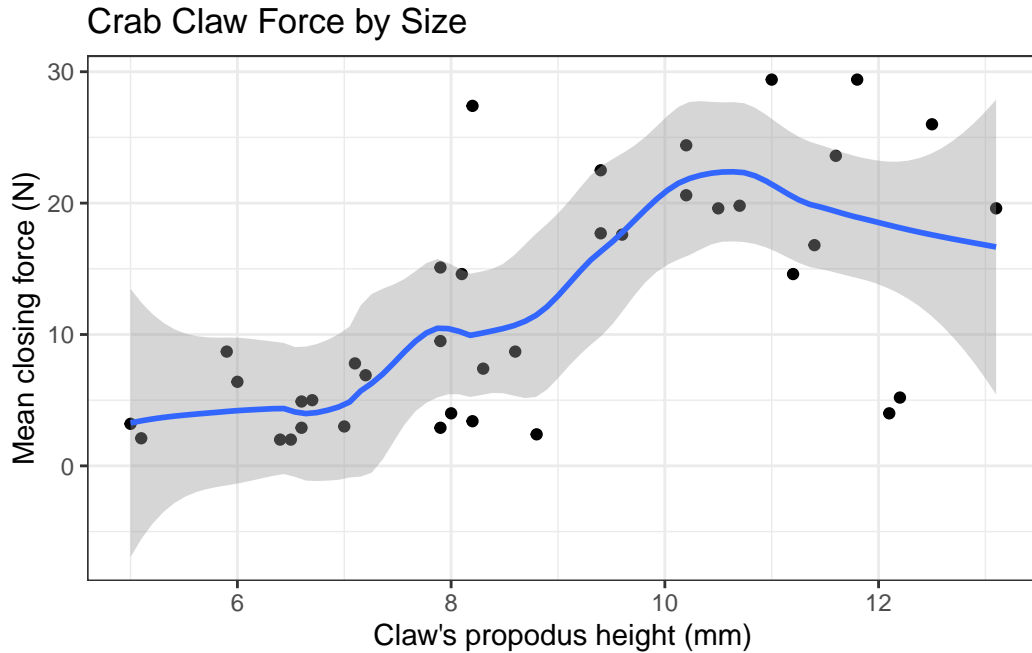
By reducing the size of the span, our resulting picture shows a much less smooth function that we generated previously.

15.4.1 Smoothing within Species

We can, of course, produce the plot above with separate smooths for each of the three species of crab.

```
ggplot(crabs, aes(x = height, y = force, group = species, color = species)) +
    geom_point(size = 3) +
    geom_smooth(method = "loess", formula = y ~ x, se = FALSE) +
    labs(title = "Crab Claw Force by Size",
         x = "Claw's propodus height (mm)", y = "Mean closing force (N)")
```

Crab Claw Force by Size
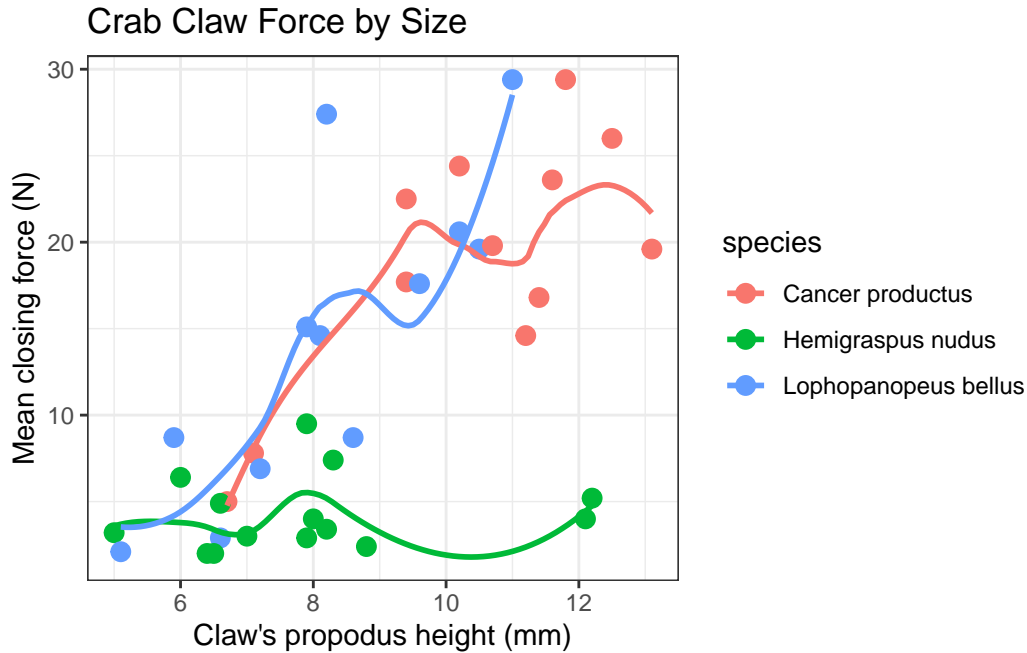
If we want to add in the confidence intervals (here I'll show them at 90% rather than the default of 95%) then this plot should be faceted. Note that by default, what is displayed when `se = TRUE` are 95% prediction intervals - the `level` function in `stat_smooth` [which can be used in place of `geom_smooth`] is used here to change the coverage percentage from 95% to 90%.

```
ggplot(crabs, aes(x = height, y = force, group = species, color = species)) +
    geom_point() +
    stat_smooth(method = "loess", formula = y ~ x, level = 0.90, se = TRUE) +
    guides(color = "none") +
    labs(title = "Crab Claw Force by Size",
        caption = "with loess smooths and 90% confidence intervals",
        x = "Claw's propodus height (mm)", y = "Mean closing force (N)") +
    facet_wrap(~ species)
```

Crab Claw Force by Size

with loess smooths and 90% confidence intervals
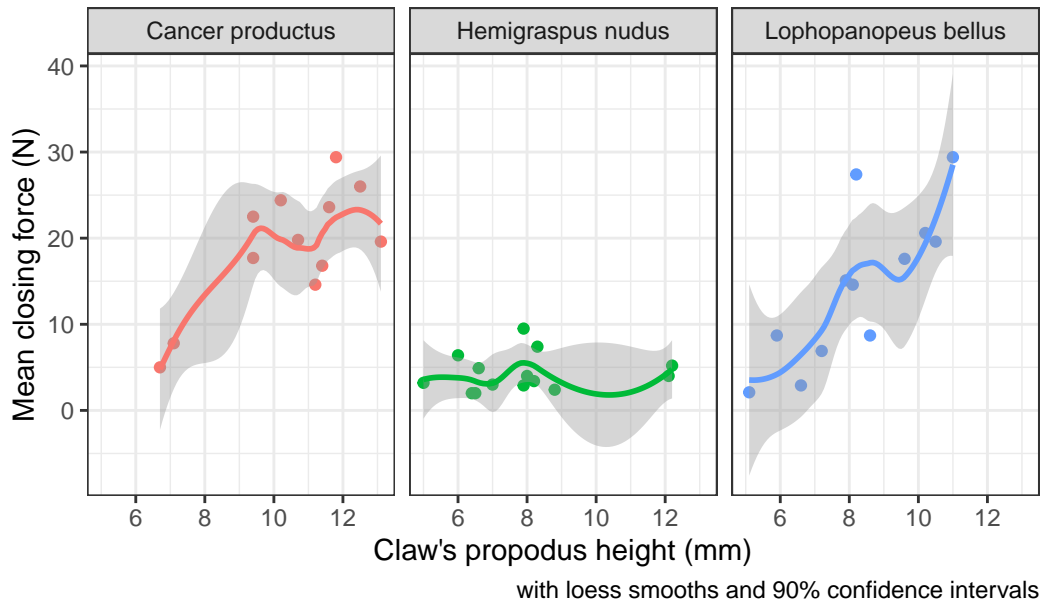
More on these and other confidence intervals later, especially in part B.

15.5 Fitting a Linear Regression Model

Suppose we plan to use a simple (least squares) linear regression model to describe force as a function of height. Is a least squares model likely to be an effective choice here?

The plot below shows the regression line predicting closing force as a function of propodus height. Here we annotate the plot to show the actual fitted regression line, which required fitting it with the `lm` statement prior to developing the graph.

```
mod <- lm(force ~ height, data = crabs)

ggplot(crabs, aes(x = height, y = force)) +
    geom_point() +
    geom_smooth(method = "lm", formula = y ~ x,  color = "red") +
    labs(title = "Crab Claw Force by Size with Linear Regression Model",
        x = "Claw's propodus height (mm)", y = "Mean closing force (N)") +
    annotate("text", x = 11, y = 0, color = "red", fontface = "italic",
            label = paste( "Force = ", round_half_up(coef(mod)[1],3), " + ",
                        round_half_up(coef(mod)[2],3), " Height" ))
```

Crab Claw Force by Size with Linear Regression Model

Force = −11.087 + 2.635 Height
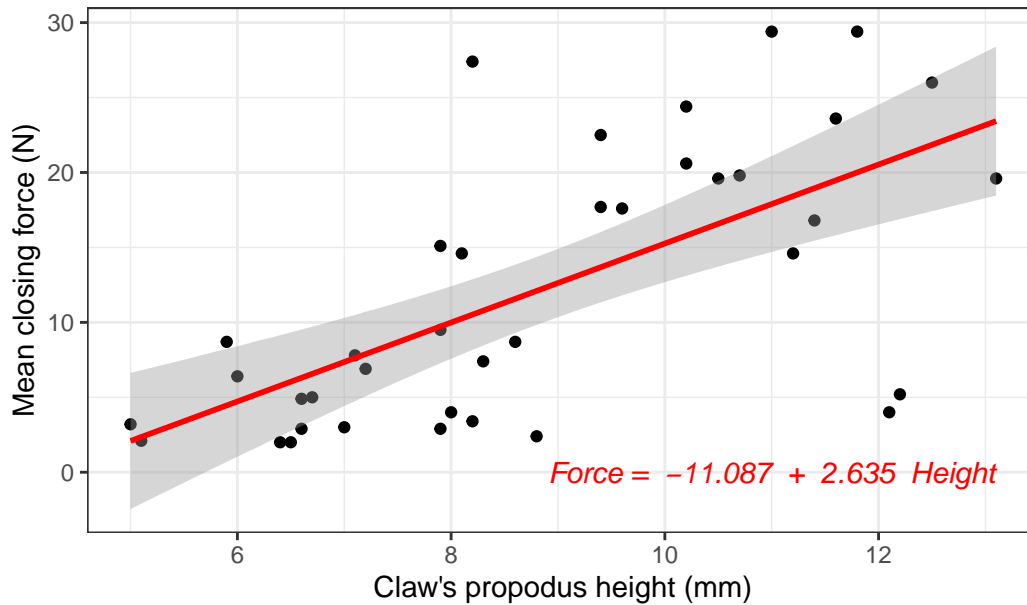
The **lm** function, again, specifies the linear model we fit to predict force using height. Here's the summary.

```
summary(lm(force ~ height, data = crabs))
```

```
Call:
lm(formula = force ~ height, data = crabs)

Residuals:
     Min       1Q    Median       3Q      Max
-16.7945  -3.8113  -0.2394   4.1444  16.8814

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -11.0869     4.6224  -2.399   0.0218 *
height        2.6348     0.5089   5.177 8.73e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.892 on 36 degrees of freedom
Multiple R-squared:  0.4268,    Adjusted R-squared:  0.4109
```

```
F-statistic:  26.8 on 1 and 36 DF,  p-value: 8.73e-06
```

Again, the key things to realize are:

- The outcome variable in this model is **force**, and the predictor variable is **height**.
- The straight line model for these data fitted by least squares is force = -11.087 + 2.635 height.
- The slope of height is positive, which indicates that as height increases, we expect that force will also increase. Specifically, we expect that for every additional mm of height, the force will increase by 2.635 Newtons.
- The multiple R-squared (squared correlation coefficient) is 0.427, which implies that 42.7% of the variation in force is explained using this linear model with height. It also implies that the Pearson correlation between force and height is the square root of 0.427, or 0.653.

15.6 Is a Linear Model Appropriate?

The zoology (at least as described in Ramsey and Schafer (2002)) suggests that the actual nature of the relationship would be represented by a log-log relationship, where the log of force is predicted by the log of height.

This log-log model is an appropriate model when we think that percentage increases in X (height, here) lead to constant percentage increases in Y (here, force).

To see the log-log model in action, we plot the log of force against the log of height. We could use either base 10 (log10 in R) or natural (log in R) logarithms.

```
ggplot(crabs, aes(x = log(height), y = log(force))) +
    geom_point() +
    geom_smooth(method = "lm", formula = y ~ x) +
    labs(title = "Log-Log Model for Crabs data")
```

Log–Log Model for Crabs data
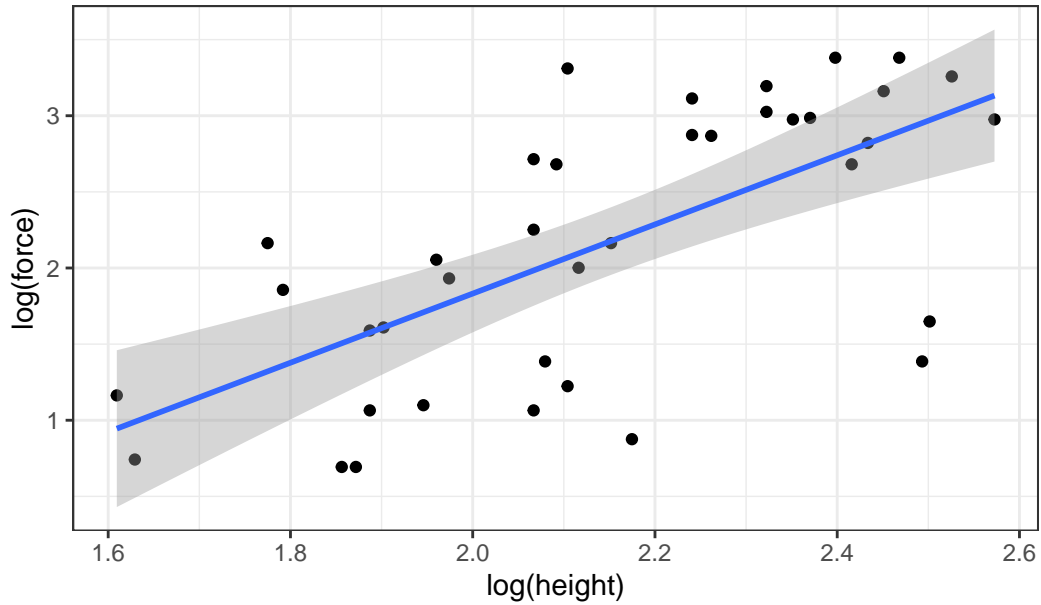
The correlations between the raw force and height and between their logarithms turn out to be quite similar, and because the log transformation is monotone in these data, there's actually no change at all in the Spearman correlations.

Correlation of	Pearson r	Spearman r
force and height	0.653	0.657
log(force) and log(height)	0.662	0.657

15.6.1 The log-log model

```
crab_loglog <- lm(log(force) ~ log(height), data = crabs)

summary(crab_loglog)
```

```
Call:
lm(formula = log(force) ~ log(height), data = crabs)

Residuals:
    Min      1Q  Median      3Q     Max
```

```
-1.5657 -0.4450  0.1884  0.4798  1.2422
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -2.7104     0.9251  -2.930  0.00585 **
log(height)   2.2711     0.4284   5.302 5.96e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 0.6748 on 36 degrees of freedom
Multiple R-squared:  0.4384,    Adjusted R-squared:  0.4228
F-statistic: 28.11 on 1 and 36 DF,  p-value: 5.96e-06
```

Our regression equation is log(force) = -2.71 + 2.271 log(height).

So, for example, if we found a crab with propodus height = 10 mm, our prediction for that crab's claw force (in Newtons) based on this log-log model would be...

- log(force) = -2.71 + 2.271 log(10)
- log(force) = -2.71 + 2.271 x 2.3025851
- log(force) = 2.5190953
- and so predicted force = exp(2.5190953) = 12.4173582 Newtons, which, naturally, we would round to 12.417 Newtons to match the data set's level of precision.

15.6.2 How does this compare to our original linear model?

```
crab_linear <- lm(force ~ height, data = crabs)

summary(crab_linear)
```

```
Call:
lm(formula = force ~ height, data = crabs)
```

```
Residuals:
     Min       1Q   Median       3Q      Max
-16.7945  -3.8113  -0.2394   4.1444  16.8814
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -11.0869     4.6224  -2.399   0.0218 *
```

```
height          2.6348      0.5089    5.177 8.73e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.892 on 36 degrees of freedom
Multiple R-squared:  0.4268,    Adjusted R-squared:  0.4109
F-statistic:  26.8 on 1 and 36 DF,  p-value: 8.73e-06
```

The linear regression equation is force = -11.087 + 2.635 height.

So, for example, if we found a crab with propodus height = 10 mm, our prediction for that crab's claw force (in Newtons) based on this linear model would be…

- force = -11.0869025 + 2.6348232 x 10
- force = -11.0869025 + 26.3482321
- so predicted force = 15.2613297, which we would round to 15.261 Newtons.

So, it looks like the two models give meaningfully different predictions.

15.7 Making Predictions with a Model

The `broom` package's `augment` function provides us with a consistent method for obtaining predictions (also called fitted values) for a new crab or for our original data. Suppose we want to predict the `force` level for two new crabs: one with height = 10 mm, and another with height = 12 mm.

```
newcrab <- tibble(crab = c("Crab_A", "Crab_B"), height = c(10, 12))

augment(crab_linear, newdata = newcrab)
```

```
# A tibble: 2 x 3
  crab    height .fitted
  <chr>    <dbl>   <dbl>
1 Crab_A      10    15.3
2 Crab_B      12    20.5
```

Should we want to obtain a prediction interval, we can use the `predict` function:

```
predict(crab_linear, newdata = newcrab, interval = "prediction", level = 0.95)
```

```
     fit      lwr      upr
1 15.26133 1.048691 29.47397
2 20.53098 5.994208 35.06774
```

We'd interpret this result as saying that the linear model's predicted force associated with a single new crab claw with propodus height 10 mm is 15.3 Newtons, and that a 95% prediction interval for the true value of such a force for such a claw is between 1.0 and 29.5 Newtons. More on prediction intervals later.

15.7.1 Predictions After a Transformation

We can also get predictions from the log-log model. The default choice is a 95% prediction interval.

```
predict(crab_loglog, newdata = newcrab, interval = "prediction")
```

```
     fit      lwr      upr
1 2.519095 1.125900 3.912291
2 2.933174 1.515548 4.350800
```

Of course, these predictions describe the `log(force)` for such a crab claw. To get the prediction in terms of simple force, we'd need to back out of the logarithm, by exponentiating our point estimate and the prediction interval endpoints.

```
exp(predict(crab_loglog, newdata = newcrab, interval = "prediction"))
```

```
      fit      lwr      upr
1 12.41736 3.082989 50.01341
2 18.78716 4.551916 77.54044
```

We'd interpret this result as saying, for the first new crab, that the log-log model's predicted force associated with a single new crab claw with propodus height 10 mm is 12.4 Newtons, and that a 95% prediction interval for the true value of such a force for such a claw is between 3.1 and 50.0 Newtons.

15.7.2 Comparing Model Predictions

Suppose we wish to build a plot of force vs height with a straight line for the linear model's predictions, and a new curve for the log-log model's predictions, so that we can compare and contrast the implications of the two models on a common scale. The `predict` function, when not given a new data frame, will use the existing predictor values that are in our `crabs` data. Such predictions are often called fitted values.

To put the two sets of predictions on the same scale despite the differing outcomes in the two models, we'll exponentiate the results of the log-log model, and build a little data frame containing the heights and the predicted forces from that model.

```
loglogdat <- tibble(height = crabs$height, force = exp(predict(crab_loglog)))
```

A cleaner way to do this might be to use the `augment` function directly from `broom`:

```
augment(crab_loglog)
```

```
# A tibble: 38 x 8
   `log(force)` `log(height)` .fitted   .resid   .hat .sigma   .cooksd .std.resid
          <dbl>         <dbl>   <dbl>    <dbl>  <dbl>  <dbl>     <dbl>      <dbl>
 1         1.39          2.08    2.01 -6.26e-1 0.0280  0.676 1.28e- 2    -0.941
 2         2.71          2.07    1.98  7.31e-1 0.0287  0.673 1.79e- 2     1.10
 3         1.61          1.90    1.61 -1.15e-4 0.0499  0.684 8.06e-10    -0.000175
 4         1.06          1.89    1.58 -5.11e-1 0.0530  0.679 1.69e- 2    -0.778
 5         1.16          1.61   0.945  2.18e-1 0.142   0.683 1.01e- 2     0.349
 6         2.25          2.07    1.98  2.68e-1 0.0287  0.683 2.39e- 3     0.402
 7         3.11          2.24    2.38  7.35e-1 0.0301  0.673 1.90e- 2     1.11
 8         2.00          2.12    2.10 -9.44e-2 0.0266  0.684 2.75e- 4    -0.142
 9         2.68          2.42    2.78 -9.55e-2 0.0561  0.684 6.30e- 4    -0.146
10         2.16          2.15    2.18 -1.32e-2 0.0263  0.684 5.34e- 6    -0.0199
# i 28 more rows
```

Now, we're ready to use the `geom_smooth` approach to plot the linear fit, and `geom_line` (which also fits curves) to display the log-log fit.

```
ggplot(crabs, aes(x = height, y = force)) +
    geom_point() +
    geom_smooth(method = "lm", se = FALSE,
                formula = y ~ x, col="blue", linetype = 2) +
    geom_line(data = loglogdat, col = "red", linetype = 2, size = 1) +
    annotate("text", 7, 12, label = "Linear Model", col = "blue") +
```

```
        annotate("text", 10, 8, label = "Log-Log Model", col = "red") +
        labs(title = "Comparing the Linear and Log-Log Models for Crab Claw data")
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```


Comparing the Linear and Log–Log Models for Crab Claw data
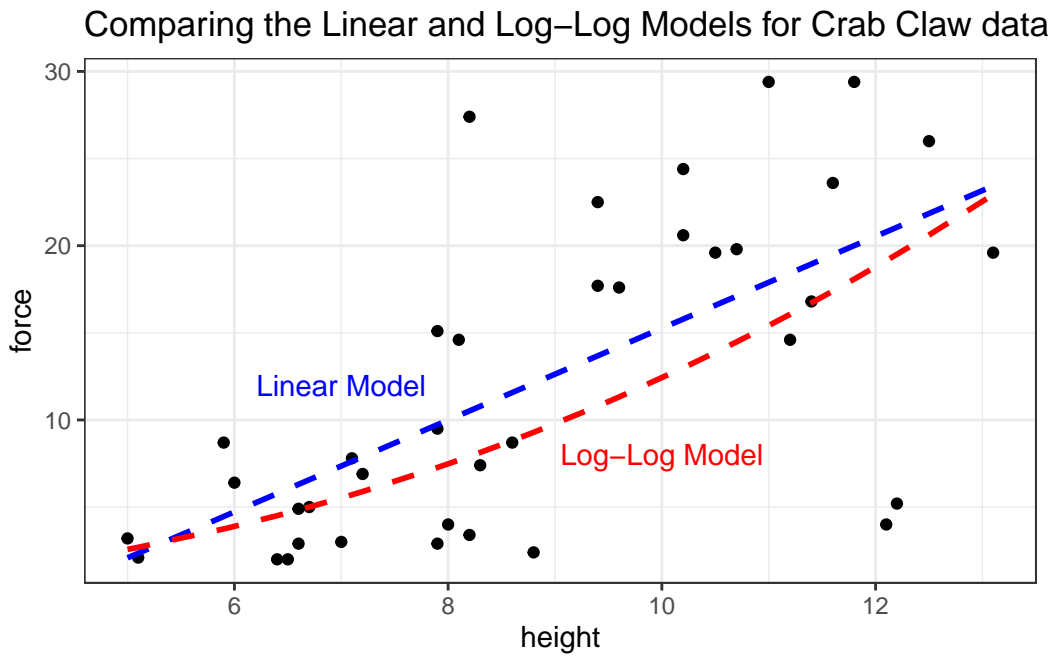
Based on these 38 crabs, we see some modest differences between the predictions of the two models, with the log-log model predicting generally lower closing force for a given propodus height than would be predicted by a linear model.

16 Dehydration Recovery

16.1 Setup: Packages Used Here

```
knitr::opts_chunk$set(comment = NA)

library(knitr)
library(broom)
library(GGally)
library(ggstance)
library(kableExtra)
library(modelsummary)
library(patchwork)
library(tidyverse)

theme_set(theme_bw())
```

We will also use the `favstats` function from the `mosaic` package.

16.2 The Data

The `hydrate` data describe the degree of recovery that takes place 90 minutes following treatment of moderate to severe dehydration, for 36 children diagnosed at a hospital's main pediatric clinic.

Upon diagnosis and study entry, patients were treated with an electrolytic solution at one of seven `dose` levels (0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0 mEq/l) in a frozen, flavored, ice popsicle. The degree of rehydration was determined using a subjective scale based on physical examination and parental input, converted to a 0 to 100 point scale, representing the percent of recovery (`recov.score`). Each child's `age` (in years) and `weight` (in pounds) are also available.

First, we'll check ranges (and for missing data) in the `hydrate` file.

```
hydrate <- read_csv("data/hydrate.csv")

summary(hydrate)
```

```
      id           recov.score         dose            age
 Min.   : 1.00   Min.   : 44.00   Min.   :0.000   Min.   : 3.000
 1st Qu.: 9.75   1st Qu.: 61.50   1st Qu.:1.000   1st Qu.: 5.000
 Median :18.50   Median : 71.50   Median :1.500   Median : 6.500
 Mean   :18.50   Mean   : 71.56   Mean   :1.569   Mean   : 6.667
 3rd Qu.:27.25   3rd Qu.: 80.00   3rd Qu.:2.500   3rd Qu.: 8.000
 Max.   :36.00   Max.   :100.00   Max.   :3.000   Max.   :11.000
     weight
 Min.   :22.00
 1st Qu.:34.50
 Median :47.50
 Mean   :46.89
 3rd Qu.:57.25
 Max.   :76.00
```

There are no missing values, and all of the ranges make sense. There are no especially egregious problems to report.

16.3 A Scatterplot Matrix

Next, we'll use a scatterplot matrix to summarize relationships between the outcome recov.score and the key predictor dose as well as the ancillary predictors age and weight, which are of less interest, but are expected to be related to our outcome. The one below uses the ggpairs function in the GGally package, as introduced in Part A of the Notes. We place the outcome in the bottom row, and the key predictor immediately above it, with age and weight in the top rows, using the select function within the 'ggpairs call.

```
hydrate |>
  select(age, weight, dose, recov.score) |>
  ggpairs()
```
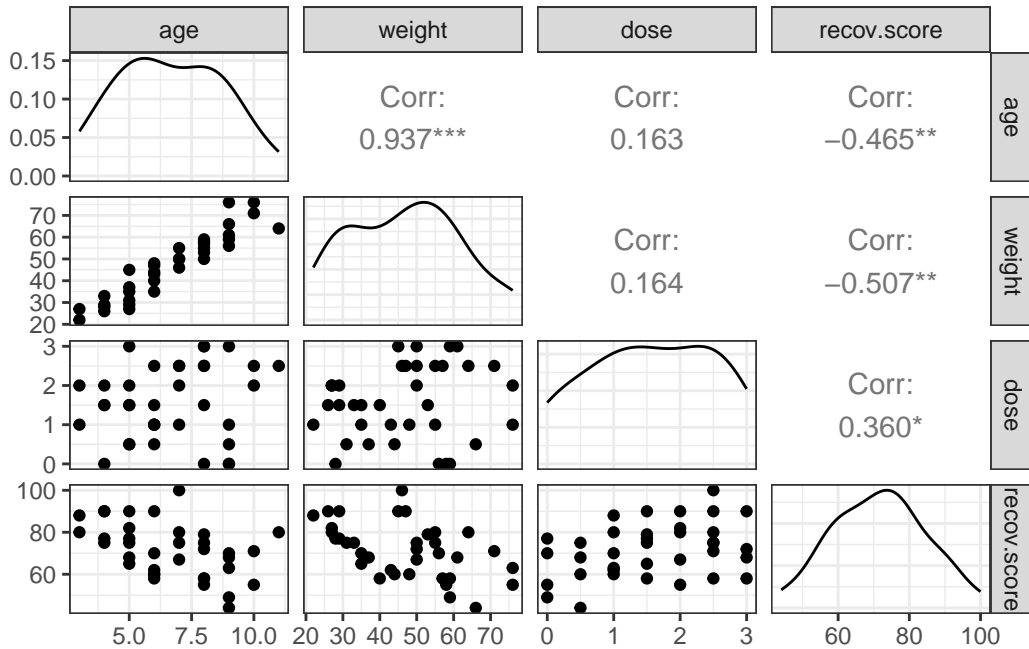
What can we conclude here?

- It looks like `recov.score` has a moderately strong negative relationship with both `age` and `weight` (with correlations in each case around -0.5), but a positive relationship with `dose` (correlation = 0.36).
- The distribution of `recov.score` looks to be pretty close to Normal. No potential predictors (`age`, `weight` and `dose`) show substantial non-Normality.
- `age` and `weight`, as we'd expect, show a very strong and positive linear relationship, with $r = 0.94$
- Neither `age` nor `weight` shows a meaningful relationship with `dose`. ($r = 0.16$)

16.4 Are the recovery scores well described by a Normal model?

Next, we'll do a more thorough graphical summary of our outcome, recovery score.

```
p1 <- ggplot(hydrate, aes(sample = recov.score)) +
  geom_qq(col = '#440154') + geom_qq_line(col = "red") +
  theme(aspect.ratio = 1) +
  labs(title = "Normal Q-Q plot: hydrate")

p2 <- ggplot(hydrate, aes(x = recov.score)) +
```

```
    geom_histogram(aes(y = stat(density)),
                   bins = 10, fill = '#440154', col = '#FDE725') +
    stat_function(fun = dnorm,
                  args = list(mean = mean(hydrate$recov.score),
                              sd = sd(hydrate$recov.score)),
                  col = "red", lwd = 1.5) +
    labs(title = "Density Function: hydrate")

  p3 <- ggplot(hydrate, aes(x = recov.score, y = "")) +
    geom_boxplot(fill = '#440154', outlier.color = '#440154') +
    labs(title = "Boxplot: hydrate", y = "")

  p1 + (p2 / p3 + plot_layout(heights = c(4,1)))
```

```
Warning: `stat(density)` was deprecated in ggplot2 3.4.0.
i Please use `after_stat(density)` instead.
```
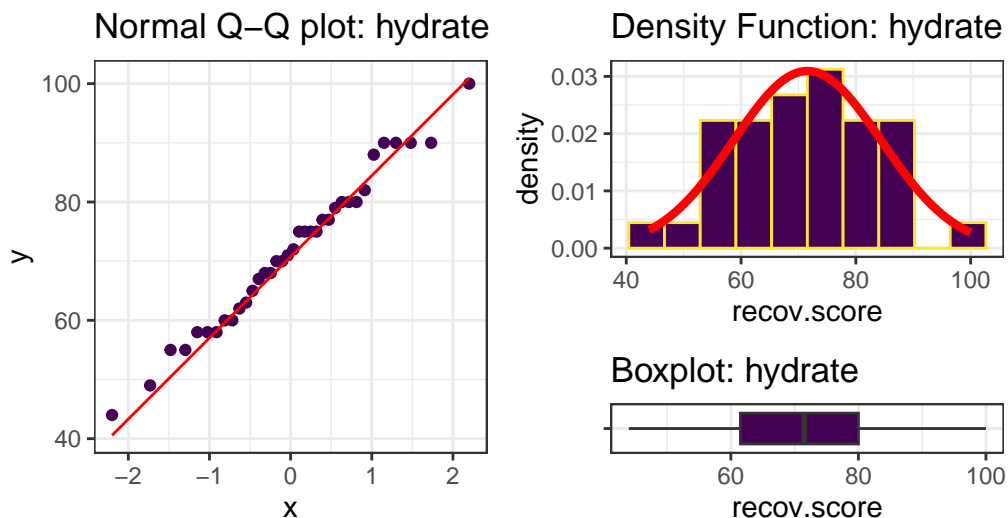


```
  mosaic::favstats(~ recov.score, data = hydrate) |> kable(digits = 1)
```

min	Q1	median	Q3	max	mean	sd	n	missing
44	61.5	71.5	80	100	71.6	12.9	36	0

I see no serious problems with assuming Normality for these recovery scores. Our outcome variable doesn't in any way *need* to follow a Normal distribution, but it's nice when it does, because summaries involving means and standard deviations make sense.

16.5 Simple Regression: Using Dose to predict Recovery

To start, consider a simple (one predictor) regression model using `dose` alone to predict the % Recovery (`recov.score`). Ignoring the `age` and `weight` covariates, what can we conclude about this relationship?

16.6 The Scatterplot, with fitted Linear Model

```
ggplot(hydrate, aes(x = dose, y = recov.score)) +
    geom_point(size = 2) +
    geom_smooth(method = "lm", formula = y ~ x, col = "red") +
    labs(title = "Simple Regression model for the hydrate data",
        x = "Dose (mEq/l)", y = "Recovery Score (points)")
```
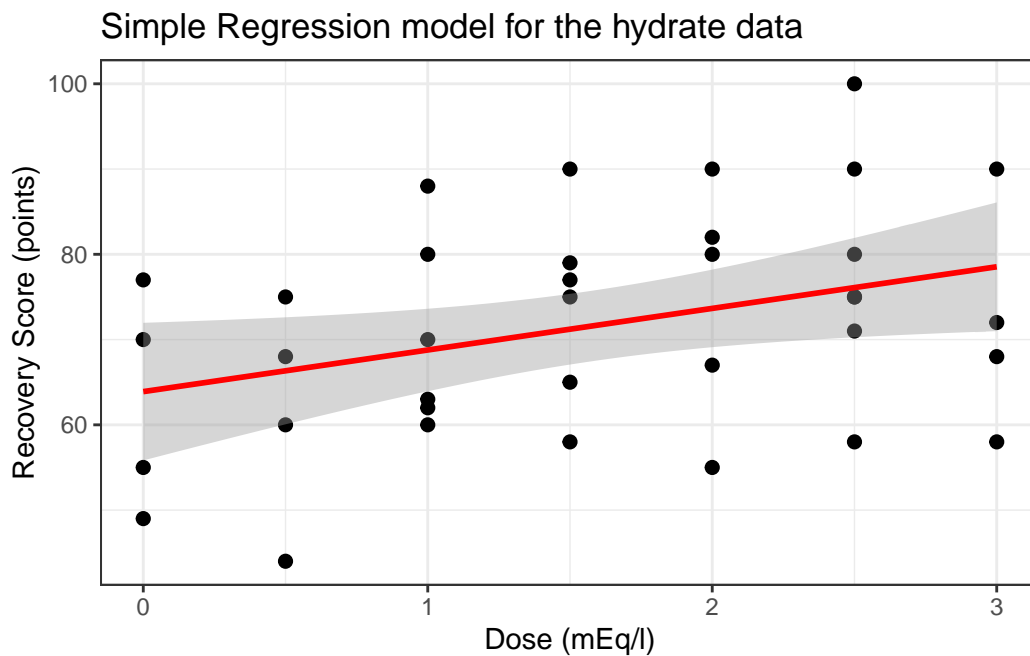
16.7 The Fitted Linear Model

To obtain the fitted linear regression model, we use the `lm` function:

```
m1 <- lm(recov.score ~ dose, data = hydrate)
```

```
tidy(m1) |> kbl(digits = 2)
```

term	estimate	std.error	statistic	p.value
(Intercept)	63.90	3.97	16.09	0.00
dose	4.88	2.17	2.25	0.03

So, our fitted regression model (prediction model) is `recov.score` $= 63.9 + 4.88$ `dose`.

16.7.1 Confidence Intervals

We can obtain confidence intervals around the coefficients of our fitted model with `tidy`, too.

```
tidy(m1, conf.int = TRUE, conf.level = 0.90) |>
  kbl(digits = 2)
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	63.90	3.97	16.09	0.00	57.18	70.61
dose	4.88	2.17	2.25	0.03	1.21	8.55

So, our 90% confidence interval for the slope of `dose` ranges from 1.21 to 8.55.

16.8 Coefficient Plots with `modelplot`

The `modelplot()` function from the `modelsummary` package can provide us with one potential graph.

```
modelplot(m1, conf_level = 0.90)
```

Coefficient estimates and 90% confidence intervals
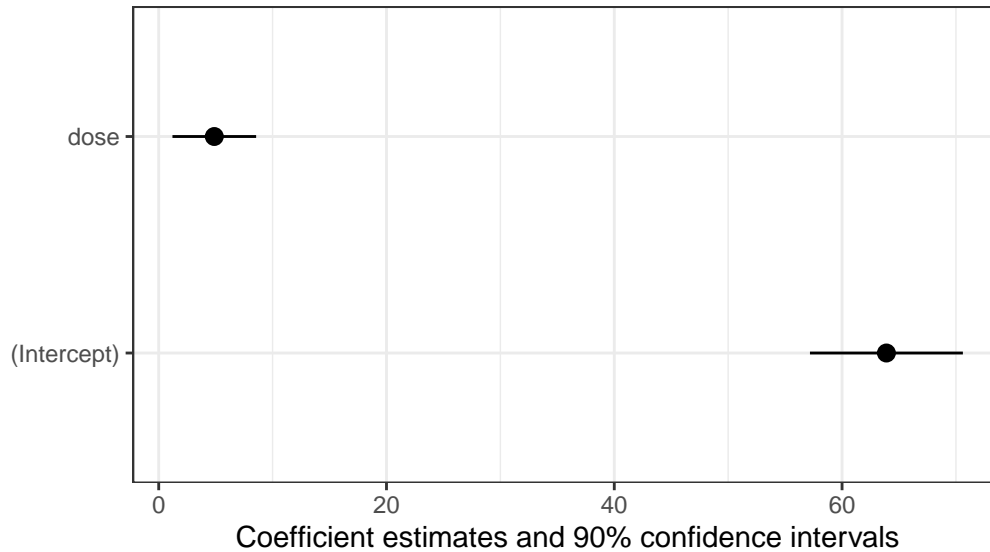
16.9 Coefficient Plots with `ggstance`

The `tidy` method makes it easy to construct coefficient plots using `ggplot2`, and we'll also make use of the `geom_crossbarh` function from the `ggstance` package.

```
td <- tidy(m1, conf.int = TRUE, conf.level = 0.90)

ggplot(td, aes(x = estimate, y = term, col = term)) +
  geom_point() +
  geom_crossbarh(aes(xmin = conf.low, xmax = conf.high)) +
  geom_vline(xintercept = 0) +
  guides(col = "none") +
  labs(title = "Estimates with 90% confidence intervals from m1 in hydrate")
```

Warning: Using the `size` aesthetic with geom_polygon was deprecated in ggplot2 3.4.0.
i Please use the `linewidth` aesthetic instead.

Warning: Using the `size` aesthetic with geom_segment was deprecated in ggplot2 3.4.0.
i Please use the `linewidth` aesthetic instead.

Estimates with 90% confidence intervals from m1 in hydra
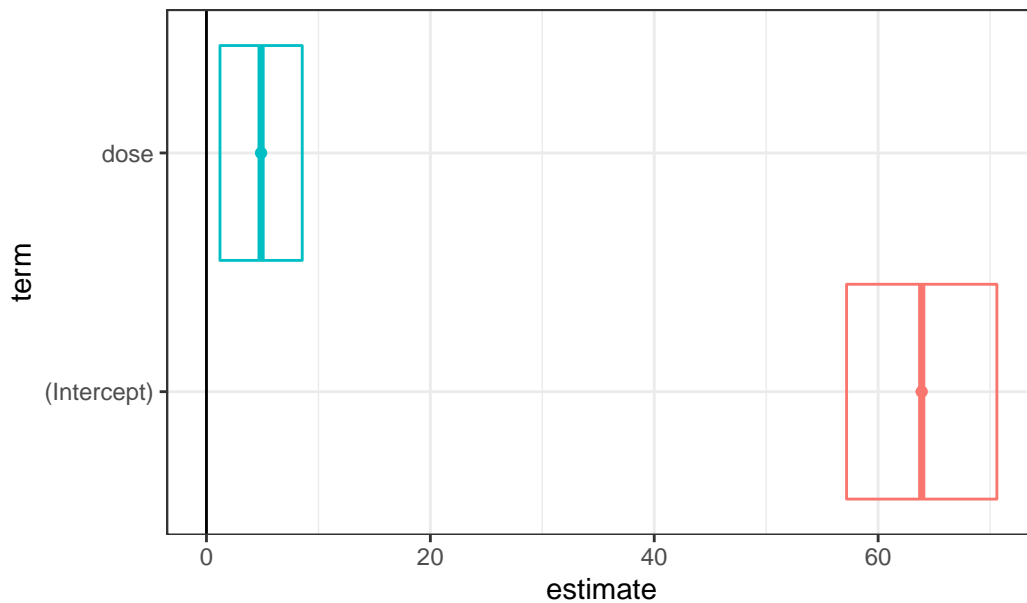
Another option would be to use `geom_errorbarh` in this setting, perhaps with a different color scheme...

```
td <- tidy(m1, conf.int = TRUE, conf.level = 0.90)
ggplot(td, aes(x = estimate, y = term, col = term)) +
  geom_point() +
  geom_errorbarh(aes(xmin = conf.low, xmax = conf.high)) +
  geom_vline(xintercept = 0) +
  scale_color_viridis_d(end = 0.5) +
  guides(col = "none") +
  labs(title = "Estimates with 90% confidence intervals from m1 in hydrate")
```

```
Warning: Using the `size` aesthetic with geom_path was deprecated in ggplot2 3.4.0.
i Please use the `linewidth` aesthetic instead.
```

Estimates with 90% confidence intervals from m1 in hydra
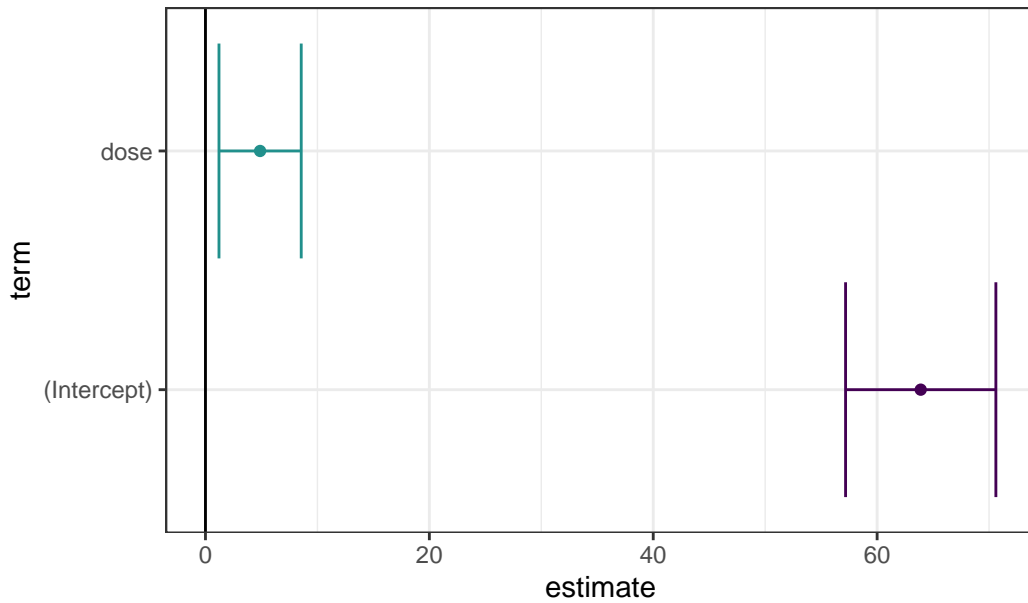
16.10 The Summary Output

To get a more complete understanding of the fitted model, we'll summarize it.

```
summary(lm(recov.score ~ dose, data = hydrate))
```

```
Call:
lm(formula = recov.score ~ dose, data = hydrate)

Residuals:
     Min       1Q   Median       3Q      Max
-22.3360  -7.2763   0.0632   8.4233  23.9028

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   63.896      3.970  16.093   <2e-16 ***
dose           4.881      2.172   2.247   0.0313 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 12.21 on 34 degrees of freedom
Multiple R-squared:  0.1293,    Adjusted R-squared:  0.1037
F-statistic: 5.047 on 1 and 34 DF,  p-value: 0.03127
```

16.10.1 Model Specification

1. The first part of the output specifies the model that has been fit.

   - Here, we have a simple regression model that predicts `recov.score` on the basis of `dose`.
   - Notice that we're treating `dose` here as a quantitative variable. If we wanted `dose` to be treated as a factor, we'd have specified that in the model.

16.10.2 Residual Summary

2. The second part of the output summarizes the regression **residuals** across the subjects involved in fitting the model.

   - The **residual** is defined as the Actual value of our outcome minus the predicted value of that outcome fitted by the model.
   - In our case, the residual for a given child is their actual `recov.score` minus the predicted `recov.score` according to our model, for that child.
   - The residual summary gives us a sense of how "incorrect" our predictions are for the `hydrate` observations.
     - A positive residual means that the observed value was higher than the predicted value from the linear regression model, so the prediction was too low.
     - A negative residual means that the observed value was lower than the predicted value from the linear regression model, so the prediction was too high.
     - The residuals will center near 0 (the ordinary least squares model fitting process is designed so the mean of the residuals will always be zero)
     - We hope to see the median of the residuals also be near zero, generally. In this case, the median prediction is 0.06 point too low.
     - The minimum and maximum show us the largest prediction errors, made in the subjects used to fit this model.
     - Here, we predicted a recovery score that was 22.3 points too high for one patient, and another of our predicted recovery scores was 23.9 points too low.
     - The middle half of our predictions were between 8.4 points too low and 7.3 points too high.

16.10.3 Coefficients Output

3. The **Coefficients** output begins with a table of the estimated coefficients from the regression equation.

- Generally, we write a simple regression model as $y = \beta_0 + \beta_1 x$.
- In the `hydrate` model, we have `recov.score` $= \beta_0 + \beta_1$ `dose`.
- The first column of the table gives the estimated β coefficients for our model
    - Here the estimated intercept $\hat{\beta}_0 = 63.9$
    - The estimated slope of dose $\hat{\beta}_1 = 4.88$
    - Thus, our model is `recov.score` $= 63.9 + 4.88$ `dose`

We interpret these coefficients as follows:

- The intercept (63.9) is the predicted `recov.score` for a patient receiving a `dose` of 0 mEq/l of the electrolytic solution.
- The slope (4.88) of the `dose` is the predicted *change* in `recov.score` associated with a 1 mEq/l increase in the dose of electrolytic solution.

    - Essentially, if we have two children like the ones studied here, and we give Roger a popsicle with dose X and Sarah a popsicle with dose X + 1, then this model predicts that Sarah will have a recovery score that is 4.88 points higher than will Roger.
    - From the confidence interval output we saw previously with the function `confint(lm(recov.score ~ dose))`, we are 95% confident that the true slope for `dose` is between (0.47, 9.30) mEq/l. We are also 95% confident that the true intercept is between (55.8, 72.0).

16.10.4 Correlation and Slope

If we like, we can use the `cor` function to specify the Pearson correlation of `recov.score` and `dose`, which turns out to be 0.36. - Note that the **slope** in a simple regression model will follow the sign of the Pearson correlation coefficient, in this case, both will be positive.

```
hydrate |> select(recov.score, dose) |> cor()
```

```
            recov.score      dose
recov.score    1.000000 0.359528
dose           0.359528 1.000000
```

16.10.5 Coefficient Testing

```
summary(lm(recov.score ~ dose, data = hydrate))
```

```
Call:
lm(formula = recov.score ~ dose, data = hydrate)

Residuals:
     Min       1Q   Median       3Q      Max
-22.3360  -7.2763   0.0632   8.4233  23.9028

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   63.896      3.970  16.093   <2e-16 ***
dose           4.881      2.172   2.247   0.0313 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 12.21 on 34 degrees of freedom
Multiple R-squared:  0.1293,    Adjusted R-squared:  0.1037
F-statistic: 5.047 on 1 and 34 DF,  p-value: 0.03127
```

Next to each coefficient in the summary regression table is its estimated standard error, followed by the coefficient's t value (the coefficient value divided by the standard error), and the associated two-tailed p value for the test of:

- H_0: This coefficient's β value $= 0$ vs.
- H_A: This coefficient's β value $\neq 0$.

For the slope coefficient, we can interpret this choice as:

- H_0: This predictor adds minimal detectable predictive value to the model vs.
- H_A: This predictor adds some predictive value to the model.

In the **hydrate** simple regression model, by running either **tidy** with or just the **confint** function shown below, we can establish a confidence interval for each of the estimated regression coefficients.

```
tidy(m1, conf.int = TRUE, conf.level = 0.95) |> kable(digits = 2)
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	63.90	3.97	16.09	0.00	55.83	71.96
dose	4.88	2.17	2.25	0.03	0.47	9.30

```
confint(m1, level = .95)
```

```
                2.5 %     97.5 %
(Intercept) 55.826922 71.964589
dose         0.465695  9.295466
```

If the slope of dose was in fact zero, then this would mean that knowing the dose information would be of no additional value in predicting the outcome over just guessing the mean of `recov.score` for every subject.

So, since the confidence interval for the slope of dose does not include zero, it appears that there is at least some evidence that the model `m1` is more effective than a model that ignores the `dose` information (and simply predicts the mean of `recov.score` for each subject.) That's not saying much, actually.

16.10.6 Summarizing the Quality of Fit

4. The next part of the regression summary output is a summary of fit quality.

The **residual standard error** estimates the standard deviation of the prediction errors made by the model.

- If assumptions hold, the model will produce residuals that follow a Normal distribution with mean 0 and standard deviation equal to this residual standard error.

  - So we'd expect roughly 95% of our residuals to fall between -2(12.21) and +2(12.21), or roughly -24.4 to +24.4 and that we'd see virtually no residuals outside the range of -3(12.21) to +3(12.21), or roughly -36.6 to +36.6.
  - The output at the top of the summary tells us about the observed regression residuals, and that they actually range from -22 to +24.
  - In context, it's hard to know whether or not we should be happy about this. On a scale from 0 to 100, rarely missing by more than 24 seems OK to me, but not terrific.

- The **degrees of freedom** here are the same as the denominator degrees of freedom in the ANOVA to follow. The calculation is $n - k$, where $n =$ the number of observations and k is the number of coefficients estimated by the regression (including the intercept and any slopes).

– Here, there are 36 observations in the model, and we fit $k = 2$ coefficients; the slope and the intercept, as in any simple regression model, so df = 36 - 2 = 34.

The multiple R-squared value is usually just referred to as R-squared.

- This is interpreted as the proportion of variation in the outcome variable that has been accounted for by our regression model.

    – Here, we've accounted for just under 13% of the variation in % Recovery using Dose.

- The R in multiple R-squared is the Pearson correlation of `recov.score` and `dose`, which in this case is 0.3595.

    – Squaring this value gives the R-squared for this simple regression.
    – (0.3595)^2 = 0.129

R-squared is greedy.

- R-squared will always suggest that we make our models as big as possible, often including variables of dubious predictive value.
- As a result, there are various methods for adjusting or penalizing R-squared so that we wind up with smaller models.
- The **adjusted R-squared** is often a useful way to compare multiple models for the same response.

    – $R^2_{adj} = 1 - \frac{(1-R^2)(n-1)}{n-k}$, where $n =$ the number of observations and k is the number of coefficients estimated by the regression (including the intercept and any slopes).
    – So, in this case, $R^2_{adj} = 1 - \frac{(1-0.1293)(35)}{34} = 0.1037$
    – The adjusted R-squared value is not, technically, a proportion of anything, but it is comparable across models for the same outcome.
    – The adjusted R-squared will always be less than the (unadjusted) R-squared.

16.10.7 ANOVA F test

5. The last part of the standard summary of a regression model is the overall ANOVA F test.

The hypotheses for this test are:

- H0: Each of the coefficients in the model (other than the intercept) has $\beta = 0$ vs.
- HA: At least one regression slope has $\beta \neq 0$

Since we are doing a simple regression with just one predictor, the ANOVA F test hypotheses are exactly the same as the t test for dose:

- H0: The slope for `dose` has $\beta = 0$ vs.
- HA: The slope for `dose` has $\beta \neq 0$

In this case, we have an F statistic of 5.05 on 1 and 34 degrees of freedom, yielding $p = 0.03$

This provides some evidence that "something" in our model (here, `dose` is the only predictor) predicts the outcome to a degree beyond that easily attributed to chance alone. This is not actually surprising, nor is it especially interesting. The confidence interval for the slope is definitely more interesting than this.

- In *simple regression* (regression with only one predictor), the t test for the slope (`dose`) always provides the same p value as the ANOVA F test.

    - The F test statistic in a *simple regression* is always by definition just the square of the slope's t test statistic.
    - Here, F = 5.047, and this is the square of t = 2.247 from the Coefficients output

This test is basically just a combination of the R-squared value (13%) and the sample size. We don't learn much from it that's practically interesting or useful.

16.11 Viewing the complete ANOVA table

We can obtain the complete ANOVA table associated with this particular model, and the details behind this F test using the `anova` function:

```
anova(lm(recov.score ~ dose, data = hydrate))
```

```
Analysis of Variance Table

Response: recov.score
          Df Sum Sq Mean Sq F value  Pr(>F)
dose       1  752.2  752.15  5.0473 0.03127 *
Residuals 34 5066.7  149.02
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- The R-squared for our regression model is equal to the η^2 for this ANOVA model.

    - If we divide SS(dose) = 752.2 by the total sum of squares (752.2 + 5066.7), we'll get the multiple R-squared [0.1293]

- Note that this is *not* the same ANOVA model we would get if we treated `dose` as a factor with seven levels, rather than as a quantitative variable.

16.12 Using `glance` to summarize the model's fit

When applied to a linear model, the `glance` function from the `broom` package summarizes 12 characteristics of the model's fit.

Let's look at the eight of these that we've already addressed.

```
glance(m1) |> select(r.squared:df, df.residual, nobs) |>
  kbl(digits = c(3, 3, 1, 2, 3, 0, 0, 0))
```

r.squared	adj.r.squared	sigma	statistic	p.value	df	df.residual	nobs
0.129	0.104	12.2	5.05	0.031	1	34	36

- We've discussed the R-square value, shown in `r.squared`.
- We've also discussed the adjusted R-square value, in `adj.r.squared`
- `sigma` is the residual standard error.
- `statistic` is the ANOVA F statistic.
- `p.value` is the p value associated with the ANOVA F statistic.
- `df` is the numerator degrees of freedom (here, the `df` associated with `dose`) for the ANOVA test associated with this model.
- `df.residual` is the denominator degrees of freedom (here the `df` associated with `residual`) for that same ANOVA test.
- Remember that the F-statistic at the bottom of the summary output provides these last four statistics, as well.
- `nobs` is the number of observations (rows) used to fit the model.

Now, let's look at the remaining four summaries:

```
glance(m1) |> select(logLik:deviance) |>
  kbl(digits = 1)
```

logLik	AIC	BIC	deviance
-140.1	286.3	291	5066.7

- `logLik` is the log-likelihood value for the model, and is most commonly used for a model (like the ordinary least squares model fit by `lm` that is fit using the method of maximum likelihood). Thus, the log-likelihood value will be maximized in this fit.
- `AIC` is the Akaike Information Criterion for the model. When comparing models fitted by maximum likelihood to the same outcome variable (using the same transformation, for example), the smaller the AIC, the better the fit.

- **BIC** is the Bayes Information Criterion for the model. When comparing models fitted by maximum likelihood to the same outcome variable (using the same transformation, for example), the smaller the BIC, the better the fit. BIC often prefers models with fewer coefficients to estimate than does AIC.

    – **AIC** and **BIC** can be estimated using several different approaches in R, but we'll need to use the same one across multiple models if we're comparing the results, because the concepts are only defined up to a constant.

- **deviance** is the fitted model's deviance, a measure of lack of fit. It is a generalization of the residual sum of squares seen in the ANOVA table, and takes the same value in the case of a simple linear regression model fit with **lm** as we have here. For some generalized linear models, we'll use this for hypothesis testing, just as the ANOVA table does in the linear model case.

16.13 Plotting Residuals vs. Fitted Values

To save the residuals and predicted (fitted) values from this simple regression model, we can use the **resid** and **fitted** commands, respectively, or we can use the **augment** function in the **broom** package to obtain a tidy data set containing these objects and others.

```
aug_m1 <- augment(m1)

ggplot(aug_m1, aes(x = .fitted, y = .resid)) +
    geom_point() +
    geom_smooth(method = "loess", formula = y ~ x, se = F) +
    geom_smooth(method = "lm", formula = y ~ x, se = F, col = "red") +
    labs(title = "Residuals vs. Fitted values for Model m1")
```

Residuals vs. Fitted values for Model m1
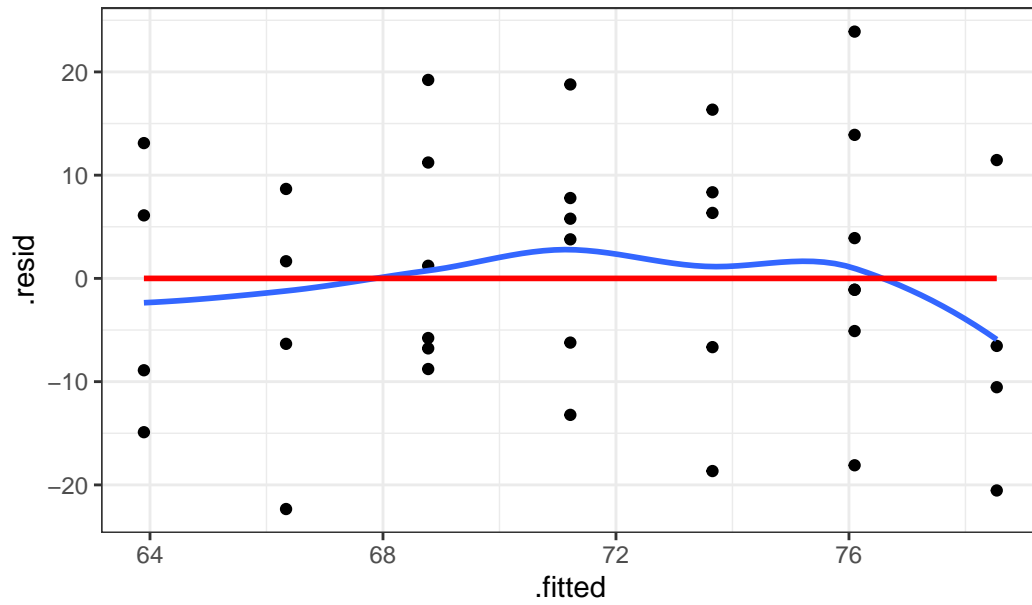
We can also obtain a plot of residuals vs. fitted values for `m1` using the following code from base R.

```r
plot(m1, which = 1)
```

Residuals vs Fitted

lm(recov.score ~ dose)
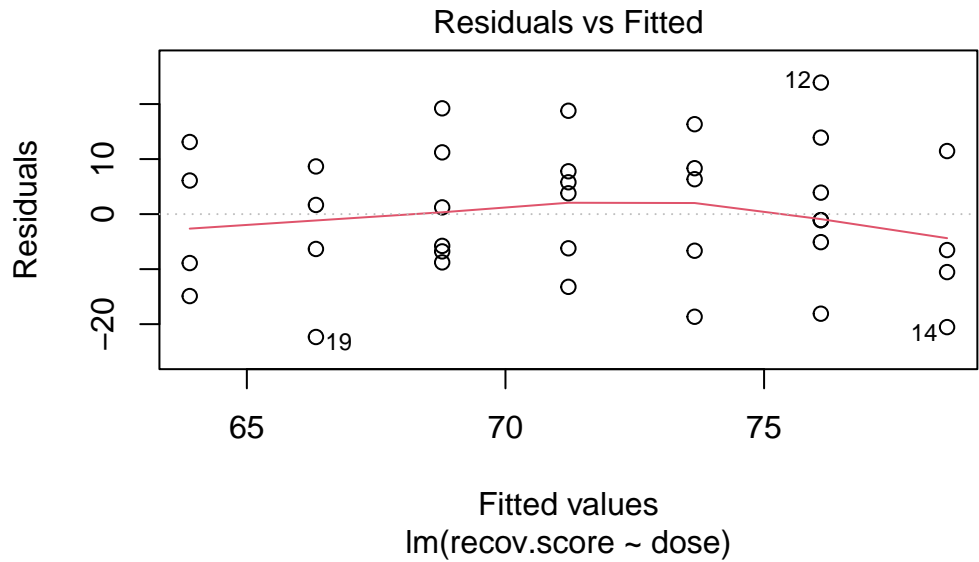
We hope in this plot to see a generally random scatter of points, perhaps looking like a "fuzzy football". Since we only have seven possible `dose` values, we obtain only seven distinct predicted values, which explains the seven vertical lines in the plot. Here, the smooth red line indicates a gentle curve, but no evidence of a strong curve, or any other regular pattern in this residual plot.

17 The WCGS

17.1 Setup: Packages Used Here

```
knitr::opts_chunk$set(comment = NA)

library(broom)
library(GGally)
library(ggridges)
library(janitor)
library(kableExtra)
library(patchwork)
library(tidyverse)

theme_set(theme_bw())
```

We will also use the `favstats` function from the `mosaic` package, even though I won't load `mosaic` here.

17.2 The Western Collaborative Group Study (`wcgs`)

Vittinghoff et al. (2012) explore data from the Western Collaborative Group Study (WCGS) in some detail[1]. We'll touch lightly on some key issues in this Chapter.

> The Western Collaborative Group Study (WCGS) was designed to test the hypothesis that the so-called Type A behavior pattern (TABP) - "characterized particularly by excessive drive, aggressiveness, and ambition, frequently in association with a relatively greater preoccupation with competitive activity, vocational deadlines, and similar pressures" - is a cause of coronary heart disease (CHD). Two additional goals, developed later in the study, were (1) to investigate the comparability of formulas developed in WCGS and in the Framingham Study (FS) for prediction of CHD risk, and (2) to determine how addition of TABP to an existing

---

[1]For more on the WCGS, you might look at http://www.epi.umn.edu/cvdepi/study-synopsis/western-collaborative-group-study/

multivariate prediction formula affects ability to select subjects for intervention programs.

The study enrolled over 3,000 men ages 39-59 who were employed in San Francisco or Los Angeles, during 1960 and 1961.

In the code chunk below, after importing the data and creating a tibble with `read_csv`, I used `mutate(across(where(is.character), as_factor)` to convert all variables containing character data into factors.

```r
wcgs <- read_csv("data/wcgs.csv") |>
    mutate(across(where(is.character), as_factor))

wcgs
```

```
# A tibble: 3,154 x 22
       id   age agec   height weight lnwght wghtcat    bmi   sbp lnsbp   dbp  chol
    <dbl> <dbl> <fct>   <dbl>  <dbl>  <dbl> <fct>    <dbl> <dbl> <dbl> <dbl> <dbl>
 1   2343    50 46-50      67    200   5.30 170-200   31.3   132  4.88    90   249
 2   3656    51 51-55      73    192   5.26 170-200   25.3   120  4.79    74   194
 3   3526    59 56-60      70    200   5.30 170-200   28.7   158  5.06    94   258
 4  22057    51 51-55      69    150   5.01 140-170   22.1   126  4.84    80   173
 5  12927    44 41-45      71    160   5.08 140-170   22.3   126  4.84    80   214
 6  16029    47 46-50      64    158   5.06 140-170   27.1   116  4.75    76   206
 7   3894    40 35-40      70    162   5.09 140-170   23.2   122  4.80    78   190
 8  11389    41 41-45      70    160   5.08 140-170   23.0   130  4.87    84   212
 9  12681    50 46-50      71    195   5.27 170-200   27.2   112  4.72    70   130
10  10005    43 41-45      68    187   5.23 170-200   28.4   120  4.79    80   233
# i 3,144 more rows
# i 10 more variables: behpat <fct>, dibpat <fct>, smoke <fct>, ncigs <dbl>,
#   arcus <dbl>, chd69 <fct>, typchd69 <dbl>, time169 <dbl>, t1 <dbl>,
#   uni <dbl>
```

Here, we have 3154 rows (subjects) and 22 columns (variables).

17.2.1 Structure of `wcgs`

We can specify the (sometimes terrible) variable names, through the `names` function, or we can add other elements of the structure, so that we can identify items of particular interest.

```r
str(wcgs)
```

```
tibble [3,154 x 22] (S3: tbl_df/tbl/data.frame)
 $ id      : num [1:3154] 2343 3656 3526 22057 12927 ...
 $ age     : num [1:3154] 50 51 59 51 44 47 40 41 50 43 ...
 $ agec    : Factor w/ 5 levels "46-50","51-55",..: 1 2 3 2 4 1 5 4 1 4 ...
 $ height  : num [1:3154] 67 73 70 69 71 64 70 70 71 68 ...
 $ weight  : num [1:3154] 200 192 200 150 160 158 162 160 195 187 ...
 $ lnwght  : num [1:3154] 5.3 5.26 5.3 5.01 5.08 ...
 $ wghtcat : Factor w/ 4 levels "170-200","140-170",..: 1 1 1 2 2 2 2 2 1 1 ...
 $ bmi     : num [1:3154] 31.3 25.3 28.7 22.1 22.3 ...
 $ sbp     : num [1:3154] 132 120 158 126 126 116 122 130 112 120 ...
 $ lnsbp   : num [1:3154] 4.88 4.79 5.06 4.84 4.84 ...
 $ dbp     : num [1:3154] 90 74 94 80 80 76 78 84 70 80 ...
 $ chol    : num [1:3154] 249 194 258 173 214 206 190 212 130 233 ...
 $ behpat  : Factor w/ 4 levels "A1","A2","B3",..: 1 1 1 1 1 1 1 1 1 1 1 ...
 $ dibpat  : Factor w/ 2 levels "Type A","Type B": 1 1 1 1 1 1 1 1 1 1 1 ...
 $ smoke   : Factor w/ 2 levels "Yes","No": 1 1 2 2 2 1 2 1 2 1 ...
 $ ncigs   : num [1:3154] 25 25 0 0 0 80 0 25 0 25 ...
 $ arcus   : num [1:3154] 1 0 1 1 0 0 0 0 1 0 ...
 $ chd69   : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 1 ...
 $ typchd69: num [1:3154] 0 0 0 0 0 0 0 0 0 0 ...
 $ time169 : num [1:3154] 1367 2991 2960 3069 3081 ...
 $ t1      : num [1:3154] -1.63 -4.06 0.64 1.12 2.43 ...
 $ uni     : num [1:3154] 0.486 0.186 0.728 0.624 0.379 ...
```

17.2.2 Codebook for `wcgs`

This table was lovingly hand-crafted, and involved a lot of typing. We'll look for better ways in 432.

Name	Stored As	Type	Details (units, levels, etc.)
id	integer	(nominal)	ID #, nominal and uninteresting
age	integer	quantitative	age, in years - no decimal places
agec	factor (5)	(ordinal)	age: 35-40, 41-45, 46-50, 51-55, 56-60
height	integer	quantitative	height, in inches
weight	integer	quantitative	weight, in pounds
lnwght	number	quantitative	natural logarithm of weight
wghtcat	factor (4)	(ordinal)	wt: < 140, 140-170, 170-200, > 200
bmi	number	quantitative	body-mass index: 703 * weight in lb / (height in in)^2
sbp	integer	quantitative	systolic blood pressure, in mm Hg
lnsbp	number	quantitative	natural logarithm of sbp
dbp	integer	quantitative	diastolic blood pressure, mm Hg

Name	Stored As	Type	Details (units, levels, etc.)
chol	integer	quantitative	total cholesterol, mg/dL
behpat	factor (4)	(nominal)	behavioral pattern: A1, A2, B3 or B4
dibpat	factor (2)	(binary)	behavioral pattern: A or B
smoke	factor (2)	(binary)	cigarette smoker: Yes or No
ncigs	integer	quantitative	number of cigarettes smoked per day
arcus	integer	(nominal)	arcus senilis present (1) or absent (0)
chd69	factor (2)	(binary)	CHD event: Yes or No
typchd69	integer	(4 levels)	event: 0 = no CHD, 1 = MI or SD, 2 = silent MI, 3 = angina
time169	integer	quantitative	follow-up time in days
t1	number	quantitative	heavy-tailed (random draws)
uni	number	quantitative	light-tailed (random draws)

17.2.3 Quick Summary

```
summary(wcgs)
```

```
      id               age            agec           height          weight
 Min.   : 2001   Min.   :39.00   46-50: 750   Min.   :60.00   Min.   : 78
 1st Qu.: 3741   1st Qu.:42.00   51-55: 528   1st Qu.:68.00   1st Qu.:155
 Median :11406   Median :45.00   56-60: 242   Median :70.00   Median :170
 Mean   :10478   Mean   :46.28   41-45:1091   Mean   :69.78   Mean   :170
 3rd Qu.:13115   3rd Qu.:50.00   35-40: 543   3rd Qu.:72.00   3rd Qu.:182
 Max.   :22101   Max.   :59.00                Max.   :78.00   Max.   :320


    lnwght           wghtcat          bmi             sbp             lnsbp
 Min.   :4.357   170-200:1171   Min.   :11.19   Min.   : 98.0   Min.   :4.585
 1st Qu.:5.043   140-170:1538   1st Qu.:22.96   1st Qu.:120.0   1st Qu.:4.787
 Median :5.136   > 200  : 213   Median :24.39   Median :126.0   Median :4.836
 Mean   :5.128   < 140  : 232   Mean   :24.52   Mean   :128.6   Mean   :4.850
 3rd Qu.:5.204                  3rd Qu.:25.84   3rd Qu.:136.0   3rd Qu.:4.913
 Max.   :5.768                  Max.   :38.95   Max.   :230.0   Max.   :5.438


      dbp             chol         behpat        dibpat       smoke
 Min.   : 58.00   Min.   :103.0   A1: 264   Type A:1589   Yes:1502
 1st Qu.: 76.00   1st Qu.:197.2   A2:1325   Type B:1565   No :1652
 Median : 80.00   Median :223.0   B3:1216
 Mean   : 82.02   Mean   :226.4   B4: 349
 3rd Qu.: 86.00   3rd Qu.:253.0
```

```
 Max.   :150.00    Max.    :645.0
                   NA's    :12
     ncigs            arcus          chd69          typchd69            time169
 Min.   : 0.0   Min.   :0.0000   No :2897   Min.   :0.0000   Min.   :  18
 1st Qu.: 0.0   1st Qu.:0.0000   Yes: 257   1st Qu.:0.0000   1st Qu.:2842
 Median : 0.0   Median :0.0000              Median :0.0000   Median :2942
 Mean   :11.6   Mean   :0.2985              Mean   :0.1363   Mean   :2684
 3rd Qu.:20.0   3rd Qu.:1.0000              3rd Qu.:0.0000   3rd Qu.:3037
 Max.   :99.0   Max.   :1.0000              Max.   :3.0000   Max.   :3430
                NA's   :2
       t1                uni
 Min.   :-47.43147   Min.   :0.0007097
 1st Qu.: -1.00337   1st Qu.:0.2573755
 Median :  0.00748   Median :0.5157779
 Mean   : -0.03336   Mean   :0.5052159
 3rd Qu.:  0.97575   3rd Qu.:0.7559902
 Max.   : 47.01623   Max.   :0.9994496
 NA's   :39
```

For a more detailed description, we might consider `Hmisc::describe`, `psych::describe`, `mosaic::inspect`, etc., as we've done (for instance) in Chapter 3 and Chapter 7.

17.3 Are the SBPs Normally Distributed?

Consider the question of whether the distribution of the systolic blood pressure results is well-approximated by the Normal, where we'll make use of tools based on our discussion in Chapter 11.

```r
res <- mosaic::favstats(~ sbp, data = wcgs)
```

```
Registered S3 method overwritten by 'mosaic':
  method                           from
  fortify.SpatialPolygonsDataFrame ggplot2
```

```r
bin_w <- 5 # specify binwidth

ggplot(wcgs, aes(x = sbp)) +
    geom_histogram(binwidth = bin_w,
                   fill = "orchid",
                   col = "blue") +
```

```
    stat_function(
        fun = function(x) dnorm(x, mean = res$mean,
                                sd = res$sd) *
            res$n * bin_w,
        col = "navy") +
    labs(title = "Systolic BP for `wcgs` subjects",
     x = "Systolic BP (mm Hg)", y = "",
     caption = "Superimposed Normal model")
```

Systolic BP for `wcgs` subjects
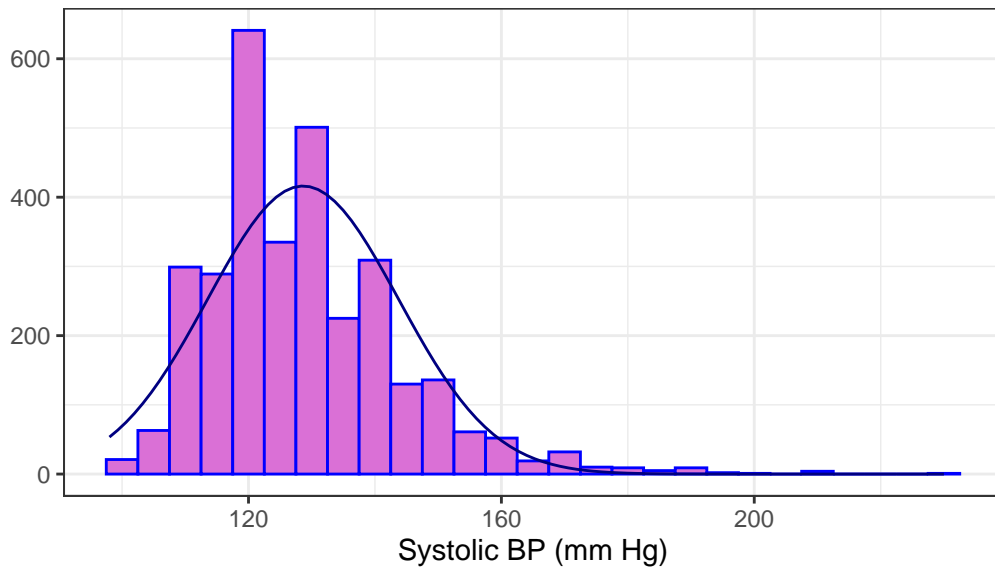
Superimposed Normal model

Since the data contain both `sbp` and `lnsbp` (its natural logarithm), let's compare them. Note that in preparing the graph, we'll need to change the location for the text annotation.

```
res <- mosaic::favstats(~ lnsbp, data = wcgs)
bin_w <- 0.05 # specify binwidth

ggplot(wcgs, aes(x = lnsbp)) +
    geom_histogram(binwidth = bin_w,
                   fill = "orange",
                   col = "blue") +
    stat_function(
        fun = function(x) dnorm(x, mean = res$mean,
```

```
                                        sd = res$sd) *
                res$n * bin_w,
            col = "navy") +
    labs(title = "ln(Systolic BP) for `wcgs` subjects",
     x = "ln(Systolic BP)", y = "",
     caption = "Superimposed Normal model")
```

ln(Systolic BP) for `wcgs` subjects
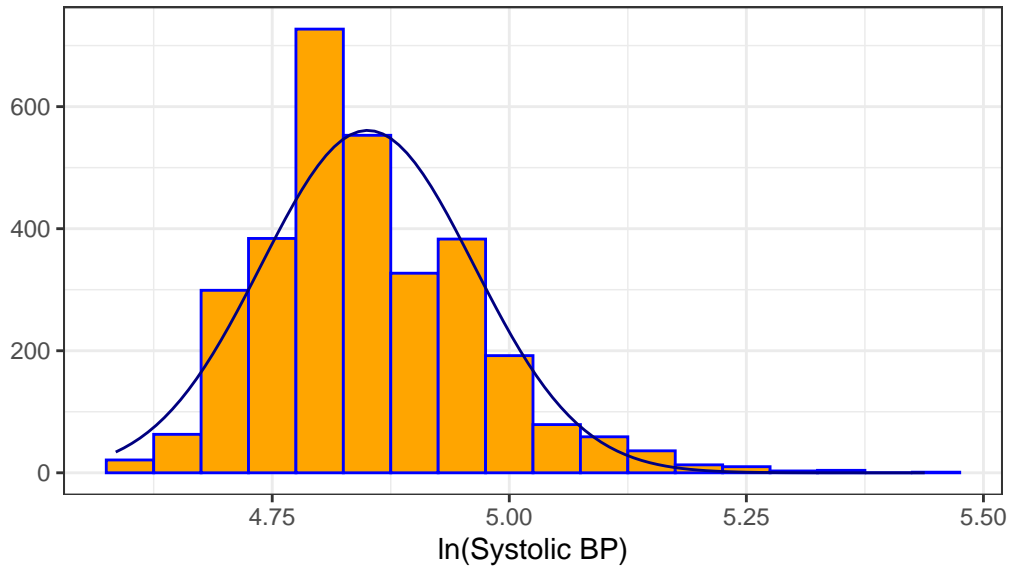
Superimposed Normal model

We can also look at Normal Q-Q plots, for instance...

```
p1 <- ggplot(wcgs, aes(sample = sbp)) +
    geom_qq(color = "orchid") +
    geom_qq_line(color = "red") +
    labs(y = "Ordered SBP", title = "sbp in wcgs")

p2 <- ggplot(wcgs, aes(sample = lnsbp)) +
    geom_qq(color = "orange") +
    geom_qq_line(color = "red") +
    labs(y = "Ordered ln(SBP)", title = "ln(sbp) in wcgs")

## next step requires library(patchwork)
```

```
p1 + p2 +
    plot_annotation(title = "Normal Q-Q plots of SBP and ln(SBP) in wcgs")
```

Normal Q–Q plots of SBP and ln(SBP) in wcgs
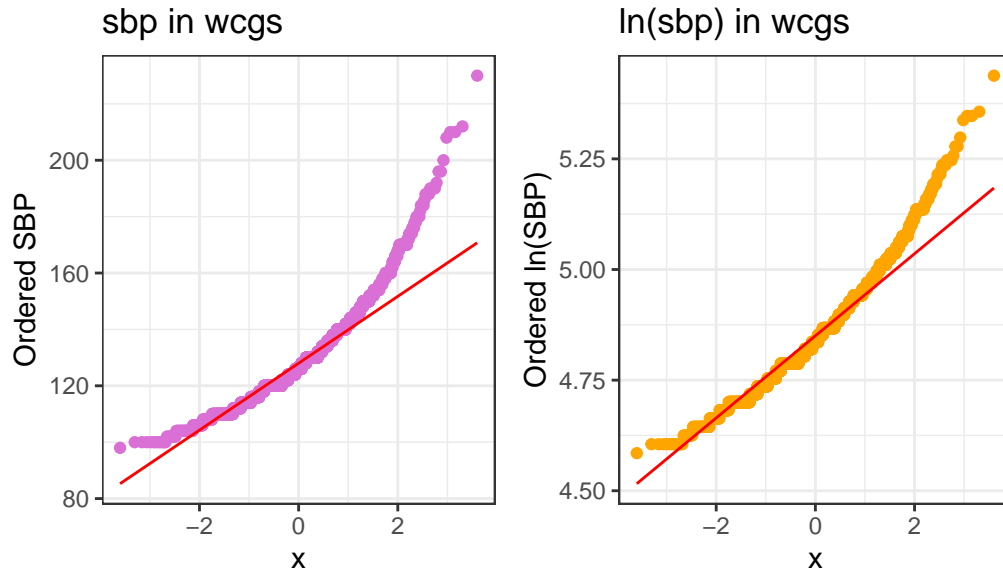
There's at best a small improvement from `sbp` to `lnsbp` in terms of approximation by a Normal distribution.

17.4 Identifying and Describing SBP outliers

It looks like there's an outlier (or a series of them) in the SBP data.

```
ggplot(wcgs, aes(x = "", y = sbp)) +
    geom_violin() +
    geom_boxplot(width = 0.3, fill = "royalblue",
                 outlier.color = "royalblue") +
    labs(title = "Boxplot with Violin of SBP in `wcgs` data",
         y = "Systolic Blood Pressure (mm Hg)",
         x = "") +
    coord_flip()
```

354

Boxplot with Violin of SBP in `wcgs` data
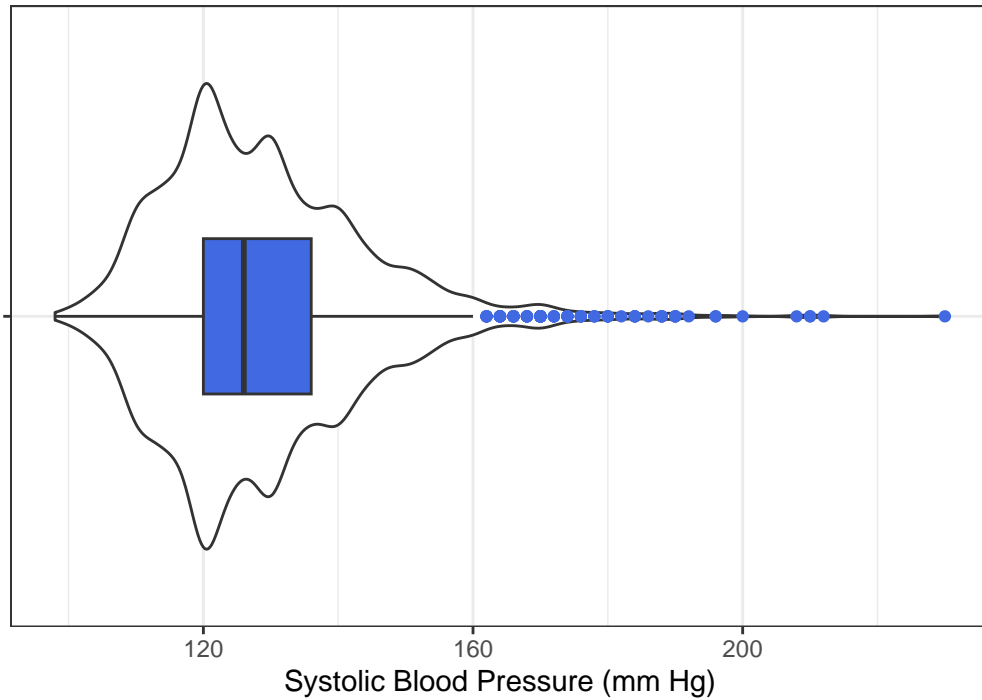

```
mosaic::favstats(wcgs$sbp)
```

```
 min  Q1 median  Q3 max     mean        sd    n missing
  98 120    126 136 230 128.6328 15.11773 3154       0
```

```
Hmisc::describe(wcgs$sbp)
```

```
wcgs$sbp
        n  missing distinct     Info     Mean      Gmd      .05      .10
     3154        0       62    0.996    128.6    16.25      110      112
      .25      .50      .75      .90      .95
      120      126      136      148      156

lowest :  98 100 102 104 106, highest: 200 208 210 212 230
```

The maximum value here is 230, and is clearly the most extreme value in the data set. One way to gauge this is to describe that observation's **Z score**, the number of standard deviations

away from the mean that the observation falls. Here, the maximum value, 230 is 6.71 standard deviations above the mean, and thus has a Z score of 6.7.

A negative Z score would indicate a point below the mean, while a positive Z score indicates, as we've seen, a point above the mean. The minimum systolic blood pressure, 98 is 2.03 standard deviations *below* the mean, so it has a Z score of -2.

Recall that the Empirical Rule (described in Chapter 11) suggests that if a variable follows a Normal distribution, it would have approximately 95% of its observations falling inside a Z score of (-2, 2), and 99.74% falling inside a Z score range of (-3, 3). Do the systolic blood pressures appear Normally distributed?

17.5 Does Weight Category Relate to SBP?

The data are collected into four groups based on the subject's weight (in pounds).

```
ggplot(wcgs, aes(x = wghtcat, y = sbp)) +
    geom_violin() +
    geom_boxplot(aes(fill = wghtcat), width = 0.3, notch = TRUE) +
    scale_fill_viridis_d() +
    guides(fill = "none") +
    labs(title = "Boxplot of Systolic BP by Weight Category in WCGS",
         x = "Weight Category", y = "Systolic Blood Pressure")
```

Boxplot of Systolic BP by Weight Category in WCGS
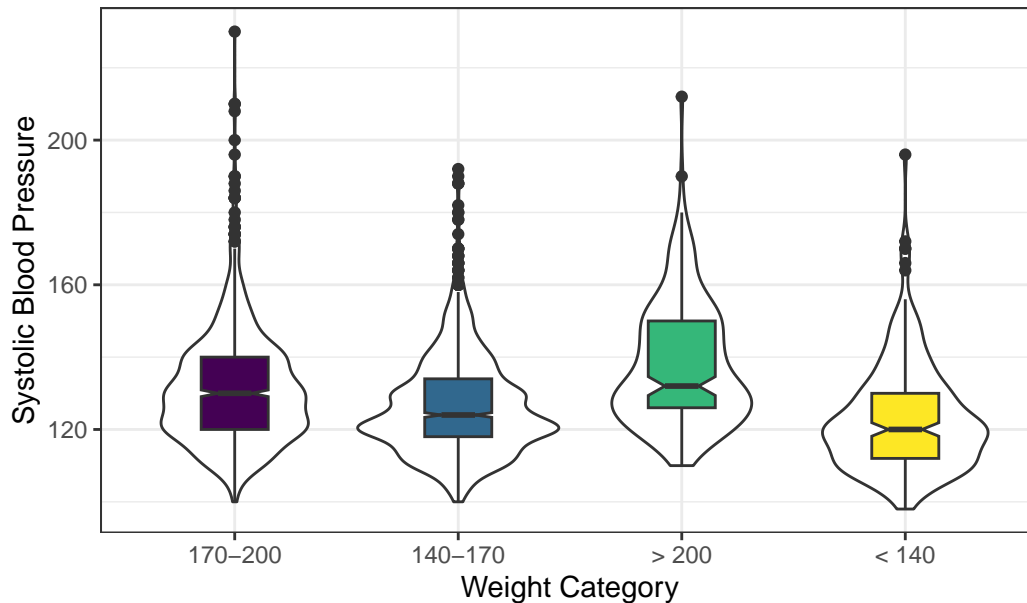
17.6 Re-Leveling a Factor

Well, that's not so good. We really want those weight categories (the *levels*) to be ordered more sensibly.

```
wcgs |> tabyl(wghtcat)
```

```
 wghtcat     n     percent
 170-200  1171 0.37127457
 140-170  1538 0.48763475
   > 200   213 0.06753329
   < 140   232 0.07355739
```

Like all *factor* variables in R, the categories are specified as levels. We want to change the order of the levels in a new version of this factor variable so they make sense. There are multiple ways to do this, but I prefer the `fct_relevel` function from the `forcats` package (part of the tidyverse.) Which order is more appropriate?

I'll add a new variable to the `wcgs` data called `weight_f` that relevels the `wghtcat` data.

```r
wcgs <- wcgs |>
    mutate(weight_f = fct_relevel(wghtcat, "< 140", "140-170", "170-200", "> 200"))

wcgs |> tabyl(weight_f)
```

```
weight_f    n    percent
   < 140  232 0.07355739
 140-170 1538 0.48763475
 170-200 1171 0.37127457
   > 200  213 0.06753329
```

For more on the **forcats** package, check out Hadley Wickham and Grolemund (2023), especially its section on Factors.

