

Data Science for Biological, Medical and Health Research: Notes for PQHS/CRSP/MPHP 431

Thomas E. Love

2020-09-21

Contents

Working with These Notes	9
The 431 Course online	9
What You'll Find Here	9
Setting Up R	10
Initial Setup of R Packages	11
The <code>Love-boost.R</code> script	11
Additional R Packages installed for this book	12
1 Data Science	13
1.1 Data Science Project Cycle	14
1.2 Data Science and the 431 Course	15
1.3 What The Course Is and Isn't	15
Part A. Exploring Data	19
2 Looking at the Palmer Penguins	19
2.1 Package Loading, then Dealing with Missing Data	19
2.2 Counting Things and Making Tables	20
2.3 Visualizing the Data in a Graph (or a few...)	21
2.4 Six Ways To "Improve" This Graph	23
2.5 A Little Reflection	24
3 NHANES: Initial Exploring	25
3.1 The NHANES data: Collecting a Sample	25
3.2 Age and Height	26
3.3 Subset of Subjects with Known Age and Height	27
3.4 Age-Height and Sex?	29
3.5 Creating A New Subset: Ages 21-79	32
3.6 Distribution of Heights	33
3.7 Height and Sex	36
3.8 Looking at Pulse Rate	41
3.9 General Health Status	49
3.10 Conclusions	57

4 Data Structures and Types of Variables	59
4.1 Data require structure and context	59
4.2 A New NHANES Adult Sample	60
4.3 Quantitative Variables	64
4.4 Qualitative (Categorical) Variables	66
5 Summarizing Quantitative Variables	67
5.1 The <code>summary</code> function for Quantitative data	67
5.2 Measuring the Center of a Distribution	68
5.3 Measuring the Spread of a Distribution	71
5.4 Measuring the Shape of a Distribution	76
5.5 More Detailed Numerical Summaries for Quantitative Variables .	79
6 Summarizing Categorical Variables	85
6.1 The <code>summary</code> function for Categorical data	85
6.2 Tables to describe One Categorical Variable	86
6.3 The Mode of a Categorical Variable	87
6.4 <code>describe</code> in the <code>Hmisc</code> package	88
6.5 Cross-Tabulations	90
6.6 Constructing Tables Well	94
6.7 Gaining Control over Tables in R: the <code>gt</code> package	96
7 NHANES National Youth Fitness Survey (<code>nnyfs</code>)	97
7.1 The Variables included in <code>nnyfs</code>	98
7.2 Looking over A Few Variables	100
7.3 Additional Numeric Summaries	111
7.4 Additional Summaries from <code>favstats</code>	112
7.5 The Histogram	112
7.6 The Dot Plot to display a distribution	117
7.7 The Frequency Polygon	118
7.8 Plotting the Probability Density Function	119
7.9 The Boxplot	120
7.10 A Simple Comparison Boxplot	122
7.11 Using <code>describe</code> in the <code>psych</code> library	125
7.12 Assessing Skew	126
7.13 Assessing Kurtosis (Heavy-Tailedness)	127
7.14 The <code>describe</code> function in the <code>Hmisc</code> package	128
7.15 What Summaries to Report	129
8 Assessing Normality	131
8.1 Empirical Rule Interpretation of the Standard Deviation	131
8.2 Describing Outlying Values with Z Scores	132
8.3 Comparing a Histogram to a Normal Distribution	133
8.4 Does a Normal model work well for the <code>waist</code> circumference? .	135
8.5 The Normal Q-Q Plot	137
8.6 Interpreting the Normal Q-Q Plot	138

8.7 Can a Normal Distribution Fit the <code>nnyfs</code> <code>energy</code> data Well?	146
9 Using Transformations to “Normalize” Distributions	151
9.1 The Ladder of Power Transformations	151
9.2 Using the Ladder	152
9.3 Protein Consumption in the NNYFS data	152
9.4 Can we transform the <code>protein</code> data?	155
9.5 What if we considered all 9 available transformations?	158
9.6 A Simulated Data Set	161
9.7 What if we considered all 9 available transformations?	166
10 Summarizing data within subgroups	169
10.1 Using <code>dplyr</code> and <code>summarise</code> to build a tibble of summary information	169
10.2 Another Example	171
10.3 Boxplots to Relate an Outcome to a Categorical Predictor	173
10.4 Using Multiple Histograms to Make Comparisons	181
10.5 Using Multiple Density Plots to Make Comparisons	182
10.6 A Ridgeline Plot	185
11 Straight Line Models and Correlation	189
11.1 Assessing A Scatterplot	189
11.2 Correlation Coefficients	195
11.3 The Pearson Correlation Coefficient	196
11.4 Studying Correlation through 6 Examples	196
11.5 Estimating Correlation from Scatterplots	203
11.6 The Spearman Rank Correlation	208
12 Studying Crab Claws (<code>crabs</code>)	215
12.1 Association of Size and Force	217
12.2 The <code>loess</code> smooth	219
12.3 Fitting a Linear Regression Model	223
12.4 Is a Linear Model Appropriate?	225
12.5 Making Predictions with a Model	228
13 The Western Collaborative Group Study	233
13.1 The Western Collaborative Group Study (<code>wcgs</code>) data set	233
13.2 Are the SBPs Normally Distributed?	237
13.3 Identifying and Describing SBP outliers	240
13.4 Does Weight Category Relate to SBP?	242
13.5 Re-Leveling a Factor	242
13.6 Are Weight and SBP Linked?	245
13.7 SBP and Weight by Arcus Senilis groups?	246
13.8 Linear Model for SBP-Weight Relationship: subjects without Arcus Senilis	248

13.9 Linear Model for SBP-Weight Relationship: subjects with Arcus Senilis	249
13.10 Including Arcus Status in the model	250
13.11 Predictions from these Linear Models	251
13.12 Scatterplots with Facets Across a Categorical Variable	252
13.13 Scatterplot and Correlation Matrices	252
14 Re-Expression, Tukey's Ladder & Box-Cox Plot	255
14.1 “Linearize” The Association between Quantitative Variables	255
14.2 A New Tool: the Box-Cox Plot	255
14.3 A Simulated Example	256
14.4 Checking on a Transformation or Re-Expression	259
15 Dehydration Recovery in Kids: A Small Study	263
15.1 A Scatterplot Matrix	264
15.2 Are the recovery scores well described by a Normal model?	265
15.3 Simple Regression: Using Dose to predict Recovery	266
15.4 The Scatterplot, with fitted Linear Model	266
15.5 The Fitted Linear Model	267
15.6 The Summary Output	268
15.7 Viewing the complete ANOVA table	273
15.8 Plotting Residuals vs. Fitted Values	274
16 Highlights of What We've Seen So Far	277
16.1 Key Graphical Descriptive Summaries for Quantitative Data	277
16.2 Key Numerical Descriptive Summaries for Quantitative Data	277
16.3 The Empirical Rule - Interpreting a Standard Deviation	278
16.4 Identifying “Outliers” Using Fences and/or Z Scores	278
16.5 Summarizing Bivariate Associations: Scatterplots and Regression Lines	279
16.6 Summarizing Bivariate Associations With Correlations	279
17 Confidence Intervals for a Mean	281
17.1 Love-boost.R is something we'll start using now	281
17.2 Introduction	281
17.3 This Chapter's Goals	282
17.4 Serum Zinc Levels in 462 Teenage Males (serzinc)	282
17.5 Our Goal: A Confidence Interval for the Population Mean	283
17.6 Exploratory Data Analysis for Serum Zinc	283
17.7 Defining a Confidence Interval	285
17.8 Estimating the Population Mean from the Serum Zinc data	286
17.9 Confidence vs. Significance Level	286
17.10 The Standard Error of a Sample Mean	287
17.11 The t distribution and CIs for a Mean	287
17.12 Building the CI in R	289
17.13 Using an intercept-only regression model	289

17.14Interpreting the Result	290
17.15What if we want a 95% or 99% confidence interval instead?	291
17.16Using the <code>broom</code> package with the t test	291
17.17One-sided vs. Two-sided Confidence Intervals	292
17.18Bootstrap Confidence Intervals	294
17.19Resampling is A Big Idea	294
17.20When is a Bootstrap Confidence Interval Reasonable?	294
17.21Bootstrap confidence interval for the mean: Process	295
17.22Using R to estimate a bootstrap CI	295
17.23Comparing Bootstrap and T-Based Confidence Intervals	296
17.24Using the Bootstrap to develop other CIs	297
17.25One-Tailed Bootstrap Confidence Intervals	297
17.26Wilcoxon Signed Rank Procedure for CIs	299
17.27Wilcoxon Signed Rank-based CI in R	300
17.28General Advice	301
18 The Ibuprofen in Sepsis Randomized Clinical Trial	303
18.1 Comparing Two Groups	305
18.2 Key Questions for Comparing with Independent Samples	306
18.3 Exploratory Data Analysis	307
18.4 Estimating the Difference in Population Means	310
18.5 t-based CI for population mean1 - mean2 difference	311
18.6 Wilcoxon-Mann-Whitney “Rank Sum” CI	314
18.7 Bootstrapping: A More Robust Approach	315
18.8 Summary: Specifying A Two-Sample Study Design	316
18.9 Results for the <code>sepsis</code> study	317
18.10Categorizing the Outcome and Comparing Rates	321
18.11Estimating the Difference in Proportions	321

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, and use the hamburger icon (horizontal bars) at the top of the document to open or close the table of contents.
2. At the top of the document, you'll see additional icons which you can click to
 - search the document,
 - change the size, font or color scheme of the page, and
 - download a PDF or EPUB (Kindle-readable) version of the entire document.
3. The document will be updated (unpredictably) throughout the semester.

The 431 Course online

The **main web page** for the 431 course in Fall 2020 is <https://thomaselove.github.io/431/>. Go there for all information related to the course.



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.

Everything you see here is available to you as HTML or PDF. You will also have access to the R Markdown files, which contain the code which generates everything in the document, including all of the R results. We will demonstrate the use of R Markdown (this document is generated with the additional help of an R package called `bookdown`) and RStudio (the “program” we use to interface with the R language) in class.

All data and R code related to these notes are also available to you.

Setting Up R

These Notes make extensive use of

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

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

If you need an even gentler 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 <http://moderndive.com/>, which provides an introduction to statistical and data sciences via R at Ismay and Kim (2019).

These notes were written using R Markdown. R Markdown, like R and R Studio, is free and open source.

R Markdown is described as an *authoring framework* for data science, which lets you

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

This description comes from <http://rmarkdown.rstudio.com/lesson-1.html> which you can visit to get an overview and quick tour of what's possible with R Markdown.

Another excellent resource to learn more about R Markdown tools is the Communicate section (especially the R Markdown chapter) of Grolemund and Wickham (2019).

Initial Setup of R Packages

To start, I'll present a series of commands I run at the beginning of these Notes. These particular commands set up the output so it will look nice as either an HTML or PDF file, and also set up R to use several packages (libraries) of functions that expand its capabilities. A chunk of code like this will occur near the top of any R Markdown work.

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

library(knitr)
library(magrittr)
library(janitor)
library(NHANES)
library(palmerpenguins)
library(patchwork)
library(rms)
library(tidymodels) # note: tidymodels includes the broom package
library(tidyverse) # note: tidyverse includes the dplyr and ggplot2 packages

theme_set(theme_bw())
```

I have deliberately set up this list of loaded packages to be relatively small, and will add some others later in these Notes. You only need to install a package once, but you need to reload it every time you start a new session.

The Love-boost.R script

Starting in October, we'll make use of a few scripts I've gathered for you.

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

Additional R Packages installed for this book

Some packages need to be installed on the user's system, but do not need to be loaded by R in order to run the code presented in this set of notes. These additional packages include the following.

```
boot  
car  
Epi  
GGally  
gt  
psych  
modelsummary  
mosaic  
naniar  
visdat
```

Chapter 1

Data Science

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.

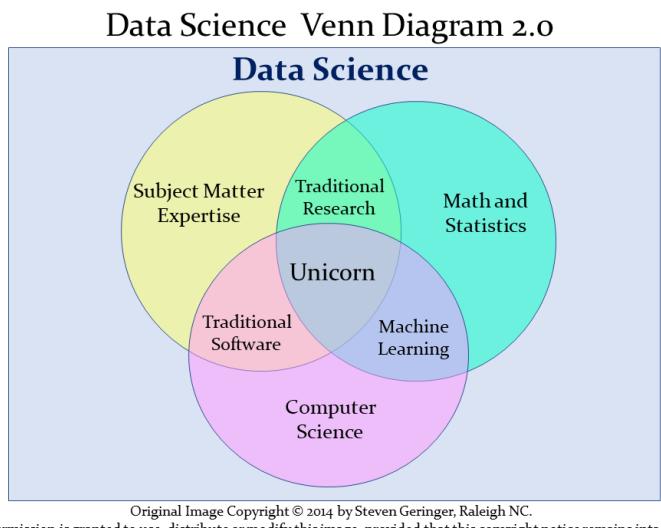


Figure 1.1: Data Science Venn Diagram from Steven Geringer

- 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 skillset, 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 (Grolemund and Wickham, 2019).

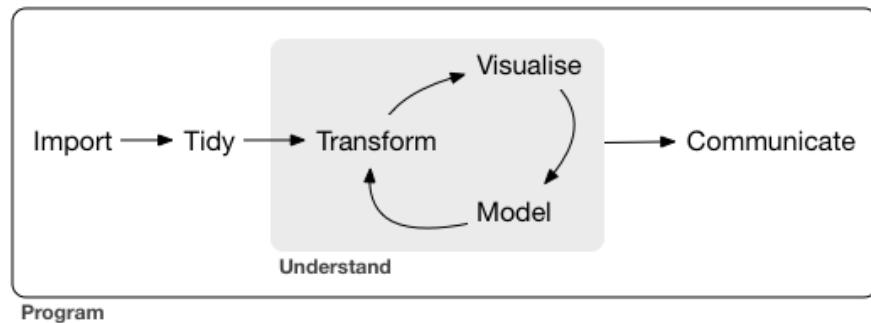


Figure 1.2: Source: R for Data Science: Introduction

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 Grolemund and Wickham (2019).

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 R Markdown, even if they are completely new to coding. A gentle introduction is provided at Ismay and Kim (2019)
- 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 Grolemund and Wickham (2019). 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 A. Exploring Data

Chapter 2

Looking at the Palmer Penguins

The data in the `palmerpenguins` package in R include size measurements, clutch observations, and blood isotope ratios for adult foraging Adélie, Chinstrap, and Gentoo penguins observed on islands in the Palmer Archipelago near Palmer Station, Antarctica. The data were collected and made available by Dr. Kristen Gorman and the Palmer Station Long Term Ecological Research (LTER) Program.

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

2.1 Package Loading, then Dealing with Missing Data

To start, let's load up the necessary R packages to manage the data and summarize it in a small table, and a plot. We've actually done this previously, but we'll repeat the steps here, because it's worth seeing what R is doing.

In this case, we'll load up five packages.

```
library(palmerpenguins) # source for the data set
library(janitor)        # some utilities for cleanup and simple tables
library(magrittr)       # provides us with the pipe %>% for code management
library(dplyr)          # part of the tidyverse: data management tools
library(ggplot2)         # part of the tidyverse: tools for plotting data
```

It's worth remembering that everything after the `#` on each line above is just a comment for the reader, and is ignored by R. We'll see later that the loading

of a single package (called `tidyverse`) gives us both the `dplyr` and `ggplot2` packages, as well as several other useful things.

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 data frame (think of this as a data set) called `new_penguins` and then take a look at that result using the following code.

```
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>          <dbl>       <int>
1 Adelie  Torgo~     39.1        18.7           181        3750
2 Adelie  Torgo~     39.5        17.4           186        3800
3 Adelie  Torgo~     40.3        18              195        3250
4 Adelie  Torgo~     36.7        19.3           193        3450
5 Adelie  Torgo~     39.3        20.6           190        3650
6 Adelie  Torgo~     38.9        17.8           181        3625
7 Adelie  Torgo~     39.2        19.6           195        4675
8 Adelie  Torgo~     41.1        17.6           182        3200
9 Adelie  Torgo~     38.6        21.2           191        3800
10 Adelie  Torgo~    34.6        21.1           198        4400
# ... with 323 more rows, and 2 more variables: sex <fct>, year <int>
```

2.2 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?

```
new_penguins %>%
  tabyl(sex, species) # tabyl comes from the janitor package
```

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

Note the strange spelling of `tabyl` here. The output is reasonably clear, but could we make that table a little prettier, and while we're at it, can we add the row and column totals to it?

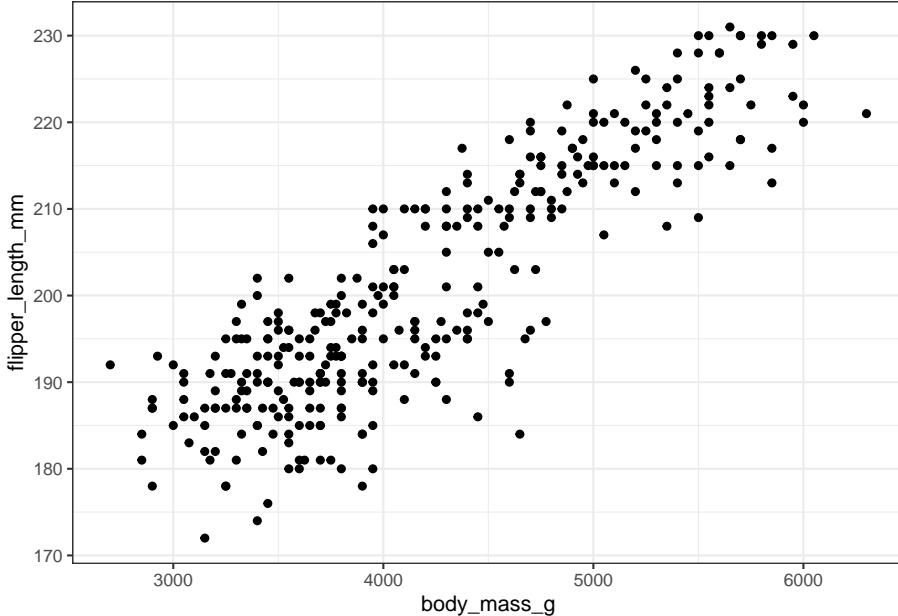
```
new_penguins %>%
  tabyl(sex, species) %>%
  adorn_totals(where = c("row", "col")) %>% # add row, column totals
  kable # one convenient way to make the table prettier
```

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

2.3 Visualizing the Data in a Graph (or a few...)

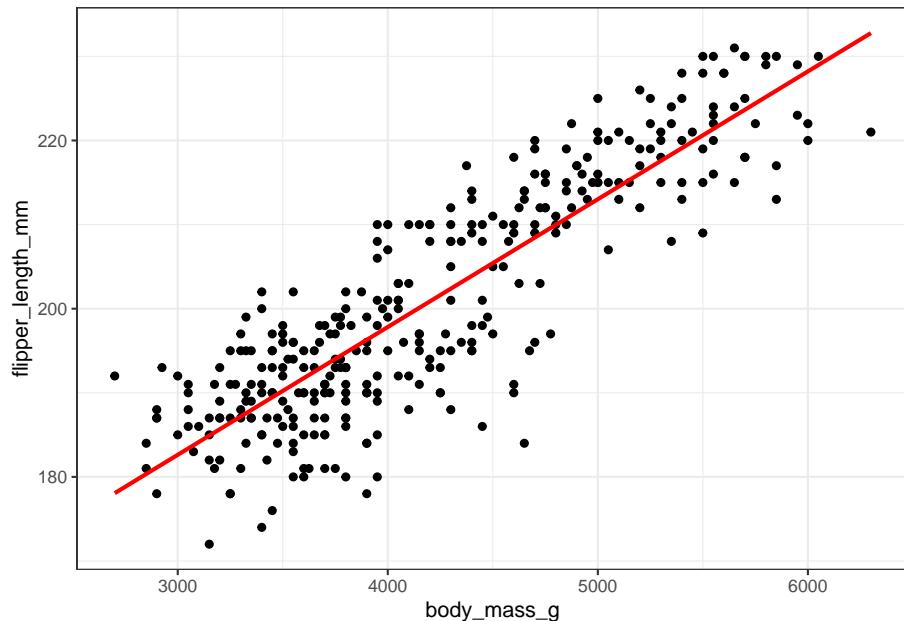
Now, let's look at the other two variables of interest. Let's create a graph showing the association of body mass with flipper length across the complete set of 333 penguins.

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



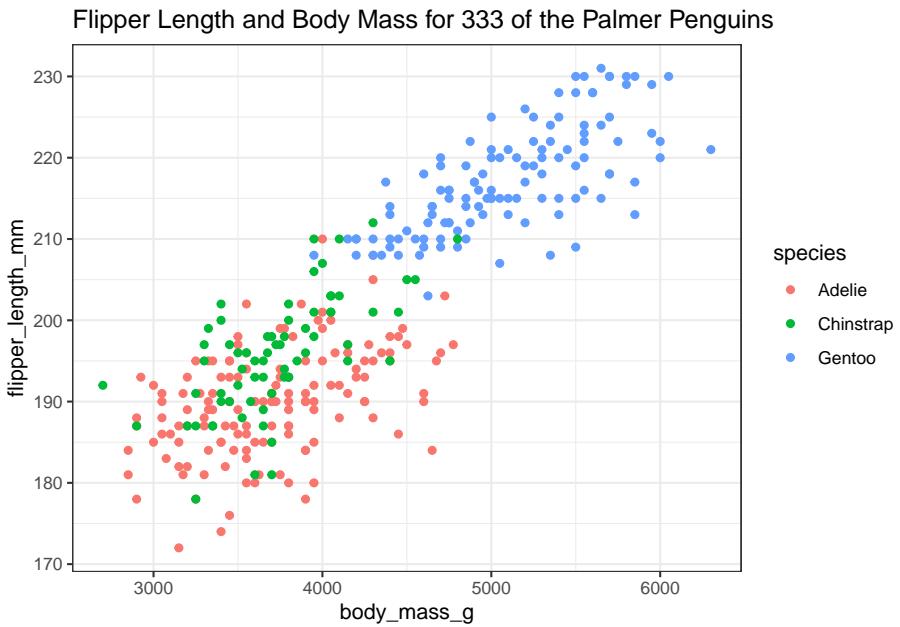
Some of you may want to include a straight-line model (fit by a classical linear regression) to this plot. One way to do that in R involves the addition of a single line of code, like this:

```
ggplot(new_penguins, aes(x = body_mass_g, y = flipper_length_mm)) +
  geom_point() +
  geom_smooth(method = "lm", 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 neaten up the theme (to get rid of the default grey background) and add a title, like this.

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

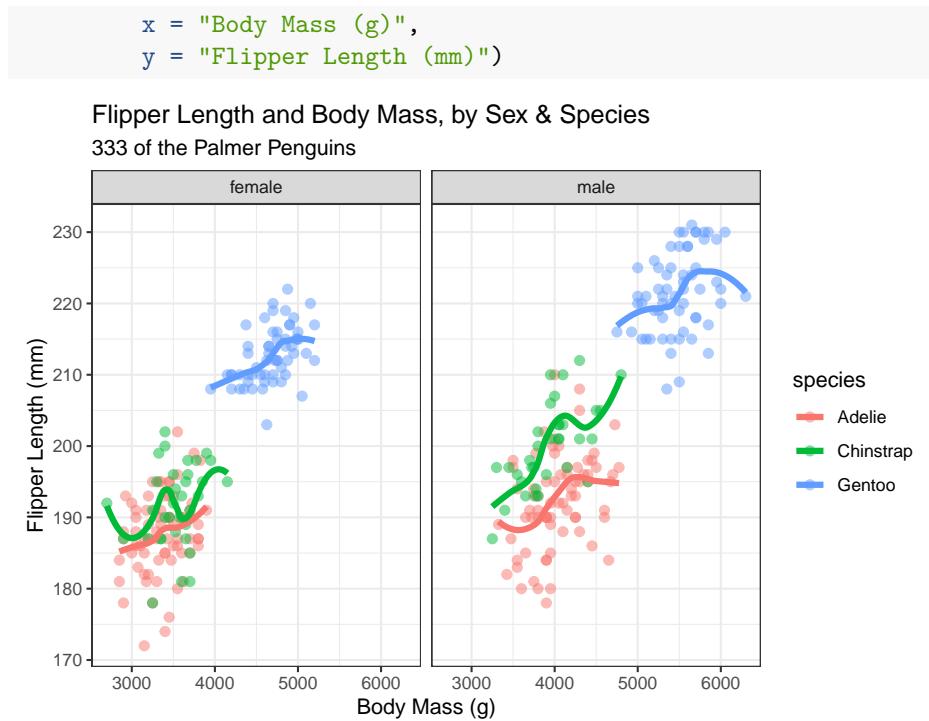


2.4 Six Ways To “Improve” This Graph

Now, let’s build a new graph. 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", se = FALSE, size = 1.5) +
  facet_grid(~ sex) +
  theme_bw() +
  labs(title = "Flipper Length and Body Mass, by Sex & Species",
       subtitle = paste0(nrow(new_penguins), " of the Palmer Penguins"))
```



2.5 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?

Chapter 3

NHANES: Initial Exploring

We'll start by visualizing some data from the US National Health and Nutrition Examination Survey, or NHANES. We'll display R code as we go, but we'll return to all of the key coding ideas involved later in the Notes.

3.1 The NHANES data: Collecting a Sample

To begin, we'll gather a random sample of 1,000 subjects participating in NHANES, and then identify several variables of interest about those subjects¹. Some of the motivation for this example came from a Figure in Baumer et al. (2017).

```
# library(NHANES) # already loaded NHANES package/library of functions, data
set.seed(431001)
# use set.seed to ensure that we all get the same random sample
# of 1,000 NHANES subjects in our nh_data collection

nh_dat1 <- sample_n(NHANES, size = 1000) %>%
  select(ID, Gender, Age, Height)

nh_dat1
# A tibble: 1,000 x 4
  ID   Gender   Age Height
  <int> <fct>   <int>  <dbl>
1 69638 female     5    106.
2 70782 male      64    176.
```

¹For more on the NHANES data available in the NHANES package, type ?NHANES in the Console in R Studio.

```

3 52408 female    54   162.
4 59031 female    15   155.
5 64530 male      53   185.
6 71040 male      63   169.
7 55186 female    30   168.
8 60211 male      5    103.
9 55730 male      66   161.
10 68229 female   36   170.
# ... with 990 more rows

```

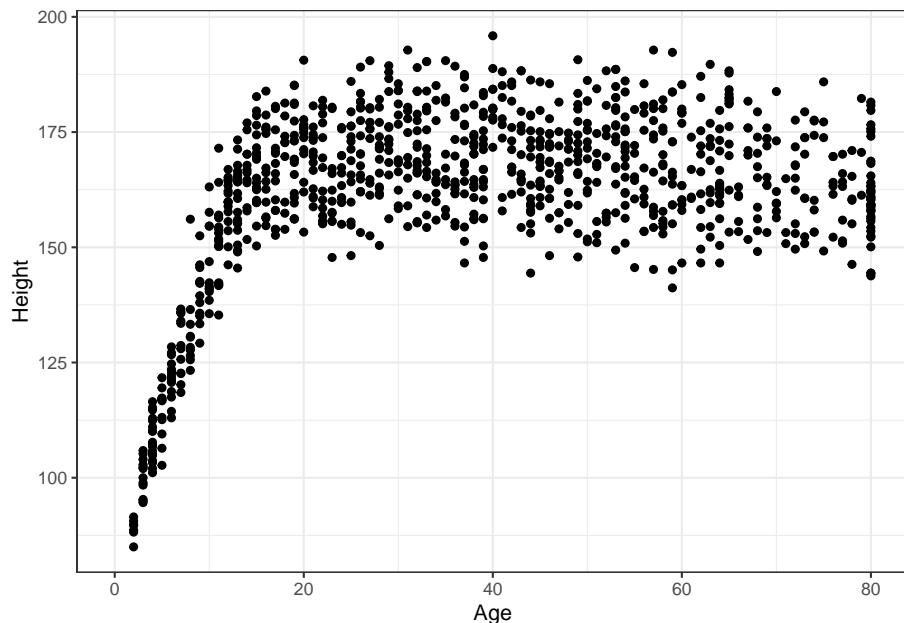
We have 1000 rows (observations) and 4 columns (variables) that describe the subjects listed in the rows.

3.2 Age and Height

Suppose we want to visualize the relationship of Height and Age in our 1,000 NHANES observations. The best choice is likely to be a scatterplot.

```
ggplot(data = nh_dat1, aes(x = Age, y = Height)) +
  geom_point()
```

Warning: Removed 37 rows containing missing values (geom_point).



We note several interesting results here.

1. As a warning, R tells us that it has “Removed 37 rows containing missing values (geom_point).” Only 963 subjects plotted here, because the remaining 37 people have missing (NA) values for either Height, Age or both.
2. Unsurprisingly, the measured Heights of subjects grow from Age 0 to Age 20 or so, and we see that a typical Height increases rapidly across these Ages. The middle of the distribution at later Ages is pretty consistent at around a Height somewhere between 150 and 175. The units aren’t specified, but we expect they must be centimeters. The Ages are clearly reported in Years.
3. No Age is reported over 80, and it appears that there is a large cluster of Ages at 80. This may be due to a requirement that Ages 80 and above be reported at 80 so as to help mask the identity of those individuals.²

As in this case, we’re going to build most of our visualizations using tools from the `ggplot2` package, which is part of the `tidyverse` series of packages. You’ll see similar coding structures throughout this Chapter, most of which are covered as well in Chapter 3 of Grolemund and Wickham (2019).

3.3 Subset of Subjects with Known Age and Height

Before we move on, let’s manipulate the data set a bit, to focus on only those subjects who have complete data on both Age and Height. This will help us avoid that warning message.

```
nh_dat2 <- nh_dat1 %>%
  filter(complete.cases(Age, Height))

summary(nh_dat2)
```

	ID	Gender	Age	Height
Min.	:51624	female:484	Min. : 2.00	Min. : 85.0
1st Qu.	:57034	male :479	1st Qu.:19.00	1st Qu.:156.2
Median	:62056		Median :37.00	Median :165.0
Mean	:61967		Mean :38.29	Mean :162.3
3rd Qu.	:67269		3rd Qu.:56.00	3rd Qu.:174.5
Max.	:71875		Max. :80.00	Max. :195.9

Note that the units and explanations for these variables are contained in the NHANES help file, available via typing `?NHANES` in the Console of R Studio, or by typing `NHANES` into the Search bar in R Studio’s Help window.

²If you visit the NHANES help file with `?NHANES`, you will see that subjects 80 years or older were indeed recorded as 80.

3.3.1 The Distinction between Gender and Sex

The `Gender` variable here is a mistake. These data refer to the biological status of these subjects, which is their `Sex`, and not the social construct of `Gender` which can be quite different. In our effort to avoid further confusion, we'll rename the variable `Gender` to instead more accurately describe what is actually measured here.

To do this, we can use this approach...

```
nh_dat2 <- nh_dat1 %>%
  rename(Sex = Gender) %>%
  filter(complete.cases(Age, Height))

summary(nh_dat2)
```

	ID	Sex	Age	Height
Min.	:51624	female:484	Min. : 2.00	Min. : 85.0
1st Qu.	:57034	male :479	1st Qu.:19.00	1st Qu.:156.2
Median	:62056		Median :37.00	Median :165.0
Mean	:61967		Mean :38.29	Mean :162.3
3rd Qu.	:67269		3rd Qu.:56.00	3rd Qu.:174.5
Max.	:71875		Max. :80.00	Max. :195.9

That's better. How many observations do we have now? We could use `dim` to find out the number of rows and columns in this new data set.

```
dim(nh_dat2)
```

```
[1] 963 4
```

Or, we could simply list the data set and read off the result.

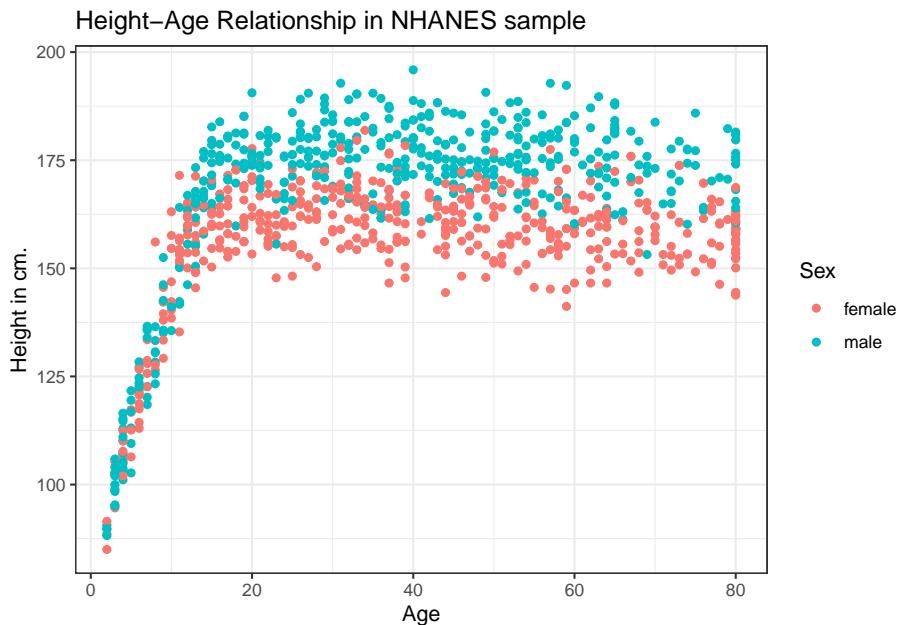
```
nh_dat2
```

```
# A tibble: 963 x 4
  ID   Sex     Age Height
  <int> <fct>  <int>  <dbl>
1 69638 female    5   106.
2 70782 male     64   176.
3 52408 female    54   162.
4 59031 female    15   155.
5 64530 male     53   185.
6 71040 male     63   169.
7 55186 female    30   168.
8 60211 male     5    103.
9 55730 male     66   161.
10 68229 female   36   170.
# ... with 953 more rows
```

3.4 Age-Height and Sex?

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

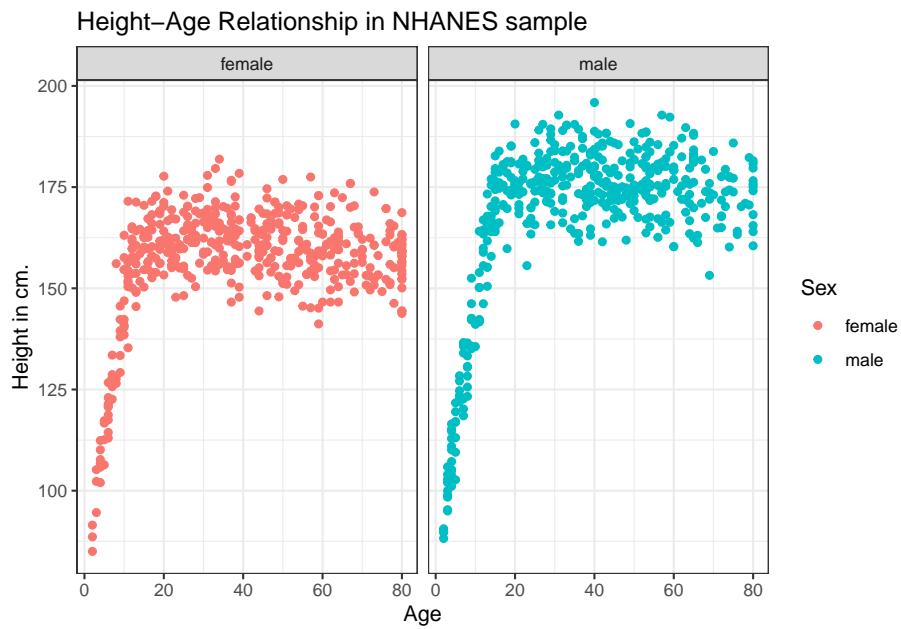
```
ggplot(data = nh_dat2, aes(x = Age, y = Height, color = Sex)) +
  geom_point() +
  labs(title = "Height-Age Relationship in NHANES sample",
       y = "Height in cm.")
```



3.4.1 Can we show the Female and Male relationships in separate panels?

Sure.

```
ggplot(data = nh_dat2, aes(x = Age, y = Height, color = Sex)) +
  geom_point() +
  labs(title = "Height-Age Relationship in NHANES sample",
       y = "Height in cm.") +
  facet_wrap(~ Sex)
```

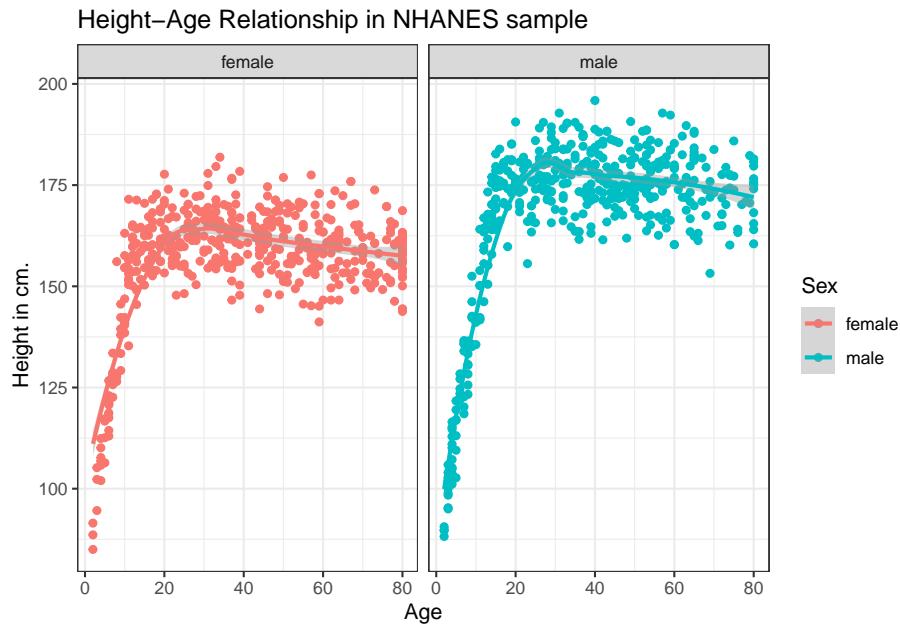


3.4.2 Can we add a smooth curve to show the relationship in each plot?

Yep, and let's change the theme of the graph to remove the gray background, too.