17.6.1 SBP by Weight Category

```r
ggplot(wcgs, aes(x = weight_f, y = sbp, fill = weight_f)) +
    geom_boxplot(notch = TRUE) +
    scale_fill_viridis_d() +
    guides(fill = "none") +
    labs(title = "Systolic Blood Pressure by Reordered Weight Category in WCGS",
         x = "Weight Category", y = "Systolic Blood Pressure")
```

Systolic Blood Pressure by Reordered Weight Category in WC
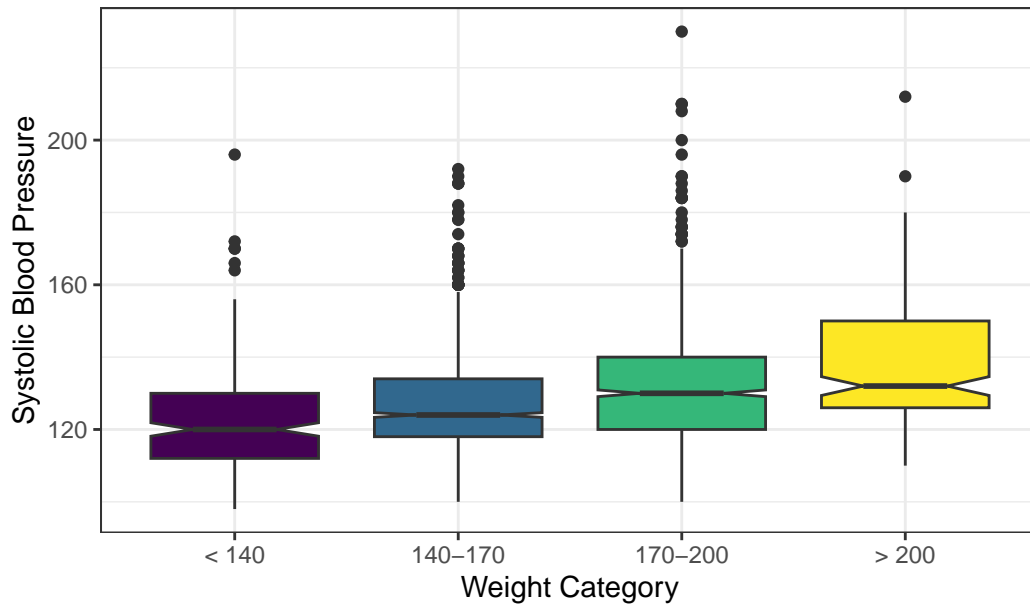
We might see some details well with a **ridgeline plot**, too.

```
ggplot(wcgs, aes(x = sbp, y = weight_f, fill = weight_f, height = ..density..)) +
    ggridges::geom_density_ridges(scale = 2) +
    scale_fill_viridis_d() +
    guides(fill = "none") +
    labs(title = "SBP by Weight Category (wcgs)",
         x = "Systolic Blood Pressure",
         y = "Weight Category")
```

```
Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
i Please use `after_stat(density)` instead.
```

```
Picking joint bandwidth of 3.74
```

SBP by Weight Category (wcgs)
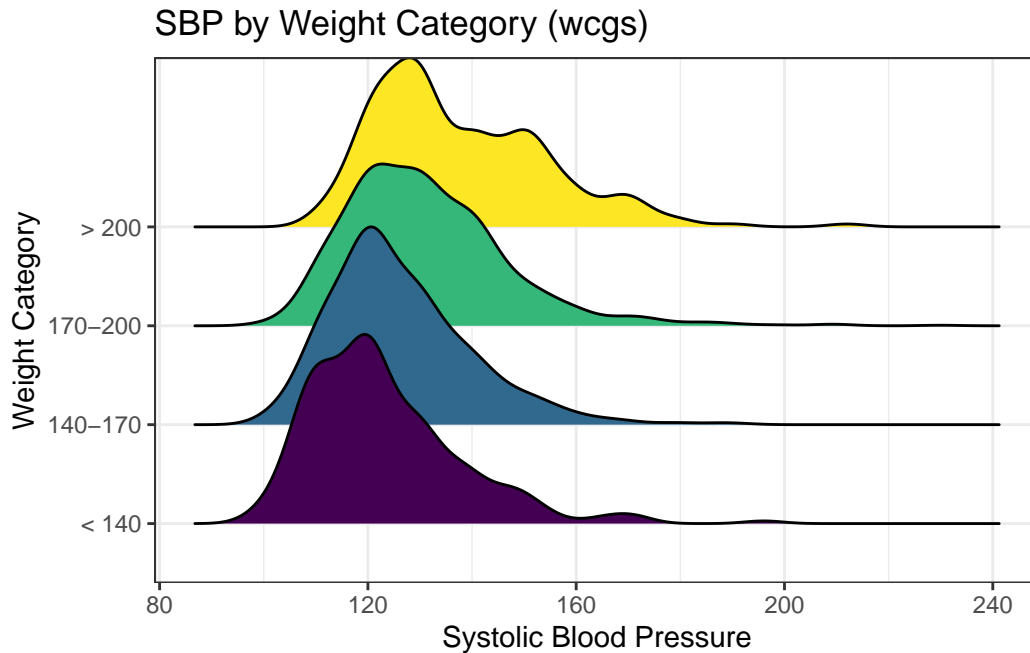
As the plots suggest, patients in the heavier groups generally had higher systolic blood pressures.

```
mosaic::favstats(sbp ~ weight_f, data = wcgs)
```

```
  weight_f min  Q1 median  Q3 max     mean       sd    n missing
1    < 140  98 112    120 130 196 123.1379 14.73394  232       0
2  140-170 100 118    124 134 192 126.2939 13.65294 1538       0
3  170-200 100 120    130 140 230 131.1136 15.57024 1171       0
4    > 200 110 126    132 150 212 137.8685 16.75522  213       0
```

17.7 Are Weight and SBP Linked?

Let's build a scatter plot of SBP (Outcome) by Weight (Predictor), rather than breaking down into categories.

```
ggplot(wcgs, aes(x = weight, y = sbp)) +
    geom_point(size=3, shape=1, color="forestgreen") + ## default size = 2
    geom_smooth(method = "lm", se = FALSE, col = "red", formula = y ~ x) +
    geom_smooth(method = "loess", col = "blue", formula = y ~ x) +
```

```
ggtitle("SBP vs. Weight in 3,154 WCGS Subjects")
```
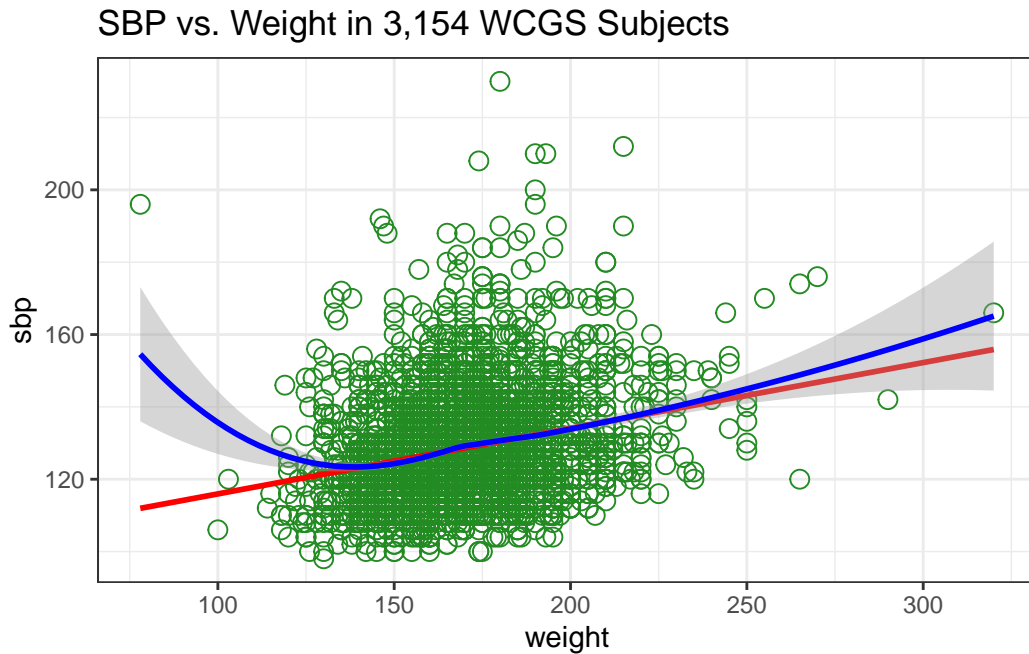

- The mass of the data is hidden from us - showing 3154 points in one plot can produce little more than a blur where there are lots of points on top of each other.
- Here the least squares regression line (in red), and loess scatterplot smoother, (in blue) can help.

The relationship between systolic blood pressure and weight appears to be very close to linear, but of course there is considerable scatter around that generally linear relationship. It turns out that the Pearson correlation of these two variables is 0.253.

17.8 SBP and Weight by Arcus Senilis groups?

An issue of interest to us will be to assess whether the SBP-Weight relationship we see above is similar among subjects who have been diagnosed with arcus senilis and those who have not.

> Arcus senilis is an old age syndrome where there is a white, grey, or blue opaque ring in the corneal margin (peripheral corneal opacity), or white ring in front of the periphery of the iris. It is present at birth but then fades; however, it is quite commonly present in the elderly. It can also appear earlier in life as a result of hypercholesterolemia.

Wikipedia article on Arcus Senilis, retrieved 2017-08-15

Let's start with a quick look at the `arcus` data.

```
wcgs |> tabyl(arcus)
```

```
 arcus    n       percent valid_percent
     0 2211 0.7010145847     0.7014594
     1  941 0.2983512999     0.2985406
    NA    2 0.0006341154            NA
```

We have 2 missing values, so we probably want to do something about that before plotting the data, and we may also want to create a factor variable with more meaningful labels than 1 (which means yes, arcus senilis is present) and 0 (which means no, it isn't.)

```
wcgs <- wcgs |>
    mutate(arcus_f = fct_recode(factor(arcus),
                                "Arcus senilis" = "1",
                                "No arcus senilis" = "0"),
           arcus_f = fct_relevel(arcus_f, "Arcus senilis"))

wcgs |> tabyl(arcus_f, arcus)
```

```
         arcus_f    0   1 NA_
   Arcus senilis    0 941   0
No arcus senilis 2211   0   0
            <NA>    0   0   2
```

Let's build a version of the `wcgs` data that eliminates all missing data in the variables of immediate interest, and then plot the SBP-weight relationship in groups of patients with and without arcus senilis.

```
wcgs_temp <- wcgs |>
    filter(complete.cases(arcus_f, sbp, weight))

ggplot(wcgs_temp, aes(x = weight, y = sbp, group = arcus_f)) +
    geom_point(shape = 1) +
    geom_smooth(method = "lm", col = "red", formula = y ~ x) +
    geom_smooth(method = "loess", se = FALSE, col = "blue", formula = y ~ x) +
    labs(title = "SBP vs. Weight by Arcus Senilis status",
```

term	estimate	std.error	statistic	p.value
(Intercept)	95.92	2.56	37.54	0
weight	0.19	0.01	12.77	0

```
        caption = "3,152 WCGS with known arcus senilis status") +
    facet_wrap(~ arcus_f)
```

SBP vs. Weight by Arcus Senilis status

3,152 WCGS with known arcus senilis status
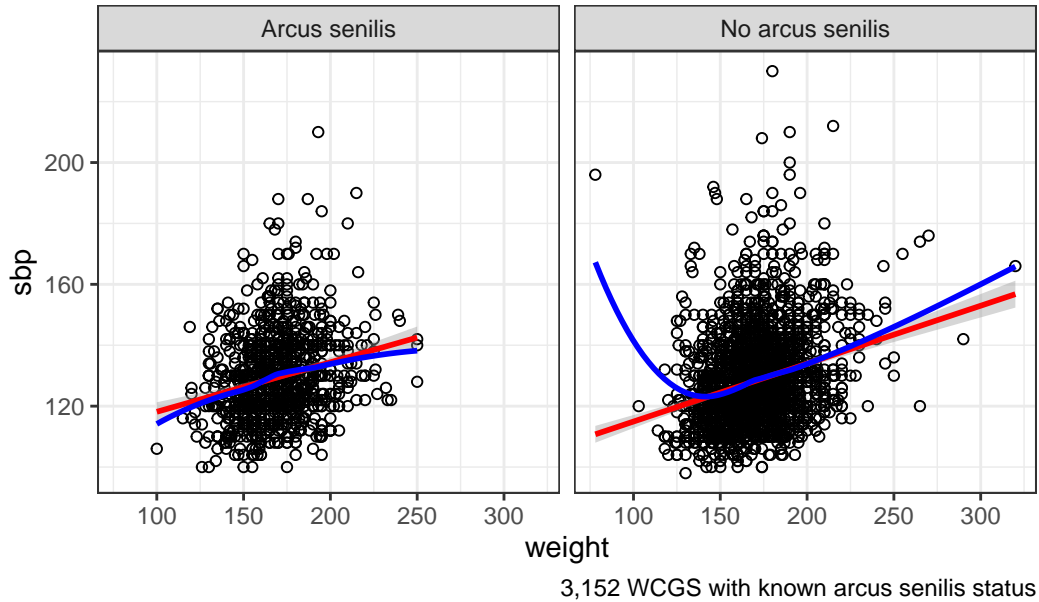
17.9 Linear Model for SBP-Weight Relationship: subjects without Arcus Senilis

```
model.noarcus <-
    lm(sbp ~ weight, data = filter(wcgs, arcus == 0))

tidy(model.noarcus) |>
    kbl(digits = 2) |>
    kable_styling(full_width = FALSE)
```

r.squared	adj.r.squared	sigma	statistic	p.value	AIC
0.069	0.068	14.8	163	0	18194

```
glance(model.noarcus) |>
    select(r.squared:p.value, AIC) |>
    kbl(digits = c(3, 3, 1, 1, 3, 0)) |>
    kable_styling(full_width = FALSE)
```

```
summary(model.noarcus)
```

```
Call:
lm(formula = sbp ~ weight, data = filter(wcgs, arcus == 0))

Residuals:
    Min      1Q  Median      3Q     Max
-29.011 -10.251  -2.447   7.553  99.848

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  95.9219     2.5552   37.54   <2e-16 ***
weight        0.1902     0.0149   12.77   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 14.8 on 2209 degrees of freedom
Multiple R-squared:  0.0687,    Adjusted R-squared:  0.06828
F-statistic:   163 on 1 and 2209 DF,  p-value: < 2.2e-16
```

The linear model for the 2211 patients without Arcus Senilis has R-squared = 6.87%.

- The regression equation is 95.92 - 0.19 weight, for those patients without Arcus Senilis.

term	estimate	std.error	statistic	p.value
(Intercept)	101.88	3.76	27.13	0
weight	0.16	0.02	7.39	0

r.squared	adj.r.squared	sigma	statistic	p.value	AIC
0.055	0.054	14.2	54.6	0	7667

17.10 Linear Model for SBP-Weight Relationship: subjects with Arcus Senilis

```
model.witharcus <-
    lm(sbp ~ weight, data = filter(wcgs, arcus == 1))

tidy(model.witharcus) |>
    kbl(digits = 2) |>
    kable_styling(full_width = FALSE)

glance(model.witharcus) |>
    select(r.squared:p.value, AIC) |>
    kbl(digits = c(3, 3, 1, 1, 3, 0)) |>
    kable_styling(full_width = FALSE)

summary(model.witharcus)
```

```
Call:
lm(formula = sbp ~ weight, data = filter(wcgs, arcus == 1))

Residuals:
    Min      1Q  Median      3Q     Max
-30.335  -9.636  -1.961   7.973  76.738

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 101.87847    3.75572  27.126  < 2e-16 ***
weight        0.16261    0.02201   7.388 3.29e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 14.19 on 939 degrees of freedom
```

term	estimate	std.error	statistic	p.value
(Intercept)	95.92	2.52	38.00	0.00
weight	0.19	0.01	12.92	0.00
arcus	5.96	4.62	1.29	0.20
weight:arcus	-0.03	0.03	-1.02	0.31

r.squared	adj.r.squared	sigma	statistic	p.value	AIC
0.066	0.065	14.6	74.1	0	25861

```
Multiple R-squared:  0.05494,    Adjusted R-squared:  0.05393
F-statistic: 54.58 on 1 and 939 DF,  p-value: 3.29e-13
```

The linear model for the 941 patients with Arcus Senilis has R-squared = 5.49%.

- The regression equation is 101.88 - 0.163 weight, for those patients with Arcus Senilis.

17.11 Including Arcus Status in the model

```
model3 <- lm(sbp ~ weight * arcus, data = filter(wcgs, !is.na(arcus)))

tidy(model3) |>
    kbl(digits = 2) |>
    kable_styling(full_width = FALSE)

glance(model3) |>
    select(r.squared:p.value, AIC) |>
    kbl(digits = c(3, 3, 1, 1, 3, 0)) |>
    kable_styling(full_width = FALSE)

summary(model3)
```

```
Call:
lm(formula = sbp ~ weight * arcus, data = filter(wcgs, !is.na(arcus)))

Residuals:
    Min      1Q  Median      3Q     Max
-30.335 -10.152  -2.349   7.669  99.848
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  95.92190    2.52440  37.998   <2e-16 ***
weight        0.19017    0.01472  12.921   <2e-16 ***
arcus         5.95657    4.61972   1.289    0.197
weight:arcus -0.02756    0.02703  -1.019    0.308
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 14.62 on 3148 degrees of freedom
Multiple R-squared:  0.06595,   Adjusted R-squared:  0.06506
F-statistic: 74.09 on 3 and 3148 DF,  p-value: < 2.2e-16
```

The actual regression equation in this setting includes both weight, and an indicator variable (1 = yes, 0 = no) for arcus senilis status, and the product term combining weight and that 1/0 indicator. In 432, we'll spend substantial time and energy discussing these product terms, but we'll not do much of that in 431.

- Note the use of the product term `weight*arcus` in the setup of the model to allow both the slope of weight and the intercept term in the model to change depending on arcus senilis status.

    - For a patient who has arcus, the regression equation is SBP = 95.92 + 0.19 weight + 5.96 (1) - 0.028 weight (1) = 101.88 + 0.162 weight.
    - For a patient without arcus senilis, the regression equation is SBP = 95.92 + 0.19 weight + 5.96 (0) - 0.028 weight (0) = 95.92 + 0.19 weight.

The linear model including the interaction of weight and arcus to predict sbp for the 3152 patients with known Arcus Senilis status has R-squared = 6.6%. Again, we'll discuss interaction more substantially in 432.

17.12 Predictions from these Linear Models

What is our predicted SBP for a subject weighing 175 pounds?

How does that change if our subject weighs 200 pounds?

Recall that

- *Without* Arcus Senilis, linear model for SBP = 95.9 + 0.19 x weight
- *With* Arcus Senilis, linear model for SBP = 101.9 + 0.16 x weight

So the predictions for a 175 pound subject are:

- 95.9 + 0.19 x 175 = 129 mm Hg without Arcus Senilis, and

- 101.9 + 0.16 x 175 = 130 mm Hg with Arcus Senilis.

And thus, the predictions for a 200 pound subject are:

- 95.9 + 0.19 x 200 = 134 mm Hg without Arcus Senilis, and

- 101.9 + 0.16 x 200 = 134.4 mm Hg with Arcus Senilis.

17.13 Scatterplots with Facets Across a Categorical Variable

We can use facets in `ggplot2` to show scatterplots across the levels of a categorical variable, like `behpat`.

```
ggplot(wcgs, aes(x = weight, y = sbp, col = behpat)) +
    geom_point() +
    facet_wrap(~ behpat) +
    geom_smooth(method = "lm", se = FALSE,
                formula = y ~ x, col = "black") +
    guides(color = "none") +
    theme(strip.text = element_text(face="bold", size=rel(1.25), color="white"),
          strip.background = element_rect(fill="royalblue")) +
    labs(title = "Scatterplots of SBP vs. Weight within Behavior Pattern")
```

Scatterplots of SBP vs. Weight within Behavior Pattern
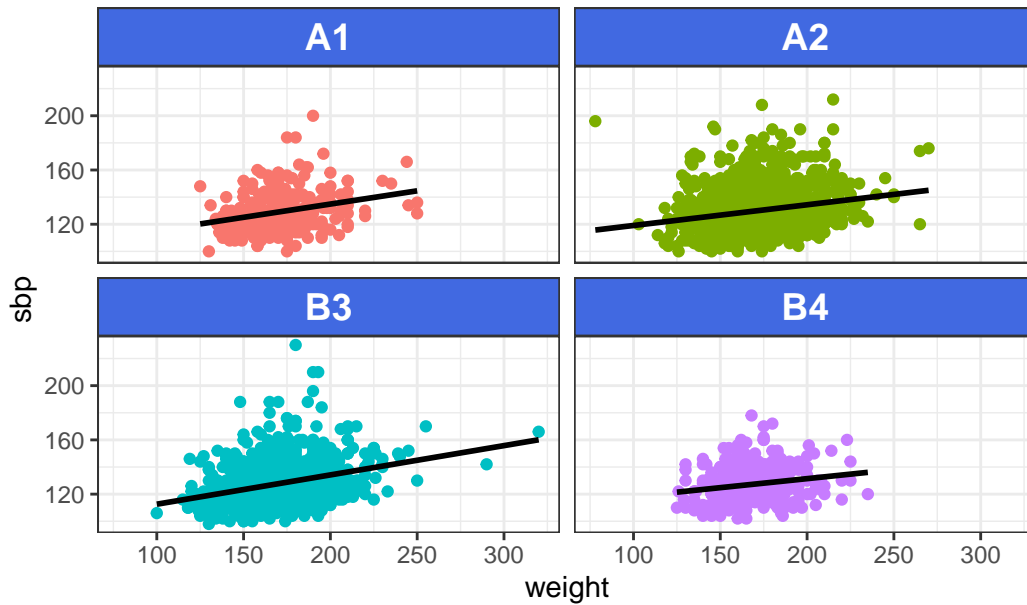
17.14 Scatterplot and Correlation Matrices

A **scatterplot matrix** can be very helpful in understanding relationships between multiple variables simultaneously. There are several ways to build such a thing, including the `pairs` function...

```
pairs (~ sbp + age + weight + height, data=wcgs,
       main="Simple Scatterplot Matrix")
```

Simple Scatterplot Matrix
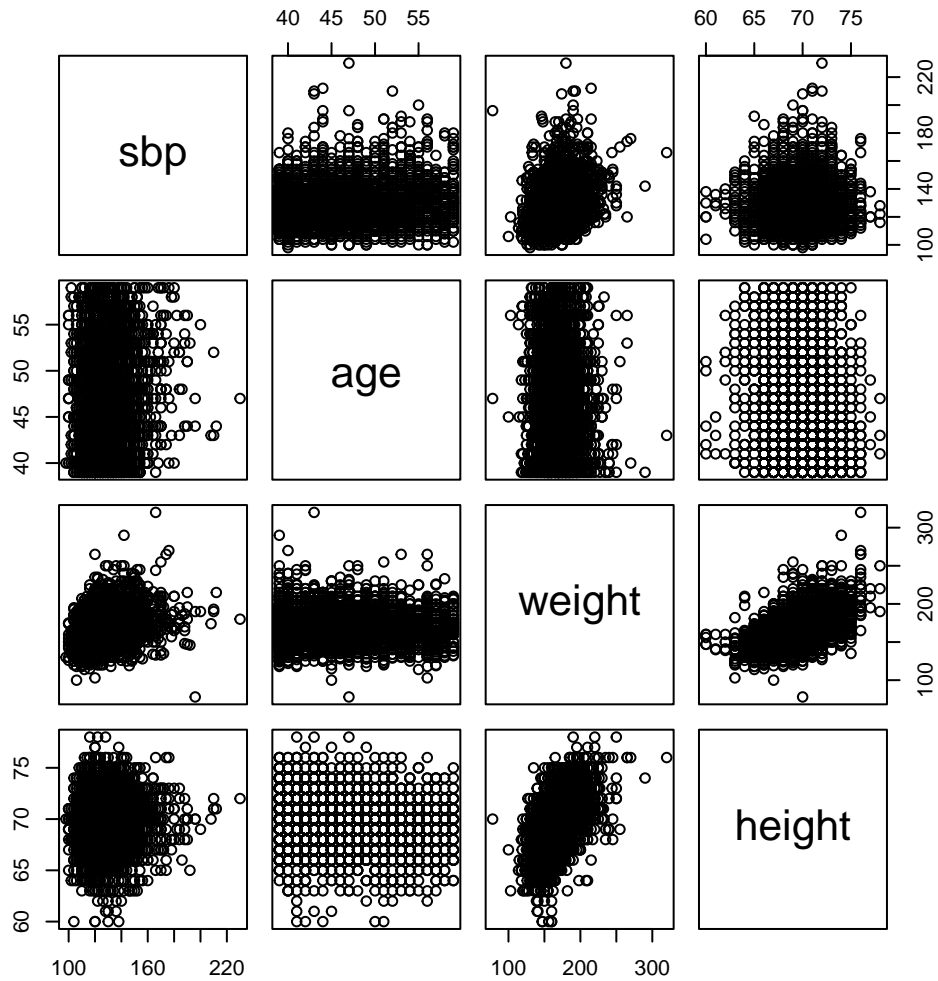
17.14.1 Displaying a Correlation Matrix

```
wcgs |>
    select(sbp, age, weight, height) |>
    cor() |>
    kbl(digits = 3) |>
    kable_styling(full_width = FALSE)
```

	sbp	age	weight	height
sbp	1.000	0.166	0.253	0.018
age	0.166	1.000	-0.034	-0.095
weight	0.253	-0.034	1.000	0.533
height	0.018	-0.095	0.533	1.000

17.14.2 Using the `GGally` package

The `ggplot2` system doesn't have a built-in scatterplot system. There are some nice add-ins in the world, though. One option I sort of like is in the `GGally` package, which can produce both correlation matrices and scatterplot matrices.

The `ggpairs` function provides a density plot on each diagonal, Pearson correlations on the upper right and scatterplots on the lower left of the matrix.

```
ggpairs(wcgs |>
          select(sbp, age, weight, height),
        title = "Scatterplot Matrix via ggpairs")
```


Scatterplot Matrix via ggpairs
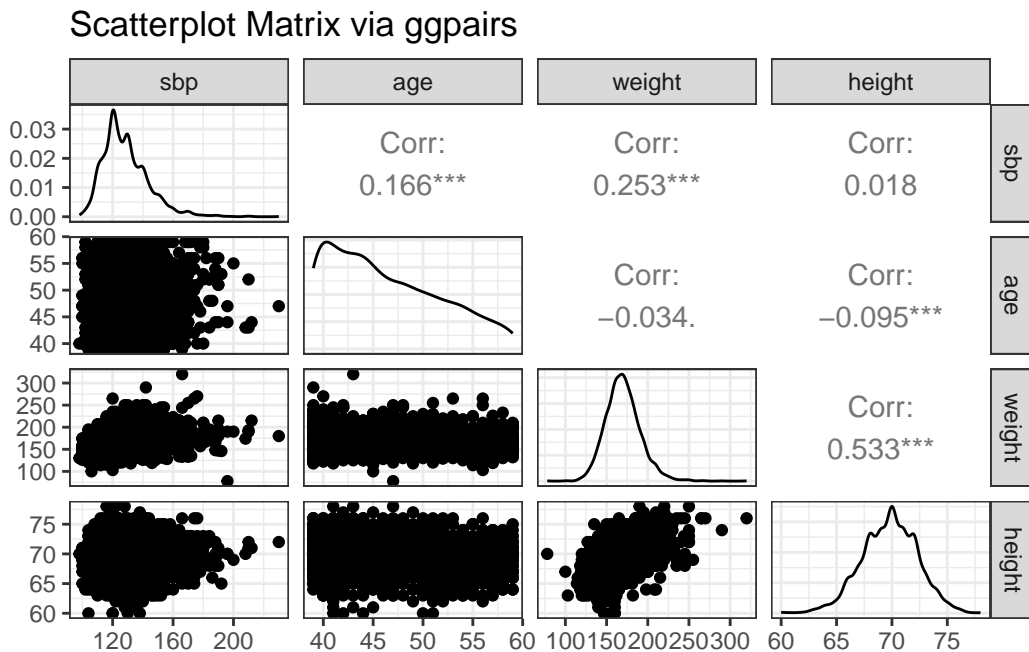
Part II

Part B. Comparing Summaries

18 Hypothesis Testing: What is it good for?

18.1 Introduction

Hypothesis testing uses sample data to attempt to reject the hypothesis that nothing interesting is happening – that is, to reject the notion that chance alone can explain the sample results[1]. We can, in many settings, use confidence intervals to summarize the results, as well, and confidence intervals and hypothesis tests are closely connected.

In particular, it's worth stressing that:

- **A significant effect is not necessarily the same thing as an interesting effect.** For example, results calculated from large samples are nearly always "significant" even when the effects are quite small in magnitude. Before doing a test, always ask if the effect is large enough to be of any practical interest. If not, why do the test?

- **A non-significant effect is not necessarily the same thing as no difference.** A large effect of real practical interest may still produce a non-significant result simply because the sample is too small.

- **There are assumptions behind all statistical inferences.** Checking assumptions is crucial to validating the inference made by any test or confidence interval.

18.2 Five Steps in any Hypothesis Test

1. Specify the null hypothesis, H_0 (which usually indicates that there is no difference or no association between the results in various groups of subjects)
2. Specify the research or alternative hypothesis, H_A, sometimes called H_1 (which usually indicates that there is some difference or some association between the results in those same groups of subjects).
3. Specify the test procedure or test statistic to be used to make inferences to the population based on sample data. Here is where we usually specify α, the probability of incorrectly rejecting H_0 that we are willing to accept. In the absence of other information, we often use $\alpha = 0.05$

---

[1]Some of this is adapted from @GoodHardin, and @Utts1999

4. Obtain the data, and summarize it to obtain the relevant test statistic, which gets summarized as a p value.
5. Use the p value to either

- **reject** H_0 in favor of the alternative H_A (concluding that there is a statistically significant difference/association at the α significance level) or
- **retain** H_0 (and conclude that there is no statistically significant difference/association at the α significance level)

18.3 Type I and Type II Error

Once we know how unlikely the results would have been if the null hypothesis were true, we must make one of two choices:

1. The p value is not small enough to convincingly rule out chance. Therefore, we cannot reject the null hypothesis as an explanation for the results.
2. The p value was small enough to convincingly rule out chance. We reject the null hypothesis and accept the alternative hypothesis.

How small must the p value be in order to rule out the null hypothesis? The standard choice is 5%. This standardization has some substantial disadvantages[2]. It is simply a convention that has become accepted over the years, and there are many situations for which a 5% cutoff is unwise. While it does give a specific level to keep in mind, it suggests a rather mindless cutpoint having nothing to do with the importance of the decision nor the costs or losses associated with outcomes.

18.4 The Courtroom Analogy

Consider the analogy of the jury in a courtroom.

1. The evidence is not strong enough to convincingly rule out that the defendant is innocent. Therefore, we cannot reject the null hypothesis, or innocence of the defendant.
2. The evidence was strong enough that we are willing to rule out the possibility that an innocent person (as stated in the null hypothesis) produced the observed data. We reject the null hypothesis, that the defendant is innocent, and assert the alternative hypothesis.

Consistent with our thinking in hypothesis testing, in many cases we would not accept the hypothesis that the defendant is innocent. We would simply conclude that the evidence was not strong enough to rule out the possibility of innocence.

---

[2]Ingelfinger JA, Mosteller F, Thibodeau LA and Ware JH (1987) Biostatistics in Clinical Medicine, 2nd Edition, New York: MacMillan. pp. 156-157.

The p value is the probability of getting a result as extreme or more extreme than the one observed if the proposed null hypothesis is true. Notice that it is not valid to actually accept that the null hypothesis is true. To do so would be to say that we are essentially convinced that chance alone produced the observed results – a common mistake.

18.5 Significance vs. Importance

Remember that a statistically significant relationship or difference does not necessarily mean an important one. A result that is significant in the statistical meaning of the word may not be significant clinically. Statistical significance is a technical term. Findings can be both statistically significant and practically significant or either or neither.

When we have large samples, we will regularly find small differences that have a small p value even though they have no practical importance. At the other extreme, with small samples, even large differences will often not be large enough to create a small p value. The notion of statistical significance has not helped science, and we won't perpetuate it any further.

18.6 What does Dr. Love dislike about p values and "statistical significance"?

A lot of things. A major issue is that I believe that p values are impossible to explain in a way that is both [a] technically correct and [b] straightforward at the same time. As evidence of this, you might want to look at this article and associated video by Christie Aschwanden at 538.com

The notion of a p value was an incredibly impressive achievement back when Wald and others were doing the work they were doing in the 1940s, and might still have been useful as recently as 10 years ago. But the notion of a p value relies on a lot of flawed assumptions, and null hypothesis significance testing is fraught with difficulties. Nonetheless, researchers use p values every day.

18.7 The ASA Articles in 2016 and 2019 on Statistical Significance and P-Values

However, my primary motivation for taking the approach I'm taking comes from the pieces in two key reference collections we'll read and discuss more thoroughly in 431 and 432.

1. The American Statistical Association's 2016 Statement on p-Values: Context, Process and Purpose.

The ASA Statement on p-Values and Statistical Significance (Wasserstein and Lazar 2016) was developed primarily because after decades, warnings about the don'ts had gone mostly unheeded. The statement was about what not to do, because there is widespread agreement about the don'ts.

2. Statistical Inference in the 21st Century: A World Beyond $p < 0.05$ from 2019 in *The American Statistician*

This is a world where researchers are free to treat "p = 0.051" and "p = 0.049" as not being categorically different, where authors no longer find themselves constrained to selectively publish their results based on a single magic number. In this world, where studies with "p < 0.05" and studies with "p > 0.05" are not automatically in conflict, researchers will see their results more easily replicated- and, even when not, they will better understand why. As we venture down this path, we will begin to see fewer false alarms, fewer overlooked discoveries, and the development of more customized statistical strategies. Researchers will be free to communicate all their findings in all their glorious uncertainty, knowing their work is to be judged by the quality and effective communication of their science, and not by their p-values. As "statistical significance" is used less, statistical thinking will be used more. The ASA Statement on P-Values and Statistical Significance started moving us toward this world…. Now we must go further.

The ASA Statement on P-Values and Statistical Significance stopped just short of recommending that declarations of "statistical significance" be abandoned. We take that step here. We conclude, based on our review of the articles in this special issue and the broader literature, that it is time to stop using the term "statistically significant" entirely. Nor should variants such as "significantly different," "p < 0.05," and "nonsignificant" survive, whether expressed in words, by asterisks in a table, or in some other way…. Regardless of whether it was ever useful, a declaration of "statistical significance" has today become meaningless.

For the moment, I will say this. I emphasize confidence intervals over p values, which is at best a partial solution. But …

1. Very rarely does a situation emerge in which a p value can be available in which looking at the associated confidence interval isn't far more helpful for making a comparison of interest.
2. The use of a p value requires making at least as many assumptions about the population, sample, individuals and data as does a confidence interval.
3. Most null hypotheses are clearly not exactly true prior to data collection, and so the test summarized by a p value is of questionable value most of the time.
4. No one has a truly adequate definition of a p value, in terms of both precision and parsimony. Brief, understandable definitions always fail to be technically accurate.
5. Bayesian approaches avoid some of these pitfalls, but come with their own issues.

6. Many smart people agree with me, and have sworn off of p values whenever they can.

Again, we'll look at these issues in greater depth later in the course.

18.8 Errors in Hypothesis Testing

In testing hypotheses, there are two potential decisions and each one brings with it the possibility that a mistake has been made.

Let's use the courtroom analogy. Here are the potential choices and associated potential errors. Although the seriousness of errors depends on the seriousness of the crime and punishment, the potential error for choice 2 is usually more serious.

1. We cannot rule out that the defendant is innocent, so (s)he is set free without penalty.

   - Potential Error: A criminal has been erroneously freed.

2. We believe that there is enough evidence to conclude that the defendant is guilty.

   - Potential Error: An innocent person is convicted / penalized and a guilty person remains free.

As another example, consider being tested for disease. Most tests for diseases are not 100% accurate. The lab technician or physician must make a choice:

1. In the opinion of the medical practitioner, you are healthy. The test result was weak enough to be called "negative" for the disease.

   - Potential Error: You actually have the disease but have been told that you do not. This is called a **false negative**.

2. In the opinion of the medical practitioner, you have the disease. The test results were strong enough to be called "positive" for the disease.

   - Potential Error: You are actually healthy but have been told you have the disease. This is called a **false positive**.

18.9 The Two Types of Hypothesis Testing Errors

–	H_A is true	H_0 is true
Test Rejects H_0	Correct Decision	Type I Error (False Positive)
Test Retains H_0	Type II Error (False Negative)	Correct Decision

- A Type I error can only be made if the null hypothesis is actually true.
- A Type II error can only be made if the alternative hypothesis is actually true.

18.10 The Significance Level is the Probability of a Type I Error

If the null hypothesis is true, the p value is the probability of making an error by choosing the alternative hypothesis instead. Alpha (α) is defined as the probability of rejecting the null hypothesis when the null hypothesis is actually true, creating a Type I error. This is also called the significance level, so that $100(1-\alpha)$ is the confidence level – the probability of correctly concluding that there is no difference (retaining H_0) when the null hypothesis is true.

18.11 The Probability of avoiding a Type II Error is called Power

A Type II error is made if the alternative hypothesis is true, but you fail to choose it. The probability depends on exactly which part of the alternative hypothesis is true, so that computing the probability of making a Type II error is not feasible. The power of a test is the probability of making the correct decision when the alternative hypothesis is true. Beta (β) is defined as the probability of concluding that there was no difference, when in fact there was one (a Type II error). Power is then just $1 - \beta$, the probability of concluding that there was a difference, when, in fact, there was one.

Traditionally, people like the power of a test to be at least 80%, meaning that β is at most 0.20. Often, I'll be arguing for 90% as a minimum power requirement, or we'll be presenting a range of power calculations for a variety of sample size choices.

18.12 Incorporating the Costs of Various Types of Errors

Which error is more serious in medical testing, where we think of our H_0: patient is healthy vs. H_A: disease is present?

It depends on the disease and on the consequences of a negative or positive test result. A false negative in a screening test for cancer could lead to a fatal delay in treatment, whereas a false positive would probably lead to a retest. A more troublesome example occurs in testing for an infectious disease. Inevitably, there is a trade-off between the two types of errors. It all depends on the consequences.

It would be nice if we could specify the probability that we were making an error with each potential decision. We could then weigh the consequence of the error against its probability. Unfortunately, in most cases, we can only specify the conditional probability of making a Type I error, given that the null hypothesis is true.

In deciding whether to reject a null hypothesis, we will need to consider the consequences of the two potential types of errors. If a Type I error is very serious, then you should reject the null hypothesis only if the p value is very small. Conversely, if a Type II error is more serious,

you should be willing to reject the null hypothesis with a larger p value, perhaps 0.10 or 0.20, instead of 0.05.

- α is the probability of rejecting H_0 when H_0 is true.

    - So 1 - α, the confidence level, is the probability of retaining H_0 when that's the right thing to do.

- β is the probability of retaining H_0 when H_A is true.

    - So 1 - β, the power, is the probability of rejecting H_0 when that's the right thing to do.

–	H_A is True	H_0 is True
Test Rejects H_0	Correct Decision (1 - β)	Type I Error (α)
Test Retains H_0	Type II Error (β)	Correct Decision (1 - α)

18.13 Power and Sample Size Considerations

For most statistical tests, it is theoretically possible to estimate the power of the test in the design stage, (before any data are collected) for various sample sizes, so we can hone in on a sample size choice which will enable us to collect data only on as many subjects as are truly necessary.

A power calculation is likely the most common element of an scientific grant proposal on which a statistician is consulted. This is a fine idea in theory, but in practice...

- The tests that have power calculations worked out in intensive detail using R are mostly those with more substantial assumptions. Examples include t tests that assume population normality, common population variance and balanced designs in the independent samples setting, or paired t tests that assume population normality in the paired samples setting.
- These power calculations are also usually based on tests rather than confidence intervals, which would be much more useful in most settings. Simulation is your friend here.
- Even more unfortunately, this process of doing power and related calculations is **far more of an art than a science**.

- As a result, the value of many power calculations is negligible, since the assumptions being made are so arbitrary and poorly connected to real data.
- On several occasions, I have stood in front of a large audience of medical statisticians actively engaged in clinical trials and other studies that require power calculations for funding. When I ask for a show of hands of people who have had power calculations prior to such a study whose assumptions matched the eventual data perfectly, I get lots of laughs. It doesn't happen.
- Even the underlying framework that assumes a power of 80% with a significance level of 5% is sufficient for most studies is pretty silly.

All that said, I feel obliged to show you some examples of power calculations done using R, and provide some insight on how to make some of the key assumptions in a way that won't alert reviewers too much to the silliness of the enterprise, and so I will do so in Chapter 22.

19 Confidence Intervals for a Mean

19.1 Setup: Packages Used Here

In this part of the course, we'll make use of a few scripts I've gathered for you, in the Love-boost R script.

```
knitr::opts_chunk$set(comment = NA)

source("data/Love-boost.R")
library(boot)
library(broom)
library(Hmisc)
library(janitor)
library(kableExtra)
library(patchwork)
library(psych)
library(tidyverse)

theme_set(theme_bw())
```

We will also use the `favstats` function from the `mosaic` package.

19.2 Introduction

The basic theory of estimation can be used to indicate the probable accuracy and potential for bias in estimating based on limited samples. A point estimate provides a single best guess as to the value of a population or process parameter.

A confidence interval is a particularly useful way to convey to people just how much error one must allow for in a given estimate. In particular, a confidence interval allows us to quantify just how close we expect, for instance, the sample mean to be to the population or process mean. The computer will do the calculations; we need to interpret the results.

The key things that we will need to trade off are cost vs. precision, and precision vs. confidence in the correctness of the statement. Often, if we are dissatisfied with the width of the confidence

interval and want to make it smaller, we have little choice but to reconsider the sample – larger samples produce shorter intervals.

19.3 This Chapter's Goals

Suppose that we are interested in learning something about a population or process, from which we can obtain a sample that consists of a subset of potential results from that population or process. The main goal for many of the parametric models that are a large part of statistics is to estimate population parameters, like a population mean, or regression coefficient, on the basis of a sample. When we do this, we want to describe not only our best guess at the parameter (referred to as a *point estimate*) but also say something useful about the uncertainty in our estimate, to let us more completely assess what the data have to tell us. A key tool for doing this is a **confidence interval**.

Essentially every textbook on introductory statistics describes the development of a confidence interval, at least for a mean. Good supplemental resources are highlighted in the references I've provided in the course syllabus.

We'll develop confidence intervals to compare parameters about two populations (either through matched pairs or independent samples) with confidence intervals soon. Here, we'll consider the problem of estimating a confidence interval to describe the mean (or median) of the population represented by a single sample of quantitative data.

19.4 Serum Zinc Levels in 462 Teenage Males (`serzinc`)

The `serzinc` data include serum zinc levels in micrograms per deciliter that have been gathered for a sample of 462 males aged 15-17, My source for these data is Appendix B1 of Pagano and Gauvreau (2000). Serum zinc deficiency has been associated with anemia, loss of strength and endurance, and it is thought that 25% of the world's population is at risk of zinc deficiency. Such a deficiency can indicate poor nutrition, and can affect growth and vision, for instance. "Typical" values[1] are said to be 0.66-1.10 mcg/ml, which is 66 - 110 micrograms per deciliter.

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

names(serzinc)
```

```
[1] "ID"    "zinc"
```

---

[1]Reference values for those over the age of 10 years at http://www.mayomedicallaboratories.com/test-catalog/Clinical+and+Interpretive/8620 , visited 2019-09-17.

min	Q1	median	Q3	max	mean	sd	n	missing
50	76	86	98	153	87.94	16	462	0

```
mosaic::favstats(~ zinc, data = serzinc) |>
  kbl(digits = 2) |>
  kable_styling()
```

19.5 Our Goal: A Confidence Interval for the Population Mean

After we assess the data a bit, and are satisfied that we understand it, our first inferential goal will be to produce a **confidence interval for the true (population) mean** of males age 15-17 based on this sample, assuming that these 462 males are a random sample from the population of interest, that each serum zinc level is drawn independently from an identical distribution describing that population.

To do this, we will have several different procedures available, including:

1. A confidence interval for the population mean based on a t distribution, when we assume that the data are drawn from an approximately Normal distribution, using the sample standard deviation. (Interval corresponding to a t test, and it will be a good choice when the data really are approximately Normally distributed.)
2. A resampling approach to generate a bootstrap confidence interval for the population mean, which does not require that we assume either that the population standard deviation is known, nor that the data are drawn from an approximately Normal distribution, but which has some other weaknesses.
3. 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).

19.6 Exploratory Data Analysis for Serum Zinc

19.6.1 Graphical Summaries

The code presented below builds:

- a histogram (with Normal model superimposed),
- a boxplot (with median notch) and
- a Normal Q-Q plot (with guiding straight line through the quartiles)

for the `zinc` results from the `serzinc` tibble.

```
p1 <- ggplot(serzinc, aes(sample = zinc)) +
  geom_qq(col = "dodgerblue") + geom_qq_line(col = "navy") +
  theme(aspect.ratio = 1) +
  labs(title = "Normal Q-Q plot")

p2 <- ggplot(serzinc, aes(x = zinc)) +
  geom_histogram(aes(y = stat(density)),
                 bins = 10, fill = "dodgerblue", col = "white") +
  stat_function(fun = dnorm,
                args = list(mean = mean(serzinc$zinc),
                            sd = sd(serzinc$zinc)),
                col = "navy", lwd = 1.5) +
  labs(title = "Histogram and Normal Density")

p3 <- ggplot(serzinc, aes(x = zinc, y = "")) +
  geom_boxplot(fill = "dodgerblue", outlier.color = "dodgerblue") +
  stat_summary(fun = "mean", geom = "point",
               shape = 23, size = 3, fill = "white") +
  labs(title = "Boxplot", y = "")

p1 + (p2 / p3 + plot_layout(heights = c(4,1))) +
  plot_annotation(title = "Serum Zinc (micrograms per deciliter) for 462 Teenage Males")
```

Warning: `stat(density)` was deprecated in ggplot2 3.4.0.
i Please use `after_stat(density)` instead.

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
X1	1	462	87.94	16	86	87.17	16.31	50	153	103	0.62	0.87	0.74

Serum Zinc (micrograms per deciliter) for 462 Teenage Males
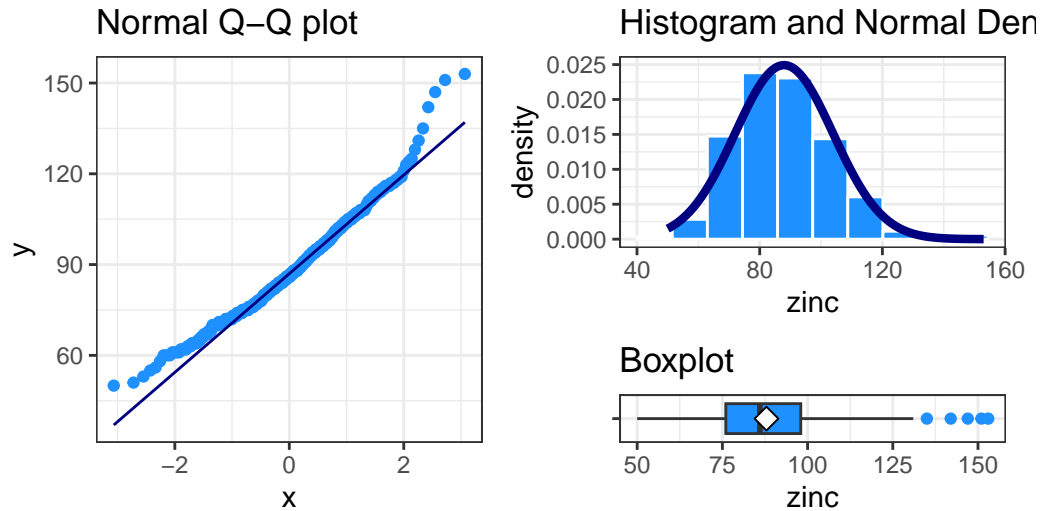
These results include some of the more useful plots and numerical summaries when assessing shape, center and spread. The `zinc` data in the `serzinc` data frame appear to be slightly right skewed, with five outlier values on the high end of the scale, in particular.

19.6.2 Numerical Summaries

This section describes some numerical summaries of interest to augment the plots in summarizing the center, spread and shape of the distribution of serum zinc among these 462 teenage males. Note that I use `summarise` here to be sure that the `dplyr` package's `summarise/summarize` function is used, rather than the one contained in `Hmisc`. I could also have used `dplyr::summarize()` to accomplish the same goal.

```
describe(serzinc$zinc) |>  # from psych package
  kbl(digits = 2) |> kable_styling()


serzinc |>
    summarise(mean(zinc), median(zinc), sd(zinc),
```

mean(zinc)	median(zinc)	sd(zinc)	skew1
87.94	86	16	0.12

```
        skew1 = (mean(zinc) - median(zinc))/sd(zinc)) |>
  kbl(digits = 2) |> kable_styling()
```

The skew1 value here (mean - median divided by the standard deviation) backs up our graphical assessment, that the data are slightly right skewed.

Rounded to two decimal places, the standard deviation of the serum zinc data turns out to be 16, and so the standard error of the mean, shown as `se` in the `psych::describe` output, is 16 divided by the square root of the sample size, $n = 462$. This standard error is about to become quite important to us in building statistical inferences about the mean of the entire population of teenage males based on this sample.

19.7 Defining a Confidence Interval

A confidence interval for a population or process mean uses data from a sample (and perhaps some additional information) to identify a range of potential values for the population mean, which, if certain assumptions hold, can be assumed to provide a reasonable estimate for the true population mean. A confidence interval consists of:

1. An interval estimate describing the population parameter of interest (here the population mean), and
2. A probability statement, expressed in terms of a confidence level.

19.8 Estimating the Population Mean from the Serum Zinc data

As an example, suppose that we are willing to assume that the mean serum zinc level across the entire population of teenage males, μ, follows a Normal distribution (and so, summarizing it with a mean is a rational thing to do.) Suppose that we are also willing to assume that the 462 teenage males contained in the `serzinc` tibble are a random sample from that complete population. While we know the mean of the sample of 462 boys, we don't know μ, the mean across all teenage males. So we need to estimate it.

Earlier we estimated that a 90% confidence interval for the mean serum zinc level (μ) across the entire population of teenage males was (86.71, 89.16) micrograms per deciliter. How should we interpret this result?

- Some people think this means that there is a 90% chance that the true mean of the population, μ, falls between 86.71 and 89.16 micrograms per deciliter. That's not correct.

- The population mean is a constant **parameter** of the population of interest. That constant is not a random variable, and does not change. So the actual probability of the population mean falling inside that range is either 0 or 1.
- Our confidence is in our process.

    - It's in the sampling method (random sampling) used to generate the data, and in the assumption that the population follows a Normal distribution.
    - It's captured in our accounting for one particular type of error (called *sampling error*) in developing our interval estimate, while assuming all other potential sources of error are negligible.

So, what's closer to the truth is:

- If we used this same method to sample data from the true population of teenage males, and built 100 such 90% confidence intervals, then about 90 of them would contain the true population mean.

19.9 Confidence vs. Significance Level

We've estimated a 90% confidence interval for the population mean serum zinc level among teenage boys using the `serzinc` data.

- We call $100(1-\alpha)\%$, here, 90%, or 0.90, the *confidence* level, and
- $\alpha = 10\%$, or 0.10 is called the *significance* level.

If we had instead built a series of 100 different 95% confidence intervals, then about 95 of them would contain the true value of μ.

Let's look more closely at the issue of estimating a population **mean** based on a sample of observations. We will need three critical pieces - the sample, the confidence level, and the margin of error, which is based on the standard error of a sample mean, when we are estimating a population mean.

19.10 The Standard Error of a Sample Mean

The standard error, generally, is the name we give to the standard deviation associated with any particular parameter estimate.

- If we are using a sample mean based on a sample of size n to estimate a population mean, the **standard error of that sample mean** is the standard deviation of the measurements in the population, divided by the square root of the sample size.

- We often estimate this particular standard error with s (the sample standard deviation) divided by the square root of the sample size.
- Other statistics have different standard errors.

  - $\sqrt{p(1-p)/n}$ is the standard error of the sample proportion p estimated using a sample of size n.
  - $\sqrt{\frac{1-r^2}{n-2}}$ is the standard error of the sample Pearson correlation r estimated using n pairs of observations.
  - $\sqrt{\frac{SD_1^2}{n_1} + \frac{SD_2^2}{n_2}}$ is the standard error of the difference between two means $\bar{x}_1$ and $\bar{x}_2$, estimated using samples of sizes n_1 and n_2 with sample standard devistions SD_1 and SD_2, respectively.

In developing a confidence interval for a population mean, we may be willing to assume that the data in our sample are drawn from a Normally distributed population. If so, the most common and useful means of building a confidence interval makes use of the t distribution (sometimes called Student's t) and the notion of a *standard error*.

19.11 The t distribution and CIs for a Mean

In practical settings, we will use the t distribution to estimate a confidence interval from a population mean whenever we:

- are willing to assume that the sample is drawn at random from a population or process with a Normal distribution,
- are using our sample to estimate both the mean and standard deviation, and
- have a small sample size.

19.11.1 The Formula

The two-sided $100(1-\alpha)\%$ confidence interval (based on a t test) is:

$$\bar{x} \pm t_{\alpha/2, n-1}(s/\sqrt{n})$$

where $t_{\alpha/2, n-1}$ is the value that cuts off the top $\alpha/2$ percent of the t distribution, with $n-1$ degrees of freedom.

We obtain the relevant cutoff value in R by substituting in values for `alphaover2` and `n-1` into the following line of R code:

```
qt(alphaover2, df = n-1, lower.tail=FALSE)
```

19.11.2 Student's t distribution

Student's t distribution looks a lot like a Normal distribution, when the sample size is large. Unlike the normal distribution, which is specified by two parameters, the mean and the standard deviation, the t distribution is specified by one parameter, the degrees of freedom.

- t distributions with large numbers of degrees of freedom are more or less indistinguishable from the standard Normal distribution.
- t distributions with smaller degrees of freedom (say, with df < 30, in particular) are still symmetric, but are more outlier-prone than a Normal distribution

```
p1 <- ggplot(data.frame(x = c(-3, 3)), aes(x)) +
  stat_function(fun = dt, args = list(df = 1)) +
  stat_function(fun = dnorm, col = "red") +
  labs(title = "t with 1 df", y = "Density", x = "")

p2 <- ggplot(data.frame(x = c(-3, 3)), aes(x)) +
  stat_function(fun = dt, args = list(df = 3)) +
  stat_function(fun = dnorm, col = "red") +
  labs(title = "t with 3 df", y = "Density", x = "")

p3 <- ggplot(data.frame(x = c(-3, 3)), aes(x)) +
  stat_function(fun = dt, args = list(df = 5)) +
  stat_function(fun = dnorm, col = "red") +
  labs(title = "t with 5 df", y = "Density", x = "")

p4 <- ggplot(data.frame(x = c(-3, 3)), aes(x)) +
  stat_function(fun = dt, args = list(df = 10)) +
  stat_function(fun = dnorm, col = "red") +
  labs(title = "t with 10 df", y = "Density", x = "")

p5 <- ggplot(data.frame(x = c(-3, 3)), aes(x)) +
  stat_function(fun = dt, args = list(df = 20)) +
  stat_function(fun = dnorm, col = "red") +
  labs(title = "t with 20 df", y = "Density", x = "")

p6 <- ggplot(data.frame(x = c(-3, 3)), aes(x)) +
  stat_function(fun = dt, args = list(df = 30)) +
  stat_function(fun = dnorm, col = "red") +
  labs(title = "t with 30 df", y = "Density", x = "")

(p1 + p2 + p3) / (p4 + p5 + p6) +
```

```
plot_annotation(
    title = "Various t distributions and the Standard Normal",
    caption = "In each plot, the Standard Normal distribution is in red")
```

Various t distributions and the Standard Normal
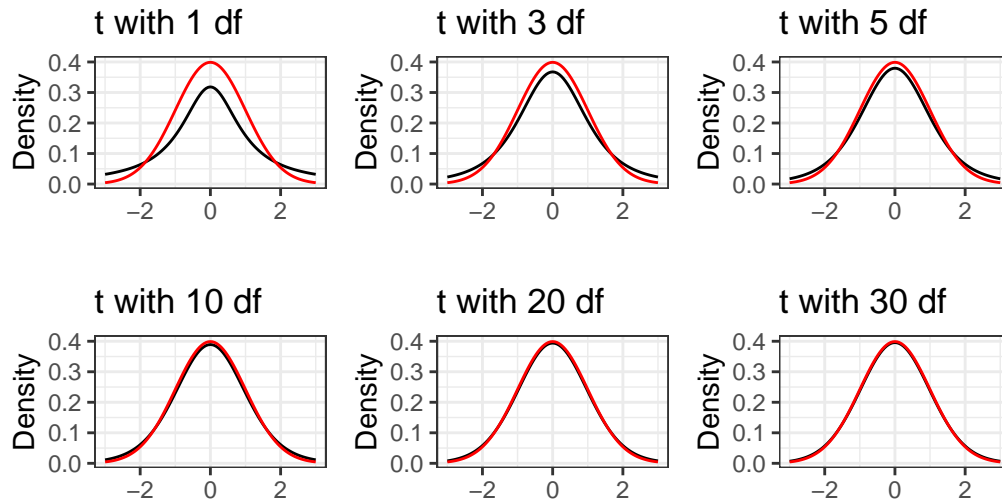
In each plot, the Standard Normal distribution is in red

19.12 Building the CI in R

Suppose we wish to build a 90% confidence interval for the true mean serum zinc level across the entire population of teenage males. The confidence level will be 90%, or 0.90, and so the α value, which is 1 - confidence = 0.10.

So what we know going in is that:

- We want $\alpha = 0.10$, because we're creating a 90% confidence interval.
- The sample size $n = 462$ serum zinc measurements.
- The sample mean of those measurements, $\bar{x} = 87.937$ micrograms per deciliter.
- The sample standard deviation of those measurements, $s = 16.005$ micrograms per deciliter.

390

19.13 Using an intercept-only regression model

in the context of fitting an intercept-only linear regression model. An intercept-only model is fitted by putting the number 1 on the right hand side of our linear model. The resulting model simply fits the overall mean of the data as a prediction for all subjects.

```
model_zinc <- lm(zinc ~ 1, data = serzinc)
```

```
summary(model_zinc)
```

```
Call:
lm(formula = zinc ~ 1, data = serzinc)

Residuals:
    Min      1Q  Median      3Q     Max
-37.937 -11.937  -1.937  10.063  65.063

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  87.9372     0.7446   118.1   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 16 on 461 degrees of freedom
```

```
confint(model_zinc, level = 0.90)
```

```
              5 %     95 %
(Intercept) 86.71 89.16446
```

Generally, though, I'll use the `tidy()` function in the `broom` package to obtain the key information from a model like this:

```
tidy(model_zinc, conf.int = TRUE, conf = 0.90) |>
    kbl(digits = 2) |> kable_styling()
```

As an alternative, we could also use the `t.test` function, which can build (in this case) a two-sided confidence interval for the zinc levels like this:

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	87.94	0.74	118.1	0	86.71	89.16

estimate	statistic	p.value	parameter	conf.low	conf.high	method	alternative
87.94	118.1	0	461	86.71	89.16	One Sample t-test	two.sided

```
tt <- t.test(serzinc$zinc,
             conf.level = 0.90,
             alternative = "two.sided")

tt
```

```
	One Sample t-test

data:  serzinc$zinc
t = 118.1, df = 461, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
90 percent confidence interval:
 86.71000 89.16446
sample estimates:
mean of x
 87.93723
```

and the `tidy()` function from the `broom` package works here, too.

```
tidy(tt, conf.int = TRUE, conf = 0.90) |>
  kbl(digits = 2) |> kable_styling()
```

And again, our 90% confidence interval for the true population mean serum zinc level, based on our sample of 462 patients, is (86.71, 89.16) micrograms per deciliter[2].

19.14 Interpreting the Result

An appropriate interpretation of the 90% two-sided confidence interval above follows:

- (86.71, 89.16) micrograms per deciliter is a 90% two-sided confidence interval for the population mean serum zinc level among teenage males.

---

[2]Since the measured zinc levels appear as integers, we should probably be rounding even further in our confidence interval, down to perhaps one decimal place.

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	87.94	0.74	118.1	0	86.47	89.4

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	87.94	0.74	118.1	0	86.01	89.86

- Our point estimate for the true population mean serum zinc level is 87.94. The values in the interval (86.71, 89.16) represent a reasonable range of estimates for the true population mean serum zinc level, and we are 90% confident that this method of creating a confidence interval will produce a result containing the true population mean serum zinc level.
- Were we to draw 100 samples of size 462 from the population described by this sample, and use each such sample to produce a confidence interval in this manner, approximately 90 of those confidence intervals would cover the true population mean serum zinc level.

19.15 What if we want a 95% or 99% confidence interval instead?

We can obtain them using `tidy` and the same modeling approach.

```r
tidy(model_zinc, conf.int = TRUE, conf.level = 0.95) |>
  kbl(digits = 2) |> kable_styling()
```

```r
tidy(model_zinc, conf.int = TRUE, conf.level = 0.99) |>
  kbl(digits = 2) |> kable_styling()
```

19.16 Using the `broom` package with the t test

The `broom` package takes the messy output of built-in functions in R, such as `lm`, `t.test` or `wilcox.test`, and turns them into tidy data frames. A detailed description of the package and three of its key functions is found at https://github.com/tidyverse/broom.

For example, we can use the `tidy` function within `broom` to create a single-row tibble of the key results from a t test.

```r
tt <- t.test(serzinc$zinc, conf.level = 0.95, alternative = "two.sided")
tidy(tt) |> kbl(digits = 2) |> kable_styling()
```

estimate	statistic	p.value	parameter	conf.low	conf.high	method	alternative
87.94	118.1	0	461	86.47	89.4	One Sample t-test	two.sided

We can thus pull the endpoints of a 95% confidence interval directly from this output. `broom` also has a `glance` function, which returns the same information as `tidy` in the case of a t-test.

19.16.1 Effect of Changing the Confidence Level

Below, we see two-sided confidence intervals for various levels of α.

Confidence Level	α	Two-Sided Interval Estimate for Zinc Level Population Mean, μ	Point Estimate of μ
80% or 0.80	0.20	(87, 88.9)	87.9
90% or 0.90	0.10	(86.7, 89.2)	87.9
95% or 0.95	0.05	(86.5, 89.4)	87.9
99% or 0.99	0.01	(86, 89.9)	87.9

What happens to the width of the confidence interval in this table as the confidence level changes?

19.17 One-sided vs. Two-sided Confidence Intervals

Occasionally, we want to estimate either an upper limit for the population mean μ, or a lower limit for μ, but not both.

```
t.test(serzinc$zinc, conf.level = 0.90, alternative = "greater")
```

```
    One Sample t-test

data:  serzinc$zinc
t = 118.1, df = 461, p-value < 2.2e-16
alternative hypothesis: true mean is greater than 0
90 percent confidence interval:
 86.98161      Inf
sample estimates:
```

```
mean of x
 87.93723
```

```
t.test(serzinc$zinc, conf.level = 0.90, alternative = "less")
```

```
    One Sample t-test

data:  serzinc$zinc
t = 118.1, df = 461, p-value = 1
alternative hypothesis: true mean is less than 0
90 percent confidence interval:
     -Inf 88.89285
sample estimates:
mean of x
 87.93723
```

Note the relationship between the *two-sided* 80% confidence interval, and the *one-sided* 90% confidence intervals.

Confidence	α	Type of Interval	Interval Estimate for Zinc Level Population Mean, μ
80% (.80)	0.20	Two-Sided	(86.98, 88.89)
90% (.90)	0.10	One-Sided (Less Than)	$\mu < 88.89.$
90% (.90)	0.10	One-Sided (Greater Than)	$\mu > 86.98.$

Why does this happen? The 80% two-sided interval is placed so as to cut off the top 10% of the distribution with its upper bound, and the bottom 10% of the distribution with its lower bound. The 90% "less than" one-sided interval is placed so as to have its lower bound cut off the top 10% of the distribution.

The same issue appears when we consider two-sided 90% and one-sided 95% confidence intervals.

Confidence	α	Type of Interval	Interval Estimate for Zinc Level Population Mean, μ
90% (.90)	0.10	Two-Sided	(86.71, 89.16)
95% (.95)	0.05	One-Sided (Less Than)	$\mu < 89.16.$
95% (.95)	0.05	One-Sided (Greater Than)	$\mu > 86.71.$

Again, the 90% two-sided interval cuts off the top 5% and bottom 5% of the distribution with its bounds. The 95% "less than" one-sided interval also has its lower bound cut off the top 5% of the distribution.

19.18 Bootstrap Confidence Intervals

The bootstrap (and in particular, what's known as bootstrap resampling) is a really good idea that you should know a little bit about.

If we want to know how accurately a sample mean estimates the population mean, we would ideally like to take a very, very large sample, because if we did so, we could conclude with something that would eventually approach mathematical certainty that the sample mean would be very close to the population mean.

But we can rarely draw enormous samples. So what can we do?

19.19 Resampling is A Big Idea

One way to find out how precise our estimates are is to run them on multiple samples of the same size. This *resampling* approach was codified originally by Brad Efron in 1979.

Oversimplifying a lot, the idea is that if we sample (with replacement) from our current sample, we can draw a new sample of the same size as our original.

- And if we repeat this many times, we can generate as many samples of, say, 462 zinc levels, as we like.
- Then we take these thousands of samples and calculate (for instance) the sample mean for each, and plot a histogram of those means.
- If we then cut off the top and bottom 5% of these sample means, we obtain a reasonable 90% confidence interval for the population mean.

19.20 When is a Bootstrap Confidence Interval Reasonable?

A bootstrapped interval estimate for the population mean, μ, will be reasonable as long as we're willing to believe that:

- the original sample was a random sample (or at least a completely representative sample) from a population,
- and that the samples are independent of each other,

even if the population of interest doesn't follow a Normal, or even a symmetric distribution.

A downside of the bootstrap is that you and I will get (somewhat) different answers if we resample from the same data without setting the same random seed.

19.21 Bootstrap confidence interval for the mean: Process

To avoid the Normality assumption, and take advantage of modern computing power, we use R to obtain a bootstrap confidence interval for the population mean based on a sample.

What the computer does:

1. Re-sample the data with replacement, until it obtains a new sample that is equal in size to the original data set.
2. Calculates the statistic of interest (here, a sample mean.)
3. Repeat the steps above many times (the default is 1,000 using our approach) to obtain a set of 1,000 sample means.
4. Sort those 1,000 sample means in order, and estimate the 95% confidence interval for the population mean based on the middle 95% of the 1,000 bootstrap samples.
5. Send us a result, containing the sample mean, and a 95% confidence interval for the population mean

19.22 Using R to estimate a bootstrap CI

The command that we use to obtain a Confidence Interval for μ using the basic nonparametric bootstrap and without assuming a Normally distributed population, is `smean.cl.boot`, a part of the `Hmisc` package in R.

```
set.seed(431)
smean.cl.boot(serzinc$zinc, B = 1000, conf.int = 0.90) ## from Hmisc
```

```
    Mean     Lower     Upper
87.93723 86.76775 89.20617
```

- Remember that the t-based 90% CI for μ was (86.71, 89.16), according to the following output…

```
tidy(lm(zinc ~ 1, data = serzinc), conf.int = TRUE, conf.level = 0.90)
```

```
# A tibble: 1 x 7
  term        estimate std.error statistic p.value conf.low conf.high
  <chr>          <dbl>     <dbl>     <dbl>   <dbl>    <dbl>     <dbl>
1 (Intercept)     87.9     0.745      118.       0     86.7      89.2
```

19.23 Comparing Bootstrap and T-Based Confidence Intervals

- The `smean.cl.boot` function (unlike most R functions) deletes missing data automatically, as does the `smean.cl.normal` function, which can also be used to produce the t-based confidence interval.

```
set.seed(431)
smean.cl.boot(serzinc$zinc, B = 1000, conf.int = 0.90)
```

```
    Mean     Lower     Upper
87.93723 86.76775 89.20617
```

```
smean.cl.normal(serzinc$zinc, conf.int = 0.90)
```

```
    Mean     Lower     Upper
87.93723 86.71000 89.16446
```

Bootstrap resampling confidence intervals do not follow the general confidence interval strategy using a point estimate plus or minus a margin for error.

- A bootstrap interval is often asymmetric, and while it will generally have the point estimate (the sample mean) near its center, for highly skewed data, this will not necessarily be the case.
- We will usually use either 1,000 (the default) or 10,000 bootstrap replications for building confidence intervals – practically, it makes little difference.

19.23.1 Bootstrap Resampling: Advantages and Caveats

The bootstrap may seem like the solution to all estimation problems. In theory, we could use the same approach to find a confidence interval for any other parameter – it's not perfect, but it is very useful. Bootstrap procedures exist for virtually any statistical comparison - the t-test analog is just one of many possibilities, and bootstrap methods are rapidly gaining on more traditional approaches in the literature thanks mostly to faster computers.

The great advantage of the bootstrap is its relative simplicity, but don't forget that many of the original assumptions of the t-based confidence interval still hold.

- Using a bootstrap does eliminate the need to worry about the Normality assumption in small sample size settings, but it still requires independent and identically distributed samples from the population of interest.

The bootstrap produces clean and robust inferences (such as confidence intervals) in many tricky situations. It is still possible that the results can be both:

- **inaccurate** (i.e. they can include the true value of the unknown population mean less often than the stated confidence probability) and
- **imprecise** (i.e., they can include more extraneous values of the unknown population mean than is desirable).

19.24 Using the Bootstrap to develop other CIs

19.24.1 Changing the Confidence Level

What if we wanted to change the confidence level?

```
set.seed(431654)
smean.cl.boot(serzinc$zinc, B = 1000, conf.int = 0.95)
```

```
    Mean    Lower    Upper
87.93723 86.51066 89.42002
```

```
set.seed(431321)
smean.cl.boot(serzinc$zinc, B = 1000, conf.int = 0.99)
```

```
    Mean    Lower    Upper
87.93723 86.20657 89.68619
```

19.25 One-Tailed Bootstrap Confidence Intervals

If you want to estimate a one tailed confidence interval for the population mean using the bootstrap, then the procedure is as follows:

1. Determine α, the significance level you want to use in your one-sided confidence interval. Remember that α is 1 minus the confidence level. Let's assume we want a 90% one-sided interval, so $\alpha = 0.10$.
2. Double α to determine the significance level we will use in the next step to fit a two-sided confidence interval.
3. Fit a two-sided confidence interval with confidence level $100(1 - 2 * \alpha)$. Let the bounds of this interval be (a, b).
4. The one-sided (greater than) confidence interval will have a as its lower bound.
5. The one-sided (less than) confidence interval will have b as its upper bound.

Suppose that we want to find a 95% one-sided upper bound for the population mean serum zinc level among teenage males, μ, using the bootstrap.

Since we want a 95% confidence interval, we have $\alpha = 0.05$. We double that to get $\alpha = 0.10$, which implies we need to instead fit a two-sided 90% confidence interval.

```
set.seed(43101)
smean.cl.boot(serzinc$zinc, B = 1000, conf.int = 0.90)  ## from Hmisc package
```

```
    Mean     Lower     Upper
87.93723 86.70509 89.11266
```

The upper bound of this two-sided 90% CI will also be the upper bound for a 95% one-sided CI.

19.25.1 Bootstrap CI for the Population Median

If we are willing to do a small amount of programming work in R, we can obtain bootstrap confidence intervals for other population parameters besides the mean. One statistic of common interest is the median. How do we find a confidence interval for the population median using a bootstrap approach? The easiest way I know of makes use of the **boot** package, as follows.

In step 1, we specify a new function to capture the medians from our sample.

```
f.median <- function(y, id)
{    median ( y[id])  }
```

In step 2, we call the **boot.ci** function from the **boot** package.

```
set.seed(431787)
boot.ci(boot(serzinc$zinc, f.median, 1000), conf=0.90, type="basic")
```

	min	Q1	median	Q3	max	mean	sd	n	missing
	50	76	86	98	153	87.94	16	462	0

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = boot(serzinc$zinc, f.median, 1000), conf = 0.9,
    type = "basic")

Intervals :
Level      Basic
90%   (84, 87 )
Calculations and Intervals on Original Scale
```

This yields a 90% confidence interval for the population median serum zinc level. Recall that the sample median for the serum zinc levels in our sample of 462 teenage males was 86 micrograms per deciliter.

```
mosaic::favstats(~ zinc, data = serzinc) |>
  kbl(digits = 2) |> kable_styling()
```

Actually, the `boot.ci` function can provide up to five different types of confidence interval (see the help file) if we change to `type="all"`, and some of those other versions have attractive properties. However, we'll stick with the basic approach in 431.

19.25.2 Bootstrap CI for the IQR

If for some reason, we want to find a 95% confidence interval for the population value of the inter-quartile range via the bootstrap, we can do it.

```
IQR(serzinc$zinc)
```

```
[1] 22
```

```
f.IQR <- function(y, id)
{    IQR (y[id]) }

set.seed(431207)
boot.ci(boot(serzinc$zinc, f.IQR, 1000),
```

```
        conf=0.95, type="basic")
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = boot(serzinc$zinc, f.IQR, 1000), conf = 0.95,
    type = "basic")

Intervals :
Level      Basic
95%   (20.00, 24.24 )
Calculations and Intervals on Original Scale
```

19.26 Wilcoxon Signed Rank Procedure for CIs

It turns out to be difficult, without the bootstrap, to estimate an appropriate confidence interval for the median of a population, which might be an appealing thing to do, particularly if the sample data are clearly not Normally distributed, so that a median seems like a better summary of the center of the data. Bootstrap procedures are available to perform the task.

The Wilcoxon signed rank approach can be used as an alternative to t-based procedures to build interval estimates for the population *pseudo-median* when the population cannot be assumed to follow a Normal distribution.

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

19.26.1 What is a Pseudo-Median?

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.

402

estimate	statistic	p.value	conf.low	conf.high	method	a
87.5	106953	0	86	88.5	Wilcoxon signed rank test with continuity correction	t

19.27 Wilcoxon Signed Rank-based CI in R

```
wilcox.test(serzinc$zinc, conf.int = TRUE, conf.level = 0.95)
```

```
        Wilcoxon signed rank test with continuity correction

data:  serzinc$zinc
V = 106953, p-value < 2.2e-16
alternative hypothesis: true location is not equal to 0
95 percent confidence interval:
 85.99997 88.50002
sample estimates:
(pseudo)median
      87.49996
```

19.27.1 Interpreting the Wilcoxon CI for the Population Median

If we're willing to believe the `zinc` levels come from a population with a symmetric distribution, the 95% Confidence Interval for the population median would be (86, 88.5)

For a non-symmetric population, this only applies to the *pseudo-median.*

Note that the pseudo-median (87.5) is actually closer here to the sample mean (87.9) than it is to the sample median (86).

19.27.2 Using the `broom` package with the Wilcoxon test

We can also use the `tidy` function within `broom` to create a single-row tibble of the key results from a Wilcoxon test, so long as we run `wilcox.test` specifying that we want a confidence interval.

```
wt <- wilcox.test(serzinc$zinc, conf.int = TRUE, conf.level = 0.95)
tidy(wt) |> kbl(digits = 2) |> kable_styling()
```

19.28 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. Finally, 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.

20 Ibuprofen in Sepsis

20.1 Setup: Packages Used Here

```
knitr::opts_chunk$set(comment = NA)

source("data/Love-boost.R")
library(broom)
library(Epi)
library(ggridges)
library(Hmisc)
library(kableExtra)
library(janitor)
library(tidyverse)

theme_set(theme_bw())
```

In addition to the `Love-boost.R` script, we will also use the `favstats` function from the `mosaic` package.

20.2 The Trial

Our next study is a randomized controlled trial comparing ibuprofen vs. placebo in patients with sepsis, which uses an *independent samples* design to compare two samples of quantitative data. We will be working with a sample from the Ibuprofen in Sepsis study, which is also studied in Dupont (2002). Quoting the abstract from Bernard et al. (1997):

> Ibuprofen has been shown to have effects on sepsis in humans, but because of their small samples (fewer than 30 patients), previous studies have been inadequate to assess effects on mortality. We sought to determine whether ibuprofen can alter rates of organ failure and mortality in patients with the sepsis syndrome, how the drug affects the increased metabolic demand in sepsis (e.g., fever, tachypnea, tachycardia, hypoxemia, and lactic acidosis), and what potential adverse effects the drug has in the sepsis syndrome.

In this study, patients meeting specific criteria (including elevated temperature) for a diagnosis of sepsis were recruited if they fulfilled an additional set of study criteria in the intensive care unit at one of seven participating centers.

The full trial involved 455 patients, of which our sample includes 300. 150 of our patients were randomly assigned to the Ibuprofen group and 150 to the Placebo group[1]. I picked the `sepsis` sample we will work with excluding patients with missing values for our outcome of interest, and then selected a random sample of 150 Ibuprofen and 150 Placebo patients from the rest of the group, and converted the temperatures and changes from Fahrenheit to Celsius. The data are gathered in the `sepsis` data file.

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

For the moment, we focus on two variables:

- **treat**, which specifies the treatment group (intravenous Ibuprofen or intravenous Placebo), which was assigned via randomization to each patient, and
- **temp_drop**, the outcome of interest, measured as the change from baseline to 2 hours later in degrees Celsius. Positive values indicate improvement, that is, a *drop* in temperature over the 2 hours following the baseline measurement.

The `sepsis.csv` file also contains each subject's

- *id*, which is just a code
- *race* (three levels: White, AfricanA or Other)
- *apache* = baseline APACHE II score, a severity of disease score ranging from 0 to 71 with higher scores indicating more severe disease and a higher mortality risk
- *temp_0* = baseline temperature, degrees Celsius.

but for the moment, we won't worry about those.

```
sepsis <- sepsis |>
    mutate(treat = factor(treat),
           race = factor(race))

summary(sepsis)
```

```
      id                 treat          race          apache
 Length:300         Ibuprofen:150   AfricanA: 80   Min.   : 0.0
 Class :character   Placebo  :150   Other   : 23   1st Qu.:10.0
 Mode  :character                   White   :197   Median :14.0
```

---

[1]This was a *double-blind* study, where neither the patients nor their care providers know, during the execution of the trial, what intervention group was assigned to each patient.

```
                                     Mean   :15.4
                                     3rd Qu.:20.0
                                     Max.   :35.0

     temp_0            temp_drop
 Min.   :33.10   Min.    :-2.7000
 1st Qu.:37.48   1st Qu.:-0.1000
 Median :38.20   Median : 0.3000
 Mean   :38.00   Mean    : 0.3083
 3rd Qu.:38.70   3rd Qu.: 0.7000
 Max.   :41.70   Max.    : 3.1000
```

20.3 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.

20.3.1 Model-Based Comparisons and ANOVA/Regression

Comparisons based on independent samples of quantitative variables are also frequently accomplished through other equivalent methods, including the analysis of variance approach and dummy variable regression, both of which produce identical confidence intervals to the pooled variance t test for the same comparison.

We will also discuss some of the main ideas in developing, designing and analyzing statistical experiments, specifically in terms of making comparisons. The ideas we will present in this

section allow for the comparison of more than two populations in terms of their population means. The statistical techniques employed analyze the sample variance in order to test and estimate the population means and for this reason the method is called the analysis of variance (ANOVA), and we will discuss this approach alone, and within the context of a linear regression model using dummy or indicator variables.

20.4 Key Questions for Comparing with Independent Samples

20.4.1 What is the population under study?

- All patients in the intensive care unit with sepsis who meet the inclusion and exclusion criteria of the study, at the entire population of health centers like the ones included in the trial.

20.4.2 What is the sample? Is it representative of the population?

- The sample consists of 300 patients. It is a convenient sample from the population under study.
- This is a randomized clinical trial. 150 of the patients were assigned to Ibuprofen, and the rest to Placebo. It is this treatment assignment that is randomized, not the selection of the sample as a whole.
- In expectation, randomization of individuals to treatments, as in this study, should be expected to eliminate treatment selection bias.

20.4.3 Who are the subjects / individuals within the sample?

- 150 patients who received Ibuprofen and a completely different set of 150 patients who received Placebo.
- There is no match or link between the patients. They are best thought of as independent samples.

20.4.4 What data are available on each individual?

- The key variables are the treatment indicator (Ibuprofen or Placebo) and the outcome (drop in temperature in the 2 hours following administration of the randomly assigned treatment.)

20.4.5 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.

20.5 Exploratory Data Analysis

Consider the following boxplot with violin of the `temp_drop` data within each `treat` group.

```
ggplot(sepsis, aes(x = treat, y = temp_drop, fill = treat)) +
    geom_violin() +
    geom_boxplot(width = 0.3, fill = "white") +
    scale_fill_viridis_d() +
    guides(fill = "none") +
    labs(title = "Boxplot of Temperature Drop in Sepsis Patients",
        x = "", y = "Drop in Temperature (degrees C)") +
    coord_flip()
```


Boxplot of Temperature Drop in Sepsis Patients
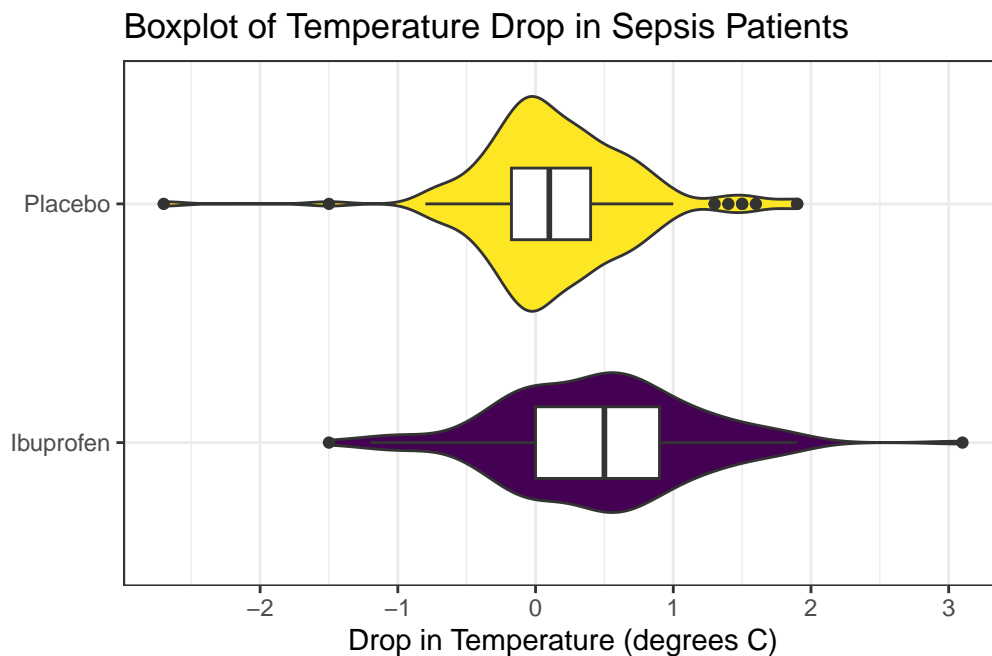
Next, we'll consider faceted histograms of the data.

```
ggplot(sepsis, aes(x = temp_drop, fill = treat, color = treat)) +
    geom_histogram(bins = 20) +
    scale_fill_viridis_d() +
    scale_color_viridis_d(direction = -1) +
    guides(fill = "none", color = "none") +
    labs(title = "Histograms of Temperature Drop in Sepsis Patients",
        x = "Drop in Temperature (degrees Celsius") +
    facet_wrap(~ treat)
```


Histograms of Temperature Drop in Sepsis Patients
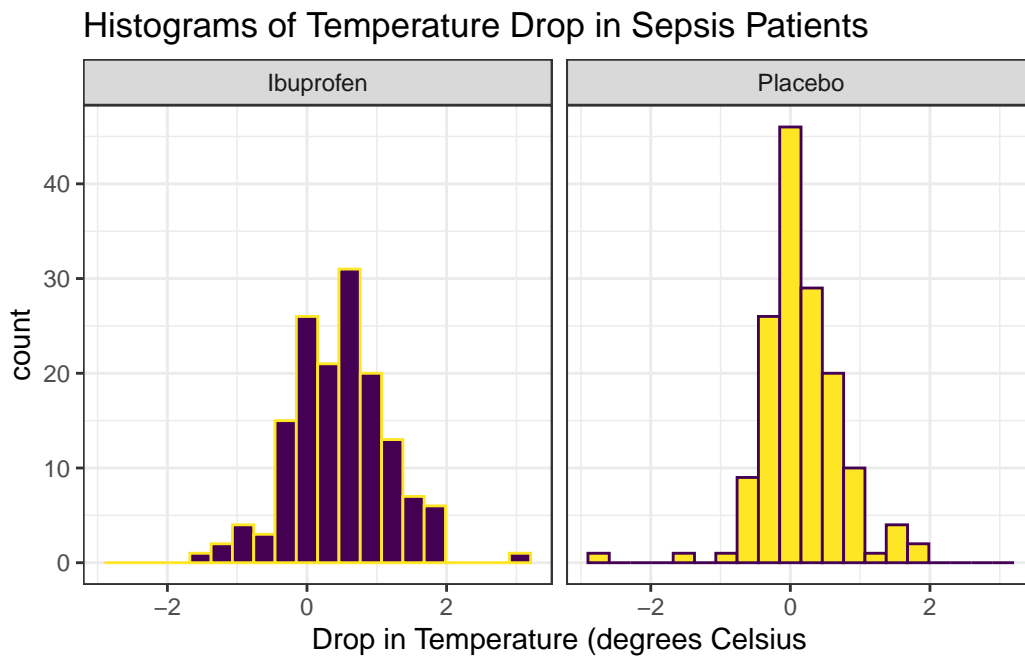
Here's a pair of Normal Q-Q plots. It's not hard to use a Normal model to approximate the Ibuprofen data, but such a model is probably not a good choice for the Placebo results.

```
ggplot(sepsis, aes(sample = temp_drop)) +
    geom_qq() + geom_qq_line(col = "red") +
    facet_wrap(~ treat) +
    labs(y = "Temperature Drop Values (in degrees C)")
```
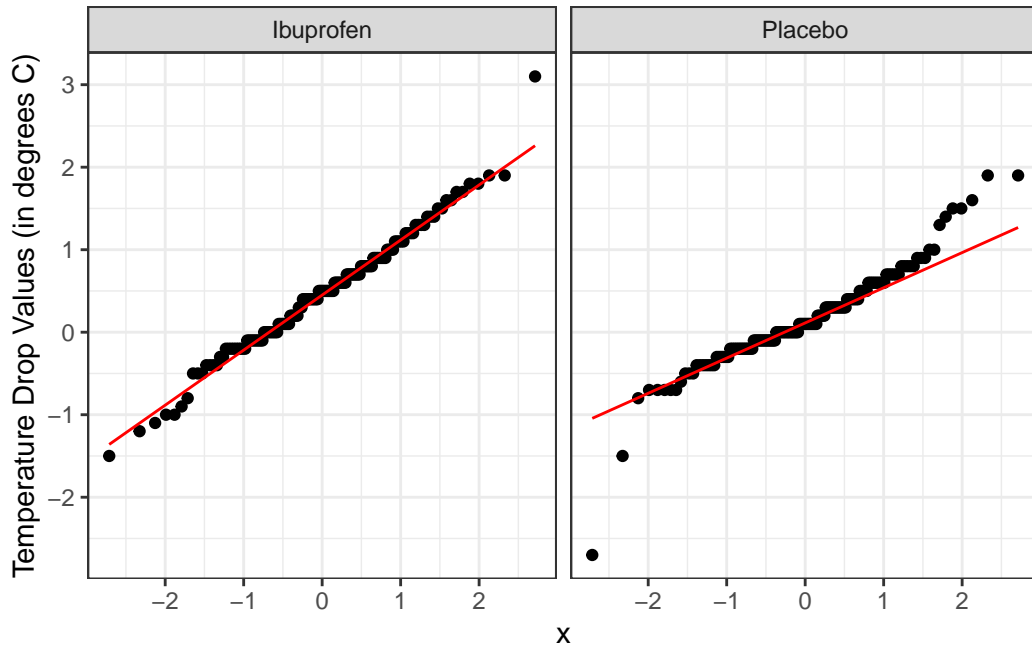
We'll could perhaps also look at a ridgeline plot, with some extra functions and theming from the `ggridges` package.

```
ggplot(sepsis, aes(x = temp_drop, y = treat, fill = treat)) +
    geom_density_ridges(scale = 0.9) +
    guides(fill = "none") +
    labs(title = "Temperature Drop in Sepsis Patients",
         x = "Drop in Temperature (degrees Celsius)", y = "") +
    theme_ridges()
```

```
Picking joint bandwidth of 0.182
```

Temperature Drop in Sepsis Patients
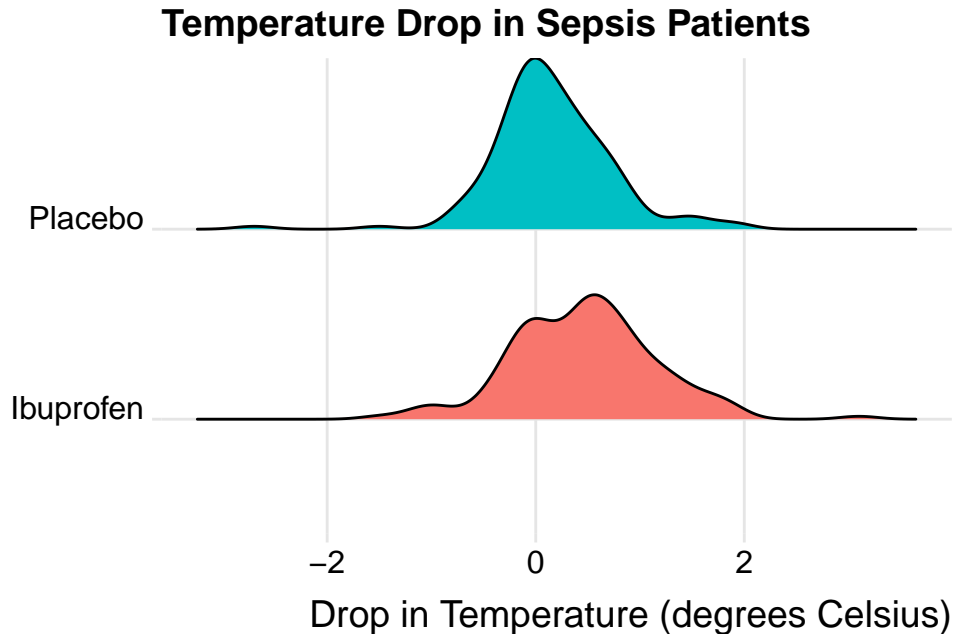
The center of the ibuprofen distribution is shifted a bit towards the more positive (greater improvement) direction, it seems, than is the distribution for the placebo patients. This conclusion matches what we see in some key numerical summaries, within the treatment groups.

```
mosaic::favstats(temp_drop ~ treat, data = sepsis)
```

```
    treat  min     Q1 median  Q3 max      mean        sd   n missing
1 Ibuprofen -1.5  0.000    0.5 0.9 3.1 0.4640000 0.6877919 150       0
2   Placebo -2.7 -0.175    0.1 0.4 1.9 0.1526667 0.5709637 150       0
```

20.6 Estimating the Difference in Population Means

Next, we will build a point estimate and 90% confidence interval for the difference between the mean `temp_drop` if treated with Ibuprofen and the mean `temp_drop` if treated with Placebo. We'll use a regression model with a single predictor (the `treat` group) to do this.

```
model_sep <- lm(temp_drop ~ treat == "Ibuprofen", data = sepsis)

tidy(model_sep, conf.int = TRUE, conf.level = 0.90) |>
    kbl(digits = 3) |> kable_styling()
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	0.153	0.052	2.958	0.003	0.068	0.238
treat == "Ibuprofen"TRUE	0.311	0.073	4.266	0.000	0.191	0.432

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	0.464	0.052	8.991	0	0.379	0.549
treatPlacebo	-0.311	0.073	-4.266	0	-0.432	-0.191

The point estimate for the "Ibuprofen - Placebo" difference in population means is 0.311 degrees C, and the 90% confidence interval is (0.191, 0.432) degrees C.

We could also have run the model like this:

```
model_sep2 <- lm(temp_drop ~ treat, data = sepsis)

tidy(model_sep2, conf.int = TRUE, conf.level = 0.90) |>
    kbl(digits = 3) |> kable_styling()
```

and would therefore conclude that the *Placebo - Ibuprofen* difference was estimated as -0.311, with 90% confidence interval (-0.432, -0.191), which is of course equivalent to our previous estimate.

Fundamentally, this regression model approach is identical to a **two-sample t test, assuming equal population variances**, also called a **pooled t test**. This is just one possible way for us to estimate the difference between population means, as it turns out.

20.7 t-based CI for population mean1 - mean2 difference

20.7.1 The Pooled t procedure

The most commonly used t-procedure for building a confidence interval assumes not only that each of the two populations being compared follows a Normal distribution, but also that they have the same population variance. This is the pooled t-test, and it is what people usually mean when they describe a two-sample t test.

```
t.test(temp_drop ~ treat,
       data = sepsis,
       conf.level = 0.90,
       alt = "two.sided",
       var.equal = TRUE)
```

estimate	estimate1	estimate2	statistic	p.value	parameter	conf.low	conf.high	method
0.31	0.46	0.15	4.27	0	298	0.19	0.43	Two Sample t-test

```
    Two Sample t-test

data:  temp_drop by treat
t = 4.2656, df = 298, p-value = 2.68e-05
alternative hypothesis: true difference in means between group Ibuprofen and group Placebo is
90 percent confidence interval:
 0.1909066 0.4317600
sample estimates:
mean in group Ibuprofen    mean in group Placebo
            0.4640000                0.1526667
```

Or, we can use `tidy` on this object:

```
tt1 <- t.test(temp_drop ~ treat,
              data = sepsis,
              conf.level = 0.90,
              alt = "two.sided",
              var.equal = TRUE)
tidy(tt1) |> kbl(digits = 2) |> kable_styling()
```

20.7.2 Using linear regression to obtain a pooled t confidence interval

As we've seen, and will demonstrate again below, a linear regression model, using the same outcome and predictor (group) as the pooled t procedure, produces the same confidence interval, again, under the assumption that the two populations we are comparing follow a Normal distribution with the same (population) variance.

```
model1 <- lm(temp_drop ~ treat, data = sepsis)

tidy(model1, conf.int = TRUE, conf.level = 0.90)
```

```
# A tibble: 2 x 7
  term          estimate std.error statistic  p.value conf.low conf.high
  <chr>            <dbl>     <dbl>     <dbl>    <dbl>    <dbl>     <dbl>
1 (Intercept)      0.464    0.0516      8.99 2.91e-17    0.379     0.549
2 treatPlacebo    -0.311    0.0730     -4.27 2.68e- 5   -0.432    -0.191
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	0.464	0.052	8.991	0	0.362	0.566
treatPlacebo	-0.311	0.073	-4.266	0	-0.455	-0.168

We see that our point estimate from the linear regression model is that the difference in `temp_drop` is -0.3113333, where Ibuprofen subjects have higher `temp_drop` values than do Placebo subjects, and that the 90% confidence interval for this difference ranges from -0.43176 to -0.1909066.

We can obtain a t-based confidence interval for each of the parameter estimates in a linear model directly using `tidy` from the `broom` package. Linear models usually summarize only the estimate and standard error. Remember that a reasonable approximation in large samples to a 95% confidence interval for a regression estimate (slope or intercept) can be obtained from estimate plus or minus two times the standard error.

```
tidy(model1, conf.int = TRUE, conf.level = 0.95) |>
  kbl(digits = 3) |> kable_styling()
```

So, in the case of the `treatPlacebo` estimate, we can obtain an approximate 95% confidence interval with (-0.457, -0.165). Compare this to the 95% confidence interval available from the model directly, shown in the tidied output above, or with the `confint` command below, and you'll see only a small difference.

Note that we can also use `summary` and `confint` to build our estimates.

```
summary(model1)
```

```
Call:
lm(formula = temp_drop ~ treat, data = sepsis)

Residuals:
    Min      1Q  Median      3Q     Max
-2.85267 -0.36400 -0.05267  0.34733  2.63600

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.46400    0.05161   8.991  < 2e-16 ***
treatPlacebo -0.31133    0.07299  -4.266 2.68e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 0.6321 on 298 degrees of freedom
Multiple R-squared:  0.05755,   Adjusted R-squared:  0.05438
F-statistic:  18.2 on 1 and 298 DF,  p-value: 2.68e-05
```

```
confint(model1, level = 0.95)
```

```
                   2.5 %      97.5 %
(Intercept)    0.3624351   0.5655649
treatPlacebo  -0.4549679  -0.1676988
```

20.7.3 The Welch t procedure

The default confidence interval based on the t test for independent samples in R uses something called the Welch test, in which the two populations being compared are not assumed to have the same variance. Each population is assumed to follow a Normal distribution.

```
t.test(temp_drop ~ treat, data = sepsis, conf.level = 0.90, alt = "two.sided")
```

```
    Welch Two Sample t-test

data:  temp_drop by treat
t = 4.2656, df = 288.24, p-value = 2.706e-05
alternative hypothesis: true difference in means between group Ibuprofen and group Placebo is
90 percent confidence interval:
 0.1908939 0.4317728
sample estimates:
mean in group Ibuprofen    mean in group Placebo
              0.4640000                0.1526667
```

Tidying works in this situation, too.

```
tt0 <- t.test(temp_drop ~ treat,
              data = sepsis, conf.level = 0.90, alt = "two.sided")
```

```
tidy(tt0) |> kbl(digits = 2) |> kable_styling()
```

When there is a *balanced design*, that is, when the same number of observations appear in each of the two samples, then the Welch t test and the Pooled t test produce the same confidence interval. Differences appear if the sample sizes in the two groups being compared are different.

estimate	estimate1	estimate2	statistic	p.value	parameter	conf.low	conf.high	method
0.31	0.46	0.15	4.27	0	288.24	0.19	0.43	Welch Two Sample

estimate	statistic	p.value	conf.low	conf.high	method	alt
0.3	14614.5	0	0.2	0.4	Wilcoxon rank sum test with continuity correction	two

20.8 Wilcoxon-Mann-Whitney "Rank Sum" CI

As in the one-sample case, a rank-based alternative attributed to Wilcoxon (and sometimes to Mann and Whitney) provides a two-sample comparison of the pseudomedians in the two `treat` groups in terms of `temp_drop`. This is called a **rank sum** test, rather than the Wilcoxon **signed rank** test that is used for inference about a single sample. Here's the resulting 90% confidence interval for the difference in pseudomedians.

```
wt <- wilcox.test(temp_drop ~ treat, data = sepsis,
                  conf.int = TRUE, conf.level = 0.90,
                  alt = "two.sided")

wt
```

```
	Wilcoxon rank sum test with continuity correction

data:  temp_drop by treat
W = 14614, p-value = 7.281e-06
alternative hypothesis: true location shift is not equal to 0
90 percent confidence interval:
 0.1999699 0.4000330
sample estimates:
difference in location
             0.3000368
```

```
tidy(wt) |> kbl(digits = 2) |> kable_styling()
```

20.9 Bootstrapping: A More Robust Approach

Within a script called `Love-boost.R`, I have provided the following R code to create a function called `bootdif`.

```
bootdif <-
  function(y, g, conf.level=0.95, B.reps = 2000) {
    lowq = (1 - conf.level)/2
    g <- as.factor(g)
    a <- attr(Hmisc::smean.cl.boot(y[g==levels(g)[1]], B=B.reps, reps=TRUE),'reps')
    b <- attr(Hmisc::smean.cl.boot(y[g==levels(g)[2]], B=B.reps, reps=TRUE),'reps')
    meandif <- diff(tapply(y, g, mean, na.rm=TRUE))
    a.b <- quantile(b-a, c(lowq,1-lowq))
    res <- c(meandif, a.b)
    names(res) <- c('Mean Difference',lowq, 1-lowq)
    res
  }
```

Running this code will place a new function called `bootdif` in your environment, which will help us calculate an appropriate confidence interval using a bootstrap procedure. The `bootdif` function contained in the `Love-boost.R` script is a slightly edited version of the function at http://biostat.mc.vanderbilt.edu/wiki/Main/BootstrapMeansSoftware.

20.9.1 Bootstrap CI for the Sepsis study

Note that this approach uses a comma to separate the outcome variable (here, `temp_drop`) from the variable identifying the exposure groups (here, `treat`).

```
set.seed(431212)

bootdif(sepsis$temp_drop, sepsis$treat, conf.level = 0.90)
```

```
Mean Difference                0.05                0.95
     -0.3113333          -0.4313667          -0.1833000
```

This approach calculates a 90% confidence interval for the difference in means between the two treatment groups. Note that the sign is in the opposite direction from what we've seen in our previous work. We can tell from the mean difference (and the summarized means from the data in each group) that this approach is finding a confidence interval using a bootstrap procedure for the Placebo - Ibuprofen difference, specifically (-0.431, -0.183).

```
mosaic::favstats(temp_drop ~ treat, data = sepsis)
```

```
      treat  min      Q1 median  Q3 max      mean        sd   n missing
1 Ibuprofen -1.5   0.000    0.5 0.9 3.1 0.4640000 0.6877919 150       0
2   Placebo -2.7  -0.175    0.1 0.4 1.9 0.1526667 0.5709637 150       0
```

To find a confidence interval using this bootstrap approach for the Ibuprofen - Placebo difference, we just need to switch the signs, and conclude that the 90% bootstrap confidence interval for that difference would be (0.183, 0.431).

20.10 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 significance level (or, 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?
8. If we have paired samples, what does the distribution of sample paired differences tell us about which inferential procedure to use?
9. If we have independent samples, what does the distribution of each individual sample tell us about which inferential procedure to use?

20.11 Results for the `sepsis` study

1. The outcome is `temp_drop`, the change in body temperature (in °C) from baseline to 2 hours later, so that positive numbers indicate drops in temperature (a good outcome.)
2. The groups are **Ibuprofen** and **Placebo** as contained in the `treat` variable in the `sepsis` tibble.
3. The data were collected using independent samples. The Ibuprofen subjects are not matched or linked to individual Placebo subjects - they are separate groups.
4. The subjects of the study aren't drawn from a random sample of the population of interest, but they are randomly assigned to their respective treatments (Ibuprofen and Placebo) which will provide the reasoned basis for our inferences.
5. We'll use a 10% significance level (or 90% confidence level) in this setting, as we did in our previous work on these data.
6. We'll use a two-sided testing and confidence interval approach.

Questions 7 and 8 don't apply, because these are independent samples of data, rather than paired samples.

To address question 9, we'll need to look at the data in each sample, as we did previously to allow us to assess the Normality of the distributions of (separately) the `temp_drop` results in the Ibuprofen and Placebo groups. We'll repeat those below.

```
ggplot(sepsis, aes(x = treat, y = temp_drop, fill = treat)) +
    geom_violin() +
    geom_boxplot(width = 0.3, fill = "white") +
    stat_summary(fun = "mean", geom = "point",
                 shape = 23, size = 3, fill = "red") +
    scale_fill_viridis_d(alpha = 0.3) +
    guides(fill = "none") +
    labs(title = "Boxplot of Temperature Drop in Sepsis Patients",
         x = "", y = "Drop in Temperature (degrees C)") +
    coord_flip()
```


Boxplot of Temperature Drop in Sepsis Patients
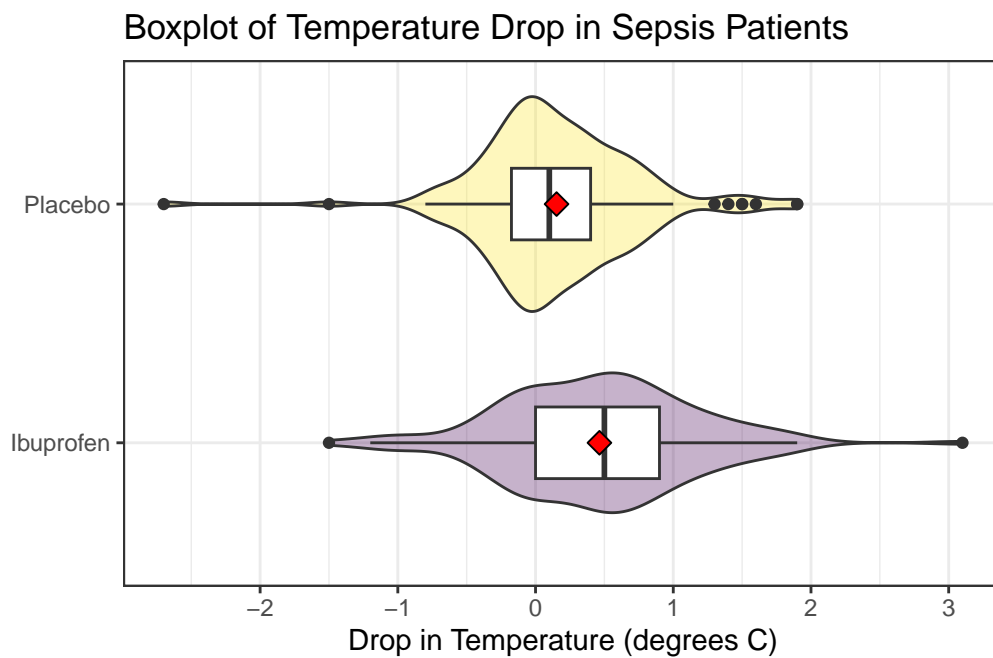
```
ggplot(sepsis, aes(sample = temp_drop)) +
    geom_qq() + geom_qq_line(col = "red") +
    facet_wrap(~ treat) +
    labs(y = "Temperature Drop Values (in degrees C)")
```
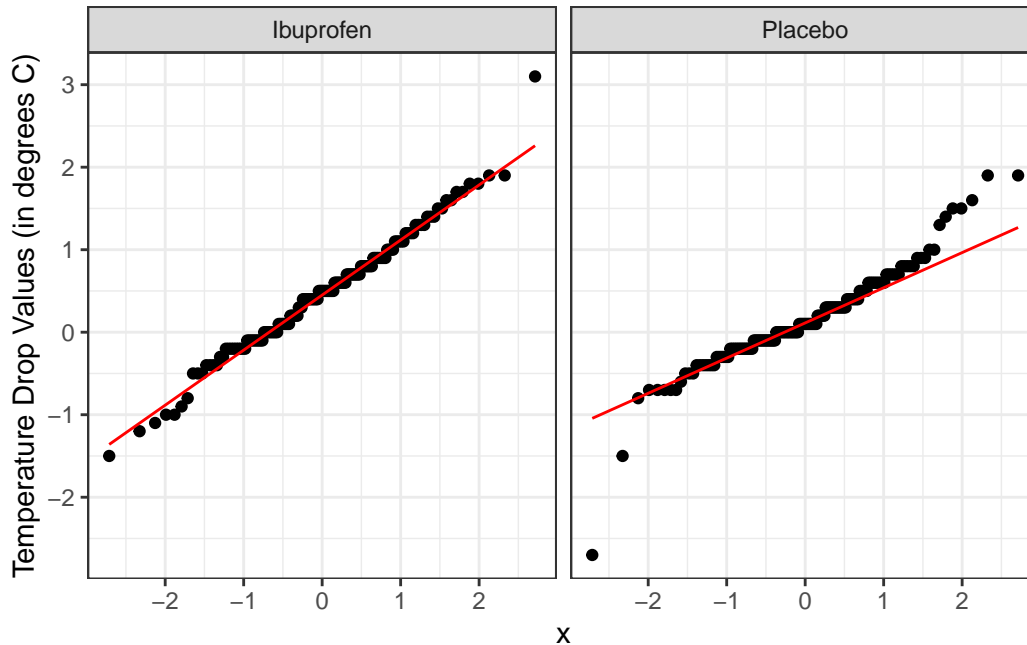
From these plots we conclude that the data in the Ibuprofen sample follow a reasonably Normal distribution, but this isn't quite as true for the Placebo sample. It's hard to know whether the apparent Placebo group outliers will affect whether the Normal distribution assumption is reasonable, so we can see if the confidence intervals change much when we *don't* assume Normality (for instance, comparing the bootstrap to the t-based approaches), as a way of understanding whether a Normal model has a large impact on our conclusions.

20.11.1 Sepsis Estimation Results

Here's a set of confidence interval estimates (we'll use 90% confidence here) using the methods discussed in this Chapter.

```
mosaic::favstats(temp_drop ~ treat, data = sepsis)
```

```
      treat  min     Q1 median  Q3 max      mean        sd   n missing
1 Ibuprofen -1.5  0.000    0.5 0.9 3.1 0.4640000 0.6877919 150       0
2   Placebo -2.7 -0.175    0.1 0.4 1.9 0.1526667 0.5709637 150       0
```

```
s_pooled_t_test <- t.test(temp_drop ~ treat, data = sepsis, conf.level = 0.90,
                          alt = "two.sided", var.equal = TRUE)
```

```
tidy(s_pooled_t_test) |>
    select(conf.low, conf.high)
```

```
# A tibble: 1 x 2
  conf.low conf.high
     <dbl>     <dbl>
1    0.191     0.432
```

```
s_welch_t_test <- t.test(temp_drop ~ treat, data = sepsis, conf.level = 0.90,
                         alt = "two.sided", var.equal = FALSE)

tidy(s_welch_t_test) |>
    select(estimate, conf.low, conf.high)
```

```
# A tibble: 1 x 3
  estimate conf.low conf.high
     <dbl>    <dbl>     <dbl>
1    0.311    0.191     0.432
```

```
s_wilcoxon_test <- wilcox.test(temp_drop ~ treat, data = sepsis,
                               conf.int = TRUE, conf.level = 0.90,
                               alt = "two.sided")

tidy(s_wilcoxon_test) |>
    select(estimate, conf.low, conf.high)
```

```
# A tibble: 1 x 3
  estimate conf.low conf.high
     <dbl>    <dbl>     <dbl>
1    0.300    0.200     0.400
```

```
set.seed(431212)
s_bootstrap <- bootdif(sepsis$temp_drop, sepsis$treat, conf.level = 0.90)

s_bootstrap
```

```
Mean Difference               0.05               0.95
    -0.3113333         -0.4313667         -0.1833000
```

Procedure	Compares...	Point Estimate	90% CI
Pooled t	Means	0.311	(0.191, 0.432)
Welch t	Means	0.311	(0.191, 0.432)
Bootstrap	Means	0.311	(0.183, 0.431)
Wilcoxon rank sum	Pseudo-Medians	0.3	(0.2, 0.4)

What conclusions can we draw in this setting?

20.12 Categorizing the Outcome and Comparing Rates

Suppose we were interested in comparing the percentage of patients in each arm of the trial (Ibuprofen vs. Placebo) that showed an improvement in their temperature (`temp_drop > 0`). To build the cross-tabulation of interest, we could create a new variable, called `dropped` which indicates whether the subject's temperature dropped, and then use `tabyl`.

```
sepsis <- sepsis |>
    mutate(dropped = ifelse(temp_drop > 0, "Drop", "No Drop"))

sepsis |> tabyl(treat, dropped)
```

```
    treat Drop No Drop
Ibuprofen  107      43
  Placebo   80      70
```

Our primary interest is in comparing the percentage of Ibuprofen patients whose temperature dropped to the percentage of Placebo patients whose temperature dropped.

```
sepsis |> tabyl(treat, dropped) |>
    adorn_totals() |>
    adorn_percentages(denom = "row") |>
    adorn_pct_formatting(digits = 1) |>
    adorn_ns(position = "front")
```

```
    treat         Drop      No Drop
Ibuprofen 107 (71.3%)   43 (28.7%)
  Placebo  80 (53.3%)   70 (46.7%)
    Total 187 (62.3%) 113 (37.7%)
```

20.13 Estimating the Difference in Proportions

In our sample, 71.3% of the Ibuprofen subjects, and 53.3% of the Placebo subjects, experienced a drop in temperature. So our *point estimate* of the difference in percentages would be 18.0 percentage points, but we will usually set this instead in terms of proportions, so that the difference is 0.180.

Now, we'll find a confidence interval for that difference, which we can do in several ways, including the `twoby2` function in the `Epi` package.

```
table(sepsis$treat, sepsis$dropped) |> twoby2(alpha = 0.10)
```

```
2 by 2 table analysis:
------------------------------------------------------
Outcome   : Drop
Comparing : Ibuprofen vs. Placebo

          Drop No Drop    P(Drop) 90% conf. interval
Ibuprofen  107      43     0.7133    0.6490   0.7701
Placebo     80      70     0.5333    0.4661   0.5993

                                 90% conf. interval
            Relative Risk: 1.3375    1.1492   1.5567
        Sample Odds Ratio: 2.1773    1.4583   3.2509
Conditional MLE Odds Ratio: 2.1716    1.4177   3.3437
   Probability difference: 0.1800    0.0881   0.2677

            Exact P-value: 0.0019
        Asymptotic P-value: 0.0014
------------------------------------------------------
```

While there is a lot of additional output here, we'll look for now just at the Probability difference row, where we see the point estimate (0.180) and the 90% confidence interval estimate for the difference in proportions (0.088, 0.268) comparing Ibuprofen vs. Placebo for the outcome of Dropping in Temperature.

More on estimation of the difference in population proportions will be found later.

21 Comparing Means with Paired Samples

Here, we'll consider the problem of estimating a confidence interval to describe the difference in population means (or medians) based on a comparison of two samples of quantitative data, gathered using a matched pairs design.

Specifically, we'll use as our example the Lead in the Blood of Children study, described below.

21.1 Setup: Packages Used Here

```r
knitr::opts_chunk$set(comment = NA)

source("data/Love-boost.R")
library(kableExtra)
library(broom)
library(ggridges)
library(Hmisc)
library(patchwork)
library(tidyverse)

theme_set(theme_bw())
```

In addition to the `Love-boost.R` script, we will also use the `favstats` function from the `mosaic` package.

21.2 Lead in the Blood of Children

> One of the best ways to eliminate a source of variation and the errors of interpretation associated with it is through the use of matched pairs. Each subject in one group is matched as closely as possible by a subject in the other group. If a 45-year-old African-American male with hypertension is given a [treatment designed to lower their blood pressure], then we give a second, similarly built 45-year old African-American male with hypertension a placebo.

- Good (2005), section 5.2.4

21.3 The Lead in the Blood of Children Study

Morton et al. (1982) studied the absorption of lead into the blood of children. This was a matched-sample study, where the exposed group of interest contained 33 children of parents who worked in a battery manufacturing factory (where lead was used) in the state of Oklahoma. Specifically, each child with a lead-exposed parent was matched to another child of the same age, exposure to traffic, and living in the same neighborhood whose parents did not work in lead-related industries. So the complete study had 66 children, arranged in 33 matched pairs. The outcome of interest, gathered from a sample of whole blood from each of the children, was lead content, measured in mg/dl.

One motivation for doing this study is captured in the Abstract from Morton et al. (1982).

> It has been repeatedly reported that children of employees in a lead-related industry are at increased risk of lead absorption because of the high levels of lead found in the household dust of these workers.

The data are available in several places, including Table 5 of Pruzek and Helmreich (2009), in the BloodLead data set within the `PairedData` package in R, but we also make them available in the `bloodlead.csv` file. 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
```

426

- In each pair, one child was exposed (to having a parent working in the factory) and the other was not.
- Otherwise, though, each child was very similar to its matched partner.
- The data under `exposed` and `control` are the blood lead content, in mg/dl.

Our primary goal will be 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.

21.3.1 Our Key Questions for a Paired Samples Comparison

1. What is the **population** under study?

- All pairs of children living in Oklahoma near the factory in question, in which one had a parent working in a factory that exposed them to lead, and the other did not.

2. What is the **sample**? Is it representative of the population?

- The sample consists of 33 pairs of one exposed and one control child.
- This is a case-control study, where the children were carefully enrolled to meet the design criteria. Absent any other information, we're likely to assume that there is no serious bias associated with these pairs, and that assuming they represent the population effectively (and perhaps the broader population of kids whose parents work in lead-based industries more generally) may well be at least as reasonable as assuming they don't.

3. Who are the subjects / **individuals** within the sample?

- Each of our 33 pairs of children includes one exposed child and one unexposed (control) child.

4. What **data** are available on each individual?

- The blood lead content, as measured in mg/dl of whole blood.

21.3.2 Lead Study Caveats

Note that the children were not randomly selected from general populations of kids whose parents did and did not work in lead-based industries.

- To make inferences to those populations, we must make **strong assumptions** to believe, for instance, that the sample of exposed children is as representative as a random sample of children with similar exposures across the world would be.

- The researchers did have a detailed theory about how the exposed children might be at increased risk of lead absorption, and in fact as part of the study gathered additional information about whether a possible explanation might be related to the quality of hygiene of the parents (all of them were fathers, actually) who worked in the factory.
- This is an observational study, so that the estimation of a causal effect between parental work in a lead-based industry and children's blood lead content can be made, without substantial (and perhaps heroic) assumptions.

21.4 Exploratory Data Analysis for Paired Samples

We'll begin by adjusting the data in two ways.

- We'd like that first variable (`pair`) to be a `factor` rather than a `character` type in R, because we want to be able to summarize it more effectively. So we'll make that change.
- Also, we'd like to calculate the difference in lead content between the exposed and the control children in each pair, and we'll save that within-pair difference in a variable called `lead_diff`. We'll take `lead_diff = exposed - control` so that positive values indicate increased lead in the exposed child.

```
bloodlead_original <- bloodlead

bloodlead <- bloodlead_original |>
    mutate(pair = factor(pair),
           lead_diff = exposed - control)

bloodlead
```

```
# A tibble: 33 x 4
   pair   exposed control lead_diff
   <fct>    <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
 7 P07         23      10        13
 8 P08         62      15        47
 9 P09         31      16        15
10 P10         34      18        16
# i 23 more rows
```

21.4.1 The Paired Differences

To begin, we focus on `lead_diff` for our exploratory work, which is the `exposed` - `control` difference in lead content within each of the 33 pairs. So, we'll have 33 observations, as compared to the 462 in the serum zinc data, but most of the same tools are still helpful.

```r
p1 <- ggplot(bloodlead, aes(sample = lead_diff)) +
  geom_qq(col = "steelblue") + geom_qq_line(col = "navy") +
  theme(aspect.ratio = 1) +
  labs(title = "Normal Q-Q plot")

p2 <- ggplot(bloodlead, aes(x = lead_diff)) +
  geom_histogram(aes(y = stat(density)),
                 binwidth = 5, fill = "steelblue", col = "white") +
  stat_function(fun = dnorm,
                args = list(mean = mean(bloodlead$lead_diff),
                            sd = sd(bloodlead$lead_diff)),
                col = "navy", lwd = 1.5) +
  labs(title = "Histogram with Normal Density")

p3 <- ggplot(bloodlead, aes(x = lead_diff, y = "")) +
  geom_boxplot(fill = "steelblue", outlier.color = "steelblue") +
  stat_summary(fun = "mean", geom = "point",
               shape = 23, size = 3, fill = "white") +
  labs(title = "Boxplot", y = "")

p1 + (p2 / p3 + plot_layout(heights = c(4,1))) +
    plot_annotation(title = "Difference in Blood Lead Content (mg/dl) for 33 Pairs of Chil
```

```
Warning: `stat(density)` was deprecated in ggplot2 3.4.0.
i Please use `after_stat(density)` instead.
```

min	Q1	median	Q3	max	mean	sd	n	missing
-9	4	15	25	60	15.97	15.86	33	0

Difference in Blood Lead Content (mg/dl) for 33 Pairs of Children
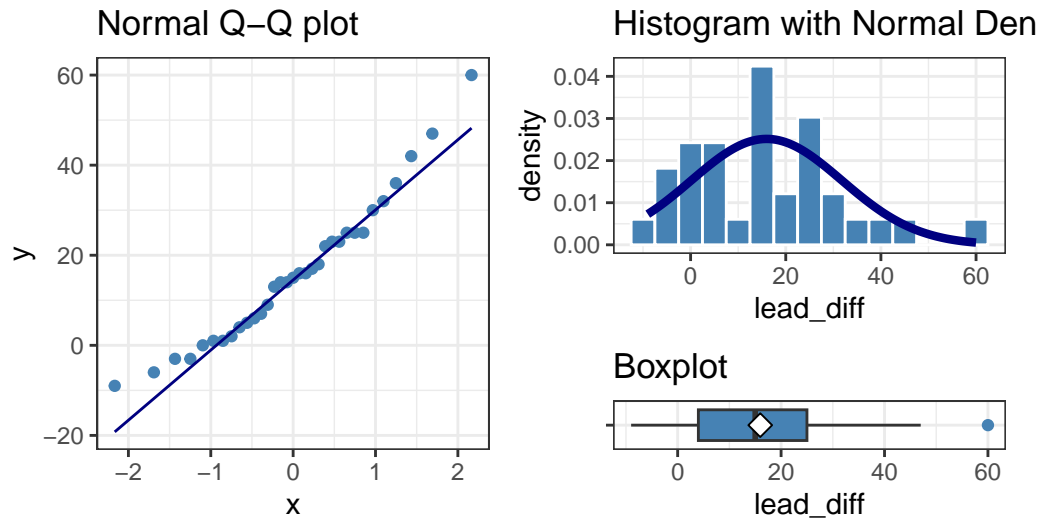
Note that in all of this work, I plotted the paired differences. One obvious way to tell if you have paired samples is that you can pair every single subject from one exposure group to a unique subject in the other exposure group. Everyone has to be paired, so the sample sizes will always be the same in the two groups.

Here's a summary of the paired differences.

```
mosaic::favstats(~ lead_diff, data = bloodlead) |>
    kbl(digits = 2) |> kable_styling()
```

```
bloodlead |> summarise(skew1 =
                        (mean(lead_diff) - median(lead_diff)) /
                        sd(lead_diff))
```

```
# A tibble: 1 x 1
  skew1
  <dbl>
1 0.0611
```

21.4.2 Impact of Matching - Scatterplot and Correlation

Here, the data are paired by the study through matching on neighborhood, age and exposure to traffic. Each individual child's outcome value is part of a pair with the outcome value for his/her matching partner. We can see this pairing in several ways, perhaps by drawing a scatterplot of the pairs.

```
ggplot(bloodlead, aes(x = control, y = exposed)) +
    geom_point(size = 2) +
    geom_smooth(method = "lm", formula = y ~ x, se = FALSE) +
    geom_text(x = 20, y = 65, col = "blue",
              label =
                  paste("Pearson r = ",
                        round(cor(bloodlead$control, bloodlead$exposed),2))) +
    labs(title = "Paired Samples in Blood Lead study",
         x = "Blood Lead Content (mg/dl) in Control Child",
         y = "Blood Lead Content (mg/dl) in Exposed Child")
```
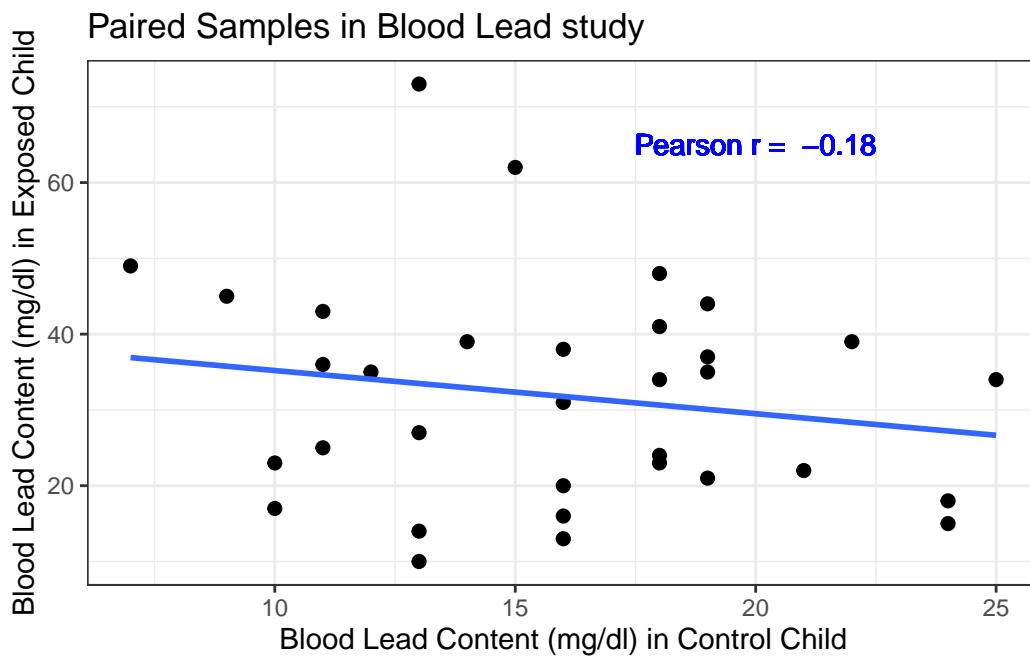

Each point here represents a **pair** of observations, one from a control child, and one from the matched exposed child. If there is a strong linear relationship (usually with a positive slope, thus positive correlation) between the paired outcomes, then the pairing will be more

helpful in terms of improving statistical power of the estimates we build than if there is a weak relationship.

- The stronger the Pearson correlation coefficient, the more helpful pairing will be.
- Here, a straight line model using the control child's blood lead content accounts for about 3.2% of the variation in blood lead content in the exposed child.
- As it turns out, pairing will have only a modest impact here on the inferences we draw in the study. We still will treat the data as paired, despite this.

21.5 Looking at Separate Samples: Using `pivot_longer`

For the purpose of estimating the difference between the exposed and control children, the summaries of the paired differences are what we'll need.

In some settings, however, we might also look at a boxplot, or violin plot, or ridgeline plot that showed the distributions of exposed and control children separately. But we will run into trouble because one variable (blood lead content) is spread across multiple columns (control and exposed.) The solution is to "pivot" the tibble from its current format to build a new, tidy tibble. Because the data aren't *tidied* here, so that we have one row for each subject and one column for each variable, we have to do some work to get them in that form for our usual plotting strategy to work well.

- `pivot_longer()` "lengthens" the data, increasing the number of rows and decreasing the number of columns.
- `pivot_wider()` performs the inverse of that transformation, "widening" the data.

In our original `bloodlead` data, if we drop the `lead_diff` addition we made, we have *wide* data, with each row representing two different subjects.

```
head(bloodlead_original, 3)
```

```
# A tibble: 3 x 3
  pair   exposed control
  <chr>    <dbl>   <dbl>
1 P01         38      16
2 P02         23      18
3 P03         41      18
```

And what we want to accomplish is to have one row for each subject, instead of one row for each pair of subjects. So we want to make the data **longer**.

```r
bloodlead_longer <- bloodlead_original |>
    pivot_longer(
        cols = -c(pair),
        names_to = "status",
        values_to = "lead_level")

bloodlead_longer
```

```
# A tibble: 66 x 3
   pair  status  lead_level
   <chr> <chr>        <dbl>
 1 P01   exposed         38
 2 P01   control         16
 3 P02   exposed         23
 4 P02   control         18
 5 P03   exposed         41
 6 P03   control         18
 7 P04   exposed         18
 8 P04   control         24
 9 P05   exposed         37
10 P05   control         19
# i 56 more rows
```

For more on this approach (in this case, we're making the data "longer" and its opposite would be be making the data "wider"), visit the Tidy data chapter in Hadley Wickham and Grolemund (2023) and the tidyr repository on Github at https://github.com/tidyverse/tidyr.

And now, we can plot as usual to compare the two samples.

First, we'll look at a boxplot, showing all of the data.

```r
ggplot(bloodlead_longer, aes(x = status, y = lead_level)) +
    geom_violin() +
    geom_boxplot(aes(fill = status), width = 0.2) +
    stat_summary(fun = "mean", geom = "point",
                 shape = 23, size = 3, fill = "blue") +
    scale_fill_viridis_d(alpha = 0.5) +
    guides(fill = "none") +
    coord_flip() +
    labs(title = "Boxplot of Lead Content in Exposed and Control kids")
```

Boxplot of Lead Content in Exposed and Control kids
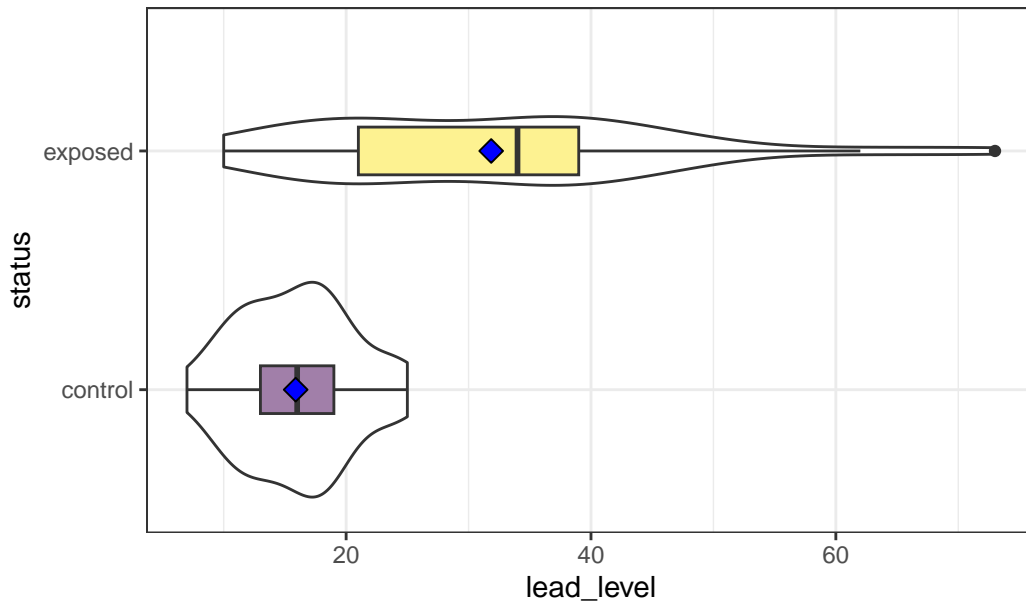
We'll also look at a ridgeline plot, because Dr. Love likes them, even though they're really more useful when we're comparing more than two samples.

```
ggplot(bloodlead_longer, aes(x = lead_level, y = status, fill = status)) +
    geom_density_ridges(scale = 0.9) +
    guides(fill = "none") +
    labs(title = "Lead Content in Exposed and Control kids") +
    theme_ridges()
```

```
Picking joint bandwidth of 4.01
```

status	min	Q1	median	Q3	max	mean	sd	n	missing
control	7	13	16	19	25	15.88	4.54	33	0
exposed	10	21	34	39	73	31.85	14.41	33	0

Lead Content in Exposed and Control kids
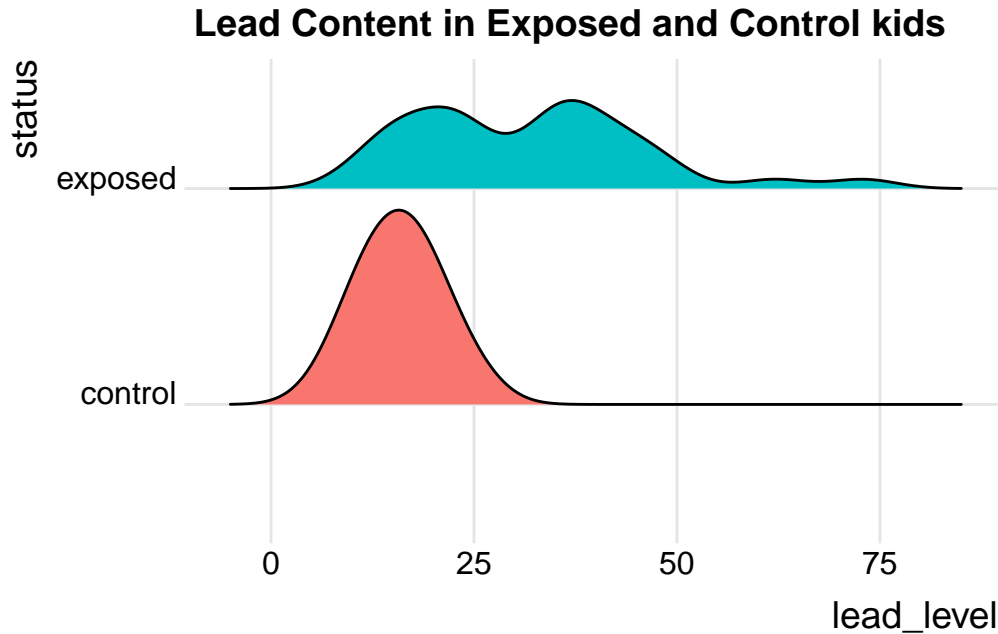
Both the center and the spread of the distribution are substantially larger in the exposed group than in the controls. Of course, numerical summaries show these patterns, too.

```
mosaic::favstats(lead_level ~ status, data = bloodlead_longer) |>
    kbl(digits = 2) |> kable_styling()
```

21.6 Estimating the Difference in Means with Paired Samples

Suppose we want to estimate the difference in the mean blood level across the population of children represented by the sample taken in this study. To do so, we must take advantage of the matched samples design, and complete our estimation on the paired differences, treating them as if they were a single sample of data.

One way to accomplish this is simply to run the usual intercept-only linear regression model on the paired differences.

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	15.97	2.76	5.78	0	11.29	20.65

```
model_lead <- lm(lead_diff ~ 1, data = bloodlead)

tidy(model_lead, conf.int = TRUE, conf.level = 0.90) |>
    kbl(digits = 2) |> kable_styling()
```

Our point estimate for the difference (exposed - control) in lead levels is 15.97 mg/dl, and our 90% confidence interval is (11.29, 20.65) mg/dl.

21.6.1 Paired Data in Longer Format?

If we had the data in "longer" format, as in `bloodlead_longer`, with the pairs identified by the `pair` variable, then we could obtained the same confidence interval using:

```
model2_lead <- lm(lead_level ~ status + factor(pair), data = bloodlead_longer)

tidy(model2_lead, conf.int = TRUE, conf.level = 0.90) |>
    kbl(digits = 2) |> kable_styling()
```

and the key elements are found in the `statusexposed` row, which we can focus on nicely (since the output of the `tidy()` function is always a tibble) with:

```
tidy(model2_lead, conf.int = TRUE, conf.level = 0.90) |>
    filter(term == "statusexposed") |>
    kbl(digits = 2) |> kable_styling()
```

and again, we have our 90% confidence interval estimate of the population mean difference between exposed and control children.

21.7 Matched Pairs vs. Two Independent Samples

These data were NOT obtained from two independent samples, but rather from matched pairs.

- We only have matched pairs if each individual observation in the "treatment" group is matched to one and only one observation in the "control" group by the way in which the data were gathered. Paired (or matched) data can arise in several ways.

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	19.02	8.05	2.36	0.02	5.38	32.65
statusexposed	15.97	2.76	5.78	0.00	11.29	20.65
factor(pair)P02	-6.50	11.22	-0.58	0.57	-25.50	12.50
factor(pair)P03	2.50	11.22	0.22	0.83	-16.50	21.50
factor(pair)P04	-6.00	11.22	-0.53	0.60	-25.00	13.00
factor(pair)P05	1.00	11.22	0.09	0.93	-18.00	20.00
factor(pair)P06	-3.50	11.22	-0.31	0.76	-22.50	15.50
factor(pair)P07	-10.50	11.22	-0.94	0.36	-29.50	8.50
factor(pair)P08	11.50	11.22	1.03	0.31	-7.50	30.50
factor(pair)P09	-3.50	11.22	-0.31	0.76	-22.50	15.50
factor(pair)P10	-1.00	11.22	-0.09	0.93	-20.00	18.00
factor(pair)P11	-6.00	11.22	-0.53	0.60	-25.00	13.00
factor(pair)P12	-13.50	11.22	-1.20	0.24	-32.50	5.50
factor(pair)P13	-7.00	11.22	-0.62	0.54	-26.00	12.00
factor(pair)P14	-13.50	11.22	-1.20	0.24	-32.50	5.50
factor(pair)P15	-11.00	11.22	-0.98	0.33	-30.00	8.00
factor(pair)P16	-9.00	11.22	-0.80	0.43	-28.00	10.00
factor(pair)P17	-7.50	11.22	-0.67	0.51	-26.50	11.50
factor(pair)P18	-15.50	11.22	-1.38	0.18	-34.50	3.50
factor(pair)P19	0.00	11.22	0.00	1.00	-19.00	19.00
factor(pair)P20	-0.50	11.22	-0.04	0.96	-19.50	18.50
factor(pair)P21	-5.50	11.22	-0.49	0.63	-24.50	13.50
factor(pair)P22	0.00	11.22	0.00	1.00	-19.00	19.00
factor(pair)P23	1.00	11.22	0.09	0.93	-18.00	20.00
factor(pair)P24	6.00	11.22	0.53	0.60	-13.00	25.00
factor(pair)P25	4.50	11.22	0.40	0.69	-14.50	23.50
factor(pair)P26	-3.50	11.22	-0.31	0.76	-22.50	15.50
factor(pair)P27	0.00	11.22	0.00	1.00	-19.00	19.00
factor(pair)P28	3.50	11.22	0.31	0.76	-15.50	22.50
factor(pair)P29	2.50	11.22	0.22	0.83	-16.50	21.50
factor(pair)P30	-12.50	11.22	-1.11	0.27	-31.50	6.50
factor(pair)P31	16.00	11.22	1.43	0.16	-3.00	35.00
factor(pair)P32	-9.00	11.22	-0.80	0.43	-28.00	10.00
factor(pair)P33	-7.00	11.22	-0.62	0.54	-26.00	12.00

term	estimate	std.error	statistic	p.value	conf.low	conf.high
statusexposed	15.97	2.76	5.78	0	11.29	20.65

- The most common is a "pre-post" study where subjects are measured both before and after an exposure happens.
- In observational studies, we often match up subjects who did and did not receive an exposure so as to account for differences on things like age, sex, race and other covariates. This is what happens in the Lead in the Blood of Children study.

- If the data are from paired samples, we should (and in fact) must form paired differences, with no subject left unpaired.

  - If we cannot line up the data comparing two samples of quantitative data so that the links between the individual "treated" and "control" observations to form matched pairs are evident, then the data are not paired.
  - If the sample sizes were different, we'd know we have independent samples, because matched pairs requires that each subject in the "treated" group be matched to a single, unique member of the "control" group, and thus that we have exactly as many "treated" as "control" subjects.
  - But having as many subjects in one treatment group as the other (which is called a *balanced design*) is only necessary, and not sufficient, for us to conclude that matched pairs are used.

As Bock, Velleman, and De Veaux (2004) suggest,

> ... if you know the data are paired, you can take advantage of that fact - in fact, you *must* take advantage of it. ... You must decide whether the data are paired from understanding how they were collected and what they mean. ... There is no test to determine whether the data are paired.

21.8 Estimating the Population Mean of the Paired Differences

There are two main approaches used frequently to estimate the population mean of paired differences.

- Estimation using the t distribution (and assuming at least an approximately Normal distribution for the paired differences)
- Estimation using the bootstrap (which doesn't require the Normal assumption)

In addition, we might consider estimating an alternate statistic when the data don't follow a symmetric distribution, like the median, with the bootstrap. In other settings, a rank-based alternative called the Wilcoxon signed rank test is available to estimate a psuedo-median. All of these approaches mirror what we did with a single sample, earlier in these Notes.

estimate	statistic	p.value	parameter	conf.low	conf.high	method	alternative
15.97	5.78	0	32	11.29	20.65	One Sample t-test	two.sided

21.9 t-based CI for Population Mean of Paired Differences

In R, there are at least five different methods for obtaining the t-based confidence interval for the population difference in means between paired samples. They are all mathematically identical. The key idea is to calculate the paired differences (exposed - control, for example) in each pair, and then treat the result as if it were a single sample and apply the methods developed for that situation earlier in these Notes.

21.9.1 Method 1

We can use the single-sample approach, applied to the variable containing the paired differences. Let's build a **90%** two-sided confidence interval for the population mean of the difference in blood lead content across all possible pairs of an exposed (parent works in a lead-based industry) and a control (parent does not) child.

```
tt1 <- t.test(bloodlead$lead_diff, conf.level = 0.90,
                          alt = "two.sided")

tt1
```

```
    One Sample t-test

data:  bloodlead$lead_diff
t = 5.783, df = 32, p-value = 2.036e-06
alternative hypothesis: true mean is not equal to 0
90 percent confidence interval:
 11.29201 20.64738
sample estimates:
mean of x
  15.9697
```

```
tidy(tt1) |> kbl(digits = 2) |> kable_styling()
```

The 90% confidence interval is (11.29, 20.65) according to this t-based procedure. An appropriate interpretation of the 90% two-sided confidence interval would be:

estimate	statistic	p.value	parameter	conf.low	conf.high	method	alternative
15.97	5.78	0	32	10.34	21.59	One Sample t-test	two.sided

- (11.29, 20.65) milligrams per deciliter is a 90% two-sided confidence interval for the population mean difference in blood lead content between exposed and control children.
- Our point estimate for the true population difference in mean blood lead content is 15.97 mg.dl. The values in the interval (11.29, 20.65) mg/dl represent a reasonable range of estimates for the true population difference in mean blood lead content, and we are 90% confident that this method of creating a confidence interval will produce a result containing the true population mean difference.
- Were we to draw 100 samples of 33 matched pairs from the population described by this sample, and use each such sample to produce a confidence interval in this manner, approximately 90 of those confidence intervals would cover the true population mean difference in blood lead content levels.

21.9.2 Method 2

Or, we can apply the single-sample approach to a calculated difference in blood lead content between the exposed and control groups. Here, we'll get a **95%** two-sided confidence interval for that difference, instead of the 90% interval we obtained above.

```
tt2 <- t.test(bloodlead$exposed - bloodlead$control,
        conf.level = 0.95, alt = "two.sided")

tt2
```

```
    One Sample t-test

data:  bloodlead$exposed - bloodlead$control
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
```

```
tidy(tt2) |> kbl(digits = 2) |> kable_styling()
```

estimate	statistic	p.value	parameter	conf.low	conf.high	method	alternative
15.97	5.78	0	32	9.21	Inf	Paired t-test	greater

21.9.3 Method 3

Or, we can provide R with two separate samples (unaffected and affected) and specify that the samples are paired. Here, we'll get a **99% one-sided** confidence interval (lower bound) for the population mean difference in blood lead content.

```r
tt3 <- t.test(bloodlead$exposed, bloodlead$control, conf.level = 0.99,
        paired = TRUE, alt = "greater")

tt3
```

```
    Paired t-test

data:  bloodlead$exposed and bloodlead$control
t = 5.783, df = 32, p-value = 1.018e-06
alternative hypothesis: true mean difference is greater than 0
99 percent confidence interval:
 9.207658      Inf
sample estimates:
mean difference
       15.9697
```

```r
tidy(tt3) |> kbl(digits = 2) |> kable_styling()
```

Again, the three different methods using `t.test` for paired samples will all produce identical results if we feed them the same confidence level and type of interval (two-sided, greater than or less than).

21.9.4 Method 4

We can also use an intercept-only linear regression model to estimate the population mean of the paired differences with a two-tailed confidence interval, by creating a variable containing those paired differences.

```r
model_lead <- lm(lead_diff ~ 1, data = bloodlead)
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	15.97	2.76	5.78	0	10.34	21.59

term	estimate	std.error	statistic	p.value	conf.low	conf.high
statusexposed	15.97	2.76	5.78	0	10.34	21.59

```
tidy(model_lead, conf.int = TRUE, conf.level = 0.95) |>
  kbl(digits = 2) |> kable_styling()
```

21.9.5 Method 5

If we have the data in a longer format, with a variable identifying the matched pairs, we can use a different specification for a linear model to obtain the same estimate.

```
model2_lead <- lm(lead_level ~ status + factor(pair), data = bloodlead_longer)

tidy(model2_lead, conf.int = TRUE, conf.level = 0.95) |>
    filter(term == "statusexposed") |> kbl(digits = 2) |> kable_styling()
```

21.9.6 Assumptions

If we are building a confidence interval based on a sample of observations drawn from a population, then we must pay close attention to the assumptions of those procedures. The confidence interval procedure for the population mean paired difference using the t distribution assumes that:

1. We want to estimate the population mean paired difference.
2. We have drawn a sample of paired differences at random from the population of interest.
3. The sampled paired differences are drawn from the population set of paired differences independently and have identical distributions.
4. The population follows a Normal distribution. At the very least, the sample itself is approximately Normal.

21.10 Bootstrap CI for mean difference using paired samples

The same bootstrap approach is used for paired differences as for a single sample. We use the `smean.cl.boot()` function in the `Hmisc` package to obtain bootstrap confidence intervals for the population mean of the paired differences in blood lead content.

```
set.seed(431555)
smean.cl.boot(bloodlead$lead_diff, B = 1000, conf.int = 0.95)
```

```
    Mean    Lower    Upper
15.96970 10.81742 21.48788
```

Note that in this case, the confidence interval for the difference in means is a bit less wide than the 95% confidence interval generated by the t test, which was (10.34, 21.59). It's common for the bootstrap to produce a narrower range (i.e. an apparently more precise estimate) for the population mean, but it's not automatic that the endpoints from the bootstrap will be inside those provided by the t test, either.

For example, this bootstrap CI doesn't contain the t-test based interval, since its upper bound exceeds that of the t-based interval:

```
set.seed(431002)
smean.cl.boot(bloodlead$lead_diff, B = 1000, conf.int = 0.95)
```

```
    Mean    Lower    Upper
15.96970 10.81667 21.66667
```

This demonstration aside, the appropriate thing to do when applying the bootstrap to specify a confidence interval is select a seed and the number (B = 1,000 or 10,000, usually) of desired bootstrap replications, then run the bootstrap just once and move on, rather than repeating the process multiple times looking for a particular result.

21.10.1 Assumptions

The bootstrap confidence interval procedure for the population mean (or median) of a set of paired differences assumes that:

1. We want to estimate the population mean of the paired differences (or the population median).
2. We have drawn a sample of observations at random from the population of interest.
3. The sampled observations are drawn from the population of paired differences independently and have identical distributions.
4. We are willing to put up with the fact that different people (not using the same random seed) will get somewhat different confidence interval estimates using the same data.

As we've seen, a major part of the bootstrap's appeal is the ability to relax some assumptions.

estimate	statistic	p.value	conf.low	conf.high	method	a
15.5	499	0	11	20.5	Wilcoxon signed rank test with continuity correction	t

21.11 Wilcoxon Signed Rank-based CI for paired samples

We could also use the Wilcoxon signed rank procedure to generate a CI for the pseudo-median of the paired differences.

```r
wt <- wilcox.test(bloodlead$lead_diff, conf.int = TRUE,
                  conf.level = 0.90, exact = FALSE)
wt
```

```
    Wilcoxon signed rank test with continuity correction

data:  bloodlead$lead_diff
V = 499, p-value = 1.155e-05
alternative hypothesis: true location is not equal to 0
90 percent confidence interval:
 10.99992 20.49998
sample estimates:
(pseudo)median
      15.49996
```

```r
tidy(wt) |> kbl(digits = 2) |> kable_styling()
```

As in the one sample case, we can revise this code slightly to specify a different confidence level, or gather a one-sided rather than a two-sided confidence interval.

21.11.1 Assumptions

The Wilcoxon signed rank confidence interval procedure in working with paired differences assumes that:

1. We want to estimate the population **pseudo-median** of the paired differences.
2. We have drawn a sample of observations at random from the population of paired differences of interest.
3. The sampled observations are drawn from the population of paired differences independently and have identical distributions.

4. The population follows a symmetric distribution. At the very least, the sample itself shows no substantial skew, so that the sample pseudo-median is a reasonable estimate for the population median.

21.12 Choosing a Confidence Interval Approach

Suppose we want to find a confidence interval for the population mean difference between two populations based on matched pairs.

1. If we are willing to assume that the population distribution is **Normal**

   - we usually use a t-based CI.

2. If we are **unwilling** to assume that the population is Normal,

   - use a **bootstrap** procedure to get a CI for the population mean, or even the median
   - but are willing to assume the population is symmetric, consider a **Wilcoxon signed rank** procedure to get a CI for the median, rather than the mean.

The two methods you'll use most often are the bootstrap (especially if the data don't appear to be at least pretty well fit by a Normal model) and the t-based confidence intervals (if the data do appear to fit a Normal model reasonably well.)

21.13 Conclusions for the `bloodlead` study

Using any of these procedures, we would conclude that the null hypothesis (that the true mean of the paired Exposed - Control differences is 0 mg/dl) is not consonant with what we see in the 90% confidence interval.

Procedure	Comparing	90% CI
Paired t	Means	11.3, 20.6
Wilcoxon signed rank	Pseudo-medians	11, 20.5
Bootstrap CI	Means	11.6, 20.6

Note that **one-sided** or **one-tailed** hypothesis testing procedures work the same way for paired samples as they did for a single sample.

21.14 The Sign test

The **sign test** is something we've skipped in our discussion so far. It is a test for consistent differences between pairs of observations, just as the paired t, Wilcoxon signed rank and bootstrap for paired samples can provide. It has the advantage that it is relatively easy to calculate by hand, and that it doesn't require the paired differences to follow a Normal distribution. In fact, it will even work if the data are substantially skewed.

- Calculate the paired difference for each pair, and drop those with difference = 0.
- Let N be the number of pairs that remain, so there are 2N data points.
- Let W, the test statistic, be the number of pairs (out of N) in which the difference is positive.
- Assuming that H0 is true, then W follows a binomial distribution with probability 0.5 on N trials.

For example, consider our data on blood lead content:

```
bloodlead$lead_diff
```

```
 [1] 22   5 23 -6 18 25 13 47 15 16   6   1   2   7   0   4 -9 -3 36 25   1 16 42 30 25
[26] 23 32 17   9 -3 60 14 14
```

Difference	# of Pairs
Greater than zero	28
Equal to zero	1
Less than zero	4

So we have $N = 32$ pairs, with $W = 28$ that are positive. We then use the `binom.test` approach in R:

```
binom.test(x = 28, n = 32, p = 0.5,
           alternative = "two.sided")
```

```
    Exact binomial test

data:  28 and 32
number of successes = 28, number of trials = 32, p-value = 1.93e-05
alternative hypothesis: true probability of success is not equal to 0.5
```

```
95 percent confidence interval:
 0.7100516 0.9648693
sample estimates:
probability of success
                  0.875
```

- A one-tailed test can be obtained by substituting in "less" or "greater" as the alternative of interest.
- 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.

21.15 Paired (Dependent) 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 and those which should be analyzed using *independent* samples. A dependent 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. Here are a few thoughts on the subject.

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.

1. 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 dependent samples analysis (such as a paired t test) is not correct.

2. 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 dependent 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 dependent samples analysis. The link between the subjects is the subject itself - each exposed subject serves as its own control.

3. The second most common strategy indicating dependent 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.

4. In order for a dependent 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.

5. 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.

21.15.1 Three "Tricky" Examples

1. Suppose we take a convenient sample of 200 patients from the population of patients who complete a blood test in April 2017 including a check of triglycerides, and who have a triglyceride level in the high category (200 to 499 mg/dl). Next, we select a patient at random from this group of 200 patients, and then identify another patient from the group of 200 who is the same age (to within 2 years) and also the same sex. We then randomly assign our intervention to one of these two patients and usual care without our intervention to the other patient. We then set these two patients aside and return to our original sample, repeating the process until we cannot find any more patients in the same age range and of the same gender. This generates a total of 77 patients who receive the intervention and 77 who do not. If we are trying to assess the effect of our intervention on triglyceride level in October 2017 using this sample of 154 people, should we use dependent (paired) or independent samples?

2. Suppose we take a convenient sample of 77 patients from the population of patients who complete a blood test in April 2017 including a check of triglycerides, and who have a triglyceride level in the high category (200 to 499 mg/dl). Next, we take a convenient sample of 77 patients from the population of patients who complete a blood test in May 2017 including a check of triglycerides, and who have a triglyceride level in the high category (200 to 499 mg/dl). We flip a coin to determine whether the intervention will be given to each of the 77 patients from April 2017 (if the coin comes up "HEADS")

or instead to each of the 77 patients from May 2017 (if the coin comes up "TAILS"). Then, we assign our intervention to the patients seen in the month specified by the coin and assign usual care without our intervention to the patients seen in the other month. If we are trying to assess the effect of our intervention on triglyceride level in October 2017 using this sample of 154 people, should we use dependent (paired) or independent samples?

3. Suppose we take a convenient sample of 200 patients from the population of patients who complete a blood test in April 2017 including a check of triglycerides, and who have a triglyceride level in the high category (200 to 499 mg/dl). For each patient, we re-measure them again in October 2017, again checking their triglyceride level. But in between, we take the first 77 of the patients in a randomly sorted list and assign them to our intervention (which takes place from June through September 2017) and take an additional group of 77 patients from the remaining part of the list and assign them to usual care without our intervention over the same time period. If we are trying to assess the effect of our intervention on each individual's change in triglyceride level (from April/May to October) using this sample of 154 people, should we use dependent (paired) or independent samples?

Answers to these "tricky" examples appear at the end of this Chapter.

21.16 A More Complete Decision Support Tool: Comparing Means

1. Are these paired or independent samples?

2. If paired samples, then are the paired differences approximately Normally distributed?

   a. If yes, then a paired t test or confidence interval is likely the best choice.
   b. If no, is the main concern outliers (with generally symmetric data), or skew?
      1. If the paired differences appear to be generally symmetric but with substantial outliers, a Wilcoxon signed rank test is an appropriate choice, as is a bootstrap confidence interval for the population mean of the paired differences.
      2. If the paired differences appear to be seriously skewed, then we'll usually build a bootstrap confidence interval, although a sign test is another reasonable possibility, although it doesn't provide a confidence interval for the population mean of the paired differences.

3. If independent, is each sample Normally distributed?

   a. No –> use Wilcoxon-Mann-Whitney rank sum test or bootstrap via `bootdif`.
   b. Yes –> are sample sizes equal?
      1. Balanced Design (equal sample sizes) - use pooled t test
      2. Unbalanced Design - use Welch test

21.16.1 Answers for the Three "Tricky" Examples

Answer for 1. Our first task is to identify the outcome and the exposure groups. Here, we are comparing the distribution of our outcome (triglyceride level in October) across two exposures: (a) receiving the intervention and (b) not receiving the intervention. We have a sample of 77 patients receiving the intervention, and a different sample of 77 patients receiving usual care. Each of the 77 subjects receiving the intervention is matched (on age and sex) to a specific subject not receiving the intervention. So, we can calculate paired differences by taking the triglyceride level for the exposed member of each pair and subtracting the triglyceride level for the usual care member of that same pair. Thus our comparison of the exposure groups should be accomplished using a *dependent* samples analysis, such as a paired t test.

Answer for 2. Again, we begin by identifying the outcome (triglyceride level in October) and the exposure groups. Here, we compare two exposures: (a) receiving the intervention and (b) receiving usual care. We have a sample of 77 patients receiving the intervention, and a different sample of 77 patients receiving usual care. But there is no pairing or matching involved. There is no connection implied by the way that the data were collected that implies that, for example, patient 1 in the intervention group is linked to any particular subject in the usual care group. So we need to analyze the data using independent samples.

Answer for 3. Once again, we identify the outcome (now it is the within-subject *change* in triglyceride level from April to October) and the exposure groups. Here again, we compare two exposures: (a) receiving the intervention and (b) receiving usual care. We have a sample of 77 patients receiving the intervention, and a different sample of 77 patients receiving usual care. But again, there is no pairing or matching between the patients receiving the intervention and the patients receiving usual care. While each outcome value is a difference (or change) in triglyceride levels, there's no connection implied by the way that the data were collected that implies that, for example, patient 1 in the intervention group is linked to any particular subject in the usual care group. So, again, we need to analyze the data using independent samples.

For more background and fundamental material, you might consider the Wikipedia pages on Paired Difference Test and on Blocking (statistics).

22 Sample Size and Power for Means

22.1 Setup: Package Used Here

```r
knitr::opts_chunk$set(comment = NA)

library(pwr)
```

22.2 Power and Sample Size Considerations

I'll begin here by repeating some of the material first shared in Chapter 18. For most statistical tests, it is theoretically possible to estimate the power of the test in the design stage, (before any data are collected) for various sample sizes, so we can hone in on a sample size choice which will enable us to collect data only on as many subjects as are truly necessary.

A power calculation is likely the most common element of an scientific grant proposal on which a statistician is consulted. This is a fine idea in theory, but in practice...

- The tests that have power calculations worked out in intensive detail using R are mostly those with more substantial assumptions. Examples include t tests that assume population normality, common population variance and balanced designs in the independent samples setting, or paired t tests that assume population normality in the paired samples setting.
- These power calculations are also usually based on tests rather than confidence intervals, which would be much more useful in most settings. Simulation is your friend here.
- Even more unfortunately, this process of doing power and related calculations is **far more of an art than a science**.
- As a result, the value of many power calculations is negligible, since the assumptions being made are so arbitrary and poorly connected to real data.
- On several occasions, I have stood in front of a large audience of medical statisticians actively engaged in clinical trials and other studies that require power calculations for funding. When I ask for a show of hands of people who have had power calculations prior to such a study whose assumptions matched the eventual data perfectly, I get lots of laughs. It doesn't happen.

- Even the underlying framework that assumes a power of 80% with a significance level of 5% is sufficient for most studies is pretty silly.

All that said, I feel obliged to show you some examples of power calculations done using R, and provide some insight on how to make some of the key assumptions in a way that won't alert reviewers too much to the silliness of the enterprise. All of the situations described in this Chapter are toy problems, but they may be instructive about some fundamental ideas.

22.3 Sample Size in a One-Sample t test

For a t test, R can estimate any one of the following elements, given the other four, using the `power.t.test` command, for either a one-tailed or two-tailed single-sample t test…

- n = the sample size
- δ = delta = the true difference in population means between the null hypothesis value and a particular alternative
- s = sd = the true standard deviation of the population
- α = sig.level = the significance level for the test (maximum acceptable risk of Type I error)
- 1 - β = power = the power of the t test to detect the effect of size δ

22.3.1 A Toy Example

Suppose that in a recent health survey, the average beef consumption in the U.S. per person was 90 pounds per year. Suppose you are planning a new study to see if beef consumption levels have changed. You plan to take a random sample of 25 people to build your new estimate, and test whether the current pounds of beef consumed per year is 90. Suppose you want to do a two-sided (two-tailed) test at 95% confidence (so $\alpha = 0.05$), and that you expect that the true difference will need to be at least $\delta = 5$ pounds (i.e. 85 or less or 95 or more) in order for the result to be of any real, practical interest. Suppose also that you are willing to assume that the true standard deviation of the measurements in the population is 10 pounds.

That is, of course, a lot to suppose.

Now, we want to know what power the proposed experiment will have to detect a change of 5 pounds (or more) away from the original 90 pounds, with these specifications, and how tweaking these specifications will affect the power of the study.

So, we have - n = 25 data points to be collected - $\delta = 5$ pounds is the minimum clinically meaningful effect size - s = 10 is the assumed population standard deviation, in pounds per year - α is 0.05, and we'll do a two-sided test

22.3.2 Using the `power.t.test` function

```
power.t.test(n = 25, delta = 5, sd = 10, sig.level = 0.05,
             type="one.sample", alternative="two.sided")
```

```
    One-sample t test power calculation

              n = 25
          delta = 5
             sd = 10
      sig.level = 0.05
          power = 0.6697014
    alternative = two.sided
```

So, under this study design, we would expect to detect an effect of size $\delta = 5$ pounds with just under 67% power, i.e. with a probability of incorrect retention of H_0 of just about $1/3$. Most of the time, we'd like to improve this power, and to do so, we'd need to adjust our assumptions.

22.4 Changing Assumptions

We made assumptions about the sample size n, the minimum clinically meaningful effect size (change in the population mean) δ, the population standard deviation s, and the significance level α, not to mention decisions about the test, like that we'd do a one-sample t test, rather than another sort of test for a single sample, and that we'd do a two-tailed, or two-sided test. Often, these assumptions are tweaked a bit to make the power look more like what a reviewer/funder is hoping to see.

22.4.1 Increasing Sample Size Increases Power

Suppose, we committed to using more resources and gathering data from 40 subjects instead of the 25 we assumed initially – what effect would this have on our power?

```
power.t.test(n = 40, delta = 5, sd = 10, sig.level = 0.05,
             type="one.sample", alternative="two.sided")
```

```
One-sample t test power calculation

          n = 40
      delta = 5
         sd = 10
  sig.level = 0.05
      power = 0.8693979
alternative = two.sided
```

With more samples, we should have a more powerful test, able to detect the difference with greater probability. In fact, a sample of 40 paired differences yields 87% power. As it turns out, we would need at least 44 observations with this scenario to get to 90% power, as shown in the calculation below, which puts the power in, but leaves out the sample size.

```
power.t.test(power=0.9, delta = 5, sd = 10, sig.level = 0.05,
             type="one.sample", alternative="two.sided")
```

```
One-sample t test power calculation

          n = 43.99552
      delta = 5
         sd = 10
  sig.level = 0.05
      power = 0.9
alternative = two.sided
```

We see that we would need at least 44 observations to achieve 90% power. Note: we always round the sample size up in doing a power calculation – if this calculation had actually suggested n = 43.1 paired differences were needed, we would still have rounded up to 44.

22.4.2 Increasing Effect Size will increase Power

A larger effect should be easier to detect. If we go back to our original calculation, which had 67% power to detect an effect of size $\delta = 5$, and now change the desired effect size to $\delta = 6$ pounds (i.e. a value of 84 or less or 96 or more), we should obtain a more powerful design.

```
power.t.test(n = 25, delta = 6, sd = 10, sig.level = 0.05,
             type="one.sample", alternative="two.sided")
```

```
One-sample t test power calculation

            n = 25
        delta = 6
           sd = 10
    sig.level = 0.05
        power = 0.8207213
  alternative = two.sided
```

We see that this change in effect size from 5 to 6, leaving everything else the same, increases our power from 67% to 82%. To reach 90% power, we'd need to increase the effect size we were trying to detect to at least 6.76 pounds.

```
power.t.test(n = 25, power = 0.9, sd = 10, sig.level = 0.05,
             type="one.sample", alternative="two.sided")
```

```
One-sample t test power calculation

            n = 25
        delta = 6.759051
           sd = 10
    sig.level = 0.05
        power = 0.9
  alternative = two.sided
```

- Again, note that I am rounding up here.
- Using $\delta = 6.75$ would not quite make it to 90.00% power.
- Using $\delta = 6.76$ guarantees that the power will be 90% or more, and not just round up to 90%.

22.4.3 Decreasing the Standard Deviation will increase Power

The choice of standard deviation is usually motivated by a pilot study, or else pulled out of thin air - it's relatively easy to convince yourself that the true standard deviation might be a little smaller than you'd guessed initially. Let's see what happens to the power if we reduce the sample standard deviation from 10 pounds to 9. This should make the effect of 5 pounds easier to detect, because it will have smaller variation associated with it.

```
power.t.test(n = 25, delta = 5, sd = 9, sig.level = 0.05,
             type="one.sample", alternative="two.sided")
```

```
    One-sample t test power calculation

              n = 25
          delta = 5
             sd = 9
      sig.level = 0.05
          power = 0.759672
    alternative = two.sided
```

This change in standard deviation from 10 to 9, leaving everything else the same, increases our power from 67% to nearly 76%. To reach 90% power, we'd need to decrease the standard deviation of the population paired differences to no more than 7.39 pounds.

```
power.t.test(n = 25, delta = 5, sd = NULL, power = 0.9, sig.level = 0.05,
             type="one.sample", alternative="two.sided")
```

```
    One-sample t test power calculation

              n = 25
          delta = 5
             sd = 7.397486
      sig.level = 0.05
          power = 0.9
    alternative = two.sided
```

Note I am rounding down here.

- Using s = 7.4 pounds would not quite make it to 90.00% power.

Note also that in order to get R to treat the standard debviation as unknown, I must specify it as NULL in the formula.

22.4.4 Larger Significance Level increases Power

We can trade off some of our Type II error (lack of power) for Type I error. If we are willing to trade off some Type I error (as described by the α), we can improve the power. For instance, suppose we decided to run the original test with 90% confidence.

```
power.t.test(n = 25, delta = 5, sd = 10, sig.level = 0.1,
             type="one.sample", alternative="two.sided")
```

```
    One-sample t test power calculation

              n = 25
          delta = 5
             sd = 10
      sig.level = 0.1
          power = 0.7833861
    alternative = two.sided
```

The calculation suggests that our power would thus increase from 67% to just over 78%.

22.5 Paired Sample t Tests and Power/Sample Size

For a paired-samples t test, R can estimate any one of the following elements, given the other four, using the `power.t.test` command, for either a one-tailed or two-tailed paired t test...

- n = the sample size (# of pairs) being compared
- δ = delta = the true difference in means between the two groups
- s = sd = the true standard deviation of the paired differences
- α = sig.level = the significance level for the comparison (maximum acceptable risk of Type I error)
- 1 - β = power = the power of the paired t test to detect the effect of size δ

22.5.1 A Toy Example

As a toy example, suppose you are planning a paired samples experiment involving n = 30 subjects who will each provide a "Before" and an "After" result, which is measured in days.

Suppose you want to do a two-sided (two-tailed) test at 95% confidence (so $\alpha = 0.05$), and that you expect that the true difference between the "Before" and "After" groups will have to

be at least $\delta = 5$ days to be of any real interest. Suppose also that you are willing to assume that the true standard deviation of those paired differences will be 10 days.

That is, of course, a lot to suppose.

Now, we want to know what power the proposed experiment will have to detect this difference with these specifications, and how tweaking these specifications will affect the power of the study.

So, we have - n = 30 paired differences will be collected - $\delta = 5$ days is the minimum clinically meaningful difference - s = 10 days is the assumed population standard deviation of the paired differences - α is 0.05, and we'll do a two-sided test

22.5.2 Using the `power.t.test` function

```
power.t.test(n = 30, delta = 5, sd = 10, sig.level = 0.05,
             type="paired", alternative="two.sided")
```

```
    Paired t test power calculation

             n = 30
         delta = 5
            sd = 10
     sig.level = 0.05
         power = 0.7539627
   alternative = two.sided

NOTE: n is number of *pairs*, sd is std.dev. of *differences* within pairs
```

So, under this study design, we would expect to detect an effect of size $\delta = 5$ days with 75% power, i.e. with a probability of incorrect retention of H_0 of 0.25. Most of the time, we'd like to improve this power, and to do so, we'd need to adjust our assumptions.

22.5.3 Changing Assumptions in a Power Calculation

We made assumptions about the sample size n, the minimum clinically meaningful difference in means δ, the population standard deviation s, and the significance level α, not to mention decisions about the test, like that we'd do a paired t test, rather than another sort of test for paired samples, or use an independent samples approach, and that we'd do a two-tailed, or two-sided test. Often, these assumptions are tweaked a bit to make the power look more like what a reviewer/funder is hoping to see.

22.5.4 Changing the Sample Size

Suppose, we committed to using more resources and gathering "Before" and "After" data from 40 subjects instead of the 30 we assumed initially – what effect would this have on our power?

```
power.t.test(n = 40, delta = 5, sd = 10, sig.level = 0.05,
             type="paired", alternative="two.sided")
```

```
    Paired t test power calculation

              n = 40
          delta = 5
             sd = 10
      sig.level = 0.05
          power = 0.8693979
    alternative = two.sided

NOTE: n is number of *pairs*, sd is std.dev. of *differences* within pairs
```

With more samples, we should have a more powerful test, able to detect the difference with greater probability. In fact, a sample of 40 paired differences yields 87% power. As it turns out, we would need at least 44 paired differences with this scenario to get to 90% power, as shown in the calculation below, which puts the power in, but leaves out the sample size.

```
power.t.test(power=0.9, delta = 5, sd = 10, sig.level = 0.05,
             type="paired", alternative="two.sided")
```

```
    Paired t test power calculation

              n = 43.99552
          delta = 5
             sd = 10
      sig.level = 0.05
          power = 0.9
    alternative = two.sided

NOTE: n is number of *pairs*, sd is std.dev. of *differences* within pairs
```

We see that we would need at least 44 paired differences to achieve 90% power. Note: we always round the sample size up in doing a power calculation – if this calculation had actually suggested n = 43.1 paired differences were needed, we would still have rounded up to 44.

22.5.5 Changing the Effect Size

A larger effect should be easier to detect. If we go back to our original calculation, which had 75% power to detect an effect (i.e. a true population mean difference) of size $\delta = 5$, and now change the desired effect size to $\delta = 6$, we should obtain a more powerful design.

```
power.t.test(n = 30, delta = 6, sd = 10, sig.level = 0.05,
             type="paired", alternative="two.sided")
```

```
    Paired t test power calculation

            n = 30
        delta = 6
           sd = 10
    sig.level = 0.05
        power = 0.887962
  alternative = two.sided

NOTE: n is number of *pairs*, sd is std.dev. of *differences* within pairs
```

We see that this change in effect size from 5 to 6, leaving everything else the same, increases our power from 75% to nearly 89%. To reach 90% power, we'd need to increase the effect size we were trying to detect to at least 6.13 days.

- Again, note that I am rounding up here.
- Using $\delta = 6.12$ would not quite make it to 90.00% power.
- Using $\delta = 6.13$ guarantees that the power will be 90% or more, and not just round up to 90%..

22.5.6 Changing the Standard Deviation

The choice of standard deviation is usually motivated by a pilot study, or else pulled out of thin air. It's relatively easy to convince yourself that the true standard deviation might be a little smaller than you'd guessed initially. Let's see what happens to the power if we reduce the sample standard deviation from 10 days to 9 days. This should make the effect of 5 days

460

easier to detect as being different from the null hypothesized value of 0, because it will have smaller variation associated with it.

```
power.t.test(n = 30, delta = 5, sd = 9, sig.level = 0.05,
             type="paired", alternative="two.sided")
```

```
     Paired t test power calculation

              n = 30
          delta = 5
             sd = 9
      sig.level = 0.05
          power = 0.8366514
    alternative = two.sided
```

NOTE: n is number of *pairs*, sd is std.dev. of *differences* within pairs

This change in standard deviation from 10 to 9, leaving everything else the same, increases our power from 75% to nearly 84%. To reach 90% power, we'd need to decrease the standard deviation of the population paired differences to no more than 8.16 days.

Note I am rounding down here, because using $s = 8.17$ days would not quite make it to 90.00% power. Note also that in order to get R to treat the sd as unknown, I must specify it as NULL in the formula…

```
power.t.test(n = 30, delta = 5, sd = NULL, power = 0.9,
             sig.level = 0.05, type="paired", alternative="two.sided")
```

```
     Paired t test power calculation

              n = 30
          delta = 5
             sd = 8.163989
      sig.level = 0.05
          power = 0.9
    alternative = two.sided
```

NOTE: n is number of *pairs*, sd is std.dev. of *differences* within pairs

22.5.7 Changing the Significance Level

We can trade off some of our Type II error (lack of power) for Type I error. If we are willing to trade off some Type I error (as described by the α), we can improve the power. For instance, suppose we decided to run the original test with 90% confidence.

```
power.t.test(n = 30, delta = 5, sd = 10, sig.level = 0.1,
             type="paired", alternative="two.sided")
```

```
    Paired t test power calculation

            n = 30
        delta = 5
           sd = 10
    sig.level = 0.1
        power = 0.8482542
  alternative = two.sided
```

NOTE: n is number of *pairs*, sd is std.dev. of *differences* within pairs

The calculation suggests that our power would thus increase from 75% to nearly 85%.

22.6 Two Independent Samples: Power for t Tests

For an independent-samples t test, with a balanced design (so that $n_1 = n_2$), R can estimate any one of the following elements, given the other four, using the `power.t.test` command, for either a one-tailed or two-tailed t test...

- n = the sample size in each of the two groups being compared
- δ = delta = the true difference in means between the two groups
- s = sd = the true standard deviation of the individual values in each group (assumed to be constant – since we assume equal population variances)
- α = sig.level = the significance level for the comparison (maximum acceptable risk of Type I error)
- 1 - β = power = the power of the t test to detect the effect of size δ

This method only produces power calculations for balanced designs – where the sample size is equal in the two groups. If you want a two-sample power calculation for an unbalanced design, you will need to use a different library and function in R, as we'll see.

22.7 A New Example

Suppose we plan a study of the time to relapse for patients in a drug trial, where subjects will be assigned randomly to a (new) treatment or to a placebo. Suppose we anticipate that the placebo group will have a mean of about 9 months, and want to detect an improvement (increase) in time to relapse of 50%, so that the treatment group would have a mean of at least 13.5 months. We'll use $\alpha = .10$ and $\beta = .10$, as well. Assume we'd do a two-sided test, with an equal number of observations in each group, and we'll assume the observed standard deviation of 9 months in a pilot study will hold here, as well.

We want the sample size required by the test under a two sample setting where:

- $\alpha = .10$,
- with 90% power (so that $\beta = .10$),
- and where we will have equal numbers of samples in the placebo group (group 1) and the treatment group (group 2).

- We'll plug in the observed standard deviation of 9 months.
- We'll look at detecting a change from 9 [the average in the placebo group] to 13.5 (a difference of 50%, giving delta = 4.5)
- using a two-sided pooled t-test.

The appropriate R command is:

```
power.t.test(delta = 4.5, sd = 9,
             sig.level = 0.10, power = 0.9,
             type="two.sample",
             alternative="two.sided")
```

```
    Two-sample t test power calculation

          n = 69.19782
      delta = 4.5
         sd = 9
  sig.level = 0.1
      power = 0.9
alternative = two.sided

NOTE: n is number in *each* group
```

This suggests that we will need a sample of at least 70 subjects in the treated group and an additional 70 subjects in the placebo group, for a total of 140 subjects.

22.7.1 Another Scenario

What if resources are sparse, and we'll be forced to do the study with no more than 120 subjects, overall? If we require 90% confidence in a two-sided test, what power will we have?

```
power.t.test(n = 60, delta = 4.5, sd = 9,
             sig.level = 0.10,
             type="two.sample",
             alternative="two.sided")
```

```
    Two-sample t test power calculation

              n = 60
          delta = 4.5
             sd = 9
      sig.level = 0.1
          power = 0.859484
    alternative = two.sided

NOTE: n is number in *each* group
```

It looks like the power under those circumstances would be just under 86%. Note that the n = 60 refers to half of the total sample size, since we'll need 60 drug and 60 placebo subjects in this balanced design.

22.8 Power for Independent Sample T tests with Unbalanced Designs

Using the `pwr` package, R can do sample size calculations that describe the power of a two-sample t test that does not require a balanced design using the `pwr.t2n.test` command.

Suppose we wanted to do the same study as we described above, using 100 "treated" patients but as few "placebo" patients as possible. What sample size would be required to maintain 90% power? There is one change here – the effect size d in the `pwr.t2n.test` command is specified using the difference in means δ that we used previously, divided by the standard deviation s that we used previously. So, in our old setup, we assumed delta = 4.5, sd = 9, so now we'll assume d = 4.5/9 instead.

```
pwr.t2n.test(n1 = 100, d = 4.5/9,
             sig.level = 0.1, power = 0.9,
             alternative="two.sided")
```

```
     t test power calculation

            n1 = 100
            n2 = 52.82433
             d = 0.5
     sig.level = 0.1
         power = 0.9
   alternative = two.sided
```

We would need at least 53 subjects in the "placebo" group.

22.8.1 The most efficient design for an independent samples comparison will be balanced.

- Note that if we use $n_1 = 100$ subjects in the treated group, we need at least $n_2 = 53$ in the placebo group to achieve 90% power, and a total of 153 subjects.
- Compare this to the balanced design, where we needed 70 subjects in each group to achieve the same power, thus, a total of 140 subjects.