```
ggplot(data = nh_dat2, aes(x = Age, y = Height, color = Sex)) +
  geom_point() +
  geom_smooth(method = "loess") +
  labs(title = "Height–Age Relationship in NHANES sample",
       y = "Height in cm.") +
  theme_bw() +
  facet_wrap(~ Sex)

`geom_smooth()` using formula 'y ~ x'
```

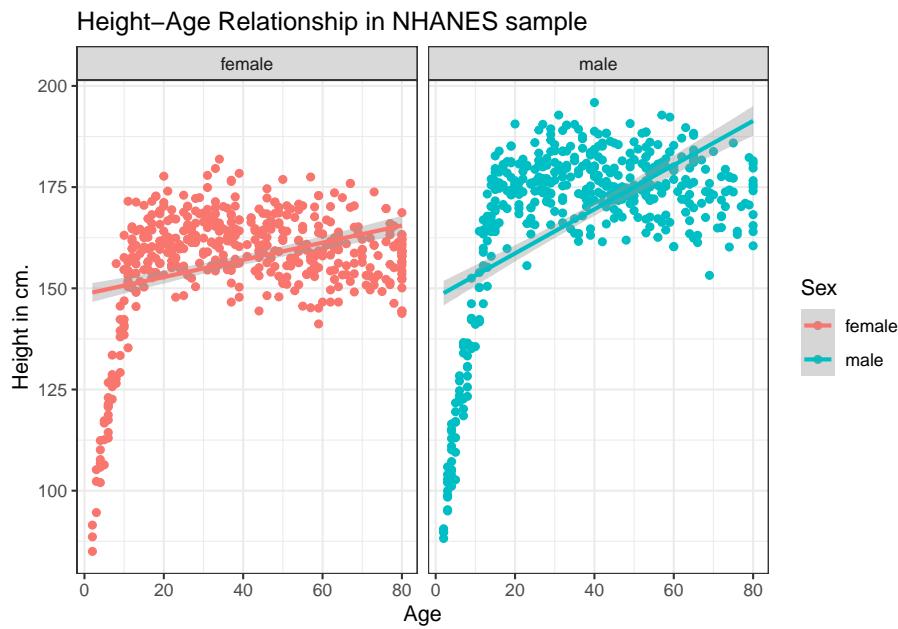


3.4.3 What if we want to assume straight line relationships?

We could look at a linear model in the plot. Does this make sense here?

```
ggplot(data = nh_dat2, aes(x = Age, y = Height, color = Sex)) +
  geom_point() +
  geom_smooth(method = "lm") +
  labs(title = "Height-Age Relationship in NHANES sample",
       y = "Height in cm.") +
  theme_bw() +
  facet_wrap(~ Sex)

`geom_smooth()` using formula 'y ~ x'
```



3.5 Creating A New Subset: Ages 21-79

Suppose we wanted to look only at those observations (subjects) whose Age is at least 21 and at most 79. Suppose also that we want to look at some of the additional variables available in NHANES. To start, we'll do the following:

1. Set the same seed for random sampling that we used earlier, so that we start with the original sample of 1000 people we built earlier. Draw that same sample of 1,000 people.
2. Filter the sample to only those people whose age is more than 20 and less than 80 years.
3. Select the variables we will use in the rest of this chapter:
 - **Age** as we've seen before, in years.
 - **Height** as we've seen before, in centimeters.
 - **Gender** which we'll rename as **Sex** again.
 - **Pulse** = 60 second pulse rate (in beats per minute).
 - **BPSysAve** = Systolic Blood Pressure, in mm Hg (and we'll rename this **SBP**).
 - **SleepTrouble** = Yes means the subject has told a health professional that they had trouble sleeping.
 - **PhysActive** = Yes means the subject does moderate or vigorous-intensity sports, fitness or recreational activity.
 - **MaritalStatus** = one of Married, Widowed, Divorced, Separated, NeverMarried or LivePartner (living with partner.)

- `HealthGen` = self-reported rating of general health, one of Excellent, Vgood (Very Good), Good, Fair or Poor.
4. Rename `Gender` as `Sex`, to more accurately describe what is being measured.
 5. Omit subjects with any missingness on *any* of the variables we've selected.

Can you see how the code below accomplishes these tasks?

```
set.seed(431001) # again, this will ensure the same sample

nh_dat3 <- sample_n(NHANES, size = 1000) %>%
  filter(Age > 20 & Age < 80) %>%
  select(ID, Gender, Age, Height,
         Pulse, BPSysAve, SleepTrouble, PhysActive,
         MaritalStatus, HealthGen) %>%
  rename(Sex = Gender, SBP = BPSysAve) %>%
  na.omit

nh_dat3

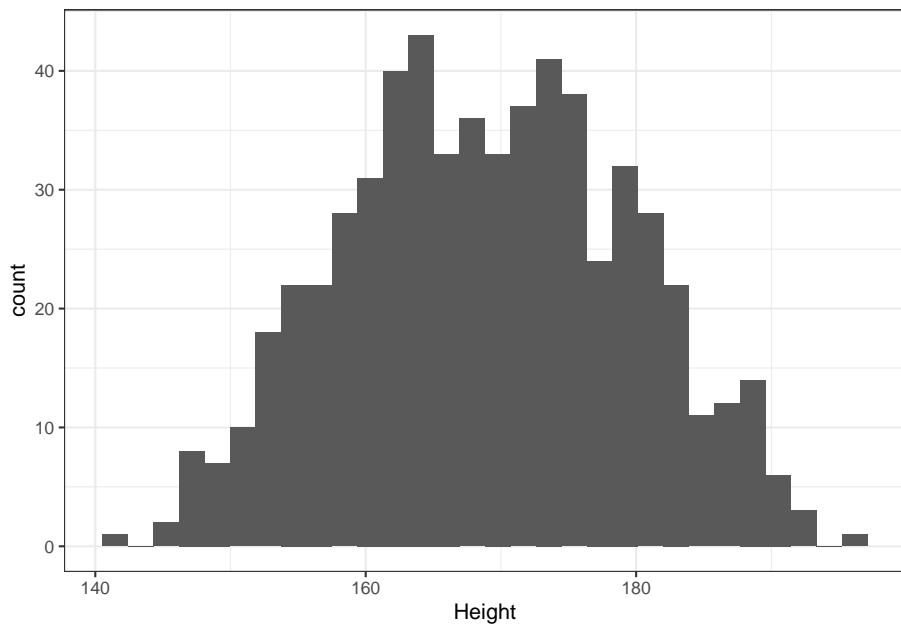
# A tibble: 603 x 10
  ID Sex     Age Height Pulse   SBP SleepTrouble PhysActive MaritalStatus
  <int> <fct> <int>  <dbl> <int> <fct>      <fct>      <fct>
1 70782 male    64    176.    78   127 No        No       Married
2 52408 fema~   54    162.    80   135 No        No       LivePartner
3 64530 male    53    185.    100   131 No        No       Married
4 71040 male    63    169.    70    124 Yes       Yes      Married
5 55186 fema~   30    168.    76    107 No        No       Married
6 55730 male    66    161.    78    133 No        No       Married
7 68229 fema~   36    170.    90    105 No        Yes      Married
8 63762 male    23    180.    66    118 No        No       Married
9 66290 fema~   63    162.    88    116 No        No       Married
10 66984 male   75    174.    84    141 No        No       Married
# ... with 593 more rows, and 1 more variable: HealthGen <fct>
```

3.6 Distribution of Heights

What is the distribution of height in this new sample?

```
ggplot(data = nh_dat3, aes(x = Height)) +
  geom_histogram()

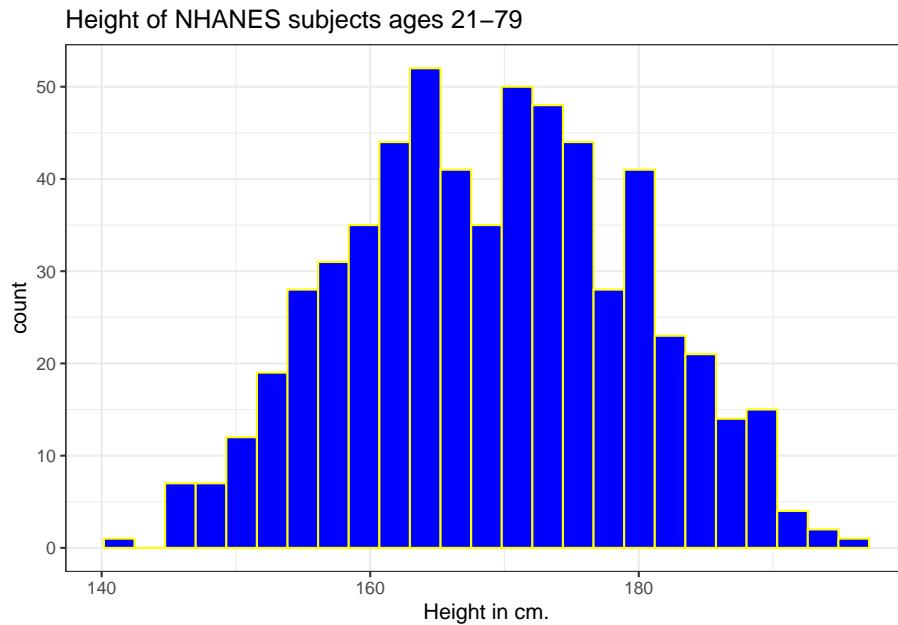
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



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.

```
ggpplot(data = nh_dat3, aes(x = Height)) +
  geom_histogram(bins = 25, col = "yellow", fill = "blue") +
  labs(title = "Height of NHANES subjects ages 21-79",
       x = "Height in cm.")
```

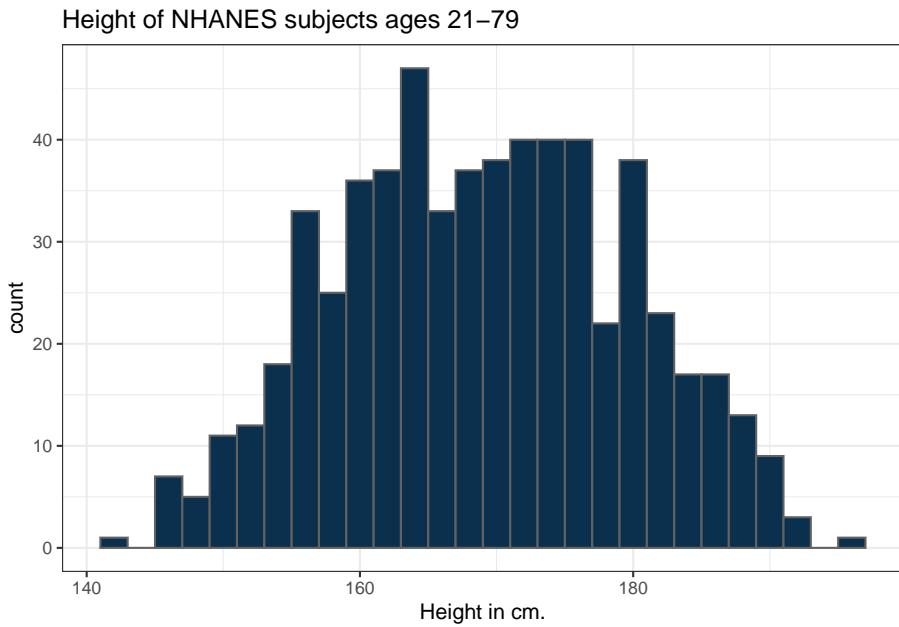


3.6.1 Changing a Histogram's Fill and Color

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.

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

ggplot(data = nh_dat3, aes(x = Height)) +
  geom_histogram(binwidth = 2, col = cwru.gray, fill = cwru.blue) +
  labs(title = "Height of NHANES subjects ages 21-79",
       x = "Height in cm.") +
  theme_bw()
```

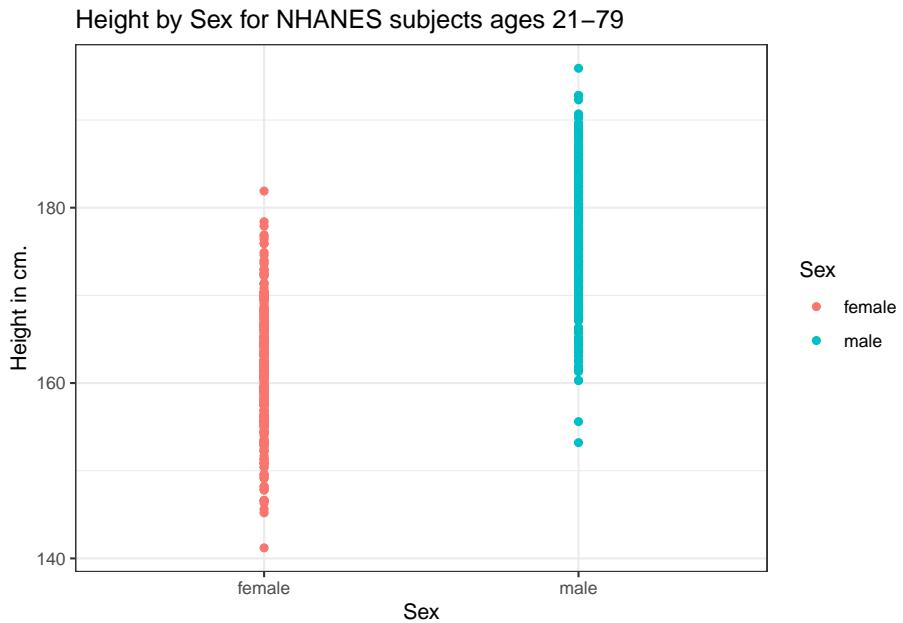


Note the other changes to the graph above.

1. We changed the theme to replace the gray background.
2. We changed the bins for the histogram, to gather observations into groups of 2 cm. each.

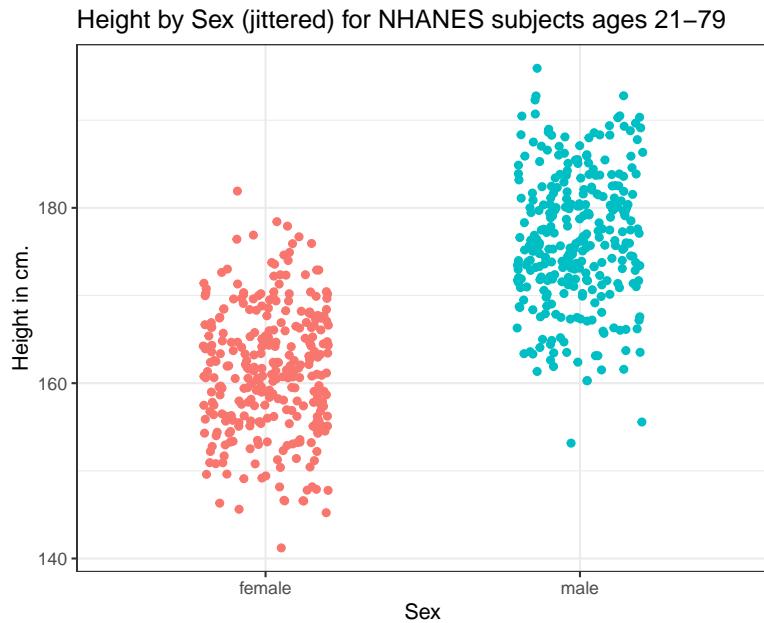
3.7 Height and Sex

```
ggplot(data = nh_dat3, aes(x = Sex, y = Height, color = Sex)) +
  geom_point() +
  labs(title = "Height by Sex for NHANES subjects ages 21-79",
       y = "Height in cm.")
```



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

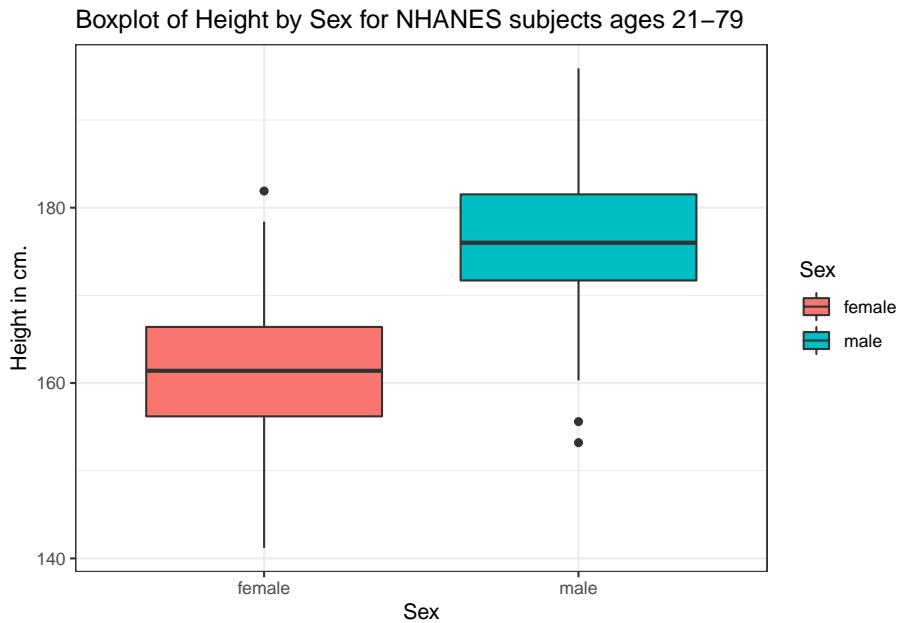
```
ggplot(data = nh_dat3, aes(x = Sex, y = Height, color = Sex)) +
  geom_jitter(width = 0.2) +
  labs(title = "Height by Sex (jittered) for NHANES subjects ages 21-79",
       y = "Height in cm.")
```



Perhaps it might be better to summarise the distribution in a different way. We might consider a boxplot of the data.

3.7.1 A Boxplot of Height by Sex

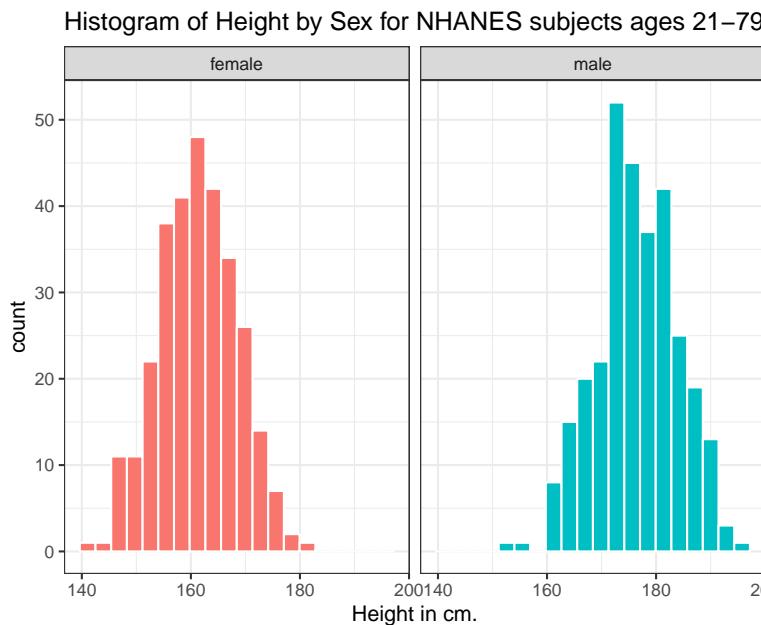
```
ggplot(data = nh_dat3, aes(x = Sex, y = Height, fill = Sex)) +
  geom_boxplot() +
  labs(title = "Boxplot of Height by Sex for NHANES subjects ages 21-79",
       y = "Height in cm.")
```



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

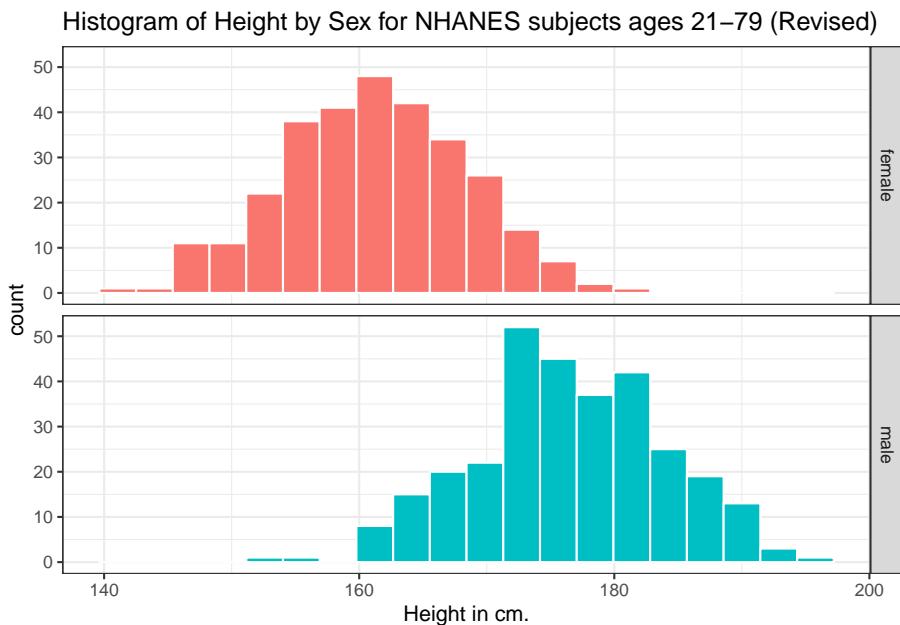
3.7.2 Histograms of Height by Sex

```
ggplot(data = nh_dat3, aes(x = Height, fill = Sex)) +  
  geom_histogram(color = "white", bins = 20) +  
  labs(title = "Histogram of Height by Sex for NHANES subjects ages 21-79",  
       x = "Height in cm.") +  
  facet_wrap(~ Sex)
```



Can we redraw these histograms so that they are a little more comparable, and to get rid of the unnecessary legend?

```
ggplot(data = nh_dat3, aes(x = Height, fill = Sex)) +
  geom_histogram(color = "white", bins = 20) +
  labs(title = "Histogram of Height by Sex for NHANES subjects ages 21–79 (Revised)",
       x = "Height in cm.") +
  guides(fill = FALSE) +
  facet_grid(Sex ~ .)
```

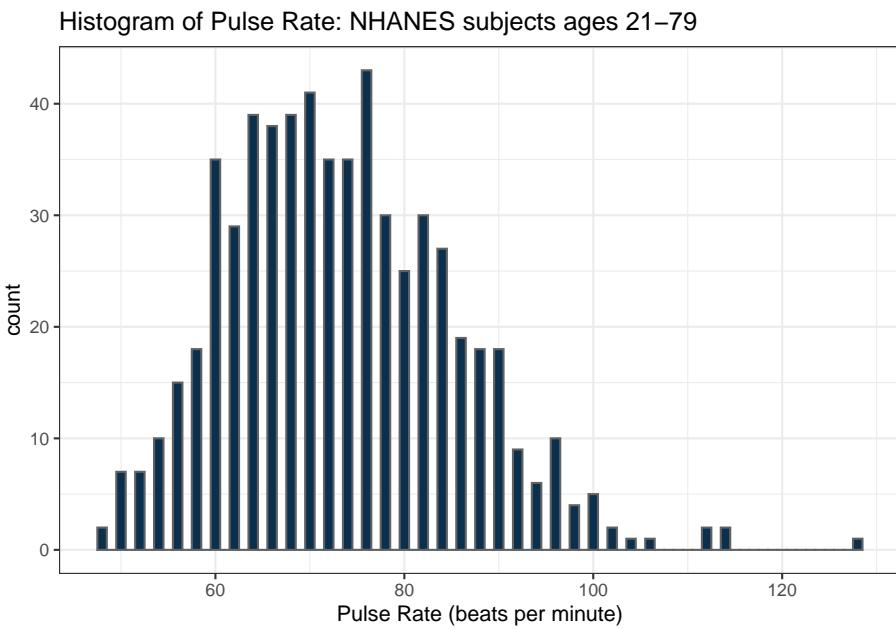


3.8 Looking at Pulse Rate

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

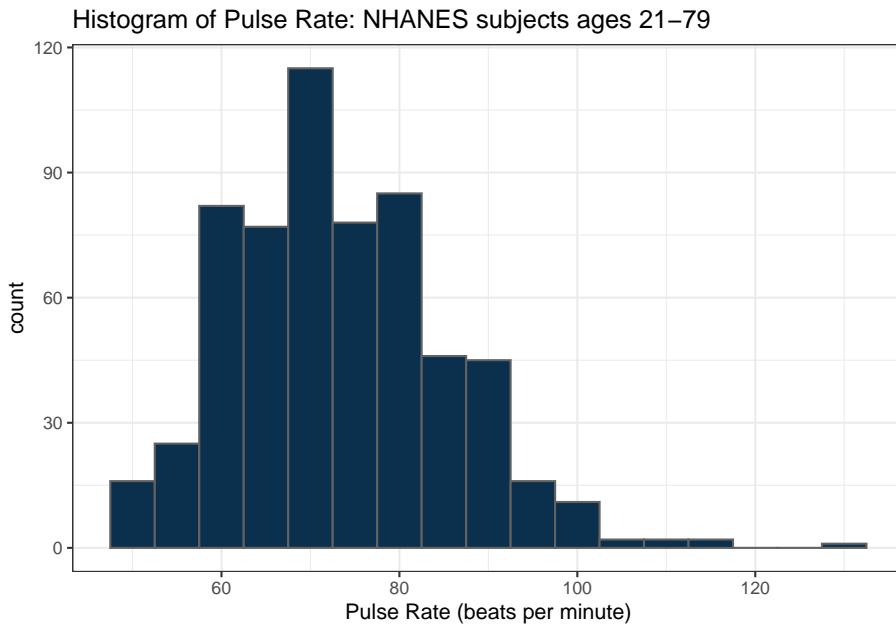
Here's a histogram, again with CWRU colors, for the pulse rates in our sample.

```
ggplot(data = nh_dat3, aes(x = Pulse)) +
  geom_histogram(binwidth = 1, fill = cwrublue, col = cwrugray) +
  labs(title = "Histogram of Pulse Rate: NHANES subjects ages 21-79",
       x = "Pulse Rate (beats per minute)")
```



Suppose we instead bin up groups of 5 beats per minute together as we plot the Pulse rates.

```
ggplot(data = nh_dat3, aes(x = Pulse)) +
  geom_histogram(binwidth = 5, fill = cwrugrey, col = cwrugray) +
  labs(title = "Histogram of Pulse Rate: NHANES subjects ages 21-79",
       x = "Pulse Rate (beats per minute)")
```

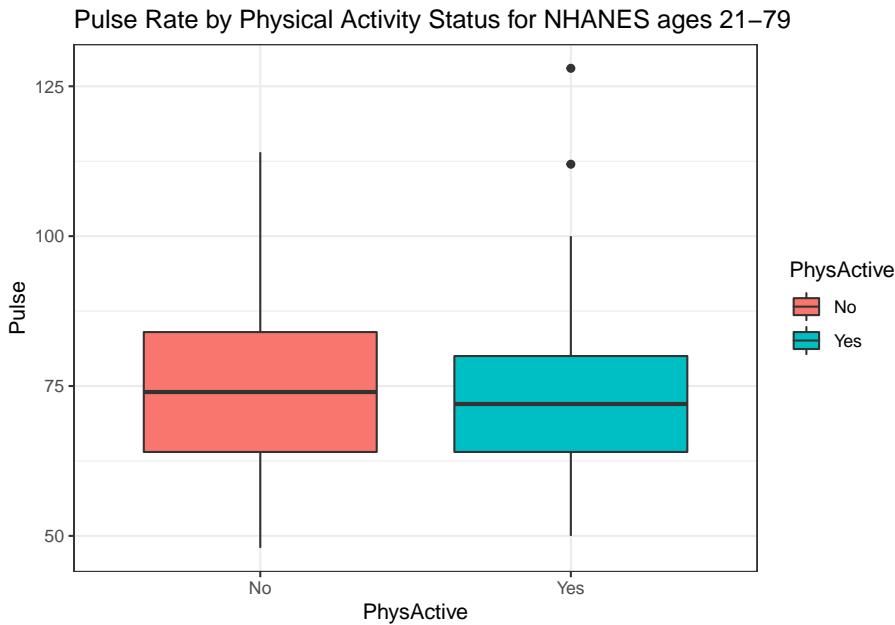


Which is the more useful representation will depend a lot on what questions you're trying to answer.

3.8.1 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_dat3, aes(y = Pulse, x = PhysActive, fill = PhysActive)) +
  geom_boxplot() +
  labs(title = "Pulse Rate by Physical Activity Status for NHANES ages 21-79")
```



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_dat3 %>%
  group_by(PhysActive) %>%
  summarise(count = n(), mean(Pulse), median(Pulse)) %>%
  knitr::kable(digits = 2)
```

``summarise()` ungrouping output (override with `.`groups` argument)`

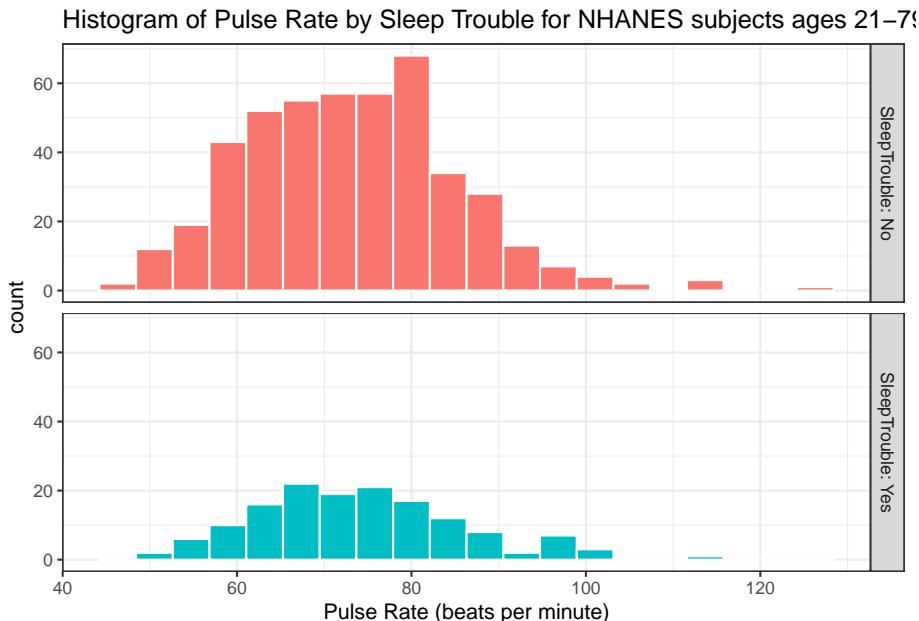
PhysActive	count	mean(Pulse)	median(Pulse)
No	293	74.21	74
Yes	310	72.37	72

The `knitr::kable(digits = 2)` piece of this command tells R Markdown to generate a table with some attractive formatting, and rounding any decimals to two figures.

3.8.2 Pulse by Sleeping Trouble

```
ggplot(data = nh_dat3, aes(x = Pulse, fill = SleepTrouble)) +
  geom_histogram(color = "white", bins = 20) +
  labs(title = "Histogram of Pulse Rate by Sleep Trouble for NHANES subjects ages 21–79",
       x = "Pulse Rate (beats per minute)") +
```

```
guides(fill = FALSE) +
  facet_grid(SleepTrouble ~ ., labeller = "label_both")
```



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

```
nh_dat3 %>%
  group_by(SleepTrouble) %>%
  summarise(count = n(), mean(Pulse), median(Pulse)) %>%
  knitr::kable(digits = 2)
```

`summarise()` ungrouping output (override with `.`groups` argument)

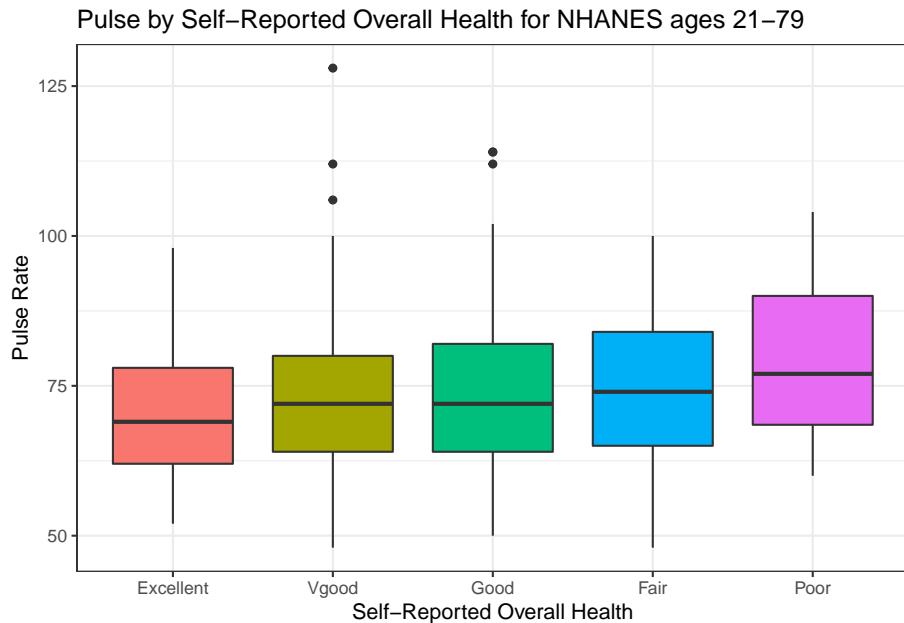
SleepTrouble	count	mean(Pulse)	median(Pulse)
No	457	73.05	72
Yes	146	73.96	72

3.8.3 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_dat3, aes(x = HealthGen, y = Pulse, fill = HealthGen)) +
  geom_boxplot() +
  labs(title = "Pulse by Self-Reported Overall Health for NHANES ages 21-79",
```

```
x = "Self-Reported Overall Health", y = "Pulse Rate") +
guides(fill = FALSE)
```



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

```
nh_dat3 %>%
  group_by(HealthGen) %>%
  summarise(count = n(), mean(Pulse), median(Pulse)) %>%
  knitr::kable(digits = 2)
```

``summarise()` ungrouping output (override with `~.groups` argument)`

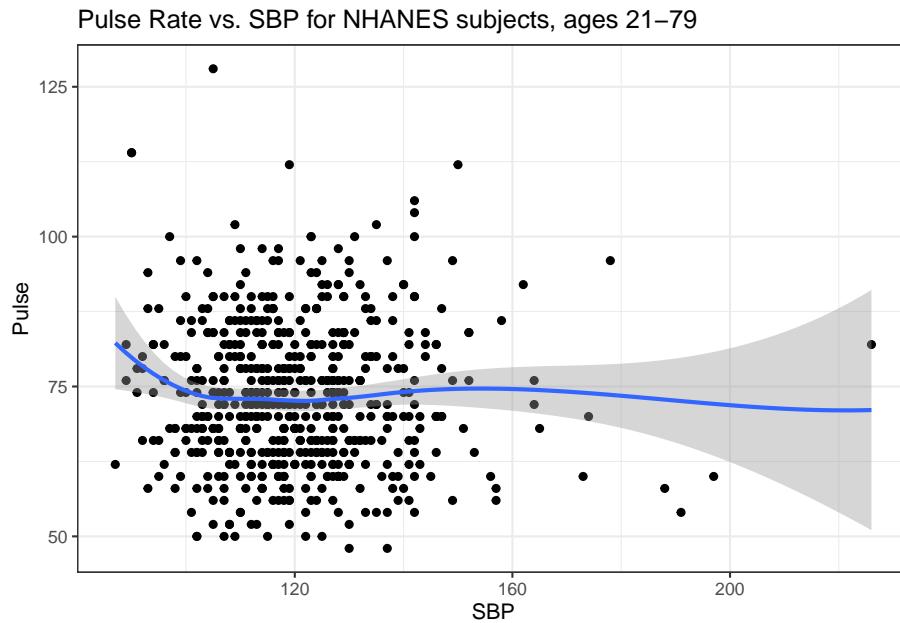
HealthGen	count	mean(Pulse)	median(Pulse)
Excellent	64	69.97	69
Vgood	196	72.81	72
Good	238	73.66	72
Fair	83	74.22	74
Poor	22	79.09	77

3.8.4 Pulse Rate and Systolic Blood Pressure

```
ggplot(data = nh_dat3, aes(x = SBP, y = Pulse)) +
geom_point() +
```

```
geom_smooth(method = "loess") +
  labs(title = "Pulse Rate vs. SBP for NHANES subjects, ages 21-79")

`geom_smooth()` using formula 'y ~ x'
```



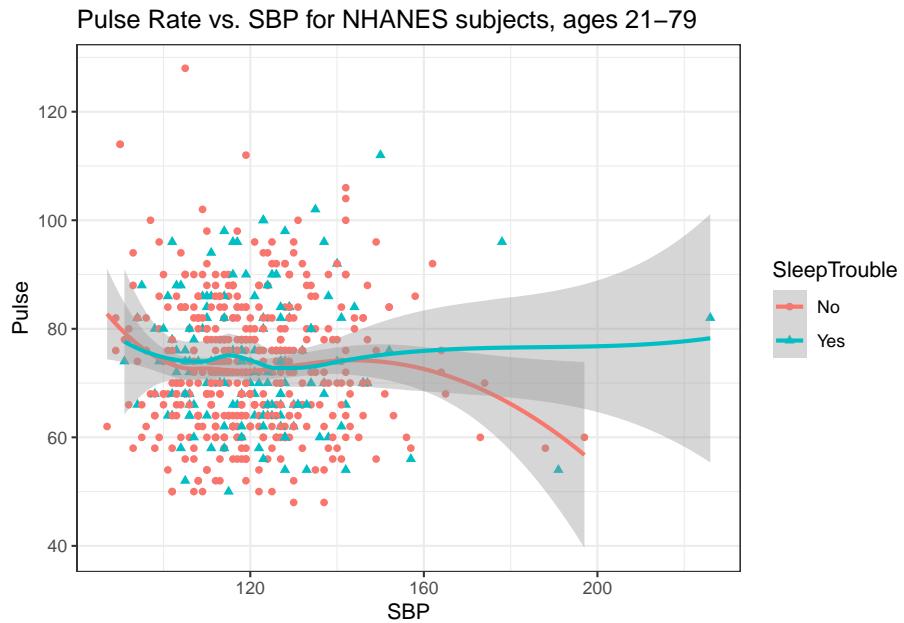
3.8.5 Sleep Trouble vs. No Sleep Trouble?