We saw earlier that a test with 60 subjects in each group would yield just under 86% power. Suppose we instead built a test with 80 subjects in the treated group, and 40 in the placebo group, then what would our power be?

```
pwr.t2n.test(n1 = 80, n2 = 40, d = 4.5/9,
             sig.level = 0.10,
             alternative="two.sided")
```

```
     t test power calculation

            n1 = 80
            n2 = 40
             d = 0.5
     sig.level = 0.1
         power = 0.821823
   alternative = two.sided
```

As we'd expect, the power is stronger for a balanced design than for an unbalanced design with the same overall sample size.

Note that I used a two-sided test to establish my power calculation – in general, this is the most conservative and defensible approach for any such calculation, **unless there is a strong and specific reason to use a one-sided approach in building a power calculation, don't**.

23 Two Examples Comparing Means

23.1 Setup: Packages Used Here

```
knitr::opts_chunk$set(comment = NA)

source("data/Love-boost.R")
library(kableExtra)
library(patchwork)
library(tidyverse)

theme_set(theme_bw())
```

In addition to the `Love-boost.R` script, we will also use the `favstats` function from the `mosaic` package.

23.2 A Study of Battery Life

Should you buy generic rather than brand-name batteries? Bock, Velleman, and De Veaux (2004) descibe a designed experiment to test battery life. A (male) student obtained six pairs of AA alkaline batteries from two major battery manufacturers; a well-known brand name and a generic brand, so that battery brand was the factor of interest.

To estimate the difference in mean lifetimes across the two manufacturers, the student kept a battery-powered CD player with the same CD running continuously, with the volume control fixed at 5, and measured the time until no more music was heard through the headphones. (He ran an initial trial to find out approximately how long that would take, so he didn't have to spend the first 3 hours of each run listening to the same CD.) The outcome was the time in minutes until the sound stopped. To account for changes in the CD player's performance over time, he randomized the run order by choosing pairs of batteries (the CD-player required two batteries to run) at random.

Here are the results for the 6 brand name and 6 generic tests, in minutes, found in the `battery.csv` data file, where `run` indicates the order in which the tests were run...

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

battery
```

```
# A tibble: 12 x 4
     run  test type          time
   <dbl> <dbl> <chr>        <dbl>
 1     1     1 brand name   191.
 2     2     2 brand name   206.
 3     6     3 brand name   199.
 4     8     4 brand name   172.
 5     9     5 brand name   184
 6    12     6 brand name   170.
 7     3     1 generic      194
 8     4     2 generic      204.
 9     5     3 generic      204.
10     7     4 generic      206.
11    10     5 generic      222.
12    11     6 generic      209.
```

23.2.1 Question 1. What is the outcome under study?

We are studying battery lifetimes (time until the sound stopped) in minutes.

23.2.2 Question 2. What are the treatment/exposure groups?

We are comparing the two brands of batteries: the well-known vs. the generic.

23.2.3 Question 3. Are the data collected using paired or independent samples?

Of course, if we had different numbers of samples in the two groups, then we'd know without further thought that independent samples were required. Since we have 6 observations in the brand name group, and also have 6 observations in the generic group, i.e. a balanced design, we need to pause now to decide whether paired or independent samples testing is appropriate in this setting.

Two samples are paired if each data point in one sample is naturally linked to a specific data point in the other sample. So, do we have paired or independent samples?

- Despite the way I've set up the data table, there is no particular reason to pair, say, run #1 (a brand name run) with any particular experimental run in the generic group. So the samples are independent. This is not a matched-pairs design.
- In each trial, the student either used two of the well-known batteries, or two of the generic batteries.
- Any of the tests/confidence intervals for the independent samples methods suggests a statistically significant (at the 5% level) difference between the generic and brand name batteries.

23.2.4 Question 4. Are the data a random sample from the population of interest?

Probably not. The data are likely to come from a convenient sample of batteries. I don't know how this might bias the study, though. It seems unlikely that there would be a particular bias unless, for example, the well-known batteries were substantially older or younger than the generic.

23.2.5 Question 5. What significance level will we use?

We have no reason not to use a 95% confidence level.

23.2.6 Question 6. Are we using a one-sided or two-sided comparison?

We could argue for a one-sided comparison, but I'll be safe and use the two-sided version.

23.2.7 Question 9. What does the distribution of outcomes in each group tell us?

```
ggplot(battery, aes(x = type, y = time, fill = type)) +
  geom_jitter(alpha = 0.75, width = 0.125) +
  geom_boxplot(alpha = 0.5) +
  stat_summary(fun = "mean", geom = "point",
               shape = 23, size = 3, fill = "white") +
  coord_flip() +
  guides(fill = "none", col = "none") +
  labs(title = "Battery Running Time, by Manufacturer",
       y = "Running Time (minutes)", x = "Manufacturer")
```

Battery Running Time, by Manufacturer
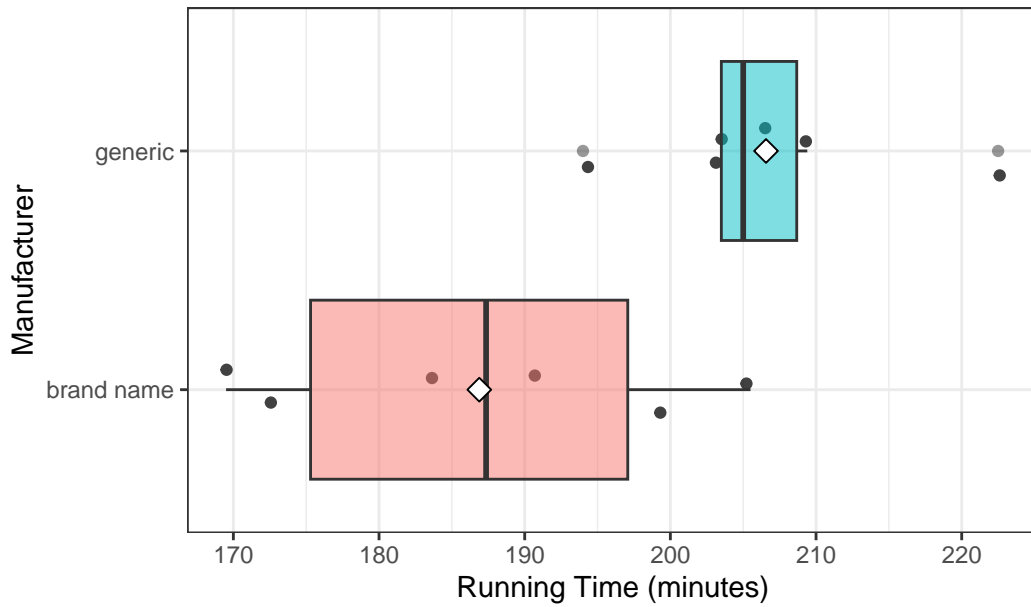
We can generate histograms, too, but that's an issue, because we have so few observations.

```
ggplot(battery, aes(x = time, fill = type)) +
  geom_histogram(bins = 6, col = "white") +
  facet_wrap(~ type) +
  guides(fill = "none") +
  labs(title = "Battery Running Time, by Manufacturer")
```

type	min	Q1	median	Q3	max	mean	sd	n	missing
brand name	169.5	175.3	187.35	197.07	205.5	186.88	14.37	6	0
generic	194.0	203.5	205.00	208.68	222.5	206.57	9.37	6	0

Battery Running Time, by Manufacturer
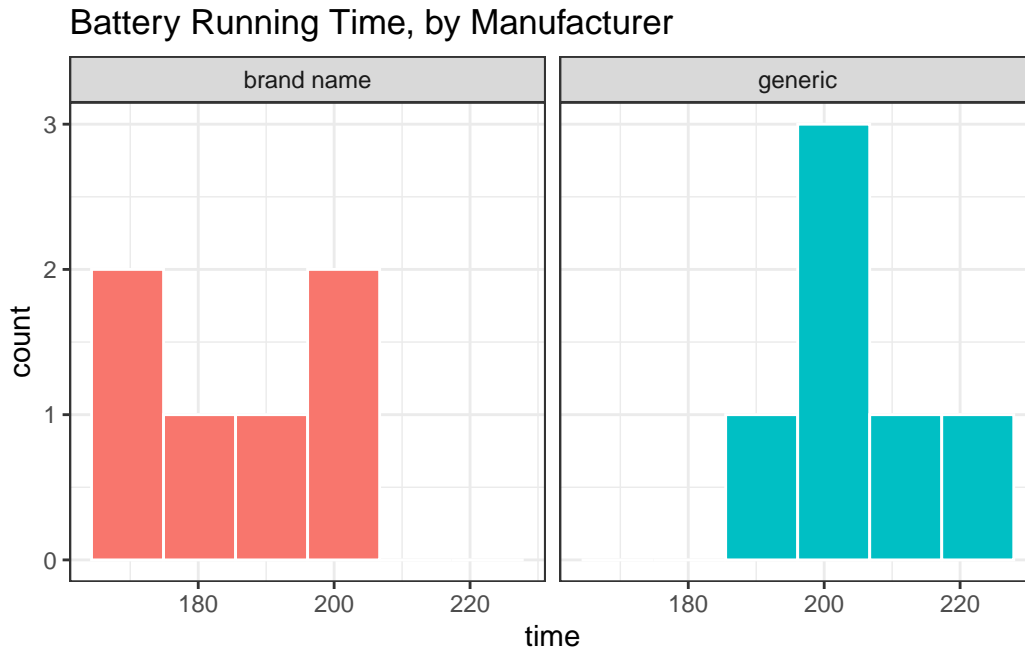
```
mosaic::favstats(time ~ type, data = battery) |>
  kbl(digits = 2) |> kable_styling()
```

It sure looks like the generic batteries lasted longer. And they also look like they were more consistent. The sample means are 206.6 for the generic group, 186.9 minutes for brand name, so the point estimate of the difference is 19.7 minutes.

The question is: can we be confident that the difference we observe here is more than just random fluctuation, at a 5% significance level?

23.2.8 Inferential Results for the Battery Study

In the table below, I have summarized the two-sided testing results for most of the ways in which we have looked at a two sample comparison so far, with 95% confidence intervals. If the samples really are paired, then we must choose from the paired samples comparisons described in the table. If the samples really are independent, then we must choose from the independent samples comparisons.

23.2.9 Paired Samples Approaches

Method	p Value	95% CI for Generic - Brand Name
Paired t	0.058	-1.0, 40.4
Wilcoxon signed rank	0.063	-2.0, 39.9
Bootstrap via `smean.cl.boot`	–	6.7, 33.0

23.2.10 Independent Samples Approaches

Method	p Value	95% CI for Generic - Brand Name
Pooled t	0.018	4.1, 35.3
Welch's t	0.021	3.7, 35.6
Wilcoxon Mann Whitney rank sum	0.030	3.3, 37.0
Bootstrap via `bootdif`	–	7.7, 32.2

23.3 The Breakfast Study: Does Oat Bran Cereal Lower Serum LDL Cholesterol?

Norman and Streiner (2014) describe a crossover study that was conducted to investigate whether oat bran cereal helps to lower serum cholesterol levels in hypercholesterolemic males. Fourteen such individuals were randomly placed on a diet that included either oat bran or corn flakes; after two weeks, their low-density lipoprotein (LDL) cholesterol levels, in mmol/l were recorded. Each subject was then switched to the alternative diet. After a second two-week period, the LDL cholesterol level of each subject was again recorded.

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

breakfast
```

```
# A tibble: 14 x 3
   subject cornflakes oatbran
     <dbl>      <dbl>   <dbl>
 1       1       4.61    3.84
 2       2       6.42    5.57
 3       3       5.4     5.85
 4       4       4.54    4.8
```

```
 5       5       3.98    3.68
 6       6       3.82    2.96
 7       7       5.01    4.41
 8       8       4.34    3.72
 9       9       3.8     3.49
10      10       4.56    3.84
11      11       5.35    5.26
12      12       3.89    3.73
13      13       2.25    1.84
14      14       4.24    4.14
```

23.3.1 Question 1. What is the outcome under study?

We are studying levels of LDL cholesterol, in mmol/l. Note that if we wanted to convert to a more familiar scale, specifically mg/dl, we would multiply the mmol/l by 18, as it turns out.

23.3.2 Question 2. What are the treatment/exposure groups?

We are comparing subjects after two weeks of eating corn flakes to the same subjects after two weeks of eating oat bran.

23.3.3 Question 3. Are the data collected using paired or independent samples?

These are matched pairs, paired by subject. Each subject produced an oat bran result and a corn flakes result.

23.3.4 Question 4. Are the data a random sample from the population of interest?

Probably not. The data are likely to come from a convenient sample of 14 individuals but they were randomly assigned to cornflakes first or to oat bran first, then crossed over.

23.3.5 Question 5. What significance level will we use?

We have no reason not to use our usual 95% confidence level, so `alpha` = 0.05

23.3.6 Question 6. Are we using a one-sided or two-sided comparison?

We could argue for a one-sided comparison, but I'll be safe and use the two-sided version.

23.3.7 Question 7. Did pairing help reduce nuisance variation?

After we drop the `breakfast.csv` file into the `breakfast` data frame, we look at the correlation of cornflakes and oatbran results across our 14 subjects.

```
cor(breakfast$cornflakes, breakfast$oatbran)
```

```
[1] 0.9233247
```

The sample Pearson correlation coefficient is very strong and positive at 0.92, so the paired samples approach will use these data far more effectively than the (incorrect) independent samples approach.

23.3.8 Question 8. What does the distribution of paired differences tell us?

We summarize the distribution of the paired differences (cornflakes - oatbran) below.

```
breakfast <- breakfast |>
  mutate(diffs = cornflakes - oatbran)

p1 <- ggplot(breakfast, aes(sample = diffs)) +
  geom_qq(col = "darkslategray") + geom_qq_line(col = "navy") +
  theme(aspect.ratio = 1) +
  labs(title = "Normal Q-Q plot")

p2 <- ggplot(breakfast, aes(x = diffs)) +
  geom_histogram(aes(y = stat(density)),
                 bins = 5, fill = "darkslategray", col = "white") +
  stat_function(fun = dnorm,
                args = list(mean = mean(breakfast$diffs),
                            sd = sd(breakfast$diffs)),
                col = "navy", lwd = 1.5) +
  labs(title = "Histogram with Normal Density")

p3 <- ggplot(breakfast, aes(x = diffs, y = "")) +
  geom_boxplot(fill = "darkslategray", outlier.color = "darkslategray") +
  stat_summary(fun = "mean", geom = "point",
               shape = 23, size = 3, fill = "white") +
  labs(title = "Boxplot", y = "")

p1 + (p2 / p3 + plot_layout(heights = c(4,1))) +
```

```
      plot_annotation(title = "Difference in LDL (Corn Flakes - Oat Bran)")
```

Difference in LDL (Corn Flakes – Oat Bran)
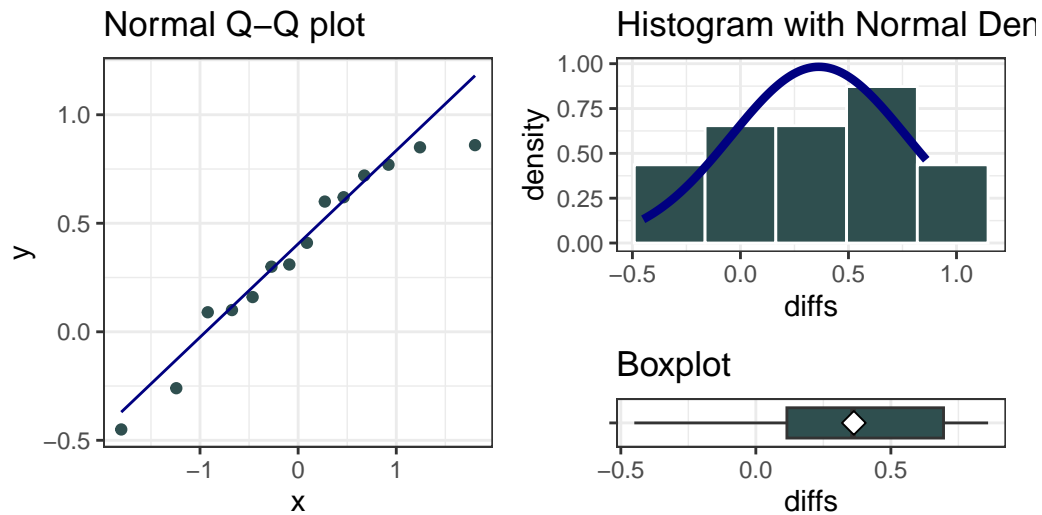
The Normal distribution doesn't look too ridiculous in this case for the paired (cornflakes-oatbran) differences. Suppose we assume Normality and run the paired t test.

```
t.test(breakfast$cornflakes - breakfast$oatbran)
```

```
    One Sample t-test

data:  breakfast$cornflakes - breakfast$oatbran
t = 3.3444, df = 13, p-value = 0.005278
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 0.1284606 0.5972537
sample estimates:
mean of x
0.3628571
```

Based on this sample of 14 subjects in the crossover study, we observe a 95% confidence interval for the difference between the LDL cholesterol levels after eating corn flakes and eating oat bran that is entirely positive, suggesting that LDL levels were detectably higher (according to this t test procedure) after eating corn flakes than after eating oat bran.

23.4 Power, Sample Size and the Breakfast Study

As a preview of what's next to come, let's investigate these promising results a bit further. Suppose that in a new study, you wish to be able to detect a difference in LDL cholesterol between two exposures: subjects who eat cornflakes (as in the original study) and subjects who continue to eat cornflakes but also take a supplemental dosage of what you believe to be the crucial ingredient in oatbran.

Suppose you believe that the effect of taking the new supplement will be about half the size of the effect you observed in the original breakfast study on hypercholesterolemic males, but that males generally may be more likely to take your supplement regularly than switch from cornflakes to a less appetizing breakfast choice, making your supplement attractive.

What sample size will be required to yield 90% power to detect an effect half the size of the effect we observed in the breakfast study, in a new paired samples study using a two-tailed 5% significance level? What if we only required 80% power?

23.4.1 The Setup

We want to know n, the minimum required sample size for the new study, and we have:

- A specified effect size of half of what we saw in the breakfast study, where the sample mean difference between cornflakes and oatbran was 0.36 mmol/l, so our effect size is assumed to be `delta` = 0.18 mmol/l.
- An assumed standard deviation equal to the standard deviation of the differences in the pilot breakfast study, which turns out to have been s = 0.41 mmol/l.
- We also have a pre-specified `alpha` = 0.05 using a two-tailed test.
- We also want the power to be at least 90% for our new study.

23.4.2 The R Calculations

Question 1. What sample size will be required to yield 90% power to detect an effect half the size of the effect we observed in the breakfast study, in a new paired samples study using a two-tailed 5% significance level?

```
power.t.test(delta = 0.18, sd = 0.41, sig.level = 0.05,
             power = 0.9, type="paired", alternative="two.sided")
```

```
     Paired t test power calculation

              n = 56.47119
          delta = 0.18
             sd = 0.41
      sig.level = 0.05
          power = 0.9
    alternative = two.sided
```

NOTE: n is number of *pairs*, sd is std.dev. of *differences* within pairs

And so our new study will require at least **57 subjects** (each measured in two circumstances, so 114 total measurements) in order to achieve at least 90% power to detect the difference of 0.18 mmol/l while meeting these specifications.

Question 2. What if we were willing to accept only 80% power?

```
power.t.test(delta = 0.18, sd = 0.41, sig.level = 0.05,
             power = 0.8, type="paired", alternative="two.sided")
```

```
     Paired t test power calculation

              n = 42.68269
          delta = 0.18
             sd = 0.41
      sig.level = 0.05
          power = 0.8
    alternative = two.sided
```

NOTE: n is number of *pairs*, sd is std.dev. of *differences* within pairs

It turns out that this would require at least **43 subjects**.

23.4.3 Independent samples, instead of paired samples?

What would happen if, instead of doing a paired samples study, we did one using independent samples? Assuming we used a balanced design, and assigned the same number of different people at random to either the oatbran supplement or regular cornflakes alone, we could do such a study, but it would require many more people to obtain similar power to the paired samples study.

```
power.t.test(delta = 0.18, sd = 0.41, sig.level = 0.05,
             power = 0.9, type="two.sample", alternative="two.sided")
```

```
    Two-sample t test power calculation

              n = 110
          delta = 0.18
             sd = 0.41
      sig.level = 0.05
          power = 0.9
    alternative = two.sided

NOTE: n is number in *each* group
```

In all, **220 people** would be required in the independent samples study (110 in each exposure group), as compared to only **57 people** (each measured twice) in the paired study.

24 Analysis of Variance

24.1 Setup: Packages Used Here

```
knitr::opts_chunk$set(comment = NA)

library(ggridges)
library(janitor)
library(mosaic)
library(tidyverse)

theme_set(theme_bw())
```

24.2 National Youth Fitness Survey

Recall the National Youth Fitness Survey, which we explored a small piece of in Chapter 10.
We'll look at a different part of the same survey here - specifically the 280 children whose data
are captured in the `nyfs2` file.

```
nyfs2 <- read_csv("data/nyfs2.csv", show_col_types = FALSE) |>
  clean_names()

nyfs2
```

```
# A tibble: 280 x 21
  subject_id sex    age_exam race_eth        english income_cat3 income_detail
       <dbl> <chr>     <dbl> <chr>             <dbl> <chr>        <chr>
1      73228 Male          4 5 Other or Mult~       1 Low (below~ 0 to 4999
2      72393 Male          4 2 Non-Hispanic ~       1 Low (below~ 0 to 4999
3      73303 Male          3 2 Non-Hispanic ~       1 Low (below~ 0 to 4999
4      72786 Male          5 1 Non-Hispanic ~       1 Low (below~ 0 to 4999
5      73048 Male          3 2 Non-Hispanic ~       1 Low (below~ 0 to 4999
6      72556 Female        4 2 Non-Hispanic ~       1 Low (below~ 0 to 4999
```

```
 7       72580 Female          5 2 Non-Hispanic ~       1 Low (below~ 0 to 4999
 8       72532 Female          4 4 Other Hispanic       0 Low (below~ 0 to 4999
 9       73012 Male            4 1 Non-Hispanic ~       1 Low (below~ 0 to 4999
10       72099 Male            6 1 Non-Hispanic ~       1 Low (below~ 0 to 4999
# i 270 more rows
# i 14 more variables: inc_to_pov <dbl>, weight_kg <dbl>, height_cm <dbl>,
#   bmi <dbl>, bmi_group <dbl>, bmi_cat <chr>, arm_length <dbl>,
#   arm_circ <dbl>, waist_circ <dbl>, calf_circ <dbl>, calf_skinfold <dbl>,
#   triceps_skinfold <dbl>, subscap_skinfold <dbl>, gmq <dbl>
```

24.3 Comparing Gross Motor Quotient Scores by Income Level (3 Categories)

```
nyfs2a <- nyfs2 |>
    select(subject_id, income_cat3, gmq) |>
    arrange(subject_id)
```

In this first analysis, we'll compare the population mean on the Gross Motor Quotient evaluation of these kids across three groups defined by income level. Higher values of this GMQ measure indicate improved levels of gross motor development, both in terms of locomotor and object control. See https://wwwn.cdc.gov/Nchs/Nnyfs/Y_GMX.htm for more details.

```
nyfs2a |>
    group_by(income_cat3) |>
    summarise(n = n(), mean(gmq), median(gmq))
```

```
# A tibble: 3 x 4
  income_cat3          n `mean(gmq)` `median(gmq)`
  <chr>            <int>       <dbl>         <dbl>
1 High (65K or more)  92        95.7            97
2 Low (below 25K)     98        97.0            97
3 Middle (25 - 64K)   90        95.4            94
```

Uh, oh. We should rearrange those income categories to match a natural order from low to high.

```
nyfs2a <- nyfs2a |>
  mutate(income_cat3 = fct_relevel(income_cat3,
         "Low (below 25K)", "Middle (25 - 64K)", "High (65K or more)"))
```

When working with three independent samples, I use graphs analogous to those we built for two independent samples.

```
ggplot(nyfs2a, aes(x = income_cat3, y = gmq, fill = income_cat3)) +
    geom_violin(aes(col = income_cat3), alpha = 0.5) +
    geom_boxplot(notch = TRUE, alpha = 0.75, width = 0.3) +
    stat_summary(fun = "mean", geom = "point",
                 shape = 23, size = 3, fill = "white") +
    coord_flip() +
    guides(fill = "none", col = "none") +
    labs(title = "GMQ Scores for 280 Children in NNYFS",
         y = "GMQ Score, in points", x = "Income Category")
```


GMQ Scores for 280 Children in NNYFS
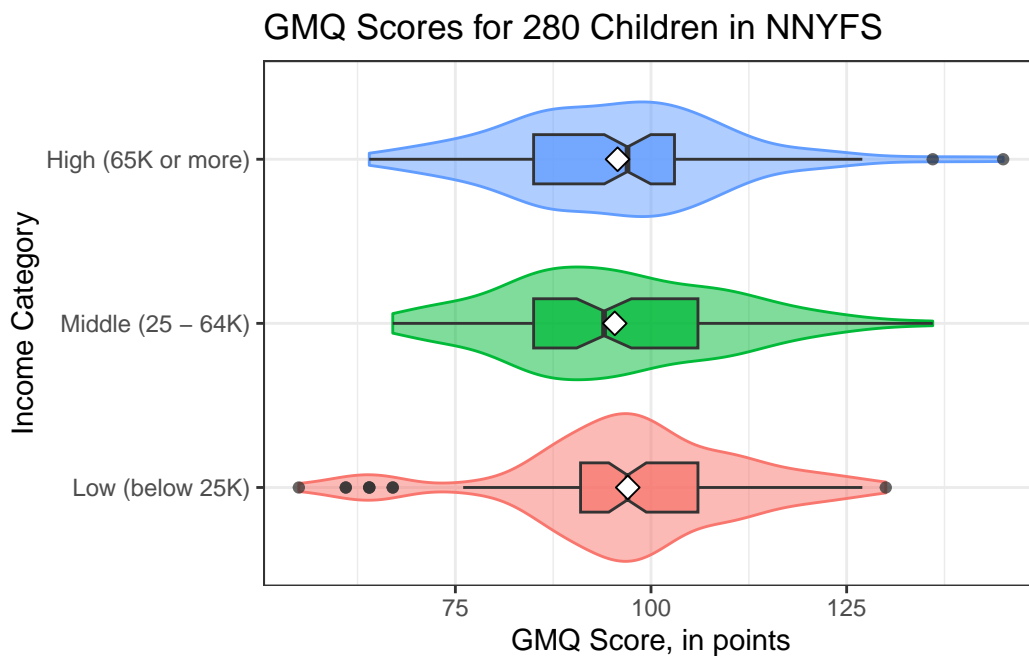
In addition to this comparison boxplot, we might consider faceted histograms.

```
ggplot(nyfs2a, aes(x = gmq, fill = income_cat3)) +
    geom_histogram(bins = 15, col = "white") +
    guides(fill = "none") +
    facet_wrap(~ income_cat3)
```
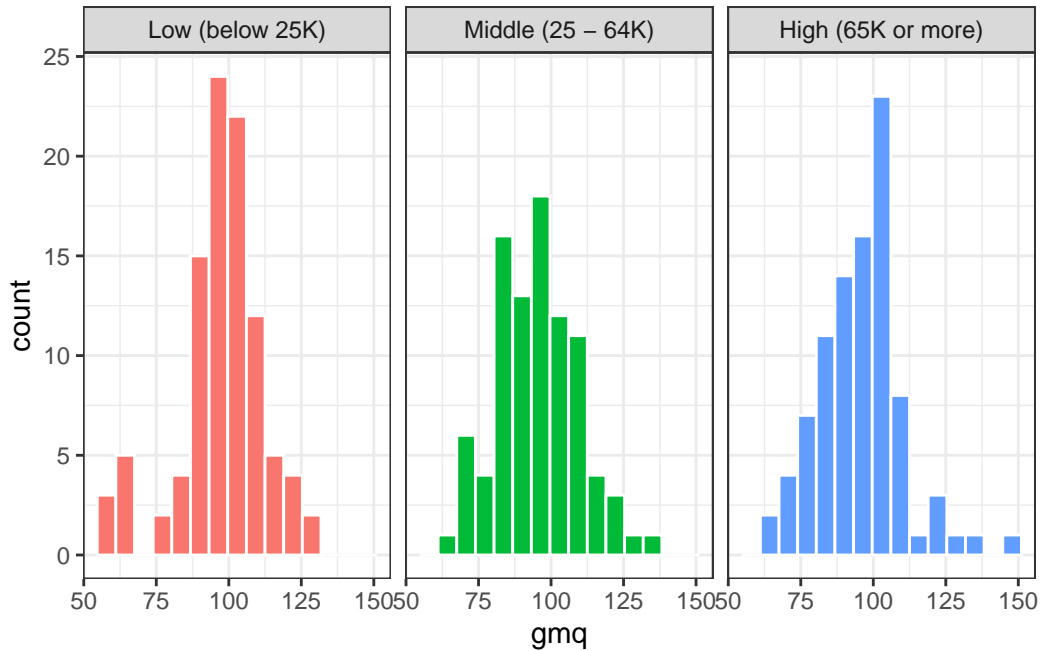
Or, if we want to ignore the (modest) sample size differences, we might consider density functions, perhaps through a ridgeline plot.

```
ggplot(nyfs2a, aes(x = gmq, y = income_cat3, fill = income_cat3)) +
    geom_density_ridges(scale = 0.9) +
    guides(fill = "none") +
    labs(title = "GMQ Score (in points) by Income Group",
        x = "GMQ Score", y = "") +
    theme_ridges()
```

```
Picking joint bandwidth of 4.7
```

GMQ Score (in points) by Income Group
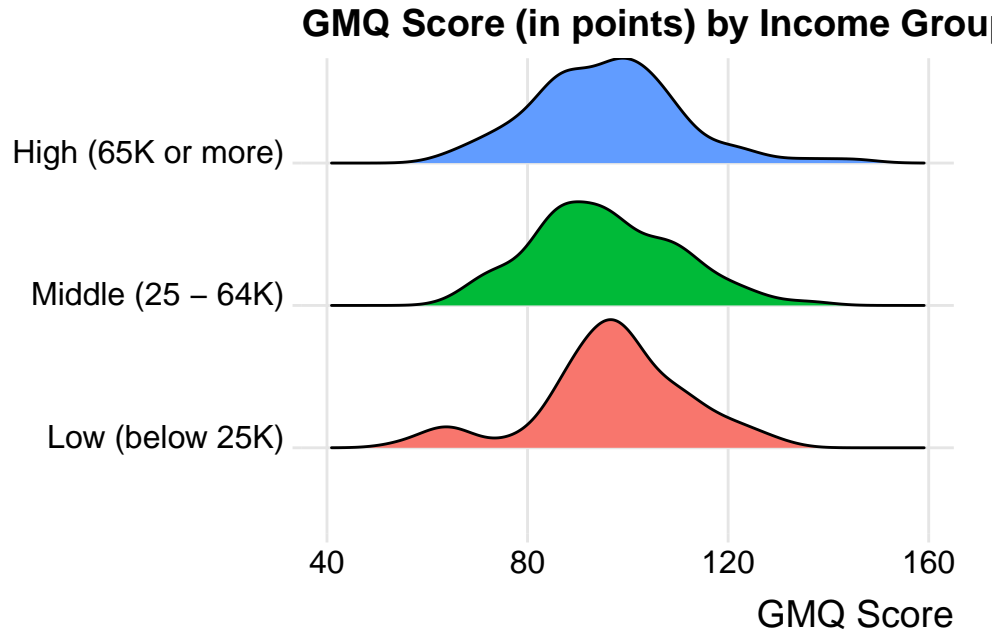
```
by(nyfs2a$gmq, nyfs2a$income_cat3, favstats)
```

```
nyfs2a$income_cat3: Low (below 25K)
 min Q1 median  Q3 max     mean       sd  n missing
  55 91      97 106 130 97.03061 14.79444 98       0
------------------------------------------------------------
nyfs2a$income_cat3: Middle (25 - 64K)
 min Q1 median  Q3 max     mean       sd  n missing
  67 85      94 106 136 95.36667 14.15123 90       0
------------------------------------------------------------
nyfs2a$income_cat3: High (65K or more)
 min Q1 median  Q3 max     mean       sd  n missing
  64 85      97 103 145 95.72826 14.49525 92       0
```

24.4 Alternative Procedures for Comparing More Than Two Means

Now, if we only had two independent samples, we'd be choosing between a pooled t test, a Welch t test, and a non-parametric procedure like the Wilcoxon-Mann-Whitney rank sum test, or even perhaps a bootstrap alternative.

In the case of more than two independent samples, we have methods analogous to the Welch test, and the rank sum test, and even the bootstrap, but we're going to be far more likely to select the **analysis of variance** (ANOVA) or an equivalent regression-based approach. These are the extensions of the pooled t test. Unless the sample outcome data are very clearly not Normally distributed, and no transformation is available which makes them appear approximately Normal in all of the groups we are comparing, we will stick with ANOVA.

24.4.1 Extending the Welch Test to > 2 Independent Samples

It is possible to extend the Welch two-sample t test (not assuming equal population variances) into an analogous one-factor analysis for comparing population means based on independent samples from more than two groups.

If we want to compare the population mean GMQ levels across those three income groups without assuming equal population variances, `oneway.test` is up to the task. The hypotheses being tested here are:

- H0: All three means are the same vs.
- HA: At least one of the population means is different than the others.

```
oneway.test(gmq ~ income_cat3, data = nyfs2a)
```

```
    One-way analysis of means (not assuming equal variances)

data:  gmq and income_cat3
F = 0.3416, num df = 2.00, denom df = 184.41, p-value = 0.7111
```

We get a p value, but this isn't much help, though, because we don't have any measure of effect size, nor do we have any confidence intervals. Like the analogous Welch t test, this approach allows us to forego the assumption of equal population variances in each of the three income groups, but it still requires us to assume that the populations are Normally distributed.

That said, most of the time when we have more than two levels of the factor of interest, we won't bother worrying about the equal population variance assumption, and will just use the one-factor ANOVA approach (with pooled variances) described below, to make the comparisons of interest.

24.4.2 Extending the Rank Sum Test to > 2 Independent Samples

It is also possible to extend the Wilcoxon-Mann-Whitney two-sample test into an analogous one-factor analysis called the **Kruskal-Wallis test** for comparing population measures of location based on independent samples from more than two groups.

If we want to compare the centers of the distributions of population GMQ score across our three income groups without assuming Normality, we can use `kruskal.test`.

The hypotheses being tested here are still as before, but for a measure of location other than the population mean

```
kruskal.test(gmq ~ income_cat3, data = nyfs2a)
```

```
    Kruskal-Wallis rank sum test

data:  gmq by income_cat3
Kruskal-Wallis chi-squared = 2.3202, df = 2, p-value = 0.3135
```

Again, note that this isn't much help, though, because we don't have any measure of effect size, nor do we have any confidence intervals.

That said, most of the time when we have more than two levels of the factor of interest, we won't bother worrying about potential violations of the Normality assumption unless they are glaring, and will just use the usual one-factor ANOVA approach (with pooled variances) described below, to make the comparisons of interest.

24.4.3 Can we use the bootstrap to compare more than two means?

Sure. There are both ANOVA and ANCOVA analogues using the bootstrap, and in fact, there are power calculations based on the bootstrap, too. If you want to see some example code, look at https://sammancuso.com/2017/11/01/model-based-bootstrapped-anova-and-ancova/

24.5 The Analysis of Variance

Extending the two-sample t test (assuming equal population variances) into a comparison of more than two samples uses the **analysis of variance** or ANOVA.

This is an analysis of a continuous outcome variable on the basis of a single categorical factor, in fact, it's often called one-factor ANOVA or one-way ANOVA to indicate that the outcome is being split up into the groups defined by a single factor.

The null hypothesis is that the population means are all the same, and the alternative is that this is not the case. When there are just two groups, then this boils down to an F test that is equivalent to the Pooled t test.

24.5.1 The `oneway.test` approach

R will produce some elements of a one-factor ANOVA using the `oneway.test` command:

```
oneway.test(gmq ~ income_cat3, data = nyfs2a, var.equal=TRUE)
```

```
    One-way analysis of means

data:  gmq and income_cat3
F = 0.34687, num df = 2, denom df = 277, p-value = 0.7072
```

This isn't the full analysis, though, which would require a more complete ANOVA table. There are two equivalent approaches to obtaining the full ANOVA table when comparing a series of 2 or more population means based on independent samples.

24.5.2 Using the `aov` approach and the `summary` function

Here's one possible ANOVA table, which doesn't require directly fitting a linear model.

```
summary(aov(gmq ~ income_cat3, data = nyfs2a))
```

```
             Df Sum Sq Mean Sq F value Pr(>F)
income_cat3   2    146   72.85   0.347  0.707
Residuals   277  58174  210.01
```

24.5.3 Using the `anova` function after fitting a linear model

An equivalent way to get identical results in a slightly different format runs the linear model behind the ANOVA approach directly.

```
anova(lm(gmq ~ income_cat3, data = nyfs2a))
```

```
Analysis of Variance Table

Response: gmq
            Df Sum Sq Mean Sq F value Pr(>F)
income_cat3   2    146  72.848  0.3469 0.7072
Residuals   277  58174 210.014
```

24.6 Interpreting the ANOVA Table

24.6.1 What are we Testing?

The null hypothesis for the ANOVA table is that the population means of the outcome across the various levels of the factor of interest are all the same, against a two-sided alternative hypothesis that the level-specific population means are not all the same.

Specifically, if we have a grouping factor with k levels, then we are testing:

- H0: All k population means are the same.
- HA: At least one of the population means is different from the others.

24.6.2 Elements of the ANOVA Table

The ANOVA table breaks down the variation in the outcome explained by the k levels of the factor of interest, and the variation in the outcome which remains (the Residual, or Error).

Specifically, the elements of the ANOVA table are:

1. the degrees of freedom (labeled Df) for the factor of interest and for the Residuals
2. the sums of squares (labeled Sum Sq) for the factor of interest and for the Residuals
3. the mean square (labeled Mean Sq) for the factor of interest and for the Residuals
4. the ANOVA F test statistic (labeled F value), which is used to generate
5. the p value for the comparison assessed by the ANOVA model, labeled Pr(>F)

24.6.3 The Degrees of Freedom

```
anova(lm(gmq ~ income_cat3, data = nyfs2a))
```

```
Analysis of Variance Table

Response: gmq
             Df Sum Sq Mean Sq F value Pr(>F)
income_cat3   2    146  72.848  0.3469 0.7072
Residuals   277  58174 210.014
```

- The **degrees of freedom** attributable to the factor of interest (here, Income category) is the number of levels of the factor minus 1. Here, we have three Income categories (levels), so df(income_cat3) = 2.
- The total degrees of freedom are the number of observations (across all levels of the factor) minus 1. We have 280 GMQ scores in the `nyfs2a` data, so the df(Total) must be 279, although the Total row isn't shown by R in its output.
- The Residual degrees of freedom are the Total df - Factor df. So, here, that's 279 - 2 = 277.

24.6.4 The Sums of Squares

```
anova(lm(gmq ~ income_cat3, data = nyfs2a))
```

```
Analysis of Variance Table

Response: gmq
             Df Sum Sq Mean Sq F value Pr(>F)
income_cat3   2    146  72.848  0.3469 0.7072
Residuals   277  58174 210.014
```

- The sum of squares (often abbreviated SS or Sum Sq) represents variation explained.
- The factor SS is the sum across all levels of the factor of the sample size for the level multiplied by the squared difference between the level mean and the overall mean across all levels. Here, SS(`income_cat3`) = 146
- The total SS is the sum across all observations of the square of the difference between the individual values and the overall mean. Here, that is $146 + 58174 = 58320$
- Residual SS = Total SS - Factor SS.
- Also of interest is a calculation called η^2, ("eta-squared"), which is equivalent to R^2 in a linear model.

  - SS(Factor) / SS(Total) = the proportion of variation in our outcome (here, GMQ) explained by the variation between groups (here, income groups)
  - In our case, $\eta^2 = 146 / (146 + 58174) = 146 / 58320 = 0.0025$
  - So, Income Category alone accounts for about 0.25% of the variation in GMQ levels observed in these data.

24.6.5 The Mean Square

```
anova(lm(gmq ~ income_cat3, data = nyfs2a))
```

```
Analysis of Variance Table

Response: gmq
            Df Sum Sq Mean Sq F value Pr(>F)
income_cat3   2    146  72.848  0.3469 0.7072
Residuals   277  58174 210.014
```

- The Mean Square is the Sum of Squares divided by the degrees of freedom, so MS(Factor) = SS(Factor)/df(Factor).
- In our case, MS(income_cat3) = SS(income_cat3)/df(income_cat3) = 146 / 2 = 72.848 (notice that R maintains more decimal places than it shows for these calculations) and
- MS(Residuals) = SS(Residuals) / df(Residuals) = 58174 / 277 = 210.014.

  - MS(Residuals) or MS(Error) is an estimate of the residual variance which corresponds to σ^2 in the underlying linear model for the outcome of interest, here GMQ.

24.6.6 The F Test Statistic and *p* Value

```
anova(lm(gmq ~ income_cat3, data = nyfs2a))
```

```
Analysis of Variance Table

Response: gmq
            Df Sum Sq Mean Sq F value Pr(>F)
income_cat3   2    146  72.848  0.3469 0.7072
Residuals   277  58174 210.014
```

- The ANOVA F test is obtained by calculating MS(Factor) / MS(Residuals). So in our case, F = 72.848 / 210.014 = 0.3469
- The F test statistic is then compared to a specific F distribution to obtain a *p* value, which is shown here to be 0.7072
- Specifically, the observed F test statistic is compared to an F distribution with numerator df = Factor df, and denominator df = Residual df to obtain the *p* value.

  - Here, we have SS(Factor) = 146 (approximately), and df(Factor) = 2, leaving MS(Factor) = 72.848

- We have SS(Residual) = 58174, and df(Residual) = 277, leaving MS(Residual) = 210.014
- MS(Factor) / MS(Residual) = F value = 0.3469, which, when compared to an F distribution with 2 and 277 degrees of freedom, yields a p value of 0.7072

24.7 The Residual Standard Error

The residual standard error is simply the square root of the variance estimate MS(Residual). Here, MS(Residual) = 210.014, so the Residual standard error = 14.49 points.

24.8 The Proportion of Variance Explained by the Factor

We will often summarize the proportion of the variation explained by the factor. The summary statistic is called eta-squared (η^2), and is equivalent to the R^2 value we have seen previously in linear regression models.

Again, η^2 = SS(Factor) / SS(Total)

Here, we have - SS(`income_cat3`) = 146 and SS(Residuals) = 58174, so SS(Total) = 58320 - Thus, η^2 = SS(Factor)/SS(Total) = 146/58320 = 0.0025

The income category accounts for 0.25% of the variation in GMQ levels: only a tiny fraction.

24.9 The Regression Approach to Compare Population Means based on Independent Samples

This approach is equivalent to the ANOVA approach, and thus also (when there are just two samples to compare) to the pooled-variance t test. We run a linear regression model to predict the outcome (here, `GMQ`) on the basis of the categorical factor with three levels (here, `income_cat3`)

```
summary(lm(gmq ~ income_cat3, data=nyfs2a))
```

```
Call:
lm(formula = gmq ~ income_cat3, data = nyfs2a)

Residuals:
    Min      1Q  Median      3Q     Max
```

```
-42.031  -9.031  -0.031   8.969  49.272
```

```
Coefficients:
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                     97.031      1.464  66.282   <2e-16 ***
income_cat3Middle (25 - 64K)    -1.664      2.116  -0.786    0.432
income_cat3High (65K or more)   -1.302      2.104  -0.619    0.536
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 14.49 on 277 degrees of freedom
Multiple R-squared:  0.002498,  Adjusted R-squared:  -0.004704
F-statistic: 0.3469 on 2 and 277 DF,  p-value: 0.7072
```

24.9.1 Interpreting the Regression Output

This output tells us many things, but for now, we'll focus just on the coefficients output, which tells us that:

- the point estimate for the population mean GMQ score across "Low" income subjects is 97.03
- the point estimate (sample mean difference) for the difference in population mean GMQ level between the "Middle" and "Low" income subjects is -1.66 (in words, the Middle income kids have lower GMQ scores than the Low income kids by 1.66 points on average.)
- the point estimate (sample mean difference) for the difference in population mean GMQ level between the "High" and "Low" income subjects is -1.30 (in words, the High income kids have lower GMQ scores than the Low income kids by 1.30 points on average.)

Of course, we knew all of this already from a summary of the sample means.

```
nyfs2a |>
    group_by(income_cat3) |>
    summarise(n = n(), mean(gmq))
```

```
# A tibble: 3 x 3
  income_cat3            n `mean(gmq)`
  <fct>             <int>       <dbl>
1 Low (below 25K)      98        97.0
2 Middle (25 - 64K)    90        95.4
3 High (65K or more)   92        95.7
```

The model for predicting GMQ is based on two binary (1/0) indicator variables, specifically, we have:

- Estimated GMQ = 97.03 - 1.66 x [1 if Middle income or 0 if not] - 1.30 x [1 if High income or 0 if not]

The coefficients section also provides a standard error and t statistic and two-sided p value for each coefficient.

24.9.2 The Full ANOVA Table

To see the full ANOVA table corresponding to any linear regression model, we run...

```
anova(lm(gmq ~ income_cat3, data=nyfs2a))
```

```
Analysis of Variance Table

Response: gmq
             Df Sum Sq Mean Sq F value Pr(>F)
income_cat3   2    146  72.848  0.3469 0.7072
Residuals   277  58174 210.014
```

24.9.3 ANOVA Assumptions

The assumptions behind analysis of variance are the same as those behind a linear model. Of specific interest are:

- The samples obtained from each group are independent.
- Ideally, the samples from each group are a random sample from the population described by that group.
- In the population, the variance of the outcome in each group is equal. (This is less of an issue if our study involves a balanced design.)
- In the population, we have Normal distributions of the outcome in each group.

Happily, the F test is fairly robust to violations of the Normality assumption.

24.10 Equivalent approach to get ANOVA Results

```
summary(aov(gmq ~ income_cat3, data = nyfs2a))
```

```
             Df Sum Sq Mean Sq F value Pr(>F)
income_cat3   2    146   72.85   0.347  0.707
Residuals   277  58174  210.01
```

So which of the pairs of means are driving the differences we see?

24.11 The Problem of Multiple Comparisons

1. Suppose we compare High to Low, using a test with $\alpha = 0.05$
2. Then we compare Middle to Low on the same outcome, also using $\alpha = 0.05$
3. Then we compare High to Middle, also with $\alpha = 0.05$

What is our overall α level across these three comparisons?

- It could be as bad as $0.05 + 0.05 + 0.05$, or 0.15.
- Rather than our nominal 95% confidence, we have something as low as 85% confidence across this set of simultaneous comparisons.

24.11.1 The Bonferroni solution

1. Suppose we compare High to Low, using a test with $\alpha = 0.05/3$
2. Then we compare Middle to Low on the same outcome, also using $\alpha = 0.05/3$
3. Then we compare High to Middle, also with $\alpha = 0.05/3$

Then across these three comparisons, our overall α can be (at worst)

- $0.05/3 + 0.05/3 + 0.05/3 = 0.05$
- So by changing our nominal confidence level from 95% to 98.333% in each comparison, we wind up with at least 95% confidence across this set of simultaneous comparisons.
- This is a conservative (worst case) approach.

Goal: Simultaneous comparisons of White vs AA, AA vs Other and White vs Other

```
pairwise.t.test(nyfs2a$gmq, nyfs2a$income_cat3, p.adjust="bonferroni")
```

```
    Pairwise comparisons using t tests with pooled SD

data:  nyfs2a$gmq and nyfs2a$income_cat3

                  Low (below 25K) Middle (25 - 64K)
Middle (25 - 64K)  1                   -
High (65K or more) 1                   1

P value adjustment method: bonferroni
```

These p values are very large.

24.11.2 Pairwise Comparisons using Tukey's HSD Method

Goal: Simultaneous (less conservative) confidence intervals and p values for our three pairwise comparisons (High vs. Low, High vs. Middle, Middle vs. Low)

```
TukeyHSD(aov(gmq ~ income_cat3, data = nyfs2a))
```

```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = gmq ~ income_cat3, data = nyfs2a)

$income_cat3
                                    diff       lwr      upr      p adj
Middle (25 - 64K)-Low (below 25K)   -1.6639456 -6.649518 3.321627 0.7116745
High (65K or more)-Low (below 25K)  -1.3023514 -6.259595 3.654892 0.8098084
High (65K or more)-Middle (25 - 64K) 0.3615942 -4.701208 5.424396 0.9845073
```

24.11.3 Plotting the Tukey HSD results

```
plot(TukeyHSD(aov(gmq ~ income_cat3, data = nyfs2a)))
```

95% family–wise confidence level

Differences in mean levels of income_cat3
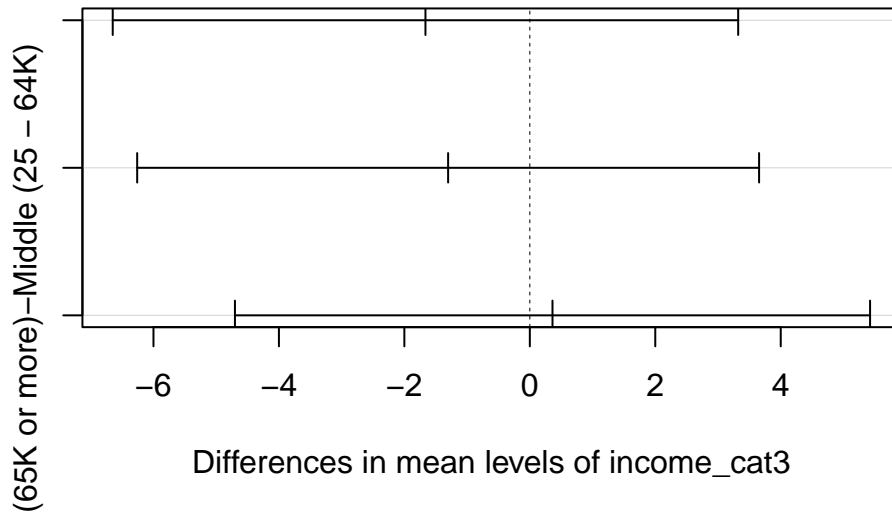
Note that the default positioning of the y axis in the plot of Tukey HSD results can be problematic. If we have longer names, in particular, for the levels of our factor, R will leave out some of the labels. We can alleviate that problem either by using the `fct_recode` function in the `forcats` package to rename the factor levels, or we can use the following code to reconfigure the margins of the plot.

```r
mar.default <- c(5,6,4,2) + 0.1 # save default plotting margins

par(mar = mar.default + c(0, 12, 0, 0))
plot(TukeyHSD(aov(gmq ~ income_cat3, data = nyfs2a)), las = 2)
```

95% family–wise confidence l

Middle (25 – 64K)–Low (below 25K)

High (65K or more)–Low (below 25K)

High (65K or more)–Middle (25 – 64K)

−6 −4 −2 0 2 4
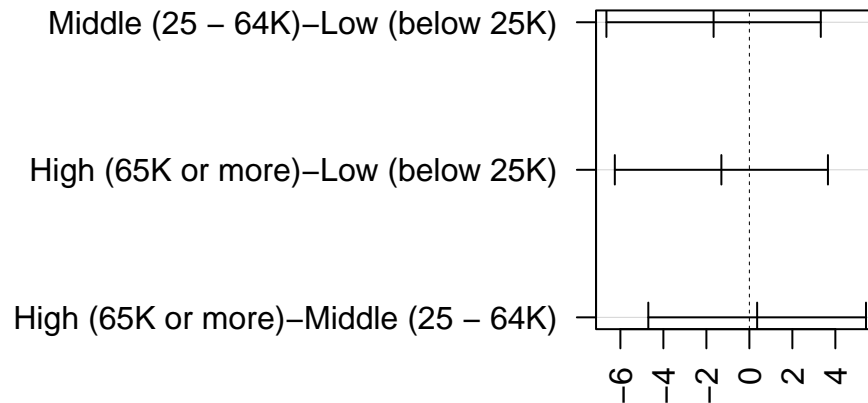
Differences in mean levels of income

```
par(mar = mar.default) # return to normal plotting margins
```

24.12 What if we consider another outcome, BMI?

We'll look at the full data set in `nyfs2` now, so we can look at BMI as a function of income.

```
nyfs2$income_cat3 <-
    fct_relevel(nyfs2$income_cat3,
                "Low (below 25K)", "Middle (25 - 64K)", "High (65K or more)")

ggplot(nyfs2, aes(x = bmi, y = income_cat3, fill = income_cat3)) +
    geom_density_ridges(scale = 0.9) +
    guides(fill = "none") +
    labs(title = "Body-Mass Index by Income Group",
         x = "Body-Mass Index", y = "") +
    theme_ridges()
```

Body–Mass Index by Income Group
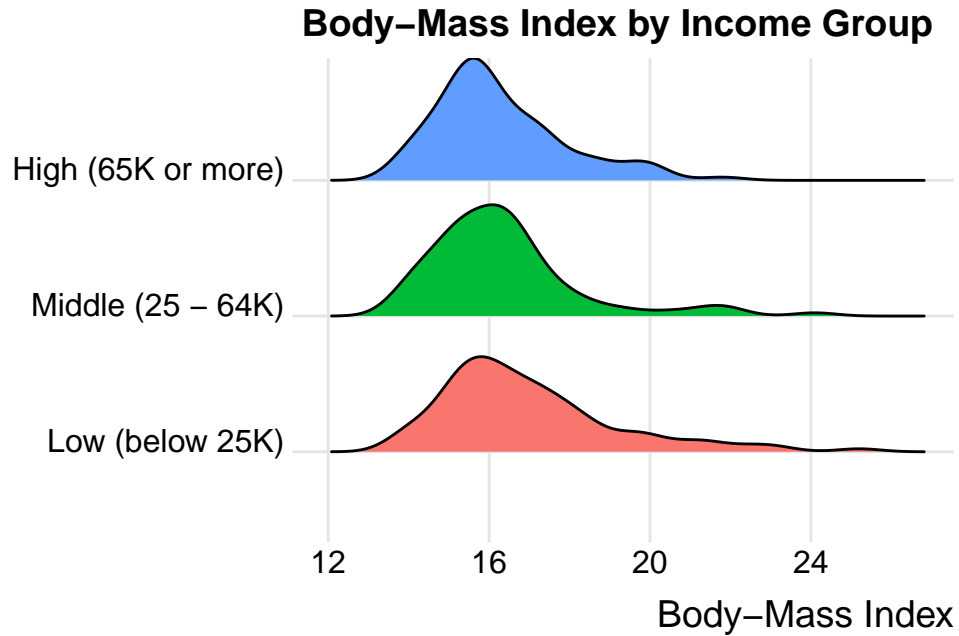
```
ggplot(nyfs2, aes(x = income_cat3, y = bmi, fill = income_cat3)) +
  geom_violin(aes(col = income_cat3), alpha = 0.5) +
  geom_boxplot(width = 0.3, notch = TRUE, alpha = 0.75) +
  stat_summary(fun = "mean", geom = "point",
               shape = 23, size = 3, fill = "white") +
  coord_flip() +
  guides(fill = "none", col = "none") +
  labs(title = "BMI for 280 Children in NNYFS",
       y = "Body-Mass Index", x = "Income Category")
```

BMI for 280 Children in NNYFS
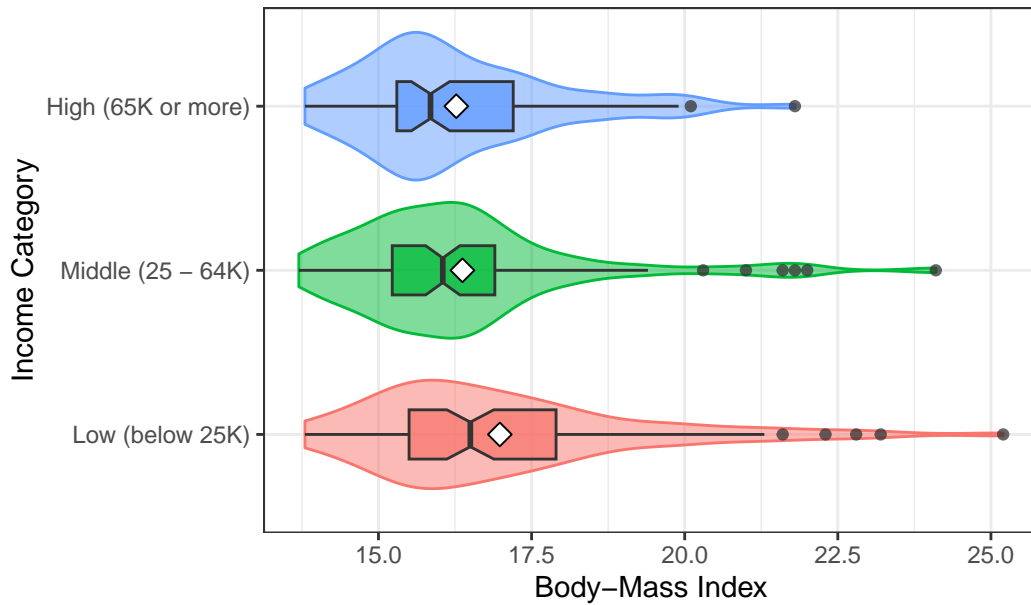
Here are the descriptive numerical summaries:

```
mosaic::favstats(bmi ~ income_cat3, data = nyfs2)
```

```
        income_cat3  min     Q1 median   Q3  max     mean       sd  n missing
1    Low (below 25K) 13.8 15.500  16.50 17.9 25.2 16.98163 2.194574 98       0
2  Middle (25 - 64K) 13.7 15.225  16.05 16.9 24.1 16.37111 1.898920 90       0
3 High (65K or more) 13.8 15.300  15.85 17.2 21.8 16.27065 1.614395 92       0
```

Here is the ANOVA table.

```
anova(lm(bmi ~ income_cat3, data = nyfs2))
```

```
Analysis of Variance Table

Response: bmi
             Df  Sum Sq Mean Sq F value  Pr(>F)
income_cat3   2   28.32 14.1583  3.8252 0.02298 *
Residuals   277 1025.26  3.7013
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Let's consider the Tukey HSD results. First, we'll create a factor with shorter labels.

```
nyfs2$inc.new <-
    fct_recode(nyfs2$income_cat3,
               "Low" = "Low (below 25K)", "Middle" = "Middle (25 - 64K)",
               "High" = "High (65K or more)")

plot(TukeyHSD(aov(bmi ~ inc.new, data = nyfs2),
              conf.level = 0.90))
```


It appears that there is a detectable difference between the `bmi` means of the "Low" group and both the "High" and "Middle" group at the 90% confidence level, but no detectable difference between "Middle" and "High." Details of those confidence intervals for those pairwise comparisons follow.

```
TukeyHSD(aov(bmi ~ inc.new, data = nyfs2),
              conf.level = 0.90)
```

```
  Tukey multiple comparisons of means
    90% family-wise confidence level

Fit: aov(formula = bmi ~ inc.new, data = nyfs2)
```

```
$inc.new
                  diff        lwr         upr      p adj
Middle-Low  -0.6105215 -1.1893722 -0.03167084 0.0775491
High-Low    -0.7109805 -1.2865420 -0.13541892 0.0306639
High-Middle -0.1004589 -0.6882764  0.48735849 0.9339289
```

25 Estimating a Population Proportion

We've focused on creating statistical inferences about a population mean, or difference between means, where we care about a quantitative outcome. Now, we'll tackle **categorical** outcomes, by estimating a confidence interval around a population proportion.

25.1 Setup: Packages Used Here

```r
knitr::opts_chunk$set(comment = NA)

source("data/Love-boost.R")
library(janitor)
library(kableExtra)
library(mosaic)
library(broom)
library(tidyverse)

theme_set(theme_bw())
```

25.2 A First Example: Serum Zinc in the "Normal" Range?

Recall that in the serum zinc study discussed in Chapter 19, we have 462 teenage male subjects, of whom 395 (or 85.5%) fell in the "normal range" of 66 to 110 micrograms per deciliter.

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

serzinc <- serzinc |>
  mutate(in_range = ifelse(zinc >= 66 & zinc <= 110, 1, 0))

serzinc |> tabyl(in_range) |>
  adorn_totals() |> adorn_pct_formatting()
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	0.855	0.016	52.133	0	0.828	0.882

```
 in_range    n percent
       0   67   14.5%
       1  395   85.5%
   Total 462  100.0%
```

Previously, we estimated a confidence interval for the *mean* of the population zinc levels. Now, we want to estimate a confidence interval for the *proportion* of the population whose serum zinc levels are in the range of 66 to 110. We want to build both a point estimate for the population proportion, and a confidence interval for the population proportion.

Now, let's identify a 95% confidence interval for the proportion of the population whose zinc levels are within the "normal" range. We have seen that 395 / 462 subjects (or a proportion of 0.855) fall in the "normal range" in our sample. For now, that will also be our *point estimate* of the proportion in the "normal range" across the entire population of teenagers like those in our sample.

```
serzinc <- serzinc |>
    mutate(in_range = ifelse(zinc > 65 & zinc < 111, 1, 0))

serzinc |> tabyl(in_range)
```

```
 in_range   n    percent
       0   67 0.1450216
       1  395 0.8549784
```

Once we've created this 0-1 variable, there are several available approaches for wrapping a confidence interval around this proportion.

25.2.1 Using an Intercept-Only Regression Again?

We might consider taking the same approach as we did with the population mean earlier:

```
model_zincprop <- lm(in_range ~ 1, data = serzinc)

tidy(model_zincprop, conf.int = TRUE, conf = 0.90) |>
    kbl(digits = 3) |> kable_minimal()
```

While there are more powerful approaches to estimate a confidence interval around this proportion, this simple approach turns out not to be too bad, so long as the sample proportion isn't very close to either 0 or 1.

25.2.2 A 100(1-α)% Confidence Interval for a Population Proportion

Suppose we want to estimate a confidence interval for an unknown population proportion, π, on the basis of a random sample of n observations from that population which yields a sample proportion of p. Note that this p is the sample proportion – it's not a p value.

- In our serum zinc example, we have n = 462 observations, with a sample proportion ("in range") of p = 0.855.

A 100(1-α)% confidence interval for the population proportion π can be created by using the standard normal distribution, the sample proportion, p, and the standard error of a sample proportion, which is defined as the square root of p multiplied by (1-p) divided by the sample size, n.

- So the standard error is estimated in our serum zinc example as:

$$\sqrt{\frac{p(1-p)}{n}} = \sqrt{\frac{0.855(1 - 0.855)}{462}} = \sqrt{0.000268} = 0.016$$

And thus, our confidence interval is $p \pm Z_{\alpha/2}\sqrt{\frac{p(1-p)}{n}}$

where $Z_{\alpha/2}$ = the value from a standard Normal distribution cutting off the top $\alpha/2$ of the distribution, obtained in R by substituting the desired $\alpha/2$ value into the following command: `qnorm(alpha/2, lower.tail=FALSE)`.

Note: This interval is reasonably accurate so long as np and n(1-p) are each at least 5.

- For the serum zinc data, we have np = (462)(0.855) = 395 and n(1-p) = 462(1 - 0.855) = 67, so this should be ok.

- For $\alpha = 0.05$, we have $Z_{\alpha/2} = 1.96$, approximately.

```
qnorm(0.025, lower.tail = FALSE)
```

[1] 1.959964

- Thus, for the serum zinc estimate, this confidence interval would be:

$$p \pm Z_{\alpha/2}\sqrt{\frac{p(1-p)}{n}} = \frac{395}{462} \pm 1.96\sqrt{\frac{0.855(1-0.855)}{462}} = 0.855 \pm 0.032$$

or (0.823, 0.887).

25.3 Using `binom.test` from the `mosaic` package

I am aware of at least seven different procedures for estimating a confidence interval for a population proportion using R. All have minor weaknesses: none is importantly different from the others in many practical situations. Five of these methods are available using the `binom.test` function from the `mosaic` package in R.

The general format for using the `binom.test` function is as follows:

```
binom.test(x = 395,  # substitute in number of successes
           n = 462,  # substitute in number of trials
           conf.level = 0.95, # default confidence level
           p = 0.5,  # default null hypothesis proportion
           ci.method = "XXX") # see below for XXX options
```

where the appropriate `ci.method` is obtained from the table below.

Approach	`ci.method` to be used
Wald	"Wald"
Clopper-Pearson	"Clopper-Pearson" or "binom.test"
Score	"Score" or "prop.test"
Agresti-Coull	"agresti-coull"
Plus4	"plus4"

25.3.1 The Wald test approach

The Wald approach can be used to establish a very similar confidence interval to the one we calculated above, based on something called the Wald test.

Here, we specify the `x` and `n` values. `n` is the total number of observations, and `x` is the number where the event of interest (in this case, serum zinc levels in the normal range) occurs. So `x` = 395 and `n` = 462.

estimate	conf.low	conf.high	statistic	parameter
0.855	0.823	0.887	395	462

method	alternative	p.value
Exact binomial test (Wald CI)	two.sided	0

```
m_wald <- binom.test(x = 395, n = 462,
                  conf.level = 0.95,
                  ci.method = "Wald")
m_wald
```

```
    Exact binomial test (Wald CI)

data:  395 out of 462
number of successes = 395, number of trials = 462, p-value < 2.2e-16
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.8228698 0.8870869
sample estimates:
probability of success
             0.8549784
```

The Wald confidence interval is always symmetric around our point estimate, and can dip below 0 or above 1.

When I fit intervals using the approaches in `mosaic::binom_test()` I will usually tidy them.

```
tidy(m_wald) |>
  select(estimate, conf.low, conf.high, statistic, parameter) |>
  kbl(digits = 3) |> kable_classic(full_width = F)
```

The other elements of the tidied result are shown below.

```
tidy(m_wald) |>
  select(method, alternative, p.value) |>
  kbl(digits = 3) |> kable_classic(full_width = F)
```

estimate	conf.low	conf.high	statistic	parameter
0.855	0.82	0.886	395	462

25.3.2 The Clopper-Pearson approach

The `binom.test` command can be used to establish an "exact" confidence interval. This uses the method of Clopper and Pearson from 1934, and is exact in the sense that it guarantees, for instance, that the confidence level associated with the interval is at least as large as the nominal level of 95%, but not that the interval isn't wider than perhaps it needs to be.

```
m_clopper <- binom.test(x = 395, n = 462,
                        conf.level = 0.95,
                        ci.method = "Clopper-Pearson")

m_clopper
```

```
data:  395 out of 462
number of successes = 395, number of trials = 462, p-value < 2.2e-16
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.8195187 0.8858100
sample estimates:
probability of success
             0.8549784
```

Clopper-Pearson is used by `stats::binom.test()` in R as well. Again, it guarantees coverage at least as large as the nominal coverage rate, but may produce wider intervals than the other methods we'll see. The 95% confidence interval by this method is (0.820, 0.886), which is in the same general range as our previous estimates.

```
tidy(m_clopper) |>
  select(estimate, conf.low, conf.high, statistic, parameter) |>
  kbl(digits = 3) |> kable_classic(full_width = F)
```

estimate	conf.low	conf.high	statistic	parameter
0.855	0.82	0.884	395	462

25.3.3 The Score approach

```
m_score <- mosaic::binom.test(x = 395, n = 462,
                              conf.level = 0.95,
                              ci.method = "Score")

m_score
```

```
	Exact binomial test (Score CI without continuity correction)

data:  395 out of 462
number of successes = 395, number of trials = 462, p-value < 2.2e-16
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.8199415 0.8841607
sample estimates:
probability of success
             0.8549784
```

The Score approach is also used by `stats::prop.test()` and creates CIs by inverting p-values from score tests. It can be applied with a continuity correction (use `ci.method` = "prop.test") or without.

In this case, we see that the Score approach and the Clopper-Pearson approach give very similar results.

```
tidy(m_score) |>
  select(estimate, conf.low, conf.high, statistic, parameter) |>
  kbl(digits = 3) |> kable_classic_2(full_width = F)
```

As mentioned, the score test can also be run incorporating something called a *continuity correction*, since we are using a Normal approximation to the exact binomial distribution to establish our margin for error. R, by default, includes this continuity correction for the Score test when we use `prop.test` to collect it.

```
m_score_cor <- binom.test(x = 395, n = 462,
                          conf.level = 0.95,
```

estimate	conf.low	conf.high	statistic	parameter
0.855	0.819	0.885	395	462

```
                    ci.method = "prop.test")

m_score_cor
```

```
    Exact binomial test (Score CI with continuity correction)

data:  395 out of 462
number of successes = 395, number of trials = 462, p-value < 2.2e-16
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.8187706 0.8851359
sample estimates:
probability of success
            0.8549784
```

```
  tidy(m_score_cor) |>
    select(estimate, conf.low, conf.high, statistic, parameter) |>
    kbl(digits = 3) |> kable_paper(full_width = F)
```

25.3.4 The Agresti-Coull Approach

```
  m_agresti <- binom.test(x = 395, n = 462,
                          conf.level = 0.95,
                          ci.method = "agresti-coull")

m_agresti
```

```
    Exact binomial test (Agresti-Coull CI)

data:  395 out of 462
number of successes = 395, number of trials = 462, p-value < 2.2e-16
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
```

estimate	conf.low	conf.high	statistic	parameter
0.855	0.82	0.884	395	462

```
 0.8198094 0.8842928
sample estimates:
probability of success
          0.8549784
```

The Agresti-Coull approach is the Wald method after adding Z successes and Z failures to the data, where Z is the appropriate quantile for a standard Normal distribution (1.96 for a 95% CI).

```
tidy(m_agresti) |>
  select(estimate, conf.low, conf.high, statistic, parameter) |>
  kbl(digits = 3) |> kable_paper(full_width = F)
```

25.3.5 The "Plus 4" approach

This approach is just the Wald method after adding 2 successes and 2 failures (so 4 observations) to the data. It will be very similar to the Agresti-Coull method if we are working with a 95% confidence interval.

```
m_plus4 <- binom.test(x = 395, n = 462,
                      conf.level = 0.95,
                      ci.method = "plus4")

m_plus4
```

```
	Exact binomial test (Plus 4 CI)

data:  395 out of 462
number of successes = 395, number of trials = 462, p-value < 2.2e-16
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.8196844 0.8841783
sample estimates:
probability of success
          0.8549784
```

estimate	conf.low	conf.high	statistic	parameter
0.855	0.82	0.884	395	462

```
tidy(m_plus4) |>
  select(estimate, conf.low, conf.high, statistic, parameter) |>
  kbl(digits = 3) |> kable_paper(full_width = F)
```

25.3.6 SAIFS: single augmentation with an imaginary failure or success

SAIFS stands for "single augmentation with an imaginary failure or success" and the method I'll describe is one of several similar approaches. The next subsection describes the R code for calculating the relevant confidence interval.

An approach I like for the estimation of a confidence interval for a single population proportion/rate[1] is to estimate the lower bound of a confidence interval with an imaginary failure added to the observed data, and estimate the upper bound of a confidence interval with an imaginary success added to the data.

Suppose we have X successes in n trials, and we want to establish a confidence interval for the population proportion of successes.

Let $p_1 = (X + 0)/(n + 1), p_2 = (X + 1)/(n + 1), q_1 = 1 - p_1, q_2 = 1 - p_2$

- The lower bound of a $100(1-\alpha)\%$ confidence interval for the population proportion of successes using the SAIFS procedure is then $LB_{SAIFS}(x, n, \alpha) = p_1 - t_{\alpha/2,n-1}\sqrt{\frac{p_1 q_1}{n}}$

- The upper bound of that same $100(1-\alpha)\%$ confidence interval for the population proportion of successes using the SAIFS procedure is $UB_{SAIFS}(x, n, \alpha) = p_2 + t_{\alpha/2,n-1}\sqrt{\frac{p_2 q_2}{n}}$

Returning to the serum zinc example, we've got 395 "successes" (value in the normal range) out of 462 "trials" (values measured), so that X = 395 and n = 462

So we have $p_1 = \frac{X+0}{n+1} = \frac{395}{463} = 0.8531$, $p_2 = \frac{X+1}{n+1} = \frac{396}{463} = 0.8553$, and $q_1 = 1 - p_1 = 0.1469$ and $q_2 = 1 - p_2 = 0.1447$

We have $n = 462$ so if we want a 95% confidence interval ($\alpha = 0.05$), then we have $t_{\alpha/2,n-1} = t_{0.025,461} = 1.9651$, which I determined using R's `qt` function:

```
qt(0.025, df = 461, lower.tail=FALSE)
```

---

[1]See Borkowf CB (2006) Constructing binomial confidence intervals with near nominal coverage by adding a single imaginary failure or success. Statistics in Medicine. 25(21): 3679-3695. doi: 10.1002/sim.2469, or get the whole PDF of the paper at http://onlinelibrary.wiley.com/doi/10.1002/sim.2469/pdf

```
[1] 1.965123
```

- Thus, our lower bound for a 95% confidence interval is $p_1 - t_{\alpha/2,n-1}\sqrt{\frac{p_1 q_1}{n}}$, or $0.8531 -$
  $1.9651\sqrt{\frac{0.8531(0.1469)}{462}}$, which is 0.8531 - 0.0324 or 0.8207.
- Our upper bound is $p_2 + t_{\alpha/2,n-1}\sqrt{\frac{p_2 q_2}{n}}$, or $0.8553 - 1.9651\sqrt{\frac{0.8553(0.1447)}{462}}$, which is 0.8553
  + 0.0323, or 0.8876.

So the 95% SAIFS confidence interval estimate for the population proportion, π, of teenage males whose serum zinc levels fall within the "normal range" is (0.821, 0.888).

25.3.7 A Function in R to Calculate the SAIFS Confidence Interval

I built an R function, called `saifs.ci` and contained in the Markdown for this document as well as the `Love-boost.R` script on the web site, which takes as its arguments a value for X = the number of successes, n = the number of trials, and conf.level = the confidence level, and produces the sample proportion, the SAIFS lower bound and upper bound for the specified two-sided confidence interval for the population proportion, using the equations above.

Here, for instance, are 95%, 90% and 99% confidence intervals for the population proportion π that we have been studying in the serum zinc data.

```
saifs.ci(x = 395, n = 462)
```

```
Sample Proportion           0.025           0.975
          0.855           0.821           0.887
```

```
saifs.ci(x = 395, n = 462, conf=0.9)
```

```
Sample Proportion            0.05            0.95
          0.855           0.826           0.882
```

```
saifs.ci(x = 395, n = 462, conf=0.99, dig=5)
```

```
Sample Proportion           0.005           0.995
        0.85498         0.81054         0.89763
```

Note that in the final interval, I asked the machine to round to five digits rather than the default of three. On my desktop (and probably yours), doing so results in this output:

```
Sample Proportion                0.005                 0.995
            0.85498            0.81054               0.89763
```

I've got some setting wrong in my `bookdown` work so that this doesn't show up above when the function is called. Sorry!

25.3.8 The `saifs.ci` function in R

```
`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)
    res <- c(p.sample, lower.bound, upper.bound)
    names(res) <- c('Sample Proportion',lowq, 1-lowq)
    res
  }
```

25.4 A Second Example: Ebola Mortality Rates through 9 Months of the Epidemic

The World Health Organization's Ebola Response Team published an article[2] 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 materials from their Table 2.

---

[2]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

As of September 14, 2014, a total of 4,507 confirmed and probable cases of Ebola virus disease (EVD) had been reported from West Africa. In our example, we will look at a set of 1,737 cases, with definitive outcomes, reported in Guinea, Liberia, Nigeria and Sierra Leone.

Across these 1,737 cases, a total of 1,229 cases led to death. Based on these sample data, what can be said about the case fatality rate in the population of EVD cases with definitive outcomes for this epidemic?

25.4.1 Working through the Ebola Virus Disease Example

We have n = 1,737 subjects, of whom we observed death in 1,229, for a sample proportion of $p = \frac{1229}{1737} = 0.708$. The standard error of that sample proportion will be

$SE(p) = \sqrt{\frac{p(1-p)}{n}} = \sqrt{\frac{0.708(1-0.708)}{1737}} = 0.011$

And our 95% confidence interval (so that we'll use $\alpha = 0.05$) for the true population proportion, π, of EVD cases with definitive outcomes, who will die is $p \pm Z_{.025}\sqrt{\frac{p(1-p)}{n}}$, or $0.708 \pm 1.96(0.011) = 0.708 \pm 0.022$, or $(0.686, 0.730)$

Note that I simply recalled from our prior work that $Z_{0.025} = 1.96$, but we can verify this:

```
qnorm(0.025, lower.tail=FALSE)
```

```
[1] 1.959964
```

Since both np=(1737)(0.708)=1230 and n(1-p)=(1737)(1-0.708)=507 are substantially greater than 5, this should be a reasonably accurate confidence interval.

We have 95% confidence in an interval estimate for the true population proportion that falls between 0.686 and 0.730. Equivalently, we could say that we're 95% confident that the true case fatality rate expressed as a percentage rather than a proportion, is between 68.6% and 73.0%.

25.4.2 Using R to estimate the CI for our Ebola example

```
ebola_wald <- binom.test(x = 1229, n = 1737, conf.level = 0.95,
                          ci.method = "Wald") |>
  tidy() |> select(estimate, conf.low, conf.high)
ebola_clop <- binom.test(x = 1229, n = 1737, conf.level = 0.95,
                          ci.method = "Clopper-Pearson") |>
  tidy() |> select(estimate, conf.low, conf.high)
```

estimate	conf.low	conf.high	approach
0.707542	0.686150	0.728934	Wald
0.707542	0.685524	0.728856	Clopper-Pearson
0.707542	0.685710	0.728457	Score
0.707542	0.685705	0.728462	Agresti-Coull
0.707542	0.685687	0.728443	Plus4

```r
ebola_scor <- binom.test(x = 1229, n = 1737, conf.level = 0.95,
                         ci.method = "Score") |>
  tidy() |> select(estimate, conf.low, conf.high)
ebola_agco <- binom.test(x = 1229, n = 1737, conf.level = 0.95,
                         ci.method = "agresti-coull") |>
  tidy() |> select(estimate, conf.low, conf.high)
ebola_plus <- binom.test(x = 1229, n = 1737, conf.level = 0.95,
                         ci.method = "plus4") |>
  tidy() |> select(estimate, conf.low, conf.high)

ebola_res <- bind_rows(ebola_wald, ebola_clop, ebola_scor,
                       ebola_agco, ebola_plus) |>
  mutate(approach = c("Wald", "Clopper-Pearson", "Score",
                      "Agresti-Coull", "Plus4"))

ebola_res |> kbl(digits = 6) |> kable_classic(full_width = FALSE)
```

This is way more precision than we can really justify, but I just want you to see that the five results are all (slightly) different.

25.4.3 Plotting the Confidence Intervals for the Ebola Virus Disease Example

```r
ggplot(ebola_res, aes(x = approach, y = estimate)) +
    geom_point() +
    geom_errorbar(aes(ymin = conf.low, ymax = conf.high)) +
    coord_flip() +
    labs(title = "95% CIs for x = 1229, n = 1737")
```

95% CIs for x = 1229, n = 1737
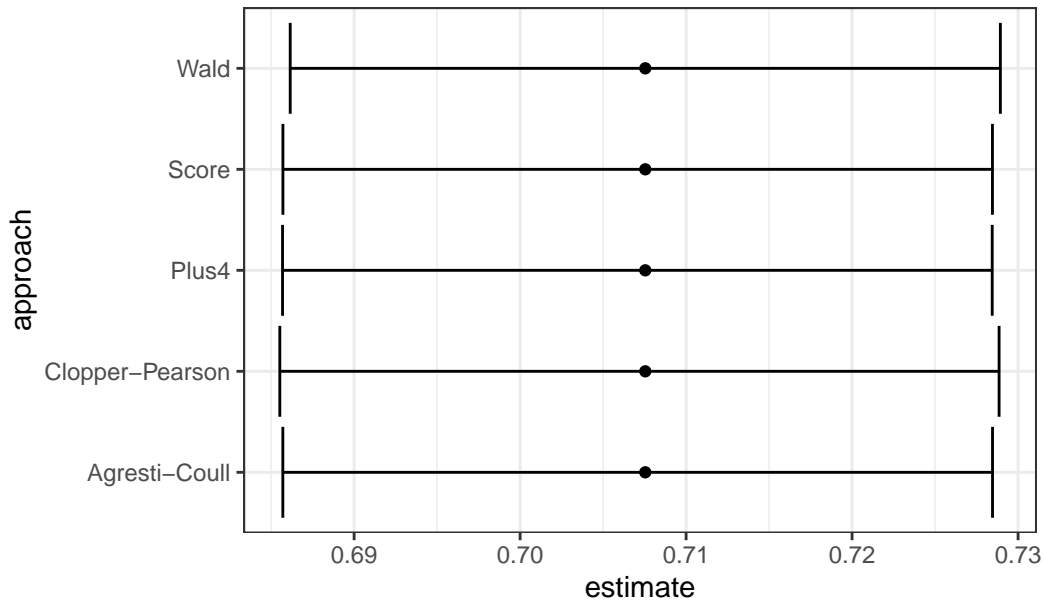
So in this case, it really doesn't matter which one you choose. With a smaller sample, we may not come to the same conclusion about the relative merits of these different approaches.

25.4.4 What about the `saifs.ci()` result?

```
saifs.ci(x = 1229, n = 1737, conf.level=0.95)
```

Sample Proportion	0.025	0.975
0.708	0.686	0.729

25.5 Can the Choice of Confidence Interval Method Matter?

Yes. This will especially be the case when we have a small sample size, and a probability of "success" that is close to either 0 or 1. For instance, suppose we run 10 trials, and obtain a single success, then use these data to estimate the true proportion of success, π.

The 90% confidence intervals under this circumstance are very different.

approach	estimate	conf.low	conf.high
Wald	0.1	-0.056	0.256
Clopper-Pearson	0.1	0.005	0.394
Score	0.1	0.023	0.348
Agresti-Coull	0.1	0.006	0.364
Plus4	0.1	0.034	0.395

```
tidy1 <- binom.test(x = 1, n = 10, conf.level = 0.90,
                    ci.method = "Wald") |> tidy()
tidy2 <- binom.test(x = 1, n = 10, conf.level = 0.90,
                    ci.method = "Clopper-Pearson") |> tidy()
tidy3 <- binom.test(x = 1, n = 10, conf.level = 0.90,
                    ci.method = "Score") |> tidy()
tidy4 <- binom.test(x = 1, n = 10, conf.level = 0.90,
                    ci.method = "agresti-coull") |> tidy()
tidy5 <- binom.test(x = 1, n = 10, conf.level = 0.90,
                    ci.method = "plus4") |> tidy()

res <- bind_rows(tidy1, tidy2, tidy3, tidy4, tidy5) |>
  mutate(approach = c("Wald", "Clopper-Pearson", "Score",
                      "Agresti-Coull", "Plus4")) |>
  select(approach, estimate, conf.low, conf.high)

res |> kbl(digits = 3) |> kable_paper(full_width = F)
```

Note that the Wald procedure doesn't force the confidence interval to appear in the $(0, 1)$ range.

```
ggplot(res, aes(x = approach, y = estimate)) +
  geom_point() +
  geom_errorbar(aes(ymin = conf.low, ymax = conf.high)) +
  geom_hline(aes(yintercept = 0), col = "red", lty = "dashed") +
  coord_flip() +
  labs(title = "90% CIs for x = 1, n = 10")
```

90% CIs for x = 1, n = 10
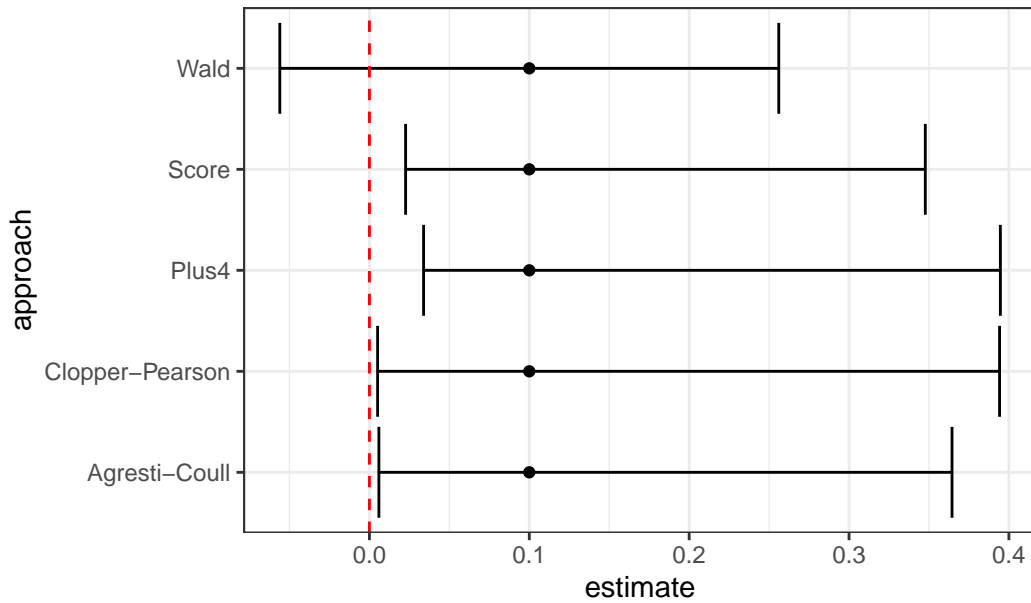
None of these three approaches is always better than any of the others. When we have a sample size below 100, or the sample proportion of success is either below 0.10 or above 0.90, caution is warranted, although in many cases, the various methods give similar responses.

26 Comparing Proportions with Two Independent Samples

Often, when analyzing data from an independent samples design, we are interested in comparing the proportions of subjects that achieve some outcome, across two levels of an exposure, or treatment. In this circumstance, we will first summarize the available data in terms of a 2x2 cross-tabulation, and then apply a series of inferential methods for 2x2 tables to obtain point and interval estimates of interest.

26.1 Setup: Packages Used Here

```r
knitr::opts_chunk$set(comment = NA)

source("data/Love-boost.R")
library(janitor)
library(mosaic)
library(tidyverse)

theme_set(theme_bw())
```

26.2 A First Example: Ibuprofen and Sepsis Trial

As we saw in Chapter 20, we are interested in comparing the percentage of patients in each arm of the trial (Ibuprofen vs. Placebo) that showed an improvement in their temperature (`temp_drop` > 0). Our primary interest is in comparing the percentage of Ibuprofen patients whose temperature dropped to the percentage of Placebo patients whose temperature dropped.

- We can summarize the data behind the two proportions we are comparing in a contingency table with two rows which identify the exposure or treatment of interest, and two columns to represent the outcomes of interest.

- In this case, we are comparing two groups of subjects based on treatment (`treat`): those who received Ibuprofen and those who received a placebo. The outcome of interest is whether the subject's temperature dropped (`temp_drop > 0`), or not.
- In the table, we place the frequency for each combination of a row and a column.
- The rows need to be mutually exclusive and collectively exhaustive: each patient must either receive Ibuprofen or Placebo. Similarly, the columns must meet the same standard: every patient's temperature either drops or does not drop.

Here's the contingency table.

```
sepsis <- read_csv("data/sepsis.csv",
                   show_col_types = FALSE) |>
    mutate(treat = factor(treat),
           race = factor(race))


sepsis <- sepsis |>
    mutate(dropped = ifelse(temp_drop > 0, "Drop", "No Drop"))

sepsis |> tabyl(treat, dropped) |>
    adorn_totals() |>
    adorn_percentages(denom = "row") |>
    adorn_pct_formatting(digits = 1) |>
    adorn_ns(position = "front")
```

```
    treat         Drop      No Drop
Ibuprofen 107 (71.3%)   43 (28.7%)
  Placebo  80 (53.3%)   70 (46.7%)
    Total 187 (62.3%)  113 (37.7%)
```

In our sample, we observe 71.3% of the 150 Ibuprofen subjects with a positive `temp_drop` as compared to 53.3% of the 150 Placebo subjects. We want to compare these two probabilities (represented using proportions as 0.713 vs. 0.533) to estimate the size of the difference between the proportions with a point estimate and 90% confidence interval.

But there are other comparisons we could make, too. The tricky part is that we have multiple ways to describe the relationship between treatment and outcome. We might compare outcome "risks" directly using the difference in probabilities, or the ratio of the two probabilities, or we might convert the risks to odds, and compare the ratio of those odds. In any case, we'll get different point estimates and confidence intervals, all of which will help us make conclusions about the evidence available in this trial speaking to differences between Ibuprofen and Placebo.

26.3 Relating a Treatment to an Outcome

The question of interest is whether the percentage of subjects whose temperature dropped is different (and probably larger) in the subjects who received Ibuprofen than in those who received the Placebo.

Treatment Arm	Dropped	Did Not Drop	Total	Proportion who dropped
Ibuprofen	107	43	150	0.713
Placebo	80	70	150	0.533

In other words, what is the relationship between the treatment and the outcome?

26.4 Definitions of Probability and Odds

- Proportion = Probability = Risk of the trait = number with trait / total
- Odds of having the trait = (number with the trait / number without the trait) to 1

If p is the proportion of subjects with a trait, then the **odds** of having the trait are $\frac{p}{1-p}$ to 1.

So, the probability of a good result (temperature drop) in this case is $\frac{107}{150} = 0.713$ in the Ibuprofen group. The **odds** of a good result are thus $\frac{0.713}{1-0.713} = 2.484$ to 1 in the Ibuprofen group.

Treatment Arm	Dropped	Did Not Drop	Total	Pr(dropped)	Odds(dropped)
Ibuprofen	107	43	150	0.713	2.484
Placebo	80	70	150	0.533	1.141

26.5 Defining the Relative Risk

Among the Ibuprofen subjects, the risk of a good outcome (drop in temperature) is 71.3% or, stated as a proportion, 0.713. Among the Placebo subjects, the risk of a good outcome is 53.3% or, stated as a proportion, 0.533.

Our "crude" estimate of the **relative risk** of a good outcome for Ibuprofen subjects as compared to Placebo subjects, is the ratio of these two risks, or $0.713/0.533 = 1.338$

- The fact that this relative risk is greater than 1 indicates that the probability of a good outcome is higher for Ibuprofen subjects than for Placebo subjects.
- A relative risk of 1 would indicate that the probability of a good outcome is the same for Ibuprofen subjects and for Placebo subjects.
- A relative risk less than 1 would indicate that the probability of a good outcome is lower for Ibuprofen subjects than for Placebo subjects.

26.6 Defining the Risk Difference

Our "crude" estimate of the **risk difference** of a good outcome for Ibuprofen subjects as compared to Placebo subjects, is 0.713 - 0.533 = 0.180 or 18.0 percentage points.

- The fact that this risk difference is greater than 0 indicates that the probability of a good outcome is higher for Ibuprofen subjects than for Placebo subjects.
- A risk difference of 0 would indicate that the probability of a good outcome is the same for Ibuprofen subjects and for Placebo subjects.
- A risk difference less than 0 would indicate that the probability of a good outcome is lower for Ibuprofen subjects than for Placebo subjects.

26.7 Defining the Odds Ratio, or the Cross-Product Ratio

Among the Ibuprofen subjects, the odds of a good outcome (temperature drop) are 2.484. Among the placebo subjects, the odds of a good outcome (temperature drop) are 1.141.

So our "crude" estimate of the **odds ratio** of a good outcome for Ibuprofen subjects as compared to placebo subjects, is 2.484 / 1.141 = 2.18

Another way to calculate this odds ratio is to calculate the **cross-product ratio**, which is equal to (a x d) / (b x c), for the 2 by 2 table with counts specified as shown:

A Generic Table	Good Outcome	Bad Outcome
Treatment Group 1	a	b
Treatment Group 2	c	d

So, for our table, we have a = 107, b = 43, c = 80, and d = 70, so the cross-product ratio is $\frac{107 x 70}{43 x 80} = \frac{7490}{3440} = 2.18$. As expected, this is the same as the "crude" odds ratio estimate.

- The fact that this odds ratio risk is greater than 1 indicates that the odds of a good outcome are higher for Ibuprofen subjects than for Placebo subjects.

- An odds ratio of 1 would indicate that the odds of a good outcome are the same for Ibuprofen subjects and for Placebo subjects.
- An odds ratio less than 1 would indicate that the odds of a good outcome are lower for Ibuprofen subjects than for Placebo subjects.

So, we have several different ways to compare the outcomes across the treatments. Are these differences and ratios large enough to rule out chance?

26.8 Comparing Rates in a 2x2 Table

What is the relationship between the treatment (Ibuprofen vs. Placebo) and the outcome (drop in temperature) in the following two-by-two table?

26.9 The `twobytwo` function in R

I built the `twobytwo` function in R (based on existing functions in the `Epi` library, which you need to have in your installed packages list in order for this to work) to do the work for this problem. All that is required is a single command, and a two-by-two table in standard epidemiological format (with the outcomes in the columns, and the treatments in the rows.)

Treatment Arm	Dropped	Did Not Drop
Ibuprofen	107	43
Placebo	80	70

The command just requires you to read off the cells of the table, followed by the labels for the two treatments, then the two outcomes, then a specification of the names of the rows (exposures) and columns (outcomes) from the table, and a specification of the confidence level you desire. We'll use 90% here.

The resulting output follows.

```
twobytwo(107, 43, 80, 70,
         "Ibuprofen", "Placebo", "Dropped", "No Drop",
         conf.level = 0.90)
```

```
2 by 2 table analysis:
------------------------------------------------------
Outcome   : Dropped
Comparing : Ibuprofen vs. Placebo
```

```
          Dropped No Drop   P(Dropped) 90% conf. interval
Ibuprofen     107      43       0.7133     0.6490   0.7701
Placebo        80      70       0.5333     0.4661   0.5993


                                   90% conf. interval
               Relative Risk: 1.3375     1.1492   1.5567
          Sample Odds Ratio: 2.1773     1.4583   3.2509
Conditional MLE Odds Ratio: 2.1716     1.4177   3.3437
     Probability difference: 0.1800     0.0881   0.2677


               Exact P-value: 0.0019
          Asymptotic P-value: 0.0014
--------------------------------------------------------
```

26.9.1 Standard Epidemiological Format

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 `treat`). The more interesting (sometimes also the more common) "treatment" is placed in the top row. That's Ibuprofen here.
- The columns of the table describe the "outcome" (which we'll take here to be whether the subject's temperature dropped.) Typically, the more common "outcome" is placed to the left.

26.9.2 Outcome Probabilities and Confidence Intervals Within the Treatment Groups

The `twobytwo` output starts with estimates of the probability (risk) of a "Dropped" outcome among subjects who fall into the two treatment groups (Ibuprofen or Placebo), along with 90% confidence intervals for each of these probabilities.

```
2 by 2 table analysis:
--------------------------------------------------------
Outcome   : Dropped
Comparing : Ibuprofen vs. Placebo

          Dropped No Drop   P(Dropped) 90% conf. interval
Ibuprofen     107      43       0.7133     0.6490   0.7701
Placebo        80      70       0.5333     0.4661   0.5993
```

The conditional probability of a temperature drop given that the subject is in the Ibuprofen group, is symbolized as Pr(Dropped | Ibuprofen) = 0.7133, and the 90% confidence interval around that proportion is (0.6490, 0.7701).

- Note that these two confidence intervals fail to overlap, and so we expect to see a fairly large difference in the estimated probability of a temperature drop when we compare Ibuprofen to Placebo.

26.9.3 Relative Risk, Odds Ratio and Risk Difference, with Confidence Intervals

These elements are followed by estimates of the relative risk, odds ratio, and risk difference, each with associated 90% confidence intervals.

```
                               90% conf. interval
            Relative Risk: 1.3375    1.1492   1.5567
        Sample Odds Ratio: 2.1773    1.4583   3.2509
Conditional MLE Odds Ratio: 2.1716    1.4177   3.3437
   Probability difference: 0.1800    0.0881   0.2677
```

- The **relative risk**, or the ratio of P(Temperature Drop | Ibuprofen) to P(Temperature Drop | Placebo), is shown first. Note that the 90% confidence interval is entirely greater than 1.
- The **odds ratio** is presented using two different definitions (the sample odds ratio is the cross-product ratio we mentioned earlier). Note that the 90% confidence interval using either approach is entirely greater than 1.
- The **probability** (or risk) **difference** [P(Temperature Drop | Ibuprofen) - P(Temperature Drop | Placebo)] is presented last. Note that the 90% confidence interval is entirely greater than 0.
- Note carefully that if there had been no difference between Ibuprofen and Placebo, the relative risk and odds ratios would be 1, but the probability difference would be zero.

26.10 Estimating a Rate More Accurately: Use (x + 2)/(n + 4) rather than x/n

Suppose you have some data involving n independent tries, with x successes. A natural estimate of the "success rate" in the data is x / n.

But, strangely enough, it turns out this isn't an entirely satisfying estimator. Alan Agresti provides substantial motivation for the $(x + 2)/(n + 4)$ estimate as an alternative[1]. This is sometimes called a *Bayesian augmentation*.

---

[1]This note comes largely from a May 15 2007 entry in Andrew Gelman's blog at http://andrewgelman.com/2007/05/15

- The big problem with x / n is that it estimates p = 0 or p = 1 when x = 0 or x = n.
- It's also tricky to compute confidence intervals at these extremes, since the usual standard error for a proportion, $\sqrt{np(1-p)}$, gives zero, which isn't quite right.
- (x + 2)/(n + 4) is much cleaner, especially when you build a confidence interval for the rate.
- The only place where (x + 2)/(n + 4) will go wrong is if n is small and the true probability is very close to 0 or 1.

For example, if n = 10, and p is 1 in a million, then x will almost certainly be zero, and an estimate of 1/12 is much worse than the simple 0/10. However, how big a deal is this? If p might be 1 in a million, you're not going to estimate it with a n = 10 experiment[2].

Applying this method to our Ibuprofen and Sepsis Trial data, we would simply add two to each frequency in the main four cells in our 2x2 table.

So instead of using

```
twobytwo(107, 43, 80, 70,
         "Ibuprofen", "Placebo", "Dropped", "No Drop",
         conf.level = 0.90)
```

the Bayesian augmentation would encourage us to look at

```
twobytwo(109, 45, 82, 72,
         "Ibuprofen", "Placebo", "Dropped", "No Drop",
         conf.level = 0.90)
```

```
2 by 2 table analysis:
------------------------------------------------------
Outcome   : Dropped
Comparing : Ibuprofen vs. Placebo

          Dropped No Drop    P(Dropped) 90% conf. interval
Ibuprofen     109      45        0.7078    0.6441   0.7643
Placebo        82      72        0.5325    0.4662   0.5977

                                   90% conf. interval
            Relative Risk: 1.3293    1.1434    1.5453
        Sample Odds Ratio: 2.1268    1.4337    3.1550
Conditional MLE Odds Ratio: 2.1215    1.3950    3.2421
   Probability difference: 0.1753    0.0845    0.2622
```

---

[2]Andrew Gelman's example is "I'm not going to try ten 100-foot golf putts, miss all of them, and then estimate my probability of success as 1/12."

```
          Exact P-value: 0.0022
      Asymptotic P-value: 0.0016
-------------------------------------------------------
```

As you can see, the odds ratio and relative risk estimates are (a little) closer to 1, and the probability difference is also a little closer to 0. The Bayesian augmentation provides a slightly more conservative set of estimates of the impact of Ibuprofen as compared to Placebo.

It is likely that the augmented version is a more accurate estimate here, but the two estimates will be comparable, generally, so long as either (a) the sample size in each exposure group is more than, say, 30 subjects, and/or (b) the sample probability of the outcome is between 10% and 90% in each exposure group.

26.11 A Second Example: Ebola Virus Disease Study, again

For instance, recall the Ebola Virus Disease study from the *New England Journal of Medicine* that we described in Section 25.4. 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.

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.

We can run these data through R, using the Bayesian augmentation (adding a death and a survival to the hospitalized and also to the not hospitalized groups.) We'll use a 95% confidence level this time.

```
twobytwo(741+2, 412+2, 488+2, 96+2,
         "Hospitalized", "Not Hospitalized", "Deceased", "Alive",
         conf.level = 0.95)
```

```
2 by 2 table analysis:
--------------------------------------------------------
Outcome   : Deceased
Comparing : Hospitalized vs. Not Hospitalized

                  Deceased Alive    P(Deceased) 95% conf. interval
Hospitalized           743   414         0.6422     0.6141   0.6693
Not Hospitalized       490    98         0.8333     0.8010   0.8613

                                    95% conf. interval
              Relative Risk:  0.7706     0.7285    0.8151
          Sample Odds Ratio:  0.3589     0.2801    0.4599
 Conditional MLE Odds Ratio:  0.3591     0.2772    0.4624
      Probability difference: -0.1912   -0.2307   -0.1490

              Exact P-value: 0.0000
          Asymptotic P-value: 0.0000
--------------------------------------------------------
```

I'll leave it as an exercise for you to interpret these results and draw some conclusions.

27 Power and Proportions

27.1 Setup: Packages Used Here

```r
source("data/Love-boost.R")

library(pwr)
```

We'll use the `Love-boost.R` script for the `twobytwo` function.

27.2 Tuberculosis Prevalence Among IV Drug Users

Consider a study to investigate factors affecting tuberculosis prevalence among intravenous drug users. The original data source is Graham NMH et al. (1992) Prevalence of Tuberculin Positivity and Skin Test Anergy in HIV-1-Seropositive and Seronegative Intravenous Drug Users. *JAMA*, 267, 369-373. Among 97 individuals who admit to sharing needles, 24 (24.7%) had a positive tuberculin skin test result; among 161 drug users who deny sharing needles, 28 (17.4%) had a positive test result. To start, we'll test the null hypothesis that the proportions of intravenous drug users who have a positive tuberculin skin test result are identical for those who share needles and those who do not.

```r
twobytwo(24, 73, 28, 133,
         "Sharing Needles", "Not Sharing",
         "TB test+", "TB test-")
```

```
2 by 2 table analysis:
-----------------------------------------------------
Outcome   : TB test+
Comparing : Sharing Needles vs. Not Sharing

                TB test+ TB test-    P(TB test+) 95% conf. interval
Sharing Needles       24       73         0.2474    0.1717    0.3427
Not Sharing           28      133         0.1739    0.1229    0.2404
```

```
                                  95% conf. interval
            Relative Risk: 1.4227    0.8772    2.3073
        Sample Odds Ratio: 1.5616    0.8439    2.8898
Conditional MLE Odds Ratio: 1.5588    0.8014    3.0191
   Probability difference: 0.0735   -0.0265    0.1807


            Exact P-value: 0.1996
       Asymptotic P-value: 0.1557
--------------------------------------------------------
```

What conclusions should we draw?

27.3 Designing a New TB Study

Now, suppose we wanted to design a new study with as many non-sharers as needle-sharers participating, and suppose that we wanted to detect any difference in the proportion of positive skin test results between the two groups that was identical to the data presented above or larger with at least 90% power, using a two-sided test and $\alpha = .05$. What sample size would be required to accomplish these aims?

27.4 Using `power.prop.test` for Balanced Designs

Our constraints are that we want to find the sample size for a two-sample comparison of proportions using a balanced design, we will use $\alpha = .05$, and power $= .90$, and that we estimate that the non-sharers will have a .174 proportion of positive tests, and we will try to detect a difference between this group and the needle sharers, who we estimate will have a proportion of .247, using a two-sided hypothesis test.

```
power.prop.test(p1 = .174, p2  = .247, sig.level = 0.05, power = 0.90)
```

```
    Two-sample comparison of proportions power calculation

             n = 653.2876
            p1 = 0.174
            p2 = 0.247
     sig.level = 0.05
         power = 0.9
```

```
    alternative = two.sided
```

```
NOTE: n is number in *each* group
```

So, we'd need at least 654 non-sharing subjects, and 654 more who share needles to accomplish the aims of the study.

27.5 How `power.prop.test` works

`power.prop.test` works much like the `power.t.test` we saw for means.

Again, we specify 4 of the following 5 elements of the comparison, and R calculates the fifth.

- The sample size (interpreted as the # in each group, so half the total sample size)
- The true probability in group 1
- The true probability in group 2
- The significance level (α)
- The power ($1 - \beta$)

The big weakness with the `power.prop.test` tool is that it doesn't allow you to work with unbalanced designs.

27.6 A Revised Scenario

Suppose we can get exactly 800 subjects in total (400 sharing and 400 non-sharing). How much power would we have to detect a difference in the proportion of positive skin test results between the two groups that was identical to the data presented above or larger, using a one-sided test, with $\alpha = .10$?

```
power.prop.test(n=400, p1=.174, p2=.247, sig.level = 0.10,
                alternative="one.sided")
```

```
    Two-sample comparison of proportions power calculation

             n = 400
            p1 = 0.174
            p2 = 0.247
     sig.level = 0.1
         power = 0.8954262
```

```
    alternative = one.sided
```

```
NOTE: n is number in *each* group
```

We would have just under 90% power to detect such an effect.

27.7 Using the `pwr` package for Unbalanced Designs

The `pwr.2p2n.test` function in the `pwr` package can help assess the power of a test to determine a particular effect size using an unbalanced design, where n_1 is not equal to n_2.

As before, we specify four of the following five elements of the comparison, and R calculates the fifth.

- `n1` = The sample size in group 1
- `n2` = The sample size in group 2
- `sig.level` = The significance level (α)
- `power` = The power ($1 - \beta$)
- `h` = the effect size h, which can be calculated separately in R based on the two proportions being compared: `p1` and `p2`.

27.7.1 Calculating the Effect Size `h`

To calculate the effect size for a given set of proportions, just use `ES.h(p1, p2)` which is available in the `pwr` package.

For instance, in our comparison, we have the following effect size.

```
ES.h(p1 = .174, p2 = .247)
```

```
[1] -0.1796783
```

27.8 Using `pwr.2p2n.test` in R

Suppose we can have 700 samples in group 1 (the not sharing group) but only half that many in group 2 (the group of users who share needles). How much power would we have to detect this same difference ($p_1 = .174$, $p_2 = .247$) with a 5% significance level in a two-sided test?

```
pwr.2p2n.test(h = ES.h(p1 = .174, p2 = .247),
              n1 = 700, n2 = 350,
              sig.level = 0.05)
```

```
     difference of proportion power calculation for binomial distribution (arcsine transforma

              h = 0.1796783
             n1 = 700
             n2 = 350
      sig.level = 0.05
          power = 0.7836768
    alternative = two.sided

NOTE: different sample sizes
```

Note that the headline for this output actually reads:

```
difference of proportion power calculation for binomial distribution
(arcsine transformation)
```

It appears we will have about 78% power under these circumstances.

27.8.1 Comparison to Balanced Design

How does this compare to the results with a balanced design using only 1000 drug users in total, so that we have 500 patients in each group?

```
pwr.2p2n.test(h = ES.h(p1 = .174, p2 = .247),
              n1 = 500, n2 = 500, sig.level = 0.05)
```

```
     difference of proportion power calculation for binomial distribution (arcsine transforma

              h = 0.1796783
             n1 = 500
             n2 = 500
      sig.level = 0.05
          power = 0.8108416
```

```
     alternative = two.sided

NOTE: different sample sizes
```

or we could instead have used...

```
  power.prop.test(p1 = .174, p2 = .247,
                  sig.level = 0.05, n = 500)
```

```
     Two-sample comparison of proportions power calculation

              n = 500
             p1 = 0.174
             p2 = 0.247
      sig.level = 0.05
          power = 0.8091808
    alternative = two.sided

NOTE: n is number in *each* group
```

Note that these two sample size estimation approaches are approximations, and use slightly different approaches, so it's not surprising that the answers are similar, but not completely identical.

28 Larger Contingency Tables

What will we do with tables describing data from more than two categories at a time, returning to the notion of independent (rather than paired or matched) samples? The chi-square tests we have already seen in our `twobytwo` table output will extend nicely to this scenario, especially the Pearson χ^2 (asymptotic) test.

28.1 Setup: Packages Used Here

```
knitr::opts_chunk$set(comment = NA)

source("data/Love-boost.R")
library(Epi)
library(janitor)
library(vcd)
library(tidyverse)

theme_set(theme_bw())
```

28.2 A 2x3 Table: Comparing Response to Active vs. Placebo

The table below, containing 2 rows and 3 columns of data (ignoring the marginal totals) specifies the number of patients who show *complete*, *partial*, or *no response* after treatment with either **active** medication or a **placebo**.

Group	None	Partial	Complete
Active	16	26	29
Placebo	24	26	18

Is there a statistically significant association here? That is to say, is there a statistically significant difference between the treatment groups in the distribution of responses?

28.2.1 Getting the Table into R

To answer this, we'll have to get the data from this contingency table into a matrix in R. Here's one approach...

```
T1 <- matrix(c(16,26,29,24,26,18), ncol=3, nrow=2, byrow=TRUE)
rownames(T1) <- c("Active", "Placebo")
colnames(T1) <- c("None", "Partial", "Complete")


T1
```

```
        None Partial Complete
Active    16      26       29
Placebo   24      26       18
```

28.2.2 Manipulating the Table's presentation

We can add margins to the matrix to get a table including row and column totals.

```
addmargins(T1)
```

```
        None Partial Complete Sum
Active    16      26       29  71
Placebo   24      26       18  68
Sum       40      52       47 139
```

Instead of the counts, we can tabulate the proportion of all patients within each cell.

```
prop.table(T1)
```

```
             None    Partial  Complete
Active  0.1151079 0.1870504 0.2086331
Placebo 0.1726619 0.1870504 0.1294964
```

Now, to actually obtain a p value and perform the significance test with H_0: rows and columns are independent vs. H_A: rows and columns are associated, we simply run a Pearson chi-square test on T1 ...

```
chisq.test(T1)
```

```
    Pearson's Chi-squared test

data:  T1
X-squared = 4.1116, df = 2, p-value = 0.128
```

Thanks to a p-value of about 0.13 (using the Pearson chi-square test) our conclusion would be to retain the null hypothesis of independence in this setting.

We could have run a Fisher's exact test, too, if we needed it.

```
  fisher.test(T1)
```

```
    Fisher's Exact Test for Count Data

data:  T1
p-value = 0.1346
alternative hypothesis: two.sided
```

The Fisher exact test p value is also 0.13. Either way, there is insufficient evidence to conclude that there is a (true) difference in the distributions of responses.

28.3 Accuracy of Death Certificates (A 6x3 Table)

The table below compiles data from six studies designed to investigate the accuracy of death certificates. The original citation is Kircher T, Nelson J, Burdo H (1985) The autopsy as a measure of accuracy of the death certificate. *NEJM*, 313, 1263-1269. 5373 autopsies were compared to the causes of death listed on the certificates. Of those, 3726 were confirmed to be accurate, 783 either lacked information or contained inaccuracies but did not require recoding of the underlying cause of death, and 864 were incorrect and required recoding. Do the results across studies appear consistent?

Date of Study	[Confirmed] Accurate	[Inaccurate] No Change	[Incorrect] Recoding	Total
1955-1965	2040	367	327	2734
1970	149	60	48	257
1970-1971	288	25	70	383
1975-1977	703	197	252	1152
1977-1978	425	62	88	575

Date of Study	[Confirmed] Accurate	[Inaccurate] No Change	[Incorrect] Recoding	Total
1980	121	72	79	272
Total	3726	783	864	5373

28.4 The Pearson Chi-Square Test of Independence

We can assess the homogeneity of the confirmation results (columns) we observe in the table using a Pearson chi-squared test of independence.

- The null hypothesis is that the rows and columns are independent.
- The alternative hypothesis is that there is an association between the rows and the columns.

```
death_table <- matrix(c(2040,367,327,149,60,48,288,25,70,703,
                        197,252,425,62,88,121,72,79), byrow=TRUE, nrow=6)
rownames(death_table) <- c("1955-65", "1970", "1970-71", "1975-77", "1977-78",
                "1980")
colnames(death_table) <- c("Confirmed", "Inaccurate", "Incorrect")

addmargins(death_table)
```

```
        Confirmed Inaccurate Incorrect  Sum
1955-65      2040        367       327 2734
1970          149         60        48  257
1970-71       288         25        70  383
1975-77       703        197       252 1152
1977-78       425         62        88  575
1980          121         72        79  272
Sum          3726        783       864 5373
```

To see the potential heterogeneity across rows in these data, we should perhaps also look at the proportions of autopsies in each of the three accuracy categories for each study.

```
addmargins(round_half_up(100*prop.table(death_table,1),1),2)
```

```
        Confirmed Inaccurate Incorrect Sum
1955-65      74.6       13.4      12.0 100
1970         58.0       23.3      18.7 100
```

```
1970-71        75.2         6.5       18.3 100
1975-77        61.0        17.1       21.9 100
1977-78        73.9        10.8       15.3 100
1980           44.5        26.5       29.0 100
```

In three of the studies, approximately 3/4 of the results were confirmed. In the other three, 45%, 58% and 61% were confirmed. It looks like there's a fair amount of variation in results across studies. To see if this is true, formally, we run Pearson's chi-square test of independence, where the null hypothesis is that the rows and columns are independent, and the alternative hypothesis is that there is an association between the rows and the columns.

```
chisq.test(death_table)
```

```
    Pearson's Chi-squared test

data:  death_table
X-squared = 209.09, df = 10, p-value < 2.2e-16
```

The chi-square test statistic is 200 on 10 degrees of freedom, yielding p < 0.0001.

Autopsies are not performed at random; in fact, many are done because the cause of death listed on the certificate is uncertain. What problems may arise if you attempt to use the results of these studies to make inference about the population as a whole?

28.5 Three-Way Tables: A 2x2xK Table and a Mantel-Haenszel Analysis

The material I discuss in this section is attributable to Jeff Simonoff and his book *Analyzing Categorical Data*. The example is taken from Section 8.1 of that book.

A three-dimensional or three-way table of counts often reflects a situation where the rows and columns refer to variables whose association is of primary interest to us, and the third factor (a layer, or strata) describes a control variable, whose effect on our primary association is something we are *controlling* for in the analysis.

28.5.1 Smoking and Mortality in the UK

In the early 1970s and then again 20 years later, in Whickham, United Kingdom, surveys yielded the following relationship between whether a person was a smoker at the time of the original survey and whether they were still alive 20 years later[1].

```
whickham1 <- matrix(c(502, 230, 443, 139), byrow=TRUE, nrow=2)
rownames(whickham1) <- c("Non-Smoker", "Smoker")
colnames(whickham1) <- c("Alive", "Dead")
addmargins(whickham1)
```

```
           Alive Dead  Sum
Non-Smoker   502  230  732
Smoker       443  139  582
Sum          945  369 1314
```

Here's the two-by-two table analysis.

```
twoby2(whickham1)
```

```
2 by 2 table analysis:
------------------------------------------------------
Outcome   : Alive
Comparing : Non-Smoker vs. Smoker

           Alive Dead    P(Alive) 95% conf. interval
Non-Smoker   502  230      0.6858    0.6512    0.7184
Smoker       443  139      0.7612    0.7248    0.7941

                                    95% conf. interval
            Relative Risk:  0.9010     0.8427    0.9633
         Sample Odds Ratio:  0.6848     0.5353    0.8761
Conditional MLE Odds Ratio:  0.6850     0.5307    0.8822
    Probability difference: -0.0754    -0.1230   -0.0266

           Exact P-value: 0.0030
       Asymptotic P-value: 0.0026
------------------------------------------------------
```

---

[1]See Appleton et al. 1996. Ignoring a Covariate: An Example of Simpson's Paradox. The American Statistician, 50, 340-341.

There is a detectable association between smoking and mortality, but it isn't the one you might expect.

- The odds ratio is 0.68, implying that the odds of having lived were only 68% as large for non-smokers as for smokers.
- Does that mean that smoking is *good* for you?

Not likely. There is a key "lurking" variable here - a variable that is related to both smoking and mortality that is obscuring the actual relationship - namely, age.

28.5.2 The `whickham` data with age, too

The table below gives the mortality experience separated into subtables by initial age group.

```
age <- c(rep("18-24", 4), rep("25-34", 4),
         rep("35-44", 4), rep("45-54", 4),
         rep("55-64", 4), rep("65-74", 4),
         rep("75+", 4))
smoking <- c(rep(c("Smoker", "Smoker", "Non-Smoker", "Non-Smoker"), 7))
status <- c(rep(c("Alive", "Dead"), 14))
counts <- c(53, 2, 61, 1, 121, 3, 152, 5,
            95, 14, 114, 7, 103, 27, 66, 12,
            64, 51, 81, 40, 7, 29, 28, 101,
            0, 13, 0, 64)

whickham2 <- tibble(smoking, status, age, counts) |>
    mutate(smoking = factor(smoking),
           status = factor(status),
           age = factor(age))

whickham2
```

```
# A tibble: 28 x 4
   smoking    status age   counts
   <fct>      <fct>  <fct>  <dbl>
 1 Smoker     Alive  18-24     53
 2 Smoker     Dead   18-24      2
 3 Non-Smoker Alive  18-24     61
 4 Non-Smoker Dead   18-24      1
 5 Smoker     Alive  25-34    121
 6 Smoker     Dead   25-34      3
```

```
 7 Non-Smoker Alive   25-34      152
 8 Non-Smoker Dead    25-34        5
 9 Smoker      Alive  35-44       95
10 Smoker      Dead   35-44       14
# i 18 more rows
```

```
whick_t2 <-
    xtabs(counts ~ smoking + status + age, data = whickham2)

whick_t2
```

```
, , age = 18-24

           status
smoking       Alive Dead
  Non-Smoker     61    1
  Smoker         53    2

, , age = 25-34

           status
smoking       Alive Dead
  Non-Smoker    152    5
  Smoker        121    3

, , age = 35-44

           status
smoking       Alive Dead
  Non-Smoker    114    7
  Smoker         95   14

, , age = 45-54

           status
smoking       Alive Dead
  Non-Smoker     66   12
  Smoker        103   27

, , age = 55-64

             status
```

```
smoking        Alive Dead
  Non-Smoker     81   40
  Smoker         64   51

, , age = 65-74

          status
smoking        Alive Dead
  Non-Smoker     28  101
  Smoker          7   29

, , age = 75+

          status
smoking        Alive Dead
  Non-Smoker      0   64
  Smoker          0   13
```

The sample odds ratios for remaining Alive comparing Non-Smokers to Smokers for each of these subtables (except the last one, where the odds ratio is undefined because of the zero cells) are calculated using the usual cross-product ratio approach, which yields the following results.

Age Group	Odds Ratio
18-24	2.30
25-34	0.75
35-44	2.40
45-54	1.44
55-64	1.61
65-74	1.15
75+	Undefined

Thus, for all age groups except 25-34 year olds, not smoking appears to be associated with higher odds of remaining alive.

Why? Not surprisingly, there is a strong association between age and mortality, with mortality rates being very low for young people (2.5% for 18-24 year olds) and increasing to 100% for 75+ year olds.

There is also an association between age and smoking, with smoking rates peaking in the 45-54 year old range and then falling off rapidly. In particular, respondents who were 65 and older at the time of the first survey had very low smoking rates (25.4%) but very high mortality

rates (85.5%). Smoking was hardly the cause, however, since even among the 65-74 year olds mortality was higher among smokers (80.6%) than it was among non-smokers (78.3%). A flat version of the table (`ftable` in R) can help us with these calculations.

```
ftable(whick_t2)
```

```
                 age 18-24 25-34 35-44 45-54 55-64 65-74 75+
smoking    status
Non-Smoker Alive         61   152   114    66    81    28   0
           Dead           1     5     7    12    40   101  64
Smoker     Alive         53   121    95   103    64     7   0
           Dead           2     3    14    27    51    29  13
```

28.5.3 Checking Assumptions: The Woolf test

We can also obtain a test (using the `woolf_test` function, in the `vcd` library) to see if the common odds ratio estimated in the Mantel-Haenszel procedure is reasonable for all age groups. In other words, the Woolf test is a test of the assumption of homogeneous odds ratios across the six age groups.

If the Woolf test is significant, it suggests that the Cochran-Mantel-Haenszel test is not appropriate, since the odds ratios for smoking and mortality vary too much in the sub-tables by age group. Here, we have the following log odds ratios (estimated using conditional maximum likelihood, rather than cross-product ratios) and the associated Woolf test.

```
## Next two results use the vcd package
oddsratio(whick_t2, log = TRUE)
```

```
log odds ratios for smoking and status by age

      18-24        25-34        35-44        45-54        55-64        65-74
 0.65018114  -0.22473479   0.84069420   0.34608770   0.47421763   0.09933253
        75+
-1.56397554
```

```
woolf_test(whick_t2)
```

```
    Woolf-test on Homogeneity of Odds Ratios (no 3-Way assoc.)
```

```
data:  whick_t2
X-squared = 3.2061, df = 6, p-value = 0.7826
```

As you can see, the Woolf test is not close to our usual standards for statistically detectable results, implying the common odds ratio is at least potentially reasonable for all age groups (or at least the ones under ages 75, where some data are available.)

28.5.4 The Cochran-Mantel-Haenszel Test

So, the marginal table looking at smoking and mortality combining all age groups isn't the most meaningful summary of the relationship between smoking and mortality. Instead, we need to look at the *conditional* association of smoking and mortality, **given age**, to address our interests.

The null hypothesis would be that, in the population, smoking and mortality are independent within strata formed by age group. In other words, H_0 requires that smoking be of no value in predicting mortality once age has been accounted for.

The alternative hypothesis would be that, in the population, smoking and mortality are associated within the strata formed by age group. In other words, H_A requires that smoking be of at least some value in predicting mortality even after age has been accounted for.

We can consider the evidence that helps us choose between these two hypotheses with a Cochran-Mantel-Haenszel test, which is obtained in R through the `mantelhaen.test` function. This test requires us to assume that, in the population and within each age group, the smoking-mortality odds ratio is the same. Essentially, this means that the association of smoking with mortality is the same for older and younger people.

```
mantelhaen.test(whick_t2, conf.level  = 0.90)
```

```
    Mantel-Haenszel chi-squared test with continuity correction

data:  whick_t2
Mantel-Haenszel X-squared = 5.435, df = 1, p-value = 0.01974
alternative hypothesis: true common odds ratio is not equal to 1
90 percent confidence interval:
 1.143198 2.041872
sample estimates:
common odds ratio
        1.52783
```

- The Cochran-Mantel-Haenszel test statistic is a bit larger than 5 (after a continuity correction) leading to a p value of 0.02, indicating strong rejection of the null hypothesis of conditional independence of smoking and survival given age.
- The estimated common conditional odds ratio is 1.53. This implies that (adjusting for age) being a non-smoker is associated with 53% higher odds of being alive 20 years later than being a smoker.
- A 90% confidence interval for that common odds ratio is (1.14, 2.04), reinforcing rejection of the null hypothesis of conditional independence (where the odds ratio would be 1).

28.5.5 Without the Continuity Correction

By default, R presents the Mantel-Haenszel test with a continuity correction, when used for a 2x2xK table. In virtually all cases, go ahead and do this, but as you can see below, the difference it makes in this case is modest.

```
mantelhaen.test(whick_t2, correct=FALSE, conf.level = 0.90)
```

```
    Mantel-Haenszel chi-squared test without continuity correction

data:  whick_t2
Mantel-Haenszel X-squared = 5.8443, df = 1, p-value = 0.01563
alternative hypothesis: true common odds ratio is not equal to 1
90 percent confidence interval:
 1.143198 2.041872
sample estimates:
common odds ratio
         1.52783
```

Part III

Part C. Building Models

29 Multiple Regression: Introduction

In Chapter 16, while working with a study of dehydration recovery in children, we discussed many of the fundamental ideas of multiple regression. There, we provided code and insight into the scatterplot and the scatterplot matrix, fit linear models and plotted the coefficients, analyzed summary output from `summary()`, `tidy()` and `glance()` as well as the ANOVA table, and plotted residuals vs. fitted values using the `augment()` function.

In the remaining chapters, we will build on that foundation in some examples, including...

- The `wcgs` data from the Western Collaborative Group Study, which we described in Chapter 17.
- The `emp_bmi` data from a study published in *BMJ Open* of a nationally representative sample of over 7000 participants in the Korean Longitudinal Study of Aging.
- The `gala` data, which describe features of the 30 Galapagos Islands.

29.1 Reminders of a few Key Concepts

1. **Scatterplots** We have often accompanied our scatterplots with regression lines estimated by the method of least squares, and by loess smooths which permit local polynomial functions to display curved relationships, and occasionally presented in the form of a scatterplot matrix to enable simultaneous comparisons of multiple two-way associations.

2. **Measures of Correlation/Association** By far the most commonly used is the Pearson correlation, which is a unitless (scale-free) measure of bivariate linear association for the variables X and Y, symbolized by r, and ranging from -1 to +1. The Pearson correlation is a function of the slope of the least squares regression line, divided by the product of the standard deviations of X and Y. We have also mentioned the *Spearman* rank correlation coefficient, which is obtained by using the usual formula for a Pearson correlation, but on the ranks (1 = minimum, n = maximum, with average ranks are applied to the ties) of the X and Y values. This approach (running a correlation of the orderings of the data) substantially reduces the effect of outliers. The result still ranges from -1 to +1, with 0 indicating no linear association.

3. **Fitting Linear Models** We have fit several styles of linear model to date, including both *simple* regressions, where our outcome Y is modeled as a linear function of a single predictor X, and *multiple* regression models, where more than one predictor is used.

547

Functions from the `broom` package can be used to yield several crucial results, in addition to those we can obtain with a `summary` of the model. These include:

- (from `tidy`) the estimated coefficients (intercept and slope(s)) of the fitted model, and
- (from `glance`) the R^2 or coefficient of determination, which specifies the proportion of variation in our outcome accounted for by the linear model, and various other summaries of the model's quality of fit
- (from `augment`) fitted values, residuals and other summaries related to individual points used to fit the model, or individual predictions made by the model, which will be helpful for assessing predictive accuracy and for developing diagnostic tools for assessing the assumptions of multiple regression.

29.2 What is important in 431?

In 431, my primary goal is to immerse you in several cases, which will demonstrate good statistical practice in the analysis of data using multiple regression models. Often, we will leave gaps for 432, but the principal goal is to get you to the point where you can do a solid (if not quite complete) analysis of data for the modeling part (Study 2) of Project B.

Key topics regarding multiple regression we cover in 431 include:

1. Describing the multivariate relationship - Scatterplots and smoothing - Correlation coefficients, Correlation matrices
2. Transformations and Re-expression - The need for transformation - Using a Box-Cox method to help identify effective transformation choices - Measuring and addressing collinearity
3. Testing the significance of a multiple regression model - T tests for individual predictors as last predictor in - Global F tests based on ANOVA to assess overall predictive significance - Incremental and Sequential testing of groups of predictors
4. Interpreting the predictive value of a model - R^2 and Adjusted R^2, along with AIC and BIC - Residual standard deviation and RMSE
5. Checking model assumptions - Residual Analysis including studentized residuals, and the major plots - Identifying points with high Leverage - Assessing Influence numerically and graphically
6. Model Selection - The importance of parsimony - Stepwise regression
7. Assessing Predictive Accuracy through Cross-Validation - Summaries of predictive error - Plotting predictions across multiple models
8. Summarizing the Key Findings of the Model, briefly and accurately - Making the distinction between causal findings and associations - The importance of logic, theory and empirical evidence. (LTE)

30 Regression Diagnostics

30.1 Setup: Packages Used Here

```r
knitr::opts_chunk$set(comment = NA)

library(broom)
library(car)
library(GGally)
library(ggrepel)
library(janitor)
library(kableExtra)
library(mosaic)
library(patchwork)
library(tidyverse)

theme_set(theme_bw())
```

30.2 Introduction

Some of this discussion comes from Bock, Velleman, and De Veaux (2004).

Multiple linear regression (also called ordinary least squares, or OLS) has four main assumptions that are derived from its model.

For a simple regression, the model underlying the regression line is $E(Y) = \beta_0 + \beta_1 x$

- where E(Y) = the expectation (mean) of the outcome Y,
- β_0 is the intercept, and
- β_1 is the slope

Now, if we have a multiple regression with three predictors x_1, x_2 and x_3, then the model becomes $E(Y) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$.

Alternatively, we can write the model to relate individual y's to the x's by adding an individual error term: $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \epsilon$

Of course, the multiple regression model is not limited to three predictors. We will often use k to represent the number of coefficients (slopes plus intercept) in the model, and will also use p to represent the number of predictors (usually $p = k$ - 1).

30.3 Building an Example: 500 subjects from WCGS

To fix ideas, suppose we return to the Western Collaborative Group Study (WCGS) data we studied in Chapter 17, which described 3154 men ages 39-59 who were free of heart disease and were employees of one of 10 California companies in 1960-1961.

We are going to select a random sample of 500 of those men, with complete data on our outcome of interest, which is total cholesterol in mg/dl (`chol`) and on our three predictors of interest, which are systolic blood pressure in mm Hg (`sbp`), body mass index (`bmi`) and behavioral pattern (`dibpat`), which is a two-category variable with levels "Type A" or "Type B".

- Individuals exhibiting a Type A behavior pattern show evidence of being competitive and self-critical without feeling joy in their efforts or accomplishments, and are more likely to have high work involvement, important life imbalance, and high blood pressure.
- Individuals exhibiting a Type B behavior pattern show evidence of being more reflective, more relaxed, less competitive and less anxious then Type A individuals.
- There is substantial evidence to discredit the Type A Behavior Pattern theory in some areas, and some scholars definitely argue that Type A behavior is not a good predictor of coronary heart disease.

Below we create this sample of 500 subjects with complete data on our outcome and each of our three predictors.

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

set.seed(12345)

wcgs_500 <- wcgs |>
    select(id, sbp, bmi, dibpat, chol) |>
    filter(complete.cases(id, sbp, bmi, dibpat, chol)) |>
    slice_sample(n = 500, replace = FALSE)
```

Next, we'll look at a scatterplot matrix using `ggpairs` from the `GGally` package, placing our outcome last, as usual.

```
ggpairs(wcgs_500 |> select(sbp, bmi, dibpat, chol),
        lower = list(combo = wrap("facethist", binwidth = 0.5)),
        title = "Scatterplot Matrix for 500 WCGS subjects")
```


Scatterplot Matrix for 500 WCGS subjects
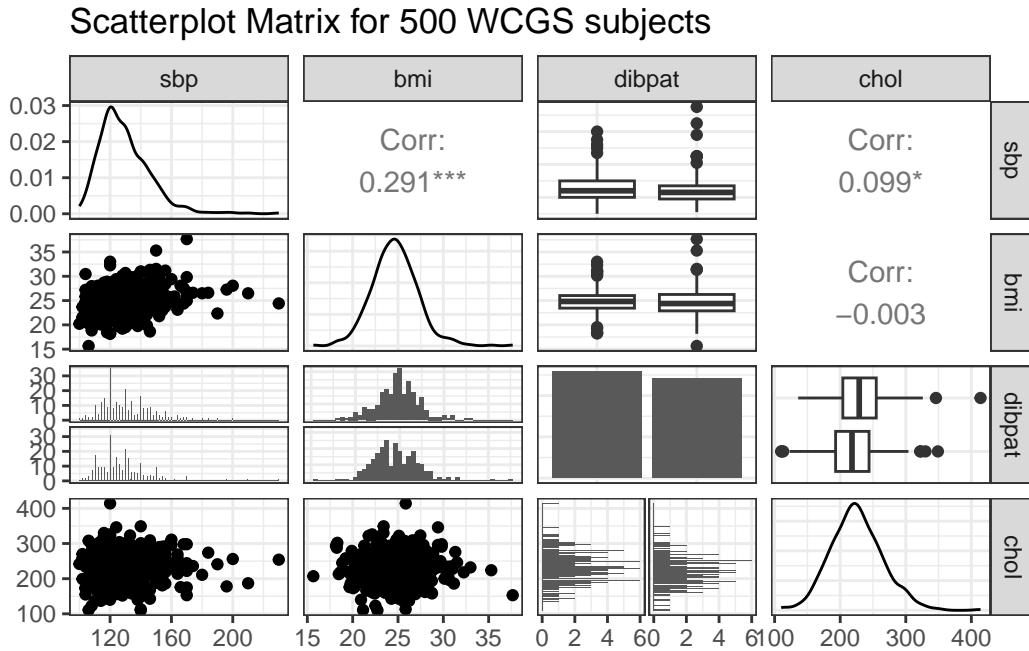
Next, we create our planned linear model, and evaluate the coefficients and fit quality.

```
chol_m1 <- lm(chol ~ sbp + bmi + dibpat, data = wcgs_500)

summary(chol_m1)
```

```
Call:
lm(formula = chol ~ sbp + bmi + dibpat, data = wcgs_500)

Residuals:
     Min       1Q   Median       3Q      Max
-113.133  -25.689   -1.932   24.036  186.390

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  214.5185    19.8825   10.789  < 2e-16 ***
```

term	estimate	std.error	conf.low	conf.high	p.value
(Intercept)	214.519	19.883	181.753	247.284	0.000
sbp	0.236	0.112	0.051	0.421	0.036
bmi	-0.589	0.729	-1.791	0.612	0.419
dibpatType B	-11.041	3.564	-16.915	-5.167	0.002

r.squared	adj.r.squared	sigma	AIC	BIC	nobs
0.03	0.024	39.7	5105.3	5126.4	500

```
sbp            0.2361     0.1125    2.099   0.03634 *
bmi           -0.5894     0.7290   -0.808   0.41921
dibpatType B -11.0408     3.5643   -3.098   0.00206 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 39.66 on 496 degrees of freedom
Multiple R-squared:  0.02954,   Adjusted R-squared:  0.02367
F-statistic: 5.033 on 3 and 496 DF,  p-value: 0.001915
```

As usual, we'll use `tidy` and `glance` from the `broom` package (part of `tidymodels`) to evaluate the coefficients in more detail, and summarize the quality of fit for our model.

```
tidy(chol_m1, conf.int = TRUE, conf.level = 0.90) |>
    select(term, estimate, std.error, conf.low, conf.high, p.value) |>
    kbl(digits = 3) |> kable_paper()
```

```
glance(chol_m1) |>
    select(r.squared, adj.r.squared, sigma, AIC, BIC, nobs) |>
    kbl(digits = c(3, 3, 1, 1, 1, 0)) |> kable_paper()
```

This isn't a terrific model, accounting for only 3% (according to R^2) of the variation in our outcome, although some of the individual predictors show effects that meet our usual standards for "detectable" results. Our goal, though, is to use this model as an example for studying residual diagnostics to search for possible violations of regression assumptions.

To assess regression assumptions, we'll have to look at residuals and fits for each of the individual points, which we can obtain use the `augment` function from the `broom` package.

```
aug_chol1 <- augment(chol_m1, data = wcgs_500)
```

This augments the original data with several summaries of the quality of fit for our individual rows in the `wcgs_500` data. Here are the first few values for some of the elements of `augment`.

id	sbp	bmi	dibpat	chol	.fitted	.resid
10155	120	18.162	Type B	294	221.104	72.896
3495	140	24.807	Type A	301	232.950	68.050
12634	124	21.616	Type A	196	231.054	-35.054
21354	114	25.381	Type A	176	226.474	-50.474
3085	190	22.349	Type A	241	246.203	-5.203
3589	140	23.054	Type B	268	222.943	45.057

id	.std.resid	.cooksd
10155	1.854	0.015
3495	1.720	0.003
12634	-0.887	0.001
21354	-1.277	0.003
3085	-0.134	0.000
3589	1.140	0.002

```
head(aug_chol1 |> select(id:.resid)) |>
  kbl(digits = 3) |> kable_minimal()
```

This includes the original data, plus the model's predicted value for `chol`, stored in `.fitted` and the residual (actual `chol` - `.fitted`) stored in `.resid`. Here are the other elements of the `augment` result.

```
head(aug_chol1 |> select(id, .std.resid:.cooksd)) |>
  kbl(digits = 3) |> kable_minimal()
```

Here, we see:

- `.std.resid` the standardized residual, which is a measure of how poorly fit the point is, scaled to have standard deviation 1 (the raw residuals, as well as these, already have mean 0)
- `.hat`, the hat matrix value, which is a measurement of the **leverage** this point has on the model. Leverage is, essentially, a measure of how unusual the point is in terms of the predictors.
- `.sigma`, the value of the residual standard error `.sigma` that would emerge if this point was deleted from the `chol_m1` model, and
- `.cooksd`, Cook's distance, or Cook's d, which measures the **influence** of this point on the model. A highly influential point (one with `.cooksd` greater than 0.5, for instance) would be a point that, if removed from the `chol_m1` model, would substantially change the fit of the model or the coefficient estimates.

30.4 The Four Key Regression Assumptions

The assumptions and conditions for a multiple regression model are nearly the same as those for simple regression. The key assumptions that must be checked in a multiple regression model are:

- Linearity Assumption
- Independence Assumption
- Equal Variance (Constant Variance / Homoscedasticity) Assumption
- Normality Assumption

Happily, R is well suited to provide us with multiple diagnostic tools to generate plots and other summaries that will let us look more closely at each of these assumptions.

30.5 The Linearity Assumption

We are fitting a linear model.

- By *linear*, we mean that each predictor value, x, appears simply multiplied by its coefficient and added to the model.
- No x appears in an exponent or some other more complicated function.
- If the regression model is true, then the outcome y is linearly related to each of the x's.

Unfortunately, assuming the model is true is not sufficient to prove that the linear model fits, so we check what Bock, Velleman, and De Veaux (2004) call the "Straight Enough Condition"

30.5.1 Initial Scatterplots for the "Straight Enough" Condition

- Scatterplots of y against each of the predictors are reasonably straight.
- The scatterplots need not show a strong (or any!) slope; we just check that there isn't a bend or other nonlinearity.
- Any substantial curve is indicative of a potential problem.
- Modest bends are not usually worthy of serious attention.

For example, in the `wcgs_500` data, here are the relevant scatterplots for the quantitative predictors (in practice, I would simply look at the scatterplot matrix shown earlier) as well as a boxplot with violin for the categorical predictor (`dibpat`).

```
p1 <- ggplot(wcgs_500, aes(x = sbp, y = chol)) +
    geom_point() +
    geom_smooth(method = "loess", se = FALSE, span = 2, formula = y ~ x) +
```

```
    geom_smooth(method = "lm", se = FALSE, col = "tomato", formula = y ~ x)

p2 <- ggplot(wcgs_500, aes(x = bmi, y = chol)) +
    geom_point() +
    geom_smooth(method = "loess", se = FALSE, span = 2, formula = y ~ x) +
    geom_smooth(method = "lm", se = FALSE, col = "tomato", formula = y ~ x)

p3 <- ggplot(wcgs_500, aes(x = dibpat, y = chol)) +
    geom_violin(aes(fill = dibpat)) +
    geom_boxplot(width = 0.3) +
    coord_flip() + guides(fill = "none")

(p1 + p2) / p3 +
    plot_layout(heights = c(2,1)) +
    plot_annotation(title = "WCGS_500 Cholesterol vs. Predictors")
```
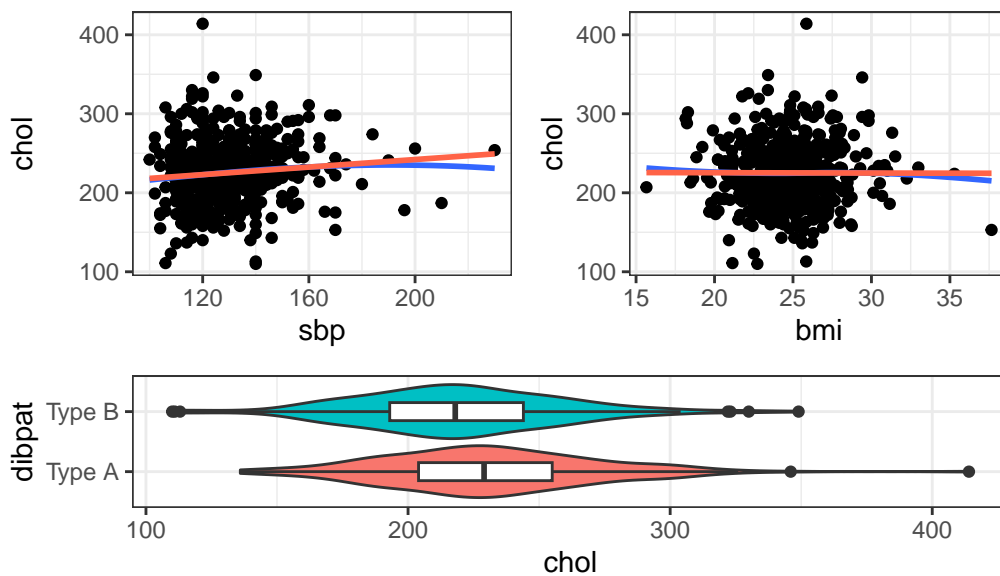

Here, I've simply placed `recov.score` on the vertical (Y) axis, plotted against each of the predictors, in turn. I've added a straight line (OLS) fit [in tomato red] and a loess smooth [in blue] to each plot to guide your assessment of the "straight enough" condition. Note the use here of `span = 2` in the `loess` smooths to produce lines that are less wiggly than they would be using the default `span = 0.75`.

- Each of these is "straight enough" for our purposes, in initially fitting the data.
- If one of these was not, we might consider a transformation of the relevant predictor, or, if all were problematic, we might transform the outcome Y.

30.5.2 Residuals vs. Predicted Values to Check for Non-Linearity

The residuals should appear to have no pattern (no curve, for instance) with respect to the predicted (fitted) values. It is a very good idea to plot the residuals against the fitted values to check for patterns, especially bends or other indications of non-linearity. For a multiple regression, the fitted values are a combination of the x's given by the regression equation, so they combine the effects of the x's in a way that makes sense for our particular regression model. That makes them a good choice to plot against. We'll check for other things in this plot, as well.

When you ask R to plot the result of a linear model, it will produce up to five separate plots: the first of which is a plot of **residuals vs. fitted values**.

30.5.2.1 Using `plot(model, which = 1)`

To obtain this plot for the model including `sbp`, `bmi` and `dibpat` to predict `chol` we have two options. The simpler approach uses the following code to indicate plot 1 using the `which` command within the `plot` function.

```
plot(chol_m1, which=1)
```

Residuals vs Fitted

lm(chol ~ sbp + bmi + dibpat)
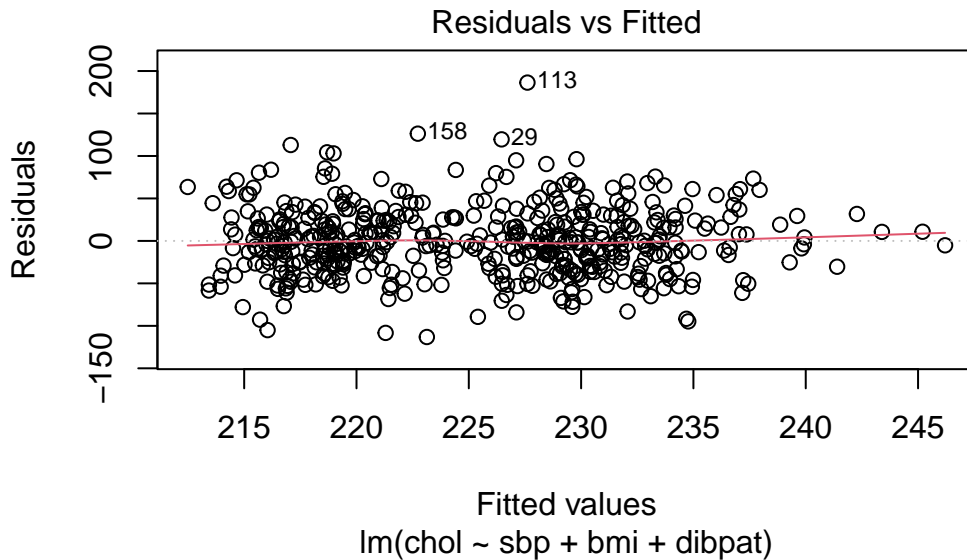
A smooth is again added (in red) to help you identify serious non-linearity. In this case, I would conclude that there were no serious problems with linearity in these data.

The plot also, by default, identifies the three values[1] with the largest (in absolute value) residuals.

- Here, these are rows 29, 113 and 158, each of which has a positive residual (i.e. they represent under-predictions by the model.) To identify these points in the data, use the `slice` function.

```
slice(wcgs_500, c(29, 113, 158))
```

```
# A tibble: 3 x 5
      id   sbp   bmi dibpat  chol
   <dbl> <dbl> <dbl> <chr>  <dbl>
1   3604   124  29.4 Type A   346
2   2280   120  25.9 Type A   414
3  10360   140  23.4 Type B   349
```

These turn out to be the observations with `id` = 3604, 2280 and 10360, respectively.

---

[1]If you wanted to identify more or fewer points, you could, i.e. 'plot(modelname, which=1, id.n = 2)'

30.5.2.2 Using `ggplot2` to plot Residuals vs. Predicted Values

```
ggplot(aug_chol1, aes(x = .fitted, y = .resid)) +
  geom_point() +
  geom_point(data = aug_chol1 |>
                slice_max(abs(.resid), n = 5),
             col = "red", size = 2) +
  geom_text_repel(data = aug_chol1 |>
                slice_max(abs(.resid), n = 5),
                aes(label = id), col = "red") +
  geom_abline(intercept = 0, slope = 0, lty = "dashed") +
  geom_smooth(method = "loess", formula = y ~ x, se = F) +
  labs(title = "Residuals vs. Fitted Values from chol_m1",
       caption = "5 largest |residuals| highlighted in red.",
       x = "Fitted Total Cholesterol", y = "Residual") +
  theme(aspect.ratio = 1)
```


5 largest |residuals| highlighted in red.
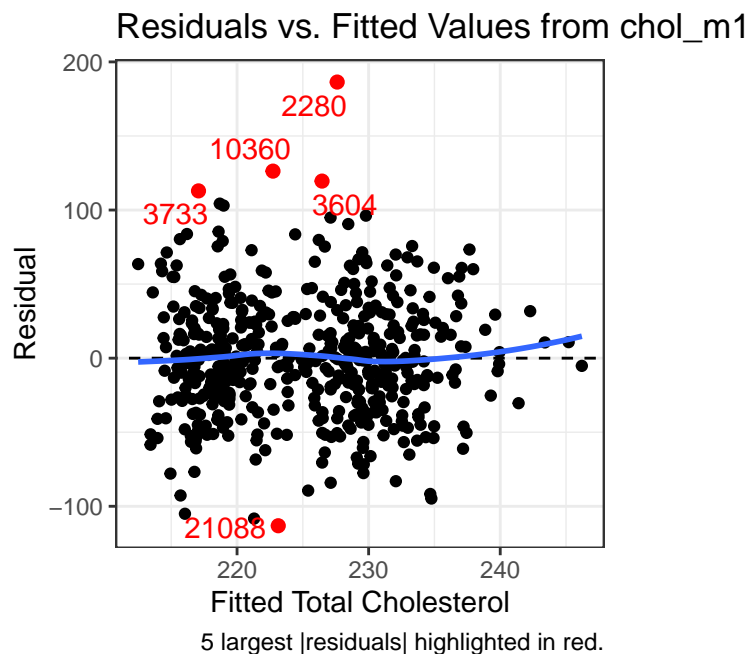
30.5.3 Residuals vs. Predictors To Further Check for Non-Linearity

If we do see evidence of non-linearity in the plot of residuals against fitted values, I usually then proceed to look at residual plots against each of the individual predictors, in turn, to try

to identify the specific predictor (or predictors) where we may need to use a transformation.

The appeal of such plots (as compared to the initial scatterplots we looked at of the outcome against each predictor) is that they eliminate the distraction of the linear fit, and let us look more closely at the non-linear part of the relationship between the outcome and each quantitative predictor. We might also look at a boxplot of the relationship between the outcome and a categorical predictor, although this is primarily for the purpose of assessing the potential for non-constant variance, rather than non-linearity.

Although I don't think you need them here, here are these plots for the residuals from our `chol_m1` model.

```
p1 <- ggplot(aug_chol1, aes(x = sbp, y = chol)) +
    geom_point() +
    geom_smooth(method = "loess", formula = y ~ x, span = 2, se = FALSE) +
    geom_smooth(method = "lm", formula = y ~ x, se = FALSE, col = "tomato")
p2 <- ggplot(aug_chol1, aes(x = bmi, y = chol)) +
    geom_point() +
    geom_smooth(method = "loess", formula = y ~ x, span = 2, se = FALSE) +
    geom_smooth(method = "lm", formula = y ~ x, se = FALSE, col = "tomato")
p3 <- ggplot(aug_chol1, aes(x = dibpat, y = chol)) +
    geom_violin(aes(fill = dibpat)) + geom_boxplot(width = 0.3) +
    coord_flip() + guides(fill = "none")

(p1 + p2) / p3 +
    plot_layout(heights = c(2,1)) +
    plot_annotation(title = "Residuals vs. Predictors in chol_m1 model")
```

Residuals vs. Predictors in chol_m1 model
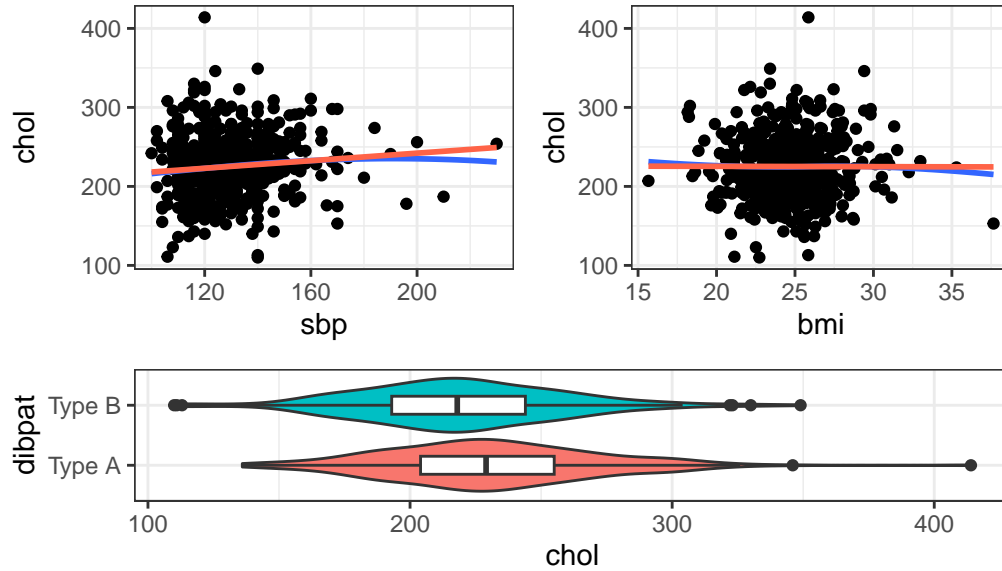
Again, I see no particularly problematic issues with the assumption of linearity in either of the scatterplots here.

30.6 The Independence Assumption

The errors in the true underlying regression model must be mutually independent, but there is no way to be sure that the independence assumption is true. Fortunately, although there can be many predictor variables, there is only one outcome variable and only one set of errors. The independence assumption concerns the errors, so we check the residuals.

Randomization condition. The data should arise from a random sample, or from a randomized experiment.

- The residuals should appear to be randomly scattered and show no patterns, trends or clumps when plotted against the predicted values.
- In the special case when an x-variable is related to time, make sure that the residuals do not have a pattern when plotted against time.

30.6.1 Residuals vs. Fitted Values to Check for Dependence

The `wcgs_500` men were not related in any way to each other except that they were drawn from the same 10 employers, and the data are cross-sectional here, so we can be pretty sure that their measurements are independent. The residuals vs. fitted values plots we've already seen show no clear trend, cycle or pattern which concerns us.

30.7 The Constant Variance Assumption

The variability of our outcome, y, should be about the same for all values of every predictor x. Of course, we can't check every combination of x values, so we look at scatterplots and check to see if the plot shows a "fan" shape - in essence, the **Does the plot thicken? Condition.**

- Scatterplots of residuals against each x or against the predicted values, offer a visual check.
- Be alert for a "fan" or a "funnel" shape showing growing/shrinking variability in one part of the plot.

Reviewing the same plots we have previously seen, I can find no evidence of substantial "plot thickening" in either the plot of residuals vs. fitted values or in any of the plots of the residuals against each predictor separately.

30.7.1 The Scale-Location Plot to Check for Non-Constant Variance

R does provide an additional plot to help assess this issue as linear model diagnostic plot 3. This one looks at the square root of the standardized residual plotted against the fitted values. You want the loess smooth in this plot to be flat, rather than sloped.

```
plot(chol_m1, which=3)
```

Scale–Location

lm(chol ~ sbp + bmi + dibpat)
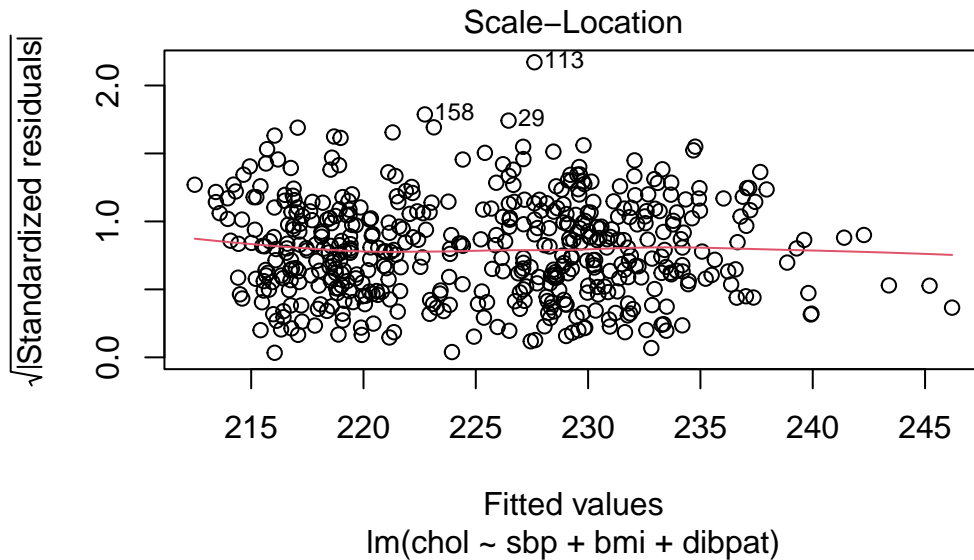
We can replicate this plot using `ggplot2` as follows. With a `loess` smooth with `span = 2`, there's no suggestion of a meaningful trend up or down in this plot.

```
ggplot(aug_chol1, aes(x = .fitted, y = sqrt(abs(.std.resid)))) +
  geom_point() +
  geom_point(data = aug_chol1 |>
               slice_max(sqrt(abs(.std.resid)), n = 3),
             col = "red", size = 1) +
  geom_text_repel(data = aug_chol1 |>
               slice_max(sqrt(abs(.std.resid)), n = 3),
               aes(label = id), col = "red") +
  geom_smooth(method = "loess", formula = y ~ x, span = 2, se = F) +
  labs(title = "Scale-Location Plot for chol_m1 model",
       caption = "3 largest |Standardized Residual| in red.",
       x = "Fitted Value of Total Cholesterol",
       y = "Square Root of |Standardized Residual|") +
  theme(aspect.ratio = 1)
```

Scale–Location Plot for chol_m1 model

3 largest |Standardized Residual| in red.
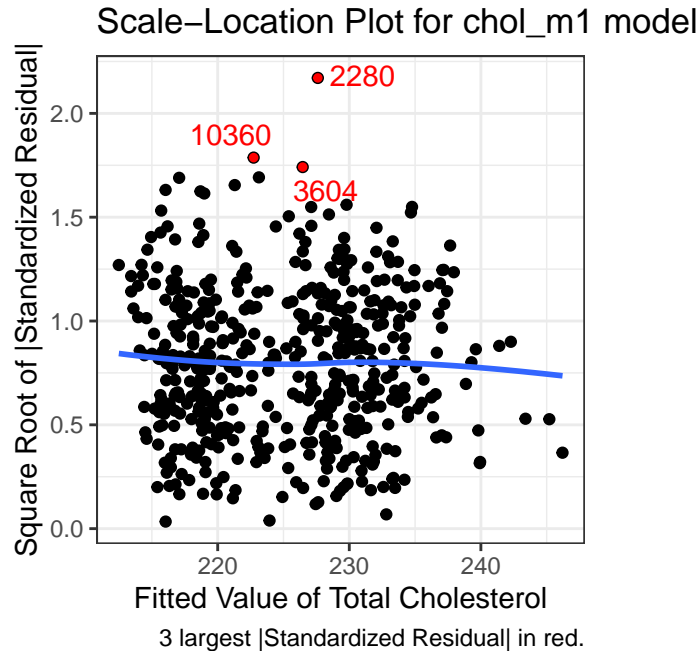
30.7.2 What does trouble look like?

It's helpful to see how this works in practice. Here are three sets of plots for two settings, where the left plots in each pair show simulated data that are homoscedastic (variance is constant across the levels of the predictor) and the right plots show data that are heteroscedastic (variance is not constant across predictor levels.) *Source*: This example is modified from http://goo.gl/weMI0U [from stats.stackexchange.com]

Here, I'll create a tibble containing fake data on a predictor (x) and two outcomes (y1 or y2). The relationship between y1 and x is meant to show no problems with the assumption of homoscedasticity. The relationship between y2 and x, on the other hand, should show severe problems with this assumption.

```
set.seed(5)

N  = 250     ## original n = 500
b0 = 3
b1 = 0.4

s2 = 5
g1 = 1.5     ## original g1 = 1.5
g2 = 0.02    ## original g2 = 0.015
```

```
x          = runif(N, min=0, max=100)
y1   = b0 + b1*x + rnorm(N, mean=0, sd=sqrt(s2              ))
y2 = b0 + b1*x + rnorm(N, mean=0, sd=sqrt(exp(g1 + g2*x)))

fake_1 <- tibble(id = 1:250, x, y1, y2)
```

First, we'll fit our two models, and obtain augmented values.

```
fake_m1 <- lm(y1 ~ x, data = fake_1) # no problems
fake_m2 <- lm(y2 ~ x, data = fake_1) # non-constant variance

aug_fake1 <- augment(fake_m1, data = fake_1)
aug_fake2 <- augment(fake_m2, data = fake_1)
```

Now, let's look at the plots, developed first using ggplot2.

First, here are the two scatterplots, for our fake_1 model with constant variance on the left, and our fake_2 model with non-constant variance on the right.

```
p1a <- ggplot(fake_1, aes(x = x, y = y1)) +
    geom_point() +
    geom_smooth(method = "loess", formula = y ~ x, span = 2, se = F) +
    geom_smooth(method = "lm", formula = y ~ x, se = F, col = "tomato") +
    labs(title = "Constant Variance (fake_m1)",
         subtitle = "Scatterplot of y1 vs. x")

p1b <- ggplot(fake_1, aes(x = x, y = y2)) +
    geom_point() +
    geom_smooth(method = "loess", formula = y ~ x, span = 2, se = F) +
    geom_smooth(method = "lm", formula = y ~ x, se = F, col = "tomato") +
    labs(title = "Non-Constant Variance (fake_m2)",
         subtitle = "Scatterplot of y2 vs. x")

p1a + p1b
```
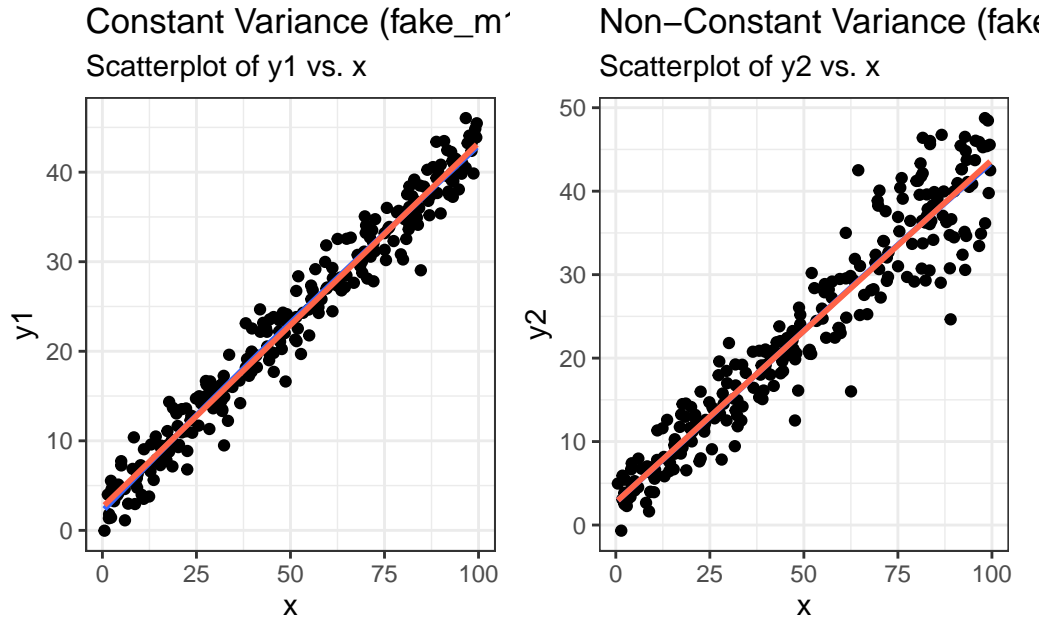
Second, here are the two plots of residuals vs. fitted values, for our `fake_1` model with constant variance on the left, and our `fake_2` model with non-constant variance on the right.

```
p2a <- ggplot(aug_fake1, aes(x = .fitted, y = .resid)) +
    geom_point() +
    geom_point(data = aug_fake1 |>
                    slice_max(abs(.resid), n = 5),
                col = "red", size = 2) +
    geom_text_repel(data = aug_fake1 |>
                        slice_max(abs(.resid), n = 5),
                aes(label = id), col = "red") +
    geom_abline(intercept = 0, slope = 0, lty = "dashed") +
    geom_smooth(method = "loess", formula = y ~ x, se = F) +
    labs(title = "Constant Variance (fake_m1)",
        subtitle = "Residuals vs. Fitted Values",
        caption = "5 largest |residuals| highlighted in red.",
        x = "Fitted Value of y1", y = "Residual") +
    theme(aspect.ratio = 1)

p2b <- ggplot(aug_fake2, aes(x = .fitted, y = .resid)) +
    geom_point() +
    geom_point(data = aug_fake2 |>
```

```
                slice_max(abs(.resid), n = 5),
            col = "red", size = 2) +
    geom_text_repel(data = aug_fake2 |>
                    slice_max(abs(.resid), n = 5),
            aes(label = id), col = "red") +
    geom_abline(intercept = 0, slope = 0, lty = "dashed") +
    geom_smooth(method = "loess", formula = y ~ x, se = F) +
    labs(title = "Non-Constant Variance (fake_m2)",
         subtitle = "Residuals vs. Fitted Values",
         caption = "5 largest |residuals| highlighted in red.",
         x = "Fitted Value of y1", y = "Residual") +
    theme(aspect.ratio = 1)

p2a + p2b
```
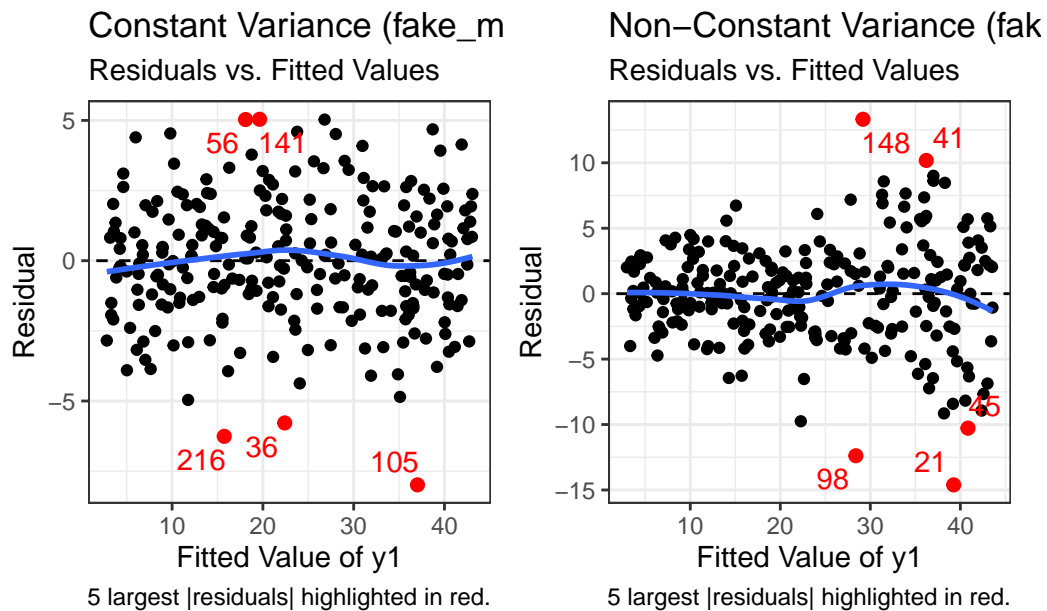

Finally, here is the scale-location plot for each scenario.

```
p3a <- ggplot(aug_fake1, aes(x = .fitted, y = sqrt(abs(.std.resid)))) +
    geom_point() +
    geom_point(data = aug_fake1 |>
                    slice_max(sqrt(abs(.std.resid)), n = 3),
```

```r
                  col = "red", size = 1) +
  geom_text_repel(data = aug_fake1 |>
                slice_max(sqrt(abs(.std.resid)), n = 3),
                aes(label = id), col = "red") +
  geom_smooth(method = "loess", formula = y ~ x, span = 2, se = F) +
  labs(title = "Constant Variance (fake_m1)",
       subtitle = "Scale-Location Plot",
       caption = "3 largest |Standardized Residual| in red.",
       x = "Fitted Value",
       y = "Square Root of |Standardized Residual|") +
  theme(aspect.ratio = 1)

p3b <- ggplot(aug_fake2, aes(x = .fitted, y = sqrt(abs(.std.resid)))) +
  geom_point() +
  geom_point(data = aug_fake2 |>
                slice_max(sqrt(abs(.std.resid)), n = 3),
             col = "red", size = 1) +
  geom_text_repel(data = aug_fake2 |>
                slice_max(sqrt(abs(.std.resid)), n = 3),
                aes(label = id), col = "red") +
  geom_smooth(method = "loess", formula = y ~ x, span = 2, se = F) +
  labs(title = "Non-Constant Variance (fake_m2)",
       subtitle = "Scale-Location Plot",
       caption = "3 largest |Standardized Residual| in red.",
       x = "Fitted Value",
       y = "Square Root of |Standardized Residual|") +
  theme(aspect.ratio = 1)

p3a + p3b
```

Constant Variance (fake_m — Scale–Location Plot

Non–Constant Variance (fak — Scale–Location Plot

3 largest |Standardized Residual| in red.
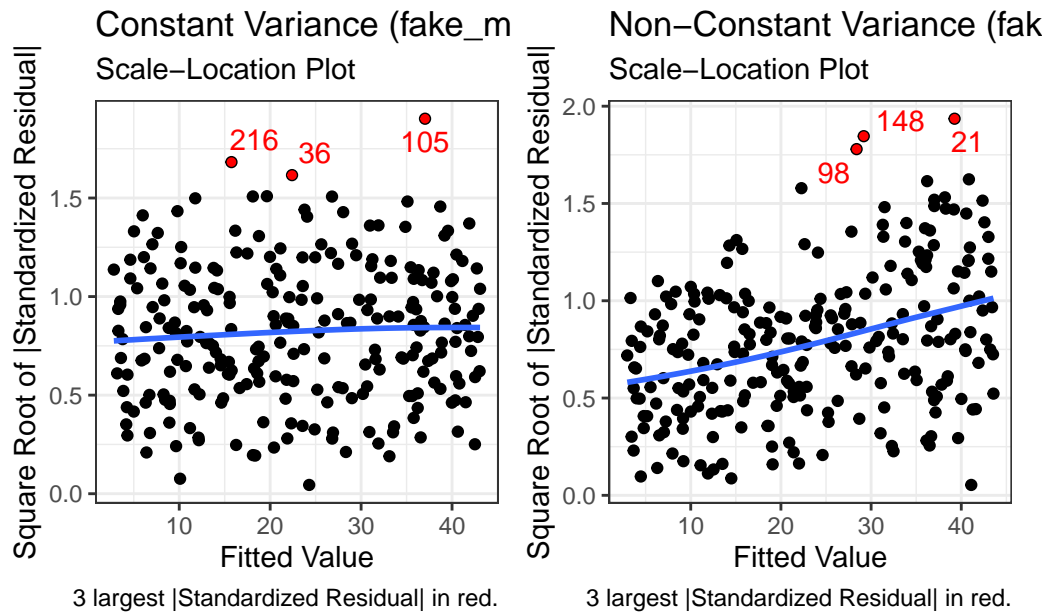
Note the funnel shape for the upper two heteroscedastic (non-constant variance) plots, and the upward sloping loess line in the last one.

Here are the relevant plots for the same scenario using base R.

```r
par(mfrow=c(3,2))
plot(y1 ~ x, data = fake_1, main="Constant Variance")
plot(y2 ~ x, data = fake_1, main="Non-Constant Variance")
plot(fake_m1, which=1)
plot(fake_m2, which=1)
plot(fake_m1, which=3)
plot(fake_m2, which=3)
```
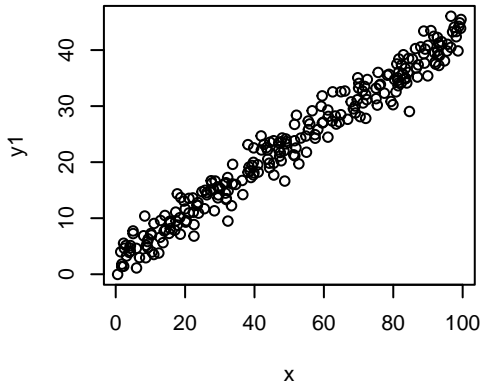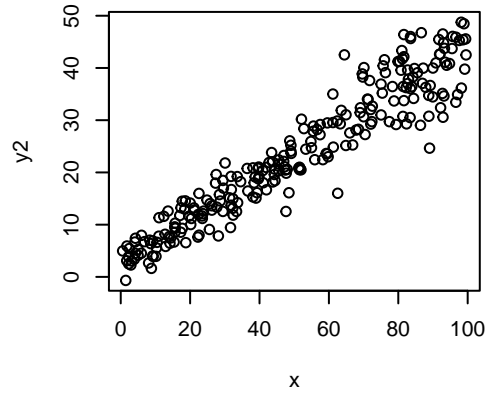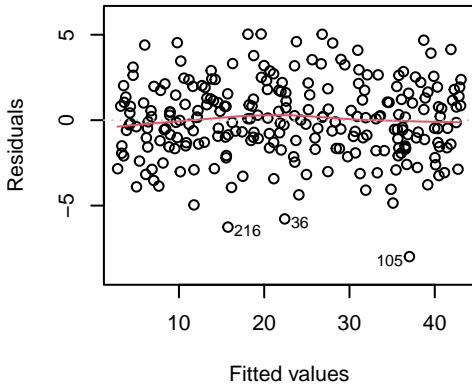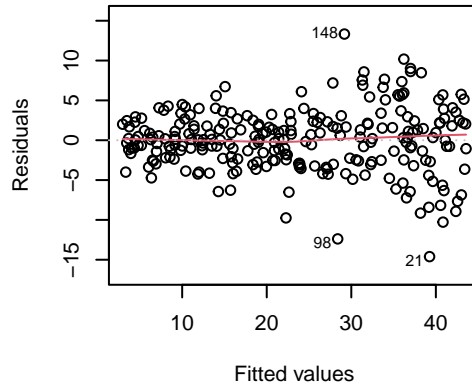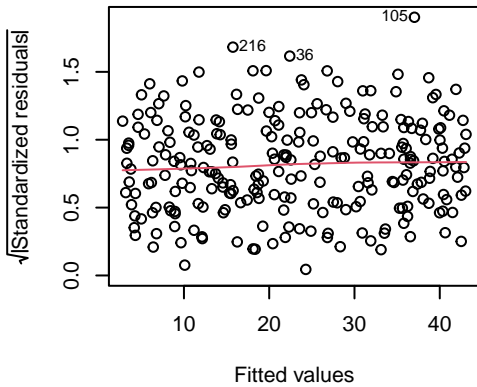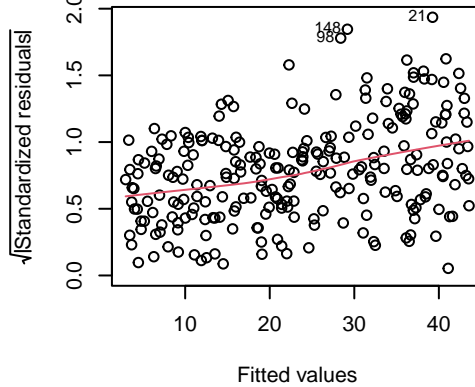
```
par(mfrow=c(1,1))
```

30.8 The Normality Assumption

If the plot is straight enough, the data are independent, and the plots don't thicken, you can now move on to the final assumption, that of Normality.

We assume that the errors around the idealized regression model at any specified values of the x-variables follow a Normal model. We need this assumption so that we can use a Student's t-model for inference. As with other times when we've used Student's t, we'll settle for the residuals satisfying the **Nearly Normal condition**. To assess this, we simply look at a histogram or Normal probability plot of the residuals. Note that the Normality Assumption also becomes less important as the sample size grows.

30.9 Outlier Diagnostics: Points with Unusual Residuals

A multiple regression model will always have a point which displays the most unusual residual (that is, the residual furthest from zero in either a positive or negative direction.) As part of our assessment of the normality assumption, we will often try to decide whether the residuals follow a Normal distribution by creating a Q-Q plot of standardized residuals.

30.9.1 Standardized Residuals

Standardized residuals are scaled to have mean zero and a constant standard deviation of 1, as a result of dividing the original (raw) residuals by an estimate of the standard deviation that uses all of the data in the data set.

The `augment` function, when applied to a linear regression model, will generate the standardized residuals and store them in the `.std.resid` column.

If multiple regression assumptions hold, then the standardized residuals (in addition to following a Normal distribution in general) will also have mean zero and standard deviation 1, with approximately 95% of values between -2 and +2, and approximately 99.74% of values between -3 and +3.

A natural check, therefore, will be to identify the most extreme/unusual standardized residual in our data set, and see whether its value is within the general bounds implied by the Empirical Rule associated with the Normal distribution. If, for instance, we see a standardized residual below -3 or above +3, this is likely to be a very poorly fit point (we would expect to see a point like this about 2.6 times for every 1,000 observations.)

A very poorly fitting point, especially if it also has high influence on the model (a concept we'll discuss shortly), may be sufficiently different from the rest of the points in the data set to merit its removal before modeling. If we did remove such a point, we'd have to have a good reason for this exclusion, and include that explanation in our report.

R's general plot set for a linear model includes (as plot 2) a Normal Q-Q plot of the standardized residuals from the model.