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_dat3,
       aes(x = SBP, y = Pulse,
           color = SleepTrouble, shape = SleepTrouble)) +
  geom_point() +
  geom_smooth(method = "loess") +
  labs(title = "Pulse Rate vs. SBP for NHANES subjects, ages 21-79")

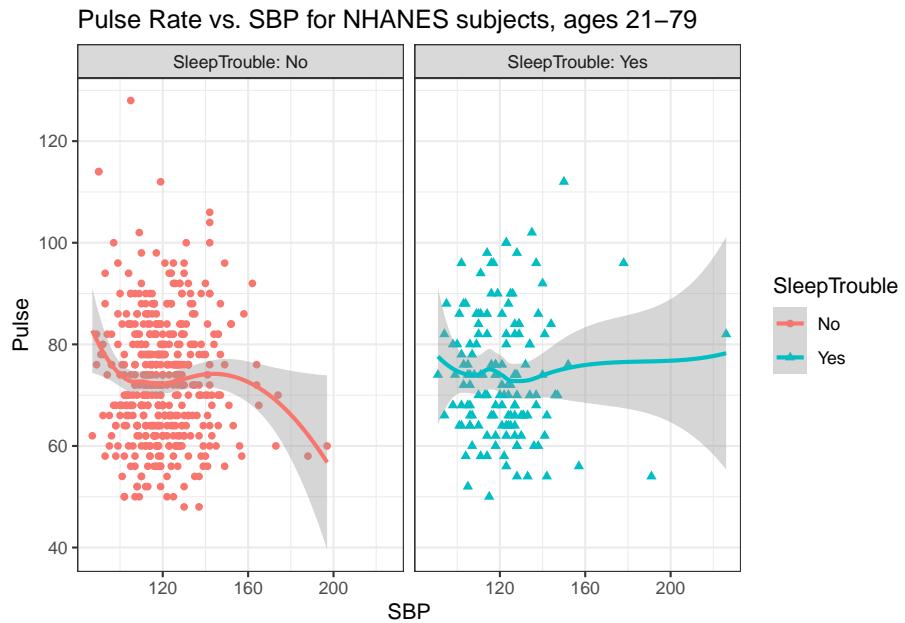
`geom_smooth()` using formula 'y ~ x'
```



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

```
ggplot(data = nh_dat3,
       aes(x = SBP, y = Pulse,
           color = SleepTrouble, shape = SleepTrouble)) +
  geom_point() +
  geom_smooth(method = "loess") +
  labs(title = "Pulse Rate vs. SBP for NHANES subjects, ages 21–79") +
  facet_wrap(~ SleepTrouble, labeller = "label_both")
```

`geom_smooth()` using formula 'y ~ x'



3.9 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_dat3 %>%
  select(HealthGen) %>%
  table()
```

	Excellent	Vgood	Good	Fair	Poor
	64	196	238	83	22

The HealthGen data are categorical, which means that summarizing them with averages isn't as appealing as looking at percentages, proportions and rates.

Another, somewhat simpler way to get a table of this sort of information uses the `tabyl` function from the `janitor` package in R.

```
# tabyl is part of the janitor package
# already loaded: library(janitor)

nh_dat3 %>%
  tabyl(HealthGen)
```

```
HealthGen    n    percent
Excellent   64  0.10613599
Vgood      196  0.32504146
Good       238  0.39469320
Fair        83  0.13764511
Poor        22  0.03648425
```

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_dat3 %>%
  tabyl(HealthGen) %>%
  adorn_totals() %>%
  adorn_pct_formatting()
```

```
HealthGen    n percent
Excellent   64  10.6%
Vgood      196  32.5%
Good       238  39.5%
Fair        83  13.8%
Poor        22  3.6%
Total      603  100.0%
```

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

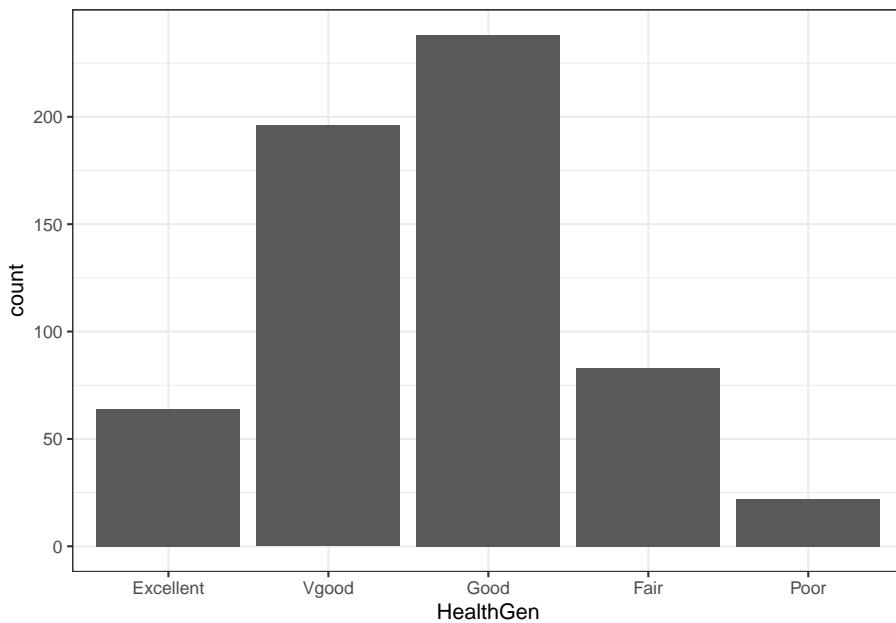
```
nh_dat3 %>%
  tabyl(MaritalStatus) %>%
  adorn_totals() %>%
  adorn_pct_formatting()
```

```
MaritalStatus    n percent
Divorced      61  10.1%
LivePartner   43  7.1%
Married       349  57.9%
NeverMarried  104  17.2%
Separated     8   1.3%
Widowed      38  6.3%
Total        603  100.0%
```

3.9.1 Bar Chart for Categorical Data

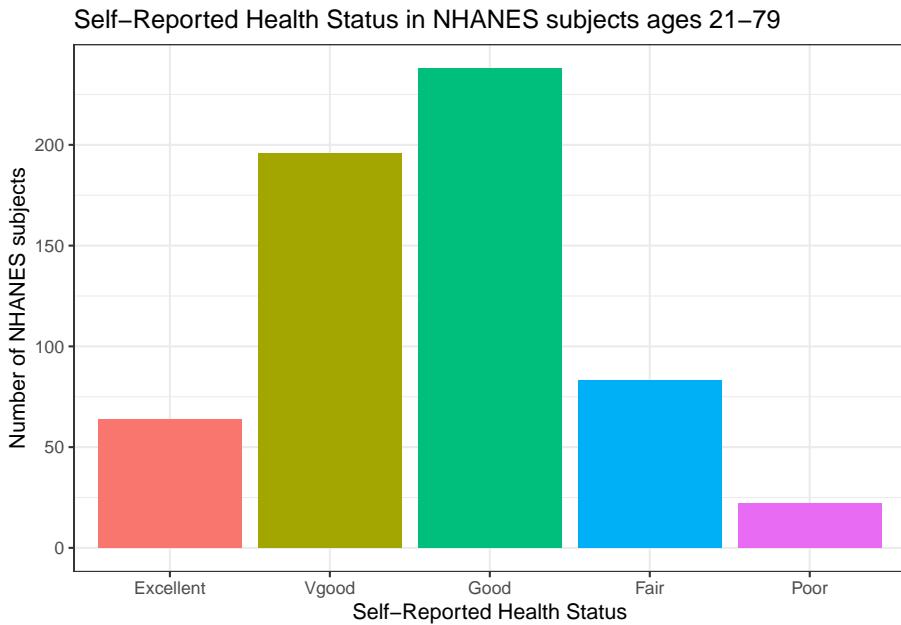
Usually, a **bar chart** is the best choice for a graphing a variable made up of categories.

```
ggplot(data = nh_dat3, aes(x = HealthGen)) +
  geom_bar()
```



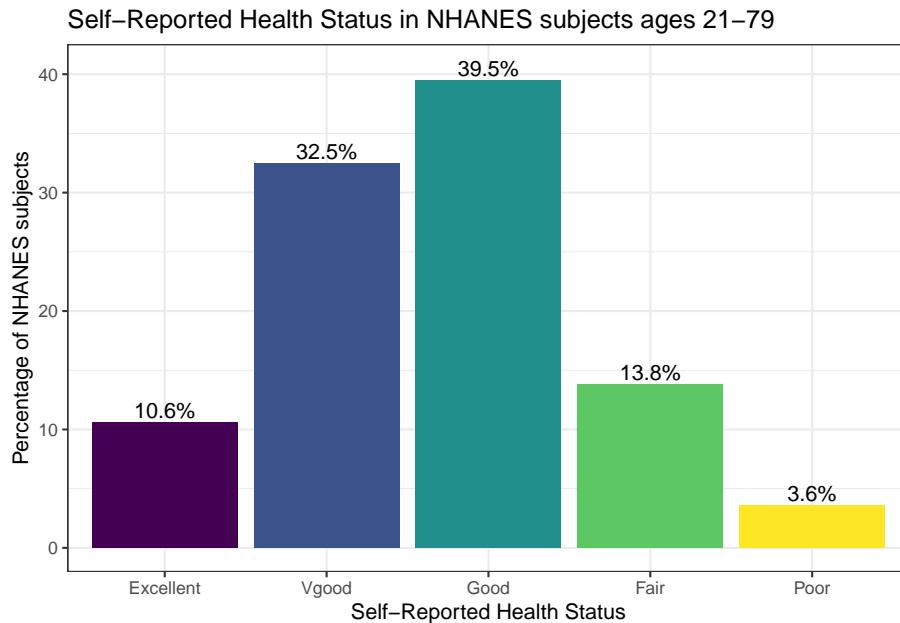
There are lots of things we can do to make this plot fancier.

```
ggplot(data = nh_dat3, aes(x = HealthGen, fill = HealthGen)) +  
  geom_bar() +  
  guides(fill = FALSE) +  
  labs(x = "Self-Reported Health Status",  
       y = "Number of NHANES subjects",  
       title = "Self-Reported Health Status in NHANES subjects ages 21-79")
```



Or, we can really go crazy...

```
nh_dat3 %>%
  count(HealthGen) %>%
  mutate(pct = round(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 = FALSE) +
  geom_text(aes(y = pct + 1,      # nudge above top of bar
                label = paste0(pct, '%')), # prettyfy
            position = position_dodge(width = .9),
            size = 4) +
  labs(x = "Self-Reported Health Status",
       y = "Percentage of NHANES subjects",
       title = "Self-Reported Health Status in NHANES subjects ages 21-79") +
  theme_bw()
```



3.9.2 Working with Tables

We can add both row and column marginal totals, and compare subjects by Sex, as follows...

```
nh_dat3 %>%
  tabyl(Sex, HealthGen) %>%
  adorn_totals(c("row", "col"))
```

	Sex	Excellent	Vgood	Good	Fair	Poor	Total
female		27	96	121	41	14	299
male		37	100	117	42	8	304
Total		64	196	238	83	22	603

If we like, we can make this look a little more polished with the `knitr::kable` function...

```
nh_dat3 %>%
  tabyl(Sex, HealthGen) %>%
  adorn_totals(c("row", "col")) %>%
  knitr::kable()
```

Sex	Excellent	Vgood	Good	Fair	Poor	Total
female	27	96	121	41	14	299
male	37	100	117	42	8	304
Total	64	196	238	83	22	603

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

```
nh_dat3 %>%
  tabyl(Sex, HealthGen) %>%
  adorn_totals("row") %>%
  adorn_percentages("row") %>%
  adorn_pct_formatting() %>%
  adorn_ns() %>%
  knitr::kable()
```

Sex	Excellent	Vgood	Good	Fair	Poor
female	9.0% (27)	32.1% (96)	40.5% (121)	13.7% (41)	4.7% (14)
male	12.2% (37)	32.9% (100)	38.5% (117)	13.8% (42)	2.6% (8)
Total	10.6% (64)	32.5% (196)	39.5% (238)	13.8% (83)	3.6% (22)

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:

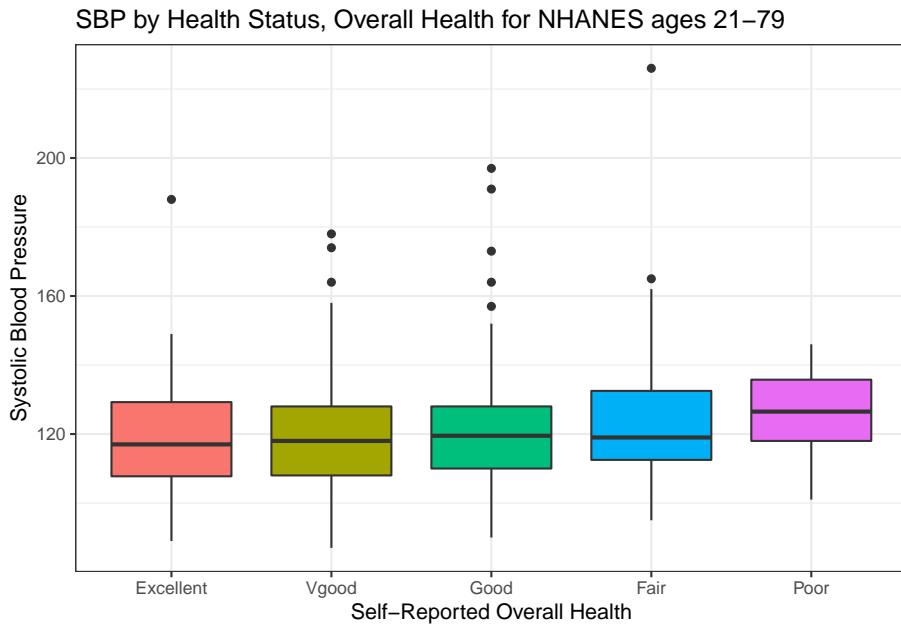
```
nh_dat3 %>%
  tabyl(Sex, HealthGen) %>%
  adorn_totals("col") %>%
  adorn_percentages("col") %>%
  adorn_pct_formatting() %>%
  adorn_ns() %>%
  knitr::kable()
```

Sex	Excellent	Vgood	Good	Fair	Poor	Total
female	42.2% (27)	49.0% (96)	50.8% (121)	49.4% (41)	63.6% (14)	49.6% (299)
male	57.8% (37)	51.0% (100)	49.2% (117)	50.6% (42)	36.4% (8)	50.4% (304)

3.9.3 SBP by General Health Status

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

```
ggplot(data = nh_dat3, aes(x = HealthGen, y = SBP, fill = HealthGen)) +
  geom_boxplot() +
  labs(title = "SBP by Health Status, Overall Health for NHANES ages 21-79",
       y = "Systolic Blood Pressure", x = "Self-Reported Overall Health") +
  guides(fill = FALSE)
```



We can see that not too many people self-identify with the “Poor” health category.

```
nh_dat3 %>%
  group_by(HealthGen) %>%
  summarise(count = n(), mean(SBP), median(SBP)) %>%
  knitr::kable()

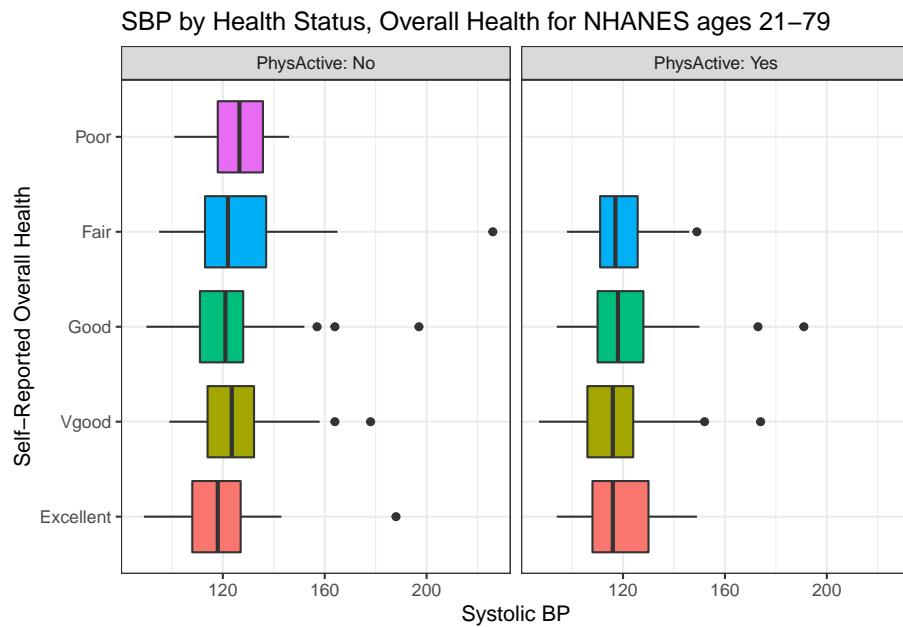
`summarise()` ungrouping output (override with `^.groups` argument)
```

HealthGen	count	mean(SBP)	median(SBP)
Excellent	64	119.1562	117.0
Vgood	196	119.0714	118.0
Good	238	120.4244	119.5
Fair	83	123.9398	119.0
Poor	22	125.8636	126.5

3.9.4 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 `labeler` within `facet_wrap` to include both the name of the (Physical Activity) variable and its value.

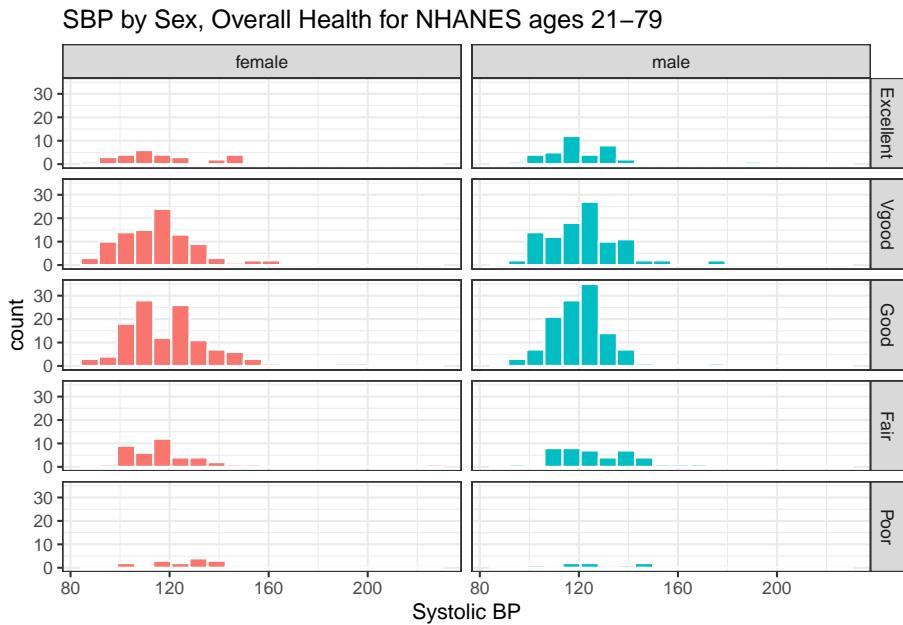
```
ggplot(data = nh_dat3, aes(x = HealthGen, y = SBP, fill = HealthGen)) +
  geom_boxplot() +
  labs(title = "SBP by Health Status, Overall Health for NHANES ages 21-79",
       y = "Systolic BP", x = "Self-Reported Overall Health") +
  guides(fill = FALSE) +
  facet_wrap(~ PhysActive, labeller = "label_both") +
  coord_flip()
```



3.9.5 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 Sex.

```
ggplot(data = nh_dat3, aes(x = SBP, fill = Sex)) +
  geom_histogram(color = "white", bins = 20) +
  labs(title = "SBP by Sex, Overall Health for NHANES ages 21-79",
       x = "Systolic BP") +
  guides(fill = FALSE) +
  facet_grid(HealthGen ~ Sex)
```



3.10 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.

Chapter 4

Data Structures and Types of Variables

4.1 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 et al. (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.

4.2 A New NHANES Adult Sample

In Chapter 3, we spent some time with a sample from the National Health and Nutrition Examination. Now, by changing the value of the `set.seed` function which determines the starting place for the random sampling, and changing some other specifications, we'll generate a new sample describing 500 adult subjects who completed the 2011-12 version of the survey when they were between the ages of 21 and 64.

Note also that what is listed in the NHANES data frame 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. I'm also changing the names of three other variables, to create `Race`, `SBP` and `DBP`.

```
# library(NHANES) # NHANES package/library of functions, data

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

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

nh_adults <- sample_n(nh_temp, size = 500)

nh_adults

# A tibble: 500 x 14
   ID Sex   Age Race Education   BMI   SBP   DBP Pulse PhysActive Smoke100
   <int> <fct> <int> <fct> <fct>   <dbl> <int> <int> <int> <fct>    <fct>
 1 71531 male    35 White Some Col~  22.4   143    90    84 Yes      No
 2 68613 fema~   61 White Some Col~  27.7   119    86   112 No      No
 3 67064 male    31 White College ~  26.6   110    76    86 Yes      Yes
 4 63924 fema~   29 Black High Sch~  41.9   98     56    74 No      Yes
 5 62840 male    60 White 8th Grade  35.8   127     0   110 No      Yes
 6 68058 male    50 White Some Col~  30.6    NA     NA    NA No      Yes
 7 68936 fema~   36 Black High Sch~  30.5   119    69    60 No      No
 8 71189 male    51 White College ~  25.6   112    70    54 Yes      Yes
 9 69936 fema~   54 Asian College ~  21.8   126    80    78 Yes      No
10 70687 male    59 White College ~  25.5   149    89    62 Yes      No
# ... with 490 more rows, and 3 more variables: SleepTrouble <fct>,
#   MaritalStatus <fct>, HealthGen <fct>
```

The data consist of 500 rows (observations) on 13 variables (columns). Essentially, we have 13 pieces of information on each of 500 adult NHANES subjects who were included in the 2011-12 panel.

4.2.1 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_adults)
```

```
[1] 500 14
```

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

```
str(nh_adults)
```

```
tibble [500 x 14] (S3: tbl_df/tbl/data.frame)
$ ID      : int [1:500] 71531 68613 67064 63924 62840 ...
$ Sex     : Factor w/ 2 levels "female","male": 2 1 2 1 2 2 1 2 1 2 ...
$ Age     : int [1:500] 35 61 31 29 60 50 36 51 54 59 ...
$ Race    : Factor w/ 6 levels "Asian","Black",...: 5 5 5 2 5 5 2 5 1 5 ...
$ Education : Factor w/ 5 levels "8th Grade","9 - 11th Grade",...: 4 4 5 3 1 4 3 5 5 5 ...
$ BMI     : num [1:500] 22.4 27.7 26.6 41.9 35.8 30.6 30.5 25.6 21.8 25.5 ...
$ SBP     : int [1:500] 143 119 110 98 127 NA 119 112 126 149 ...
$ DBP     : int [1:500] 90 86 76 56 0 NA 69 70 80 89 ...
$ Pulse   : int [1:500] 84 112 86 74 110 NA 60 54 78 62 ...
$ PhysActive : Factor w/ 2 levels "No","Yes": 2 1 2 1 1 1 1 2 2 2 ...
$ Smoke100 : Factor w/ 2 levels "No","Yes": 1 1 2 2 2 2 1 2 1 1 ...
$ SleepTrouble : Factor w/ 2 levels "No","Yes": 2 1 1 2 2 2 1 1 1 1 ...
$ MaritalStatus: Factor w/ 6 levels "Divorced","LivePartner",...: 4 6 3 5 3 3 4 3 3 6 ...
$ HealthGen  : Factor w/ 5 levels "Excellent","Vgood",...: 3 2 3 4 5 3 3 NA 3 1 ...
```

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

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

```
# A tibble: 5 x 14
  ID Sex   Age Race Education BMI  SBP  DBP Pulse PhysActive Smoke100
  <int> <fct> <int> <fct> <fct> <dbl> <int> <int> <int> <fct>   <fct>
1 66770  fema~  22  White Some Col~ 44.6   100    90   92 Yes      No
2 68754  male   57  White Some Col~ 23.2   124    85   82 No       Yes
3 70911  male   59  White College ~ 24.5   118    57   76 No       Yes
4 71393  male   27  White High Sch~ 25.7   116    61   88 Yes      No
5 70458  fema~  35  Black 9 - 11th~ 21.9   115    64   84 No       No
# ... with 3 more variables: SleepTrouble <fct>, MaritalStatus <fct>,
#   HealthGen <fct>
```

4.2.2 What are the variables?

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

```
glimpse(nh_adults)
```

```
Rows: 500
Columns: 14
$ ID      <int> 71531, 68613, 67064, 63924, 62840, 68058, 68936, 7118...
$ Sex     <fct> male, female, male, female, male, male, female, male, ...
$ Age     <int> 35, 61, 31, 29, 60, 50, 36, 51, 54, 59, 59, 27, 44, 4...
$ Race    <fct> White, White, White, Black, White, White, Black, Whit...
$ Education <fct> Some College, Some College, College Grad, High School...
$ BMI     <dbl> 22.4, 27.7, 26.6, 41.9, 35.8, 30.6, 30.5, 25.6, 21.8, ...
$ SBP     <int> 143, 119, 110, 98, 127, NA, 119, 112, 126, 149, 122, ...
$ DBP     <int> 90, 86, 76, 56, 0, NA, 69, 70, 80, 89, 75, 78, 69, 78...
$ Pulse   <int> 84, 112, 86, 74, 110, NA, 60, 54, 78, 62, 82, 68, 76, ...
$ PhysActive <fct> Yes, No, Yes, No, No, No, Yes, Yes, Yes, Yes, ...
$ Smoke100 <fct> No, No, Yes, Yes, Yes, Yes, No, Yes, No, No, No, ...
$ SleepTrouble <fct> Yes, No, No, Yes, Yes, Yes, No, No, No, No, ...
$ MaritalStatus <fct> NeverMarried, Widowed, Married, Separated, Married, ...
$ HealthGen <fct> Good, Vgood, Good, Fair, Poor, Good, Good, NA, Good, ...
```

The variables we have collected are described in the brief table below¹.

Variable	Description	Sample Values
ID	a numerical code identifying the subject	64427, 63788
Sex	sex of subject (2 levels)	male, female
Age	age (years) at screening of subject	37, 40
Race	reported race of subject (6 levels)	White, Asian
Education	educational level of subject (5 levels)	College Grad, High School
BMI	body-mass index, in kg/m ²	36.5, 18.2
SBP	systolic blood pressure in mm Hg	111, 115
DBP	diastolic blood pressure in mm Hg	72, 74
Pulse	60 second pulse rate in beats per minute	56, 102
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 lev.)	Vgood, Good

¹Descriptions 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.

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.

```
summary(nh_adults)
```

ID	Sex	Age	Race	
Min. :62199	female:221	Min. :21.00	Asian : 42	
1st Qu.:64522	male :279	1st Qu.:31.00	Black : 63	
Median :67192		Median :42.00	Hispanic: 26	
Mean :67122		Mean :41.91	Mexican : 38	
3rd Qu.:69654		3rd Qu.:53.00	White : 313	
Max. :71911		Max. :64.00	Other : 18	
Education	BMI	SBP	DBP	
8th Grade : 24	Min. :17.30	Min. : 84.0	Min. : 0.00	
9 - 11th Grade: 60	1st Qu.:23.80	1st Qu.:110.0	1st Qu.: 66.00	
High School : 81	Median :27.50	Median :118.0	Median : 72.00	
Some College :153	Mean :28.48	Mean :119.2	Mean : 72.13	
College Grad :182	3rd Qu.:31.60	3rd Qu.:127.0	3rd Qu.: 78.00	
	Max. :63.30	Max. :209.0	Max. :103.00	
	NA's :5	NA's :15	NA's :15	
Pulse	PhysActive	Smoke100	SleepTrouble	MaritalStatus
Min. : 40.00	No :215	No :297	No :380	Divorced : 51
1st Qu.: 64.00	Yes:285	Yes:203	Yes:120	LivePartner : 51
Median : 72.00				Married : 259
Mean : 73.41				NeverMarried:112
3rd Qu.: 82.00				Separated : 16
Max. :112.00				Widowed : 11
NA's :15				
HealthGen				
Excellent: 50				
Vgood :154				
Good :184				
Fair : 49				
Poor : 14				
NA's : 49				

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 and some (quantitative/numeric) variables are summarized with a minimum, quartiles and mean.

4.3 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 promised a salary of 80,000 a year if you don't know whether it will be paid in Euros, dollars, yen or Estonian kroon.
- You might be surprised to see someone whose age is 72 listed in a database on childhood diseases until you find out that age is measured in months.
- Often just seeking the units can reveal a variable whose definition is challenging - just how do we measure "friendliness", or "success," for example.
- Quantitative variables may also be classified by whether they are **continuous** or can only take on a **discrete** set of values. Continuous data may take on any value, within a defined range. Suppose we are measuring height. While height is really continuous, our measuring stick usually only lets us measure with a certain degree of precision. If our measurements are only trustworthy to the nearest centimeter with the ruler we have, we might describe them as discrete measures. But we could always get a more precise ruler. The measurement divisions we make in moving from a continuous concept to a discrete measurement are usually fairly arbitrary. Another way to think of this, if you enjoy music, is that, as suggested in Norman and Streiner (2014), a piano is a *discrete* instrument, but a violin is a *continuous* one, enabling finer distinctions between notes than the piano is capable of making. Sometimes the distinction between continuous and discrete is important, but usually, it's not.
 - The `nh_adults` data includes several quantitative variables, specifically `Age`, `BMI`, `SBP`, `DBP` and `Pulse`.
 - We know these are quantitative because they have units: `Age` in years, `BMI` in kg/m^2 , the `BP` measurements in mm Hg, and `Pulse` in beats per minute.
 - Depending on the context, we would likely treat most of these as *discrete* given that the 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 an 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 moveable, 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. 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.
 - Each of the quantitative variables in our `nh_adults` data can be thought of as ratio variables.
- 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.

4.3.1 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 follows: 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

4.4 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.

- 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_adults` 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_adults` 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 ordinal, or nominal.
 - Binary variables in the `nh_adults` 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.

Chapter 5

Summarizing Quantitative Variables

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.

5.1 The **summary** function for Quantitative data

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

```
nh_adults %>%
  select(Age, BMI, SBP, DBP, Pulse) %>%
  summary()
```

Age	BMI	SBP	DBP
Min. :21.00	Min. :17.30	Min. : 84.0	Min. : 0.00
1st Qu.:31.00	1st Qu.:23.80	1st Qu.:110.0	1st Qu.: 66.00
Median :42.00	Median :27.50	Median :118.0	Median : 72.00
Mean :41.91	Mean :28.48	Mean :119.2	Mean : 72.13
3rd Qu.:53.00	3rd Qu.:31.60	3rd Qu.:127.0	3rd Qu.: 78.00
Max. :64.00	Max. :63.30	Max. :209.0	Max. :103.00
	NA's :5	NA's :15	NA's :15
Pulse			
Min. : 40.00			

```

1st Qu.: 64.00
Median : 72.00
Mean   : 73.41
3rd Qu.: 82.00
Max.    :112.00
NA's    :15

```

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 31 or younger.
- **Median** = the **median** (50th percentile) - half of the subjects were Age 42 or younger.
- **Mean** = the **mean**, usually what one means by an *average* - the sum of the Ages divided by 500 is 41.9,
- **3rd Qu.** = the **third quartile** (75th percentile) - 25% of the subjects were Age 53 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.

5.2 Measuring the Center of a Distribution

5.2.1 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_adults %>%
  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   500   41.9    42  20953

```

¹The quantiles (sometimes referred to as percentiles) can also be summarised with a box-plot.

And again, the Mean is just the Sum (20953), divided by the number of non-missing values of Age (500), or 41.906.

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_adults %>% select(Age) %>%
  arrange(Age)
```

```
# A tibble: 500 x 1
  Age
  <int>
1 21
2 21
3 21
4 21
5 21
6 21
7 21
8 21
9 22
10 22
# ... with 490 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_adults)` to get a spreadsheet-style presentation, or use the `sort` command...

```
sort(nh_adults$Age)
```

```
[1] 21 21 21 21 21 21 21 22 22 22 22 22 22 22 22 22 22 22 22 22 23 23 23 23 23
[26] 23 23 23 23 23 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24 25 25 25 25
[51] 25 25 25 25 25 26 26 26 26 26 26 26 26 26 26 26 26 26 27 27 27 27 27 27 27
[76] 27 27 27 27 27 27 27 28 28 28 28 28 28 28 28 28 28 28 28 28 29 29 29 29
[101] 29 29 29 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 31 31 31 31 31 31 31
[126] 31 31 31 31 31 32 32 32 32 32 32 32 32 32 32 32 33 33 33 33 33 33 33
[151] 33 33 33 33 33 33 33 34 34 34 34 34 34 34 35 35 35 35 35 35 35 36 36
[176] 36 36 36 36 36 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 37 38 38
[201] 38 38 38 38 38 38 38 38 39 39 39 39 39 39 39 39 39 39 39 39 39 40 40
[226] 40 40 40 40 40 40 40 41 41 41 41 41 41 41 41 41 41 41 41 42 42 42 42
[251] 42 42 42 42 42 42 43 43 43 43 43 43 43 43 43 43 43 43 43 43 44 44
[276] 44 44 44 44 44 44 44 44 45 45 45 45 45 45 45 45 45 45 45 45 45 45 46
[301] 46 46 46 46 46 46 47 47 47 47 47 47 47 47 47 48 48 48 48 48 49 49 49
[326] 49 49 49 49 49 49 49 49 49 49 49 49 49 49 50 50 50 50 50 50 50 50 50 50
[351] 50 50 50 50 51 51 51 51 51 51 51 52 52 52 52 52 52 53 53 53
```

```
[376] 53 53 53 53 53 53 53 53 53 53 54 54 54 54 54 54 54 54 54 54 54 54 54 54 54 54 55 55  

[401] 55 55 55 55 55 55 55 55 55 56 56 56 56 56 56 56 56 56 56 56 56 56 56 56 56 56 57 57  

[426] 57 57 57 57 57 57 57 57 57 58 58 58 58 58 58 58 58 58 59 59 59 59 59 59 59 59 59  

[451] 59 59 59 59 59 59 59 60 60 60 60 60 60 60 60 60 60 60 60 61 61 61 61 61 61 61  

[476] 61 61 61 61 62 62 62 62 63 63 63 63 63 63 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_adults$Age) [250:251]
```

```
[1] 42 42
```

5.2.2 Dealing with Missingness

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

```
nh_adults %>%  
  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 values².

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

```
# A tibble: 1 x 3  
  count `mean(Pulse)` `median(Pulse)`  
    <int>          <dbl>          <int>  
1    485           73.4           72
```

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

```
nh_adults %>%  
  summarise(mean(Pulse, na.rm=TRUE), median(Pulse, na.rm=TRUE))
```

```
# A tibble: 1 x 2  
  `mean(Pulse, na.rm = TRUE)` `median(Pulse, na.rm = TRUE)`  
    <dbl>                  <int>  
1        73.4                  72
```

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

While we eventually discuss the importance of **imputation** when dealing with missing data, this doesn't apply to providing descriptive summaries of actual, observed values.

5.2.3 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_adults %>%
  group_by(Age) %>%
  summarise(count = n()) %>%
  arrange(desc(count))

`summarise()` ungrouping output (override with `.`groups` argument)

# A tibble: 44 x 2
  Age   count
  <int> <int>
1 37     18
2 49     17
3 24     15
4 27     15
5 30     15
6 43     15
7 45     15
8 50     15
9 56     15
10 59    15
# ... with 34 more rows
```

Note the use of three different “verbs” in our function there - for more explanation of this strategy, visit Grolemund and Wickham (2019).

As an alternative, the **modeest** package's **mfv** function calculates the sample mode (or most frequent value)³.

5.3 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

³See the documentation for the **modeest** package's **mfv** function to look at other definitions of the mode.

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.

5.3.1 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_adults %>%
  select(Age) %>%
  range()
```

```
[1] 21 64
```

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

```
nh_adults %>%
  select(BMI) %>%
  range(., na.rm=TRUE)
```

```
[1] 17.3 63.3
```

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_adults %>%
  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          22             31             53
```

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

```
nh_adults %>%
  select(Age, BMI, SBP, DBP, Pulse) %>%
  summary()
```

Age	BMI	SBP	DBP
Min. :21.00	Min. :17.30	Min. : 84.0	Min. : 0.00
1st Qu.:31.00	1st Qu.:23.80	1st Qu.:110.0	1st Qu.: 66.00
Median :42.00	Median :27.50	Median :118.0	Median : 72.00
Mean :41.91	Mean :28.48	Mean :119.2	Mean : 72.13
3rd Qu.:53.00	3rd Qu.:31.60	3rd Qu.:127.0	3rd Qu.: 78.00

```

Max.    :64.00   Max.    :63.30   Max.    :209.0   Max.    :103.00
      NA's    :5        NA's    :15        NA's    :15
Pulse
Min.    : 40.00
1st Qu.: 64.00
Median  : 72.00
Mean    : 73.41
3rd Qu.: 82.00
Max.    :112.00
      NA's    :15

```

5.3.2 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 summarise 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.

5.3.3 Obtaining the Variance and Standard Deviation in R

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

```

nh_adults %>%
  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>
1 152.  39.7  233.  123.  144.

```

And here are the standard deviations of those same variables.