30.9.2 Checking the Normality Assumption with a Plot

```
plot(chol_m1, which=2)
```
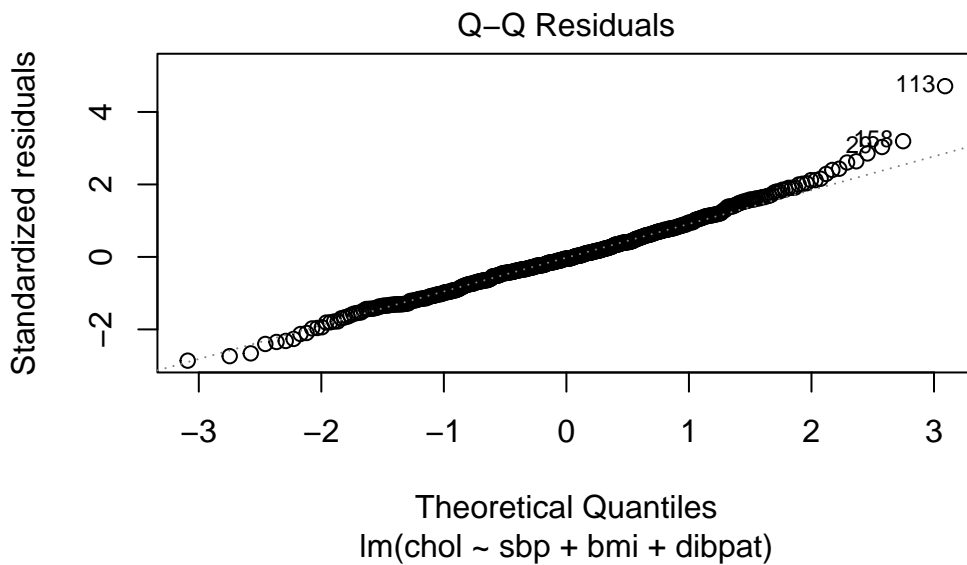

or we can use the `ggplot2` version:

```
p1 <- ggplot(aug_chol1, aes(sample = .std.resid)) +
  geom_qq() +
  geom_qq_line(col = "red") +
  labs(title = "Normal Q-Q plot",
       y = "Standardized Residual from chol_m1",
       x = "Standard Normal Quantiles") +
  theme(aspect.ratio = 1)
```

```
p2 <- ggplot(aug_chol1, aes(y = .std.resid, x = "")) +
  geom_violin(fill = "dodgerblue") +
  geom_boxplot(width = 0.3) +
  labs(title = "Box and Violin Plot",
       y = "Standardized Residual from chol_m1",
       x = "mod_1")

p1 + p2 +
  plot_layout(widths = c(2, 1)) +
  plot_annotation(
    title = "Normality of Standardized Residuals from mod_1",
    caption = paste0("n = ",
                     nrow(aug_chol1 |> select(.std.resid)),
                     " residual values are plotted here."))
```


Normality of Standardized Residuals from mod_1

The Q-Q plot of standardized residuals shows one point (in row 113, according to the plot) that is far away from our expectations under the Normal model. Let's look at that point a little more closely.

id	sbp	bmi	dibpat	chol	.fitted	.resid	.hat	.sigma	.cooksd	.std.resid
2280	120	25.86	Type A	414	227.61	186.39	0.01	38.8	0.03	4.71

```
aug_chol1 |> slice(113) |> select(id:.std.resid) |>
  kbl(digits = 2) |> kable_classic_2()
```

The standardized residual here is 4.71. That seems excessively large, in a sample of 250 observations.

30.9.3 Assessing Standardized Residuals with an Outlier Test

Is a standardized residual of 4.71 particularly unusual in a sample of 250 such standardized residuals, given that we are trying to assume that these are well modeled by a Normal distribution, supposedly with mean 0 and standard deviation 1?

Yes. The `car` library has a test called `outlierTest` which can be applied to a linear model to see if the most unusual studentized residual in absolute value is a surprise given the sample size (the studentized residual is very similar to the standardized residual - it simply uses a different estimate for the standard deviation for every point, in each case excluding the point it is currently studying)

```
outlierTest(chol_m1)
```

```
    rstudent unadjusted p-value Bonferroni p
113 4.816993         1.9404e-06   0.00097019
```

Our conclusion from the Bonferroni p value here is that there's evidence to support the belief that we could have a serious problem with Normality, in that a studentized residual of 4.82 is out of the range we might reasonably expect to see given a Normal distribution of errors and 250 observations.

Which value is that? It's the subject with `id` = 2280.

```
aug_chol1 |> slice(113) |>
    select(id:.std.resid) |> kbl(digits = 2) |> kable_classic_2()
```

In terms of solutions, we could consider dropping the point, but this would leave us with the vexing problem of how, exactly, to justify that choice. One question we might ask is whether dropping that point (row 113, or `id` = 2280) from our `chol_m1` model have a big impact?

30.10 Outlier Diagnostics: Identifying Points with Unusually High Leverage

An observation can be an outlier not just in terms of having an unusual residual, but also in terms of having an unusual combination of predictor values. Such an observation is described as having high leverage in a data set.

- The `augment` function stores the leverage for each point in the `.hat` variable.
- The average leverage value is equal to k/n, where n is the number of observations included in the regression model, and k is the number of coefficients (slopes + intercept).
- Any point with a leverage value greater than 3 times the average leverage of a point in the data set is one that should be investigated closely.

For instance, in the `wcgs_500` data, we have 500 observations, so the average leverage will be $4/500 = 0.008$ and a high leverage point would have leverage $\geq 12/500$ or .024. We can check this out directly, with a sorted list of leverage values.

```
aug_chol1 |> select(id:chol, .hat) |>
    filter(.hat > 0.024) |>
    arrange(desc(.hat))
```

```
# A tibble: 12 x 6
      id   sbp   bmi dibpat  chol   .hat
   <dbl> <dbl> <dbl> <chr>  <dbl>  <dbl>
 1 19078   230  24.4 Type B   254 0.0884
 2 10214   170  37.7 Type B   153 0.0600
 3 22025   210  26.5 Type B   187 0.0548
 4 10075   150  35.3 Type B   224 0.0401
 5 16031   200  28.1 Type A   256 0.0392
 6 11432   196  27.3 Type B   178 0.0384
 7  3085   190  22.3 Type A   241 0.0383
 8 13389   120  33.0 Type A   232 0.0303
 9 12976   106  15.7 Type B   207 0.0287
10  3182   120  32.3 Type A   218 0.0262
11 19099   104  30.5 Type A   234 0.0252
12  3235   184  26.6 Type A   274 0.0248
```

Twelve of our 500 subjects meet the standard for large leverage values (`.hat` values exceeding three times the average `.hat` value) but none of these are the outlier (`id = 2280`) we identified as having an especially poor fit (large standardized residual.)

```
aug_chol1 |>
    filter(id == 2280) |>
    select(id, .std.resid, .hat)
```

```
# A tibble: 1 x 3
     id .std.resid      .hat
  <dbl>      <dbl>    <dbl>
1  2280       4.71 0.00555
```

Another way to understand the impact of leverage is to look at a plot of residuals vs. leverage, which is diagnostic plot 5 for a linear model. Here's the base R version for our `chol_m1` model.

```
plot(chol_m1, which=5)
```
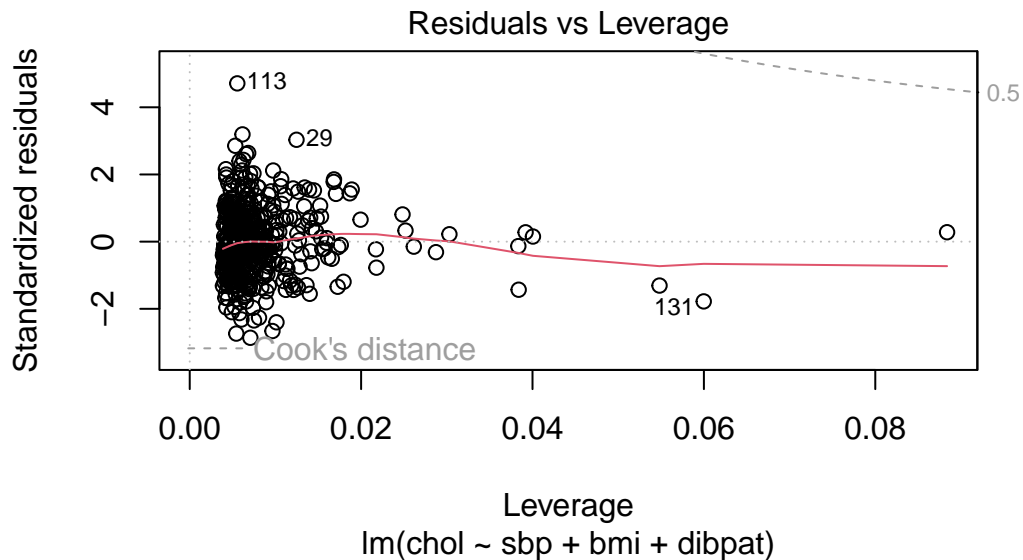

Here is the ggplot2 version.

```
ggplot(aug_chol1, aes(x = .hat, y = .std.resid)) +
    geom_point() +
    geom_point(data = aug_chol1 |> filter(.cooksd >= 0.5),
```

```
                col = "red", size = 2) +
    geom_text_repel(data = aug_chol1 |> filter(.cooksd >= 0.5),
                aes(label = id), col = "red") +
    geom_smooth(method = "loess", formula = y ~ x, span = 2, se = F) +
    geom_vline(aes(xintercept = 3*mean(.hat)), lty = "dashed") +
    labs(title = "Residuals vs. Leverage",
        x = "Leverage", y = "Standardized Residual")
```


Residuals vs. Leverage
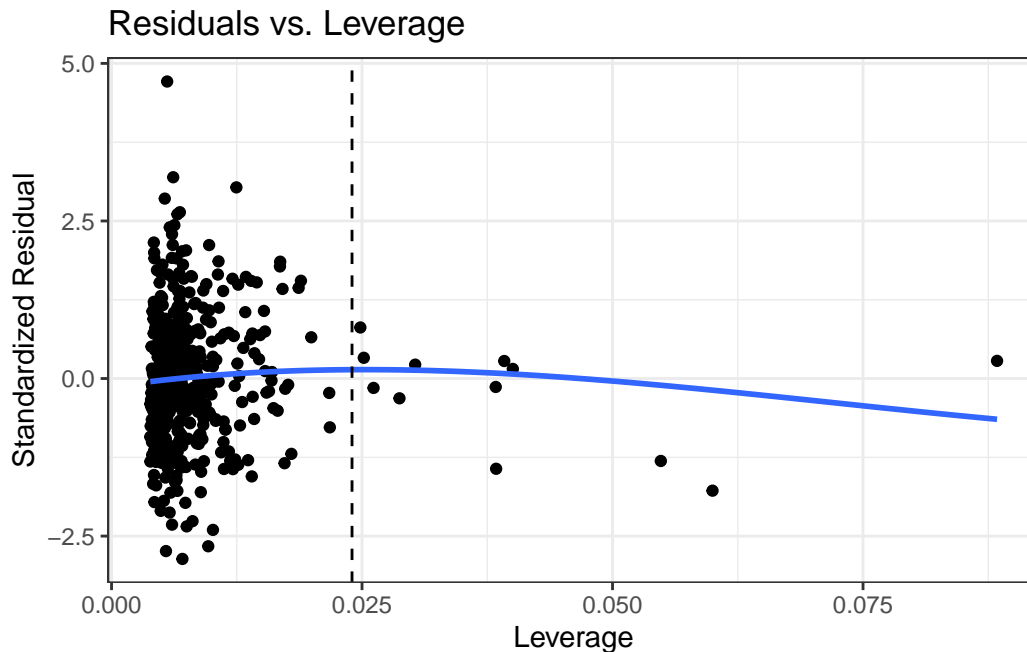
We see that the most highly leveraged points are shown furthest to the right in this plot (in fact, I've inserted a dotted vertical line at the cutpoint of `3*mean(.hat)`), but our problematic standardized residual has a very typical leverage value (at the top left of this plot.)

Let's look at our highest leveraged point ($id = 19078$ from our list above) and remember that this just means that this point is unusual in terms of its predictor values.

```
  aug_chol1 |> filter(id == 19078) |>
      select(id, sbp, bmi, dibpat, .hat)
```

```
# A tibble: 1 x 5
     id   sbp   bmi dibpat   .hat
  <dbl> <dbl> <dbl> <chr>   <dbl>
1 19078   230  24.4 Type B 0.0884
```

Compare these findings to what we see in the `wcgs_500` data overall. Let's focus on the quantitative variables involved in our model, `sbp` and `bmi`, and use the `df_stats()` function from the `mosaic` package.

```
df_stats(~ sbp + bmi, data = wcgs_500)
```

```
  response      min        Q1   median       Q3       max      mean        sd
1      sbp 100.0000 120.00000 126.0000 140.0000 230.00000 129.39800 16.548453
2      bmi  15.6605  23.03123  24.5412  26.1554  37.65281  24.65342  2.546486
    n missing
1 500       0
2 500       0
```

Note that our highly leveraged point has the largest `sbp` of all 500 subjects, although a rather typical `bmi`.

If you're wondering about the rest of our `chol_m1` model, there are only two possible values of `dibpat` and so a point cannot really be an outlier on that variable.

```
wcgs_500 |> tabyl(dibpat)
```

```
 dibpat   n percent
 Type A 259   0.518
 Type B 241   0.482
```

Remember that 12 of the points in our `chol_m1` model showed up with leverage values more than three times higher than the mean leverage.

```
aug_chol1 |>
    filter(.hat > 3 * mean(.hat)) |> nrow()
```

```
[1] 12
```

So, let's look at a plot of the two quantitative predictors in our data to see which 12 points show up as those with "high" leverage.

```
ggplot(aug_chol1, aes(x = sbp, y = bmi)) +
    geom_point(col = "gray50") +
    geom_point(data = aug_chol1 |>
                   slice_max(.hat, n = 12),
```

```
              col = "red", size = 2) +
     labs(x = "Systolic BP", y = "Body Mass Index",
          title = "Where are the highly leveraged points in our model?",
          subtitle = "wcgs_500 model chol_1, 12 highest leverage points in red")
```


Where are the highly leveraged points in our model?
wcgs_500 model chol_1, 12 highest leverage points in red
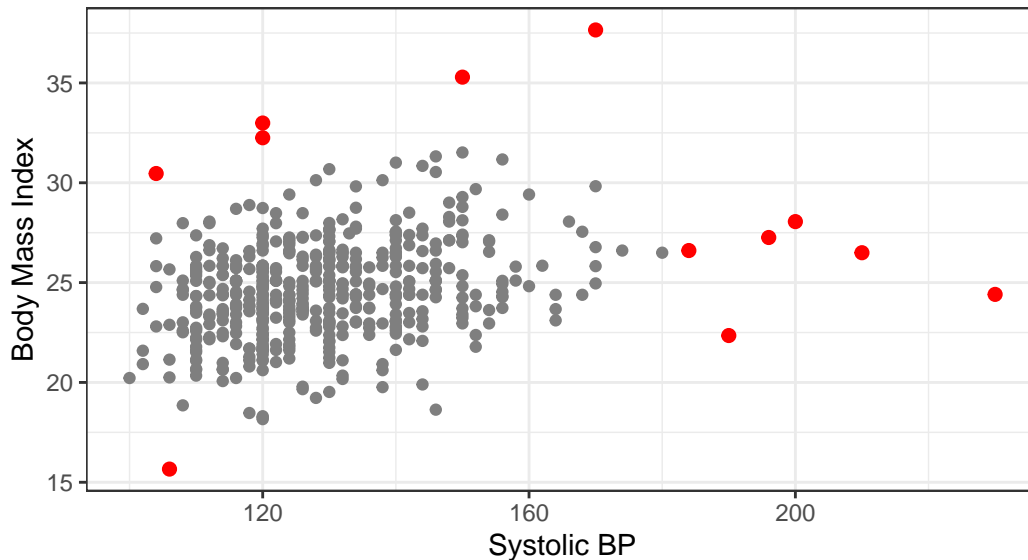
The points with high leverage (large values of `.hat`) are simply those with an unusual combination of the two quantitative predictors.

30.11 Outlier Diagnostics: Identifying Points with High Influence on the Model

A point in a regression model has high **influence** if the inclusion (or exclusion) of that point will have substantial impact on the model, in terms of the estimated coefficients, and the summaries of fit. The measure I routinely use to describe the level of influence a point has on the model is called **Cook's distance**.

As a general rule, a Cook's distance greater than 1 indicates a point likely to have substantial influence on the model, while a point in the 0.5 to 1.0 range is only occasionally worthy of investigation. Observations with Cook's distance below 0.5 are unlikely to influence the model in any substantial way.

30.11.1 Assessing the Value of Cook's Distance

You can obtain information on Cook's distance in several ways in R.

- My favorite is model diagnostic plot 5 (the residuals vs. leverage plot) which we've seen before and which I repeat below. This plot uses red contours to indicate the value of Cook's distance.

   - This is possible because influence, as measured by Cook's distance, is a function of both the observation's standardized residual and its leverage.

```
plot(chol_m1, which=5)
```
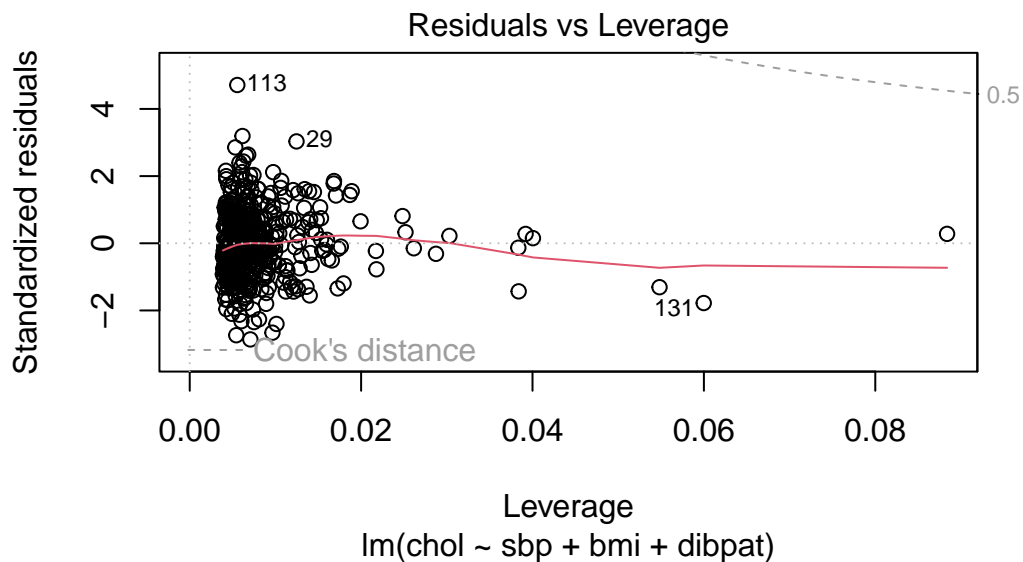

No points in this plot indicate substantial influence on our model. To do so, they'd have to be outside the Cook's distance contour shown at the top right of this plot (there is a similar arc at the bottom right, but it's too far away from the data to show up in the plot.)

30.11.2 Index Plot of Cook's Distance

Model Diagnostic Plot 4 for a linear model is an index plot of Cook's distance.

```
plot(chol_m1, which=4)
```

Cook's distance
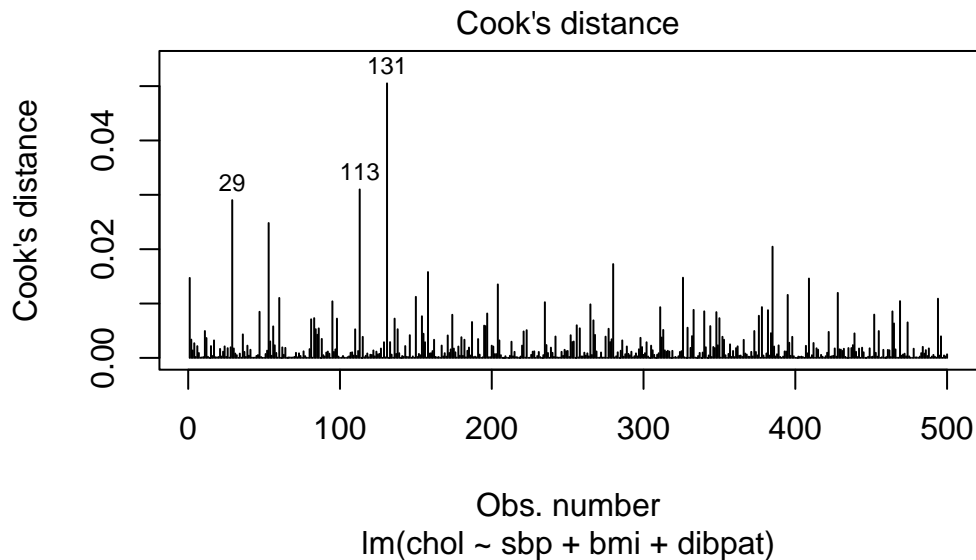
Or, if you like, run the ggplot2 version:

```
aug_chol1_ex <- aug_chol1 |>
  mutate(obsnum = 1:nrow(aug_chol1 |> select(.cooksd)))

ggplot(aug_chol1_ex, aes(x = obsnum, y = .cooksd)) +
    geom_point() +
    geom_segment(aes(x = obsnum, xend = obsnum, y = 0, yend = .cooksd)) +
    geom_text_repel(data = aug_chol1_ex |>
                slice_max(.cooksd, n = 3),
                aes(label = id)) +
    labs(x = "Row Number",
        y = "Cook's Distance",
        title = "Cook's distance Index plot for chol_m1",
        subtitle = "Subjects with the 3 largest Cook's d values are identified.")
```

id	.std.resid	.hat	.cooksd
10214	-1.779	0.060	0.051
2280	4.713	0.006	0.031
3604	3.033	0.012	0.029
22025	-1.308	0.055	0.025
11432	-1.432	0.038	0.020
12392	-2.661	0.010	0.017

Cook's distance Index plot for chol_m1
Subjects with the 3 largest Cook's d values are identified.
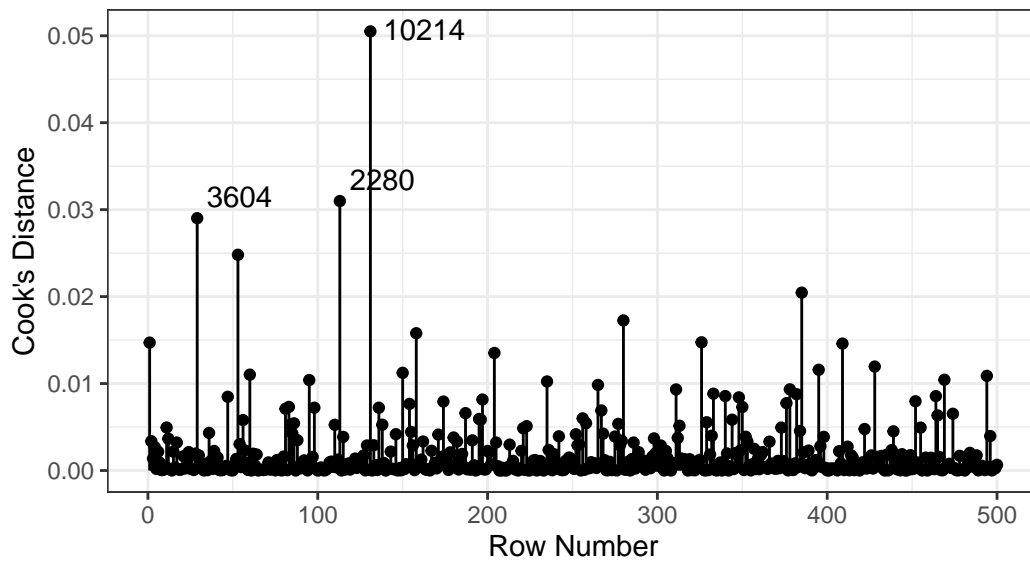
It is clear from this plot that the largest Cook's distance (somewhere between 0.05 and 0.06) is for the observation in subject 10214 (or row 131 of the data set, depending on which version you build. All of the Cook's distance values are stored in the `.cooksd` variable in our augmented results. Let's look at the six largest Cook's distance values.

```
aug_chol1 |> select(id, .std.resid, .hat, .cooksd) |>
    arrange(desc(.cooksd)) |> head() |> kbl(digits = 3) |> kable_classic()
```

Remember that we'd need a Cook's distance value to be at least 0.50 for us to worry about it in a serious way. Here, the largest we see (0.05 for `id = 10214`) is still far away from that standard.

What is the story on point 10214? Why is it unusual?

id	sbp	bmi	dibpat	chol	.fitted	.std.resid	.hat	.cooksd
10214	170	37.65	Type B	153	221.42	-1.78	0.06	0.05

```
aug_chol1 |> filter(id == 10214) |>
    select(id:.fitted, .std.resid, .hat, .cooksd) |>
    kbl(digits = 2) |> kable_classic()
```

This turns out to be the subject with the largest BMI in the data, hence some leverage, and a somewhat large (-1.78) but not enormous standardized residual. Hence the combination suggests some influence on the model.

30.12 Running a Regression Model While Excluding A Point

Suppose that we wanted to remove the point with the most influence over the model. We could fit a model to the data without $id = 10214$ to see the impact of this change. Let's try it.

```
summary(lm(chol ~ sbp + bmi + dibpat, data = wcgs_500 |> filter(id != 10214)))
```

```
Call:
lm(formula = chol ~ sbp + bmi + dibpat, data = filter(wcgs_500,
    id != 10214))

Residuals:
     Min      1Q   Median      3Q      Max
-113.062  -25.453   -2.469   23.954  186.200

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   206.3207    20.3647   10.131  < 2e-16 ***
sbp             0.2464     0.1124    2.192  0.02883 *
bmi            -0.3127     0.7438   -0.420  0.67433
dibpatType B  -10.6424     3.5635   -2.986  0.00296 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 39.57 on 495 degrees of freedom
Multiple R-squared:  0.02943,    Adjusted R-squared:  0.02355
F-statistic: 5.003 on 3 and 495 DF,  p-value: 0.001996
```

Compare these results to those obtained with the full data set, shown below.

- How is the model affected by removing subject 10214?
- What is the impact on the slopes?
- On the summary measures? Residuals?

```
summary(lm(chol ~ sbp + bmi + dibpat, data = wcgs_500))
```

```
Call:
lm(formula = chol ~ sbp + bmi + dibpat, data = wcgs_500)

Residuals:
     Min       1Q   Median       3Q      Max
-113.133  -25.689   -1.932   24.036  186.390

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   214.5185    19.8825  10.789  < 2e-16 ***
sbp             0.2361     0.1125   2.099  0.03634 *
bmi            -0.5894     0.7290  -0.808  0.41921
dibpatType B  -11.0408     3.5643  -3.098  0.00206 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 39.66 on 496 degrees of freedom
Multiple R-squared:  0.02954,    Adjusted R-squared:  0.02367
F-statistic: 5.033 on 3 and 496 DF,  p-value: 0.001915
```

While it is true that we can sometimes improve the performance of the model in some ways by removing this point, there's no good reason to do so. We can't just remove a point from the data set without a good reason (and, to be clear, "I ran my model and it doesn't fit this point well" is NOT a good reason). Good reasons would include:

- This observation was included in the sample in error, for instance, because the subject was not eligible.
- An error was made in transcribing the value of this observation to the final data set.
- And, sometimes, even "This observation is part of a meaningful subgroup of patients that I had always intended to model separately…" assuming that's true.

30.13 Summarizing Regression Diagnostics for 431

1. Check the "straight enough" condition with scatterplots of the y variable (outcome) against each x-variable (predictor), usually via the top row of a scatterplot matrix.

2. If the data are straight enough (that is, if it looks like the regression model is plausible), fit a regression model, to obtain residuals and influence measures.

   - If not, consider using the Box-Cox approach to identify a possible transformation for the outcome variable, and then recheck the straight enough condition.

The `plot` function for a fitted linear model builds five diagnostic plots.

3. [**Plot 1**] A scatterplot of the residuals against the fitted values.

   - This plot should look patternless. Check in particular for any bend (which would suggest that the data weren't all that straight after all) and for any thickening, which would indicate non-constant variance.
   - If the data are measured over time, check especially for evidence of patterns that might suggest they are not independent. For example, plot the residuals against time to look for patterns.
   - These values are stored as `.resid` and `.fitted` by the `augment()` function applied to a linear model.

4. [**Plot 3**] A scale-location plot of the square root of the standardized residuals against the fitted values to look for a non-flat loess smooth, which indicates non-constant variance. Standardized residuals are obtained via `augment` and stored in the `.std.resid` variable.

5. [**Plot 2**] If the plots above look OK, then consider a Normal Q-Q plot of the standardized residuals to check the nearly Normal condition.

6. [**Plot 5**] The final plot we often look at is the plot of residuals vs. leverage, with influence contours. Sometimes, we'll also look at [**Plot 4**] the index plot of the Cook's distance for each observation in the data set.

   - To look for points with substantial **leverage** on the model by virtue of having unusual values of the predictors - look for points whose leverage is at least 3 times as large as the average leverage value.
     - The average leverage is always k/n, where k is the number of coefficients fit by the model (including the slopes and intercept), and n is the number of observations in the model.
     - To obtain the leverage values, use the `augment()` function which stores them in `.hat`.

- To look for points with substantial **influence** on the model, that is, removing them from the model would change it substantially, consider the Cook's distance, plotted in contours in Plot 5, or in an index plot in Plot 4.
  - Any Cook's d > 1 will likely have a substantial impact on the model.
  - Even points with Cook's d > 0.5 may merit further investigation.
  - Find the Cook's distances using the `augment()` function, which stores them in `.cooksd`.

30.14 Violated Assumptions: Problems with Linearity

So what do serious assumption violations look like, and what can we do about them?

Here is a simulated example that shows a clear problem with non-linearity.

```
set.seed(4311); x1 <- rnorm(n = 100, mean = 15, sd = 5)
set.seed(4312); x2 <- rnorm(n = 100, mean = 10, sd = 5)
set.seed(4313); e1 <- rnorm(n = 100, mean = 0, sd = 15)
y <- 15 + x1 + x2^2 + e1
viol1 <- data.frame(outcome = y, pred1 = x1, pred2 = x2) |> tibble()
model_1 <- lm(outcome ~ pred1 + pred2, data = viol1)
plot(model_1, which = 1)
```


585
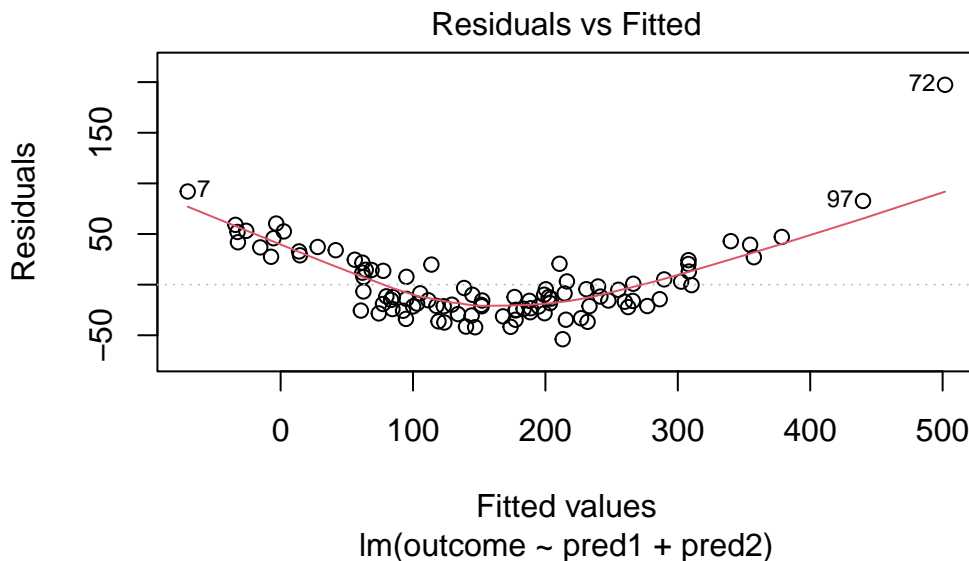
In light of this, I would be looking for a potential transformation of outcome. Does the Box-Cox plot make any useful suggestions?

```
boxCox(model_1); powerTransform(model_1)
```


Profile Log–likelihood
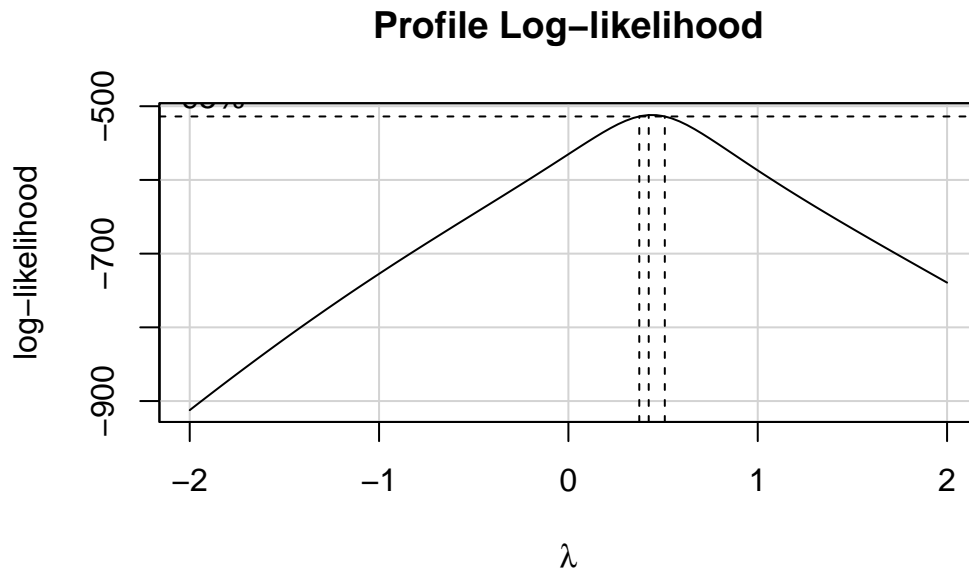
```
Estimated transformation parameter
       Y1
0.4414775
```

Note that if the outcome was negative, we would have to add some constant value to every outcome in order to get every outcome value to be positive, and Box-Cox to run. This suggests fitting a new model, using the square root of the outcome.

```
model_2 <- lm(sqrt(outcome) ~ pred1 + pred2, data = viol1)
plot(model_2, which = 1)
```

Residuals vs Fitted

Im(sqrt(outcome) ~ pred1 + pred2)
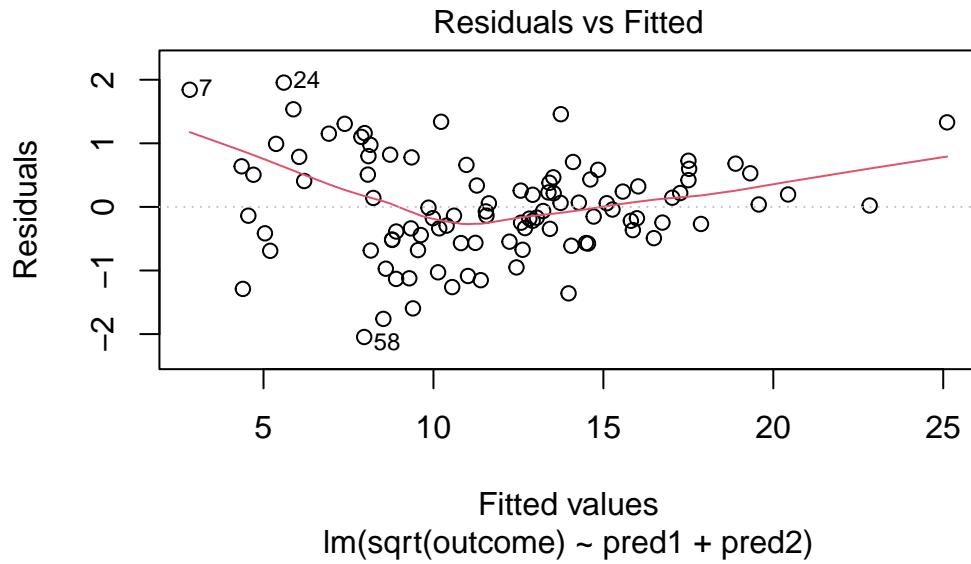
This is meaningfully better in terms of curve, but now looks a bit fan-shaped, indicating a potential problem with heteroscedasticity. Let's look at the scale-location plot for this model.

```
plot(model_2, which = 3)
```

Scale–Location

lm(sqrt(outcome) ~ pred1 + pred2)

Fitted values
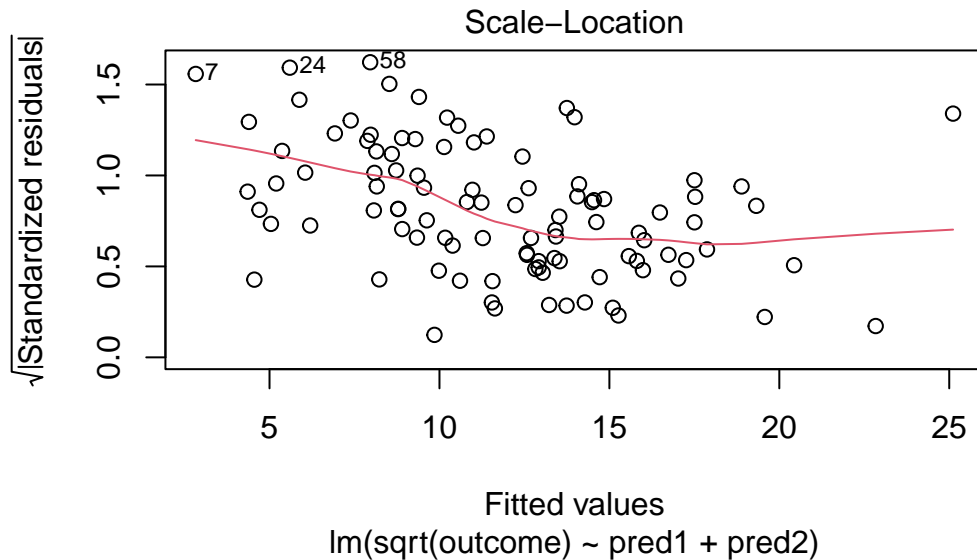
This definitely looks like there's a trend down in this plot. So the square root transformation, by itself, probably hasn't resolved assumptions sufficiently well. We'll have to be very careful about our interpretation of the model.

30.15 Problems with Non-Normality: An Influential Point

With 100 observations, a single value with a standardized residual above 3 is very surprising. In our initial model_1 here, we have a standardized residual value as large as 6, so we clearly have a problem with that outlier.

```
plot(model_1, which = 2)
```

Q–Q Residuals
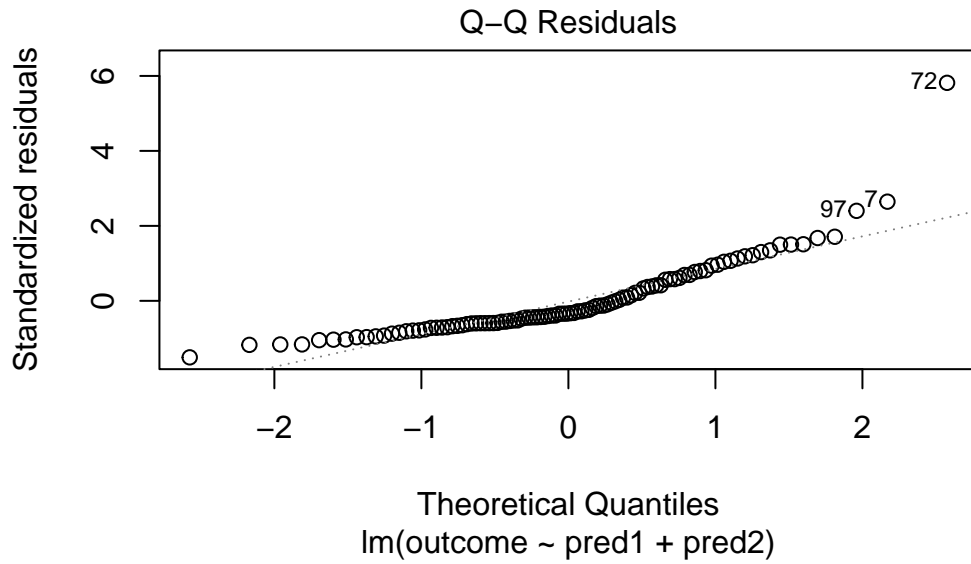
Should we, perhaps, remove point 72, and try again? Only if we have a reason beyond "it was poorly fit" to drop that point. Is point 72 highly leveraged or influential?

```
plot(model_1, which = 5)
```

Residuals vs Leverage

lm(outcome ~ pred1 + pred2)
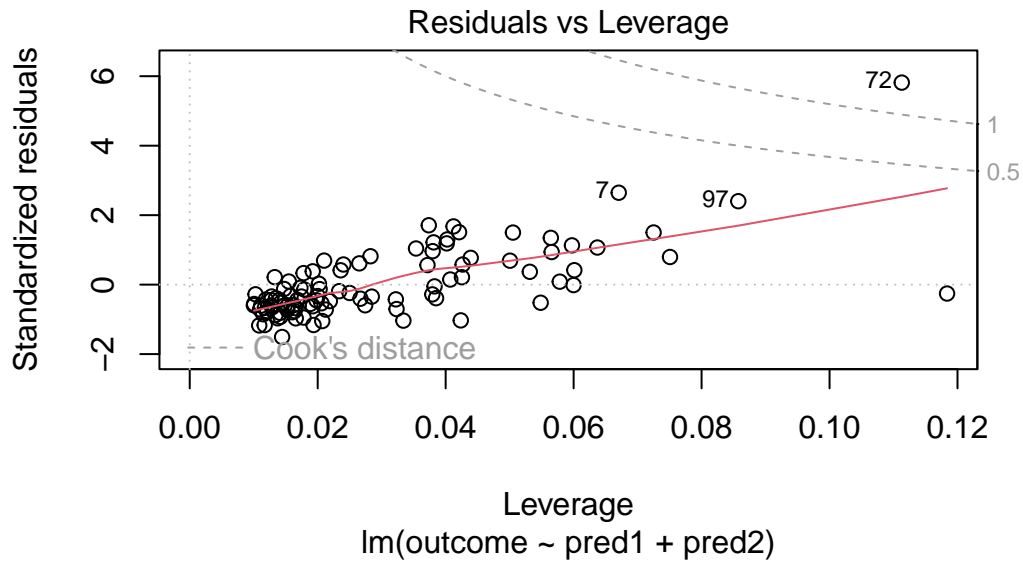
What if we drop this point (72) and fit our linear model again. Does this resolve our difficulty with the assumption of linearity?

```r
model_1_no72 <- lm(outcome ~ pred1 + pred2, data = viol1[-72,])
par(mfrow=c(1,2))
plot(model_1, which = 1, caption = "With Point 72")
plot(model_1_no72, which = 1, caption = "Without Point 72")
```
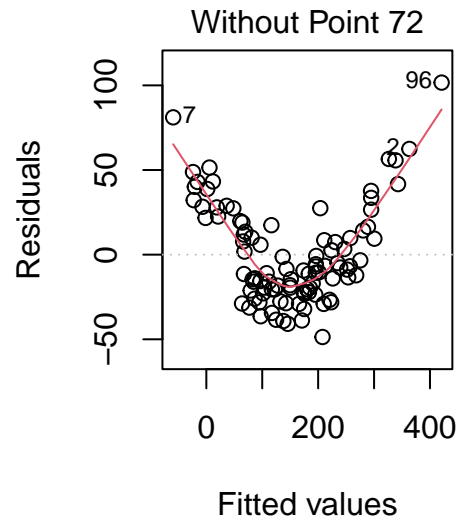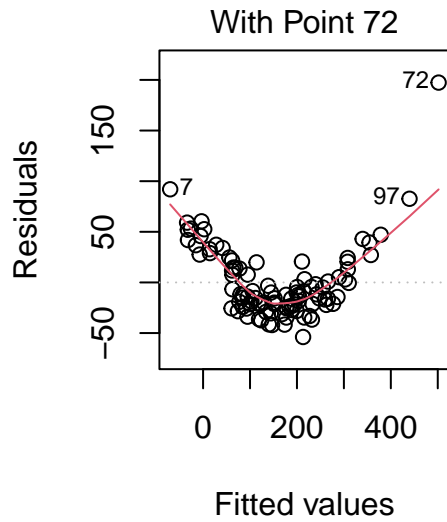
```
par(mfrow=c(1,1))
```

No, it doesn't. But what if we combine our outcome transformation with dropping point 72?

```
model_2_no72 <- lm(sqrt(outcome) ~ pred1 + pred2, data = viol1[-72,])
par(mfrow=c(1,2))
plot(model_2_no72, which = c(1,3))
```
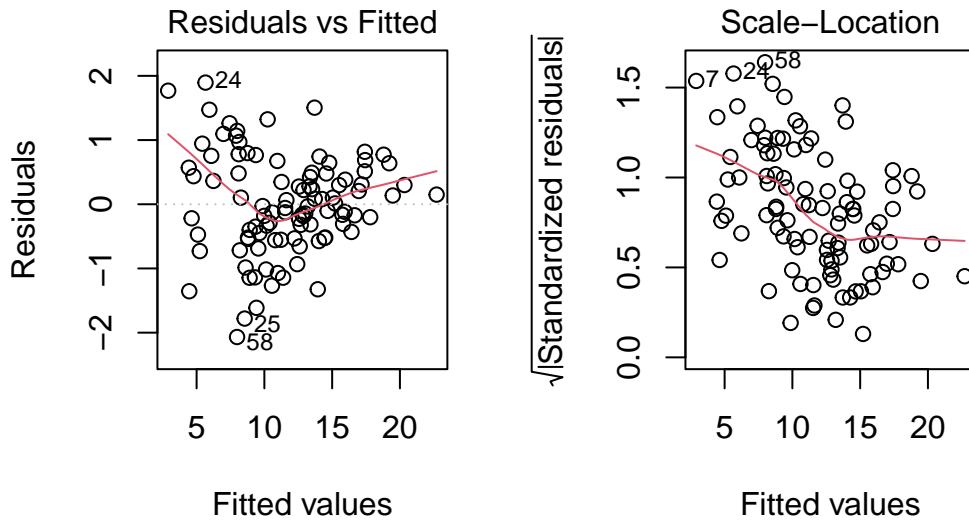
```
par(mfrow=c(1,1))
```

Nope. That still doesn't alleviate the problem of heteroscedasticity very well. At least, we no longer have any especially influential points, nor do we have substantial non-Normality.

```
par(mfrow=c(1,2))
plot(model_2_no72, which = c(2,5))
```
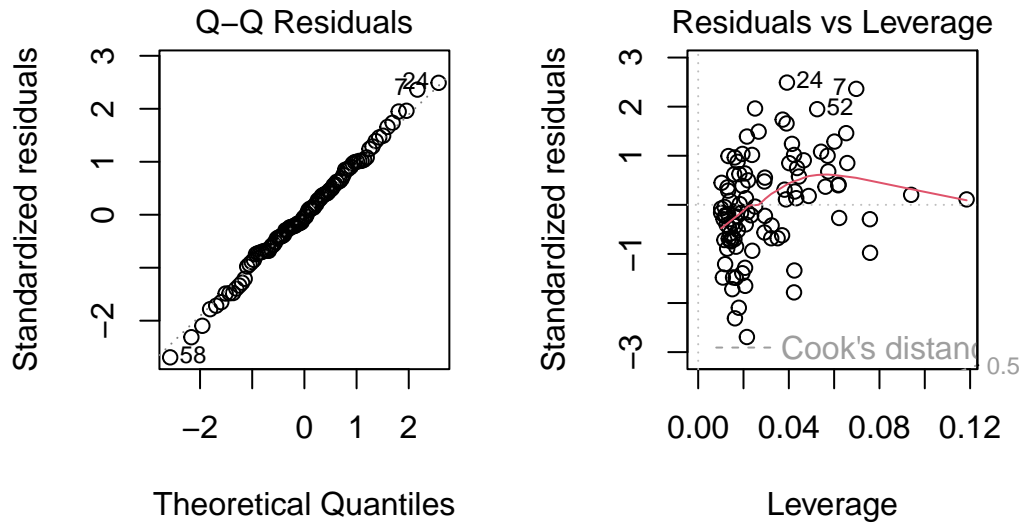
```
par(mfrow=c(1,1))
```

At this point, I would be considering potential transformations of the predictors, quite possibly fitting some sort of polynomial term or cubic spline term in the predictors, but I'll leave that for discussion in 432.

30.16 Problems with Non-Normality: Skew

```
set.seed(4314); x1 <- rnorm(n = 100, mean = 15, sd = 5)
set.seed(4315); x2 <- rnorm(n = 100, mean = 10, sd = 5)
set.seed(4316); e2 <- rnorm(n = 100, mean = 3, sd = 5)
y2 <- 50 + x1 + x2 + e2^2
viol2 <- data.frame(outcome = y2, pred1 = x1, pred2 = x2) |> tibble()
model.3 <- lm(outcome ~ pred1 + pred2, data = viol2)
plot(model.3, which = 2)
```
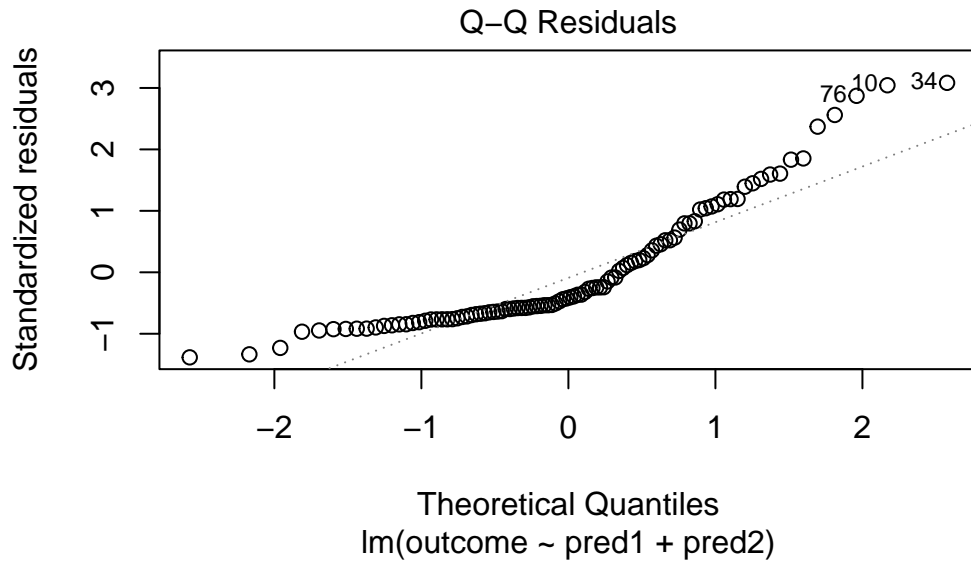
Q–Q Residuals

Skewed residuals often show up in strange patterns in the plot of residuals vs. fitted values, too, as in this case.

```
plot(model.3, which = 1)
```

Residuals vs Fitted
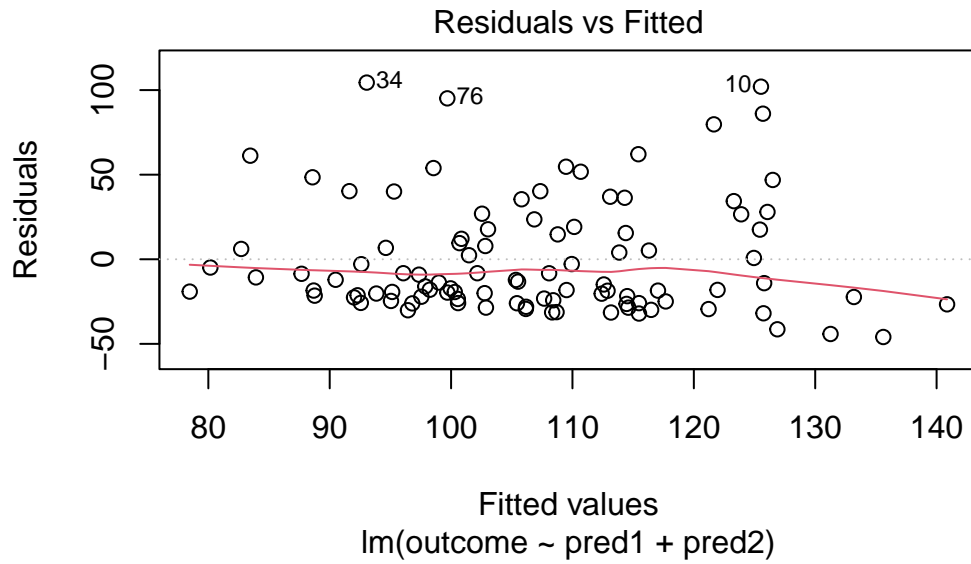
Clearly, we have some larger residuals on the positive side, but not on the negative side. Would an outcome transformation be suggested by Box-Cox?

```
boxCox(model.3); powerTransform(model.3)
```

Profile Log–likelihood
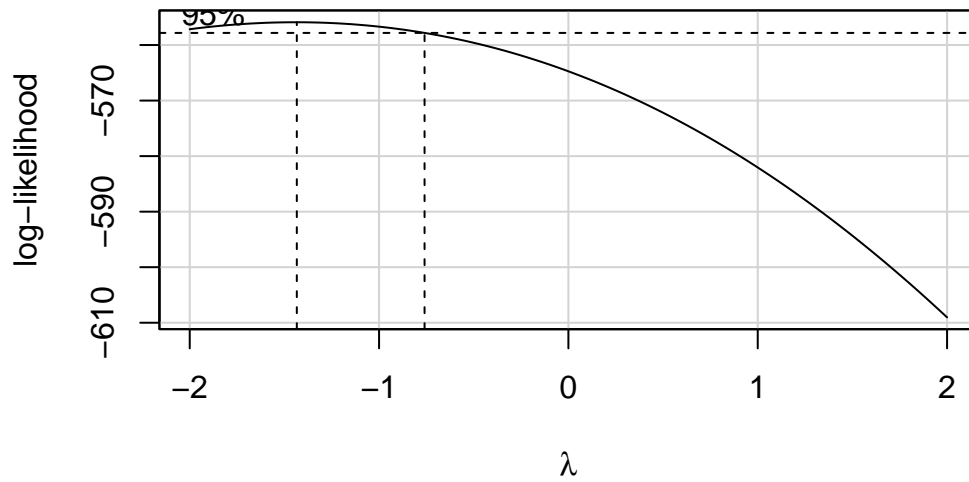

```
Estimated transformation parameter
        Y1
-1.438747
```

The suggested transformation looks like the inverse of our outcome.

```r
model.4 <- lm(1/outcome ~ pred1 + pred2, data = viol2)
plot(model.4, which = 2)
```

Q–Q Residuals

Standardized residuals

Theoretical Quantiles
lm(1/outcome ~ pred1 + pred2)
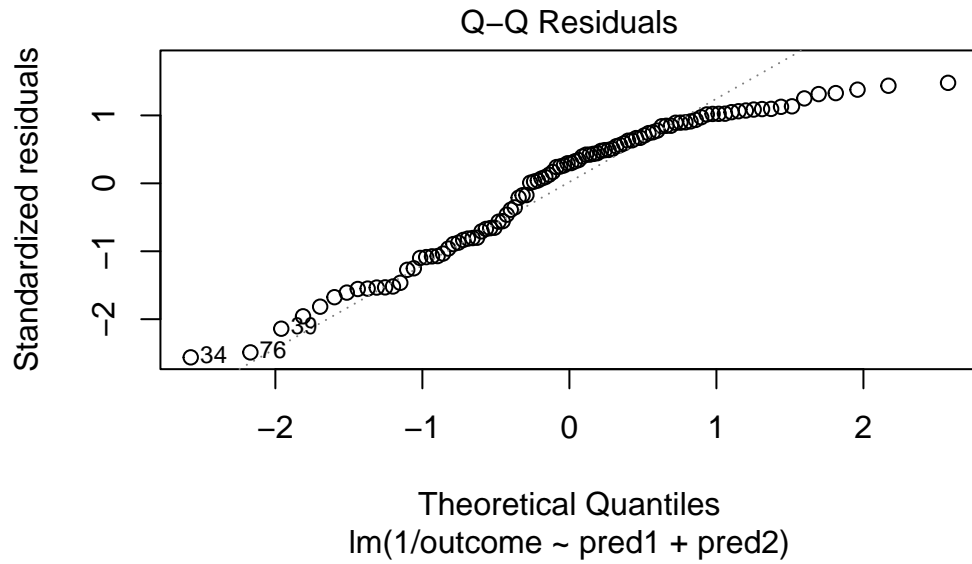
OK. That's something of an improvement. How about the other residual plots with this transformation?

```
plot(model.4, which = 1)
```

Residuals vs Fitted

Fitted values
lm(1/outcome ~ pred1 + pred2)
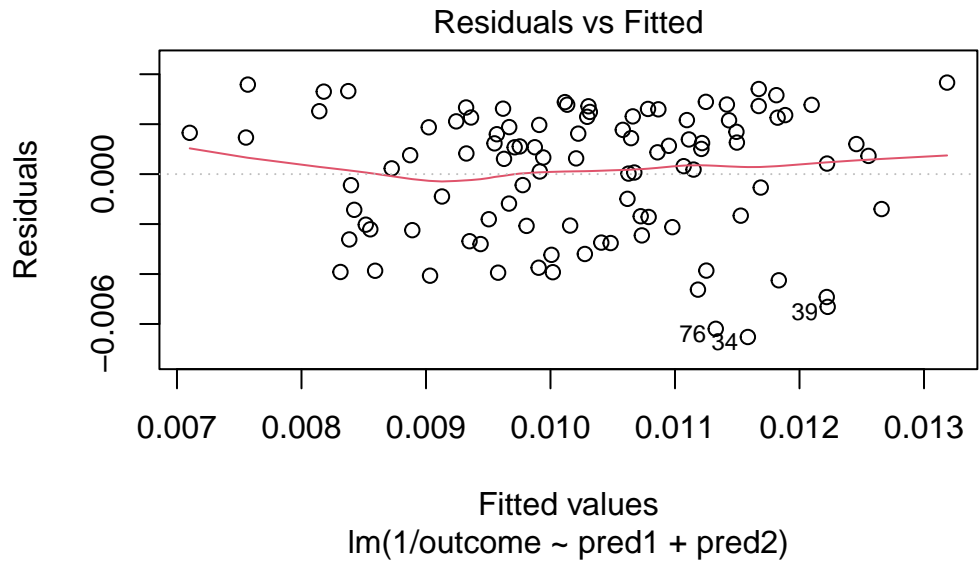
The impact of the skew is reduced, at least. I might well be satisfied enough with this, in practice.

31 Building Prediction Models for `wcgs`

Sometimes, we use a regression model for description of a multivariate relationship which requires only that we provide an adequate fit to the data at hand. Should we want to use the model for prediction, then a more appropriate standard which requires us to fit a model that does each of the following three things:

1. [fits well] Provides an adequate fit to the data at hand
2. [parsimonious] Includes only those predictors which actually have detectable predictive value
3. [predicts well out of sample] Does a better job predicting our outcome in new data than other alternative models

For now, we'll focus on a pair of issues:

a. Given a set of predictors, how should we let the computer help us understand which subsets of those predictors are most worthy of our additional attention?
b. Given a pair of models to compare, what tools should we use to determine which one better predicts new data?

31.1 Setup: Packages Used Here

```
knitr::opts_chunk$set(comment = NA)

library(broom)
library(car)
library(GGally)
library(ggrepel)
library(janitor)
library(kableExtra)
library(patchwork)
library(tidyverse)

theme_set(theme_bw())
```

We'll also use some functions from the `Hmisc` and `rms` packages.

31.2 Predicting Cholesterol Level in WCGS, Again

To address these issues, I'll again look at the `wcgs` data (Western Collaborative Group Study), described in Chapter 17, then again in Chapter 30.

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

This time, we'll try to predict the variable `chol` on the basis of some subset of the following six predictors: `age`, `bmi`, `sbp`, `dbp`, `smoke` and `dibpat`.

The steps we'll take are as follows.

1. Check the `wcgs` data for missing or out-of-range values in the variables under study, so we don't regret it later. Make a decision about how to handle issues here.
2. Partition the `wcgs` data into a training (development) sample of 2000 observations, and a test (holdout) sample of the remaining observations.
3. Using only the model development sample, fit three candidate models.

   - Model A will predict `chol` using all six predictors.
   - Model B will predict `chol` using five predictors, specifically `age`, `bmi`, `dbp`, `dibpat` and `smoke`.
   - Model C will predict `chol` using only three predictors, specifically `age`, `dbp` and `smoke`

4. Compare the fit quality of these models in the development sample to see if any of them is superior in terms of in-sample predictive quality.
5. Assess regression assumptions in each of these models in the development sample.
6. Finally moving to the holdout sample, compare the quality of predictions made by the models in terms of several criteria to see if any of the models (A, B or C) is clearly superior in terms of out-of-sample prediction.

31.3 Checking for Missing or Problematic Values

Suppose that after consulting with clinical experts, we want to ensure that:

- all `age` values are between 39 and 59 years
- all `bmi` are between 15 and 50
- all `sbp` are between 80 and 250 mm Hg
- all `dbp` are between 50 and 200 mm Hg
- all values of `sbp-dbp` are at least 10 and no more than 90 mm Hg
- all values of `chol` are between 100 and 400 mg/dl

```
  Hmisc::describe(wcgs |> select(age, bmi, sbp, dbp, smoke, dibpat, chol))
```

select(wcgs, age, bmi, sbp, dbp, smoke, dibpat, chol)

 7 Variables   3154 Observations
----
age
       n missing distinct   Info   Mean   Gmd   .05   .10
    3154   0   21   0.996   46.28   6.256   39   40
     .25   .50   .75   .90   .95
      42   45   50   55   57

lowest : 39 40 41 42 43, highest: 55 56 57 58 59
----
bmi
       n missing distinct   Info   Mean   Gmd   .05   .10
    3154   0   679   1   24.52   2.803   20.59   21.52
     .25   .50   .75   .90   .95
   22.96   24.39   25.84   27.45   28.73

lowest : 11.1906 15.6605 16.872 17.2163 17.2224
highest: 36.0425 37.2297 37.2481 37.6528 38.9474
----
sbp
       n missing distinct   Info   Mean   Gmd   .05   .10
    3154   0   62   0.996   128.6   16.25   110   112
     .25   .50   .75   .90   .95
     120   126   136   148   156

lowest : 98 100 102 104 106, highest: 200 208 210 212 230
----
dbp
       n missing distinct   Info   Mean   Gmd   .05   .10
    3154   0   42   0.992   82.02   10.51   68   70
     .25   .50   .75   .90   .95
      76   80   86   94   100

lowest : 58 60 62 64 66, highest: 125 129 130 136 150
----
smoke
       n missing distinct
    3154   0   2
```

```
Value           No   Yes
Frequency     1652  1502
Proportion   0.524 0.476
--------------------------------------------------------------------------------
dibpat
        n  missing distinct
     3154        0        2

Value         Type A Type B
Frequency       1589   1565
Proportion     0.504  0.496
--------------------------------------------------------------------------------
chol
        n  missing distinct     Info     Mean      Gmd      .05      .10
     3142       12      237        1    226.4    47.99    161.1    175.0
      .25      .50      .75      .90      .95
    197.2    223.0    253.0    280.0    302.0

lowest : 103 110 111 112 113, highest: 386 390 400 414 645
--------------------------------------------------------------------------------
```

Here are the issues I see:

- The (chol) total cholesterol levels include 12 missing values (so we need to decide what to do about them) and values of 414 and 645 that I don't think are plausible. Since chol is our outcome, I'm inclined to delete those 14 subjects from our analyses.
- The bmi value of 11.9 is much smaller than all other values, and seems implausible, so I'm inclined to delete that subject, as well.
- All of the age values are between 39 and 59, as they should be.
- The dibpat and smoke binary variables each appear to be reasonable given the timing of the data collection (the early 1960s).
- The blood pressures, individually, for sbp and dbp appear reasonable in terms of their ranges and have no missing data. It's worth it to check to see that everyone has a sbp meaningfully larger than their dbp.

```
wcgs |> mutate(bp_diff = sbp - dbp) |>
    select(id, sbp, dbp, bp_diff) |>
    slice_min(bp_diff, n = 3)
```

```
# A tibble: 3 x 4
    id   sbp   dbp bp_diff
```

```
      <dbl> <dbl> <dbl>    <dbl>
1  3822    120   100       20
2 11406    104    80       24
3 10204    110    86       24
```

This looks fine. Are any combinations out of line in the other direction, with a difference of 90 or more?

```
wcgs |> mutate(bp_diff = sbp - dbp) |>
    select(id, sbp, dbp, bp_diff) |>
    filter(bp_diff >= 90)
```

```
# A tibble: 8 x 4
     id   sbp   dbp bp_diff
   <dbl> <dbl> <dbl>   <dbl>
1  2172   208   102     106
2 12453   196    86     110
3 12280   190    98      92
4 13268   170    80      90
5  3546   188    90      98
6  2078   188    86     102
7 19078   230   118     112
8 22025   210   108     102
```

This `sbp - dbp` value is called the pulse pressure. Given that these men did not (at enrollment) have heart disease, it is surprising to see so many values separated by more than what would be typical (about 40-60 mm Hg.)

I'll be a little aggressive in my cleaning and also drop these 8 subjects with `sbp - dbp` of 90 or more mm Hg. So, we need to deal with the missingness in `chol`, the strange `bmi` value and the strange `chol` value in addition to dropping subjects with `sbp - dbp` to get to a "clean" data set here.

```
wcgs_ms <- wcgs |>
    select(id, age, bmi, sbp, dbp, smoke, dibpat, chol) |>
    filter(complete.cases(id, age, bmi, sbp,
                          dbp, smoke, dibpat, chol)) |> # drops 12 subjects
    filter(bmi > 15) |> # drops 1 subject
    filter(sbp - dbp < 90) |> # drops 7 more subjects
    filter(chol <= 400) # drops 2 more subjects

nrow(wcgs); nrow(wcgs_ms)
```

```
[1] 3154
```

```
[1] 3132
```

So we have dropped a total of 22 subjects from the data set. We could certainly consider imputing missing values of `chol`, but I am uncomfortable doing that for an outcome.

31.4 Partitioning the `wcgs_ms` sample

2. Partition the `wcgs_ms` tibble into a training (development) sample of 2000 observations, and a test (holdout) sample of the remaining observations.

Before we partition, it's always a good idea to ensure that each of the subject identifiers (here, `id`) are unique.

We could simply count the number of rows in the data set and make sure it matches the number of distinct `id` values.

```
c(nrow(wcgs_ms), n_distinct(wcgs_ms |> select(id)))
```

```
[1] 3132 3132
```

Looks good. If you prefer, check directly with the `identical()` function, which should return TRUE.

```
identical(nrow(wcgs_ms), n_distinct(wcgs_ms |> select(id)))
```

```
[1] TRUE
```

Remember to set a seed so that you can replicate the selection.

```
set.seed(431)

wcgs_ms_train <- slice_sample(wcgs_ms, n = 2000)

wcgs_ms_test <- anti_join(wcgs_ms, wcgs_ms_train, by = "id")

c(nrow(wcgs_ms), nrow(wcgs_ms_train), nrow(wcgs_ms_test))
```

```
[1] 3132 2000 1132
```

OK. This looks good. We have 2000 observations in the training sample, and the rest in the test sample.

Given a large sample size (at least 500 observations in the full data set) I would usually think about holding out somewhere between 20% and 30% of the data in this manner, but it's OK to stray a bit from those bounds, as we have here, with $1132/3132 = 36\%$ held out for the test sample.

31.5 Should we transform our outcome?

Consider the Box-Cox approach, which we'll check in our complete (model C) model.

```
boxCox(lm(chol ~ age + bmi + sbp + dbp + smoke + dibpat,
          data = wcgs_ms_train))
```

Profile Log–likelihood

It looks like we might want to consider using the square root of the cholesterol level as our outcome. We'll try that.

31.6 Scatterplot Matrix and Assessment of Collinearity

```
wcgs_ms_train <- wcgs_ms_train |>
    mutate(sqrtchol = sqrt(chol))
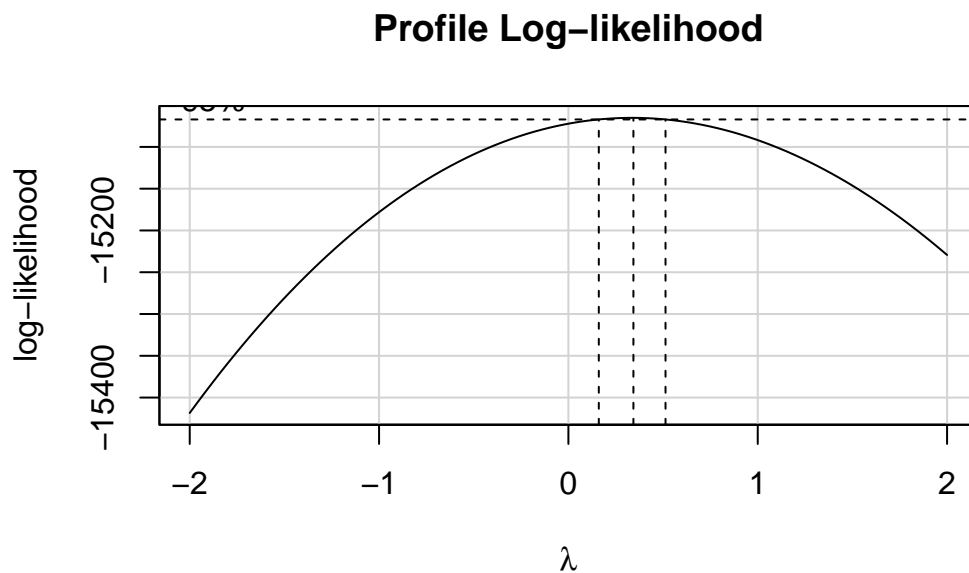
ggpairs(wcgs_ms_train |> select(age, bmi, sbp, dbp, smoke, dibpat, sqrtchol),
        lower = list(combo = wrap("facethist", binwidth = 0.5)),
        title = "2000 subjects in `wcgs_ms_train`")
```


2000 subjects in `wcgs_ms_train`

With so many observations and more than just a few predictors, it's not always easy to parse this plot. We might have considered splitting it into two parts.

Let's check for collinearity. Unsurprisingly, `sbp` and `dbp` are highly correlated with each other. Does this give us problems with variance inflation? Again, we'll consider the largest model we'll fit.

```
vif(lm(chol ~ age + bmi + sbp + dbp + smoke + dibpat,
            data = wcgs_ms_train))
```

```
     age      bmi      sbp      dbp    smoke   dibpat
1.038393 1.157091 2.536938 2.578828 1.035868 1.016123
```

The VIF is largest for `sbp` and `dbp`, suggesting that perhaps we won't need them both.

There is also a `vif` function within the **rms** package.

```
rms::vif(lm(chol ~ age + bmi + sbp + dbp + smoke + dibpat,
            data = wcgs_ms_train))
```

```
     age        bmi        sbp        dbp    smokeYes dibpatType B
1.038393    1.157091   2.536938   2.578828    1.035868     1.016123
```

Since we have candidate models using only `dbp`, let's see how they fare.

31.7 Fit our Three Candidate Models

Due to our choice to transform our outcome, we will now fit the following candidate models:

- Model A will predict `sqrt(chol)` using all six predictors.
- Model B will predict `sqrt(chol)` using five predictors, specifically `age`, `bmi`, `dbp`, `dibpat` and `smoke`.
- Model C will predict `sqrt(chol)` using only two predictors, specifically `dbp` and `smoke`

31.7.1 Three Candidate Models

```
modelA <- lm(sqrt(chol) ~ age + bmi + sbp + dbp + smoke + dibpat,
             data = wcgs_ms_train)
modelB <- lm(sqrt(chol) ~ age + bmi + dbp + smoke + dibpat,
             data = wcgs_ms_train)
modelC <- lm(sqrt(chol) ~ age + dbp + smoke,
             data = wcgs_ms_train)
```

31.7.2 Could we have fit other models?

Sure. One approach would have been to consider an automated variable selection tool, even though they're all pretty terrible. Stepwise regression is a common, if poor, choice for this task. The problem is that there's no good tool for this task, essentially, as we'll discuss further in 432.

- A stepwise regression applied to `modelA` in the training sample with backwards elimination suggests our `modelB`.

```
step(modelA)
```

```
Start:  AIC=1357.64
sqrt(chol) ~ age + bmi + sbp + dbp + smoke + dibpat

          Df Sum of Sq    RSS    AIC
- sbp      1     0.359 3916.0 1355.8
<none>                  3915.6 1357.6
- bmi      1     5.044 3920.6 1358.2
- dibpat   1    10.321 3925.9 1360.9
- age      1    22.217 3937.8 1367.0
- dbp      1    28.014 3943.6 1369.9
- smoke    1    46.322 3961.9 1379.2

Step:  AIC=1355.82
sqrt(chol) ~ age + bmi + dbp + smoke + dibpat

          Df Sum of Sq    RSS    AIC
<none>                  3916.0 1355.8
- bmi      1     4.858 3920.8 1356.3
- dibpat   1    10.204 3926.2 1359.0
- age      1    21.913 3937.9 1365.0
- smoke    1    46.021 3962.0 1377.2
- dbp      1    52.228 3968.2 1380.3


Call:
lm(formula = sqrt(chol) ~ age + bmi + dbp + smoke + dibpat, data = wcgs_ms_train)

Coefficients:
 (Intercept)          age          bmi          dbp     smokeYes
    12.03138      0.01930      0.02074      0.01835      0.30687
dibpatType B
    -0.14393
```

- A stepwise regression with forward selection from the intercept only model moving towards `modelA` in the training sample also suggests `modelB`.

```
min_model <- lm(sqrt(chol) ~ 1, data = wcgs_ms_train)
biggest <- formula(lm(sqrt(chol) ~ age + bmi + sbp + dbp + smoke + dibpat,
           data = wcgs_ms_train))
```

```r
fwd_step <- stats::step(min_model, direction = 'forward', scope = biggest)
```

```
Start:  AIC=1430.36
sqrt(chol) ~ 1

          Df Sum of Sq    RSS    AIC
+ dbp      1    83.336 4001.7 1391.1
+ sbp      1    57.718 4027.3 1403.9
+ smoke    1    39.557 4045.5 1412.9
+ age      1    39.060 4046.0 1413.1
+ dibpat   1    20.740 4064.3 1422.2
+ bmi      1    19.788 4065.2 1422.7
<none>                  4085.0 1430.4

Step:  AIC=1391.13
sqrt(chol) ~ dbp

          Df Sum of Sq    RSS    AIC
+ smoke    1    45.953 3955.7 1370.0
+ age      1    24.171 3977.5 1381.0
+ dibpat   1    16.675 3985.0 1384.8
<none>                  4001.7 1391.1
+ bmi      1     1.964 3999.7 1392.2
+ sbp      1     0.766 4000.9 1392.8

Step:  AIC=1370.04
sqrt(chol) ~ dbp + smoke

          Df Sum of Sq    RSS    AIC
+ age      1   24.3512 3931.4 1359.7
+ dibpat   1   13.4535 3942.3 1365.2
+ bmi      1    4.7652 3951.0 1369.6
<none>                 3955.7 1370.0
+ sbp      1    0.0080 3955.7 1372.0

Step:  AIC=1359.69
sqrt(chol) ~ dbp + smoke + age

          Df Sum of Sq    RSS    AIC
+ dibpat   1   10.5635 3920.8 1356.3
+ bmi      1    5.2180 3926.2 1359.0
<none>                 3931.4 1359.7
```

term	estimate	conf.low	conf.high	p.value
(Intercept)	12.058	10.933	13.183	0.000
age	0.019	0.005	0.034	0.001
bmi	0.021	-0.013	0.055	0.109
sbp	-0.001	-0.010	0.007	0.669
dbp	0.020	0.006	0.034	0.000
smokeYes	0.310	0.146	0.475	0.000
dibpatType B	-0.145	-0.308	0.018	0.022

```
+ sbp     1    0.0906 3931.3 1361.6

Step:  AIC=1356.3
sqrt(chol) ~ dbp + smoke + age + dibpat

       Df Sum of Sq    RSS    AIC
+ bmi   1    4.8581 3916.0 1355.8
<none>              3920.8 1356.3
+ sbp   1    0.1735 3920.6 1358.2

Step:  AIC=1355.82
sqrt(chol) ~ dbp + smoke + age + dibpat + bmi

       Df Sum of Sq    RSS    AIC
<none>              3916.0 1355.8
+ sbp   1    0.35901 3915.6 1357.6
```

It is tempting to conclude that `modelB` is somehow a good model, since both forwards and stepwise regression like `modelB` in our training sample given the predictors we fed to the algorithm. Resist that temptation. As you'll see, even with the pretty simple validation strategy we're going to use here, we won't wind up selecting model B.

31.7.3 Coefficients of our 3 models with `tidy`

Given the large sample size, let's look at some 99% confidence intervals for our model coefficients. We'll use `tidy` from the `broom` package.

```
tidy(modelA, conf.int = TRUE, conf.level = 0.99) |>
    select(term, estimate, conf.low, conf.high, p.value) |>
    kbl(digits = 3) |> kable_classic(full_width = F)
```

term	estimate	conf.low	conf.high	p.value
(Intercept)	12.031	10.918	13.145	0.000
age	0.019	0.004	0.034	0.001
bmi	0.021	-0.013	0.055	0.116
dbp	0.018	0.009	0.028	0.000
smokeYes	0.307	0.143	0.470	0.000
dibpatType B	-0.144	-0.307	0.019	0.023

term	estimate	conf.low	conf.high	p.value
(Intercept)	12.243	11.325	13.160	0
age	0.020	0.005	0.035	0
dbp	0.021	0.012	0.029	0
smokeYes	0.305	0.142	0.467	0

```r
tidy(modelB, conf.int = TRUE, conf.level = 0.99) |>
    select(term, estimate, conf.low, conf.high, p.value) |>
    kbl(digits = 3) |> kable_classic(full_width = F)
```

```r
tidy(modelC, conf.int = TRUE, conf.level = 0.99) |>
    select(term, estimate, conf.low, conf.high, p.value) |>
    kbl(digits = 3) |> kable_classic(full_width = F)
```

31.7.4 ANOVA comparison of the 3 models

Since model C is a subset of model B which is a subset of model A, we can compare these models with ANOVA tests.

```r
anova(modelC, modelB, modelA)
```

```
Analysis of Variance Table

Model 1: sqrt(chol) ~ age + dbp + smoke
Model 2: sqrt(chol) ~ age + bmi + dbp + smoke + dibpat
Model 3: sqrt(chol) ~ age + bmi + sbp + dbp + smoke + dibpat
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1   1996 3931.4
2   1994 3916.0  2    15.422 3.9247 0.0199 *
3   1993 3915.6  1     0.359 0.1827 0.6691
---
```

name	r2	adjr2	sigma	AIC	BIC	nobs	df	df.res
modelA	0.0415	0.0386	1.402	7035	7080	2000	6	1993
modelB	0.0414	0.0390	1.401	7034	7073	2000	5	1994
modelC	0.0376	0.0362	1.403	7037	7065	2000	3	1996

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- There appears to be a detectable improvement (using $\alpha = 0.05$) for model B as compared to model C, but not if we use $\alpha = 0.01$.
- The improvement from model B to model A doesn't appear to meet the standard for a statistically detectable impact based on this ANOVA comparison.

31.7.5 Assessing Fit Quality of our 3 models with `glance`

```
repA <- glance(modelA) |> mutate(name = "modelA")
repB <- glance(modelB) |> mutate(name = "modelB")
repC <- glance(modelC) |> mutate(name = "modelC")

fit_report <- bind_rows(repA, repB, repC)

fit_report |>
    select(name, r2 = r.squared, adjr2 = adj.r.squared, sigma,
           AIC, BIC, nobs, df, df.res = df.residual) |>
    kbl(digits = c(0,4,4,3,0,0,0,0,0)) |> kable_minimal()
```

Our conclusions are:

- Model A has the strongest R^2 value, as it must because it contains the predictors in the other models, and R^2 is greedy.
- Model B has a slightly stronger adjusted R^2 and σ than the other models, and also has the best performance according to AIC.
- Model C (the smallest of these models) shows the best in-sample BIC result.
- None of the differences we observe between these models are particularly large, and none of the models are especially effective, as judged by R^2.

31.8 Develop Residual Plots

We'll use `augment` from the `broom` package to calculate our summaries of the quality of fit at the level of the individual observations, and the fact that our fitted values refer to the square root of cholesterol level is fine for assessing assumptions in-sample.

```
aug_A <- augment(modelA, data = wcgs_ms_train)
aug_B <- augment(modelB, data = wcgs_ms_train)
aug_C <- augment(modelC, data = wcgs_ms_train)
```

31.8.1 First Set of Residual Diagnostics (3 models)

For each model, we'll start with the pair of plots that show us:

- the residuals vs. the fitted values, to check for non-linearity and non-constant variance, as well as

- Normal Q-Q plots of the standardized residuals to check for important non-Normality.

```
p1a <- ggplot(aug_A, aes(x = .fitted, y = .resid)) +
    geom_point(alpha = 0.2) +
    geom_point(data = aug_A |>
                   slice_max(abs(.resid), n = 5),
               col = "red", size = 2) +
    geom_text_repel(data = aug_A |>
                        slice_max(abs(.resid), n = 5),
                    aes(label = id), col = "red") +
    geom_abline(intercept = 0, slope = 0, lty = "dashed") +
    geom_smooth(method = "loess", formula = y ~ x, span = 2, se = F) +
    labs(title = "Model A",
         x = "Fitted sqrt(cholesterol)", y = "Residual")

p1b <- ggplot(aug_B, aes(x = .fitted, y = .resid)) +
    geom_point(alpha = 0.2) +
    geom_point(data = aug_B |>
                   slice_max(abs(.resid), n = 5),
               col = "red", size = 2) +
    geom_text_repel(data = aug_B |>
                        slice_max(abs(.resid), n = 5),
                    aes(label = id), col = "red") +
    geom_abline(intercept = 0, slope = 0, lty = "dashed") +
    geom_smooth(method = "loess", formula = y ~ x, span = 2, se = F) +
    labs(title = "Model B",
         x = "Fitted sqrt(cholesterol)", y = "Residual")

p1c <- ggplot(aug_C, aes(x = .fitted, y = .resid)) +
    geom_point(alpha = 0.2) +
```

```r
    geom_point(data = aug_C |>
                   slice_max(abs(.resid), n = 5),
               col = "red", size = 2) +
    geom_text_repel(data = aug_C |>
                        slice_max(abs(.resid), n = 5),
                    aes(label = id), col = "red") +
    geom_abline(intercept = 0, slope = 0, lty = "dashed") +
    geom_smooth(method = "loess", formula = y ~ x, span = 2, se = F) +
    labs(title = "Model C",
         caption = "In each plot, 5 largest |residuals| highlighted in red.",
         x = "Fitted sqrt(cholesterol)", y = "Residual")

p2a <- ggplot(aug_A, aes(sample = .std.resid)) +
    geom_qq() +
    geom_qq_line(col = "red") +
    labs(title = "Model A: Normal Q-Q Plot",
         y = "Model A Standardized Residual",
         x = "Standard Normal Quantiles")

p2b <- ggplot(aug_B, aes(sample = .std.resid)) +
    geom_qq() +
    geom_qq_line(col = "red") +
    labs(title = "Model B: Normal Q-Q Plot",
         y = "Model B Standardized Residual",
         x = "Standard Normal Quantiles")

p2c <- ggplot(aug_C, aes(sample = .std.resid)) +
    geom_qq() +
    geom_qq_line(col = "red") +
    labs(title = "Model C: Normal Q-Q Plot",
         y = "Model C Standardized Residual",
         x = "Standard Normal Quantiles")

(p1a + p2a) / (p1b + p2b) / (p1c + p2c) +
    plot_annotation(
        title = "Residual Diagnostics: Set 1 for Models A, B, C")
```

Residual Diagnostics: Set 1 for Models A, B, C
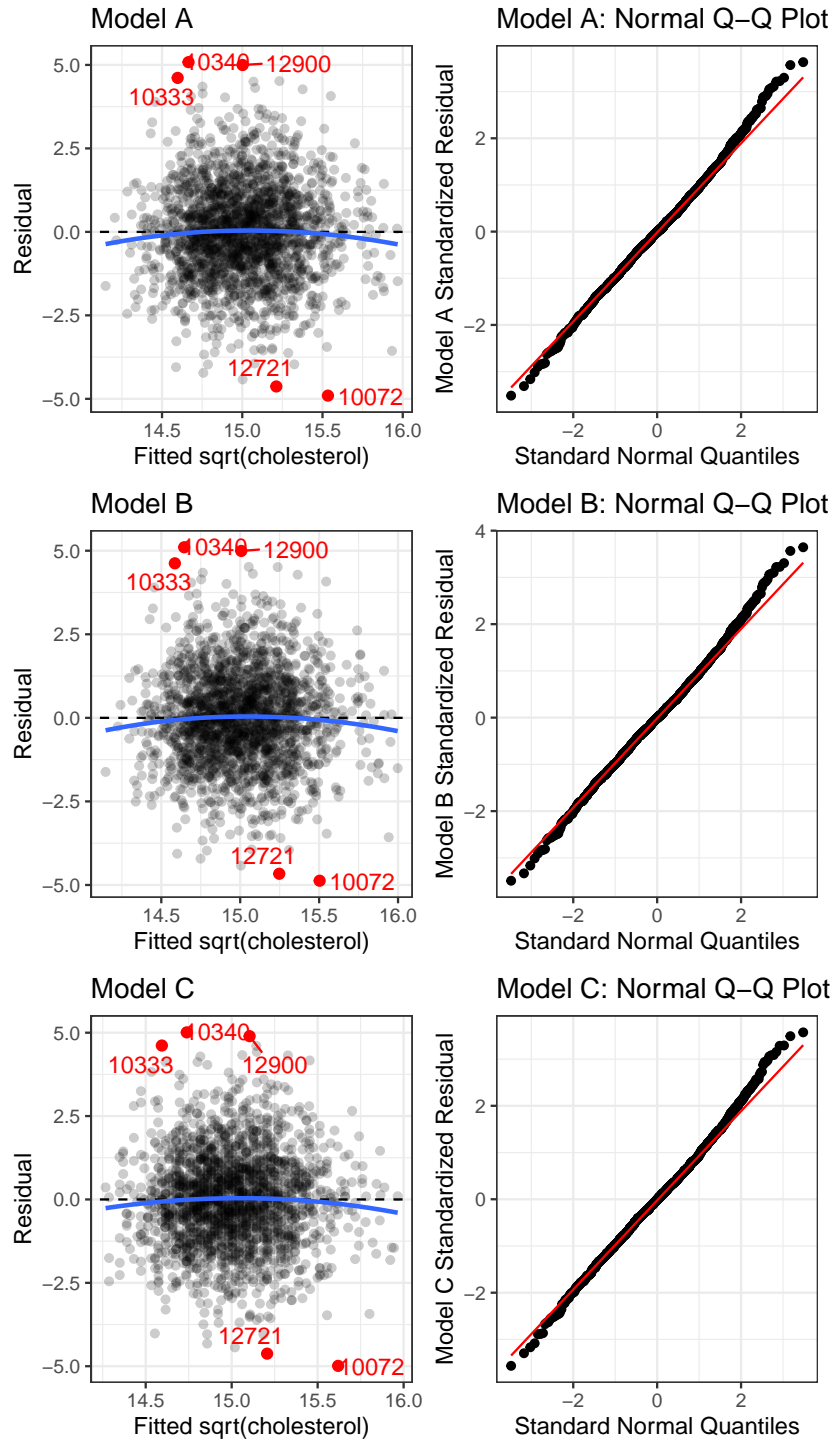
In each plot, 5 largest |residuals| highlighted in red.

In each of these plots, I see no clear signs of substantial non-linearity or any substantial problems with the assumptions of constant variance or Normality. No standardized residuals are especially unusual, and there's no sign of a substantial curve or fan shape in the plots of residuals vs. fitted values.

31.8.2 Second Set of Residual Diagnostics (3 models)

As a second check on the assumption of non-constant variance, we'll draw scale-location plots. As a second check on the potential for poorly fit or highly leveraged points to influence our results, we'll also run the residuals vs. leverage plots.

```
p3a <- ggplot(aug_A, aes(x = .fitted, y = sqrt(abs(.std.resid)))) +
  geom_point() +
  geom_smooth(method = "loess", formula = y ~ x, span = 2, se = F) +
  labs(title = "Model A Scale-Location Plot",
       x = "Fitted sqrt(cholesterol)",
       y = "Square Root of |Standardized Residual|")

p3b <- ggplot(aug_B, aes(x = .fitted, y = sqrt(abs(.std.resid)))) +
  geom_point() +
  geom_smooth(method = "loess", formula = y ~ x, span = 2, se = F) +
  labs(title = "Model B Scale-Location Plot",
       x = "Fitted sqrt(cholesterol)",
       y = "Square Root of |Standardized Residual|")

p3c <- ggplot(aug_C, aes(x = .fitted, y = sqrt(abs(.std.resid)))) +
  geom_point() +
  geom_smooth(method = "loess", formula = y ~ x, span = 2, se = F) +
  labs(title = "Model C Scale-Location Plot",
       x = "Fitted sqrt(cholesterol)",
       y = "Square Root of |Standardized Residual|")


p4a <- ggplot(aug_A, aes(x = .hat, y = .std.resid)) +
  geom_point() +
  geom_point(data = aug_A |> filter(.cooksd >= 0.5),
             col = "red", size = 2) +
  geom_text_repel(data = aug_A |> filter(.cooksd >= 0.5),
                  aes(label = id), col = "red") +
  geom_smooth(method = "loess", formula = y ~ x, span = 2, se = F) +
  geom_vline(aes(xintercept = 3*mean(.hat)), lty = "dashed") +
```

```
    labs(title = "Model A Residuals vs. Leverage",
         caption = "Red points indicate Cook's d at least 0.5",
         x = "Leverage", y = "Standardized Residual")

p4b <- ggplot(aug_B, aes(x = .hat, y = .std.resid)) +
  geom_point() +
  geom_point(data = aug_B |> filter(.cooksd >= 0.5),
             col = "red", size = 2) +
  geom_text_repel(data = aug_B |> filter(.cooksd >= 0.5),
                  aes(label = id), col = "red") +
  geom_smooth(method = "loess", formula = y ~ x, span = 2, se = F) +
  geom_vline(aes(xintercept = 3*mean(.hat)), lty = "dashed") +
  labs(title = "Model B Residuals vs. Leverage",
       caption = "Red points indicate Cook's d at least 0.5",
       x = "Leverage", y = "Standardized Residual")

p4c <- ggplot(aug_C, aes(x = .hat, y = .std.resid)) +
  geom_point() +
  geom_point(data = aug_C |> filter(.cooksd >= 0.5),
             col = "red", size = 2) +
  geom_text_repel(data = aug_C |> filter(.cooksd >= 0.5),
                  aes(label = id), col = "red") +
  geom_smooth(method = "loess", formula = y ~ x, span = 2, se = F) +
  geom_vline(aes(xintercept = 3*mean(.hat)), lty = "dashed") +
  labs(title = "Model C Residuals vs. Leverage",
       caption = "Red points indicate Cook's d at least 0.5",
       x = "Leverage", y = "Standardized Residual")

(p3a + p4a) / (p3b + p4b) / (p3c + p4c) +
    plot_annotation(
        title = "Residual Diagnostics: Set 2 for Models A, B, C")
```

Residual Diagnostics: Set 2 for Models A, B, C

Model A Scale-Location Plo

Square Root of |Standardized Residual|

Fitted sqrt(cholesterol)

Model A Residuals vs. Lever

Standardized Residual

Leverage

Red points indicate Cook's d at least 0.5

Model B Scale-Location Plo

Square Root of |Standardized Residual|

Fitted sqrt(cholesterol)

Model B Residuals vs. Lever

Standardized Residual

Leverage

Red points indicate Cook's d at least 0.5

Model C Scale-Location Plo

Square Root of |Standardized Residual|

Fitted sqrt(cholesterol)

Model C Residuals vs. Lever

Standardized Residual

Leverage

Red points indicate Cook's d at least 0.5

Again, I don't see any signs of powerful trends in the scale-location plot for any of the three models, so there's no clear sign of non-constant variance.

For each of the three models, the residuals vs. leverage plot shows no points highlighted in red (which would indicate substantial influence - a Cook's distance value above 5.) So I don't see anything to worry about.

31.8.3 Numerical Summaries of these measures (all 3 models)

We could summarize the largest standardized residual and the largest Cook's distance, for instance, for each model, with something like this.

```
aug_A |> slice_max(abs(.std.resid), n = 1) |>
    select(id, sqrtchol, .fitted, .resid, .std.resid)
```

```
# A tibble: 1 x 5
     id sqrtchol .fitted .resid .std.resid
  <dbl>    <dbl>   <dbl>  <dbl>      <dbl>
1 10340     19.7    14.7   5.08       3.63
```

```
aug_A |> slice_max(.cooksd, n = 1) |>
    select(id, sqrtchol, .fitted, .resid, .hat, .cooksd)
```

```
# A tibble: 1 x 6
     id sqrtchol .fitted .resid    .hat .cooksd
  <dbl>    <dbl>   <dbl>  <dbl>   <dbl>   <dbl>
1 10072     10.6    15.5  -4.90 0.00938  0.0167
```

31.9 Test Sample Comparisons for our 3 Models

Finally, we'll use our three candidate models (A, B and C) to predict the results in our holdout sample of 1132 observations not used to fit these models to see which model performs better in these new data.

Once again, we'll be using augment to do this work. To start, this is straightforward - we just need to specify the new data we want to predict with newdata.

```
test_A <- augment(modelA, newdata = wcgs_ms_test) |> mutate(mod_n = "Model A")
test_B <- augment(modelB, newdata = wcgs_ms_test) |> mutate(mod_n = "Model B")
```

```
test_C <- augment(modelC, newdata = wcgs_ms_test) |> mutate(mod_n = "Model C")
```

Let's look at the first two results using Model A.

```
test_A |>
    mutate(sqrtchol = sqrt(chol)) |>
    select(chol, sqrtchol, .fitted, .resid) |> head(2)
```

```
# A tibble: 2 x 4
   chol sqrtchol .fitted .resid
  <dbl>    <dbl>   <dbl>  <dbl>
1   194     13.9    15.2  -1.28
2   258     16.1    15.5  0.592
```

As we can see, the `.fitted` value is trying to predict the square root of `chol` and not `chol`. When we are doing validation in our test sample, it is more helpful to get back to the scale of our original outcome. So here, we'd square the `.fitted` value (to back out of our square root transformation) and get the actual prediction our model makes for `chol`. Then we can take the observed `chol` value and subtract the prediction of `chol` to get a new residual on the scale of our original `chol` values.

```
test_res <- bind_rows(test_A, test_B, test_C) |>
    mutate(fit_chol = .fitted^2, res_chol = chol - fit_chol) |>
    select(mod_n, id, chol, fit_chol, res_chol, everything()) |>
    arrange(id, mod_n)

test_res |> head()
```

```
# A tibble: 6 x 13
  mod_n      id  chol fit_chol res_chol   age   bmi   sbp   dbp smoke dibpat
  <chr>   <dbl> <dbl>    <dbl>    <dbl> <dbl> <dbl> <dbl> <dbl> <chr> <chr>
1 Model A  2003   181     214.    -33.2    42  23.6   110    78 No    Type B
2 Model B  2003   181     214.    -32.7    42  23.6   110    78 No    Type B
3 Model C  2003   181     216.    -35.0    42  23.6   110    78 No    Type B
4 Model A  2004   132     222.    -89.9    41  23.1   124    78 Yes   Type B
5 Model B  2004   132     222.    -89.9    41  23.1   124    78 Yes   Type B
6 Model C  2004   132     224.    -92.5    41  23.1   124    78 Yes   Type B
# i 2 more variables: .fitted <dbl>, .resid <dbl>
```

Now, we can summarize the quality of the predictions across each of the models with four summary statistics calculated across the 1132 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(res_chol)),
              medAPE = median(abs(res_chol)),
              maxAPE = max(abs(res_chol)),
              RMSPE = sqrt(mean(res_chol^2)))
```

```
# A tibble: 3 x 5
  mod_n     MAPE medAPE maxAPE RMSPE
  <chr>    <dbl>  <dbl>  <dbl> <dbl>
1 Model A   32.2   27.1   151.  40.8
2 Model B   32.2   27.1   150.  40.8
3 Model C   32.1   26.5   147.  40.8
```

Which model has the best performance in the testing sample?

- Model C has the smallest (best) MAPE, median APE, maximum APE and RMSPE.
- There are no enormous differences between the models on most of these summaries, but Model C (the smallest model) appears a bit better.

31.10 Putting it Together - which model do we like best?

It's worth remembering that none of the three models was particularly strong. Even Model A (with the full set of predictors) had an R^2 of only about 4.15%.

1. All 3 models had reasonable residual plots, so we don't have much to choose from there.
2. Within the development sample, Model B had slightly better adjusted R^2, σ and AIC results than the other models, while Model C had the best BIC.
3. Model C had the best results in the test sample, across all four summaries we examined.

In this case, I'd pick Model C from among these options, based mainly on the stronger test sample results and the fact that it's a smaller model (with fewer predictors) and if none of the models are going to do very well in terms of predicting `chol`, there's not much to choose from anyway.

32 Species Found on the Galapagos Islands

32.1 Setup: Packages Used Here

```
knitr::opts_chunk$set(comment = NA)

library(broom)
library(car)
library(GGally)
library(knitr)
library(tidyverse)

theme_set(theme_bw())
```

We'll also use a function from the `arm` package, and from the `Hmisc` package.

32.2 A Little Background

The `gala` data describe describe features of the 30 Galapagos Islands.

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

The Galapagos Islands are found about 900 km west of South America: specifically the continental part of Ecuador. The Islands form a province of Ecuador and serve as a national park and marine reserve. They are noted for their vast numbers of unique (or endemic) species and were studied by Charles Darwin during the voyage of the Beagle.

32.2.1 Sources

The data were initially presented by Johnson M and Raven P (1973) Species number and endemism: the Galapagos Archipelago revisited. *Science* 179: 893-895 and also appear in several regression texts, including my source: Faraway (2015). Note that Faraway filled in some missing data to simplify things a bit. A similar version of the data is available as part of the `faraway` library in R, but I encourage you to use the version I supply on our web site.

Figure 32.1: galapic.jpg

32.2.2 Variables in the `gala` data frame

- **id** = island identification code
- **island** = island name
- **species** = our outcome, the number of species found on the island
- **area** = the area of the island, in square kilometers
- **elevation** = the highest elevation of the island, in meters
- **nearest** = the distance from the nearest island, in kilometers
- **scruz** = the distance from Santa Cruz Island, in kilometers. Santa Cruz is the home to the largest human population in the Islands, and to the town of Puerto Ayora.
- **adjacent** = the area of the adjacent island, in square kilometers

```
gala
```

```
# A tibble: 30 x 8
      id island       species  area elevation nearest scruz adjacent
   <dbl> <chr>          <dbl> <dbl>     <dbl>   <dbl> <dbl>    <dbl>
 1     1 Baltra            58 25.1        346     0.6   0.6     1.84
 2     2 Bartolome         31  1.24       109     0.6  26.3   572.
 3     3 Caldwell           3  0.21       114     2.8  58.7     0.78
 4     4 Champion          25  0.1         46     1.9  47.4     0.18
 5     5 Coamano            2  0.05        77     1.9   1.9   904.
 6     6 Daphne.Major      18  0.34       119     8     8       1.84
 7     7 Daphne.Minor      24  0.08        93     6    12       0.34
 8     8 Darwin            10  2.33       168    34.1 290.      2.85
 9     9 Eden               8  0.03        71     0.4   0.4    18.0
10    10 Enderby            2  0.18       112     2.6  50.2     0.1
# i 20 more rows
```

```
Hmisc::describe(gala) # check for missing and inexplicable values
```

```
gala

 8  Variables      30  Observations
--------------------------------------------------------------------------------
id
       n  missing distinct      Info      Mean       Gmd       .05       .10
      30        0       30         1      15.5     10.33      2.45      3.90
     .25      .50      .75       .90       .95
    8.25    15.50    22.75     27.10     28.55
```

```
lowest :  1  2  3  4  5, highest: 26 27 28 29 30
--------------------------------------------------------------------------------
island
       n  missing distinct
      30        0       30

lowest : Baltra      Bartolome Caldwell   Champion    Coamano
highest: SantaFe     SantaMaria Seymour    Tortuga     Wolf
--------------------------------------------------------------------------------
species
       n  missing distinct     Info     Mean      Gmd      .05      .10
      30        0       27    0.999    85.23    109.5      2.0      2.9
     .25      .50      .75      .90      .95
    13.0     42.0     96.0    280.5    319.1

lowest :   2   3   5   8  10, highest: 237 280 285 347 444
--------------------------------------------------------------------------------
area
       n  missing distinct     Info     Mean      Gmd      .05      .10
      30        0       29        1    261.7    478.6   0.0390   0.0770
     .25      .50      .75      .90      .95
  0.2575   2.5900  59.2375 578.5460 782.6215

Value           0    50   100   150   550   600   900  4650
Frequency      21     2     1     1     2     1     1     1
Proportion 0.700 0.067 0.033 0.033 0.067 0.033 0.033 0.033

For the frequency table, variable is rounded to the nearest 50
--------------------------------------------------------------------------------
elevation
       n  missing distinct     Info     Mean      Gmd      .05      .10
      30        0       30        1      368    411.1    47.35    68.80
     .25      .50      .75      .90      .95
   97.75   192.00   435.25   868.20  1229.40

lowest :   25   46   49   71   76, highest:  777  864  906 1494 1707
--------------------------------------------------------------------------------
nearest
       n  missing distinct     Info     Mean      Gmd      .05      .10
      30        0       22    0.997    10.06    13.73    0.445    0.590
     .25      .50      .75      .90      .95
   0.800    3.050   10.025   34.100   40.205
```

```
lowest : 0.2  0.4  0.5  0.6  0.7 , highest: 16.5 29.1 34.1 45.2 47.4
--------------------------------------------------------------------------------
scruz
       n  missing distinct      Info      Mean      Gmd      .05      .10
      30        0       29         1     56.98    65.17     0.49     0.60
     .25      .50      .75       .90       .95
   11.02    46.65    81.08     97.73    193.90

lowest : 0     0.4   0.6   1.9   8    , highest: 93.1  95.3  119.6 254.7 290.2
--------------------------------------------------------------------------------
adjacent
       n  missing distinct      Info      Mean      Gmd      .05      .10
      30        0       21     0.998     261.1    477.8     0.10     0.10
     .25      .50      .75       .90       .95
    0.52     2.59    59.24    578.55    782.62

Value            0    50   100   550   600   900  4650
Frequency       21     2     2     2     1     1     1
Proportion 0.700 0.067 0.067 0.067 0.033 0.033 0.033

For the frequency table, variable is rounded to the nearest 50
--------------------------------------------------------------------------------
```

32.3 DTDP: A Scatterplot Matrix

After missingness and range checks, the first step in any data analysis problem is to draw the picture. The most useful picture for me in thinking about a regression problem with a reasonably small number of predictors is a scatterplot matrix.

Our outcome, that we are predicting here is the number of `species`.

We'll use five predictors:

- `area`
- `elevation`
- `nearest`
- `scruz` and
- `adjacent`.

```
ggpairs(gala |> select(area, elevation, nearest,
                       scruz, adjacent, species))
```

32.3.1 Questions about the Scatterplot Matrix

In this Chapter, I will provide you with nearly 100 questions that you should be able to answer in light of the output provided. Here are the first few. If you have questions about how to interpret the output in light of these questions, please ask about them on Piazza or in TA office hours.

1. What are we looking for in the scatterplots in the bottom row?
2. What can we learn from the Pearson correlations in the right column?
3. How do the density plots help increase our understanding of the data?
4. What about the scatterplots that are not in the top row?
5. What can we learn from the Pearson correlations that compare predictors?

32.4 Fitting A "Kitchen Sink" Linear Regression model

Next, we'll fit a multiple linear regression model to predict the number of species based on the five predictors included in the gala data frame (and scatterplot matrix above.) We use the lm command to fit the linear model, and use what is called Wilkinson-Rogers notation to specify the model.

```
model1 <- lm(species ~ area + elevation + nearest + scruz +
             adjacent, data=gala)
```

Here are the results of running `tidy()` and `glance()` from the `broom` package on `model1`.

```
tidy(model1) |>
    kable(digits = 3)
```

term	estimate	std.error	statistic	p.value
(Intercept)	7.068	19.154	0.369	0.715
area	-0.024	0.022	-1.068	0.296
elevation	0.319	0.054	5.953	0.000
nearest	0.009	1.054	0.009	0.993
scruz	-0.241	0.215	-1.117	0.275
adjacent	-0.075	0.018	-4.226	0.000

```
glance(model1) |>
    select(r.squared, adj.r.squared, sigma, statistic,
           p.value, df, df.residual, nobs, AIC, BIC) |>
    kable(digits = c(3, 3, 1, 1, 3, 0, 0, 0, 1, 1))
```

r.squared	adj.r.squared	sigma	statistic	p.value	df	df.residual	nobs	AIC	BIC
0.766	0.717	61	15.7	0	5	24	30	339.1	348.9

Here is the result of running `summary` on `model1`.

```
summary(model1)
```

```
Call:
lm(formula = species ~ area + elevation + nearest + scruz + adjacent,
    data = gala)

Residuals:
     Min       1Q   Median       3Q      Max
-111.679  -34.898   -7.862   33.460  182.584

Coefficients:
```

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.068221  19.154198    0.369 0.715351
area        -0.023938   0.022422   -1.068 0.296318
elevation    0.319465   0.053663    5.953 3.82e-06 ***
nearest      0.009144   1.054136    0.009 0.993151
scruz       -0.240524   0.215402   -1.117 0.275208
adjacent    -0.074805   0.017700   -4.226 0.000297 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 60.98 on 24 degrees of freedom
Multiple R-squared:  0.7658,    Adjusted R-squared:  0.7171
F-statistic:  15.7 on 5 and 24 DF,  p-value: 6.838e-07
```

32.4.1 Questions about the Kitchen Sink Model Summaries

What conclusions can we draw from the `summary` output for this model? Specifically …

6. What is being predicted? What is the prediction equation?
7. How do we interpret the `elevation` estimate of 0.32?
8. How do we interpret the `area` estimate of -0.02?
9. How do we interpret the intercept estimate of 7.07?
10. Overall, does our `model1` add detectable predictive value (say, at the 5% significance level) over the simplest possible model for this outcome (the model using the intercept term alone)?
11. What proportion of the variation in `species` counts does this model account for?
12. What does the residual standard error mean in this context?
13. What can we learn from the standard errors in the coefficient output?
14. What can we learn from the `t` values and `Pr(>|t|)` values in the summary output?
15. How should we interpret the meaning of the `Adjusted R-squared` value?

32.5 Finding Confidence Intervals for our Coefficient Estimates

Of course, we can find these intervals in several ways. One approach uses `tidy()` from the `broom` package.

```
tidy(model1, conf.int = TRUE, conf.level = 0.95) |>
    select(term, estimate, std.error, conf.low, conf.high) |>
    kable(digits = 3)
```

term	estimate	std.error	conf.low	conf.high
(Intercept)	7.068	19.154	-32.464	46.601
area	-0.024	0.022	-0.070	0.022
elevation	0.319	0.054	0.209	0.430
nearest	0.009	1.054	-2.166	2.185
scruz	-0.241	0.215	-0.685	0.204
adjacent	-0.075	0.018	-0.111	-0.038

Another approach would be:

```
confint(model1, level = 0.95)
```

```
                 2.5 %        97.5 %
(Intercept) -32.4641006 46.60054205
area         -0.0702158  0.02233912
elevation     0.2087102  0.43021935
nearest      -2.1664857  2.18477363
scruz        -0.6850926  0.20404416
adjacent     -0.1113362 -0.03827344
```

32.5.1 Questions about the Confidence Intervals

16. What can we learn from the provided confidence interval for `elevation`?
17. How do the confidence interval results here compare to the t tests we saw previously?

32.6 Measuring Collinearity - the Variance Inflation Factor

The **variance inflation factor** (abbreviated VIF) can be used to quantify the impact of multicollinearity in a linear regression model.

The VIF is sometimes interpreted by taking its square root, and then interpreting the result as telling you how much larger the standard error for that coefficient is, as compared to what it would be if that variable were uncorrelated with the other predictors.

In R, the `vif` function from the `car` library, when applied to a linear regression model, specifies the variance inflation factors for each of the model's coefficients, as follows.

```
vif(model1)
```

```
     area elevation    nearest    scruz  adjacent
 2.928145  3.992545  1.766099  1.675031  1.826403
```

So, for instance, the VIF of 3.99 for `elevation` implies that the standard error of the elevation coefficient is approximately 2 times larger than it would be if elevation was uncorrelated with the other predictors.

I will look closely at any VIF value that is greater than 5, although some people use a cutoff of 10.

- Another collinearity measure called tolerance is simply $1/$`VIF`.
- For example, the tolerance for `elevation` would be 0.25, and the cutoff for a potentially problematic tolerance is either 0.2 or lower, or 0.1 or lower.

To calculate the VIF for a predictor x_1, use all of the other predictors to predict x_1 and find the multiple R^2 value.

- `VIF` for $x_1 = 1 \ / \ (1 - R^2_{x_1|others})$, and `tolerance` $= (1 - R^2_{x_1|others})$.

32.7 Global (F) Testing of Overall Significance

Our Galapagos Islands species count regression model (called `model1`) predicts the count of an island's species using `area`, `elevation`, `nearest`, `scruz` and `adjacent`.

```
nullmodel <- lm(species ~ 1, data=gala)
summary(nullmodel)
```

```
Call:
lm(formula = species ~ 1, data = gala)

Residuals:
   Min     1Q Median     3Q    Max
-83.23 -72.23 -43.23  10.77 358.77

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    85.23      20.93   4.072 0.000328 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 114.6 on 29 degrees of freedom
```

```
anova(model1, nullmodel)
```

```
Analysis of Variance Table

Model 1: species ~ area + elevation + nearest + scruz + adjacent
Model 2: species ~ 1
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1     24  89231
2     29 381081 -5   -291850 15.699 6.838e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

32.7.1 Questions about the Global Test via ANOVA

18. How do we interpret the null model fit above?
19. What are the hypotheses being tested by this ANOVA output?
20. What conclusions can we draw from the ANOVA output presented here?
21. Where do we find information regarding the result for the previous question in the summary output for the linear model?
22. How would we set up an ANOVA model to test whether the "kitchen sink" model's predictive value would be meaningfully impacted by removing the adjacent predictor from the model?
23. Where do we find information regarding these result for the previous question in the summary output for the linear model?
24. How would we set an ANOVA model to test whether a model with area only would be a detectable improvement over the null model?

32.8 Sequential Testing in a Regression Model with ANOVA

```
anova(model1)
```

```
Analysis of Variance Table

Response: species
          Df Sum Sq Mean Sq F value    Pr(>F)
area       1 145470  145470 39.1262 1.826e-06 ***
elevation  1  65664   65664 17.6613 0.0003155 ***
nearest    1     29      29  0.0079 0.9300674
```

```
scruz       1  14280    14280  3.8408 0.0617324 .
adjacent    1  66406    66406 17.8609 0.0002971 ***
Residuals 24  89231     3718
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

32.8.1 Questions about Sequential Testing and ANOVA

25. What conclusions can we draw from the `area` row in the output above?
26. What conclusions can we draw from the `elevation` row?
27. Does `nearest` add statistically detectable predictive value to the model including `area` and `elevation`, but none of the other predictors?
28. Does `adjacent` add detectable predictive value as last predictor into the model?
29. Where else in the regression output can we find the answer to the previous question?
30. How does the mean square of the residuals (3718) relate to the residual standard error?
31. What percentage of the variation in the species counts is accounted for by area alone?
32. What percentage of the variation explained by the kitchen sink model would also be accounted for in a two-predictor regression model including `area` and `elevation` alone?
33. How could we use the original linear model output to whether a model using the four predictors that appear most promising here would be statistically meaningfully worse than the full model with all five predictors?
34. What does the following output do differently than the output above, and why is that potentially useful here? Why is the p value for `scruz` so different?

```
anova(lm(species ~ area + elevation + adjacent +
             scruz + nearest, data=gala))
```

```
Analysis of Variance Table

Response: species
          Df Sum Sq Mean Sq F value    Pr(>F)
area       1 145470  145470 39.1262 1.826e-06 ***
elevation  1  65664   65664 17.6613 0.0003155 ***
adjacent   1  73171   73171 19.6804 0.0001742 ***
scruz      1   7544    7544  2.0291 0.1671885
nearest    1      0       0  0.0001 0.9931506
Residuals 24  89231    3718
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

35. Consider the ANOVA below, run on a new model with `elevation` after `adjacent`. What happens? Why?

633

```
anova(lm(species ~ area + adjacent + elevation +
           scruz + nearest, data=gala))
```

Analysis of Variance Table

Response: species
          Df Sum Sq Mean Sq F value   Pr(>F)
area   1 145470 145470 39.1262 1.826e-06 ***
adjacent   1   2850   2850 0.7666   0.3900
elevation 1 135985 135985 36.5751 3.030e-06 ***
scruz   1   7544   7544 2.0291   0.1672
nearest   1   0   0 0.0001   0.9932
Residuals 24 89231   3718

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

32.9 An ANOVA table for the Model as a Whole

It's probably also worthwhile to compute a completed ANOVA table for the model as a whole. All elements are in the ANOVA tables above, or the model summary.

Group	DF	SS	MS	F	P
Regression	5	291849	58369.8	15.7	6.838e-07
Residuals	24	89231	3718.0		
Total	29	381080			

36. How did I determine the Mean Square for the Regression model?
37. What conclusions can we draw from this ANOVA table?

32.10 Assumption Checking for our Galapagos Islands models

Remember that the key assumptions of multiple linear regression are:

- [Linearity] We have also assumed that the structural part of the model is correctly specified (we've included all the predictors that need to be in the model, and expressed them in the most appropriate manner, and we've left out any predictors that don't need to be in the model.)
- [Normality] The regression makes errors that come from a Normal distribution

- [Homoscedasticity = Constant Variance] The regression makes errors that come from a distribution with constant variance at all predictor levels.
- [Independence] The regression errors are independent of each other.

In addition, we need to realize that sometimes a few observations may be particularly problematic. For instance:

1. An observation may simply not fit the model well (i.e. it creates a large residual)
2. An observation may have high leverage over the fit of the model (this happens with observations that have an unusual combination of predictor values, in particular)
3. An observation may actually have high influence on the model (in the sense that whether they are included or excluded has a large impact on the model's fit, and the value of its parameter estimates.)
4. Or any combination of high residual, leverage and influence may occur.

So it is important to check the assumptions that we can with the data we have. Our most important tools are plots and other summaries of residuals, and what are called influence statistics.

32.11 My First Plot: Studentized Residuals vs. Fitted Values

The first diagnostic plot I usually draw for a multiple regression is a scatterplot of the model's **studentized** residuals[1] (on the vertical axis) vs. the model's fitted values (on the horizontal.) This plot can be used to assess potential non-linearity, non-constant variance, and non-Normality in the residuals.

```
gala$stures <- rstudent(model1); gala$fits <- fitted(model1)
ggplot(gala, aes(x = fits, y = stures)) +
    geom_point(size = 3, shape = 1) +
    geom_smooth(col = "blue", se = FALSE, method = "loess",
                formula = y ~ x, weight = 0.5) +
    geom_hline(aes(yintercept = 0), linetype = "dashed") +
    labs(x = "Fitted Values of Species",
         y = "Studentized Residuals",
         title = "gala data: Model 1",
         subtitle = "Studentized Residuals vs. Fitted Values")
```

---

[1]More on studentized and standardized residuals later. For now, think of them like z scores.

gala data: Model 1

Studentized Residuals vs. Fitted Values

32.11.1 Questions about Studentized Residuals vs. Fitted Values

38. Consider the point at bottom right. What can you infer about this observation?
39. Why did I include the dotted horizontal line at Studentized Residual = 0?
40. What is the purpose of the thin blue line?
41. What does this plot suggest about the potential for outliers in the residuals?

32.12 Automatic Regression Diagnostics for Model 1

```
par(mfrow=c(2,2))
plot(model1)
```

636

```
par(mfrow=c(1,1))
```

32.13 Model 1: Diagnostic Plot 1

As we've seen, the first of R's automated diagnostic plots for a linear model is a plot of the residuals vs. the fitted values.

```
plot(model1, which=1)
```


lm(species ~ area + elevation + nearest + scruz + adjacent)

32.13.1 Questions about Diagnostic Plot 1: Residuals vs. Fitted Values

42. What type of regression residuals is R plotting here?
43. Which points are identified by numbers here?
44. Why did R include the gray dotted line at Residual = 0?
45. What is the purpose of the thin red line?
46. What can you tell about the potential for outliers in the model 1 residuals from the plot?
47. What are we looking for in this plot that would let us conclude there were no important assumption violations implied by it? Which assumptions can we assess with it?
48. What would we do if we saw a violation of assumptions in this plot?
49. What are the key differences between this plot and the one I showed earlier?

32.14 Diagnostic Plot 2: Assessing Normality

The second diagnostic plot prepared by R for any linear model using the plot command is a Normal Q-Q plot of the standardized residuals from the model.

```
plot(model1, which=2)
```


32.14.1 Questions about Diagnostic Plot 2: Normal Plot of Standardized Residuals

50. Which points are being identified here by number?
51. Which assumption(s) of multiple regression does this plot help us check?
52. What are we looking for in this plot that would let us conclude there were no important assumption violations implied by it?
53. What would we do if we saw a violation of assumptions in this plot?

We could also look at studentized residuals, or we could apply a more complete set of plots and other assessments of normality. Usually, I don't.

32.15 Diagnostic Plot 3: Assessing Constant Variance

The third diagnostic plot prepared by R for any linear model using the `plot` command shows the square root of the model's standardized residuals vs. its fitted values. R calls this a **scale-location plot**.

```
plot(model1, which=3)
```


32.15.1 Questions about Diagnostic Plot 3: Scale-Location Plot

54. Which points are being identified here by number?
55. Which assumption(s) of multiple regression does this plot help us check?
56. What is the role of the thin red line in this plot?
57. What are we looking for in this plot that would let us conclude there were no important assumption violations implied by it?
58. What would we do if we saw a violation of assumptions in this plot?

32.16 Obtaining Fitted Values and Residuals from a Model

We can use the `augment()` function from the `broom` package to find the predictions made by the regression model for each of the observations used to create the model.

```
gala_aug <- augment(model1)
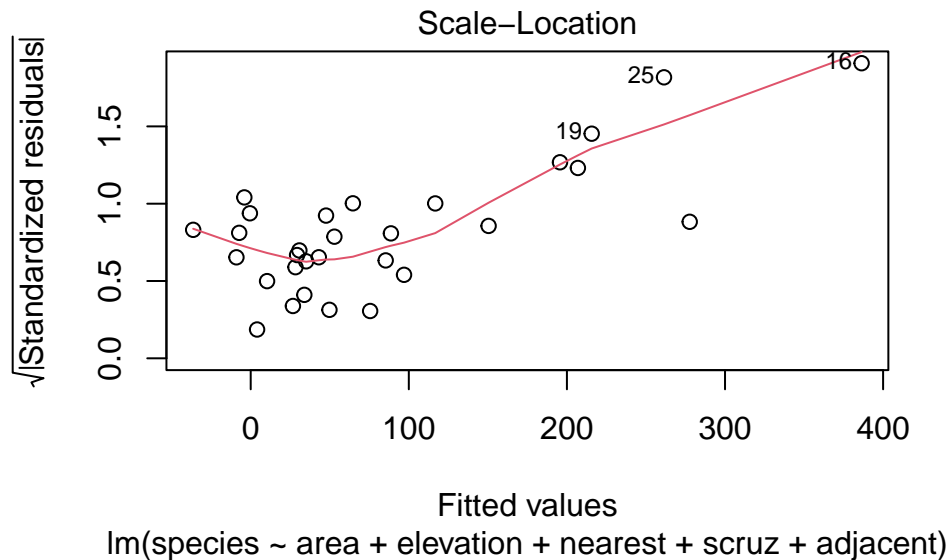
names(gala_aug)
```

```
 [1] "species"    "area"       "elevation"  "nearest"    "scruz"
 [6] "adjacent"   ".fitted"    ".resid"     ".hat"       ".sigma"
[11] ".cooksd"    ".std.resid"
```

Here, we obtain

- predicted (fitted) values for the outcome, in `.fitted`
- residuals (observed - fitted) values for each observation, in `.resid`
- leverage values in `.hat` for each observation
- standard errors associated with each observation in `.sigma`
- Cook's distance values for each observation in `.cooksd`
- standardized residuals in `.std.resid`

Here, for instance, are the first few rows.

```
head(gala_aug) |> kable(digits = 2)
```

species	area	elevation	nearest	scruz	adjacent	.fitted	.resid	.hat	.sigma	.cooksd	.std.resid
58	25.09	346	0.6	0.6	1.84	116.73	-58.73	0.08	60.97	0.01	-1.00
31	1.24	109	0.6	26.3	572.33	-7.27	38.27	0.09	61.72	0.01	0.66
3	0.21	114	2.8	58.7	0.78	29.33	-26.33	0.06	62.03	0.00	-0.45
25	0.10	46	1.9	47.4	0.18	10.36	14.64	0.07	62.21	0.00	0.25
2	0.05	77	1.9	1.9	903.82	-36.38	38.38	0.17	61.66	0.02	0.69
18	0.34	119	8.0	8.0	1.84	43.09	-25.09	0.07	62.05	0.00	-0.43

32.16.1 Questions about Fitted Values

59. Verify that the first fitted value [116.73] is in fact what you get for Baltra (observation 1) when you apply the regression equation:

```
species = 7.07 - 0.02 area + 0.32 elevation
              + 0.009 nearest - 0.24 scruz - 0.07 adjacent
```

We can compare these predictions to the actual observed counts of the number of species on each island. Subtracting the fitted values from the observed values gives us the residuals, as does the `resid` function.

```
round(resid(model1),2)
```

```
      1        2        3        4        5        6        7        8        9       10
 -58.73    38.27   -26.33    14.64    38.38   -25.09    -9.92    19.02   -20.31   -28.79
     11       12       13       14       15       16       17       18       19       20
  49.34    -3.99    62.03   -59.63    40.50   -39.40   -37.69    -2.04  -111.68   -42.48
     21       22       23       24       25       26       27       28       29       30
 -23.08    -5.55    73.05   -40.68   182.58   -23.38    89.38    -5.81   -36.94    -5.70
```

32.16.2 Questions about Residuals

60. What does a positive residual indicate?
61. What does a negative residual indicate?
62. The standard deviation of the full set of 30 residuals turns out to be 55.47. How does this compare to the residual standard error?
63. The command below identifies Santa Cruz. What does it indicate about Santa Cruz, specifically?

```
gala$island[which.max(resid(model1))]
```

```
[1] "SantaCruz"
```

64. From the results below, what is the `model1` residual for Santa Cruz? What does this imply about the `species` prediction made by Model 1 for Santa Cruz?

```
which.max(resid(model1))
```

```
25
25
```

```
round(resid(model1),2)
```

```
       1        2        3        4        5        6        7        8        9       10
  -58.73    38.27   -26.33    14.64    38.38   -25.09    -9.92    19.02   -20.31   -28.79
      11       12       13       14       15       16       17       18       19       20
   49.34    -3.99    62.03   -59.63    40.50   -39.40   -37.69    -2.04  -111.68   -42.48
      21       22       23       24       25       26       27       28       29       30
  -23.08    -5.55    73.05   -40.68   182.58   -23.38    89.38    -5.81   -36.94    -5.70
```

```
gala[which.max(resid(model1)),]
```

```
# A tibble: 1 x 10
     id island      species  area elevation nearest scruz adjacent stures  fits
  <dbl> <chr>         <dbl> <dbl>     <dbl>   <dbl> <dbl>    <dbl>  <dbl> <dbl>
1    25 SantaCruz       444  904.       864     0.6     0     0.52   4.38  261.
```

32.17 Relationship between Fitted and Observed Values

We've already seen that the `fitted` command can produce predicted values for each observations used to develop a regression model, and that the `resid` command can produce the residuals (observed - predicted) for those same observations. Returning to our original model1, let's compare the fitted values (stored earlier in `fits`) to the observed values.

```
ggplot(gala, aes(x = species, y = fits)) +
    geom_point(size = 3, shape = 1) +
    geom_abline(intercept = 0, slope = 1,
                col = "purple", linetype = "dashed") +
    labs(x = "Observed Species Count",
         y = "Model Predicted Species")
```

32.17.1 Questions about Fitted and Observed Values

65. Why did I draw the dotted purple line with y-intercept 0 and slope 1? Why is that particular line of interest?
66. If a point on this plot is in the top left here, above the dotted line, what does that mean?
67. If a point is below the dotted line here, what does that mean?
68. How does this plot display the size of an observation's residual?

32.18 Standardizing Residuals

We've already seen that the raw residuals from a regression model can be obtained using the `resid` function. Residuals are defined to have mean 0. This is one of the requirements of the least squares procedure for estimating a linear model, and their true standard deviation is effectively estimated using the residual standard error.

There are two additional types of residuals for us to be aware of: standardized residuals, and studentized (sometimes called externally standardized, or jackknife) residuals. Each approach standardizes the residuals by dividing them by a standard deviation estimate, so the resulting residuals should have mean 0 and standard deviation 1 if assumptions hold.

- **Standardized** residuals are the original (raw) residuals, scaled by a standard deviation estimate developed using the entire data set. This summary is part of what is provided by `augment` from the `broom` package, as `.std.resid`.
- **Studentized** residuals are the original (raw) residuals, scaled by a standard deviation estimate developed using the entire data set EXCEPT for this particular observation.

The `rstudent` function, when applied to a linear regression model, will the model's studentized residuals.

```
gala_aug <- gala_aug |>
    mutate(.stu.resid = rstudent(model1))

ggplot(gala_aug, aes(x = .std.resid,
                     y = .stu.resid)) +
    geom_point(size = 3, pch = 1) +
    geom_abline(intercept = 0, slope = 1, col = "red") +
    geom_label(x = 2.5, y = 1,
               label = "y = x", col = "red") +
    theme(aspect.ratio = 1) +
    labs(x = "Standardized Residuals for Model 1",
         y = "Studentized Residuals for Model 1")
```

32.18.1 Questions about Standardized and Studentized Residuals

69. From the plot above, what conclusions can you draw about the two methods of standardizing residuals as they apply in the case of our model1?

32.19 Influence Measures for Multiple Regression

R can output a series of **influence measures** for a regression model. Let me show you all of the available measures for model 1, but just for three of the data points - #1 (which is not particularly influential) and #12 and #16 (which are).

First, we'll look at the raw data:

```
gala[c(1,12,16),]
```

```
# A tibble: 3 x 10
     id island      species    area elevation nearest scruz adjacent stures  fits
  <dbl> <chr>         <dbl>   <dbl>     <dbl>   <dbl> <dbl>    <dbl>  <dbl> <dbl>
1     1 Baltra           58    25.1       346     0.6   0.6     1.84  -1.00  117.
2    12 Fernandina       93   634.       1494     4.3  95.3  4669.    -0.286  97.0
3    16 Isabela         347  4669.       1707     0.7  28.1   634.    -5.33  386.
```

And then, we'll gather the output available in the `influence.measures` function.

```
influence.measures(model1)
```

Here's an edited version of this output...

```
Influence measures of
lm(formula = species ~ area + elevation + nearest + scruz + adjacent,
data = gala) :
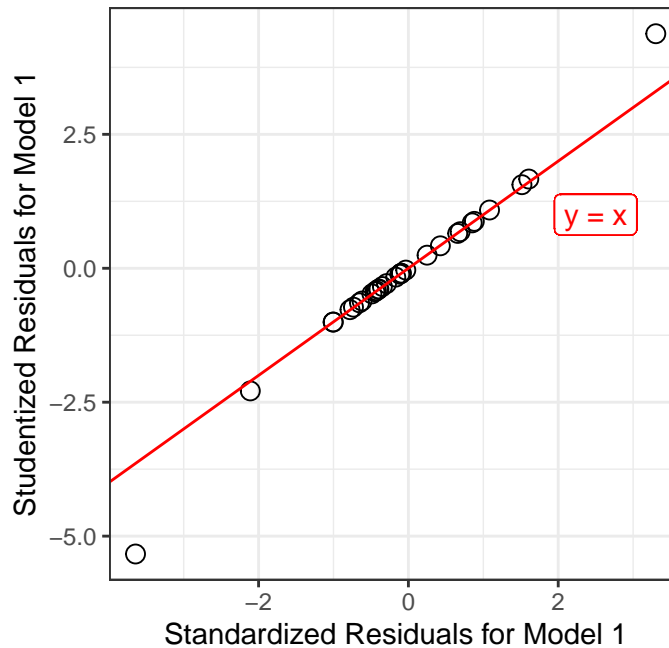
      dfb.1_   dfb.area  dfb.elvt dfb.nrst  dfb.scrz  dfb.adjc
1   -0.15064    0.13572 -0.122412  0.07684  0.084786  1.14e-01
12   0.16112    0.16395 -0.122578  0.03093 -0.059059 -8.27e-01
16  -1.18618  -20.87453  4.885852  0.36713 -1.022431 -8.09e-01

      dffit    cov.r    cook.d    hat inf
1    -0.29335  1.0835 1.43e-02 0.0787
12   -1.24249 25.1101 2.68e-01 0.9497    *
16  -29.59041  0.3275 6.81e+01 0.9685    *
```

This output presents dfbetas for each coefficient, followed by dffit statistics, covariance ratios, Cook's distance and leverage values (`hat`) along with an indicator of influence.

We'll consider each of these elements in turn.

32.20 DFBETAs

The first part of the influence measures output concerns what are generally called `dfbetas`
...

id	island	dfb.1_	dfb.area	dfb.elvt	dfb.nrst	dfb.scrz	dfb.adjc
1	Baltra	-0.151	0.136	-0.122	0.077	0.085	0.114
12	Fernandina	0.161	0.164	-0.123	0.031	-0.059	-0.827
16	Isabela	-1.186	-20.875	4.886	0.367	-1.022	-0.809

The `dfbetas` look at a standardized difference in the estimate of a coefficient (slope) that will occur if the specified point (here, `island`) is removed from the data set.

- Positive values indicate that deleting the point will yield a smaller coefficient.
- Negative values indicate that deleting the point will yield a larger coefficient.
- If the absolute value of the dfbeta is greater than $2/\sqrt{n}$, where n is the sample size, then the `dfbeta` is considered to be large.

In this case, our cutoff would be $2/\sqrt{30}$ or 0.365, so that the Isabela `dfbeta` values are all indicative of large influence. Essentially, if we remove Isabela from the data, and refit the model, our regression slopes will change a lot (see below). Fernandina has some influence as well, especially on the `adjacent` coefficient.

Predictor	Coefficient (p) **all 30 islands**	Coefficient (p) **without Isabela**
Intercept	7.07 ($p = 0.72$)	22.59 ($p = 0.11$)
area	-0.02 ($p = 0.30$)	0.30 ($p < 0.01$)
elevation	0.32 ($p < 0.01$)	0.14 ($p < 0.01$)
nearest	0.01 ($p = 0.99$)	-0.26 ($p = 0.73$)
scruz	-0.24 ($p = 0.28$)	-0.09 ($p = 0.55$)
adjacent	-0.08 ($p < 0.01$)	-0.07 ($p < 0.01$)

After the dfbetas, the `influence.measures` output presents `dffit`, covariance ratios, Cook's distance and leverage values (`hat`) for each observation, along with an indicator of influence.

```
id  island          dffit   cov.r   cook.d    hat inf
1   Baltra       -0.29335  1.0835 1.43e-02 0.0787
12  Fernandina  -1.24249 25.1101 2.68e-01 0.9497    *
16  Isabela     -29.59041  0.3275 6.81e+01 0.9685    *
```

32.21 Cook's d or Cook's Distance

The main measure of influence is Cook's Distance, also called Cook's d. Cook's d provides a summary of the influence of a particular point on all of the regression coefficients. It is a function of the standardized residual and the leverage. We've seen that the `augment()` function in the `broom` package provides both Cook's distance and leverage values for all observations in our model.

- Cook's distance values greater than 1 are generally indicators of high influence.
- Obviously, Isabela (with a value of Cook's d = 68.1) is a highly influential observation by this measure.

32.21.1 Plotting Cook's Distance

As one of its automated regression diagnostic plots, R will produce an index plot of the Cook's distance values. Note the relatively enormous influence for island 16 (Isabela).

```
plot(model1, which = 4)
```

Cook's distance

lm(species ~ area + elevation + nearest + scruz + adjacent)

32.22 DFFITS

A similar measure to Cook's distance is called `DFFITS`. The `DFFITS` value describes the influence of the point on the fitted value. It's the number of standard deviations that the fitted value changes if the observation is removed. This is defined as a function of the studentized residual and the leverage.

- If the absolute value of `DFFITS` is greater than 2 times $\sqrt{p/n-p}$, where p is the number of predictors (not including the intercept), we deem the observation influential.
- For the `gala` data, we'd consider any point with `DFFITS` greater than 2 x $\sqrt{5/(30-5)}$ = 0.894 to be influential by this standard, since n = 30 and we are estimating p = 5 slopes in our model. This is true of both Fernandina and Isabela.

32.23 Covariance Ratio

The covariance ratio `cov.r` indicates the role of the observation on the precision of estimation. If `cov.r` is greater than 1, then this observation improves the precision, overall, and if it's less than 1, the observation drops the precision of estimation, and these are the points about which we'll be most concerned.

- As with most of our other influence measures, Isabela appears to be a concern.

649

32.24 Leverage

The `hat` value is a measure of leverage. Specifically, this addresses whether or not the point in question is unusual in terms of its combination of predictor values. We've seen that the `augment()` function in the `broom` package provides leverage values for all observations in our model.

- The usual cutoff for a large leverage value is 2.5 times the average leverage across all observations, where the average leverage is equal to k/n, where n is the number of observations included in the regression model, and k is the number of model coefficients (slopes plus intercept).
- In the `gala` example, we'd regard any observation with a hat value larger than 2.5 x 6/30 = 0.5 to have large leverage. This includes Fernandina and Isabela.

32.24.1 Indicator of Influence

The little asterisk indicates an observation which is influential according to R's standards for any of these measures. You can take the absence of an asterisk as a clear indication that a point is NOT influential. Points with asterisks may or may not be influential in an important way. In practice, I usually focus on the Cook's distance to make decisions about likely influence, when the results aren't completely clear.

32.25 Building Predictions from our models

The `predict` function, when applied to a linear regression model, produces the fitted values, just as the `augment()` function did, and, as we've seen, it can be used to generate *prediction* intervals for a single new observation, or *confidence* intervals for a group of new observations with the same predictor values.

Let us, just for a moment, consider a "typical" island, exemplified by the median value of all the predictors[2]. There's a trick to creating this and dumping it in a vector I will call `x.medians`.

```
x <- model.matrix(model1)
x.medians <- apply(x, 2, function(x) median(x))
x.medians
```

```
(Intercept)        area    elevation      nearest        scruz     adjacent
       1.00        2.59       192.00         3.05        46.65         2.59
```

---

[2]This approach is motivated by @Faraway2015, pp. 52-53.

We want to use the model to predict our outcome (species) on the basis of the inputs above: a new island with values of all predictors equal to the median of the existing islands. As before, building an interval forecast around a fitted value requires us to decide whether we are:

- predicting the number of species for one particular island with the specified characteristics (in which case we use something called a prediction interval) or
- predicting the mean number of species across all islands that have the specified characteristics (in which case we use the confidence interval).

```
newdata <- data.frame(t(x.medians))
predict(model1, newdata, interval="prediction", level = 0.95)
```

```
       fit       lwr       upr
1 56.95714 -72.06155 185.9758
```

```
predict(model1, newdata, interval="confidence", level = 0.95)
```

```
       fit      lwr      upr
1 56.95714 28.52381 85.39048
```

32.25.1 Questions about the Prediction and Confidence Interval Methods

70. What is the 95% prediction interval for this new observation? Does that make sense?
71. Which interval (prediction or confidence) is wider? Does that make sense?
72. Is there an island that has characteristics that match our new medians variable?
73. What happens if we don't specify new data in making a prediction?

32.26 Making a Prediction with New Data (without `broom`)

74. How does the output below help us to make a prediction with a new data point, or series of them? Interpret the resulting intervals.

```
newdata2 <- tibble(area = 2, elevation = 100, nearest = 3,
                    scruz = 5, adjacent = 1)
predict(model1, newdata2, interval="prediction", level = 0.95)
```

```
       fit       lwr      upr
1 37.71683 -92.46825 167.9019
```

```
predict(model1, newdata2, interval="confidence", level = 0.95)
```

```
       fit      lwr     upr
1 37.71683 4.388351 71.0453
```

32.27 Scaling Predictors using Z Scores: Semi-Standardized Coefficients

We know that the interpretation of the coefficients in a regression model is sensitive to the scale of the predictors. We have already seen how to "standardize" each predictor by subtracting its mean and dividing by its standard deviation.

- Each coefficient in this semi-standardized model has the following interpretation: the expected difference in the outcome, comparing units (subjects) that differ by one standard deviation in the variable of interest, but for which all other variables are fixed at their average.
- Remember also that the intercept in such a model shows the mean outcome across all subjects.

Consider a two-variable model, using `area` and `elevation` to predict the number of `species`…

```
model2 <- lm(species ~ area + elevation, data=gala)
summary(model2)
```

```
Call:
lm(formula = species ~ area + elevation, data = gala)

Residuals:
     Min       1Q   Median       3Q      Max
-192.619  -33.534  -19.199    7.541  261.514

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 17.10519   20.94211   0.817  0.42120
area         0.01880    0.02594   0.725  0.47478
elevation    0.17174    0.05317   3.230  0.00325 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 79.34 on 27 degrees of freedom
Multiple R-squared:  0.554, Adjusted R-squared:  0.521
F-statistic: 16.77 on 2 and 27 DF,  p-value: 1.843e-05
```

Now compare these results to the ones we get after scaling the area and elevation variables. Remember that the **scale** function centers a variable on zero by subtracting the mean from each observation, and then scales the result by dividing by the standard deviation. This ensures that each regression input has mean 0 and standard deviation 1, and is thus a *z score*.

```
model2.z <- lm(species ~ scale(area) + scale(elevation),
               data=gala)
summary(model2.z)
```

```
Call:
lm(formula = species ~ scale(area) + scale(elevation), data = gala)

Residuals:
    Min      1Q   Median      3Q      Max
-192.619  -33.534  -19.199   7.541  261.514

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)         85.23      14.48   5.884 2.87e-06 ***
scale(area)         16.25      22.42   0.725  0.47478
scale(elevation)    72.41      22.42   3.230  0.00325 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 79.34 on 27 degrees of freedom
Multiple R-squared:  0.554, Adjusted R-squared:  0.521
F-statistic: 16.77 on 2 and 27 DF,  p-value: 1.843e-05
```

32.27.1 Questions about the Semi-Standardized Model

75. What changes after centering and rescaling the predictors, and what does not?
76. Why might rescaling like this be a helpful thing to do if you want to compare predictors in terms of importance?

32.28 Fully Standardized Regression Coefficients

Suppose we standardize the coefficients by also taking centering and scaling (using the z score) the outcome variable: `species`, creating a **fully standardized** model.

```
model2.zout <- lm(scale(species) ~
                     scale(area) + scale(elevation), data=gala)
summary(model2.zout)
```

```
Call:
lm(formula = scale(species) ~ scale(area) + scale(elevation),
    data = gala)

Residuals:
     Min       1Q   Median       3Q      Max
-1.68031 -0.29253 -0.16749  0.06578  2.28131

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)      4.594e-17  1.264e-01   0.000  1.00000
scale(area)      1.418e-01  1.956e-01   0.725  0.47478
scale(elevation) 6.316e-01  1.956e-01   3.230  0.00325 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6921 on 27 degrees of freedom
Multiple R-squared:  0.554, Adjusted R-squared:  0.521
F-statistic: 16.77 on 2 and 27 DF,  p-value: 1.843e-05
```

32.28.1 Questions about the Standardized Model

77. How do you interpret the value 0.142 of the `scale(area)` coefficient here? You may want to start by reviewing the summary of the original `gala` data shown here.

```
summary(gala[c("species", "area", "elevation")])
```

```
   species           area            elevation
 Min.   :  2.00   Min.   :  0.010   Min.   :  25.00
 1st Qu.: 13.00   1st Qu.:  0.258   1st Qu.:  97.75
```

```
Median : 42.00    Median :    2.590    Median : 192.00
Mean   : 85.23    Mean   :  261.709    Mean   : 368.03
3rd Qu.: 96.00    3rd Qu.:   59.237    3rd Qu.: 435.25
Max.   :444.00    Max.   :4669.320    Max.   :1707.00
```

78. How do you interpret the value 0.632 of the `scale(elevation)` coefficient in the standardized model?
79. What is the intercept in this setting? Will this be the case whenever you scale like this?
80. What are some of the advantages of looking at scaled regression coefficients?
81. Why are these called *fully* standardized coefficients while the previous page described semi-standardized coefficients?
82. What would motivate you to use one of these two methods of standardization (fully standardized or semi-standardized) vs. the other?

32.29 Robust Standardization of Regression Coefficients

Another common option for scaling is to specify lower and upper comparison points, perhaps by comparing the impact of a move from the 25th to the 75th percentile for each variable, while holding all of the other variables constant.

Occasionally, you will see robust semi-standardized regression coefficients, which measure the increase in the outcome, Y, associated with an increase in that particular predictor of one IQR (inter-quartile range).

```
gala$area.scaleiqr <-
    (gala$area - mean(gala$area)) / IQR(gala$area)
gala$elevation.scaleiqr <-
    (gala$elevation - mean(gala$elevation)) /
                        IQR(gala$elevation)

model2.iqr <- lm(species ~
                    area.scaleiqr + elevation.scaleiqr,
                data=gala)
summary(model2.iqr)
```

```
Call:
lm(formula = species ~ area.scaleiqr + elevation.scaleiqr, data = gala)

Residuals:
    Min      1Q   Median      3Q      Max
```

```
-192.619  -33.534  -19.199    7.541  261.514
```

```
Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)         85.233     14.485   5.884 2.87e-06 ***
area.scaleiqr        1.109      1.530   0.725  0.47478
elevation.scaleiqr  57.963     17.946   3.230  0.00325 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 79.34 on 27 degrees of freedom
Multiple R-squared:  0.554, Adjusted R-squared:  0.521
F-statistic: 16.77 on 2 and 27 DF,  p-value: 1.843e-05
```

32.29.1 Questions about Robust Standardization

83. How should we interpret the 57.96 value for the scaled `elevation` variable? You may want to start by considering the summary of the original elevation data below.

```
summary(gala$elevation)
```

```
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 25.00   97.75  192.00  368.03  435.25 1707.00
```

A **robust standardized coefficient** analysis measures the increase in Y (in IQR of Y) associated with an increase in the predictor of interest of one IQR.

```
gala$species.scaleiqr <- (gala$species - mean(gala$species)) / IQR(gala$species)
model2.iqrout <- lm(species.scaleiqr ~ area.scaleiqr + elevation.scaleiqr, data=gala)
model2.iqrout
```

```
Call:
lm(formula = species.scaleiqr ~ area.scaleiqr + elevation.scaleiqr,
    data = gala)

Coefficients:
       (Intercept)       area.scaleiqr  elevation.scaleiqr
        -1.013e-16           1.336e-02           6.983e-01
```

84. What can we learn from the R output above?

32.30 Scaling Inputs by Dividing by 2 Standard Deviations

It turns out that standardizing the inputs to a regression model by dividing by a standard deviation creates some difficulties when you want to include a binary predictor in the model.

Instead, Andrew Gelman recommends that you consider centering all of the predictors (binary or continuous) by subtracting off the mean, and then, for the non-binary predictors, also dividing not by one, but rather by two standard deviations.

- Such a standardization can go a long way to helping us understand a model whose predictors are on different scales, and provides an interpretable starting point.
- Another appealing part of this approach is that in the **arm** library, Gelman and his colleagues have created an R function called `standardize`, which can be used to automate the process of checking coefficients that have been standardized in this manner, after the regression model has been fit.

```
model2
```

```
Call:
lm(formula = species ~ area + elevation, data = gala)

Coefficients:
(Intercept)          area     elevation
    17.1052        0.0188        0.1717
```

```
arm::standardize(model2)
```

```
Call:
lm(formula = species ~ z.area + z.elevation, data = gala)

Coefficients:
(Intercept)        z.area   z.elevation
      85.23         32.50        144.81
```

32.30.1 Questions about Standardizing by Dividing by Two SD

85. How does this result compare to the semi-standardized regression coefficients we have seen on the last few analyses?

86. How should we interpret the `z.area` coefficient of 32.5 here? Again, you may want to start by obtaining a statistical summary of the original `area` data, as shown below.

```
summary(gala$area)
```

```
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
   0.010    0.258    2.590  261.709  59.238 4669.320
```

To standardize the outcome in this way, as well, we use:

```
arm::standardize(model2, standardize.y=TRUE)
```

```
Call:
lm(formula = z.species ~ z.area + z.elevation, data = gala)

Coefficients:
(Intercept)       z.area   z.elevation
  1.653e-19    1.418e-01     6.316e-01
```

87. How should we interpret the `z.area` coefficient of 0.142 here?
88. How does these relate to the standardized regression coefficients we've seen before?

Again, if you have questions about how to interpret the output in light of the questions contained in this Chapter, please ask us for help on Piazza or in TA office hours.

33 Introduction to Non-linear Terms

33.1 Setup: Packages Used Here

```r
knitr::opts_chunk$set(comment = NA)

library(broom)
library(car)
library(Hmisc)
library(rms)
library(mosaic)
library(patchwork)
library(tidyverse)


theme_set(theme_bw())
```

33.2 The `pollution` data

Consider the `pollution` data set, which contain 15 independent variables and a measure of mortality, describing 60 US metropolitan areas in 1959-1961. The data come from McDonald and Schwing (1973), and are available at http://www4.stat.ncsu.edu/~boos/var.select/pollution.html and our web site.

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

Here's a codebook:

Variable	Description
y	Total Age Adjusted Mortality Rate
x1	Mean annual precipitation in inches
x2	Mean January temperature in degrees Fahrenheit
x3	Mean July temperature in degrees Fahrenheit
x4	Percent of 1960 SMSA population that is 65 years of age or over

Variable	Description
x5	Population per household, 1960 SMSA
x6	Median school years completed for those over 25 in 1960 SMSA
x7	Percent of housing units that are found with facilities
x8	Population per square mile in urbanized area in 1960
x9	Percent of 1960 urbanized area population that is non-white
x10	Percent employment in white-collar occupations in 1960 urbanized area
x11	Percent of families with income under 3; 000 in 1960 urbanized area
x12	Relative population potential of hydrocarbons, HC
x13	Relative pollution potential of oxides of nitrogen, NOx
x14	Relative pollution potential of sulfur dioxide, SO2
x15	Percent relative humidity, annual average at 1 p.m.

33.3 A simple linear model

Consider the relationship between `y`, the age-adjusted mortality rate, and `x2`, the mean January temperature, across these 60 areas. I'll include both a linear model (in blue) and a loess smooth (in red.) Does the relationship appear to be linear?

```
ggplot(pollution, aes(x = x2, y = y)) +
    geom_point() +
    geom_smooth(method = "lm", col = "blue",
                formula = y ~ x, se = F) +
    geom_smooth(method = "loess", col = "red",
                formula = y ~ x, se = F)
```
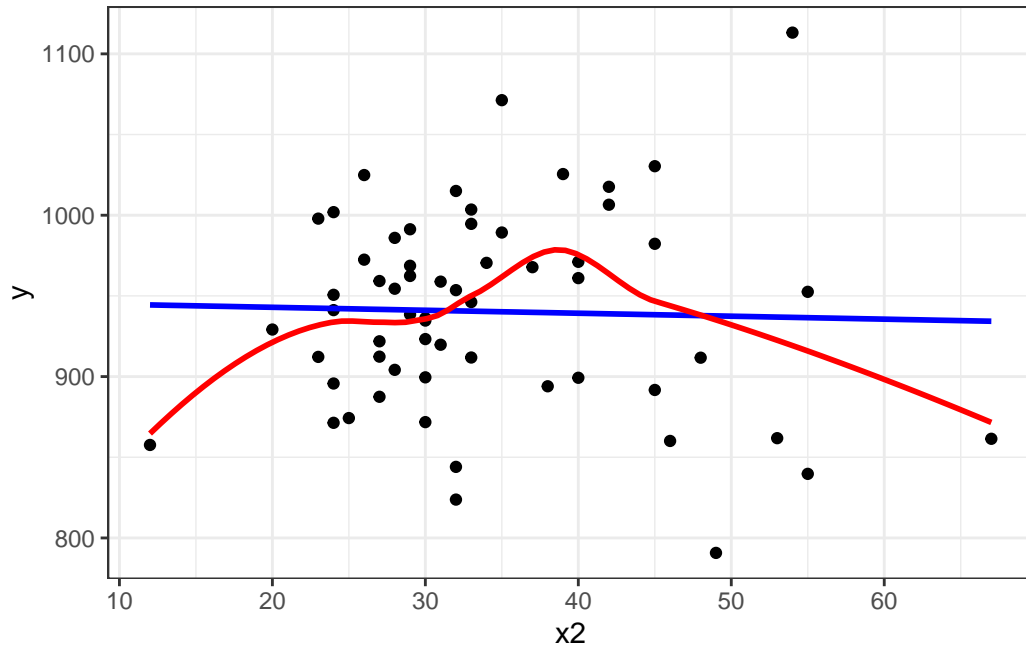
Suppose we plot the residuals that emerge from the linear model shown in blue, above. Do we see a curve in a plot of residuals against fitted values?

```
plot(lm(y ~ x2, data = pollution), which = 1)
```

Residuals vs Fitted

Fitted values
lm(y ~ x2)
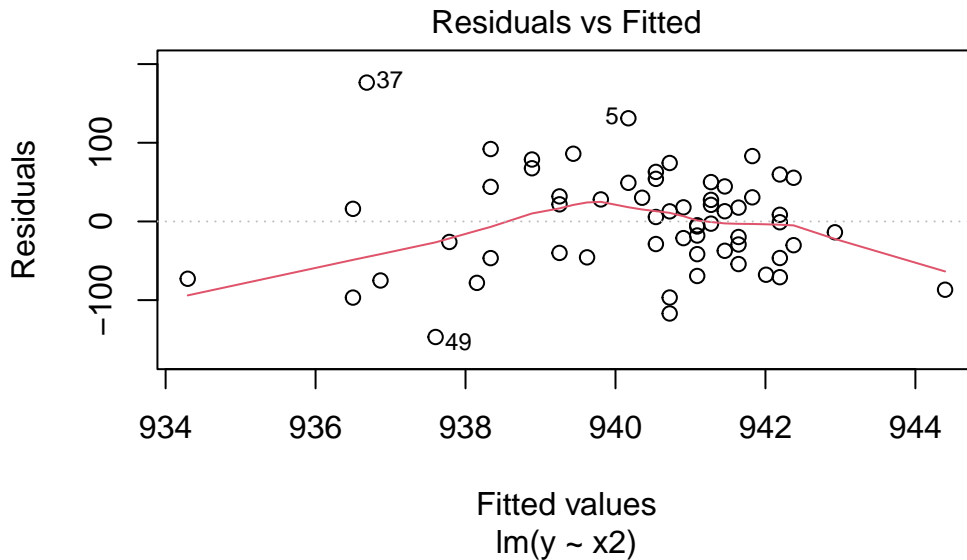
33.4 Quadratic polynomial model'

A polynomial in the variable x of degree D is a linear combination of the powers of x up to D.

For example:

- Linear: $y = \beta_0 + \beta_1 x$
- Quadratic: $y = \beta_0 + \beta_1 x + \beta_2 x^2$
- Cubic: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$
- Quartic: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4$
- Quintic: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4 + \beta_5 x^5$

Fitting such a model creates a **polynomial regression**.

33.4.1 The raw quadratic model

Let's look at a **quadratic model** which predicts y using x2 and the square of x2, so that our model is of the form:

$$y = \beta_0 + \beta_1 x_2 + \beta_2 x_2^2 + error$$

662

There are several ways to fit this exact model.

- One approach is to calculate the square of x2 within our `pollution` data set, and then feed both x2 and x2squared to `lm`.
- Another approach uses the I function within our `lm` to specify the use of both x2 and its square.
- Yet another approach uses the `poly` function within our `lm`, which can be used to specify raw models including x2 and x2squared.

```
pollution <- pollution |> mutate(x2squared = x2^2)

mod2a <- lm(y ~ x2 + x2squared, data = pollution)
mod2b <- lm(y ~ x2 + I(x2^2), data = pollution)
mod2c <- lm(y ~ poly(x2, degree = 2, raw = TRUE), data = pollution)
```

Each of these approaches produces the same model, as they are just different ways of expressing the same idea.

```
summary(mod2a)
```

```
Call:
lm(formula = y ~ x2 + x2squared, data = pollution)

Residuals:
    Min       1Q   Median       3Q      Max
-148.977  -38.651    6.889   35.312  189.346

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 785.77449   79.54086   9.879 5.87e-14 ***
x2            8.87640    4.27394   2.077   0.0423 *
x2squared    -0.11704    0.05429  -2.156   0.0353 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 60.83 on 57 degrees of freedom
Multiple R-squared:  0.07623,    Adjusted R-squared:  0.04382
F-statistic: 2.352 on 2 and 57 DF,  p-value: 0.1044
```

And if we plot the fitted values for this `mod2` using whatever approach you like, we get exactly the same result.

```
mod2a_aug <- augment(mod2a, pollution)

ggplot(mod2a_aug, aes(x = x2, y = y)) +
    geom_point() +
    geom_line(aes(x = x2, y = .fitted), col = "red") +
    labs(title = "Model 2a: Quadratic fit using x2 and x2^2")
```

Model 2a: Quadratic fit using x2^2
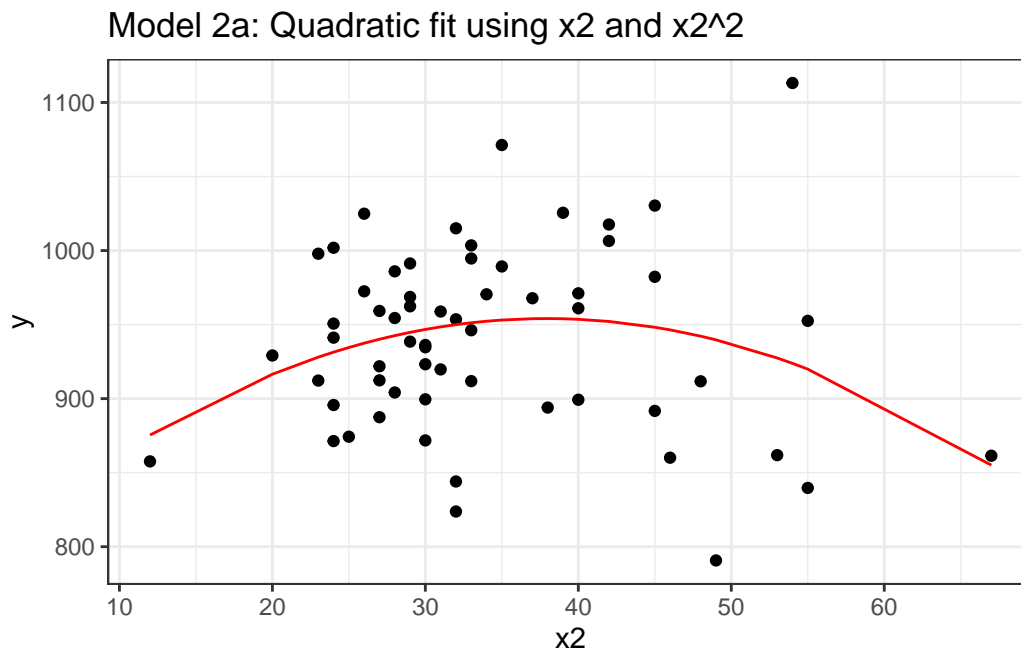

```
mod2b_aug <- augment(mod2b, pollution)

mod2c_aug <- augment(mod2c, pollution)

p1 <- ggplot(mod2b_aug, aes(x = x2, y = y)) +
    geom_point() +
    geom_line(aes(x = x2, y = .fitted), col = "red") +
    labs(title = "Model 2b: Quadratic fit")

p2 <- ggplot(mod2c_aug, aes(x = x2, y = y)) +
    geom_point() +
    geom_line(aes(x = x2, y = .fitted), col = "blue") +
    labs(title = "Model 2c: Quadratic fit")
```

```
p1 + p2
```
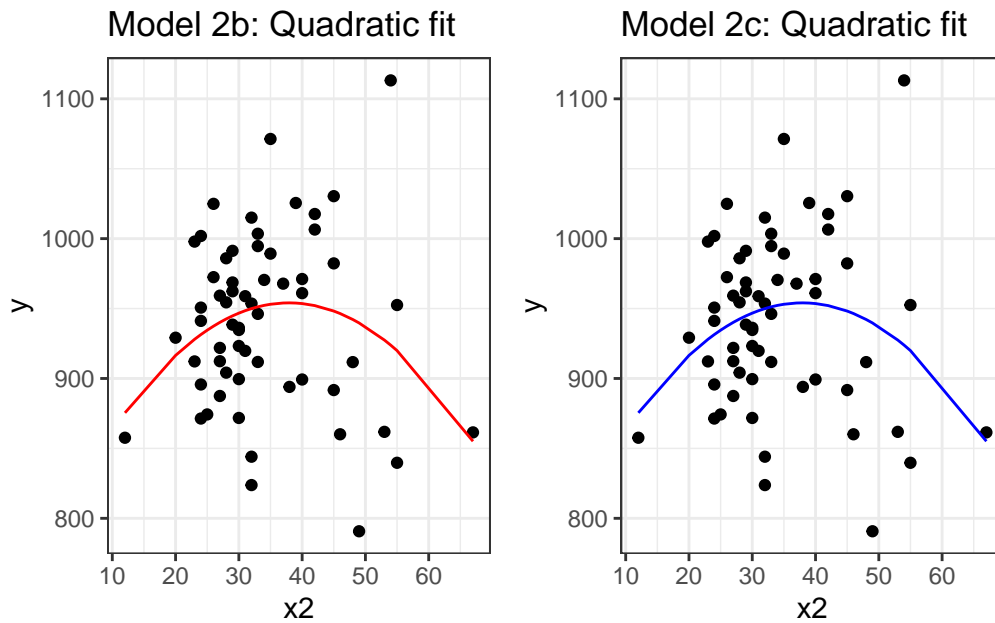

33.4.2 Raw quadratic fit after centering

Sometimes, we'll center (and perhaps rescale, too) the x2 variable before including it in a quadratic fit like this.

```
pollution <- pollution |>
    mutate(x2_c = x2 - mean(x2))

mod2d <- lm(y ~ x2_c + I(x2_c^2), data = pollution)

summary(mod2d)
```

```
Call:
lm(formula = y ~ x2_c + I(x2_c^2), data = pollution)

Residuals:
     Min       1Q   Median       3Q      Max
-148.977  -38.651    6.889   35.312  189.346
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 952.25941    9.59896  99.204   <2e-16 ***
x2_c          0.92163    0.93237   0.988   0.3271
I(x2_c^2)    -0.11704    0.05429  -2.156   0.0353 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 60.83 on 57 degrees of freedom
Multiple R-squared:  0.07623,    Adjusted R-squared:  0.04382
F-statistic: 2.352 on 2 and 57 DF,  p-value: 0.1044
```

Note that this model looks very different, with the exception of the second order quadratic term. But, it produces the same fitted values as the models we fit previously.

```
mod2d_aug <- augment(mod2d, pollution)

ggplot(mod2d_aug, aes(x = x2, y = y)) +
    geom_point() +
    geom_line(aes(x = x2, y = .fitted), col = "red") +
    labs(title = "Model 2d: Quadratic fit using centered x2 and x2^2")
```


Model 2d: Quadratic fit using centered x2 and x2^2
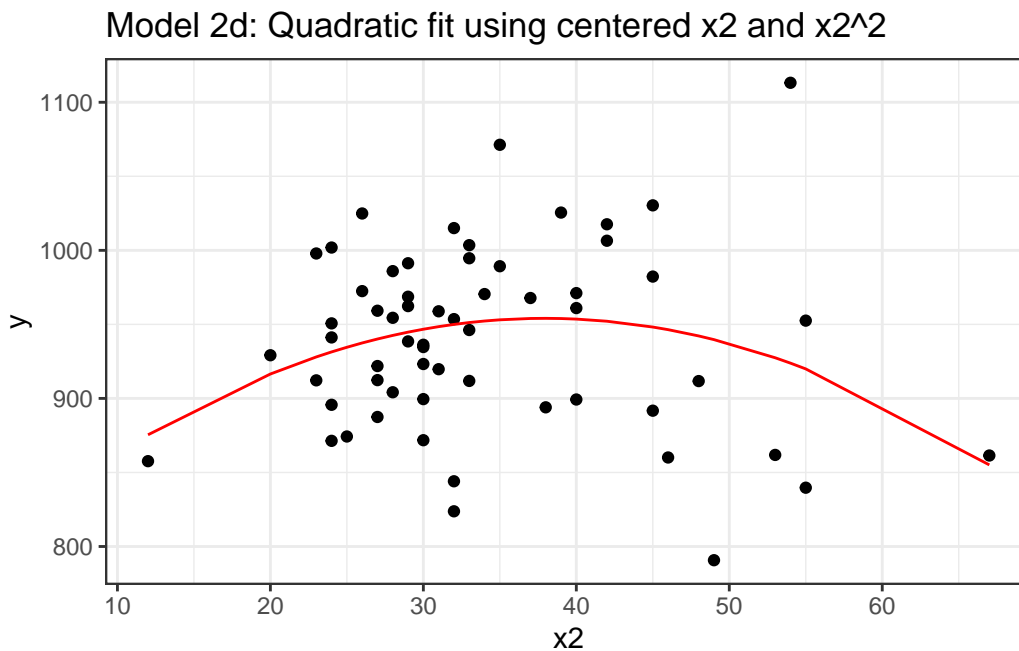
Or, if you don't believe me yet, look at the four sets of fitted values another way.

```
mosaic::favstats(~ .fitted, data = mod2a_aug)
```

```
     min       Q1  median       Q3     max     mean       sd  n missing
855.1041 936.7155 945.597 950.2883 954.073 940.3585 17.17507 60        0
```

```
mosaic::favstats(~ .fitted, data = mod2b_aug)
```

```
     min       Q1  median       Q3     max     mean       sd  n missing
855.1041 936.7155 945.597 950.2883 954.073 940.3585 17.17507 60        0
```

```
mosaic::favstats(~ .fitted, data = mod2c_aug)
```

```
     min       Q1  median       Q3     max     mean       sd  n missing
855.1041 936.7155 945.597 950.2883 954.073 940.3585 17.17507 60        0
```

```
mosaic::favstats(~ .fitted, data = mod2d_aug)
```

```
     min       Q1  median       Q3     max     mean       sd  n missing
855.1041 936.7155 945.597 950.2883 954.073 940.3585 17.17507 60        0
```

33.5 Orthogonal Polynomials

Now, let's fit an orthogonal polynomial of degree 2 to predict y using x2.

```
mod2_orth <- lm(y ~ poly(x2, 2), data = pollution)

summary(mod2_orth)
```

```
Call:
lm(formula = y ~ poly(x2, 2), data = pollution)

Residuals:
    Min      1Q  Median      3Q     Max
```

```
-148.977  -38.651     6.889    35.312  189.346
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    940.358      7.853 119.746   <2e-16 ***
poly(x2, 2)1   -14.345     60.829  -0.236   0.8144
poly(x2, 2)2  -131.142     60.829  -2.156   0.0353 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 60.83 on 57 degrees of freedom
Multiple R-squared:  0.07623,   Adjusted R-squared:  0.04382
F-statistic: 2.352 on 2 and 57 DF,  p-value: 0.1044
```

Now this looks very different in the equation, but, again, we can see that this produces exactly the same fitted values as our previous models, and the same model fit summaries. Is it, in fact, the same model? Here, we'll plot the fitted Model 2a in a red line, and this new Model 2 with Orthogonal Polynomials as blue points.

```
mod2orth_aug <- augment(mod2_orth, pollution)

ggplot(mod2orth_aug, aes(x = x2, y = y)) +
    geom_point() +
    geom_point(aes(x = x2, y = .fitted),
               col = "blue", size = 2) +
    geom_line(data = mod2a_aug, aes(x = x2, y = .fitted),
              col = "red") +
    labs(title = "Model 2 with Orthogonal Polynomial, degree 2")
```

Model 2 with Orthogonal Polynomial, degree 2
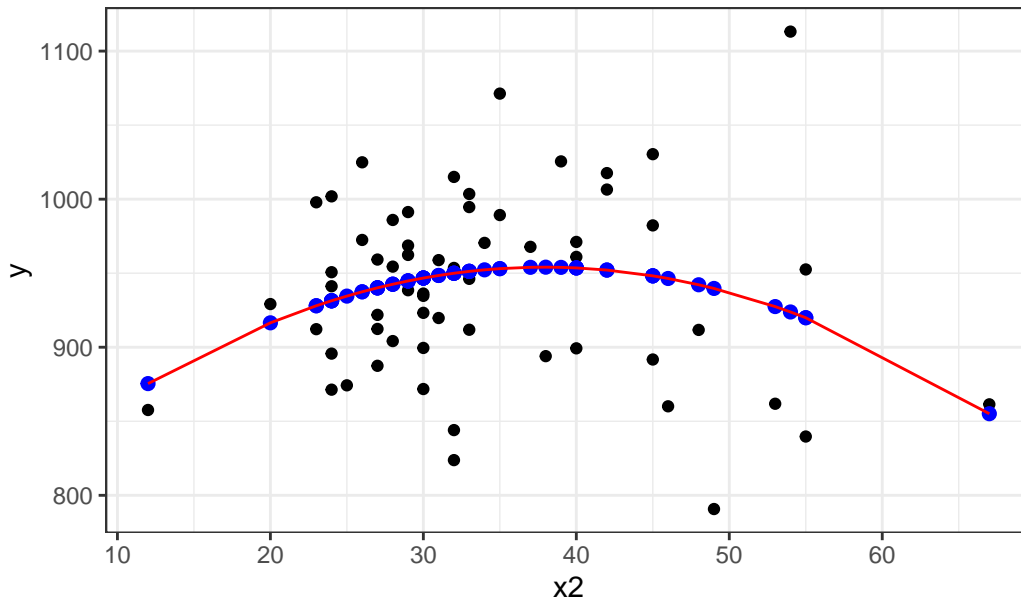
Yes, it is again the same model in terms of the predictions it makes for `y`.

By default, with `raw = FALSE`, the `poly()` function within a linear model computes what is called an **orthogonal polynomial**. An orthogonal polynomial sets up a model design matrix using the coding we've seen previously: `x2` and `x2^2` in our case, and then scales those columns so that each column is **orthogonal** to the previous ones. This eliminates the collinearity (correlation between predictors) and lets our t tests tell us whether the addition of any particular polynomial term improves the fit of the model over the lower orders.

Would the addition of a cubic term help us much in predicting `y` from `x2`?

```
mod3 <- lm(y ~ poly(x2, 3), data = pollution)
summary(mod3)
```

```
Call:
lm(formula = y ~ poly(x2, 3), data = pollution)

Residuals:
     Min       1Q   Median       3Q      Max
-146.262  -39.679    5.569   35.984  191.536

Coefficients:
```

```
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    940.358       7.917 118.772   <2e-16 ***
poly(x2, 3)1   -14.345      61.328  -0.234   0.8159
poly(x2, 3)2  -131.142      61.328  -2.138   0.0369 *
poly(x2, 3)3    16.918      61.328   0.276   0.7837
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 61.33 on 56 degrees of freedom
Multiple R-squared:  0.07748,    Adjusted R-squared:  0.02806
F-statistic: 1.568 on 3 and 56 DF,   p-value: 0.2073
```

It doesn't appear that the cubic term adds much here, if anything. The p value is not significant for the third degree polynomial, the summaries of fit quality aren't much improved, and as we can see from the plot below, the predictions don't actually change all that much.

```
mod3_aug <- augment(mod3, pollution)

ggplot(mod3_aug, aes(x = x2, y = y)) +
    geom_point() +
    geom_line(aes(x = x2, y = .fitted),
              col = "blue") +
    geom_line(data = mod2orth_aug, aes(x = x2, y = .fitted),
              col = "red") +
    labs(title = "Quadratic (red) vs. Cubic (blue) Polynomial Fits")
```

Quadratic (red) vs. Cubic (blue) Polynomial Fits
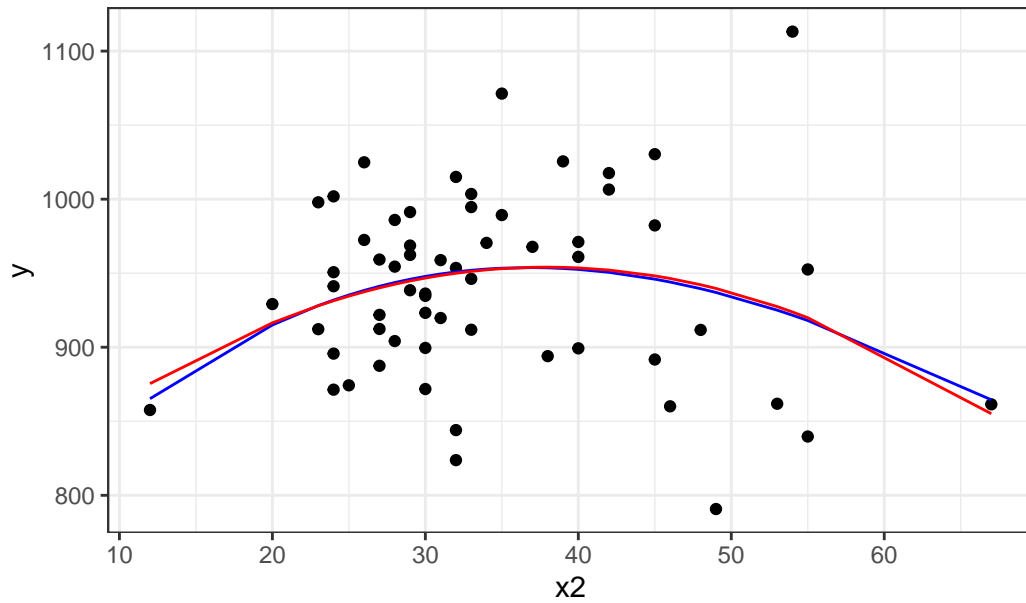
33.6 A Cubic Polynomial Model

What if we consider another predictor instead? Let's look at `x3`, the Mean July temperature in degrees Fahrenheit. Here is the `loess` smooth.

```
ggplot(pollution, aes(x = x3, y = y)) +
    geom_point() +
    geom_smooth(method = "loess",
                formula = y ~ x, se = TRUE)
```
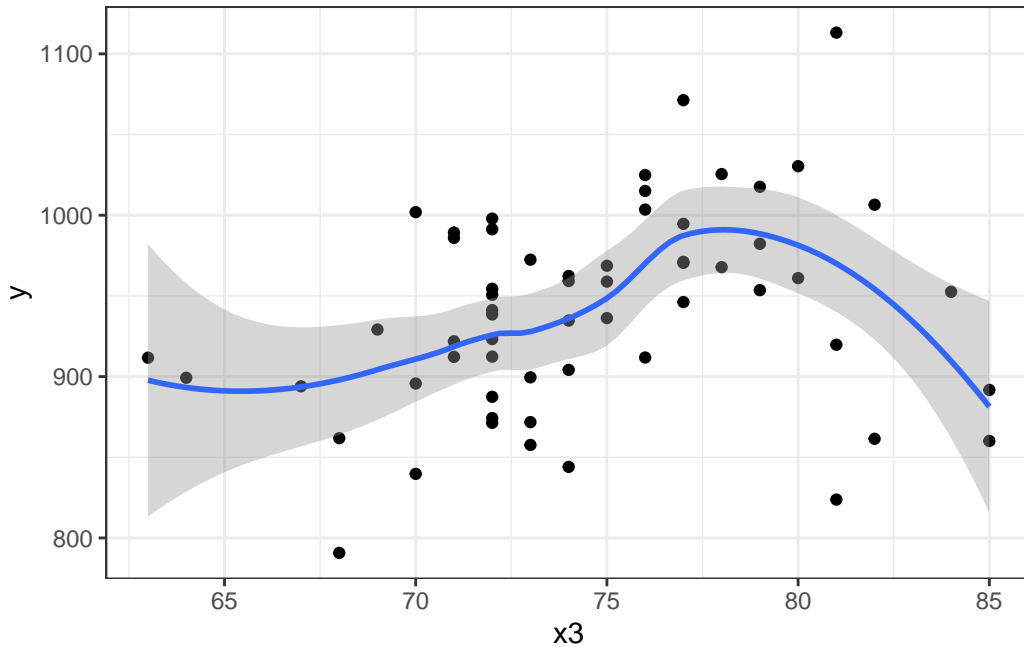
That looks pretty curvy - perhaps we need a more complex polynomial. We'll consider a linear model (`mod4_L`), a quadratic fit (`mod4_Q`) and a polynomial of degree 3: a **cubic** fit (`mod_4C`)

```
mod4_L <- lm(y ~ x3, data = pollution)
summary(mod4_L)
```

```
Call:
lm(formula = y ~ x3, data = pollution)

Residuals:
     Min       1Q   Median       3Q      Max
-139.813  -34.341    4.271   38.197  149.587

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  670.529    123.140   5.445  1.1e-06 ***
x3             3.618      1.648   2.196   0.0321 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 60.29 on 58 degrees of freedom
```

```
Multiple R-squared:  0.07674,    Adjusted R-squared:  0.06082
F-statistic: 4.821 on 1 and 58 DF,  p-value: 0.03213
```

```r
mod4_Q <- lm(y ~ poly(x3, 2), data = pollution)
summary(mod4_Q)
```

```
Call:
lm(formula = y ~ poly(x3, 2), data = pollution)

Residuals:
     Min       1Q   Median       3Q      Max
-132.004  -42.184    4.069   47.126  157.396

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    940.358      7.553 124.503   <2e-16 ***
poly(x3, 2)1   132.364     58.504   2.262   0.0275 *
poly(x3, 2)2  -125.270     58.504  -2.141   0.0365 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 58.5 on 57 degrees of freedom
Multiple R-squared:  0.1455,    Adjusted R-squared:  0.1155
F-statistic: 4.852 on 2 and 57 DF,  p-value: 0.01133
```

```r
mod4_C <- lm(y ~ poly(x3, 3), data = pollution)
summary(mod4_C)
```

```
Call:
lm(formula = y ~ poly(x3, 3), data = pollution)

Residuals:
     Min       1Q   Median       3Q      Max
-148.004  -29.998    1.441   34.579  141.396

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    940.358      7.065 133.095  < 2e-16 ***
```

```
poly(x3, 3)1  132.364      54.728   2.419  0.01886 *
poly(x3, 3)2 -125.270      54.728  -2.289  0.02588 *
poly(x3, 3)3 -165.439      54.728  -3.023  0.00377 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 54.73 on 56 degrees of freedom
Multiple R-squared:  0.2654,    Adjusted R-squared:  0.226
F-statistic: 6.742 on 3 and 56 DF,  p-value: 0.0005799
```

It looks like the cubic polynomial term is of some real importance here. Do the linear, quadratic and cubic model fitted values look different?

```
mod4_L_aug <- augment(mod4_L, pollution)

mod4_Q_aug <- augment(mod4_Q, pollution)

mod4_C_aug <- augment(mod4_C, pollution)

ggplot(pollution, aes(x = x3, y = y)) +
    geom_point() +
    geom_line(data = mod4_L_aug, aes(x = x3, y = .fitted),
              col = "blue", size = 1.25) +
    geom_line(data = mod4_Q_aug, aes(x = x3, y = .fitted),
              col = "black", size = 1.25) +
    geom_line(data = mod4_C_aug, aes(x = x3, y = .fitted),
              col = "red", size = 1.25) +
    geom_text(x = 66, y = 930, label = "Linear Fit", col = "blue") +
    geom_text(x = 64, y = 820, label = "Quadratic Fit", col = "black") +
    geom_text(x = 83, y = 900, label = "Cubic Fit", col = "red") +
    labs(title = "Linear, Quadratic and Cubic Fits predicting y with x3")
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```

Linear, Quadratic and Cubic Fits predicting y with x3
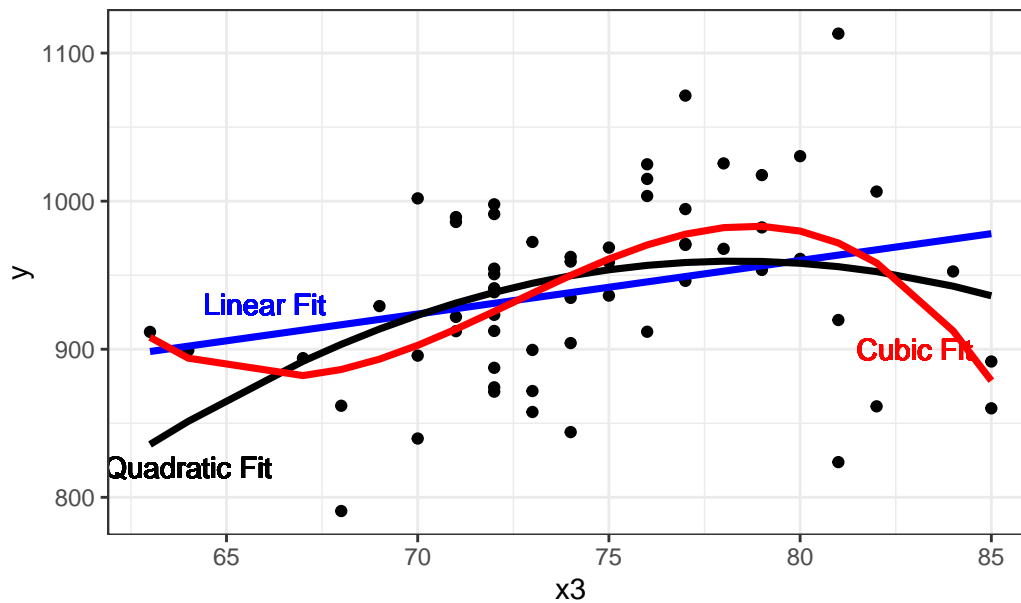
33.7 A restricted cubic spline

- A **linear spline** is a continuous function formed by connecting points (called **knots** of the spline) by line segments.
- A **restricted cubic spline** is a way to build highly complicated curves into a regression equation in a fairly easily structured way.
- A restricted cubic spline is a series of polynomial functions joined together at the knots.

  - Such a spline gives us a way to flexibly account for non-linearity without over-fitting the model.
  - Restricted cubic splines can fit many different types of non-linearities.
  - Specifying the number of knots is all you need to do in R to get a reasonable result from a restricted cubic spline.

The most common choices are 3, 4, or 5 knots. Each additional knot adds to the non-linearity, and spends an additional degree of freedom:

- 3 Knots, 2 degrees of freedom, allows the curve to "bend" once.
- 4 Knots, 3 degrees of freedom, lets the curve "bend" twice.
- 5 Knots, 4 degrees of freedom, lets the curve "bend" three times.

For most applications, three to five knots strike a nice balance between complicating the model needlessly and fitting data pleasingly. Let's consider a restricted cubic spline model for our y based on x3 again, but now with:

- in mod5a, 3 knots,
- in mod5b, 4 knots, and
- in mod5c, 5 knots

```
mod5a_rcs <- lm(y ~ rcs(x3, 3), data = pollution)
mod5b_rcs <- lm(y ~ rcs(x3, 4), data = pollution)
mod5c_rcs <- lm(y ~ rcs(x3, 5), data = pollution)
```

Here, for instance, is the summary of the 5-knot model:

```
summary(mod5c_rcs)
```

```
Call:
lm(formula = y ~ rcs(x3, 5), data = pollution)

Residuals:
     Min       1Q   Median       3Q      Max
-141.522  -32.009    1.674   31.971  147.878

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      468.113    396.319   1.181    0.243
rcs(x3, 5)x3       6.447      5.749   1.121    0.267
rcs(x3, 5)x3'    -25.633     46.810  -0.548    0.586
rcs(x3, 5)x3''   323.137    293.065   1.103    0.275
rcs(x3, 5)x3''' -612.578    396.270  -1.546    0.128

Residual standard error: 54.35 on 55 degrees of freedom
Multiple R-squared:  0.2883,    Adjusted R-squared:  0.2366
F-statistic: 5.571 on 4 and 55 DF,  p-value: 0.0007734
```

We'll begin by storing the fitted values from these three models and other summaries, for plotting.