```

nh_adults %>%
  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>
1 152.  39.7  233.  123.  144.

```

Age	BMI	SBP	DBP	Pulse
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	12.3	6.30	15.3	11.1
				12.0

5.3.4 Defining the Variance and Standard Deviation

Bock et al. (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**.

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

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.

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

5.3.5 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 $\pm 2(\text{Standard Deviation})$ contains approximately 95% of the measurements;

⁴When 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.

- Mean ± 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.

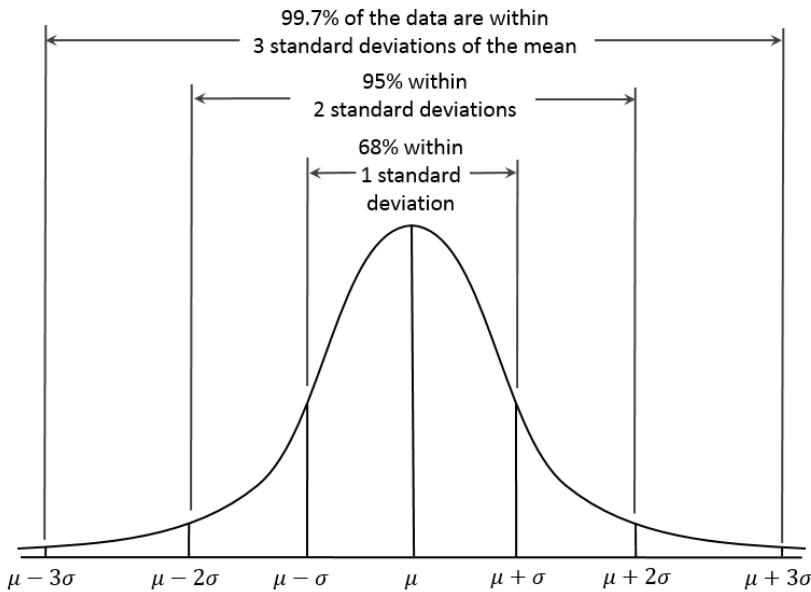


Figure 5.1: The Normal Distribution and the Empirical Rule

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

5.3.6 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.

5.4 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 summarise with a non-parametric measure of **skewness**.
- 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).

5.4.1 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.

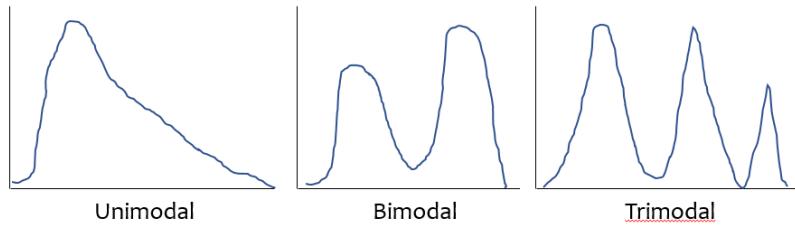


Figure 5.2: Unimodal and Multimodal Sketches

Truly multimodal distributions are usually described that way in terms of shape. For unimodal distributions, skewness and kurtosis become useful ideas.

5.4.2 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*.

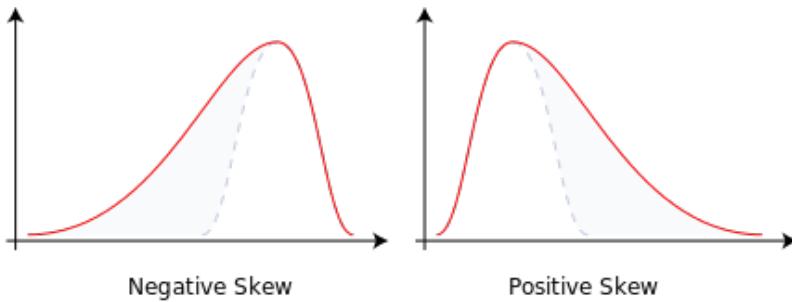


Figure 5.3: Negative (Left) Skew and Positive (Right) Skew

5.4.3 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.

```

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)

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) +

```

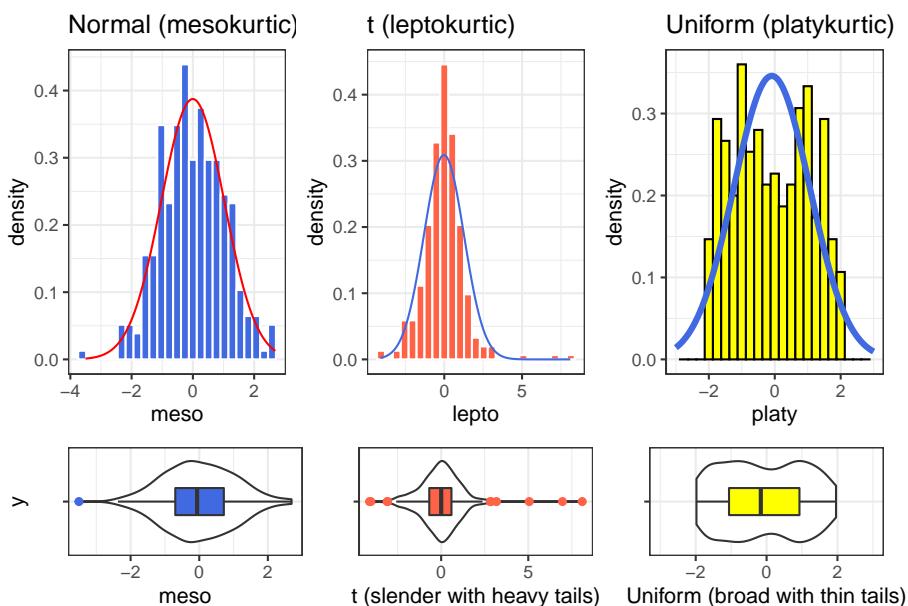
```

xlim(-3, 3) +
  labs(y = "", x = "Uniform (broad with thin tails)")

(p1 + p2 + p3) / (p1a + p2a + p3a) +
  plot_layout(heights = c(3, 1))

```

Warning: Removed 2 rows containing missing values (geom_bar).



Graphical tools are in most cases the best way to identify issues related to kurtosis.

5.5 More Detailed Numerical Summaries for Quantitative Variables

5.5.1 favstats in 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_adults)
```

min	Q1	median	Q3	max	mean	sd	n	missing
-----	----	--------	----	-----	------	----	---	---------

```
84 110      118 127 209 119.2495 15.25735 485      15
```

We could, of course, duplicate these results with a rather lengthy set of `summarise` pieces...

```
nh_adults %>%
  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(), missing = sum(is.na(SBP)))

# A tibble: 1 x 9
  min    Q1 median    Q3   max   mean    sd     n missing
  <int> <dbl> <int> <dbl> <int> <dbl> <dbl> <int>
1 84     110    118   127   209  119.   15.3   485      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_adults)
```

	Education	min	Q1	median	Q3	max	mean	sd	n	missing
1	8th Grade	95	114	122	131.50	167	123.7273	18.86085	22	2
2	9 - 11th Grade	92	108	114	125.25	170	117.3833	13.66189	60	0
3	High School	91	112	119	129.00	209	122.6104	19.68111	77	4
4	Some College	85	110	119	128.00	165	119.1812	13.52778	149	4
5	College Grad	84	109	118	126.00	171	117.9209	14.26831	177	5

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

```
nh_adults %>%
  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(), missing = sum(is.na(SBP)))

`summarise()` ungrouping output (override with `~.groups` argument)

# A tibble: 5 x 10
  Education      min    Q1 median    Q3   max   mean    sd     n missing
  <fct>        <int> <dbl> <dbl> <dbl> <int> <dbl> <dbl> <int>
1 8th Grade     95    114    122   132.   167   124.   18.9   22      0
2 9 - 11th Grade 92    108    114   125.   170   117.   13.7   60      0
3 High School   91    112    119   129    209   123.   19.7   77      0
4 Some College  85    110    119   128    165   119.   13.5   149     0
5 College Grad  84    109    118   126    171   118.   14.3   177     0
```

5.5.2 `describe` in the `psych` package

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

```
psych::describe(nh_adults %>% select(Age, BMI, SBP, DBP, Pulse))
```

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew
Age	1	500	41.91	12.35	42.0	41.86	16.31	21.0	64.0	43	0.03
BMI	2	495	28.48	6.30	27.5	27.80	5.63	17.3	63.3	46	1.35
SBP	3	485	119.25	15.26	118.0	118.25	13.34	84.0	209.0	125	1.27
DBP	4	485	72.13	11.10	72.0	72.33	8.90	0.0	103.0	103	-0.58
Pulse	5	485	73.41	12.01	72.0	73.01	11.86	40.0	112.0	72	0.30
					kurtosis	se					
Age			-1.20	0.55							
BMI			3.32	0.28							
SBP			4.63	0.69							
DBP			3.58	0.50							
Pulse			0.15	0.55							

The additional statistics presented here 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.)
- **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, \dots, Y_n , then the **mad** is defined as $\text{median}(|Y_i - \text{median}(Y_i)|)$.
 - 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.
- a measure of **skew**, which 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. The [Wikipedia page on skewness][<https://en.wikipedia.org/wiki/Skewness>] is very detailed.
- a measure of excess **kurtosis**, which refers to how outlier-prone, or heavy-tailed the shape of the distribution is, as compared to a Normal distribution.
- **se** = the standard error of the sample mean, equal to the sample sd divided by the square root of the sample size.

5.5.3 The Hmisc package's version of `describe`

```
Hmisc::describe(nh_adults %>% select(Age, BMI, SBP, DBP, Pulse))
```

```
nh_adults %>% select(Age, BMI, SBP, DBP, Pulse)
```

```
5 Variables      500 Observations
```

Age

	n	missing	distinct	Info	Mean	Gmd	.05	.10
	500	0	44	0.999	41.91	14.27	23	25
.	.25	.50	.75	.90	.95			
	31	42	53	59	61			

```
lowest : 21 22 23 24 25, highest: 60 61 62 63 64
```

BMI

	n	missing	distinct	Info	Mean	Gmd	.05	.10
	495	5	198	1	28.48	6.704	20.70	21.90
.	.25	.50	.75	.90	.95			
	23.80	27.50	31.60	35.68	41.00			

```
lowest : 17.3 17.8 18.2 18.3 18.4, highest: 47.7 54.1 54.4 56.8 63.3
```

SBP

	n	missing	distinct	Info	Mean	Gmd	.05	.10
	485	15	73	0.999	119.2	16.18	98	102
.	.25	.50	.75	.90	.95			
	110	118	127	137	143			

```
lowest : 84 85 91 92 93, highest: 170 171 182 202 209
```

DBP

	n	missing	distinct	Info	Mean	Gmd	.05	.10
	485	15	61	0.999	72.13	12.02	54.0	58.0
.	.25	.50	.75	.90	.95			
	66.0	72.0	78.0	85.6	89.0			

```
lowest : 0 41 42 44 45, highest: 98 99 100 102 103
```

Pulse

	n	missing	distinct	Info	Mean	Gmd	.05	.10
	485	15	35	0.997	73.41	13.47	54.4	60.0
.	.25	.50	.75	.90	.95			
	64.0	72.0	82.0	88.0	94.0			

```
lowest : 40 44 48 50 52, highest: 104 106 108 110 112
```

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.

5.5.4 Other options

The package `summarytools` has a function called `dfSummary` which I like and Dominic Comtois has also published Recommendations for Using `summarytools` with R Markdown. Note that this isn't really for Word documents.

The `naniar` package is helpful for wrangling and visualizing missing values, and checking imputations.

`DataExplorer` can be used for more automated exploratory data analyses (and some people also like `skimr`) and `visdat`, as well.

Chapter 6

Summarizing Categorical Variables

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_adults` data set we built in Section 4.2 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
Smoke100	Smoked 100+ cigarettes?	2	binary
SleepTrouble	Trouble sleeping?	2	binary
HealthGen	Self-report health	5	ordinal

6.1 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_adults %>%
  select(Sex, Race, Education, PhysActive, Smoke100,
         SleepTrouble, HealthGen, MaritalStatus) %>%
  summary()
```

	Sex	Race	Education	PhysActive	Smoke100
female:221	Asian : 42	8th Grade : 24	No :215	No :297	
male :279	Black : 63	9 - 11th Grade: 60	Yes:285	Yes:203	
	Hispanic: 26	High School : 81			
	Mexican : 38	Some College :153			
	White :313	College Grad :182			
	Other : 18				
	SleepTrouble	HealthGen		MaritalStatus	
No :380		Excellent: 50	Divorced : 51		
Yes:120		Vgood :154	LivePartner : 51		
		Good :184	Married :259		
		Fair : 49	NeverMarried:112		
		Poor : 14	Separated : 16		
		NA's : 49	Widowed : 11		

6.2 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_adults %>%
  tabyl(HealthGen) %>%
  adorn_pct_formatting()
```

HealthGen	n	percent	valid_percent
Excellent	50	10.0%	11.1%
Vgood	154	30.8%	34.1%
Good	184	36.8%	40.8%
Fair	49	9.8%	10.9%
Poor	14	2.8%	3.1%
<NA>	49	9.8%	-

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_adults %>%
  tabyl(HealthGen) %>%
  adorn_totals() %>%
  adorn_pct_formatting()
```

HealthGen	n	percent	valid_percent
Excellent	50	10.0%	11.1%
Vgood	154	30.8%	34.1%
Good	184	36.8%	40.8%

Fair	49	9.8%	10.9%
Poor	14	2.8%	3.1%
<NA>	49	9.8%	-
Total	500	100.0%	100.0%

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

```
nh_adults %>%
  tabyl(MaritalStatus) %>%
  adorn_totals() %>%
  adorn_pct_formatting()
```

MaritalStatus	n	percent
Divorced	51	10.2%
LivePartner	51	10.2%
Married	259	51.8%
NeverMarried	112	22.4%
Separated	16	3.2%
Widowed	11	2.2%
Total	500	100.0%

6.3 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_adults %>%
  group_by(HealthGen) %>%
  summarise(count = n()) %>%
  arrange(desc(count))

`summarise()` ungrouping output (override with `.`groups` argument)

# A tibble: 6 x 2
  HealthGen count
  <fct>     <int>
1 Good       184
2 Vgood      154
3 Excellent   50
4 Fair        49
5 <NA>        49
6 Poor        14
```

6.4 `describe` in the `Hmisc` package

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

nh_adults %>% select(Sex, Race, Education, PhysActive, Smoke100, SleepTrouble, HealthGen, MaritalStatus)

8 Variables      500 Observations
-----
Sex
  n   missing distinct
  500       0        2

  Value     female    male
  Frequency    221     279
  Proportion  0.442   0.558
-----
Race
  n   missing distinct
  500       0        6

  lowest : Asian     Black     Hispanic Mexican  White
  highest: Black     Hispanic Mexican  White     Other

  Value      Asian     Black Hispanic Mexican  White     Other
  Frequency    42      63      26      38      313      18
  Proportion  0.084   0.126   0.052   0.076   0.626   0.036
-----
Education
  n   missing distinct
  500       0        5

  lowest : 8th Grade    9 - 11th Grade High School  Some College  College Grad
  highest: 8th Grade    9 - 11th Grade High School  Some College  College Grad

  Value      8th Grade 9 - 11th Grade High School  Some College
  Frequency      24          60          81          153
  Proportion    0.048       0.120       0.162       0.306
-----
  Value      College Grad
  Frequency      182
  Proportion    0.364
```

PhysActive

n	missing	distinct
500	0	2

Value	No	Yes
-------	----	-----

Frequency	215	285
-----------	-----	-----

Proportion	0.43	0.57
------------	------	------

Smoke100

n	missing	distinct
500	0	2

Value	No	Yes
-------	----	-----

Frequency	297	203
-----------	-----	-----

Proportion	0.594	0.406
------------	-------	-------

SleepTrouble

n	missing	distinct
500	0	2

Value	No	Yes
-------	----	-----

Frequency	380	120
-----------	-----	-----

Proportion	0.76	0.24
------------	------	------

HealthGen

n	missing	distinct
451	49	5

lowest : Excellent	Vgood	Good	Fair	Poor
--------------------	-------	------	------	------

highest: Excellent	Vgood	Good	Fair	Poor
--------------------	-------	------	------	------

Value	Excellent	Vgood	Good	Fair	Poor
Frequency	50	154	184	49	14
Proportion	0.111	0.341	0.408	0.109	0.031

MaritalStatus

n	missing	distinct
500	0	6

lowest : Divorced	LivePartner	Married	NeverMarried	Separated
-------------------	-------------	---------	--------------	-----------

highest: LivePartner	Married	NeverMarried	Separated	Widowed
----------------------	---------	--------------	-----------	---------

Value	Divorced	LivePartner	Married	NeverMarried	Separated
Frequency	51	51	259	112	16
Proportion	0.102	0.102	0.518	0.224	0.032

Value	Widowed
Frequency	11
Proportion	0.022

6.5 Cross-Tabulations

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_adults %>%
  tabyl(Education, SleepTrouble) %>%
  adorn_totals(where = c("row", "col"))
```

	Education	No	Yes	Total
8th Grade	18	6	24	
9 - 11th Grade	45	15	60	
High School	62	19	81	
Some College	118	35	153	
College Grad	137	45	182	
	Total	380	120	500

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

	Education/SleepTrouble	No	Yes	Total
8th Grade	18	6	24	
9 - 11th Grade	45	15	60	
High School	62	19	81	
Some College	118	35	153	
College Grad	137	45	182	
	Total	380	120	500

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_adults %>%
  tabyl(Education, SleepTrouble) %>%
  adorn_totals(where = "row") %>%
```

```
adorn_percentages(denominator = "row") %>%
adorn_pct_formatting() %>%
adorn_title(placement = "combined")
```

Education/SleepTrouble	No	Yes
8th Grade	75.0%	25.0%
9 - 11th Grade	75.0%	25.0%
High School	76.5%	23.5%
Some College	77.1%	22.9%
College Grad	75.3%	24.7%
Total	76.0%	24.0%

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

```
nh_adults %>%
  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	4.7%	5.0%	4.8%
9 - 11th Grade	11.8%	12.5%	12.0%
High School	16.3%	15.8%	16.2%
Some College	31.1%	29.2%	30.6%
College Grad	36.1%	37.5%	36.4%

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:

```
nh_adults %>%
  tabyl(Education, SleepTrouble) %>%
  adorn_totals(where = c("row", "col")) %>%
  adorn_percentages(denominator = "all") %>%
  adorn_pct_formatting() %>%
  adorn_title(placement = "combined")
```

Education/SleepTrouble	No	Yes	Total
8th Grade	3.6%	1.2%	4.8%
9 - 11th Grade	9.0%	3.0%	12.0%
High School	12.4%	3.8%	16.2%
Some College	23.6%	7.0%	30.6%
College Grad	27.4%	9.0%	36.4%
Total	76.0%	24.0%	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_adults %>%
  tabyl(MaritalStatus, HealthGen) %>%
  adorn_totals(where = c("row")) %>%
  adorn_percentages(denominator = "row") %>%
  adorn_pct_formatting() %>%
  adorn_ns(position = "front") %>%
  adorn_title(placement = "combined") %>%
  knitr::kable()
```

MaritalStatus/HealthGen	Excellent	Vgood	Good	Fair	Poor	NA
Divorced	7 (13.7%)	14 (27.5%)	20 (39.2%)	5 (9.8%)	2 (3.9%)	3 (5.9%)
LivePartner	1 (2.0%)	18 (35.3%)	16 (31.4%)	11 (21.6%)	1 (2.0%)	4 (7.8%)
Married	23 (8.9%)	84 (32.4%)	102 (39.4%)	15 (5.8%)	4 (1.5%)	31 (11.1%)
NeverMarried	14 (12.5%)	31 (27.7%)	43 (38.4%)	13 (11.6%)	3 (2.7%)	8 (7.1%)
Separated	4 (25.0%)	4 (25.0%)	1 (6.2%)	4 (25.0%)	1 (6.2%)	2 (12.5%)
Widowed	1 (9.1%)	3 (27.3%)	2 (18.2%)	1 (9.1%)	3 (27.3%)	1 (9.1%)
Total	50 (10.0%)	154 (30.8%)	184 (36.8%)	49 (9.8%)	14 (2.8%)	49 (9.8%)

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

```
nh_adults %>%
  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")
```

MaritalStatus/HealthGen	Excellent	Vgood	Good	Fair
Divorced	7 (14.6%)	14 (29.2%)	20 (41.7%)	5 (10.4%)
LivePartner	1 (2.1%)	18 (38.3%)	16 (34.0%)	11 (23.4%)
Married	23 (10.1%)	84 (36.8%)	102 (44.7%)	15 (6.6%)
NeverMarried	14 (13.5%)	31 (29.8%)	43 (41.3%)	13 (12.5%)
Separated	4 (28.6%)	4 (28.6%)	1 (7.1%)	4 (28.6%)
Widowed	1 (10.0%)	3 (30.0%)	2 (20.0%)	1 (10.0%)
Total	50 (11.1%)	154 (34.1%)	184 (40.8%)	49 (10.9%)
Poor				
2 (4.2%)				
1 (2.1%)				
4 (1.8%)				
3 (2.9%)				
1 (7.1%)				
3 (30.0%)				
14 (3.1%)				

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

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

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

Race	HealthGen				
	Excellent	Vgood	Good	Fair	Poor
Asian	3	11	17	3	0
Black	8	11	19	11	6
Hispanic	3	3	11	4	1
Mexican	2	8	17	6	3
White	33	113	114	22	4
Other	1	8	6	3	0

6.5.1 Cross-Classifying Three Categorical Variables

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

```
nh_adults %>%
  tabyl(Smoke100, PhysActive, SleepTrouble) %>%
  adorn_title(placement = "top")
```

\$No		
PhysActive		
Smoke100	No	Yes
No	99	142
Yes	62	77

\$Yes		
PhysActive		
Smoke100	No	Yes
No	21	35
Yes	33	31

The result here is a tabyl of `Smoke100` (rows) by `PhysActive` (columns), split into a list by `SleepTrouble`. Another approach to get the same table is:

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

```
, , SleepTrouble = No
```

PhysActive		
Smoke100	No	Yes
No	99	142
Yes	62	77

```
, , SleepTrouble = Yes

    PhysActive
Smoke100  No Yes
  No      21  35
  Yes     33  31
```

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

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

		Smoke100	
		No	Yes
PhysActive	SleepTrouble		
	No	99	62
Yes	Yes	21	33
	No	142	77
	Yes	35	31

And we can do this with **dplyr** functions, as well, for example...

```
nh_adults %>%
  select(Smoke100, PhysActive, SleepTrouble) %>%
  table()
```

```
, , SleepTrouble = No

    PhysActive
Smoke100  No Yes
  No      99 142
  Yes     62  77

, , SleepTrouble = Yes

    PhysActive
Smoke100  No Yes
  No      21  35
  Yes     33  31
```

6.6 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

6.6.1 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,	1,167
Arizona	10.7 (8.3	(8.3-13.6) 1,52	1,520
Arkansas	17.8 (15.7-	(15.7-20.1)	1,470
Connecticut	12.3	(10.2-14.7) 2,2	2,270
Delaware	14.2 (12	(12.9-15.6)	2,475
Florida	11.6 (10.5-1	(10.5-12.8)	5,491
...			
Wisconsin	11.6 ((9.7-13.9) 2,7	2,771
Wyoming	10.7	(9.4-12.2) 2,910	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
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.

6.6.2 Order rows and columns sensibly

- Alabama First!
 - Size places - put the largest first. We often look most carefully at the top.
- Order time from the past to the future to help the viewer.
- If there is a clear predictor-outcome relationship, put the predictors in the rows and the outcomes in the columns.

6.6.3 Round - a lot!

- Humans cannot understand more than two digits very easily.
- We almost never care about accuracy of more than two digits.
- We can almost never justify more than two digits of accuracy statistically.
- It's also helpful to remember that we are almost invariably publishing progress to date, rather than a truly final answer.

Suppose, for instance, we report a correlation coefficient of 0.25. How many observations do you think you would need to justify such a choice?

- To report 0.25 meaningfully, we want to be sure that the second digit isn't 4 or 6.
- That requires a standard error less than 0.005
- The *standard error* of any statistic is proportional to 1 over the square root of the sample size, n .

So $\frac{1}{\sqrt{n}} \sim 0.005$, but that means $\sqrt{n} = \frac{1}{0.005} = 200$. If $\sqrt{n} = 200$, then $n = (200)^2 = 40,000$.

Do we usually have 40,000 observations?

6.6.4 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.

6.7 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 is described in substantial detail at <https://gt.rstudio.com/> and we'll get started with it soon.

Chapter 7

NHANES National Youth Fitness Survey (nnyfs)

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://www.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.

7.1 The Variables included in nnyfs

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

7.1.1 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

7.1.2 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)
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)

7.1.3 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_s Skinfold	BMXCALFF	Calf skinfold (mm)
triceps_s Skinfold	BMXTRI	Triceps skinfold (mm)
subscapular_s 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

7.1.4 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
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

100CHAPTER 7. NHANES NATIONAL YOUTH FITNESS SURVEY (NNYFS)

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

7.2 Looking over A Few Variables

Now, I'll take a look at the `nnysf` data, which I've made available in a comma-separated version (`nnysf.csv`), if you prefer, as well as in an R data set (`nnysf.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.

```
nnysf <- readRDS("data/nnysf.Rds") %>% as_tibble()

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

[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 # this is a tibble, has some nice features in a print-out like this
```

```
# 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 Fema~    15 3_Black~     9 English    28299.  0.21
2 71918 Fema~    8 3_Black~     2 English    15127.   5
3 71919 Fema~    14 2_White~    8 English    29977.   5
4 71920 Fema~    15 2_White~    8 English    80652.   0.87
5 71921 Male     3 2_White~    NA English   55592.   4.34
6 71922 Male     12 1_Hispa~   6 English    27365.   5
7 71923 Male     12 2_White~   5 English    86673.   5
8 71924 Fema~    8 4_Other~    2 English    39549.   2.74
9 71925 Male     7 1_Hispa~   0 English    42333.   0.46
10 71926 Male    8 3_Black~   2 English    15307.   1.57
# ... with 1,508 more rows, and 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>, tv_hours <dbl>, computer_hours <dbl>,
#   physical_last_week <chr>, enjoy_recess <chr>, meals_out <dbl>,
#   insured <chr>, phys_health <chr>, access_to_care <chr>, care_source <chr>,
#   asthma Ever <chr>, asthma Now <chr>, med_use <chr>, med_count <dbl>,
#   insurance <chr>
```

Tibbles are a modern reimagining of the main way in which people have stored data in R, called a data frame. Tibbles were developed to keep what time has proven to be effective, and throwing out what is not. 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           : chr [1:1518] "Female" "Female" "Female" "Female" ...
$ age_child     : num [1:1518] 15 8 14 15 3 12 12 8 7 8 ...
$ race_eth      : chr [1:1518] "3_Black Non-Hispanic" "3_Black Non-Hispanic" "2_White Non-Hispanic"
$ educ_child    : num [1:1518] 9 2 8 8 NA 6 5 2 0 2 ...
$ language       : chr [1:1518] "English" "English" "English" "English" ...
$ 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 ...
```

102CHAPTER 7. NHANES NATIONAL YOUTH FITNESS SURVEY (NNYFS)

```

$ educ_adult      : chr [1:1518] "2_9-11th Grade" "3_High School Graduate" "5_College Gr...
$ respondent     : chr [1:1518] "Child" "Mom" "Child" "Child" ...
$ salt_used       : chr [1:1518] "Yes" "Yes" "Yes" "Yes" ...
$ 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   : chr [1:1518] "2_Usual" "2_Usual" "2_Usual" "2_Usual" ...
$ 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          : chr [1:1518] NA "4_Obese" "2_Normal" "4_Obese" ...
$ 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: chr [1:1518] "No" "No" "Yes" "Yes" ...
$ enjoy_recess      : chr [1:1518] "1_Strongly Agree" "1_Strongly Agree" "3_Neither Agree...
$ meals_out          : num [1:1518] 0 2 3 2 1 1 2 1 0 2 ...
$ insured            : chr [1:1518] "Has Insurance" "Has Insurance" "Has Insurance" "Has Insu...
$ phys_health        : chr [1:1518] "1_Excellent" "3_Good" "1_Excellent" "3_Good" ...
$ access_to_care     : chr [1:1518] "Has Usual Care Source" "Has Usual Care Source" "Has Us...
$ care_source         : chr [1:1518] "Clinic or Health Center" "Doctor's Office" "Doctor's O...
$ asthma_ever        : chr [1:1518] "Never Had Asthma" "History of Asthma" "Never Had Asthma...
$ asthma_now          : chr [1:1518] "No Asthma Now" "Asthma Now" "No Asthma Now" "Asthma Now...
$ med_use             : chr [1:1518] "No Medications" "Had Medication" "No Medications" "Had M...
$ med_count           : num [1:1518] 0 1 0 2 0 0 0 0 0 0 ...
$ insurance           : chr [1:1518] "State Sponsored" "State Sponsored" "Private" "State Spo...

```

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

7.2.1 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.

7.2.2 sex

The second variable, `sex`, is listed as a character 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*.

```
nyfs %>%
  tabyl(sex) %>%
  adorn_totals() %>%
  adorn_pct_formatting()
```

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

7.2.3 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.

```
nyfs %>% 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%

104 CHAPTER 7. NHANES NATIONAL YOUTH FITNESS SURVEY (NNYFS)

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.

```
mosaic::favstats(~ age_child, data = nnyfs) %>%  
  kable(digits = 1)
```

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

7.2.4 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() %>%  
  knitr::kable()
```

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	

7.2.5 income_pov

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

```
nnys %>% select(income_pov) %>% summary()
```

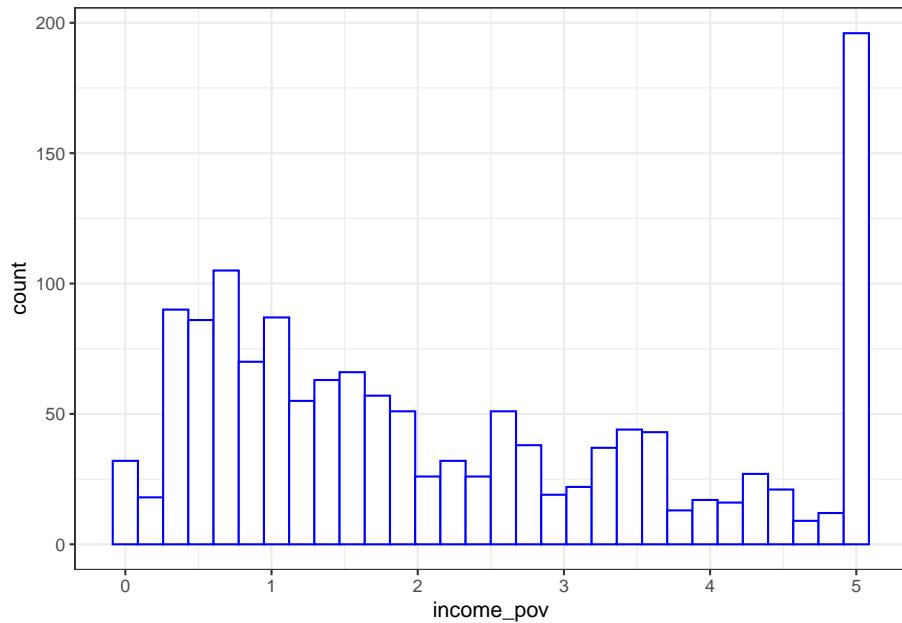
income_pov	
Min.	:0.000
1st Qu.	:0.870
Median	:1.740
Mean	:2.242
3rd Qu.	:3.520
Max.	:5.000
NA's	:89

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).

106 CHAPTER 7. NHANES NATIONAL YOUTH FITNESS SURVEY (NNYFS)



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?

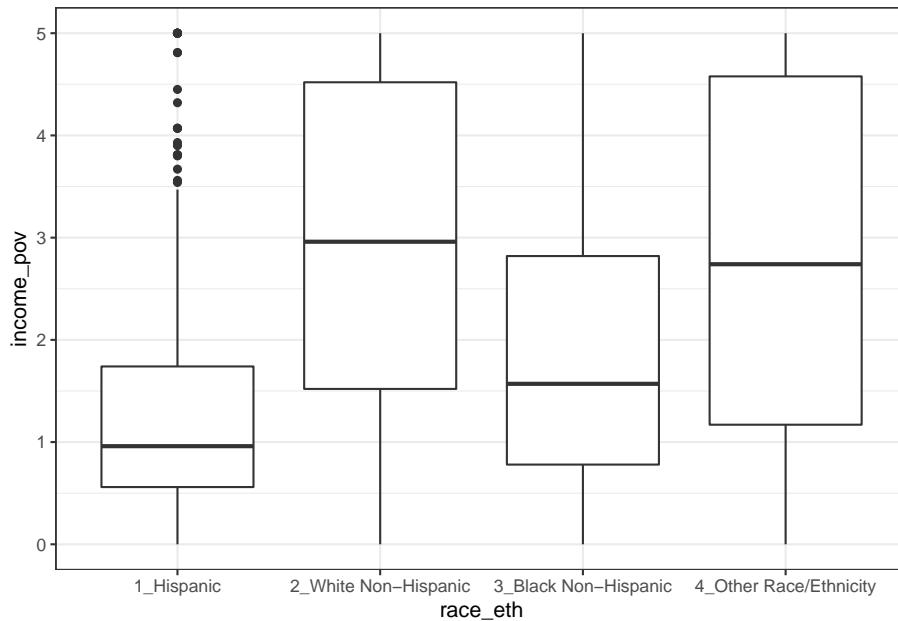
```
mosaic::favstats(income_pov ~ race_eth, data = nnyfs) %>%
  kable(digits = 1)
```

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

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).



7.2.6 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, $\text{BMI} = \text{weight in kg} / (\text{height in m})^2$. 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.

```
nyfs %>% select(bmi) %>% summary()
```

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

7.2.7 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.

```
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) %>%
  kable(digits = 1)
```

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

7.2.8 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.

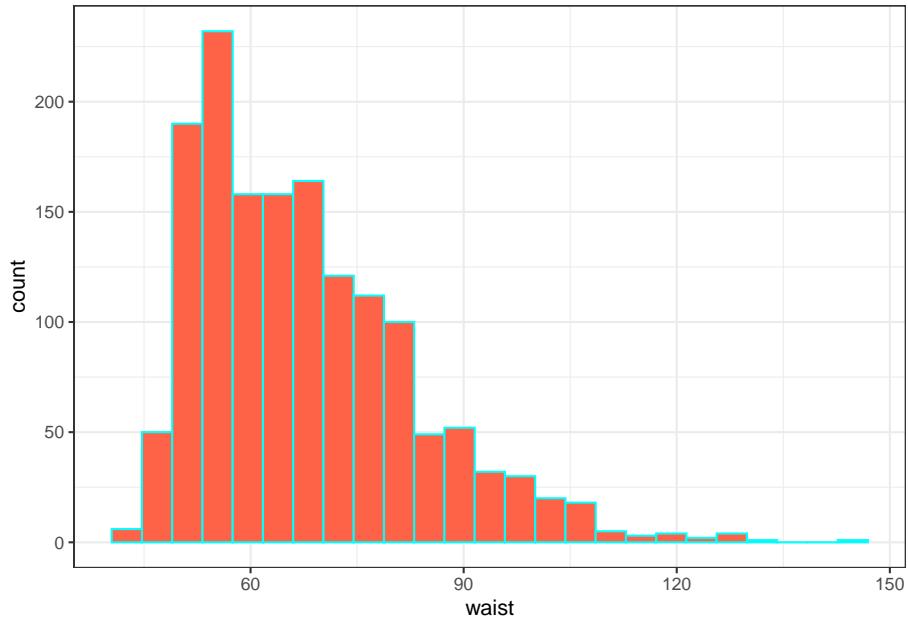
```
mosaic::favstats(~ waist, data = nnyfs)
```

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).



7.2.9 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.

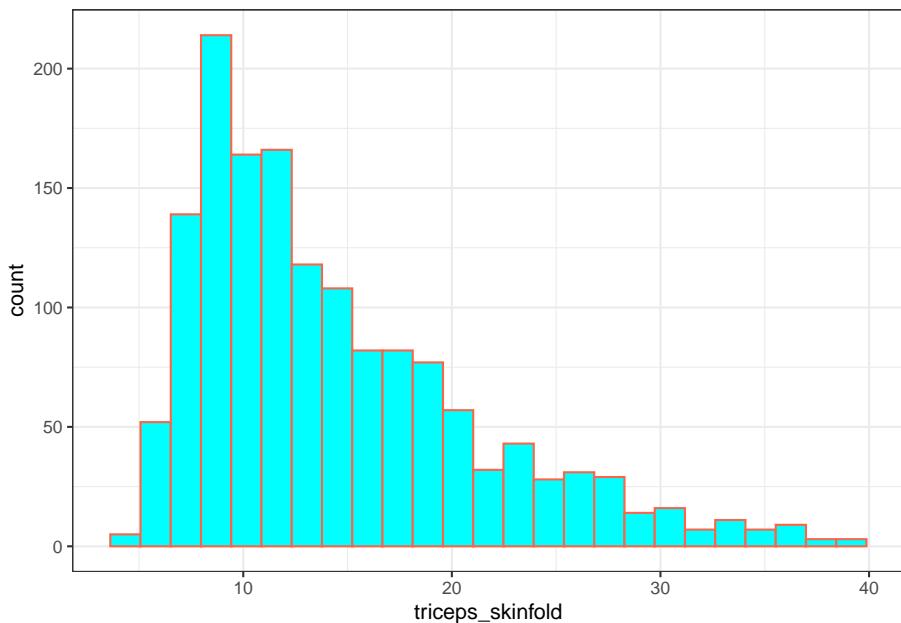
```
mosaic::favstats(~ triceps_skinfold, data = nnyfs)
```

	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.

7.3 Additional Numeric Summaries

7.3.1 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		

112CHAPTER 7. NHANES NATIONAL YOUTH FITNESS SURVEY (NNYFS)

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 5.3.1, 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
```

7.4 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)
```

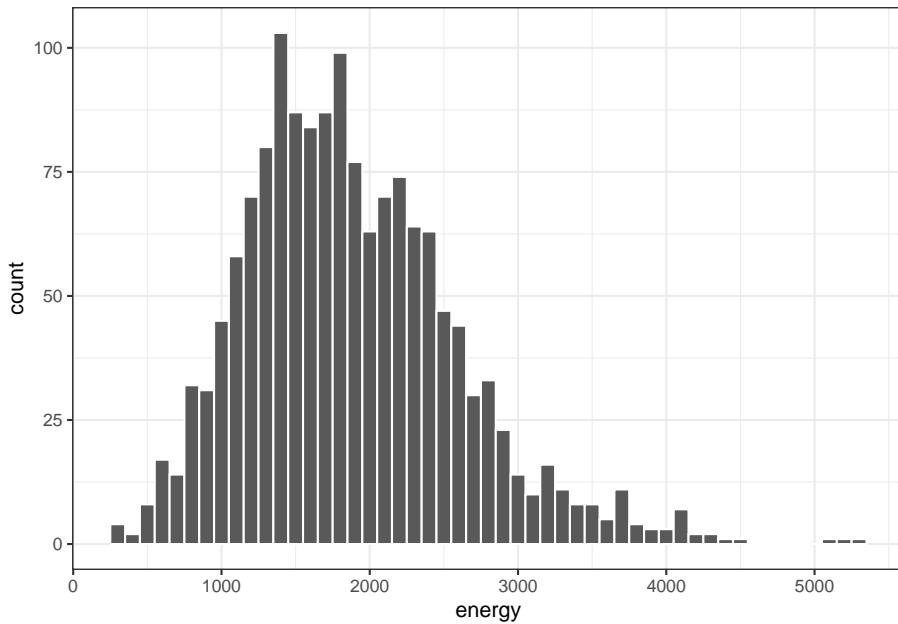
```
min      Q1 median      Q3 max      mean      sd      n missing
257 1367.5 1794.5 2306 5265 1877.157 722.3537 1518          0
```

This adds three useful results to the base summary - the standard deviation, the sample size and the number of missing observations.