```
mod5a_aug <- augment(mod5a_rcs, pollution)

mod5b_aug <- augment(mod5b_rcs, pollution)
```

```
mod5c_aug <- augment(mod5c_rcs, pollution)


p2 <- ggplot(pollution, aes(x = x3, y = y)) +
    geom_point() +
    geom_smooth(method = "loess", col = "purple",
                formula = y ~ x, se = F) +
    labs(title = "Loess Smooth")

p3 <- ggplot(mod5a_aug, aes(x = x3, y = y)) +
    geom_point() +
    geom_line(aes(x = x3, y = .fitted),
              col = "blue", size = 1.25) +
    labs(title = "RCS, 3 knots")

p4 <- ggplot(mod5b_aug, aes(x = x3, y = y)) +
    geom_point() +
    geom_line(aes(x = x3, y = .fitted),
              col = "black", size = 1.25) +
    labs(title = "RCS, 4 knots")

p5 <- ggplot(mod5c_aug, aes(x = x3, y = y)) +
    geom_point() +
    geom_line(aes(x = x3, y = .fitted),
              col = "red", size = 1.25) +
    labs(title = "RCS, 5 knots")

(p2 + p3) / (p4 + p5)
```
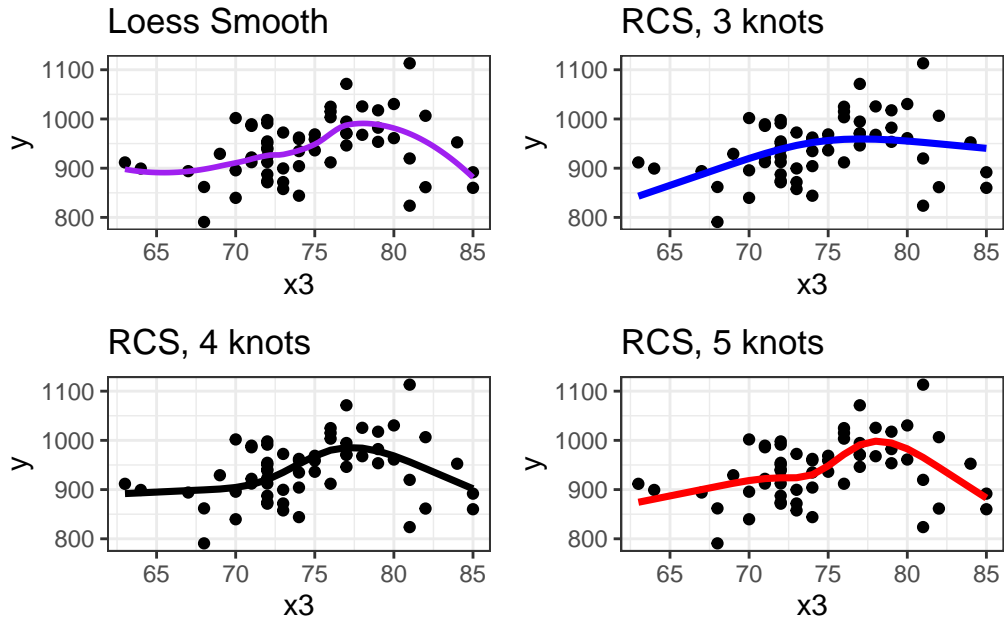
Does it seem like the fit improves markedly (perhaps approaching the loess smooth result) as we increase the number of knots?

```
anova(mod5a_rcs, mod5b_rcs, mod5c_rcs)
```

```
Analysis of Variance Table

Model 1: y ~ rcs(x3, 3)
Model 2: y ~ rcs(x3, 4)
Model 3: y ~ rcs(x3, 5)
  Res.Df    RSS Df Sum of Sq      F   Pr(>F)
1     57 194935
2     56 171448  1   23486.9 7.9503 0.006672 **
3     55 162481  1    8967.2 3.0354 0.087057 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Based on an ANOVA comparison, the fourth knot adds significant predictive value ($p = 0.0067$), but the fifth knot is borderline ($p = 0.0871$). From the `glance` function in the `broom` package, we can also look at some key summaries.

```
glance(mod5a_rcs)
```

```
# A tibble: 1 x 12
  r.squared adj.r.squared sigma statistic p.value    df logLik   AIC   BIC
      <dbl>         <dbl> <dbl>     <dbl>   <dbl> <dbl>  <dbl> <dbl> <dbl>
1     0.146         0.116  58.5      4.88  0.0111     2  -328.  663.  672.
# i 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

```
glance(mod5b_rcs)
```

```
# A tibble: 1 x 12
  r.squared adj.r.squared sigma statistic p.value    df logLik   AIC   BIC
      <dbl>         <dbl> <dbl>     <dbl>   <dbl> <dbl>  <dbl> <dbl> <dbl>
1     0.249         0.209  55.3      6.19 0.00104     3  -324.  658.  668.
# i 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

```
glance(mod5c_rcs)
```

```
# A tibble: 1 x 12
  r.squared adj.r.squared sigma statistic  p.value    df logLik   AIC   BIC
      <dbl>         <dbl> <dbl>     <dbl>    <dbl> <dbl>  <dbl> <dbl> <dbl>
1     0.288         0.237  54.4      5.57 0.000773     4  -322.  657.  669.
# i 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

Model	Knots	R^2	Adj. R^2	AIC	BIC
5a	3	0.146	0.116	663.4	671.8
5b	4	0.249	0.209	657.7	**668.2**
5c	5	0.288	**0.237**	**656.5**	669.1

Within our sample, the five-knot RCS outperforms the 3- and 4-knot versions on adjusted R^2 and AIC (barely) and does a little worse than the 4-knot RCS on BIC.

Of course, we could also use the cross-validation methods we've developed for other linear regressions to assess predictive capacity of these models. I'll skip that for now.

To see the values of `x3` where the splines place their knots, we can use the `attributes` function.

```
attributes(rcs(pollution$x3, 5))
```

$dim
[1] 60 4

$dimnames
$dimnames[[1]]
NULL

$dimnames[[2]]
[1] "pollution"   "pollution'"   "pollution''" "pollution'''"

$class
[1] "rms"

$name
[1] "pollution"

$label
[1] "pollution"

$assume
[1] "rcspline"

$assume.code
[1] 4

$parms
[1] 68 72 74 77 82

$nonlinear
[1] FALSE TRUE   TRUE   TRUE

$colnames
[1] "pollution"   "pollution'"   "pollution''" "pollution'''"

The knots in this particular 5-knot spline are placed by the computer at 68, 72, 74, 77 and 82, it seems.

There are two kinds of Multivariate Regression Models

1. [Prediction] Those that are built so that we can make accurate predictions.
2. [Explanatory] Those that are built to help understand underlying phenomena.

While those two notions overlap considerably, they do imply different things about how we strategize about model-building and model assessment. Harrell's primary concern is effective use of the available data for **prediction** - this implies some things that will be different from what we've seen in the past.

Harrell refers to multivariable regression modeling strategy as the process of **spending degrees of freedom**. The main job in strategizing about multivariate modeling is to

1. Decide the number of degrees of freedom that can be spent
2. Decide where to spend them
3. Spend them, wisely.

What this means is essentially linked to making decisions about predictor complexity, both in terms of how many predictors will be included in the regression model, and about how we'll include those predictors.

33.8 "Spending" Degrees of Freedom

- "Spending" df includes
  - fitting parameter estimates in models, or
  - examining figures built using the outcome variable Y that tell you how to model the predictors.

If you use a scatterplot of Y vs. X or the residuals of the Y-X regression model vs. X to decide whether a linear model is appropriate, then how many degrees of freedom have you actually spent?

Grambsch and O'Brien conclude that if you wish to preserve the key statistical properties of the various estimation and fitting procedures used in building a model, you can't retrieve these degrees of freedom once they have been spent.

33.8.1 Overfitting and Limits on Predictor Counts

Suppose you have a total sample size of n observations, then you really shouldn't be thinking about estimating more than $n/15$ regression coefficients, at the most.

- If k is the number of parameters in a full model containing all candidate predictors for a stepwise analysis, then k should be no greater than $n/15$.
- k should include all variables screened for association with the response, including interaction terms.

- Sometimes I hold myself to a tougher standard, or $n/50$ predictors, at maximum.

So if you have 97 observations in your data, then you can probably just barely justify the use of a stepwise analysis using the main effects alone of 5 candidate variables (with one additional DF for the intercept term) using the $n/15$ limit.

Harrell (2001) also mentions that if you have a **narrowly distributed** predictor, without a lot of variation to work with, then an even larger sample size n should be required. See Vittinghoff et al. (2012), Section 10.3 for more details.

33.8.2 The Importance of Collinearity

Collinearity denotes correlation between predictors high enough to degrade the precision of the regression coefficient estimates substantially for some or all of the correlated predictors

- Vittinghoff et al. (2012), section 10.4.1
- Can one predictor in a model be predicted well using the other predictors in the model?
  - Strong correlations (for instance, $r \geq 0.8$) are especially troublesome.
- Effects of collinearity
  - decreases precision, in the sense of increasing the standard errors of the parameter estimates
  - decreases power
  - increases the difficulty of interpreting individual predictor effects
  - overall F test is significant, but individual t tests may not be

Suppose we want to assess whether variable X_j is collinear with the other predictors in a model. We run a regression predicting X_j using the other predictors, and obtain the R^2. The VIF is defined as 1 / (1 - this R^2), and we usually interpret VIFs above 5 as indicating a serious multicollinearity problem (i.e. R^2 values for this predictor of 0.8 and above would thus concern us.)

```
vif(lm(y ~ x1 + x2 + x3 + x4 + x5 + x6,
            data = pollution))


      x1       x2       x3       x4       x5       x6
2.238862 2.058731 2.153044 4.174448 3.447399 1.792996
```

Again, you'll occasionally see the inverse of VIF reported, and this is called *tolerance*.

- tolerance = 1 / VIF

33.8.3 Collinearity in an Explanatory Model

- When we are attempting to **identify multiple independent predictors** (the explanatory model approach), then we will need to choose between collinear variables

    - options suggested by Vittinghoff et al. (2012), p. 422, include choosing on the basis of plausibility as a causal factor,
    - choosing the variable that has higher data quality (is measured more accurately or has fewer missing values.)
    - Often, we choose to include a variable that is statistically significant as a predictor, and drop others, should we be so lucky.

- Larger effects, especially if they are associated with predictors that have minimal correlation with the other predictors under study, cause less trouble in terms of potential violation of the $n/15$ rule for what constitutes a reasonable number of predictors.

33.8.4 Collinearity in a Prediction Model

- If we are primarily building a **prediction model** for which inference on the individual predictors is not of interest, then it is totally reasonable to use both predictors in the model, if doing so reduces prediction error.

    - Collinearity doesn't affect predictions in our model development sample.
    - Collinearity doesn't affect predictions on new data so long as the new data have similar relationships between predictors.
    - If our key predictor is correlated strongly with a confounder, then if the predictor remains significant after adjustment for the confounder, then this suggests a meaningful independent effect.
        * If the effects of the predictor are clearly confounded by the adjustment variable, we again have a clear result.
        * If neither is statistically significant after adjustment, the data may be inadequate.
    - If the collinearity is between adjustment variables, but doesn't involve the key predictor, then inclusion of the collinear variables is unlikely to cause substantial problems.

33.9 Spending Degrees of Freedom

We need a flexible approach to assessing non-linearity and fitting models with non-linear predictors. This will lead us to a measure of what Harrell (2001) calls **potential predictive punch** which hides the true form of the regression from the analyst so as to preserve statistical properties, but that lets us make sensible decisions about whether a predictor should be

included in a model, and the number of parameters (degrees of freedom, essentially) we are willing to devote to it.

What if we want to consider where best to spend our degrees of freedom on non-linear predictor terms, like interactions, polynomial functions or curved splines to represent our input data? The approach we'll find useful in the largest variety of settings is a combination of

1. a rank correlation assessment of potential predictive punch (using a Spearman ρ^2 plot, available in the `Hmisc` package), followed by
2. the application of restricted cubic splines to fit and assess models.

Let's try such a plot for our fifteen predictors:

```
sp2 <- Hmisc::spearman2(y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 +
                        x8 + x9 + x10 + x11 + x12 + x13 +
                        x14 + x15, data = pollution)

plot(sp2)
```
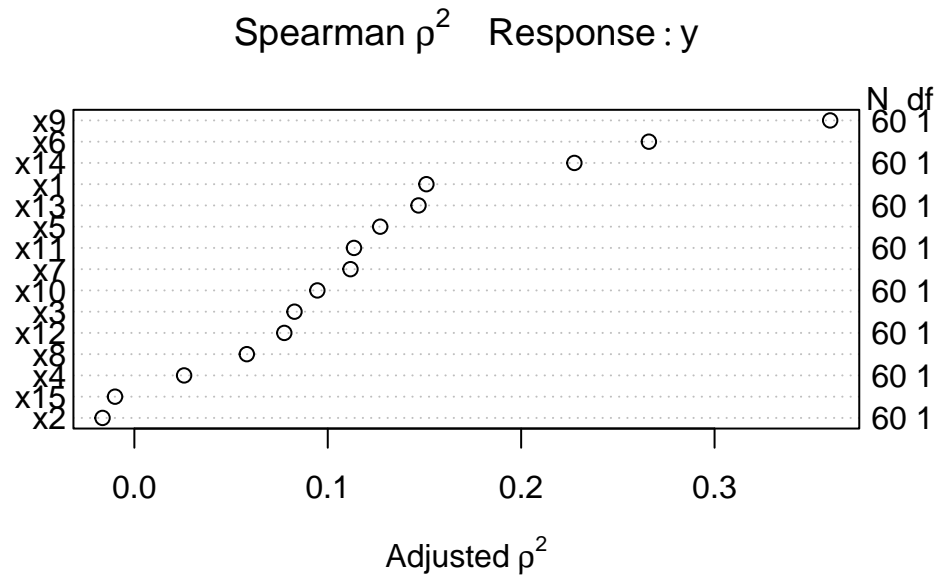

The variable with the largest adjusted squared Spearman ρ statistic in this setting is `x9`, followed by `x6` and `x14`. With only 60 observations, we might well want to restrict ourselves to a very small model. What the Spearman plot suggests is that we focus any non-linear terms

on x9 first, and then perhaps x6 and x14 as they have some potential predictive power. It may or may not work out that the non-linear terms are productive.

33.9.1 Fitting a Big Model

So, one possible model built in reaction this plot might be to fit:

- a restricted cubic spline with 5 knots on x9,
- a restricted cubic spline with 3 knots on x6,
- a quadratic polynomial on x14, and
- a linear fit to x1 and x13

That's way more degrees of freedom (4 for x9, 2 for x6, 2 for x14 and 1 each for x1 and x13 makes a total of 10 without the intercept term) than we can really justify with a sample of 60 observations. But let's see what happens.

```
mod_big <- lm(y ~ rcs(x9, 5) + rcs(x6, 3) + poly(x14, 2) +
                  x1 + x13, data = pollution)

anova(mod_big)
```

```
Analysis of Variance Table

Response: y
             Df Sum Sq Mean Sq F value    Pr(>F)
rcs(x9, 5)    4 100164 25040.9 17.8482 4.229e-09 ***
rcs(x6, 3)    2  38306 19152.8 13.6513 1.939e-05 ***
poly(x14, 2)  2  15595  7797.7  5.5579  0.006677 **
x1            1   4787  4787.3  3.4122  0.070759 .
x13           1    712   711.9  0.5074  0.479635
Residuals    49  68747  1403.0
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This `anova` suggests that we have at least some predictive value in each spline (x9 and x6) and some additional value in x14, although it's not as clear that the linear terms (x1 and x13) did much good.

33.9.2 Limitations of `lm` fits

We can certainly assess this big, complex model using `lm` in comparison to other models:

- with in-sample summary statistics like adjusted R^2, AIC and BIC,
- we can assess its assumptions with residual plots, and
- we can also compare out-of-sample predictive quality through cross-validation,

But to really delve into the details of how well this complex model works, and to help plot what is actually being fit, we'll probably want to fit the model using an alternative method for fitting linear models, called `ols`, from the `rms` package developed by Frank Harrell and colleagues. That will be the focus of our next chapter.

34 Using `ols` to fit linear models

34.1 Setup: Packages Used Here

```r
knitr::opts_chunk$set(comment = NA)

library(broom)
library(Hmisc)
library(rms)
library(patchwork)
library(tidyverse)

theme_set(theme_bw())
```

34.2 Where We Are

At the end of Chapter 33, we had fit a model to the `pollution` data that predicted our outcome y = Age-Adjusted Mortality Rate, using:

- a restricted cubic spline with 5 knots on `x9`
- a restricted cubic spline with 3 knots on `x6`
- a polynomial in 2 degrees on `x14`
- linear terms for `x1` and `x13`

but this model was hard to evaluate in some ways. Now, instead of using `lm` to fit this model, we'll use a new function called `ols` from the `rms` package developed by Frank Harrell and colleagues, in part to support ideas developed in Harrell (2001) for clinical prediction models.

34.3 Fitting a model with `ols`

We will use the `datadist` approach when fitting a linear model with `ols` from the `rms` package, so as to store additional important elements of the model fit.

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

d <- datadist(pollution)
options(datadist = "d")
```

Next, we'll fit the model using `ols` and place its results in `newmod`.

```
newmod <- ols(y ~ rcs(x9, 5) + rcs(x6, 3) + pol(x14, 2) +
                 x1 + x13,
              data = pollution, x = TRUE, y = TRUE)
newmod
```

```
Linear Regression Model

ols(formula = y ~ rcs(x9, 5) + rcs(x6, 3) + pol(x14, 2) + x1 +
    x13, data = pollution, x = TRUE, y = TRUE)


                Model Likelihood     Discrimination
                     Ratio Test            Indexes
Obs        60    LR chi2     72.02   R2          0.699
sigma37.4566     d.f.           10   R2 adj      0.637
d.f.       49    Pr(> chi2) 0.0000   g          58.961


Residuals

    Min       1Q  Median       3Q      Max
-86.189  -18.554  -1.799   18.645  104.307


            Coef       S.E.      t     Pr(>|t|)
Intercept  796.2658  162.3269   4.91  <0.0001
x9          -2.6328    6.3504  -0.41  0.6803
x9'        121.4651  124.4827   0.98  0.3340
x9''      -219.8025  227.6775  -0.97  0.3391
x9'''      151.5700  171.3867   0.88  0.3808
x6           7.6817   15.5230   0.49  0.6229
x6'        -29.4388   18.0531  -1.63  0.1094
x14          0.5652    0.2547   2.22  0.0311
x14^2       -0.0010    0.0010  -0.96  0.3407
x1           1.0717    0.7317   1.46  0.1494
x13         -0.1028    0.1443  -0.71  0.4796
```

Some of the advantages and disadvantages of fitting linear regression models with `ols` or `lm` will reveal themselves over time. For now, one advantage for `ols` is that the entire variance-covariance matrix is saved. Most of the time, there will be some value to considering both `ols` and `lm` approaches.

Most of this output should be familiar, but a few pieces are different.

34.3.1 The Model Likelihood Ratio Test

The **Model Likelihood Ratio Test** compares `newmod` to the null model with only an intercept term. It is a goodness-of-fit test that we'll use in several types of model settings this semester.

- In many settings, the logarithm of the likelihood ratio, multiplied by -2, yields a value which can be compared to a χ^2 distribution. So here, the value 72.02 is -2(log likelihood), and is compared to a χ^2 distribution with 10 degrees of freedom. We reject the null hypothesis that `newmod` is no better than the null model, and conclude instead that at least one of these predictors adds statistically significant value.

  - For `ols`, interpret the model likelihood ratio test like the global (ANOVA) F test in `lm`.
  - The likelihood function is the probability of observing our data under the specified model.
  - We can compare two nested models by evaluating the difference in their likelihood ratios and degrees of freedom, then comparing the result to a χ^2 distribution.

34.3.2 The g statistic

The **g statistic** is new and is referred to as the g-index. it's based on Gini's mean difference and is purported to be a robust and highly efficient measure of variation.

- Here, g = 58.9, which implies that if you randomly select two of the 60 areas included in the model, the average difference in predicted `y` (Age-Adjusted Mortality Rate) using this model will be 58.9.

  - Technically, g is Gini's mean difference of the predicted values.

34.4 ANOVA for an `ols` model

One advantage of the `ols` approach is that when you apply an `anova` to it, it separates out the linear and non-linear components of restricted cubic splines and polynomial terms (as well as product terms, if your model includes them.)

```
anova(newmod)
```

```
            Analysis of Variance          Response: y

Factor           d.f. Partial SS  MS          F      P
x9                 4    35219.7647  8804.9412  6.28  0.0004
 Nonlinear         3     1339.3081   446.4360  0.32  0.8121
x6                 2     9367.6008  4683.8004  3.34  0.0437
 Nonlinear         1     3730.7388  3730.7388  2.66  0.1094
x14                2    18679.6957  9339.8478  6.66  0.0028
 Nonlinear         1     1298.7625  1298.7625  0.93  0.3407
x1                 1     3009.1829  3009.1829  2.14  0.1494
x13                1      711.9108   711.9108  0.51  0.4796
TOTAL NONLINEAR    5     6656.1824  1331.2365  0.95  0.4582
REGRESSION        10   159563.8285 15956.3829 11.37 <.0001
ERROR             49    68746.8004  1402.9959
```

Unlike the `anova` approach in `lm`, in `ols` ANOVA, *partial* F tests are presented - each predictor is assessed as "last predictor in" much like the usual t tests in `lm`. In essence, the partial sums of squares and F tests here describe the marginal impact of removing each covariate from `newmod`.

We conclude that the non-linear parts of `x9` and `x6` and `x14` combined don't seem to add much value, but that overall, `x9`, `x6` and `x14` seem to be valuable. So it must be the linear parts of those variables within our model that are doing the lion's share of the work.

34.5 Effect Estimates

A particularly useful thing to get out of the `ols` approach that is not as easily available in `lm` (without recoding or standardizing our predictors) is a summary of the effects of each predictor in an interesting scale.

```
summary(newmod)
```

```
          Effects                   Response : y

Factor Low   High  Diff. Effect    S.E.    Lower 0.95 Upper 0.95
x9      4.95 15.65 10.70  40.4060 14.0790  12.1120    68.6990
x6     10.40 11.50  1.10 -18.2930  8.1499 -34.6710    -1.9153
x14    11.00 69.00 58.00  28.3480 10.6480   6.9503    49.7460
```

```
x1      32.75 43.25 10.50  11.2520  7.6833  -4.1878   26.6930
x13      4.00 23.75 19.75  -2.0303  2.8502  -7.7579    3.6973
```

This "effects summary" shows the effect on `y` of moving from the 25th to the 75th percentile of each variable (along with a standard error and 95% confidence interval) while holding the other variable at the level specified at the bottom of the output.

The most useful way to look at this sort of analysis is often a plot.

```
plot(summary(newmod))
```
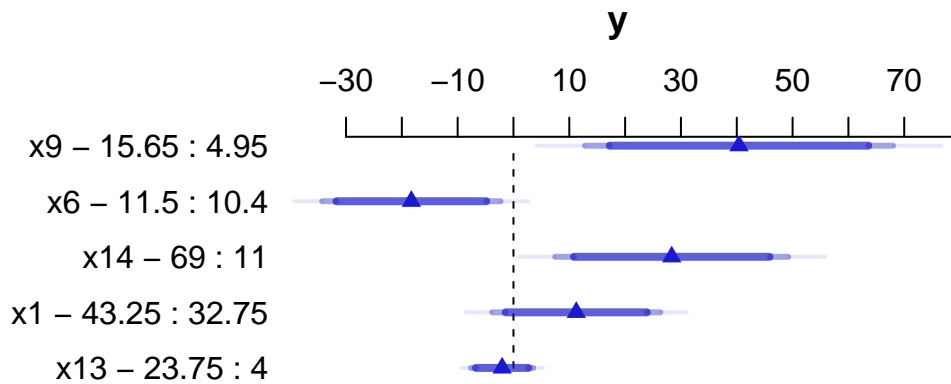

For `x9` note from the `summary` above that the 25th percentile is 4.95 and the 75th is 15.65. Our conclusion is that the estimated effect of moving `x9` from 4.95 to 15.65 is an increase of 40.4 on `y`, with a 95% CI of (12.1, 68.7).

For a categorical variable, the low level is shown first and then the high level.

The plot shows the point estimate (arrow head) and then the 90% (narrowest bar), 95% (middle bar) and 99% (widest bar in lightest color) confidence intervals for each predictor's effect.

- It's easier to distinguish this in the `x9` plot than the one for `x13`.
- Remember that what is being compared is the first value to the second value's impact on the outcome, with other predictors held constant.

34.5.1 Simultaneous Confidence Intervals

These confidence intervals make no effort to deal with the multiple comparisons problem, but just fit individual 95% (or whatever level you choose) confidence intervals for each predictor. The natural alternative is to make an adjustment for multiple comparisons in fitting the confidence intervals, so that the set of (in this case, five - one for each predictor) confidence intervals for effect sizes has a family-wise 95% confidence level. You'll note that the effect estimates and standard errors are unchanged from those shown above, but the confidence limits are a bit wider.

```
summary(newmod, conf.type=c('simultaneous'))
```

```
            Effects                  Response : y

Factor Low   High  Diff. Effect    S.E.     Lower 0.95 Upper 0.95
x9      4.95 15.65 10.70  40.4060 14.0790    3.1139    77.6980
x6     10.40 11.50  1.10 -18.2930  8.1499  -39.8800     3.2933
x14    11.00 69.00 58.00  28.3480 10.6480    0.1452    56.5510
x1     32.75 43.25 10.50  11.2520  7.6833   -9.0982    31.6030
x13     4.00 23.75 19.75  -2.0303  2.8502   -9.5794     5.5189
```

Remember that if you're looking for the usual `lm` summary for an `ols` object, use `summary.lm`, and that the `display` function from `arm` does not recognize `ols` objects.

34.6 The `Predict` function for an `ols` model

The `Predict` function is very flexible, and can be used to produce individual or simultaneous confidence limits.

```
Predict(newmod, x9 = 12, x6 = 12, x14 = 40,
        x1 = 40, x13 = 20) # individual limits
```

```
  x9 x6 x14 x1 x13      yhat     lower    upper
1 12 12  40 40   20 923.0982 893.0984 953.098

Response variable (y): y

Limits are 0.95 confidence limits
```

```
Predict(newmod, x9 = 5:15) # individual limits
```

```
     x9    x6 x14 x1 x13      yhat     lower     upper
1     5 11.05  30 38    9 913.7392 889.4802 937.9983
2     6 11.05  30 38    9 916.3490 892.0082 940.6897
3     7 11.05  30 38    9 921.3093 898.9657 943.6529
4     8 11.05  30 38    9 927.6464 907.0355 948.2574
5     9 11.05  30 38    9 934.3853 913.3761 955.3946
6    10 11.05  30 38    9 940.5510 917.8371 963.2648
7    11 11.05  30 38    9 945.2225 921.9971 968.4479
8    12 11.05  30 38    9 948.2885 926.4576 970.1194
9    13 11.05  30 38    9 950.2608 930.3003 970.2213
10   14 11.05  30 38    9 951.6671 932.2370 971.0971
11   15 11.05  30 38    9 953.0342 932.1662 973.9021

Response variable (y): y

Adjust to: x6=11.05 x14=30 x1=38 x13=9

Limits are 0.95 confidence limits
```

```
Predict(newmod, x9 = 5:15, conf.type = 'simult')
```

```
     x9    x6 x14 x1 x13      yhat     lower     upper
1     5 11.05  30 38    9 913.7392 882.4900 944.9885
2     6 11.05  30 38    9 916.3490 884.9945 947.7035
3     7 11.05  30 38    9 921.3093 892.5275 950.0912
4     8 11.05  30 38    9 927.6464 901.0965 954.1964
5     9 11.05  30 38    9 934.3853 907.3223 961.4484
6    10 11.05  30 38    9 940.5510 911.2922 969.8098
7    11 11.05  30 38    9 945.2225 915.3047 975.1403
8    12 11.05  30 38    9 948.2885 920.1671 976.4100
9    13 11.05  30 38    9 950.2608 924.5487 975.9728
10   14 11.05  30 38    9 951.6671 926.6383 976.6959
11   15 11.05  30 38    9 953.0342 926.1531 979.9152

Response variable (y): y

Adjust to: x6=11.05 x14=30 x1=38 x13=9

Limits are 0.95 confidence limits
```

The plot below shows the individual effects in `newmod` in five subpanels, using the default approach of displaying the same range of values as are seen in the data. Note that each panel shows point and interval estimates of the effects, and spot the straight lines in `x1` and `x13`, the single bends in `x14` and `x6` and the wiggles in `x9`, corresponding to the amount of non-linearity specified in the model.

```
ggplot(Predict(newmod))
```
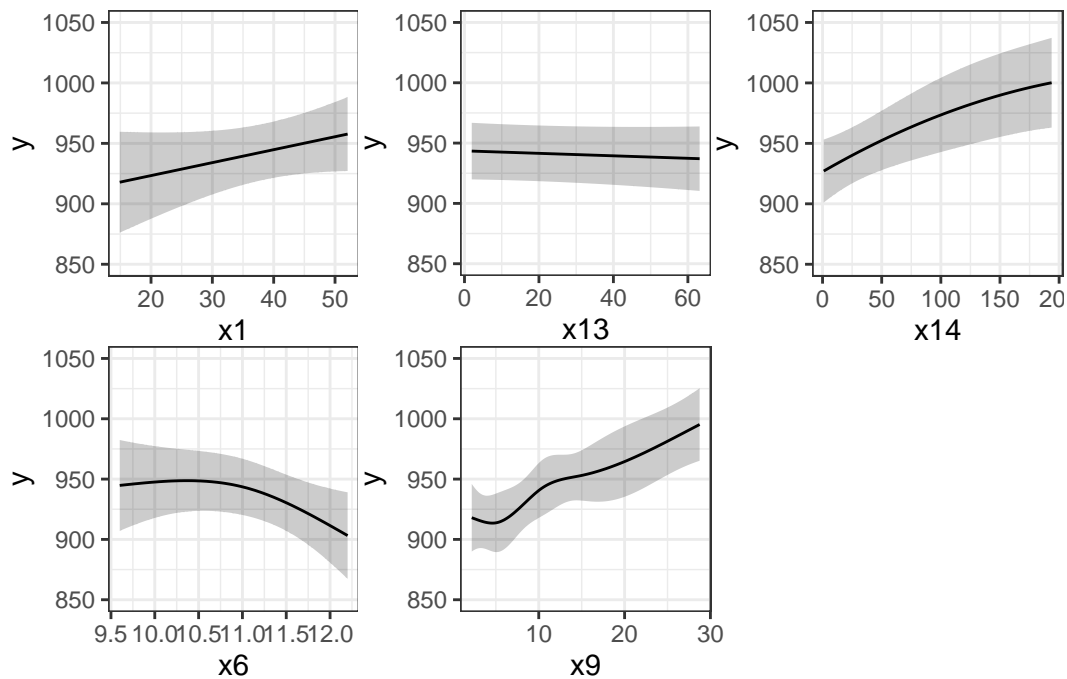

34.7 Checking Influence via `dfbeta`

For an `ols` object, we have several tools for looking at residuals. The most interesting to me is `which.influence` which is reliant on the notion of `dfbeta`.

- DFBETA is estimated for each observation in the data, and each coefficient in the model.
- The DFBETA is the difference in the estimated coefficient caused by deleting the observation, scaled by the coefficient's standard error estimated with the observation deleted.
- The `which.influence` command applied to an `ols` model produces a list of all of the predictors estimated by the model, including the intercept.

    - For each predictor, the command lists all observations (by row number) that, if removed from the model, would cause the estimated coefficient (the "beta") for that predictor to change by at least some particular cutoff.

– The default is that the DFBETA for that predictor is 0.2 or more.

```
which.influence(newmod)
```

```
$Intercept
[1] "2"  "11" "28" "32" "37" "49" "59"


$x9
[1] "2"  "3"  "6"  "9"  "31" "35" "49" "57" "58"


$x6
[1] "2"  "11" "15" "28" "32" "37" "50" "56" "59"


$x14
[1] "2"  "6"  "7"  "12" "13" "16" "32" "37"


$x1
[1] "7"  "18" "32" "37" "49" "57"


$x13
[1] "29" "32" "37"
```

The implication here, for instance, is that if we drop row 3 from our data frame, and refit the model, this will have a meaningful impact on the estimate of x9 but not on the other coefficients. But if we drop, say, row 37, we will affect the estimates of the intercept, x6, x14, x1, and x13.

34.7.1 Using the `residuals` command for `dfbetas`

To see the **dfbeta** values, standardized according to the approach I used above, you can use the following code (I'll use `head` to just show the first few rows of results) to get a matrix of the results.

```
head(residuals(newmod, type = "dfbetas"))
```

```
    Intercept           x9          x9'         x9''        x9'''           x6
1   0.03071160 -0.023775487 -0.004055111  0.01205425 -0.03260003 -0.02392315
2  -0.38276573 -0.048404993 -0.142293606  0.17009666 -0.22350621  0.44737372
3   0.17226780 -0.426153536  0.350913139 -0.32949129  0.25777913 -0.10263448
4   0.06175110 -0.006460916  0.024828272 -0.03009337  0.04154812 -0.06254145
```

```
5  0.16875200  0.039839994 -0.058178534  0.06449504 -0.07772208 -0.18058630
6  0.03322073  0.112699877 -0.203543632  0.23987378 -0.35201736 -0.04075617
          x6'          x14        x14^2           x1          x13
1  0.01175375 -0.06494414  0.060929683 -0.011042644  0.03425156
2 -0.48562818  0.19372285 -0.212186731 -0.107830147 -0.01503250
3  0.05005284 -0.02049877  0.014059330  0.010793169  0.04924166
4  0.05498432  0.01135031 -0.001877983 -0.005490454 -0.01254111
5  0.16151742  0.02723710  0.065483158  0.003326357 -0.05570035
6  0.02900006 -0.21508009  0.171627718  0.019241676  0.05775536
```

34.7.2 Using the `residuals` command for other summaries

The `residuals` command will also let you get ordinary residuals, leverage values and `dffits` values, which are the normalized differences in predicted values when observations are omitted. See `?residuals.ols` for more details.

```
temp <- tibble(area = 1:60)
temp$residual <- residuals(newmod, type = "ordinary")
temp$leverage <- residuals(newmod, type = "hat")
temp$dffits <- residuals(newmod, type = "dffits")

ggplot(temp, aes(x = area, y = dffits)) +
    geom_point() +
    geom_line()
```
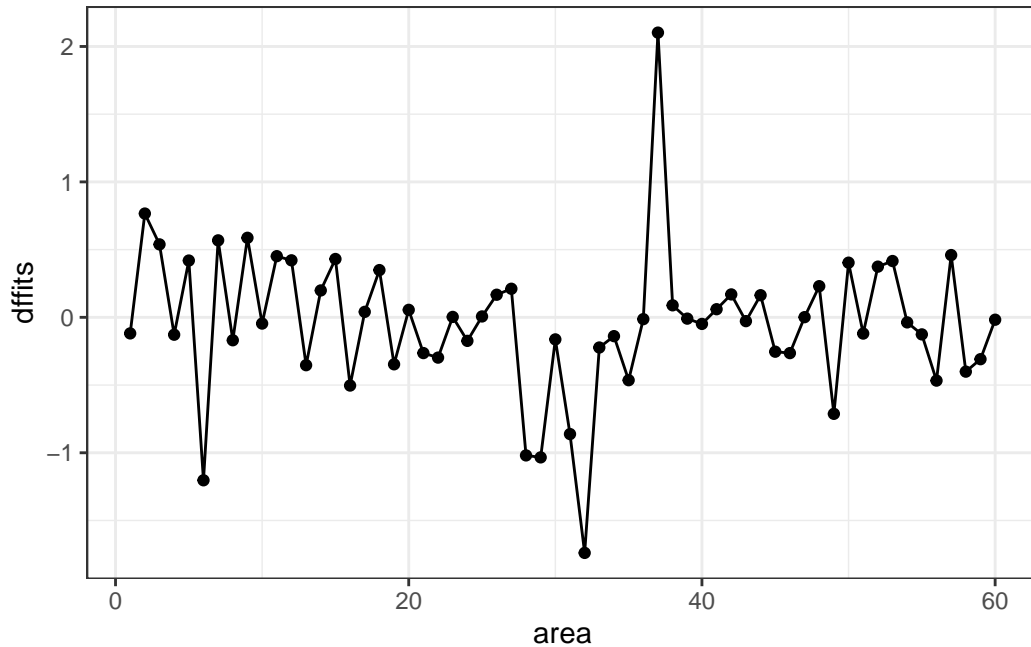
It appears that point 37 has the largest (positive) `dffits` value. Recall that point 37 seemed influential on several predictors and the intercept term. Point 32 has the smallest (or largest negative) `dffits`, and also appears to have been influential on several predictors and the intercept.

```
which.max(temp$dffits)
```

37
37

```
which.min(temp$dffits)
```

32
32

34.8 Model Validation and Correcting for Optimism

We have learned about splitting our regression models into **training** samples and **test** samples, performing variable selection work on the training sample to identify two or three candidate

models (perhaps via a stepwise approach), and then comparing the predictions made by those models in a test sample.

The `validate` function allows us to perform cross-validation of our models for some summary statistics (and then correct those statistics for optimism in describing likely predictive accuracy) in an easy way.

`validate` develops:

- Resampling validation with or without backward elimination of variables
- Estimates of the *optimism* in measures of predictive accuracy
- Estimates of the intercept and slope of a calibration model

$$(\text{observed y}) = \text{Intercept} + \text{Slope (predicted y)}$$

with the following code...

```
set.seed(432002); validate(newmod, method = "boot", B = 40)
```

```
          index.orig training      test  optimism index.corrected  n
R-square      0.6989    0.7426    0.5749    0.1676          0.5312 40
MSE        1145.7800  963.9565 1617.4042 -653.4478       1799.2278 40
g            58.9614   59.7891   54.6444    5.1447         53.8168 40
Intercept     0.0000    0.0000   96.6990  -96.6990         96.6990 40
Slope         1.0000    1.0000    0.8961    0.1039          0.8961 40
```

So, for `R-square` we see that our original estimate was 0.6989

- Our estimated `R-square` across `n` = 40 training samples was 0.7426, but in the resulting tests, the average `R-square` was only 0.5749
- This suggests an optimism of 0.7426 - 0.5749 = 0.1676 (after rounding).
- We then apply that optimism to obtain a new estimate of R^2 corrected for overfitting, at 0.5312, which is probably a better estimate of what our results might look like in new data that were similar to (but not the same as) the data we used in building `newmod` than our initial estimate of 0.6989

We also obtain optimism-corrected estimates of the mean squared error (square of the residual standard deviation), the g index, and the intercept and slope of the calibration model. The "corrected" slope is a shrinkage factor that takes overfitting into account.

34.9 Building a Nomogram for Our Model

Another nice feature of an `ols` model object is that we can picture the model with a **nomogram** easily. Here is model `newmod`.

```
plot(nomogram(newmod))
```
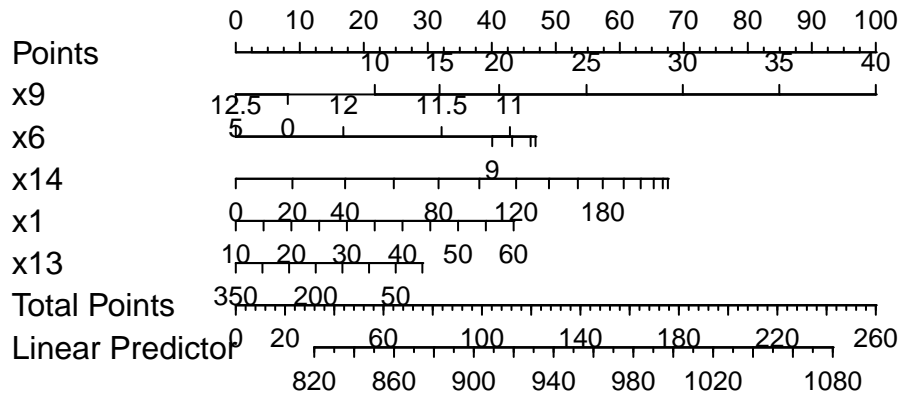

For this model, we can use this plot to predict `y` as follows:

1. find our values of `x9` on the appropriate line
2. draw a vertical line up to the points line to count the points associated with our subject
3. repeat the process to obtain the points associated with `x6`, `x14`, `x1`, and `x13`. Sum the points.
4. draw a vertical line down from that number in the Total Points line to estimate `y` (the Linear Predictor) = Age-Adjusted Mortality Rate.

The impact of the non-linearity is seen in the `x6` results, for example, which turn around from 9-10 to 11-12. We also see non-linearity's effects in the scales of the non-linear terms in terms of points awarded.

An area with a combination of predictor values leading to a total of 100 points, for instance, would lead to a prediction of a Mortality Rate near 905. An area with a total of 140 points would have a predicted Mortality Rate of 955, roughly.

35 BMI and Employment Study

This chapter was developed originally in 2018 and 2019, and has only been lightly revised since. In part, I have done this to let you see some other options. One appealing feature is that this chapter (eventually) uses the `mice` package to do some multiple imputation, and work with missing data, instead of `naniar`. While we'll have other approaches for this in 432, at least you'll see one possibility.

35.1 Setup: Packages Used Here

```
knitr::opts_chunk$set(comment = NA)

library(GGally)
library(mice)
library(tidyverse)

theme_set(theme_bw())
```

We'll also use a function from the `Hmisc` package.

35.2 The Data

A 2016 study published in *BMJ Open* looked at the differential relationship between employment status and body-mass index among middle-aged and elderly adults living in South Korea[1]. Data from this study were available online thanks to the Dryad data package[2]. The original data came from a nationally representative sample of 7228 participants in the Korean Longitudinal Study of Aging. I sampled these data, and did some data "rectangling" (wrangling) to build the `emp_bmi.csv` file on our web site.

---

[1] See Noh J, Kim J, Park J, Oh I, Kwon YD (2016) Age and gender differential relationship between employment status and body mass index among middle-aged and elderly adults: a cross-sectional study. BMJ Open 6(11): e012117. http://dx.doi.org/10.1136/bmjopen-2016-012117

[2] Noh J, Kim J, Park J, Oh I, Kwon YD (2016) Data from: Age and gender differential relationship between employment status and body mass index among middle aged and elderly adults: a cross-sectional study. Dryad Digital Repository. http://dx.doi.org/10.5061/dryad.ng8mn

The available data in `emp_bmi` describe 999 subjects, and included are 8 variables:

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

Variable	Description	NA?
pid	subject identification number (categorical)	0
bmi	our outcome, quantitative, body-mass index	0
age	subject's age (between 51 and 95)	1
gender	subject's gender (male or female)	0
employed	employment status indicator (1/0)	1
married	marital status indicator (1/0)	1
alcohol	3-level factor	2
education	4-level factor	5

```r
Hmisc::describe(emp_bmi)
```

```
emp_bmi

 8  Variables       999  Observations
--------------------------------------------------------------------------------
pid
       n  missing distinct      Info      Mean       Gmd       .05       .10
     999        0      999         1     31774     20178      3506      6961
     .25      .50      .75       .90       .95
   16671    31761    46902     54635     58433


lowest :    22    41    52    82   112, highest: 61471 61481 61532 61641 61691
--------------------------------------------------------------------------------
bmi
       n  missing distinct      Info      Mean       Gmd       .05       .10
     999        0      373         1     23.24     2.808     19.21     20.13
     .25      .50      .75       .90       .95
   21.54    23.24    24.78     26.56     27.34


lowest : 15.6  15.63 16.22 16.44 16.53, highest: 31.11 31.24 32.05 33.5  34.67
--------------------------------------------------------------------------------
age
       n  missing distinct      Info      Mean       Gmd       .05       .10
     998        1       43     0.999     66.29     11.56        52        53
     .25      .50      .75       .90       .95
```

```
       57          66          74          81          83

lowest : 51 52 53 54 55, highest: 89 90 92 93 95
--------------------------------------------------------------------------------
gender
      n  missing distinct
    999        0        2

Value       female    male
Frequency      570     429
Proportion   0.571   0.429
--------------------------------------------------------------------------------
employed
      n  missing distinct     Info      Sum     Mean      Gmd
    998        1        2    0.723      404   0.4048   0.4824


--------------------------------------------------------------------------------
married
      n  missing distinct     Info      Sum     Mean      Gmd
    998        1        2    0.548      758   0.7595   0.3657


--------------------------------------------------------------------------------
alcohol
      n  missing distinct
    997        2        3

Value                  alcohol dependent                      heavy drinker
Frequency                             45                                306
Proportion                         0.045                              0.307

Value       normal drinker or non-drinker
Frequency                            646
Proportion                         0.648
--------------------------------------------------------------------------------
education
      n  missing distinct
    994        5        4

Value     1 elem school grad or lower        2 middle school grad
Frequency                          421                         180
Proportion                       0.424                       0.181

Value               3 high school grad    4 college grad or higher
```

```
Frequency                          292                          101
Proportion                       0.294                        0.102
--------------------------------------------------------------------------
```

35.2.1 Specifying Outcome and Predictors for our Model

In the original study, a key goal was to understand the relationship between employment and body-mass index. Our goal in this example will be to create a model to predict `bmi` focusing on employment status (so our key predictor is `employed`) while accounting for the additional predictors `age`, `gender`, `married`, `alcohol` and `education`. A natural thing to do would be to consider interactions of these predictor variables (for example, does the relationship between `bmi` and `employed` change when comparing men to women?) but we'll postpone that discussion until 432.

35.2.2 Dealing with Missing Predictor Values

```
md.pattern(emp_bmi)
```
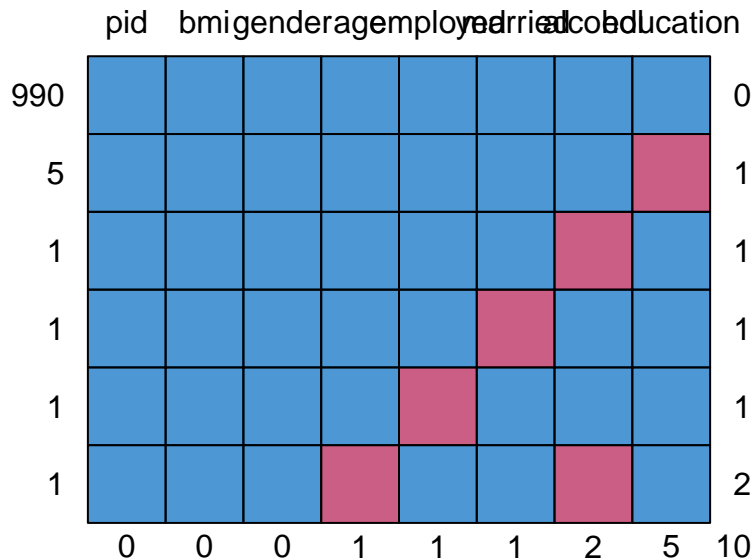


```
    pid bmi gender age employed married alcohol education
990   1   1      1   1        1       1       1         1  0
```

```
5       1   1     1   1     1      1      1        0   1
1       1   1     1   1     1      1      0        1   1
1       1   1     1   1     1      0      1        1   1
1       1   1     1   1     0      1      1        1   1
1       1   1     1   0     1      1      0        1   2
        0   0     0   1     1      1      2        5  10
```

We will eventually build a model to predict `bmi` using all of the other variables besides `pid`. So we'll eventually have to account for the 9 people with missing values (one of whom has two missing values, as we see above.) What I'm going to do in this example is to first build a complete-case analysis on the 990 subjects without missing values and then, later, do multiple imputation to account for the 9 subjects with missing values (and their 10 actual missing values) sensibly.

I'll put the "complete cases" data set of 990 subjects in `emp_bmi_noNA`.

```
emp_bmi_noNA <- emp_bmi |> na.omit()
emp_bmi_noNA
```

```
# A tibble: 990 x 8
     pid   bmi   age gender employed married alcohol                 education
   <dbl> <dbl> <dbl> <chr>     <dbl>   <dbl> <chr>                    <chr>
 1    22  20.8    58 male          1       1 heavy drinker            4 colleg~
 2    41  21.4    76 female        0       1 normal drinker or non-dr~ 1 elem s~
 3    52  20.9    66 female        1       1 heavy drinker            1 elem s~
 4    82  23.7    67 male          1       1 alcohol dependent        2 middle~
 5   112  25.5    63 female        0       1 heavy drinker            1 elem s~
 6   181  20.5    51 female        1       1 heavy drinker            3 high s~
 7   182  25.3    51 male          1       1 alcohol dependent        3 high s~
 8   411  20.8    66 female        1       1 normal drinker or non-dr~ 2 middle~
 9   491  20.8    64 female        0       1 normal drinker or non-dr~ 3 high s~
10   531  24.3    84 female        0       0 normal drinker or non-dr~ 1 elem s~
# i 980 more rows
```

```
colSums(is.na(emp_bmi_noNA))
```

```
      pid       bmi       age    gender  employed   married   alcohol education
        0         0         0         0         0         0         0         0
```

704

35.3 The "Kitchen Sink" Model

A "kitchen sink" model includes all available predictors.

```
ebmodel.1 <- lm(bmi ~ age + gender + employed + married +
                   alcohol + education, data = emp_bmi_noNA)
summary(ebmodel.1)
```

```
Call:
lm(formula = bmi ~ age + gender + employed + married + alcohol +
    education, data = emp_bmi_noNA)

Residuals:
    Min      1Q  Median      3Q     Max
-7.7033 -1.6057 -0.0494  1.4992 11.7303

Coefficients:
                                          Estimate Std. Error t value Pr(>|t|)
(Intercept)                               26.15391    0.88356  29.601  < 2e-16 ***
age                                       -0.04260    0.01065  -3.998 6.86e-05 ***
gendermale                                 0.29811    0.20271   1.471   0.1417
employed                                  -0.45761    0.19153  -2.389   0.0171 *
married                                    0.09438    0.21280   0.444   0.6575
alcoholheavy drinker                       0.25317    0.40727   0.622   0.5343
alcoholnormal drinker or non-drinker       0.14121    0.40766   0.346   0.7291
education2 middle school grad             -0.28862    0.24020  -1.202   0.2298
education3 high school grad               -0.50123    0.22192  -2.259   0.0241 *
education4 college grad or higher         -0.79862    0.31068  -2.571   0.0103 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.513 on 980 degrees of freedom
Multiple R-squared:  0.02251,   Adjusted R-squared:  0.01353
F-statistic: 2.508 on 9 and 980 DF,  p-value: 0.007716
```

35.4 Using Categorical Variables (Factors) as Predictors

We have six predictors here, and five of them are categorical. Note that R recognizes each kind of variable in this case and models them appropriately. Let's look at the coefficients of

our model.

35.4.1 `gender`: A binary variable represented by letters

The `gender` variable contains the two categories: male and female, and R recognizes this as a factor. When building a regression model with such a variable, R assigns the first of the two levels of the factor to the baseline, and includes in the model an indicator variable for the second level. By default, R assigns each factor a level order alphabetically.

So, in this case, we have:

```
is.factor(emp_bmi_noNA$gender)
```

```
[1] FALSE
```

```
levels(emp_bmi_noNA$gender)
```

```
NULL
```

As you see in the model, the `gender` information is captured by the indicator variable `gendermale`, which is 1 when `gender = male` and 0 otherwise.

So, when our model includes:

```
Coefficients:      Estimate Std. Error t value Pr(>|t|)
gendermale          0.29811    0.20271   1.471   0.1417
```

this means that a male subject is predicted to have an outcome that is 0.29811 points higher than a female subject, if they have the same values of all of the other predictors.

Note that if we wanted to switch the levels so that "male" came first (and so that R would use "male" as the baseline category and "female" as the 1 value in an indicator), we could do so with the `forcats` package and the `fct_relevel` command. Building a model with this version of `gender` will simply reverse the sign of our indicator variable, but not change any of the other output.

```
emp_bmi_noNA$gender.2 <- fct_relevel(emp_bmi_noNA$gender, "male", "female")
revised.model <- lm(bmi ~ age + gender.2 + employed + married +
                  alcohol + education, data = emp_bmi_noNA)
summary(revised.model)
```

```
Call:
lm(formula = bmi ~ age + gender.2 + employed + married + alcohol +
    education, data = emp_bmi_noNA)

Residuals:
    Min      1Q  Median      3Q     Max
-7.7033 -1.6057 -0.0494  1.4992 11.7303

Coefficients:
                                       Estimate Std. Error t value Pr(>|t|)
(Intercept)                            26.45202    0.94648  27.948  < 2e-16 ***
age                                    -0.04260    0.01065  -3.998 6.86e-05 ***
gender.2female                         -0.29811    0.20271  -1.471   0.1417
employed                               -0.45761    0.19153  -2.389   0.0171 *
married                                 0.09438    0.21280   0.444   0.6575
alcoholheavy drinker                    0.25317    0.40727   0.622   0.5343
alcoholnormal drinker or non-drinker    0.14121    0.40766   0.346   0.7291
education2 middle school grad          -0.28862    0.24020  -1.202   0.2298
education3 high school grad            -0.50123    0.22192  -2.259   0.0241 *
education4 college grad or higher      -0.79862    0.31068  -2.571   0.0103 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.513 on 980 degrees of freedom
Multiple R-squared:  0.02251,   Adjusted R-squared:  0.01353
F-statistic: 2.508 on 9 and 980 DF,  p-value: 0.007716
```

Note that the two categories here need to be both *mutually exclusive* (a subject cannot be in more than one category) and *collectively exhaustive* (all subjects must fit into this set of categories) in order to work properly as a regression predictor.

35.4.2 `employed`: A binary variable represented a 1/0 indicator

The `employed` and `married` variables are each described using an indicator variable, which is 1 if the condition of interest holds and 0 if it does not. R doesn't recognize this as a factor, but rather as a quantitative variable. However, this is no problem for modeling, where we just need to remember that if `employed` = 1, the subject is employed, and if `employed` = 0, the subject is not employed, to interpret the results. The same approach is used for `married`.

```
Coefficients:    Estimate Std. Error t value Pr(>|t|)
```

```
employed          -0.45761    0.19153  -2.389   0.0171 *
married            0.09438    0.21280   0.444   0.6575
```

So, in our model, if subject A is employed, they are expected to have an outcome that is 0.46 points lower (-0.46 points higher) than subject B who is not employed but otherwise identical to subject A.

Similarly, if subject X is married, and subject Y is unmarried, but they otherwise have the same values of all predictors, then our model will predict a `bmi` for X that is 0.094 points higher than for Y.

35.4.3 `alcohol`: A three-category variable coded by names

Our `alcohol` information divides subjects into three categories, which are:

- normal drinker or non-drinker
- heavy drinker
- alcohol dependent

R builds a model using $k - 1$ predictors to describe a variable with k levels. As mentioned previously, R selects a baseline category when confronted with a factor variable, and it always selects the first level as the baseline. The levels are sorted alphabetically, unless we tell R to sort them some other way. So, we have

```
Coefficients:                        Estimate Std. Error t value Pr(>|t|)
alcoholheavy drinker                  0.25317    0.40727   0.622   0.5343
alcoholnormal drinker or non-drinker  0.14121    0.40766   0.346   0.7291
```

How do we interpret this?

- Suppose subject A is alcohol dependent, B is a heavy drinker and C is a normal drinker or non-drinker, but subjects A-C have the same values of all other predictors.
- Our model predicts that B would have a BMI that is 0.25 points higher than A.
- Our model predicts that C would have a BMI that is 0.14 points higher than A.

A good way to think about this...

	Subject	Status	alcoholheavy drinker	alcoholnormal drinker or non-drinker
	A	alcohol dependent	0	0
	B	heavy drinker	1	0

	Subject	Status	alcoholheavy drinker	alcoholnormal drinker or non-drinker
	C	normal drinker or non-drinker	0	1

and so, with two variables, we cover each of these three possible `alcohol` levels.

When we have an ordered variable like this one, we usually want the baseline category to be at either end of the scale (either the highest or the lowest, but not something in the middle.) Another good idea in many settings is to use as the baseline category the most common category. Here, the baseline R chose was "alcohol dependent" which is the least common category, so I might want to use the `fct_relevel` function again to force R to choose, say, normal drinker/non-drinker as the baseline category.

```
emp_bmi_noNA$alcohol.2 <- fct_relevel(emp_bmi_noNA$alcohol,
          "normal drinker or non-drinker", "heavy drinker")
revised.model.2 <- lm(bmi ~ age + gender + employed + married +
              alcohol.2 + education, data = emp_bmi_noNA)
summary(revised.model.2)
```

```
Call:
lm(formula = bmi ~ age + gender + employed + married + alcohol.2 +
    education, data = emp_bmi_noNA)

Residuals:
    Min      1Q  Median      3Q     Max
-7.7033 -1.6057 -0.0494  1.4992 11.7303

Coefficients:
                                Estimate Std. Error t value Pr(>|t|)
(Intercept)                     26.29512    0.82860  31.734  < 2e-16 ***
age                             -0.04260    0.01065  -3.998 6.86e-05 ***
gendermale                       0.29811    0.20271   1.471   0.1417
employed                        -0.45761    0.19153  -2.389   0.0171 *
married                          0.09438    0.21280   0.444   0.6575
alcohol.2heavy drinker           0.11196    0.19647   0.570   0.5689
alcohol.2alcohol dependent      -0.14121    0.40766  -0.346   0.7291
education2 middle school grad   -0.28862    0.24020  -1.202   0.2298
education3 high school grad     -0.50123    0.22192  -2.259   0.0241 *
education4 college grad or higher -0.79862  0.31068  -2.571   0.0103 *
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.513 on 980 degrees of freedom
Multiple R-squared:  0.02251,    Adjusted R-squared:  0.01353
F-statistic: 2.508 on 9 and 980 DF,  p-value: 0.007716
```

How do we interpret this revised model?

- Again, subject A is alcohol dependent, B is a heavy drinker and C is a normal drinker or non-drinker, but subjects A-C have the same values of all other predictors.
- Our model predicts that B would have a BMI that is 0.11 points higher than C.
- Our model predicts that A would have a BMI that is 0.14 points lower than C.

So, those are the same conclusions, just rephrased.

35.4.4 t tests and multi-categorical variables

The usual "last predictor in" t test works perfectly for binary factors, but suppose we have a factor like `alcohol` which is represented by two different indicator variables. If we want to know whether the `alcohol` information, as a group, adds statistically significant value to the model that includes all of the other predictors, then our best strategy is to compare two models - one with the alcohol information, and one without.

```
model.with.a <- lm(bmi ~ age + gender + alcohol + employed + married + education,
                   data = emp_bmi_noNA)
model.no.a <- lm(bmi ~ age + gender + employed + married + education,
                 data = emp_bmi_noNA)
anova(model.with.a, model.no.a)
```

```
Analysis of Variance Table

Model 1: bmi ~ age + gender + alcohol + employed + married + education
Model 2: bmi ~ age + gender + employed + married + education
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1    980 6189.9
2    982 6193.6 -2   -3.6288 0.2873 0.7504
```

The p value for both of the indicator variables associated with `alcohol` combined is 0.75, according to an ANOVA F test with 2 degrees of freedom.

Note that we can get the same information from an ANOVA table of the larger model if we add the `alcohol` predictor to the model last.

```
anova(lm(bmi ~ age + gender + employed + married + education + alcohol,
         data = emp_bmi_noNA))
```

```
Analysis of Variance Table

Response: bmi
           Df Sum Sq Mean Sq F value   Pr(>F)
age         1   55.5  55.510  8.7884 0.003105 **
gender      1    0.4   0.369  0.0585 0.809000
employed    1   30.7  30.733  4.8657 0.027626 *
married     1    0.4   0.374  0.0592 0.807746
education   3   51.9  17.311  2.7408 0.042202 *
alcohol     2    3.6   1.814  0.2873 0.750382
Residuals 980 6189.9   6.316
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Again, we see p for the two `alcohol` indicators is 0.75.

35.4.5 `education`: A four-category variable coded by names

The `education` variable's codes are a little better designed. By preceding the text with a number for each code, we force R to attend to the level order we want to see.

```
Coefficients:                           Estimate Std. Error t value Pr(>|t|)
education2 middle school grad           -0.28862    0.24020  -1.202   0.2298
education3 high school grad             -0.50123    0.22192  -2.259   0.0241 *
education4 college grad or higher       -0.79862    0.31068  -2.571   0.0103 *
```

Since we have four education levels, we need those three indicator variables.

- `education2 middle school grad` is 1 if the subject is a middle school graduate, and 0 if they have some other status
- `education3 high school grad` is 1 if the subject is a high school graduate, and 0 if they have some other status
- `education4 college grad or higher` is 1 if the subject is a college graduate or has more education, and 0 if they have some other status.

- So the subjects with only elementary school or lower education are represented by zeros in all three indicators.

Suppose we have four subjects now, with the same values of all other predictors, but different levels of education.

Subject	Education	Estimated BMI
A	elementary school or less	A
B	middle school grad	A - 0.289
C	high school grad	A - 0.501
D	college grad	A - 0.799

Note that the four categories are *mutually exclusive* (a subject cannot be in more than one category) and *collectively exhaustive* (all subjects must fit into this set of categories.) As we have seen, this is a requirement of categorical variables in a regression analysis.

Let's run the ANOVA test for the `education` information captured in those three indicator variables...

```
anova(lm(bmi ~ age + gender + employed + married + alcohol + education,
         data = emp_bmi_noNA))
```

```
Analysis of Variance Table

Response: bmi
           Df Sum Sq Mean Sq F value   Pr(>F)
age         1   55.5  55.510  8.7884 0.003105 **
gender      1    0.4   0.369  0.0585 0.809000
employed    1   30.7  30.733  4.8657 0.027626 *
married     1    0.4   0.374  0.0592 0.807746
alcohol     2    3.5   1.750  0.2771 0.758055
education   3   52.1  17.354  2.7476 0.041820 *
Residuals 980 6189.9   6.316
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

So, as a group, the three indicator variables add statistically significant predictive value at the 5% significance level, since the F test for those three variables has $p = 0.042$

35.4.6 Interpreting the Kitchen Sink Model

So, again, here's our model.

```
ebmodel.1 <- lm(bmi ~ age + gender + employed + married +
                    alcohol + education, data = emp_bmi_noNA)
ebmodel.1
```

```
Call:
lm(formula = bmi ~ age + gender + employed + married + alcohol +
    education, data = emp_bmi_noNA)

Coefficients:
                         (Intercept)                                    age
                            26.15391                               -0.04260
                          gendermale                               employed
                             0.29811                               -0.45761
                             married                  alcoholheavy drinker
                             0.09438                                0.25317
alcoholnormal drinker or non-drinker        education2 middle school grad
                             0.14121                               -0.28862
           education3 high school grad    education4 college grad or higher
                            -0.50123                               -0.79862
```

If we wanted to predict a BMI level for a new subject like the ones used in the development of this model, that prediction would be:

- 26.15
- minus 0.426 times the subject's `age`
- plus 0.298 if the subject's `gender` was `male`
- minus 0.458 if the subject's employment status was `employed`
- plus 0.253 if the subject's `alcohol` classification was `heavy drinker`
- plus 0.141 if the subject's `alcohol` classification was `normal drinker or non-drinker`
- minus 0.289 if the subject's `education` classification was `2 middle school grad`
- minus 0.501 if the subject's `education` classification was `3 high school grad`
- minus 0.799 if the subject's `education` classification was `4 college grad or higher`

35.5 Scatterplot Matrix with Categorical Predictors

Let's look at a scatterplot matrix of a few key predictors, with my favorite approach (at least for quantitative predictors)...

```
ggpairs(emp_bmi_noNA |> select(age, gender, employed, education, bmi))
```
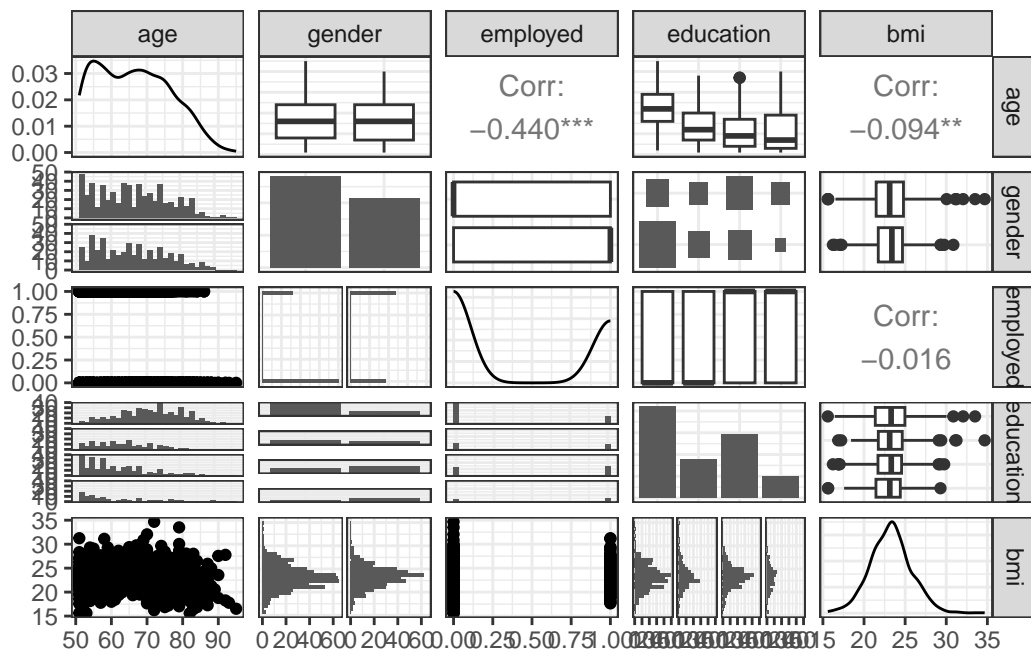

35.6 Residual Plots when we have Categorical Predictors

Here are the main residual plots from the kitchen sink model `ebmodel.1` defined previously.

```
par(mfrow=c(2,2))
plot(ebmodel.1)
```
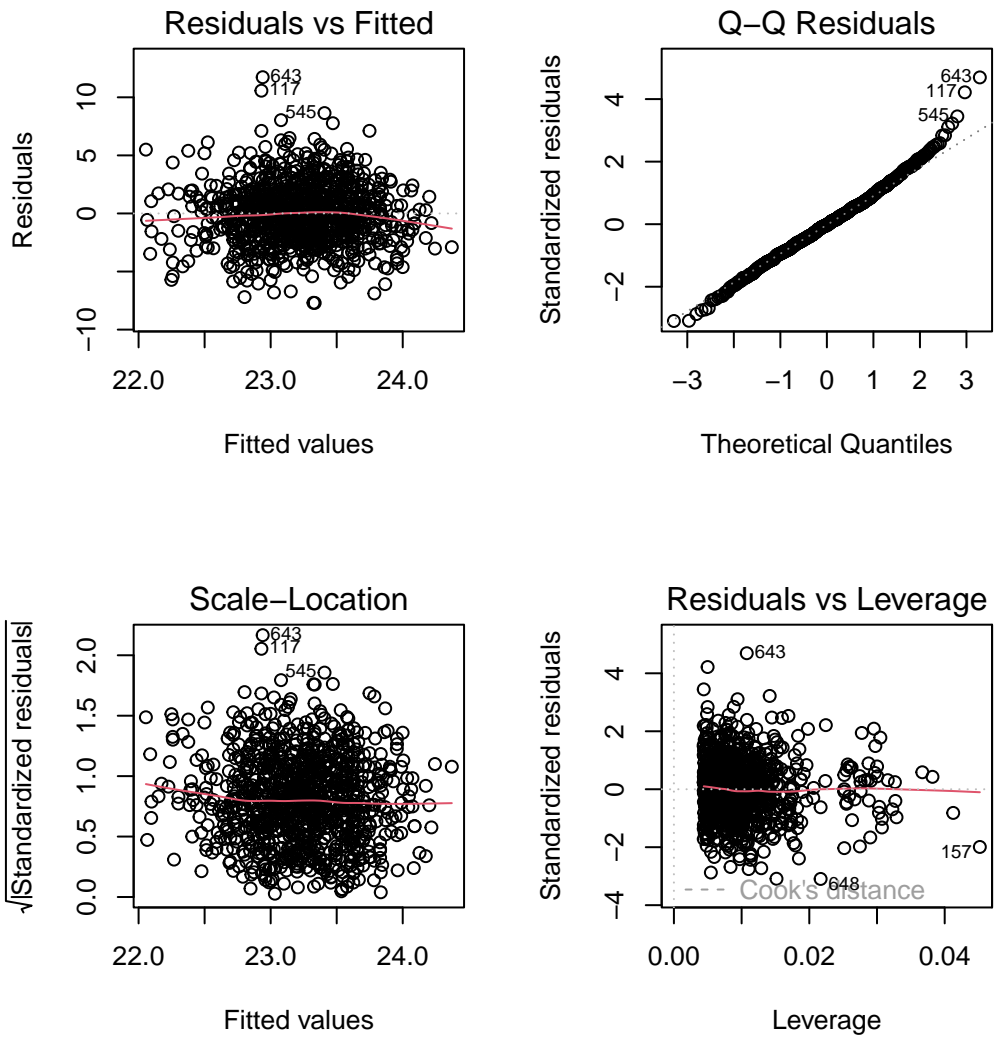
```
par(mfrow=c(1,1))
```

Sometimes, in small samples, the categorical variables will make the regression residuals line up in somewhat strange patterns. But in this case, there's no real problem. The use of categorical variables also has some impact on leverage, as it's hard for a subject to be a serious outlier in terms of a predictor if that predictor only has a few possible levels.

35.7 Stepwise Regression and Categorical Predictors

When R does backwards elimination for stepwise model selection, it makes decisions about each categorical variable as in/out across all of the indicator variables simultaneously, as you'd hope.

```
step(ebmodel.1)
```

```
Start:  AIC=1834.64
bmi ~ age + gender + employed + married + alcohol + education

            Df Sum of Sq    RSS    AIC
- alcohol    2     3.629 6193.6 1831.2
- married    1     1.243 6191.2 1832.8
<none>                    6189.9 1834.6
- gender     1    13.660 6203.6 1834.8
- education  3    52.063 6242.0 1836.9
- employed   1    36.057 6226.0 1838.4
- age        1   100.976 6290.9 1848.7

Step:  AIC=1831.22
bmi ~ age + gender + employed + married + education

            Df Sum of Sq    RSS    AIC
- married    1     1.026 6194.6 1829.4
<none>                    6193.6 1831.2
- gender     1    19.071 6212.6 1832.3
- education  3    51.934 6245.5 1833.5
- employed   1    34.736 6228.3 1834.8
- age        1   108.599 6302.2 1846.4

Step:  AIC=1829.39
bmi ~ age + gender + employed + education

            Df Sum of Sq    RSS    AIC
<none>                    6194.6 1829.4
- gender     1    22.169 6216.7 1830.9
- education  3    51.282 6245.9 1831.5
- employed   1    34.390 6229.0 1832.9
- age        1   125.431 6320.0 1847.2
```

```
Call:
lm(formula = bmi ~ age + gender + employed + education, data = emp_bmi_noNA)

Coefficients:
                    (Intercept)                              age
                       26.50206                         -0.04459
                     gendermale                         employed
                        0.34057                         -0.44571
      education2 middle school grad     education3 high school grad
                       -0.28787                         -0.49772
education4 college grad or higher
                       -0.79037
```

Note that the stepwise approach first drops two degrees of freedom (two indicator variables) for `alcohol` and then drops the one degree of freedom for `married` before it settles on a model with `age`, `gender`, `education` and `employed`.

35.8 Pooling Results after Multiple Imputation

As mentioned earlier, having built a model using complete cases, we should probably investigate the impact of multiple imputation on the missing observations. We'll fit 100 imputations using the `emp_bmi` data and then fit a pooled regression model across those imputations.

```
emp_bmi_mi <- mice(emp_bmi, m = 100, maxit = 5,
                   printFlag = FALSE, seed = 4312021)
```

Now, we'll fit the pooled kitchen sink regression model to these imputed data sets and pool them.

```
model.empbmi.mi <-
    with(emp_bmi_mi, lm(bmi ~ age + gender + employed +
                            married + alcohol + education))
summary(pool(model.empbmi.mi))
```

```
                        term     estimate  std.error   statistic
1                (Intercept) 26.15304370 0.88053709  29.7012402
2                        age -0.04258223 0.01063989  -4.0021319
3                 gendermale  0.29794032 0.20237455   1.4722223
4                   employed -0.45695506 0.19112904  -2.3908196
5                    married  0.09430912 0.21247583   0.4438581
```

717

```
6                      alcoholheavy drinker  0.25298858 0.40270904   0.6282168
7  alcoholnormal drinker or non-drinker  0.14108076 0.40341263   0.3497182
8          education2 middle school grad -0.28861113 0.23994418 -1.2028261
9            education3 high school grad -0.50119826 0.22138848 -2.2638859
10    education4 college grad or higher -0.79852096 0.30947717 -2.5802258
        df        p.value
1  979.908 0.000000e+00
2  979.908 6.750699e-05
3  979.908 1.412819e-01
4  979.908 1.699872e-02
5  979.908 6.572432e-01
6  979.908 5.300083e-01
7  979.908 7.266253e-01
8  979.908 2.293342e-01
9  979.908 2.379907e-02
10 979.908 1.001814e-02
```

OK. That's it for now.

A Getting Data Into R

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>
```

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_new` with the following command:

```
nnyfs_new <- read_rds("data/nnyfs.Rds")
```

Here are the results...

```
nnyfs_new
```

```
# A tibble: 1,518 x 45
      SEQN sex     age_child race_eth     educ_child language sampling_wt income_pov
     <dbl> <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>, ...
```

Using `read_csv` to read in a comma-separated version of a data file

We have provided the `nnyfs.csv` 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.csv` file to a new tibble in R called `nnyfs_new2` with the following command:

```
nnyfs_new2 <- read_csv("data/nnyfs.csv")
```

```
Rows: 1518 Columns: 45
-- Column specification -----------------------------------------------------
Delimiter: ","
chr (18): sex, race_eth, language, educ_adult, respondent, salt_used, diet_y...
dbl (27): SEQN, age_child, educ_child, sampling_wt, income_pov, age_adult, e...

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.
```

```
nnyfs_new2
```

```
# A tibble: 1,518 x 45
     SEQN sex    age_child race_eth    educ_child language sampling_wt income_pov
    <dbl> <chr>      <dbl> <chr>           <dbl> <chr>          <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 <chr>, respondent <chr>,
#   salt_used <chr>, energy <dbl>, protein <dbl>, sugar <dbl>, fat <dbl>,
#   diet_yesterday <chr>, water <dbl>, plank_time <dbl>, height <dbl>,
#   weight <dbl>, bmi <dbl>, bmi_cat <chr>, arm_length <dbl>, waist <dbl>,
```

```
#     arm_circ <dbl>, calf_circ <dbl>, calf_skinfold <dbl>,
#     triceps_skinfold <dbl>, subscapular_skinfold <dbl>, active_days <dbl>, ...
```

If you also want to convert the **character** variables to **factors**, as you will often want to do before analyzing the results, you should instead use:

```
nnyfs_new3 <- read_csv("data/nnyfs.csv") %>%
    mutate(across(where(is.character), as_factor))
```

```
Rows: 1518 Columns: 45
-- Column specification ---------------------------------------------------------
Delimiter: ","
chr (18): sex, race_eth, language, educ_adult, respondent, salt_used, diet_y...
dbl (27): SEQN, age_child, educ_child, sampling_wt, income_pov, age_adult, e...

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.
```

```
nnyfs_new3
```

```
# A tibble: 1,518 x 45
     SEQN sex     age_child race_eth     educ_child language sampling_wt income_pov
    <dbl> <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>, ...
```

Note that, for example, `sex` and `race_eth` are now listed as factor (`fctr`) variables. One place where this distinction between `character` and `factor` variables matters is when you summarize the data.

```
summary(nnyfs_new2$race_eth)
```

```
  Length     Class      Mode
    1518 character character
```

```
summary(nnyfs_new3$race_eth)
```

```
  3_Black Non-Hispanic    2_White Non-Hispanic              1_Hispanic
                   338                     610                     450
4_Other Race/Ethnicity
                   120
```

Converting Character Variables into Factors

The command you want to create `newdata` from `olddata` is:

```
newdata <- olddata %>%
    mutate(across(where(is.character), as_factor))
```

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

Consider visiting the software tutorials page under the R and Data heading on our main web site.

B Session Information

It is often helpful to include a summary of your R session, specifically some information about your operating system, the version of R and the packages you made use of in creating your document.

We demonstrate two approaches to including this below, each of which you'll use in submitting Lab and Project assignments over the course of the semester.

B.1 Using sessionInfo()

One approach is to use the `sessionInfo()` function available in base R.

```r
sessionInfo()
```

```
R version 4.3.1 (2023-06-16 ucrt)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 11 x64 (build 22621)

Matrix products: default


locale:
[1] LC_COLLATE=English_United States.utf8
[2] LC_CTYPE=English_United States.utf8
[3] LC_MONETARY=English_United States.utf8
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.utf8

time zone: America/New_York
tzcode source: internal

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base
```

```
loaded via a namespace (and not attached):
 [1] compiler_4.3.1    fastmap_1.1.1     cli_3.6.1        tools_4.3.1
 [5] htmltools_0.5.6.1 rstudioapi_0.15.0 rmarkdown_2.25   knitr_1.44
 [9] jsonlite_1.8.7    xfun_0.40         digest_0.6.33    rlang_1.1.1
[13] evaluate_0.22
```

B.2 Using session_info()

Another approach is to use the `session_info()` function available in the `sessioninfo` package. Professor Love has a slight preference for this approach, but it's not an important distinction to make.

```
sessioninfo::session_info()
```

```
- Session info ---------------------------------------------------------------
 setting  value
 version  R version 4.3.1 (2023-06-16 ucrt)
 os       Windows 11 x64 (build 22621)
 system   x86_64, mingw32
 ui       RTerm
 language (EN)
 collate  English_United States.utf8
 ctype    English_United States.utf8
 tz       America/New_York
 date     2023-10-15
 pandoc   3.1.1 @ C:/Program Files/RStudio/resources/app/bin/quarto/bin/tools/ (via rmarkdown

- Packages -------------------------------------------------------------------
 package     * version date (UTC) lib source
 cli           3.6.1   2023-03-23 [1] CRAN (R 4.3.1)
 digest        0.6.33  2023-07-07 [1] CRAN (R 4.3.1)
 evaluate      0.22    2023-09-29 [1] CRAN (R 4.3.1)
 fastmap       1.1.1   2023-02-24 [1] CRAN (R 4.3.1)
 htmltools     0.5.6.1 2023-10-06 [1] CRAN (R 4.3.1)
 jsonlite      1.8.7   2023-06-29 [1] CRAN (R 4.3.1)
 knitr         1.44    2023-09-11 [1] CRAN (R 4.3.1)
 rlang         1.1.1   2023-04-28 [1] CRAN (R 4.3.1)
 rmarkdown     2.25    2023-09-18 [1] CRAN (R 4.3.1)
 rstudioapi    0.15.0  2023-07-07 [1] CRAN (R 4.3.1)
 sessioninfo   1.2.2   2021-12-06 [1] CRAN (R 4.3.1)
 xfun          0.40    2023-08-09 [1] CRAN (R 4.3.1)
```

```
[1] C:/Users/thoma/AppData/Local/R/win-library/4.3
[2] C:/Program Files/R/R-4.3.1/library
```

C References

Bernard, Gordon R., Arthur P. Wheeler, James A. Russell, Roland Schein, Warren R. Summer, Kenneth P. Steinberg, William J. Fulkerson, et al. 1997. "The Effects of Ibuprofen on the Physiology and Survival of Patients with Sepsis." *New England Journal of Medicine* 336: 912–18. http://www.nejm.org/doi/full/10.1056/NEJM199703273361303#t=article.

Bock, David E., Paul F. Velleman, and Richard D. De Veaux. 2004. *Stats: Modelling the World.* Boston MA: Pearson Addison-Wesley.

Dupont, William D. 2002. *Statistical Modeling for Biomedical Researchers.* New York: Cambridge University Press.

Faraway, Julian J. 2015. *Linear Models with r.* Second. Boca Raton, FL: CRC Press.

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, and Deborah Nolan. 2017. *Teaching Statistics: A Bag of Tricks.* Second Edition. Oxford, UK: Oxford University Press.

Good, Phillip I. 2005. *Introduction to Statistics Through Resampling Methods and r/s-PLUS.* Hoboken, NJ: Wiley.

Hadley Wickham, Mine Çetinyaka-Rundel, and Garrett Grolemund. 2023. *R for Data Science.* Second. O'Reilly. https://r4ds.hadley.nz/.

Harrell, Frank E. 2001. *Regression Modeling Strategies.* New York: Springer.

Ismay, Chester, and Albert Y. Kim. 2022. *ModernDive: Statistical Inference via Data Science.* http://moderndive.com/.

McDonald, Gary C., and Richard C. Schwing. 1973. "Instabilities of Regression Estimates Relating Air Pollution to Mortality." *Technometrics* 15 (3): 463–81.

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.

Norman, Geoffrey R., and David L. Streiner. 2014. *Biostatistics: The Bare Essentials.* Fourth. People's Medical Publishing House.

Pagano, Marcello, and Kimberlee Gauvreau. 2000. *Principles of Biostatistics.* Second. Duxbury Press.

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.

Ramsey, Fred L., and Daniel W. Schafer. 2002. *The Statistical Sleuth: A Course in Methods of Data Analysis.* Second. Pacific Grove, CA: Duxbury.

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/.

Wainer, Howard. 1997. *Visual Revelations: Graphical Tales of Fate and Deception from Napoleon Bonaparte to Ross Perot.* New York: Springer-Verlag.

———. 2005. *Graphic Discovery: A Trout in the Milk and Other Visual Adventures.* Princeton, NJ: Princeton University Press.

———. 2013. *Medical Illuminations: Using Evidence, Visualization and Statistical Thinking to Improve Healthcare.* New York: Oxford University Press.

Yamada, SB, and EG Boulding. 1998. "Claw Morphology, Prey Size Selection and Foraging Efficiency in Generalist and Specialist Shell-Breaking Crabs." *Journal of Experimental Marine Biology and Ecology* 220: 191–211. http://www.science.oregonstate.edu/~yamadas/SylviaCV/BehrensYamada_Boulding1998.pdf.