7.5 The Histogram

As we saw in 3, 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")
```



7.5.1 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

$$h = \frac{2 * IQR}{n^{1/3}}$$

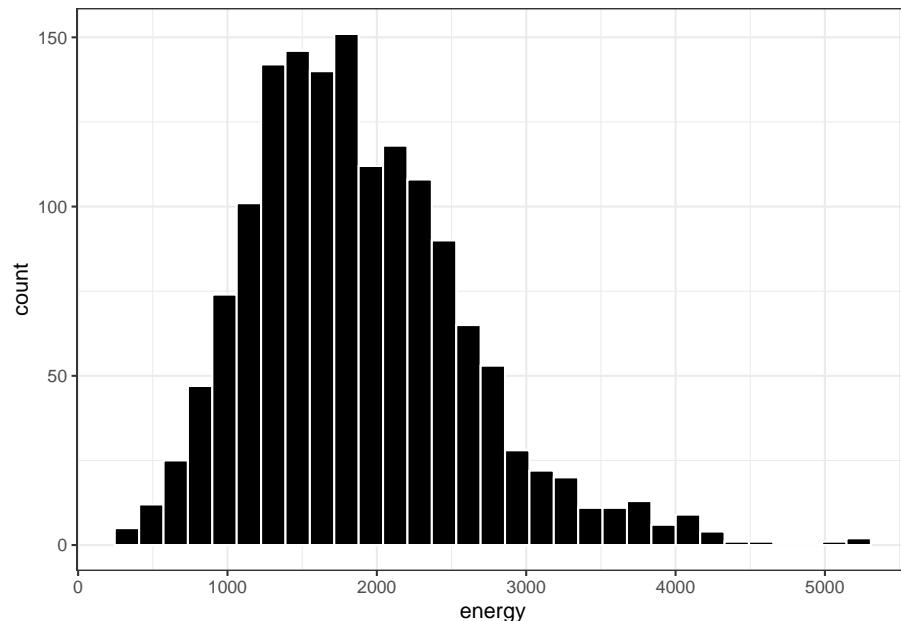
so that the number of bins is equal to the range of the data set (maximum - minimum) divided by h .

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.

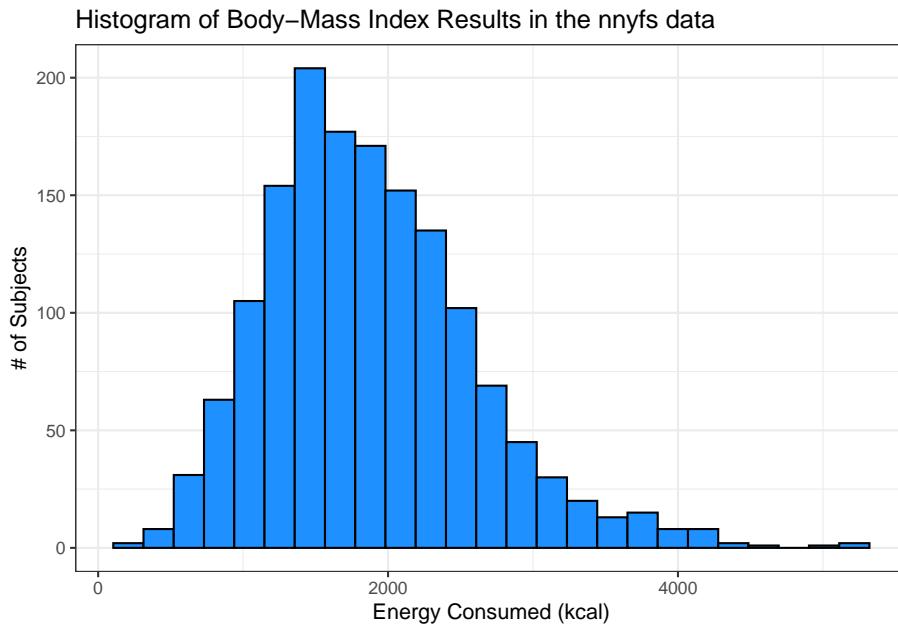
```
bw <- 2 * IQR(nnyfs$energy) / length(nnyfs$energy)^(1/3)
ggplot(data = nnyfs, aes(x = energy)) +
  geom_histogram(binwidth=bw, color = "white", fill = "black")
```



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")
```



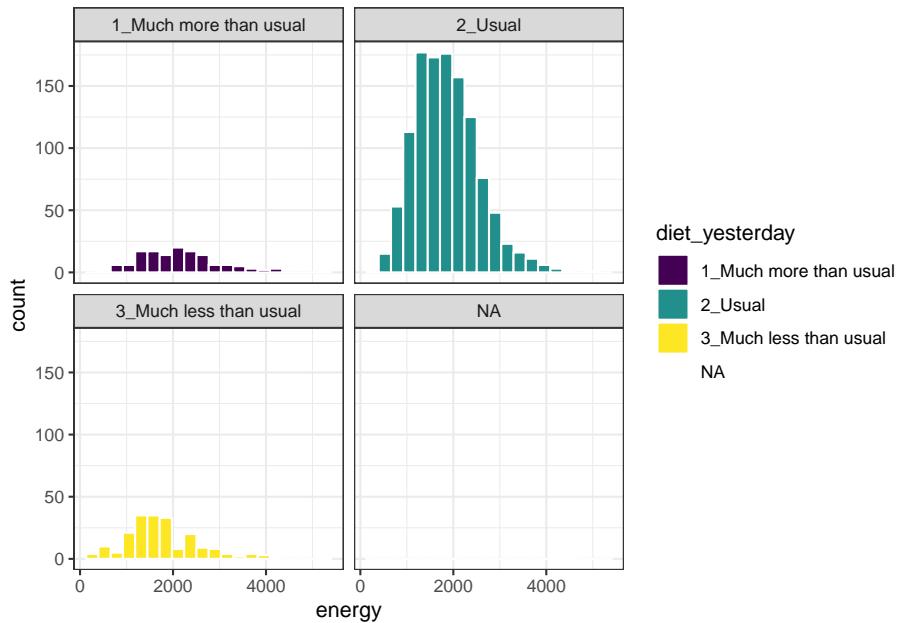
7.5.2 A Note on Colors

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`.

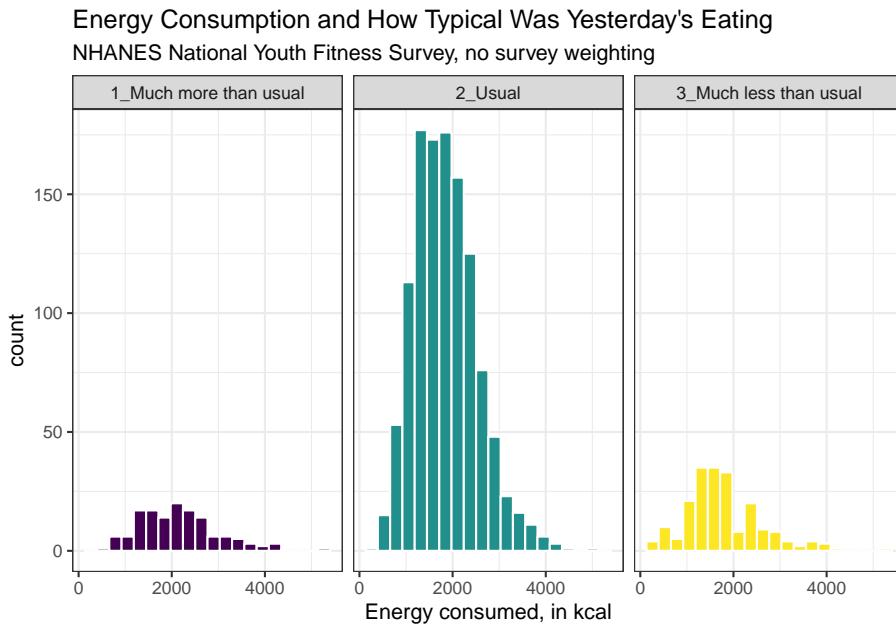
Here's a comparison of several histograms, looking at `energy` consumed as a function of whether yesterday was typical in terms of food consumption.

```
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 %>% filter(complete.cases(energy, diet_yesterday)) %>%
  ggplot(data = ., aes(x = energy, fill = diet_yesterday)) +
  geom_histogram(bins = 20, col = "white") +
  scale_fill_viridis_d() +
  guides(fill = FALSE) +
  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")
```

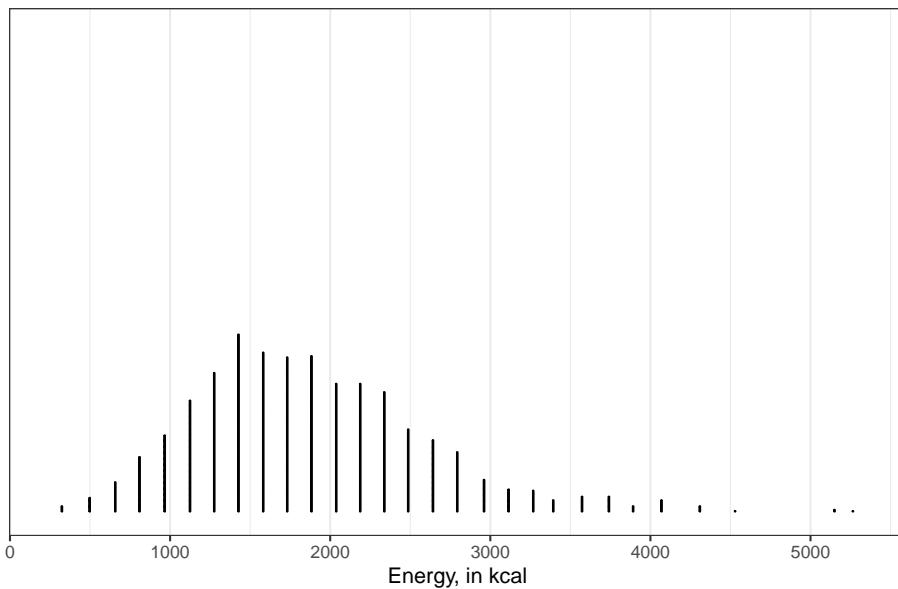


7.6 The Dot Plot to display a distribution

We can plot the distribution of a single continuous variable using the `dotplot` geom:

```
ggplot(data = nnyfs, aes(x = energy)) +
  geom_dotplot(dotsize = 0.05, binwidth=150) +
  scale_y_continuous(NULL, breaks = NULL) # hides y-axis since it is meaningless
  labs(title = "Dotplot of nnyfs Kilocalories consumed",
       x = "Energy, in kcal")
```

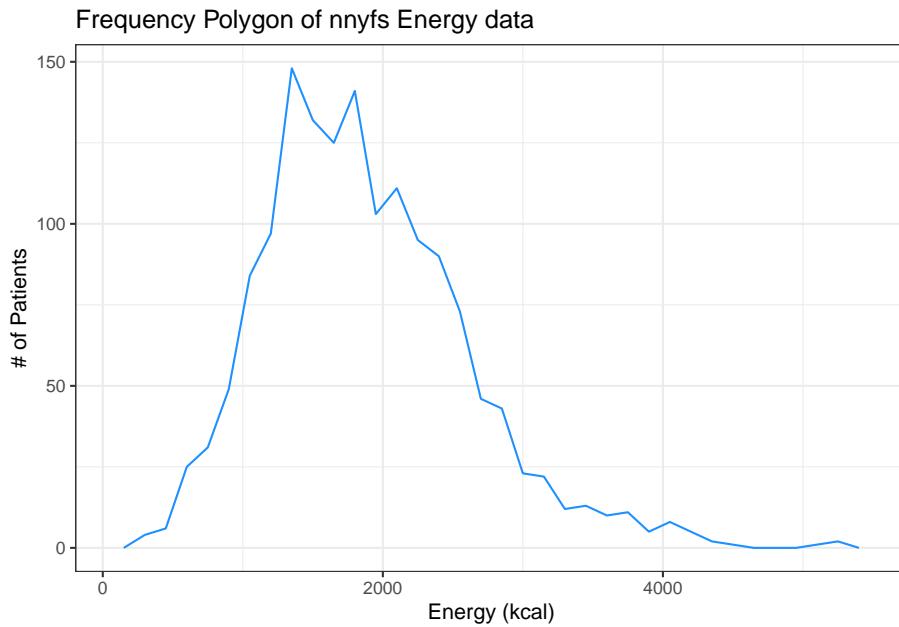
Dotplot of nnyfs Kilocalories consumed



7.7 The Frequency Polygon

We can plot the distribution of a single continuous variable using the `freqpoly` geom:

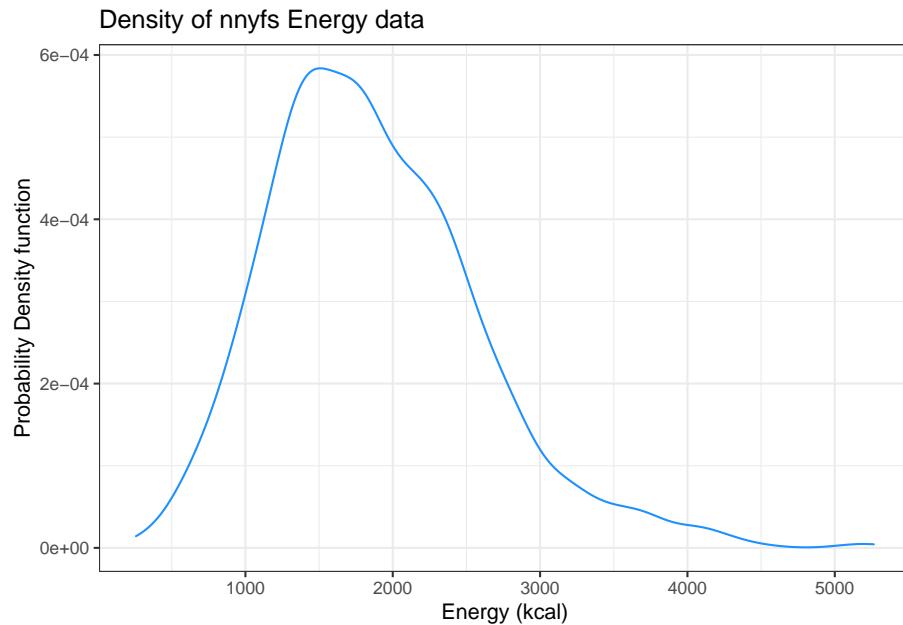
```
ggplot(data = nnyfs, aes(x = energy)) +
  geom_freqpoly(binwidth = 150, color = "dodgerblue") +
  labs(title = "Frequency Polygon of nnyfs Energy data",
       x = "Energy (kcal)", y = "# of Patients")
```



7.8 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")
```



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.

7.9 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.

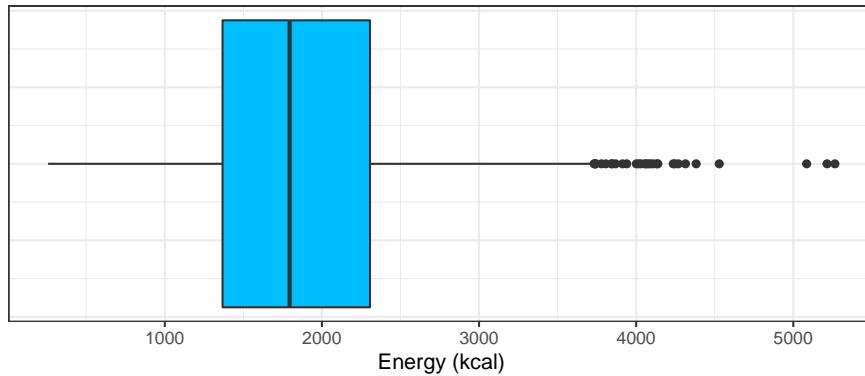
7.9.1 Drawing a Boxplot for One Variable in ggplot2

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



7.9.2 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 $IQR = 75\text{th percentile} - 25\text{th percentile}$, then the upper fence is $75\text{th percentile} + 1.5 \times IQR$, and the lower fence is $25\text{th percentile} - 1.5 \times IQR$.

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.

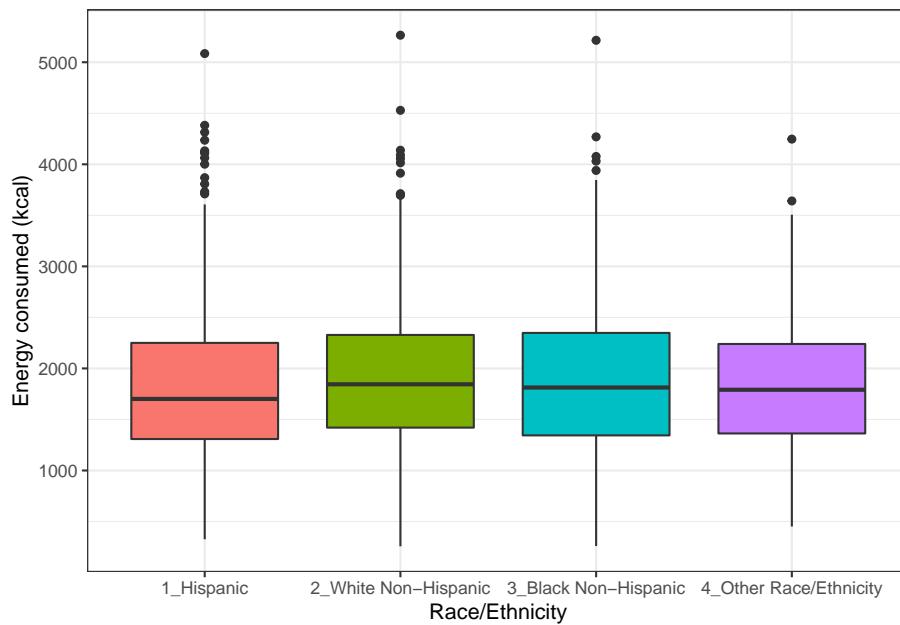
- This plot is indicating clearly that there is some asymmetry (skew) in the data, specifically right skew.
- The standard R uses is to indicate as outliers any points that are more than 1.5 inter-quartile ranges away from the edges of the box.

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.

7.10 A Simple Comparison Boxplot

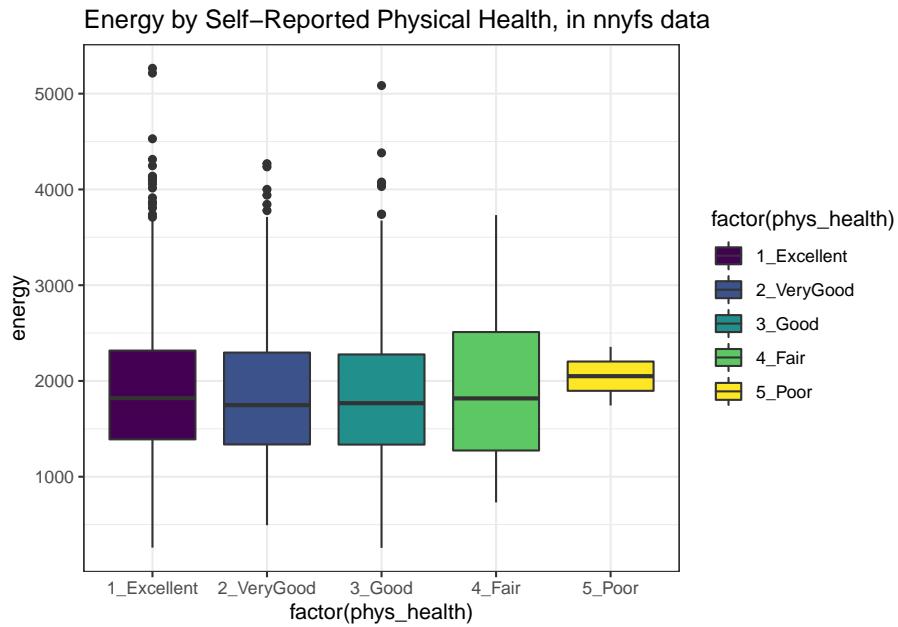
Boxplots are most often used for comparison. We can build boxplots using `ggplot2`, as well, and we'll discuss that in detail later. For now, here's a boxplot built to compare the `energy` results by the subject's race/ethnicity.

```
ggplot(nnyfs, aes(x = factor(race_eth), y = energy, fill=factor(race_eth))) +
  geom_boxplot() +
  guides(fill = FALSE) +
  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.

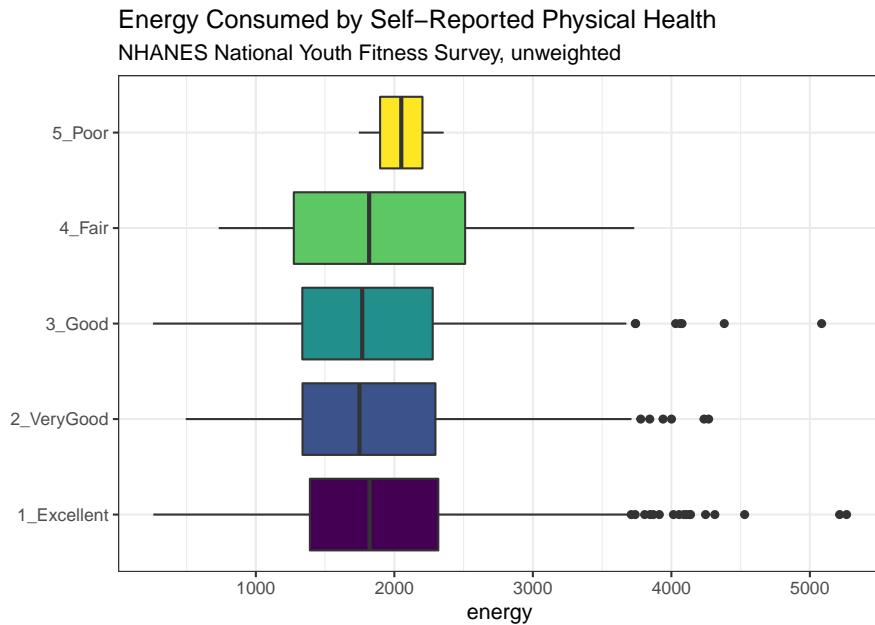
```
ggplot(nnyfs, aes(x = factor(phys_health), y = energy, fill = factor(phys_health))) +
  geom_boxplot() +
  scale_fill_viridis_d() +
  labs(title = "Energy by Self-Reported Physical Health, in nnyfs data")
```



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=FALSE) +
  labs(title = "Energy Consumed by Self-Reported Physical Health",
       subtitle = "NHANES National Youth Fitness Survey, unweighted",
       x = "")
```



7.11 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 = $sd / \sqrt{\text{sample size}}$, useful in inference

7.11.1 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 10% trimmed mean in some settings, but that's just me.

```
mean(nnyfs$energy, trim=.1)
```

```
[1] 1827.1
```

7.11.2 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
```

7.12 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 ($\text{mean} < \text{median}$) is indicated by negative skewness values, while right skew ($\text{mean} > \text{median}$) is indicated by positive values. The skew value will be near zero for data that follow a symmetric distribution.

7.12.1 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.

7.13 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 <i>platykurtic</i> (kurtosis < 0)	“Normalish” <i>mesokurtic</i> (kurtosis = 0)	Heavy-tailed <i>leptokurtic</i> (kurtosis > 0)

```
psych::kurtosi(nnyfs$energy)
```

```
[1] 1.130539
```

7.13.1 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
```

7.14 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()
```

```
.
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: 125.8 126.0 127.0 132.3 144.7
-----
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 13.0, highest: 42.8 43.0 46.9 48.2 48.3
-----
```

More on the `Info` value in `Hmisc::describe` is available here

7.15 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.

Chapter 8

Assessing Normality

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

8.1 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...

```
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?

8.2 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).

8.2.1 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.

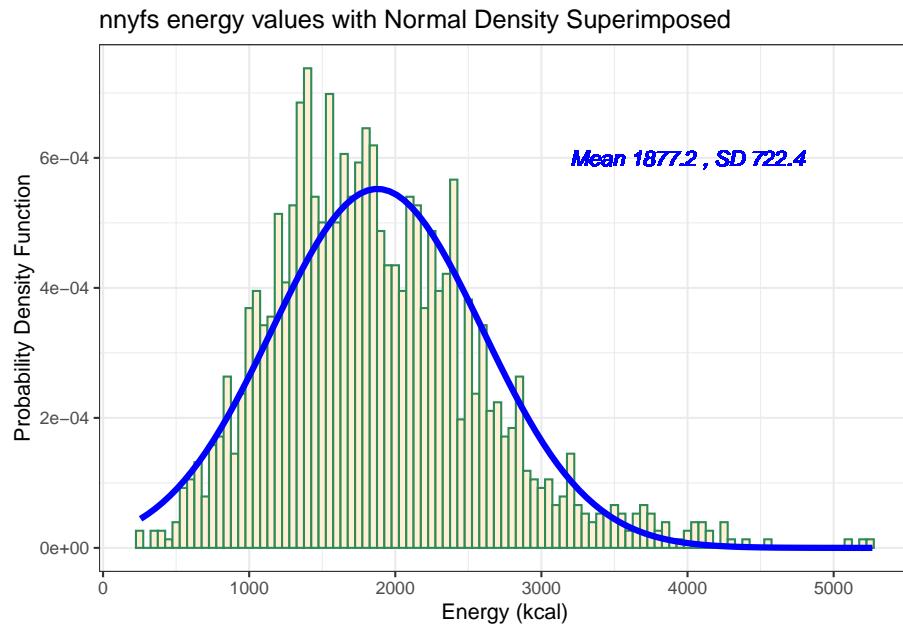
8.3 Comparing a Histogram to a Normal Distribution

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 = "nnys energy values with Normal Density Superimposed",
       x = "Energy (kcal)", y = "Probability Density Function")
```



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?

8.3.1 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.

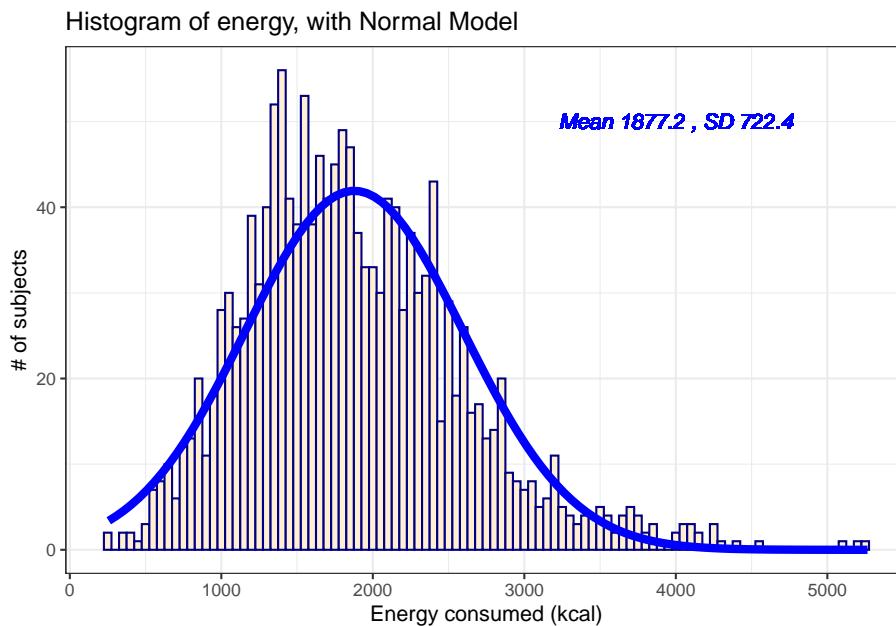
```
res <- mosaic::favstats(~ energy, data = nnyfs)
bin_w <- 50 # specify binwidth

ggplot(nnyfs, aes(x = energy)) +
  geom_histogram(binwidth = bin_w,
                 fill = "papayawhip",
                 col = "navy") +
  theme_bw() +
  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")

```



8.4 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

nnnyfs %>% filter(complete.cases(waist)) %>%
  ggplot(., aes(x = waist)) +
  geom_histogram(binwidth = bin_w,
                 fill = "antiquewhite",

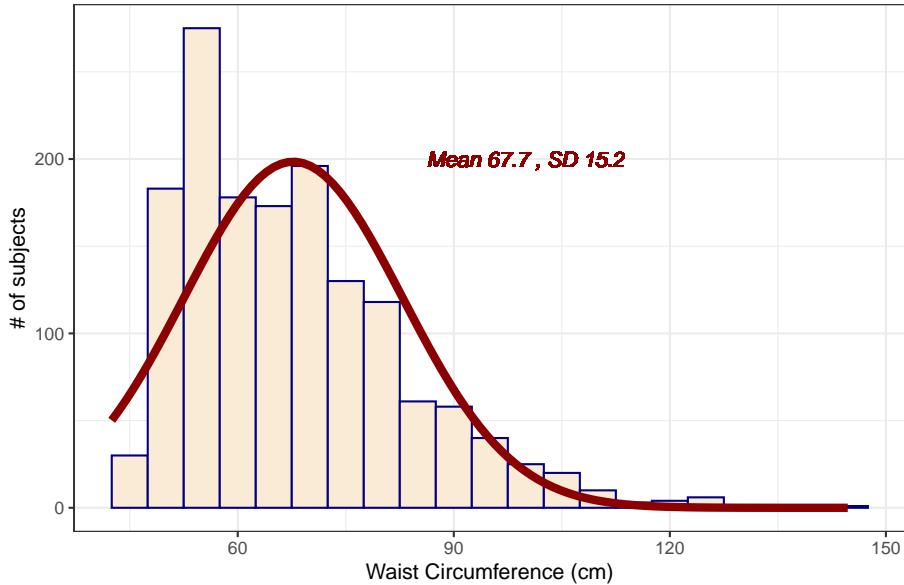
```

```

            col = "navy") +
  theme_bw() +
  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

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%.

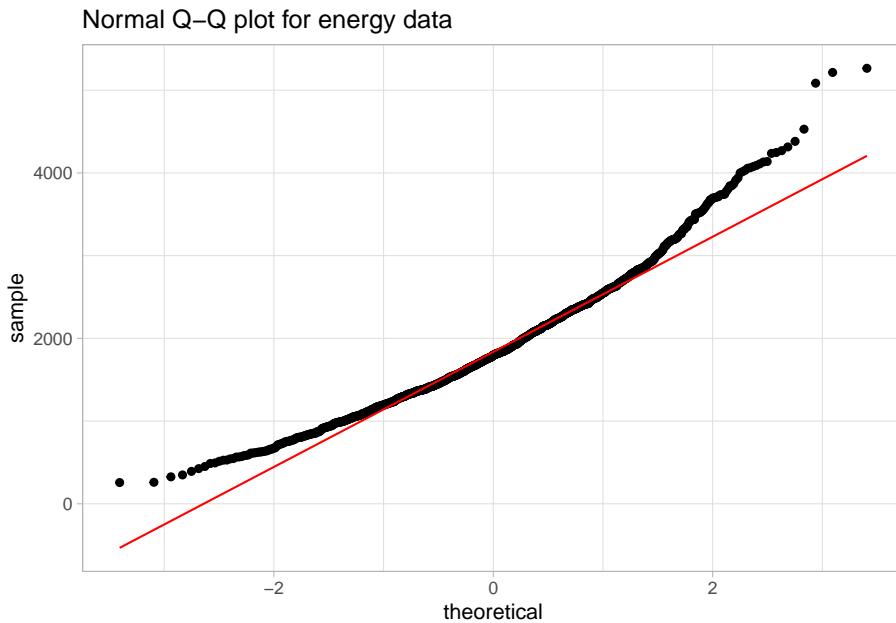
How does the Normal approximation work for waist circumference, according to the Empirical Rule?

8.5 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")
```



8.6 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)

8.6.1 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

# simulate 200 observations from a Normal(20, 5) distribution and place them
# in the d variable within the temp.1 data frame
temp.1 <- data.frame(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") +
  theme_bw() +
  stat_function(
    fun = function(x) dnorm(x, mean = res$mean,
                           sd = res$sd) *
      res$n * bin_w,
    col = "blue", size = 1.5) +
```

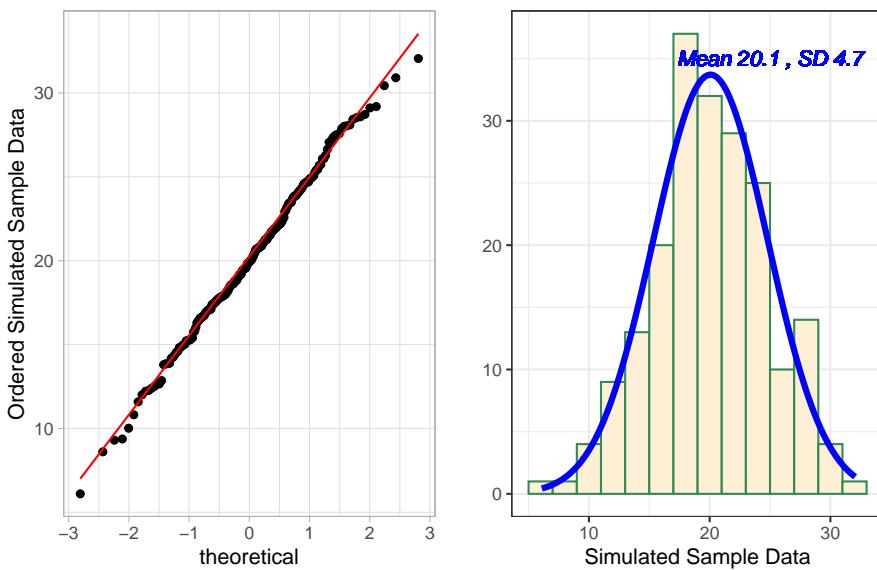
```

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 distribution")

```

200 observations from a simulated Normal distribution



```
# 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

8.6.2 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 <- data.frame(e1 = rbeta(200, 5, 2), e2 = rbeta(200, 1, 5))

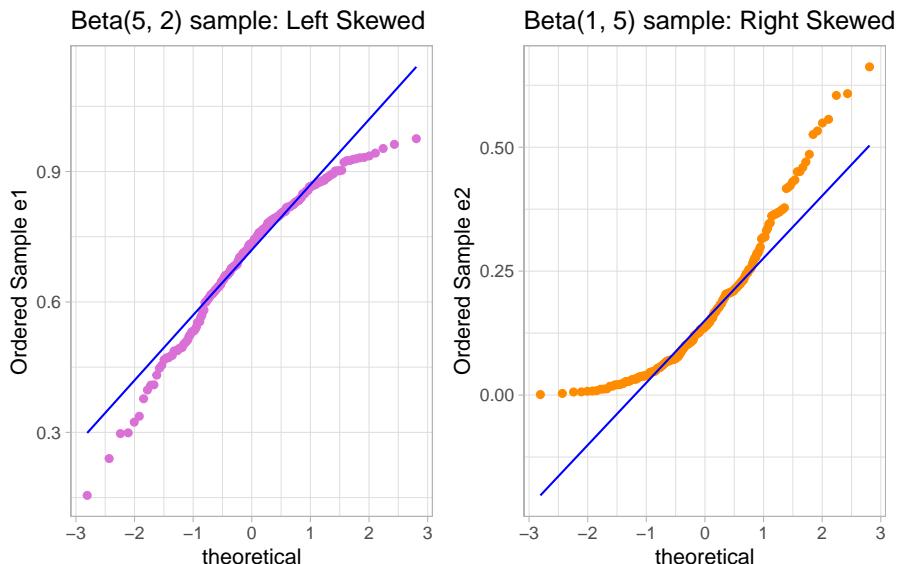
p1 <- ggplot(temp.2, aes(sample = e1)) +
  geom_qq(col = "orchid") + geom_qq_line(col = "blue") +
  theme_light() +
  labs(y = "Ordered Sample e1",
       title = "Beta(5, 2) sample: Left Skewed")

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 Beta distributions")

```

200 observations from simulated Beta distributions



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

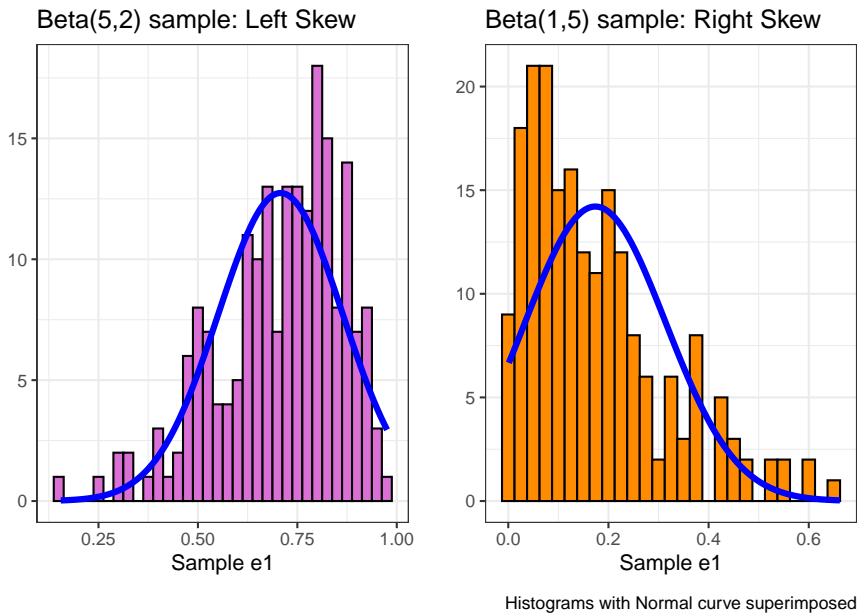
```

```
p1 <- ggplot(temp.2, aes(x = e1)) +
  geom_histogram(binwidth = bin_w1,
                 fill = "orchid",
                 col = "black") +
  theme_bw() +
  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") +
  theme_bw() +
  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")
```



8.6.3 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.

8.6.4 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 <- data.frame(s1 = rcauchy(200, location=10, scale = 1),
                      s2 = runif(200, -30, 30))
```

```

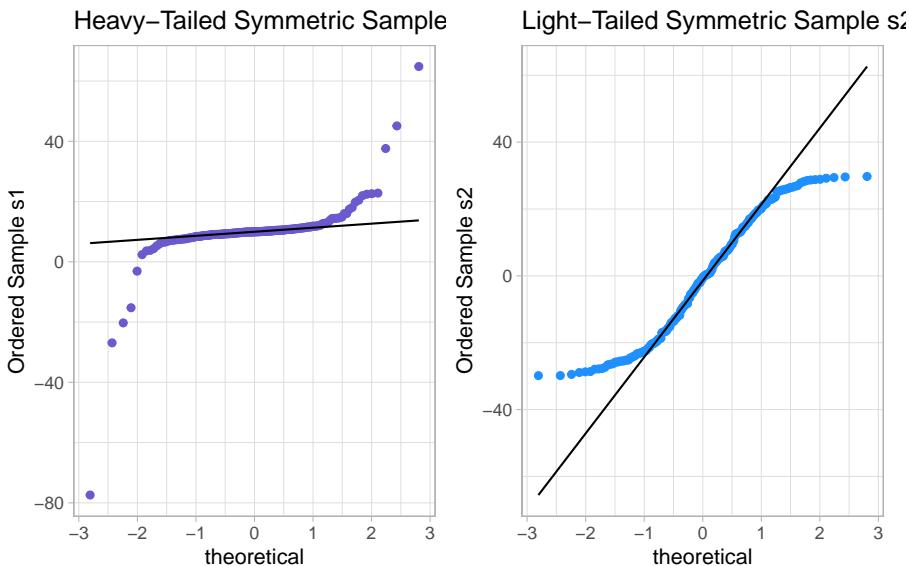
p1 <- ggplot(temp.3, aes(sample = s1)) +
  geom_qq(col = "slateblue") + geom_qq_line(col = "black") +
  theme_light() +
  labs(y = "Ordered Sample s1",
       title = "Heavy-Tailed Symmetric Sample s1")

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

p1 <- ggplot(temp.3, aes(x = s1)) +
  geom_histogram(binwidth = bin_w1,
                 fill = "slateblue",
                 col = "white") +
  theme_bw() +

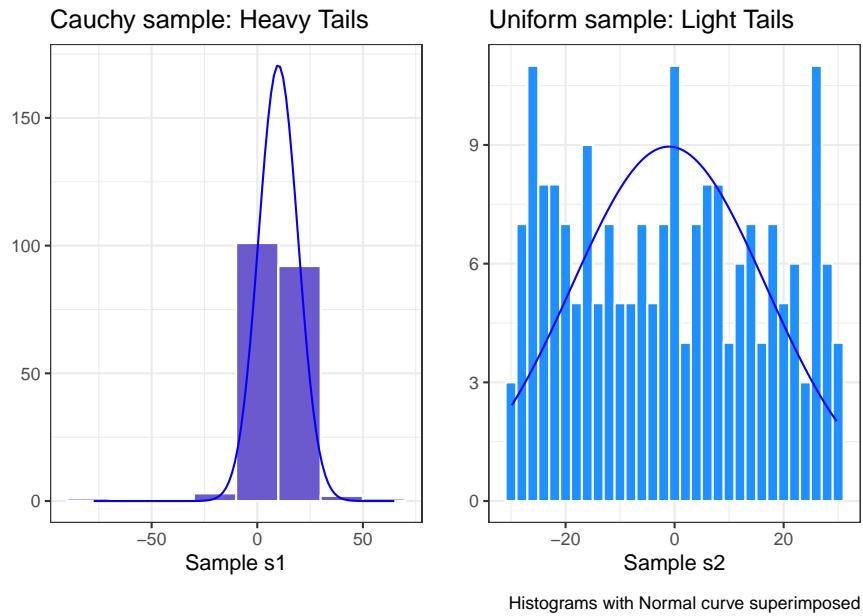
```

```
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") +
  theme_bw() +
  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")
```



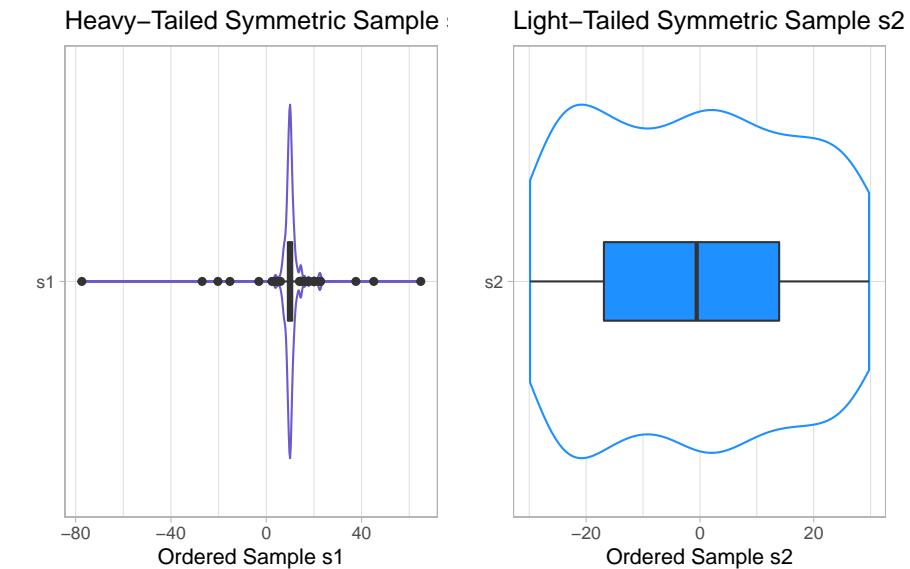
Instead, boxplots (here augmented with violin plots) can be more helpful when thinking about light-tailed vs. heavy-tailed distributions.

```
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
```

8.7 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

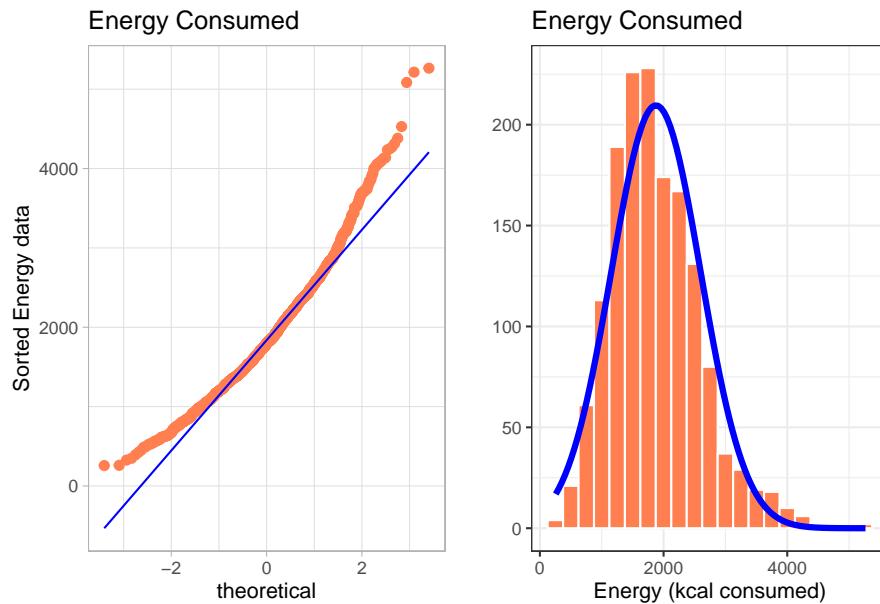
p2 <- ggplot(nnyfs, aes(x = energy)) +
  geom_histogram(binwidth = bin_w,
                 fill = "coral",
                 col = "white") +
  theme_bw() +
  stat_function()
```

```

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() +

```

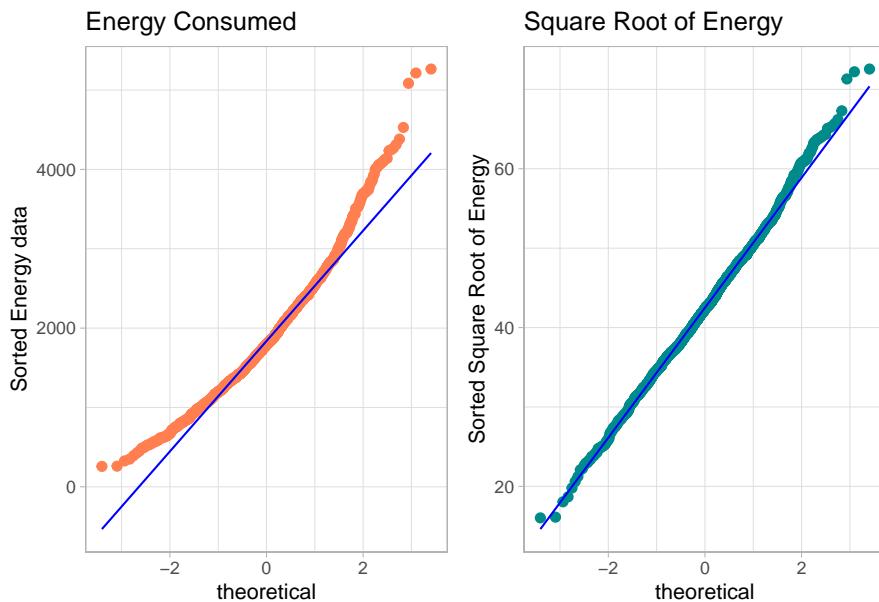
```

  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 models superimposed.

```

res <- mosaic::favstats(~ energy, data = nnyfs)
bin_w <- 250 # specify binwidth

p1 <- ggplot(nnyfs, aes(x = energy)) +

```

```

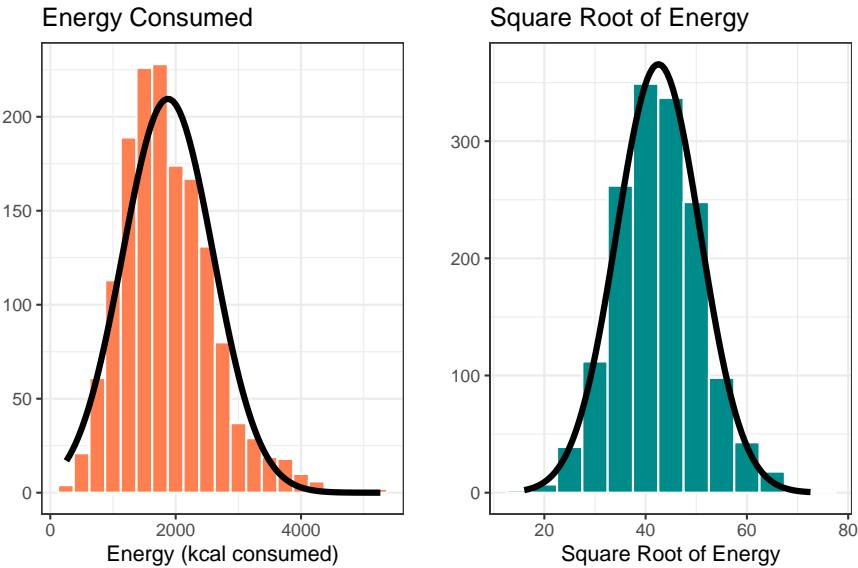
geom_histogram(binwidth = bin_w,
               fill = "coral",
               col = "white") +
  theme_bw() +
  stat_function(
    fun = function(x) dnorm(x, mean = res$mean,
                            sd = res$sd) *
      res$n * bin_w,
    col = "black", size = 1.5) +
  labs(x = "Energy (kcal consumed)", 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") +
  theme_bw() +
  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 $\text{sqrt}(\text{energy})$ 

```
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 next Chapter provides some guidance about choosing from a class of power transformations that can reduce the impact of non-Normality in unimodal data.

Chapter 9

Using Transformations to “Normalize” Distributions

- 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.

9.1 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	$x^{-2} = 1/x^2$

9.2 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.

9.3 Protein Consumption in the NNYFS data

Here are the protein consumption (in grams) results from the NNYFS data.

```
mosaic::favstats(~ protein, data = nnyfs)

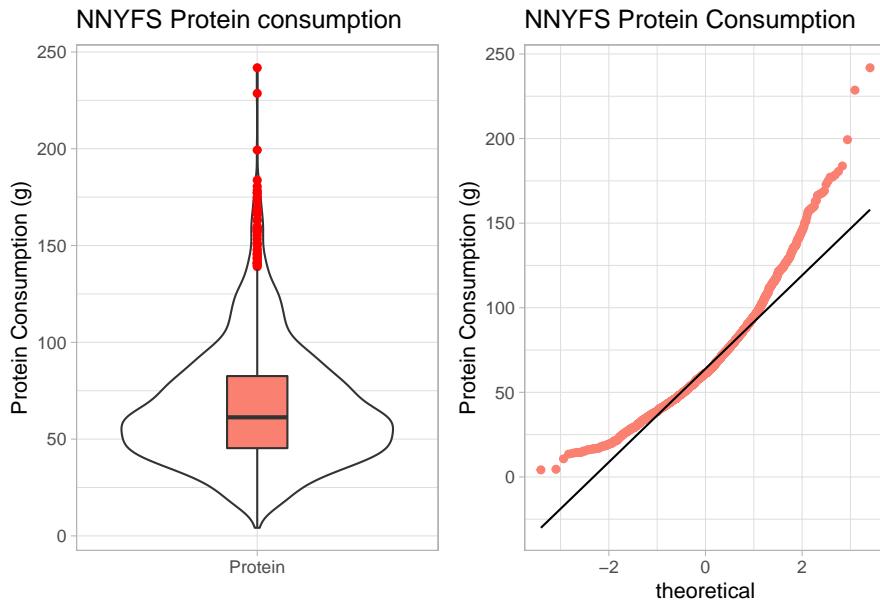
    min      Q1 median      Q3      max      mean       sd      n missing
4.18 45.33 61.255 82.565 241.84 66.90148 30.96319 1518        0

p1 <- ggplot(nnyfs, aes(x = "Protein", y = protein)) +
  geom_violin() +
  geom_boxplot(width = 0.2, fill = "salmon",
               outlier.color = "red") +
  theme_light() +
  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") +
  theme_light()
```

```
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.

9.3.1 Using patchwork to compose plots

For me, 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") +
  theme_light() +
  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")

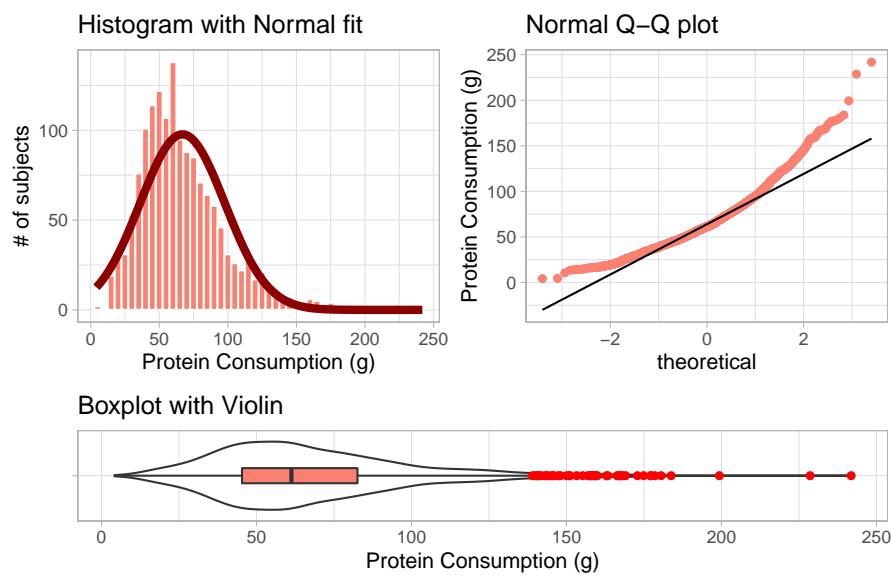
p2 <- ggplot(nnyfs, aes(sample = protein)) +
  geom_qq(col = "salmon") +
  geom_qq_line(col = "black") +
  theme_light() +
  labs(title = "Normal Q-Q plot",
       y = "Protein Consumption (g)")

p3 <- ggplot(nnyfs, aes(x = "", y = protein)) +
  geom_violin() +
  geom_boxplot(width = 0.2, fill = "salmon",
               outlier.color = "red") +
  theme_light() +
  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



For more on the `patchwork` package, check out its repository at <https://github.com/thomasp85/patchwork>.

9.4 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.

9.4.1 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

p1 <- ggplot(nnyfs, aes(x = sqrt(protein))) +
  geom_histogram(binwidth = bin_w,
                 fill = "salmon",
                 col = "white") +
  theme_light() +
  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") +
  theme_light() +
  labs(title = "Normal Q-Q plot",
       y = "Square Root of Protein Consumption (g)")

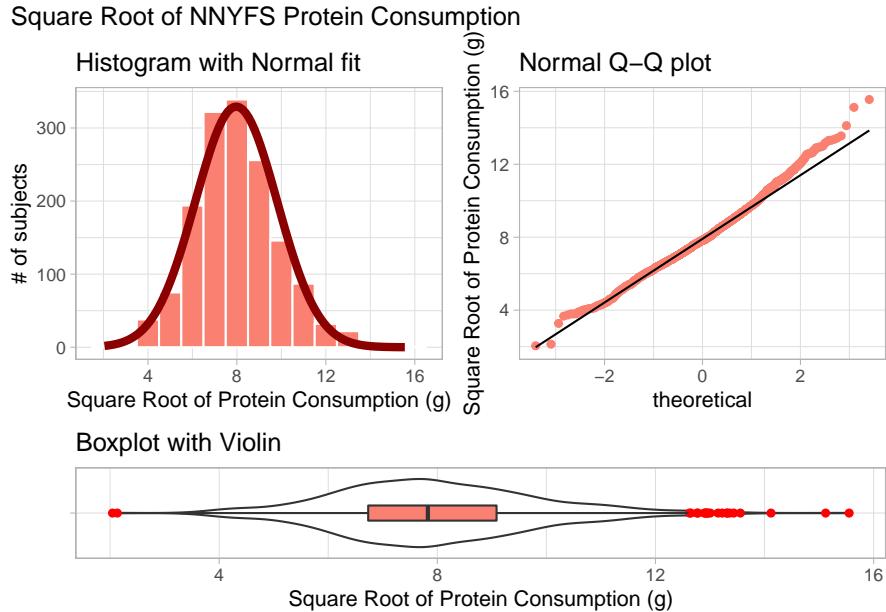
p3 <- ggplot(nnyfs, aes(x = "", y = sqrt(protein))) +
  geom_violin() +
  geom_boxplot(width = 0.2, fill = "salmon",
               outlier.color = "red") +
  theme_light() +
  coord_flip()
```

```

  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")

```



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?

9.4.2 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.

```

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") +
  theme_light() +
  stat_function(
    fun = dnorm,
    args = list(mu = mean(log(protein)),
               sd = sd(log(protein)))
  )

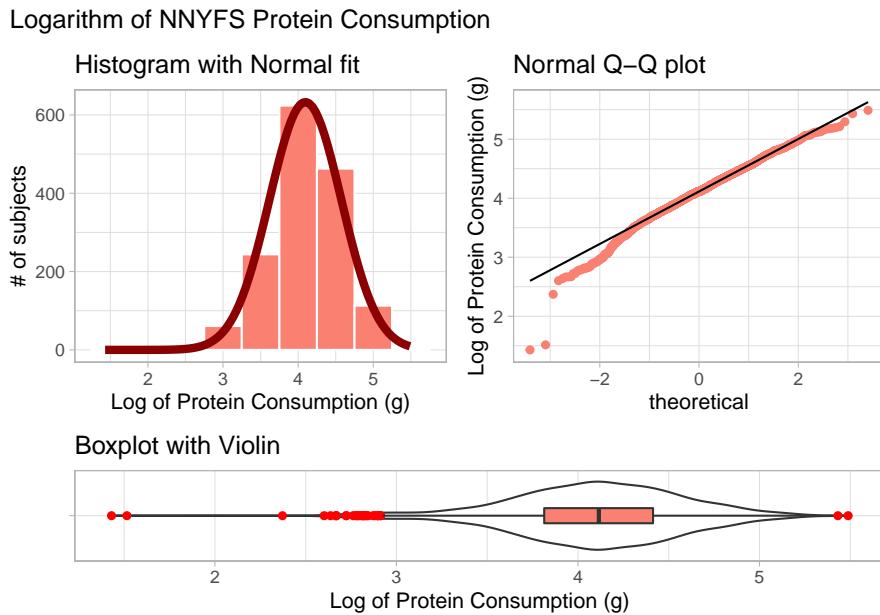
```

```
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") +
  theme_light() +
  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") +
  theme_light() +
  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.

9.4.3 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.

9.5 What if we considered all 9 available transformations?

```
p1 <- ggplot(nnyfs, aes(sample = protein^3)) +
  geom_qq(col = "salmon") +
  geom_qq_line(col = "black") +
  theme_light() +
  labs(title = "Cube (power 3)",
       y = "Protein, Cubed")

p2 <- ggplot(nnyfs, aes(sample = protein^2)) +
  geom_qq(col = "salmon") +
```

```

geom_qq_line(col = "black") +
theme_light() +
labs(title = "Square (power 2)",
y = "Protein, Squared")

p3 <- ggplot(nnyfs, aes(sample = protein)) +
geom_qq(col = "salmon") +
geom_qq_line(col = "black") +
theme_light() +
labs(title = "Original Data",
y = "Protein (g)")

p4 <- ggplot(nnyfs, aes(sample = sqrt(protein))) +
geom_qq(col = "salmon") +
geom_qq_line(col = "black") +
theme_light() +
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") +
theme_light() +
labs(title = "log (power 0)",
y = "Natural Log of Protein")

p6 <- ggplot(nnyfs, aes(sample = protein-0.5)) +
geom_qq(col = "salmon") +
geom_qq_line(col = "black") +
theme_light() +
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") +
theme_light() +
labs(title = "Inverse (power -1)",
y = "1/Protein")

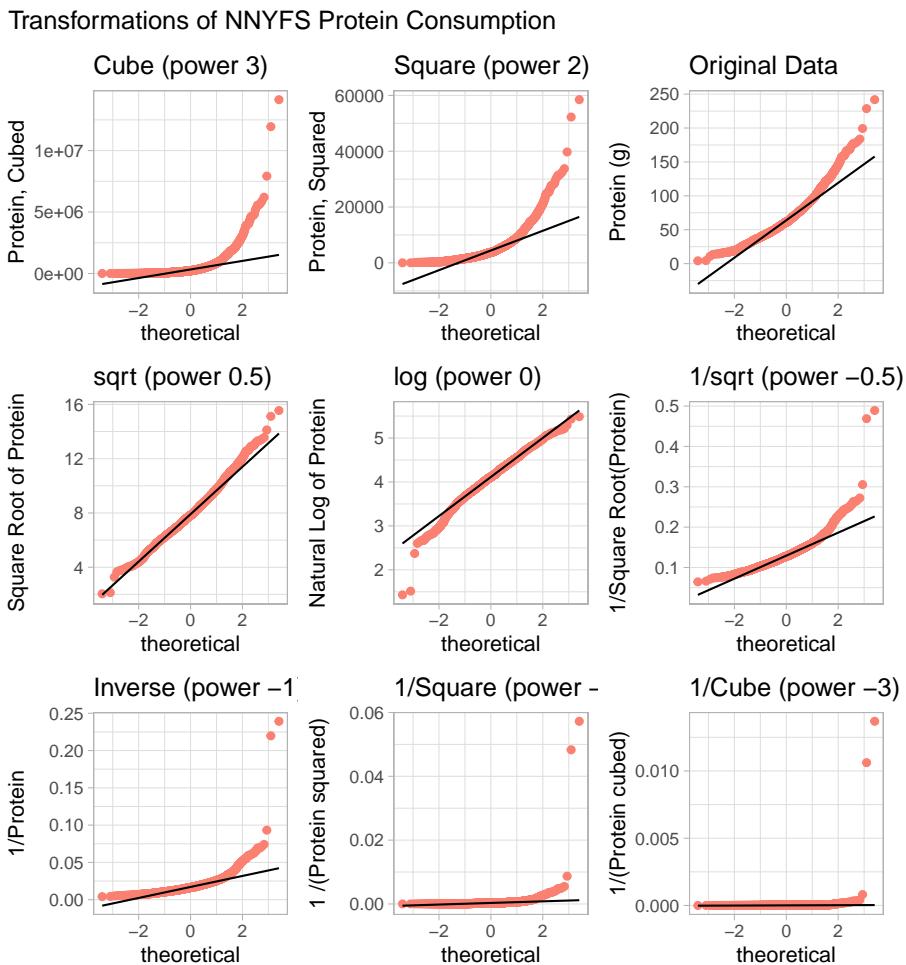
p8 <- ggplot(nnyfs, aes(sample = 1/(protein2))) +
geom_qq(col = "salmon") +
geom_qq_line(col = "black") +

```

```
theme_light() +
  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") +
  theme_light() +
  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")
```



The square root still appears to be the best choice of transformation here, even after we consider all 8 transformation of the raw data.

9.6 A Simulated Data Set

```
set.seed(431);
data2 <- data.frame(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 = data2)
bin_w <- 4 # specify binwidth
```

```

p1 <- ggplot(data2, aes(x = sample2)) +
  geom_histogram(binwidth = bin_w,
                 fill = "royalblue",
                 col = "white") +
  theme_light() +
  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")

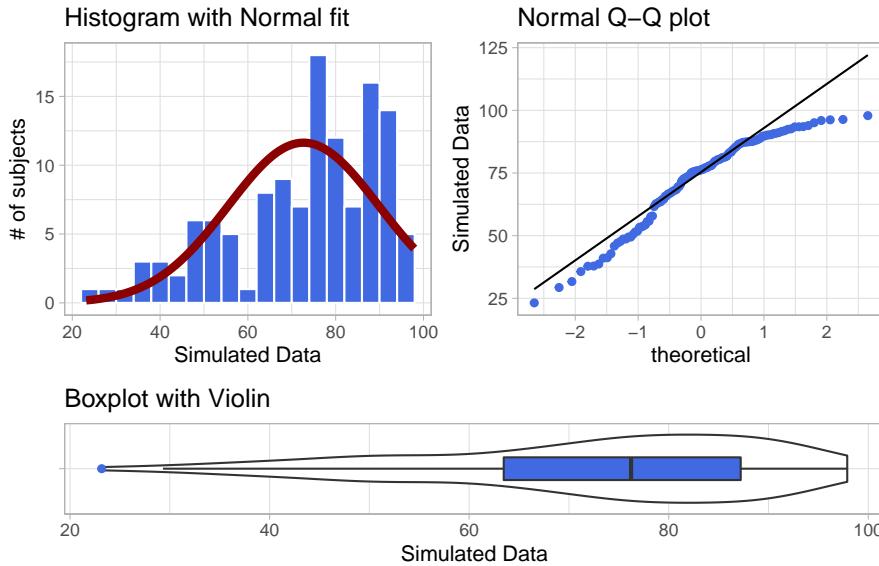
p2 <- ggplot(data2, aes(sample = sample2)) +
  geom_qq(col = "royalblue") +
  geom_qq_line(col = "black") +
  theme_light() +
  labs(title = "Normal Q-Q plot",
       y = "Simulated Data")

p3 <- ggplot(data2, aes(x = "", y = sample2)) +
  geom_violin() +
  geom_boxplot(width = 0.3, fill = "royalblue",
               outlier.color = "royalblue") +
  theme_light() +
  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?

```
res <- mosaic::favstats(~ sample2^2, data = data2)
bin_w <- 600 # specify binwidth

p1 <- ggplot(data2, aes(x = sample2^2)) +
  geom_histogram(binwidth = bin_w,
                 fill = "royalblue",
                 col = "white") +
  theme_light() +
  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(data2, aes(sample = sample2^2)) +
  geom_qq(col = "royalblue") +
  geom_qq_line(col = "black") +
  theme_light() +
  labs(title = "Normal Q-Q plot",
```

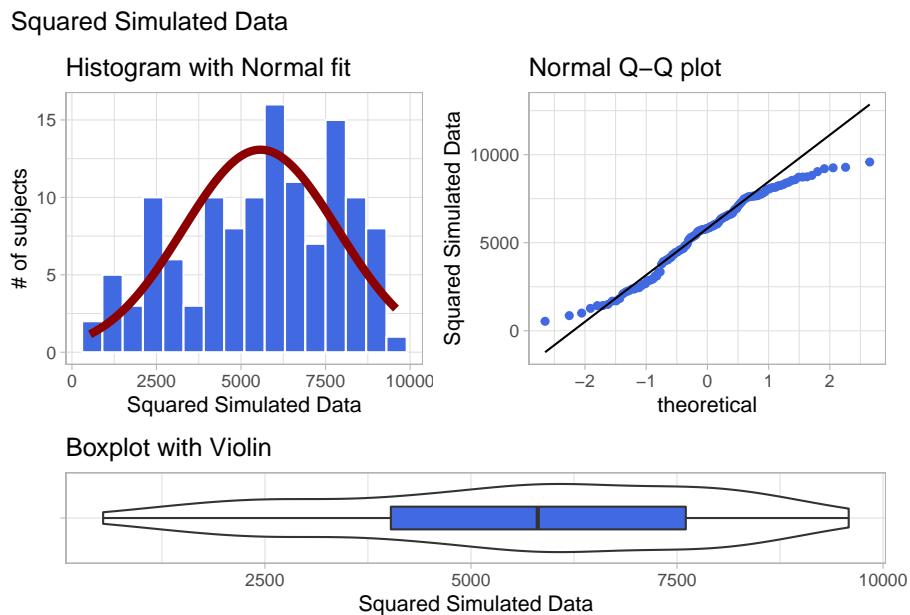
```

y = "Squared Simulated Data")

p3 <- ggplot(data2, aes(x = "", y = sample2^2)) +
  geom_violin() +
  geom_boxplot(width = 0.3, fill = "royalblue",
               outlier.color = "royalblue") +
  theme_light() +
  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 = data2)
bin_w <- 100000 # specify binwidth

p1 <- ggplot(data2, aes(x = sample2^3)) +
  geom_histogram(binwidth = bin_w,
                 fill = "royalblue",
                 col = "white") +
  theme_light() +
  stat_function(
    fun = dnorm,
    args = list(mean = mean(sample2^3),
               sd = sd(sample2^3)),
    color = "black",
    size = 1.5)

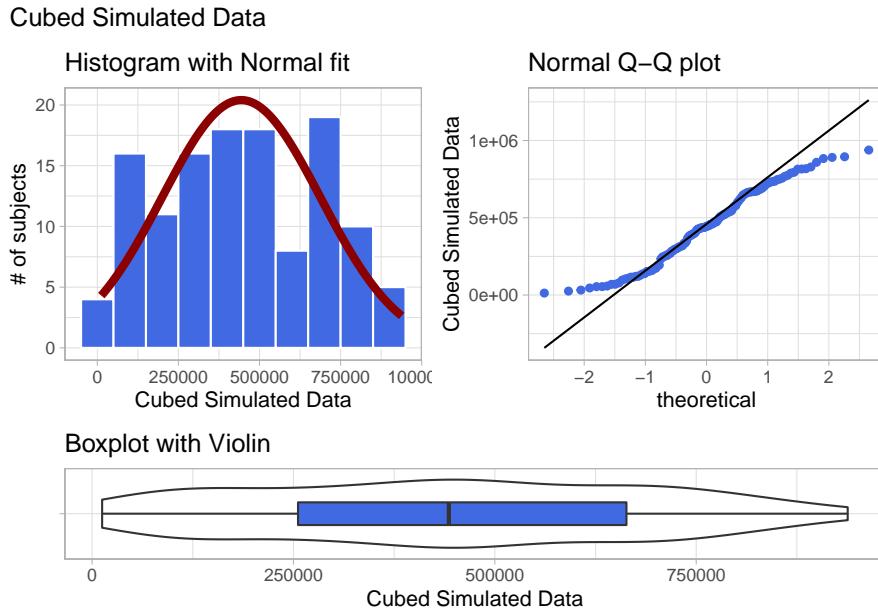
```

```
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(data2, aes(sample = sample2^3)) +
  geom_qq(col = "royalblue") +
  geom_qq_line(col = "black") +
  theme_light() +
  labs(title = "Normal Q-Q plot",
       y = "Cubed Simulated Data")

p3 <- ggplot(data2, aes(x = "", y = sample2^3)) +
  geom_violin() +
  geom_boxplot(width = 0.3, fill = "royalblue",
               outlier.color = "royalblue") +
  theme_light() +
  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")
```



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.

9.7 What if we considered all 9 available transformations?

```
p1 <- ggplot(data2, aes(sample = sample2^3)) +
  geom_qq(col = "royalblue") +
  geom_qq_line(col = "black") +
  theme_light() +
  labs(title = "Cube (power 3)")

p2 <- ggplot(data2, aes(sample = sample2^2)) +
  geom_qq(col = "royalblue") +
  geom_qq_line(col = "black") +
  theme_light() +
  labs(title = "Square (power 2)")

p3 <- ggplot(data2, aes(sample = sample2)) +
```

```

geom_qq(col = "royalblue") +
geom_qq_line(col = "black") +
theme_light() +
labs(title = "Original Data")

p4 <- ggplot(data2, aes(sample = sqrt(sample2))) +
geom_qq(col = "royalblue") +
geom_qq_line(col = "black") +
theme_light() +
labs(title = "sqrt (power 0.5)")

p5 <- ggplot(data2, aes(sample = log(sample2))) +
geom_qq(col = "royalblue") +
geom_qq_line(col = "black") +
theme_light() +
labs(title = "log (power 0)")

p6 <- ggplot(data2, aes(sample = sample2^(0.5))) +
geom_qq(col = "royalblue") +
geom_qq_line(col = "black") +
theme_light() +
labs(title = "1/sqrt (power -0.5)")

p7 <- ggplot(data2, aes(sample = 1/sample2)) +
geom_qq(col = "royalblue") +
geom_qq_line(col = "black") +
theme_light() +
labs(title = "Inverse (power -1)")

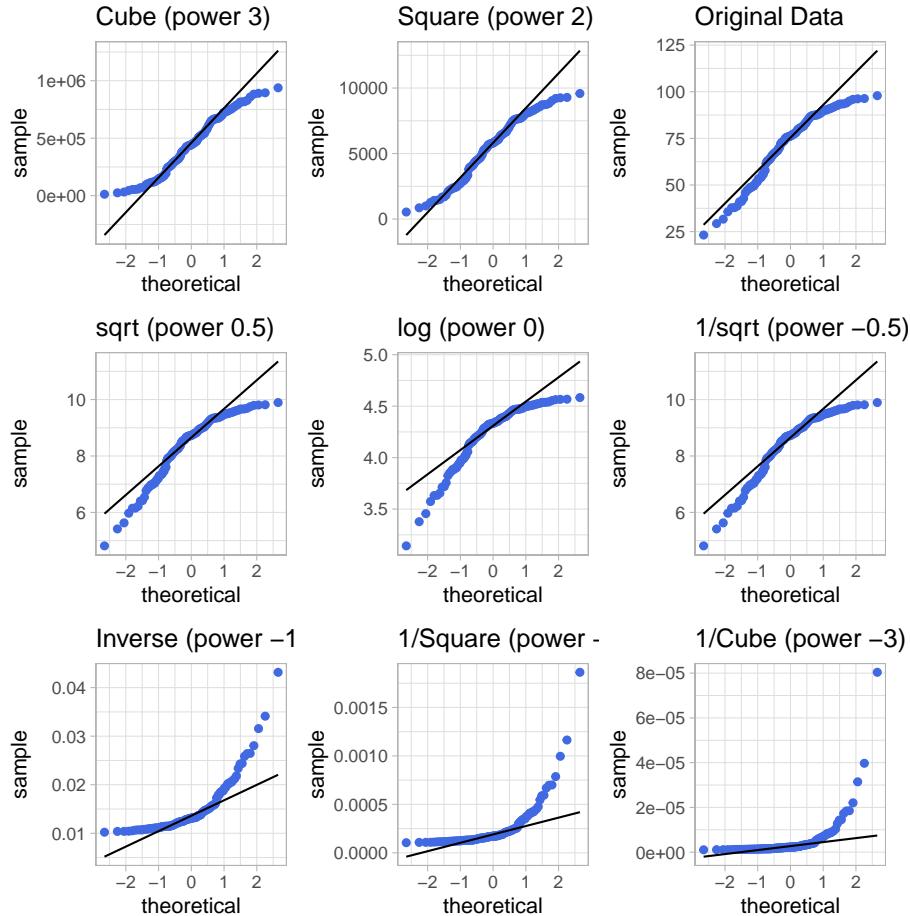
p8 <- ggplot(data2, aes(sample = 1/(sample2^2))) +
geom_qq(col = "royalblue") +
geom_qq_line(col = "black") +
theme_light() +
labs(title = "1/Square (power -2)")

p9 <- ggplot(data2, aes(sample = 1/(sample2^3))) +
geom_qq(col = "royalblue") +
geom_qq_line(col = "black") +
theme_light() +
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.

```
rm(p1, p2, p3, p4, p5, p6, p7, p8, p9, res, bin_w, data2)
```

Chapter 10

Summarizing data within subgroups

10.1 Using dplyr and summarise to build a tibble of summary information

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.

```
nyfs %>%
  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
  <chr>           <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.

```
nyfs %>%
  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
  <chr>           <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 = FALSE) +
  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 = FALSE) +
  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 = FALSE) +
  labs(title = "Sugar Comparison")

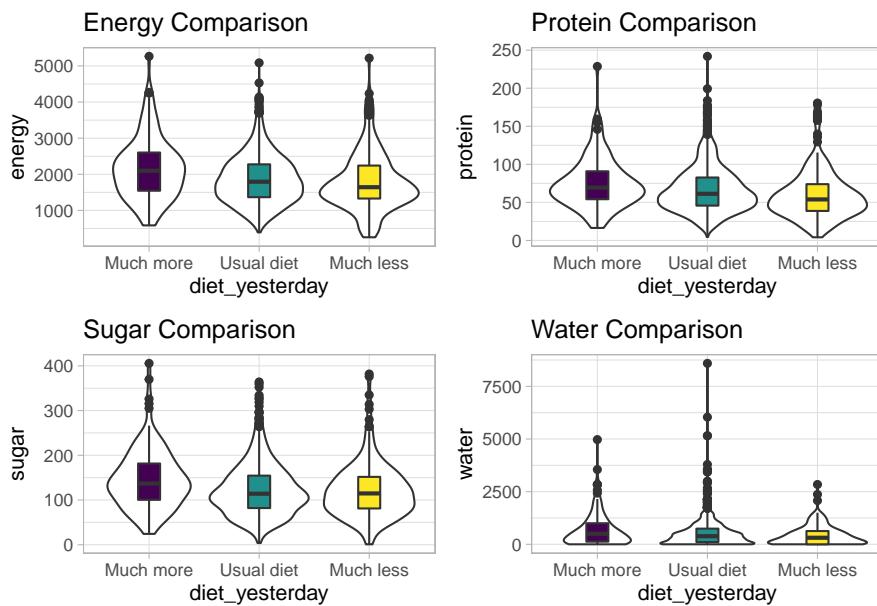
p4 <- ggplot(temp_dat, aes(x = diet_yesterday, y = water)) +
```

```

geom_violin() +
  geom_boxplot(aes(fill = diet_yesterday), width = 0.2) +
  theme_light() +
  scale_fill_viridis_d() +
  guides(fill = FALSE) +
  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.

10.2 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))

```

```
`summarise()` ungrouping output (override with `groups` argument)

# A tibble: 5 x 5
  bmi_cat      mean     sd median skew_1
  <chr>       <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))
```

```
`summarise()` ungrouping output (override with `groups` argument)

# A tibble: 4 x 6
  bmi_cat      count    mean     sd median skew_1
  <chr>       <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) %>%
  kable(digits = 1)
```

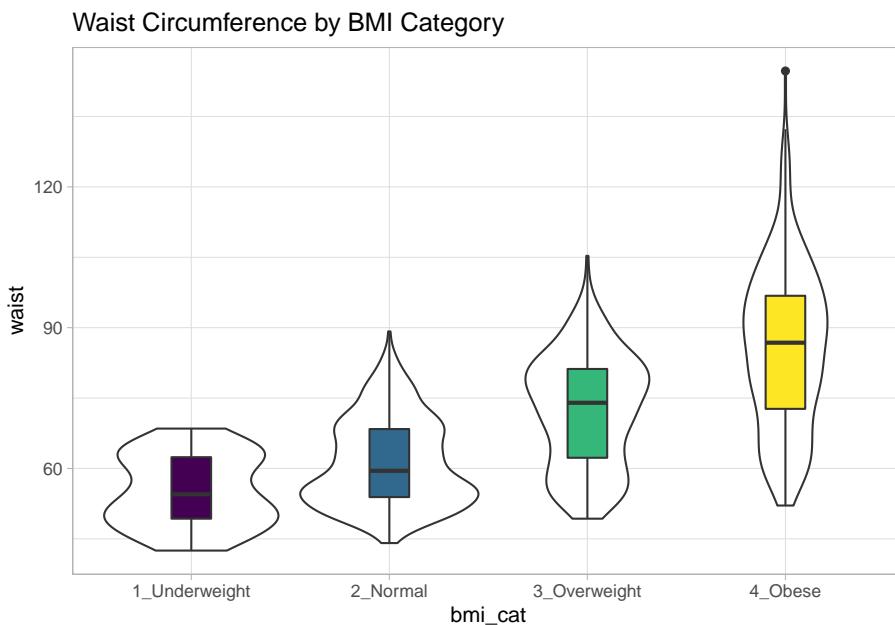
bmi_cat	min	Q1	median	Q3	max	mean	sd	n	missing
1_Underweight	42.5	49.3	54.5	62.4	68.5	55.2	7.6	41	0
2_Normal	44.1	53.9	59.5	68.4	89.2	61.2	9.4	917	3
3_Overweight	49.3	62.3	74.0	81.2	105.3	72.3	11.9	258	0
4_Obese	52.1	72.7	86.8	96.8	144.7	85.6	17.1	294	1

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 %>%
  filter(complete.cases(bmi_cat, waist)) %>%
  ggplot(., 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 = FALSE) +
  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 Grolemund and Wickham (2019).

10.3 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.

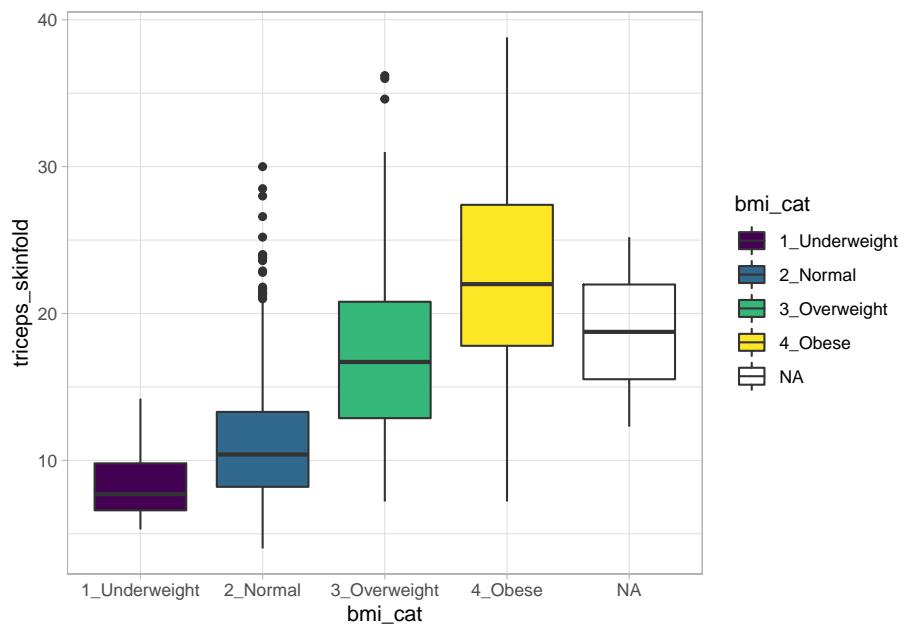
```

ggplot(nnyfs, aes(x = bmi_cat, y = triceps_skinfold,
                  fill = bmi_cat)) +

```

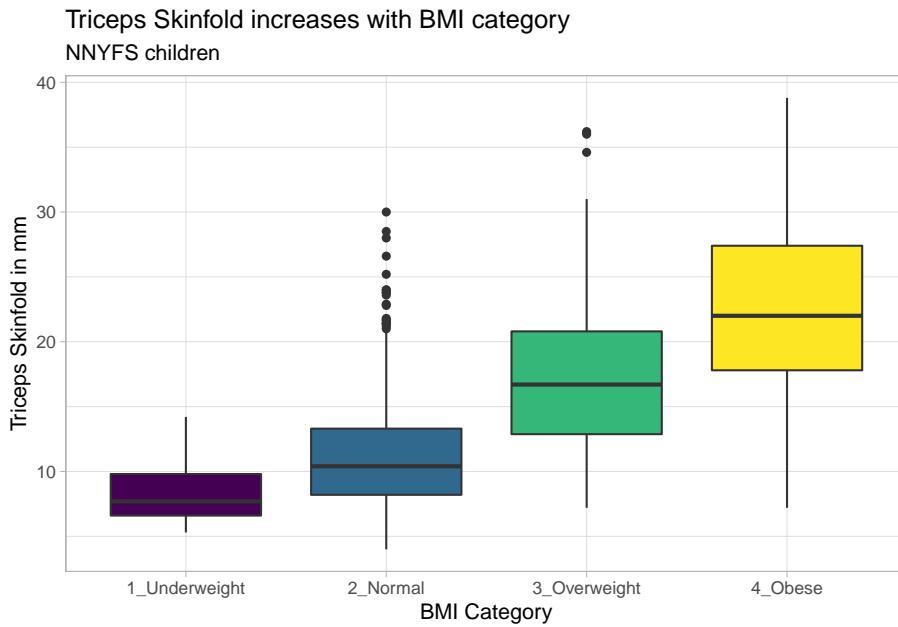
```
geom_boxplot() +
scale_fill_viridis_d() +
theme_light()
```

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 %>%
filter(complete.cases(bmi_cat, triceps_skinfold)) %>%
ggplot(., aes(x = bmi_cat, y = triceps_skinfold,
fill = bmi_cat)) +
geom_boxplot() +
scale_fill_viridis_d() +
guides(fill = FALSE) +
theme_light() +
labs(x = "BMI Category", y = "Triceps Skinfold in mm",
title = "Triceps Skinfold increases with BMI category",
subtitle = "NNYFS children")
```

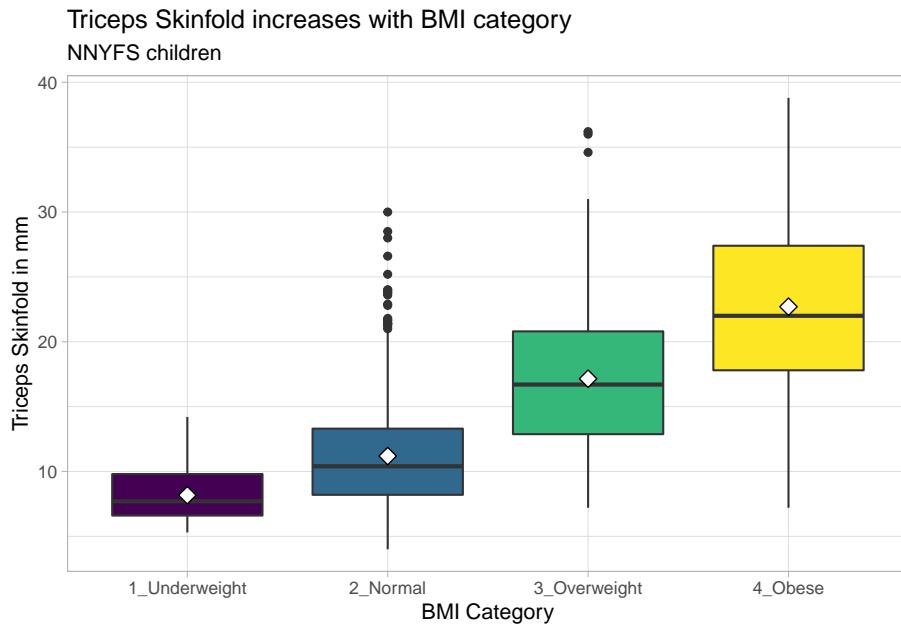


As always, the boxplot shows the five-number summary (minimum, 25th percentile, median, 75th percentile and maximum) in addition to highlighting candidate outliers.

10.3.1 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.)

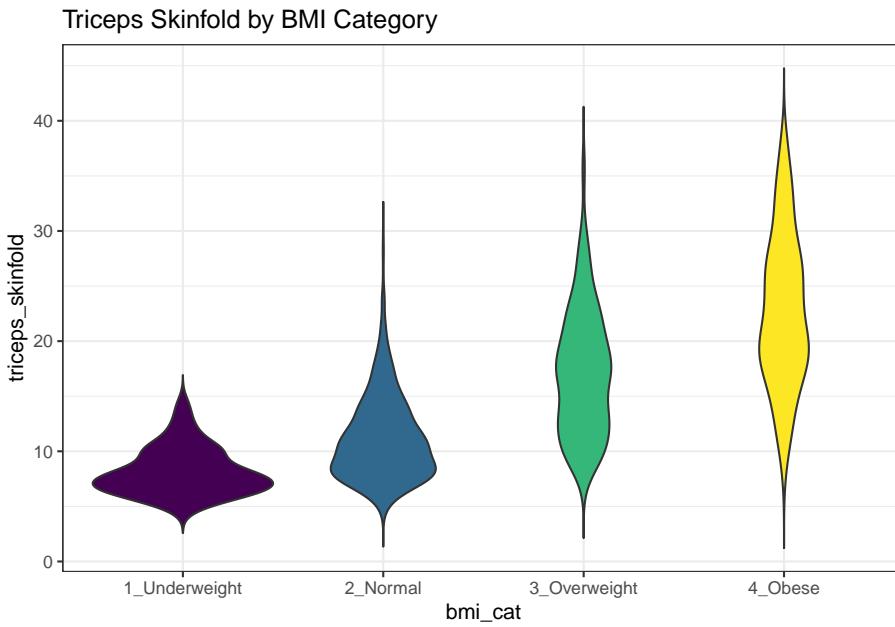
```
nnyfs %>%
  filter(complete.cases(bmi_cat, triceps_skinfold)) %>%
  ggplot(., aes(x = bmi_cat, y = triceps_skinfold,
                 fill = bmi_cat)) +
  geom_boxplot() +
  stat_summary(fun="mean", geom="point",
              shape=23, size=3, fill="white") +
  scale_fill_viridis_d() +
  guides(fill = FALSE) +
  theme_light() +
  labs(x = "BMI Category", y = "Triceps Skinfold in mm",
       title = "Triceps Skinfold increases with BMI category",
       subtitle = "NNYFS children")
```



10.3.2 Building a Violin Plot

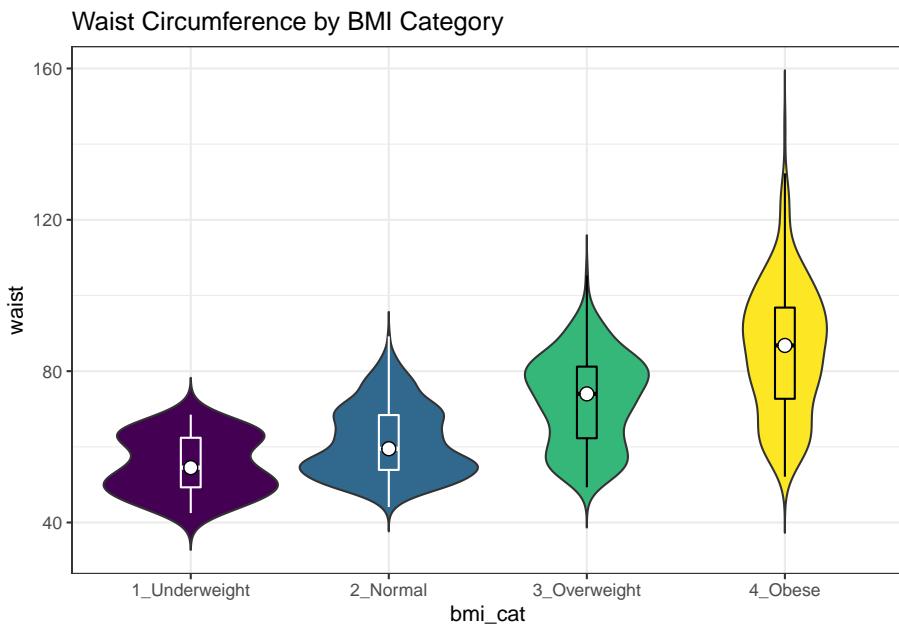
There are a number of other plots which compare distributions of data sets. An interesting one is called a **violin plot**. A violin plot is a kernel density estimate, mirrored to form a symmetrical shape.

```
nnyfs %>%
  filter(complete.cases(triceps_skinfold, bmi_cat)) %>%
  ggplot(., 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")
```



Traditionally, these plots are shown with overlaid boxplots and a white dot at the median, like this example, now looking at waist circumference again.

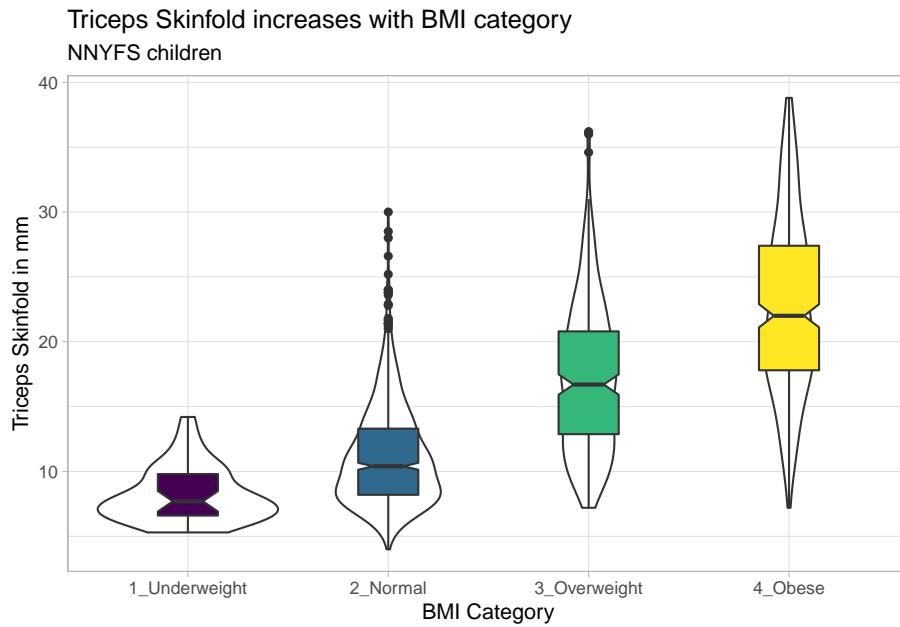
```
nyfs %>%
  filter(complete.cases(waist, bmi_cat)) %>%
  ggplot(., aes(x = bmi_cat, y = waist,
    fill = bmi_cat)) +
  geom_violin(trim=FALSE) +
  geom_boxplot(width=.1, outlier.colour=NA,
    color = c(rep("white",2), rep("black",2))) +
  stat_summary(fun=median, geom="point",
    fill="white", shape=21, size=3) +
  scale_fill_viridis_d() +
  guides(fill = "none") +
  labs(title = "Waist Circumference by BMI Category")
```



10.3.3 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.

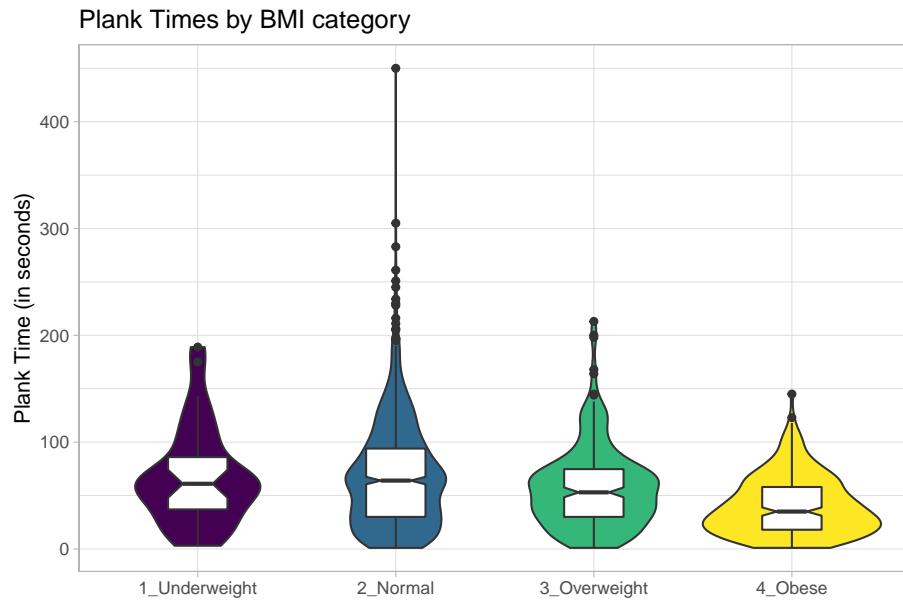
```
nnyfs %>%
  filter(complete.cases(bmi_cat, triceps_skinfold)) %>%
  ggplot(., aes(x = bmi_cat, y = triceps_skinfold)) +
  geom_violin() +
  geom_boxplot(aes(fill = bmi_cat), width = 0.3, notch = TRUE) +
  scale_fill_viridis_d() +
  guides(fill = FALSE) +
  theme_light() +
  labs(x = "BMI Category", y = "Triceps Skinfold in mm",
       title = "Triceps Skinfold increases with BMI category",
       subtitle = "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.

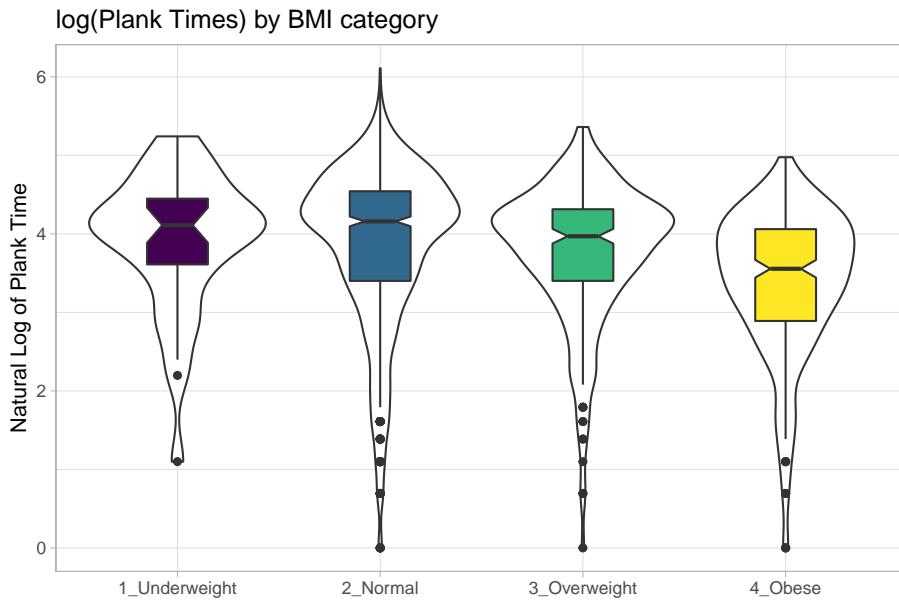
```
nnyfs %>%
  filter(complete.cases(bmi_cat, plank_time)) %>%
  ggplot(., aes(x=bmi_cat, y=plank_time)) +
  geom_violin(aes(fill = bmi_cat)) +
  geom_boxplot(width = 0.3, notch=TRUE) +
  scale_fill_viridis_d() +
  guides(fill = "none") +
  theme_light() +
  labs(title = "Plank Times by BMI category",
       x = "", y = "Plank Time (in seconds)")
```



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?

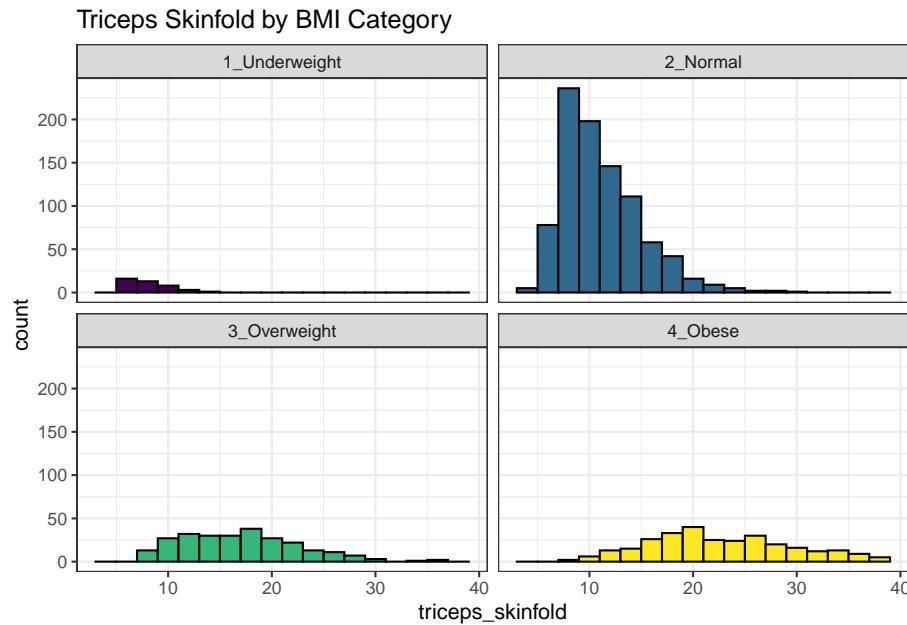
```
nnyfs %>%
  filter(complete.cases(bmi_cat, plank_time)) %>%
  ggplot(., aes(x=bmi_cat, y = log(plank_time))) +
  geom_violin() +
  geom_boxplot(aes(fill = bmi_cat), width = 0.3, notch=TRUE) +
  scale_fill_viridis_d() +
  guides(fill = "none") +
  theme_light() +
  labs(title = "log(Plank Times) by BMI category",
       x = "", y = "Natural Log of Plank Time")
```



10.4 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.

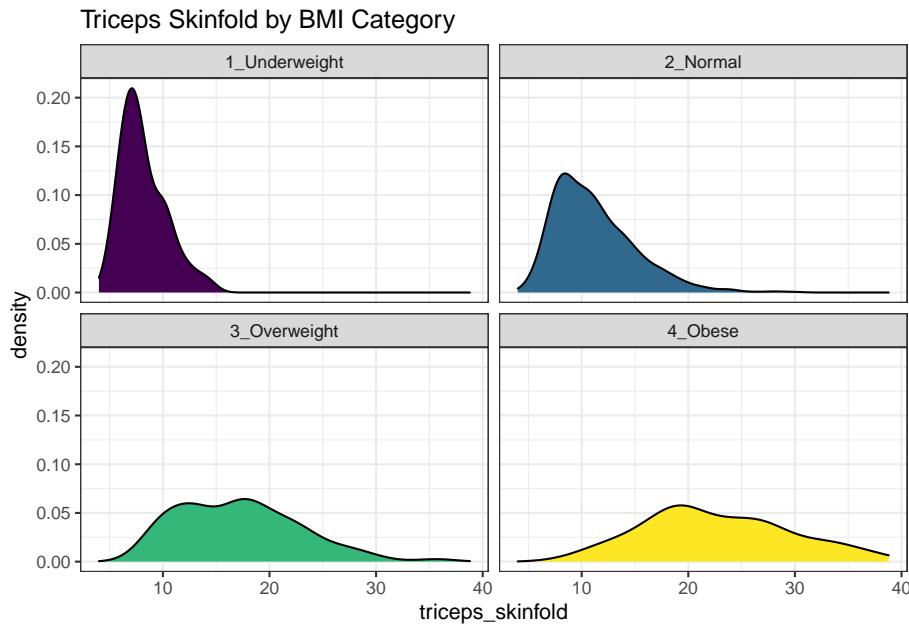
```
nnyfs %>%
  filter(complete.cases(triceps_skinfold, bmi_cat)) %>%
  ggplot(., aes(x=triceps_skinfold, fill = bmi_cat)) +
  geom_histogram(binwidth = 2, color = "black") +
  facet_wrap(~ bmi_cat) +
  scale_fill_viridis_d() +
  guides(fill = "none") +
  labs(title = "Triceps Skinfold by BMI Category")
```



10.5 Using Multiple Density Plots to Make Comparisons

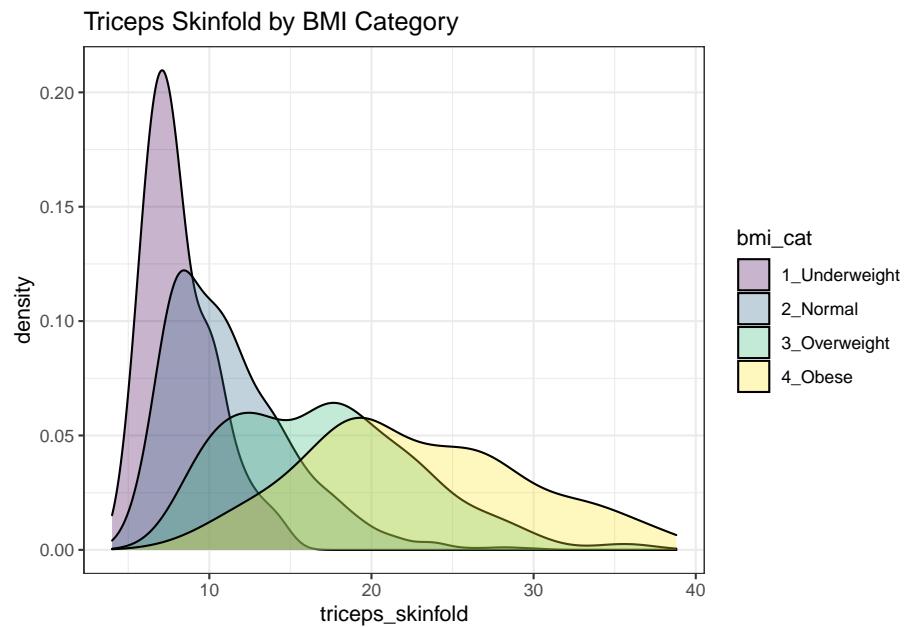
Or, we can make a series of density plots to describe multiple groups of data.

```
nyfs %>%
  filter(complete.cases(triceps_skinfold, bmi_cat)) %>%
  ggplot(., 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")
```



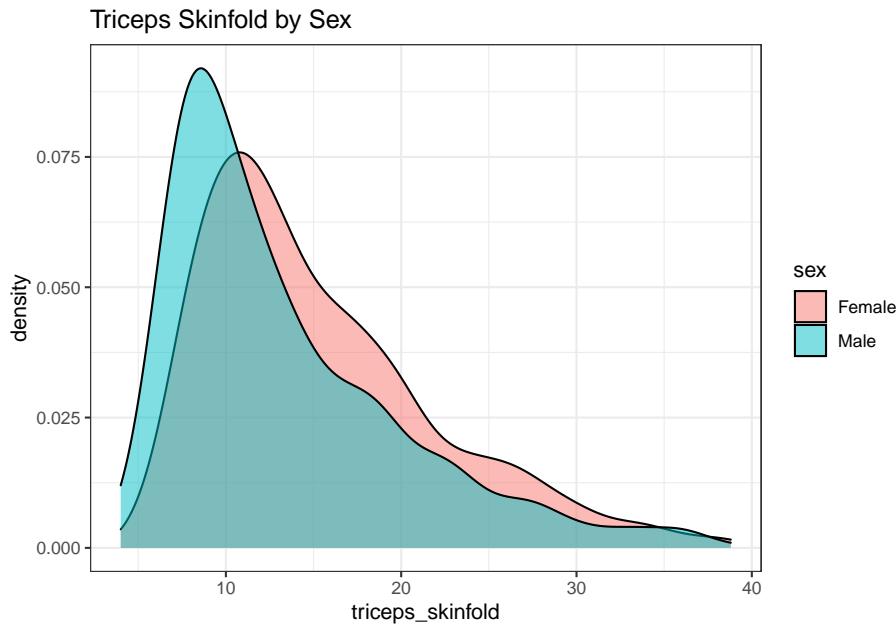
Or, we can plot all of the densities on top of each other with semi-transparent fills.

```
nnys %>%
  filter(complete.cases(triceps_skinfold, bmi_cat)) %>%
  ggplot(., aes(x=triceps_skinfold, fill = bmi_cat)) +
  geom_density(alpha=0.3) +
  scale_fill_viridis_d() +
  labs(title = "Triceps Skinfold by BMI Category")
```



This really works better when we are comparing only two groups, like females to males.

```
nnyfs %>%
  filter(complete.cases(triceps_skinfold, sex)) %>%
  ggplot(., aes(x=triceps_skinfold, fill = sex)) +
  geom_density(alpha=0.5) +
  labs(title = "Triceps Skinfold by Sex")
```

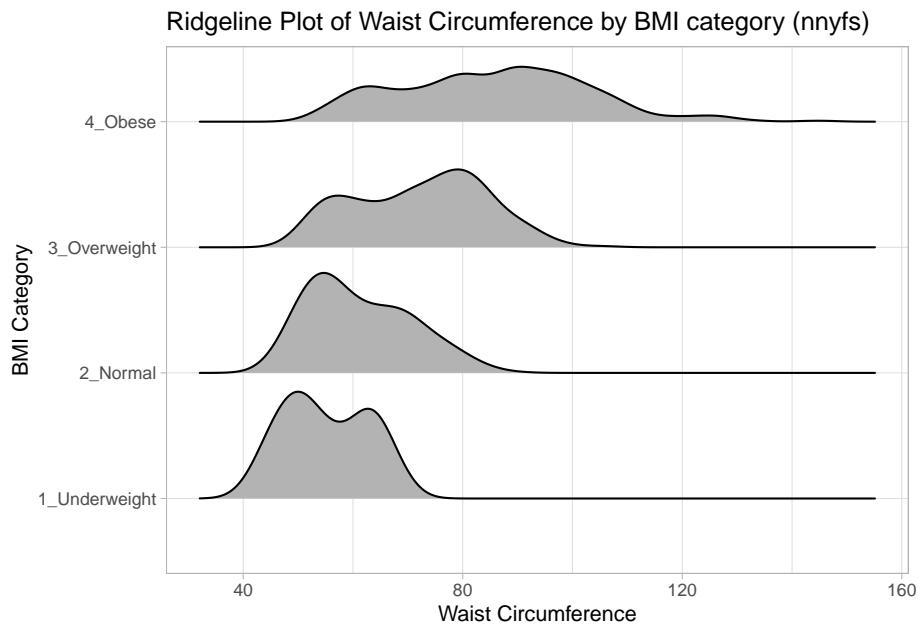


10.6 A Ridgeline Plot

Some people don't like violin plots - for example, see <https://simplystatistics.org/2017/07/13/the-joy-of-no-more-violin-plots/>. A relatively new alternative plot is available. This shows the distribution of several groups simultaneously, especially when you have lots of subgroup categories, and is called a **ridgeline plot**.

```
nnyfs %>%
  filter(complete.cases(waist, bmi_cat)) %>%
  ggplot(., aes(x = waist, y = bmi_cat, height = ..density..)) +
  ggridges::geom_density_ridges(scale = 0.85) +
  theme_light() +
  labs(title = "Ridgeline Plot of Waist Circumference by BMI category (nnyfs)",
       x = "Waist Circumference", y = "BMI Category")
```

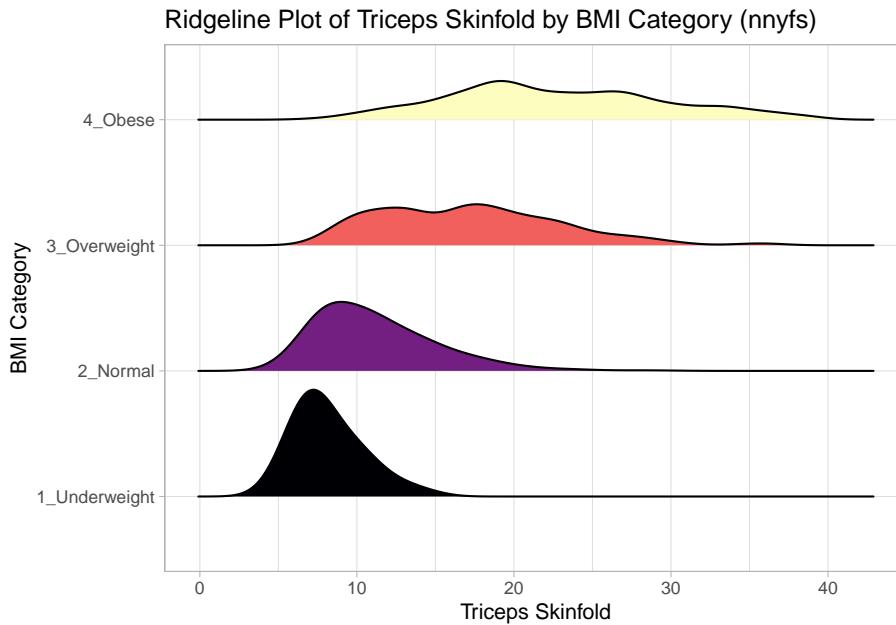
Picking joint bandwidth of 3.47



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 %>%
  filter(complete.cases(bmi_cat, triceps_skinfold)) %>%
  mutate(bmi_cat = fct_reorder(bmi_cat,
                               triceps_skinfold,
                               .fun = median)) %>%
  ggplot(., 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 = FALSE) +
  labs(title = "Ridgeline Plot of Triceps Skinfold by BMI Category (nnyfs)",
       x = "Triceps Skinfold", y = "BMI Category") +
  theme_light()
```

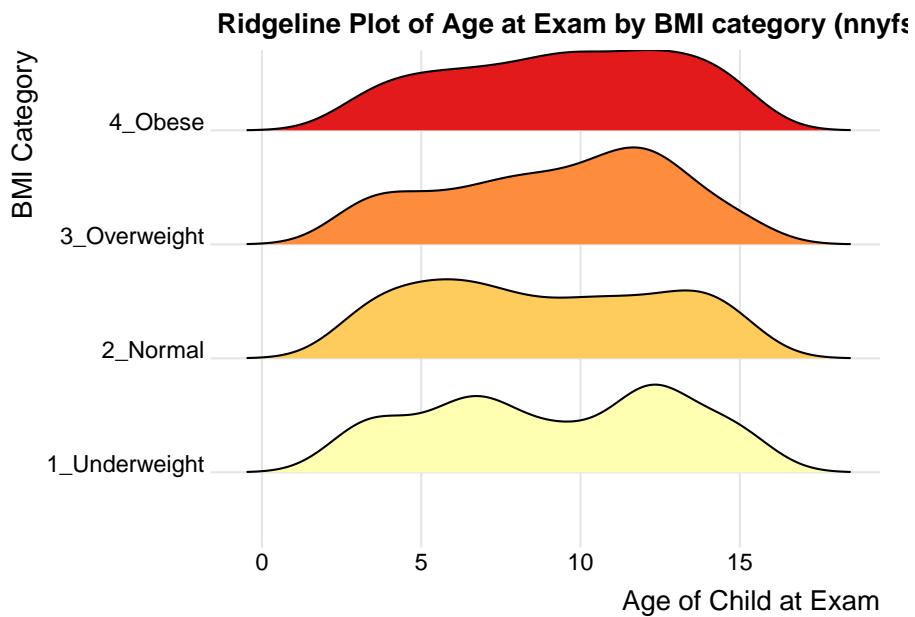
Picking joint bandwidth of 1.37



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 %>%
  filter(complete.cases(bmi_cat, age_child)) %>%
  mutate(bmi_cat = reorder(bmi_cat, age_child, median)) %>%
  ggplot(aes(x = age_child, y = bmi_cat, fill = bmi_cat, height = ..density..)) +
  ggridges::geom_density_ridges(scale = 0.85) +
  scale_fill_brewer(palette = "YlOrRd") +
  guides(fill = FALSE) +
  labs(title = "Ridgeline Plot of Age at Exam by BMI category (nnyfs)",
       x = "Age of Child at Exam", y = "BMI Category") +
  ggridges::theme_ridges()
```

Picking joint bandwidth of 1.15



Chapter 11

Straight Line Models and Correlation

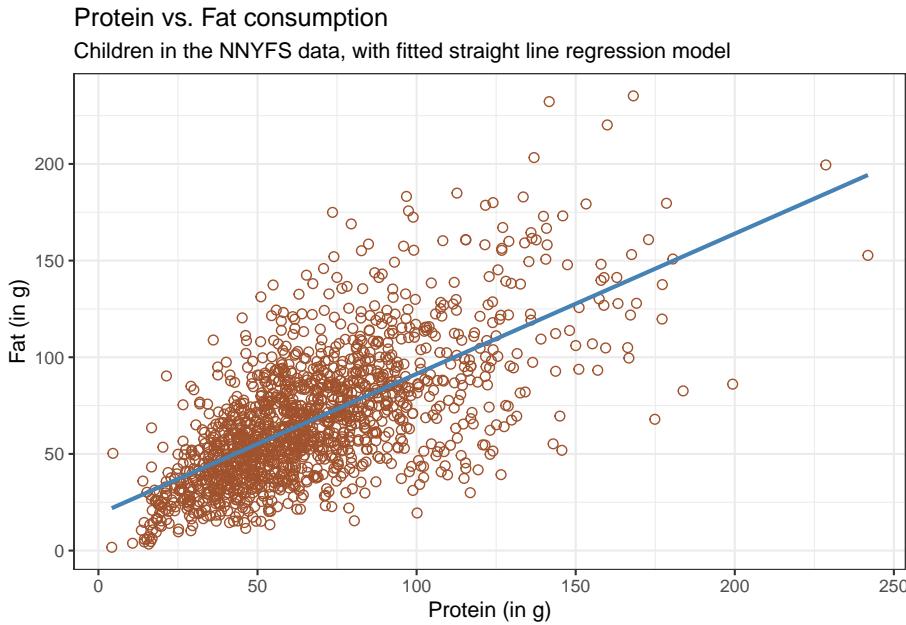
11.1 Assessing A Scatterplot

Let's consider the relationship of `protein` and `fat` consumption for children in the `nnyfs` data.

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

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.

```
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") +
  theme_bw() +
  labs(title = "Protein vs. Fat consumption",
       subtitle = "Children in the NNYFS data, with fitted straight line regression model",
       x = "Protein (in g)", y = "Fat (in g)")
```



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.

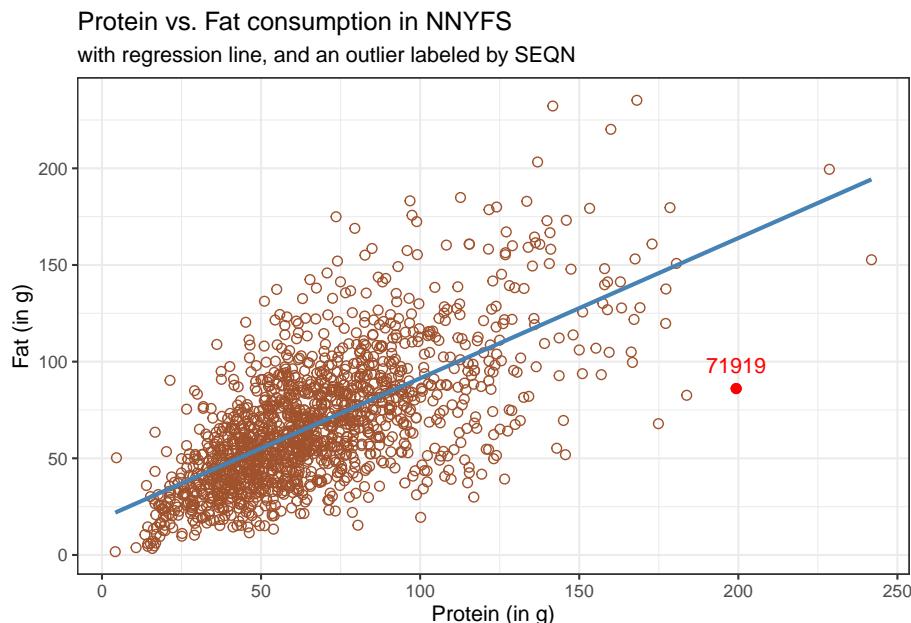
11.1.1 Highlighting an unusual point

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 cre-

ate 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 data frame s1
s1 <- filter(nnyfs, 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") +
  theme_bw() +
  geom_point(data = s1, size = 2, col = "red") +
  geom_text(data = s1, label = s1$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.

```
s1 %>% select(SEQN, sex, race_eth, age_child, protein, fat) %>% kable()
```

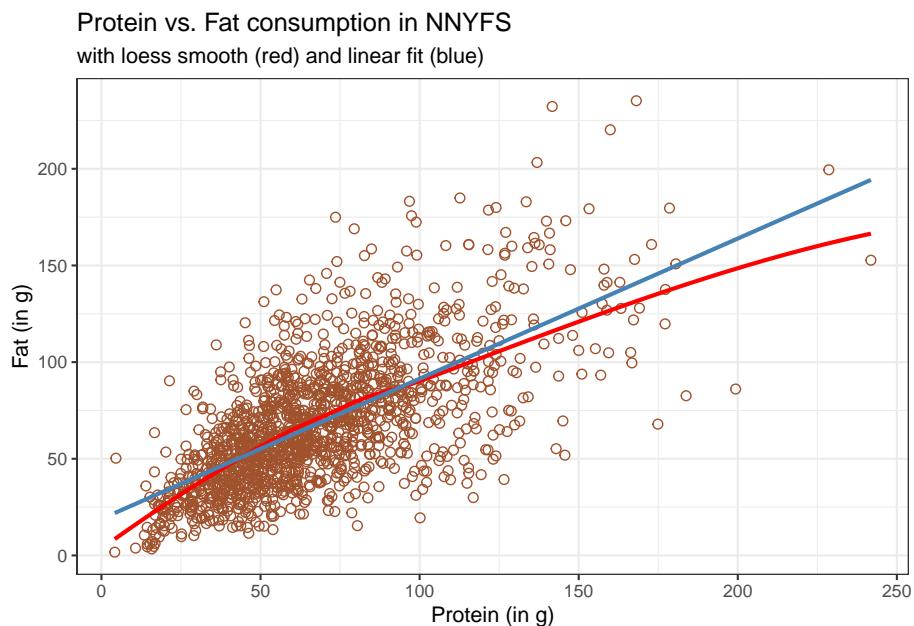
SEQN	sex	race_eth	age_child	protein	fat
71919	Female	2_White Non-Hispanic	14	199.33	86.08

Now, does it seem to you like a straight line model will describe this protein-fat relationship well?

11.1.2 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") +
  theme_bw() +
  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)")
```



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 polyno-

mial to be used. Normally, this is either 1 or 2 - more complex functions are rarely needed for simple scatterplot smoothing.

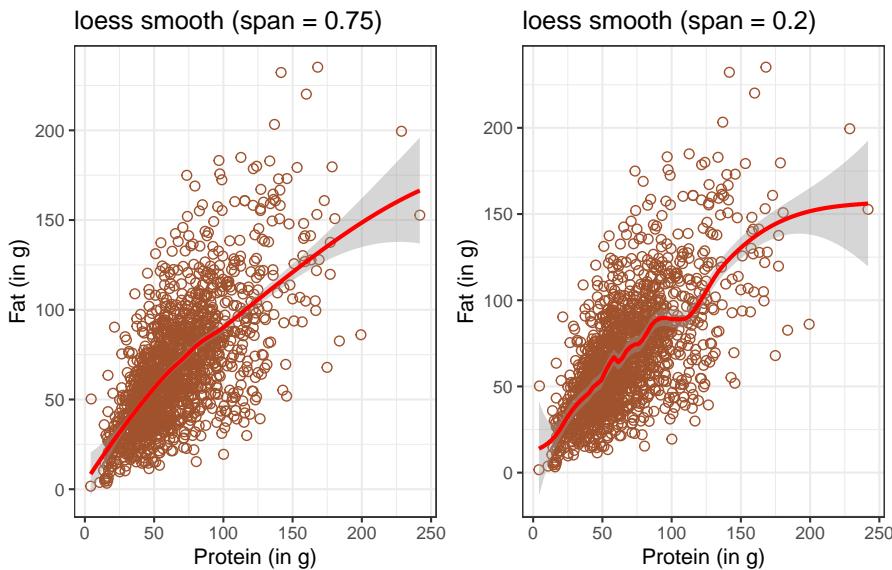
In addition to the curve, smoothing procedures can also provide confidence intervals around their main fitted line. Consider the following plot, which adjusts the span and also adds in the confidence intervals.

```
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) +
  theme_bw() +
  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) +
  theme_bw() +
  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.

11.1.3 What Line Does R Fit?

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

```
lm(fat ~ protein, data = nnyfs)
```

Call:
`lm(formula = fat ~ protein, data = nnyfs)`

Coefficients:
`(Intercept) protein`
`18.8945 0.7251`

So the fitted line is specified as

$$\text{fat} = 18.8945 + 0.7251 \text{ protein}$$

A detailed summary of the fitted linear regression model is also available.

```
summary(lm(fat ~ protein, data = nnyfs))
```

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 ***

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

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) %>%
  kable(digits = 3)
```

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

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) %>%
  kable(digits = 3)
```

r.squared
0.445

We'll spend a lot of time working with these regression summaries in this course.

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 $\text{fat} = 18.9 + 0.725 \text{ protein}$
- 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?

11.2 Correlation Coefficients

Two different 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 %$% cor(fat, protein)
```

```
[1] 0.6671209
```

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.

11.3 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:

$$r_{XY} = \frac{1}{n-1} \sum_{i=1}^n \left(\frac{x_i - \bar{x}}{s_x} \right) \left(\frac{y_i - \bar{y}}{s_y} \right)$$

11.4 Studying Correlation through 6 Examples

The `correx1` data file contains six different sets of (x,y) points, identified by the `set` variable.

```
correx1 <- read_csv("data/correx1.csv")
```

```
Parsed with column specification:
cols(
  set = col_character(),
  x = col_double(),
  y = col_double()
)
summary(correx1)

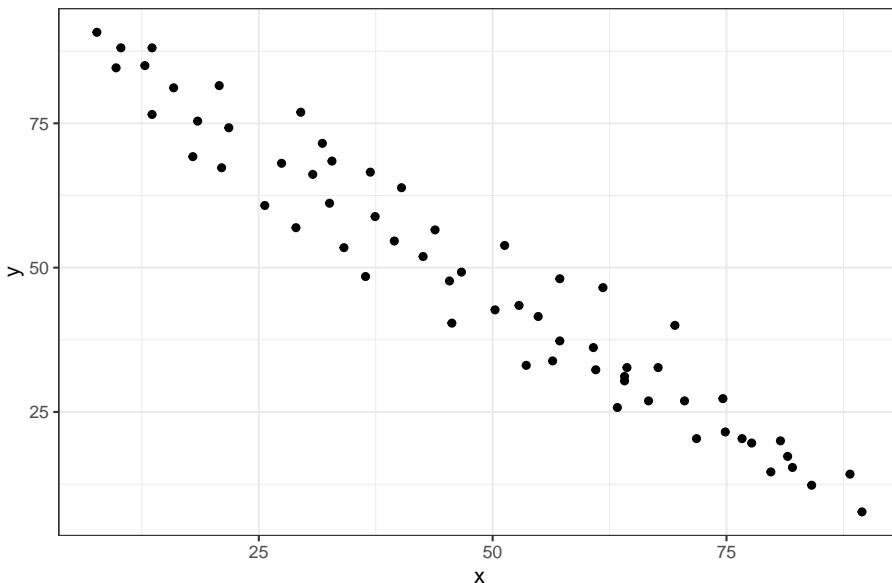
      set              x              y
Length:277    Min.   : 5.897   Min.   : 7.308
Class :character  1st Qu.:29.487  1st Qu.:30.385
Mode  :character  Median :46.154  Median :46.923
                  Mean   :46.529  Mean   :49.061
                  3rd Qu.:63.333  3rd Qu.:68.077
                  Max.   :98.205  Max.   :95.385
```

11.4.1 Data Set Alex

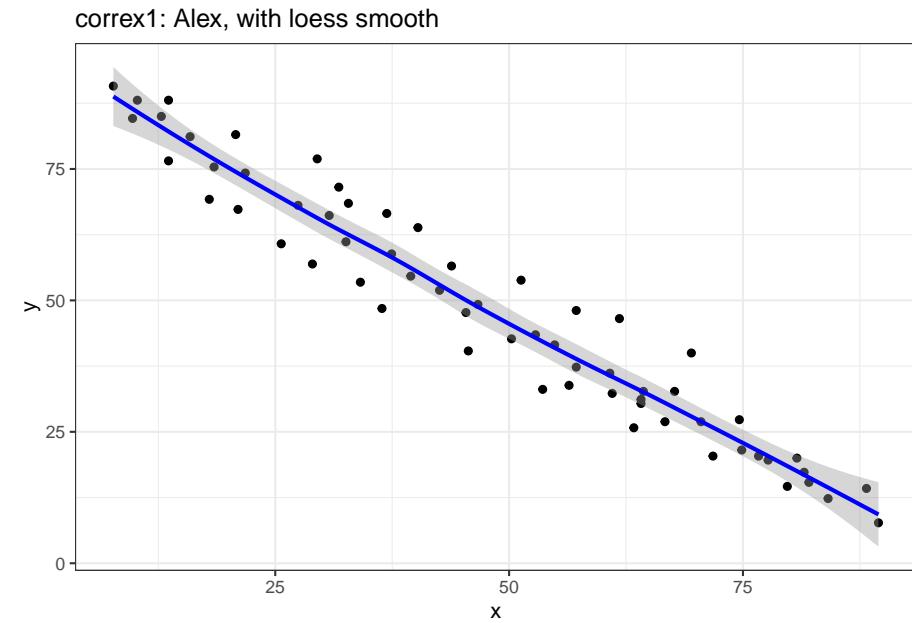
Let's start by working with the **Alex** data set.

```
ggplot(filter(correx1, set == "Alex"), aes(x = x, y = y)) +
  geom_point() +
  labs(title = "correx1: Data Set Alex")
```

correx1: Data Set Alex



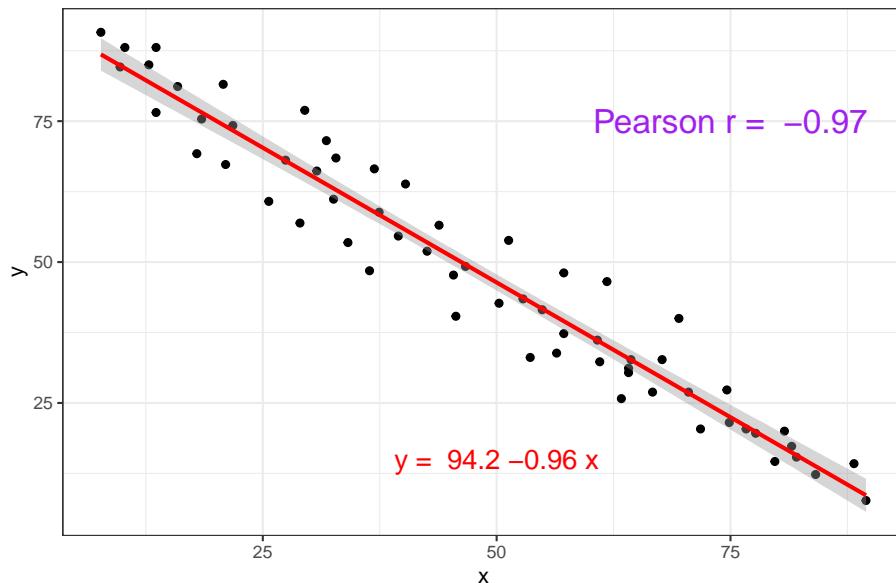
```
ggplot(filter(correx1, set == "Alex"), aes(x = x, y = y)) +
  geom_point() +
  geom_smooth(col = "blue") +
  labs(title = "correx1: Alex, with loess smooth")
```



```
setA <- filter(correx1, set == "Alex")

ggplot(setA, aes(x = x, y = y)) +
  geom_point() +
  geom_smooth(method = "lm", col = "red") +
  labs(title = "correx1: Alex, with Fitted Linear Model") +
  annotate("text", x = 75, y = 75, col = "purple", size = 6,
          label = paste("Pearson r = ", signif(cor(setA$x, setA$y), 3))) +
  annotate("text", x = 50, y = 15, col = "red", size = 5,
          label = paste("y = ", signif(coef(lm(setA$y ~ setA$x))[1], 3),
                        signif(coef(lm(setA$y ~ setA$x))[2], 2), "x"))
```

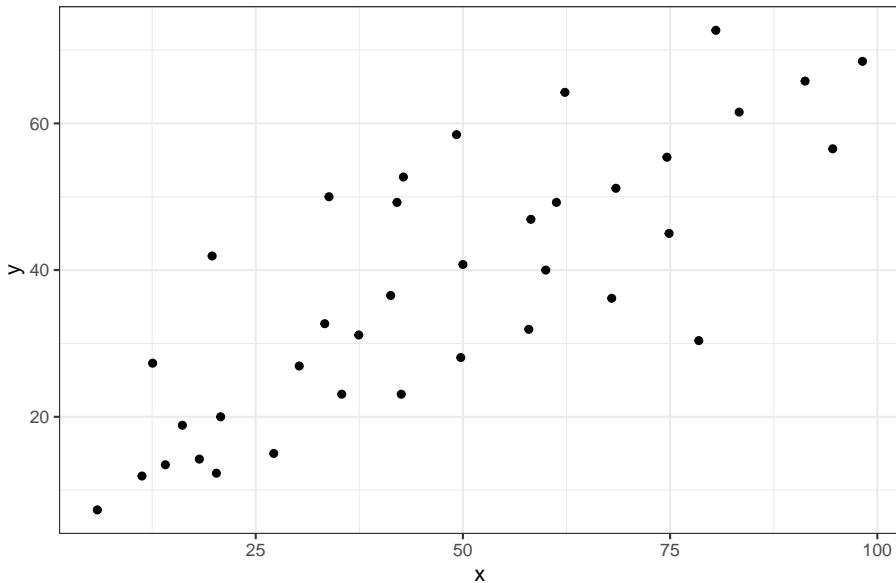
correx1: Alex, with Fitted Linear Model



11.4.2 Data Set Bonnie

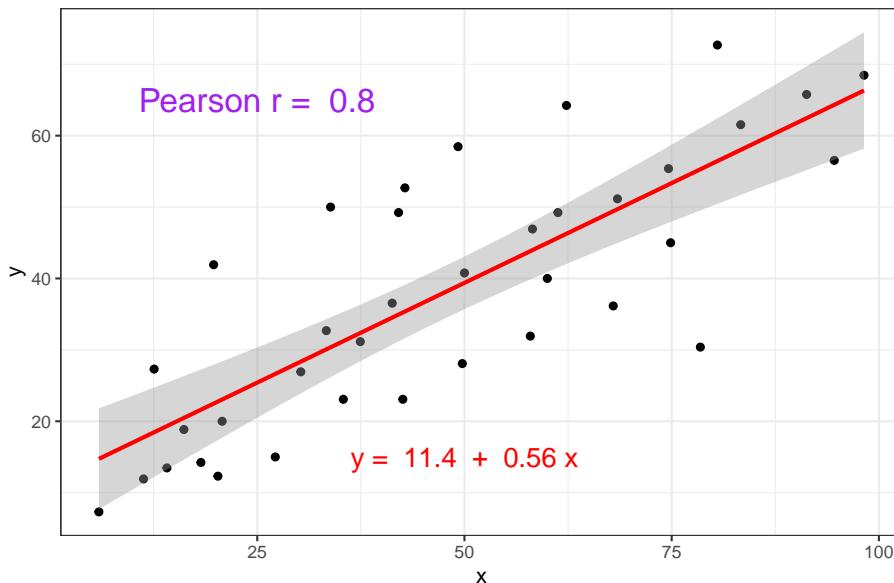
```
setB <- dplyr::filter(correx1, set == "Bonnie")  
  
ggplot(setB, aes(x = x, y = y)) +  
  geom_point() +  
  labs(title = "correx1: Data Set Bonnie")
```

correx1: Data Set Bonnie



```
ggplot(setB, aes(x = x, y = y)) +
  geom_point() +
  geom_smooth(method = "lm", col = "red") +
  labs(title = "correx1: Bonnie, with Fitted Linear Model") +
  annotate("text", x = 25, y = 65, col = "purple", size = 6,
          label = paste("Pearson r = ", signif(cor(setB$x, setB$y), 2))) +
  annotate("text", x = 50, y = 15, col = "red", size = 5,
          label = paste("y = ", signif(coef(lm(setB$y ~ setB$x))[1], 3),
                        " + ",
                        signif(coef(lm(setB$y ~ setB$x))[2], 2), "x"))
```

correx1: Bonnie, with Fitted Linear Model



11.4.3 Correlations for All Six Data Sets in the Correx1 Example

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(cor(x, y, use="complete"),2))

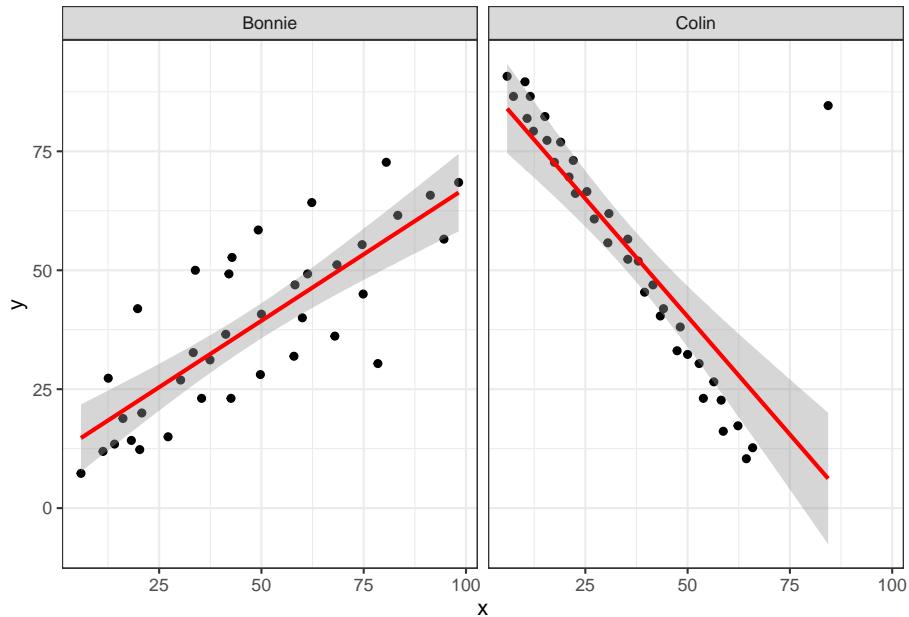
`summarise()` ungrouping output (override with `^.groups` argument)
knitr::kable(tab1)
```

set	Pearson r
Alex	-0.97
Bonnie	0.80
Colin	-0.80
Danielle	0.00
Earl	-0.01
Fiona	0.00

11.4.4 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", col = "red") +  
  facet_wrap(~ set)
```



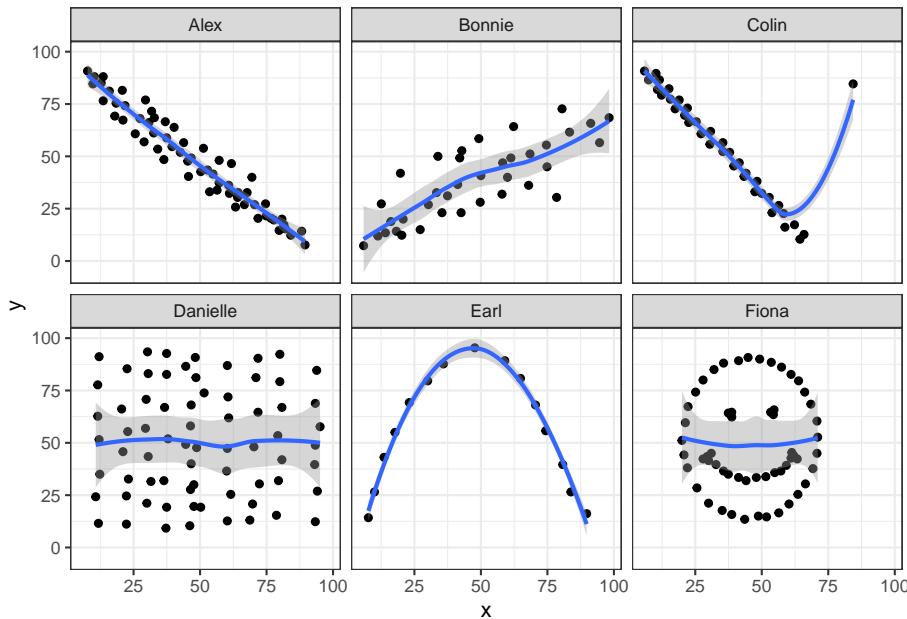
Uh, oh. It looks like the point in Colin at the top right is twisting what would otherwise be a very straight regression model with an extremely strong negative correlation. There's no better way to look for outliers than to examine the scatterplot.

11.4.5 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") +
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.

```
rm(setA, setB, setBC, tab1)
```

11.5 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 <- read_csv("data/correx2.csv")

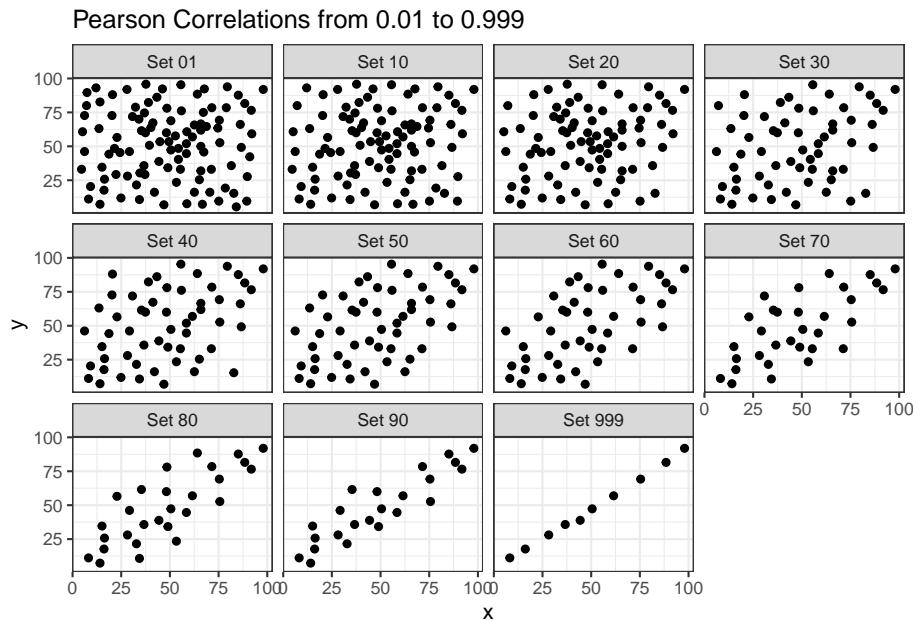
Parsed with column specification:
cols(
  set = col_character(),
  x = col_double(),
  y = col_double(),
  group = col_double()
)
correx2 %>%
  group_by(set) %>%
  summarise(cor = round(cor(x, y, use="complete"), 3))

`summarise()` ungrouping output (override with `.`groups` argument)

# 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).

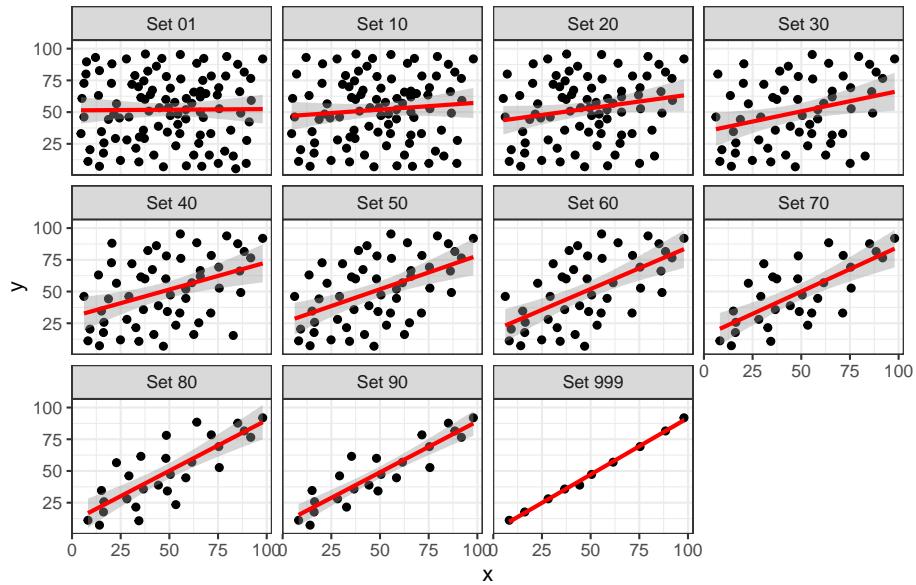
```
ggplot(corrrex2, aes(x = x, y = y)) +
  geom_point() +
  facet_wrap(~ set) +
  labs(title = "Pearson Correlations from 0.01 to 0.999")
```



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.

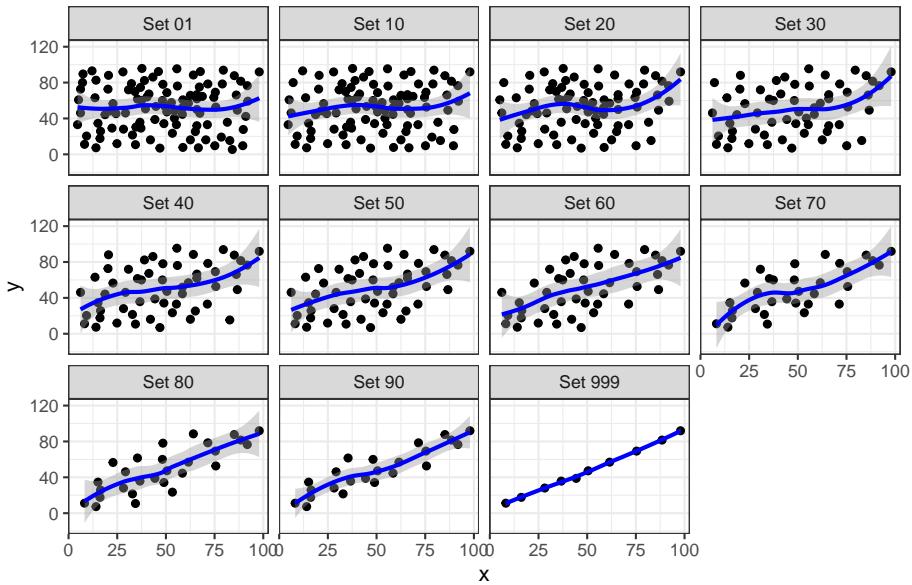
```
ggplot(correx2, aes(x = x, y = y)) +
  geom_point() +
  geom_smooth(method = "lm", col = "red") +
  facet_wrap(~ set) +
  labs(title = "R will fit a straight line to anything.")
```

R will fit a straight line to anything.



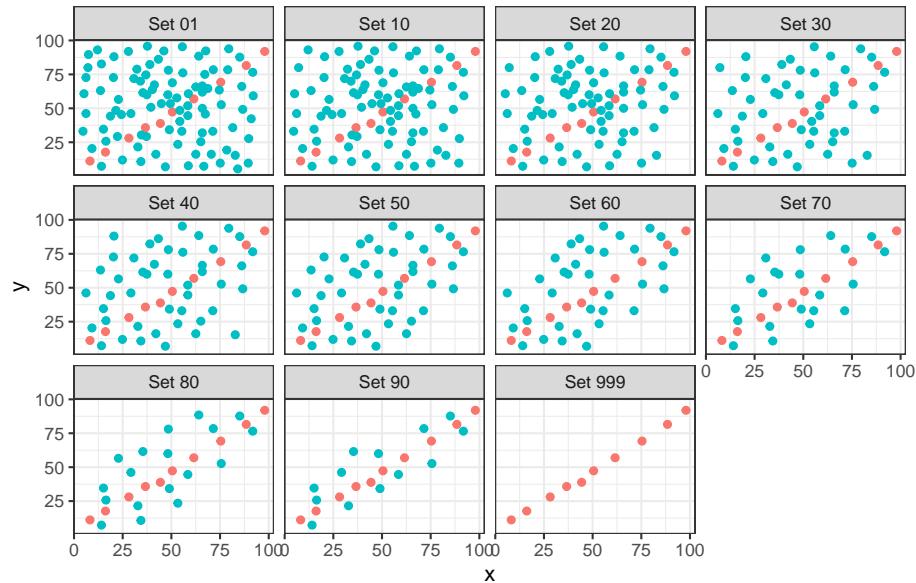
```
ggplot(correx2, aes(x = x, y = y)) +
  geom_point() +
  geom_smooth(col = "blue") +
  facet_wrap(~ set) +
  labs(title = "Even if a loess smooth suggests non-linearity.")
```

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.

11.6 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.

11.6.1 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.

11.6.2 Comparing Pearson and Spearman Correlations

Let's look at the `nnyfs` data again.

```
nnyfs %$% cor(fat, protein)
[1] 0.6671209
nnyfs %$% cor(fat, protein, method = "spearman")
[1] 0.6577489
```

The Spearman and Pearson correlations are not especially different in this case.

11.6.3 Spearman vs. Pearson Example 1

The next few plots describe relationships where we anticipate the Pearson and Spearman correlations might differ in their conclusions.

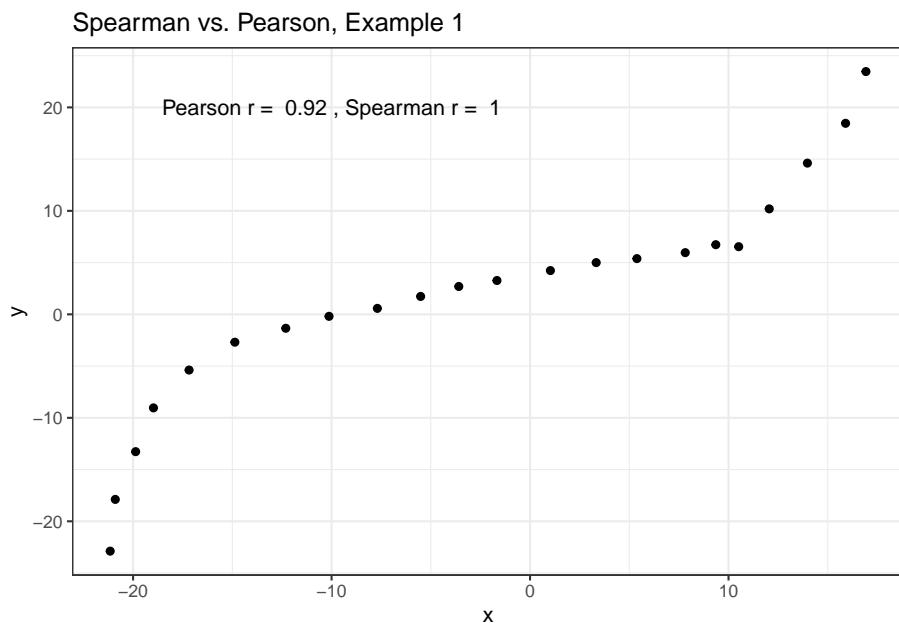
```
spear1 <- read_csv("data/spear1.csv")
Parsed with column specification:
cols(
  x = col_double(),
  y = col_double()
)
spear2 <- read_csv("data/spear2.csv")
Parsed with column specification:
cols(
  x = col_double(),
  y = col_double()
)
spear3 <- read_csv("data/spear3.csv")
Parsed with column specification:
cols(
  x = col_double(),
  y = col_double()
)
```

```
spear4 <- read_csv("data/spear4.csv")

Parsed with column specification:
cols(
  x = col_double(),
  y = col_double()
)
# these are just toy examples with
# two columns per data set and no row numbering
```

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(spear1, aes(x = x, y = y)) +
  geom_point() +
  labs(title = "Spearman vs. Pearson, Example 1") +
  annotate("text", x = -10, y = 20,
           label = paste("Pearson r = ",
                         signif(cor(spear1$x, spear1$y), 2),
                         ", Spearman r = ",
                         signif(cor(spear1$x, spear1$y, method = "spearman"), 2)))
```

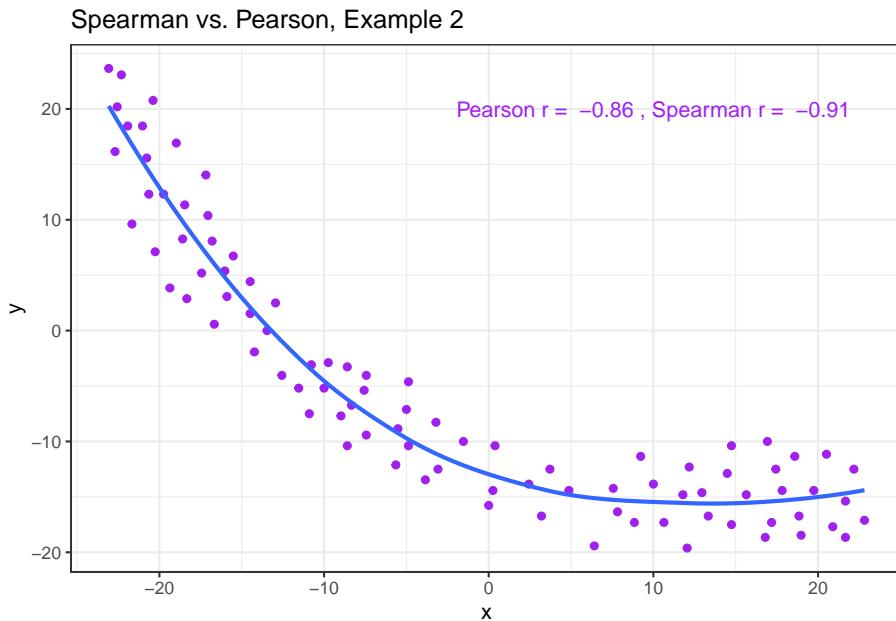


So, a positive Spearman correlation corresponds to an increasing (but not necessarily linear) association between x and y.

11.6.4 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(spear2, aes(x = x, y = y)) +
  geom_point(col = "purple") +
  geom_smooth(method = "loess", se = FALSE) +
  labs(title = "Spearman vs. Pearson, Example 2") +
  annotate("text", x = 10, y = 20, col = "purple",
           label = paste("Pearson r = ",
                         signif(cor(spear2$x, spear2$y), 2),
                         ", Spearman r = ",
                         signif(cor(spear2$x, spear2$y, method = "spearman"), 2)))
```



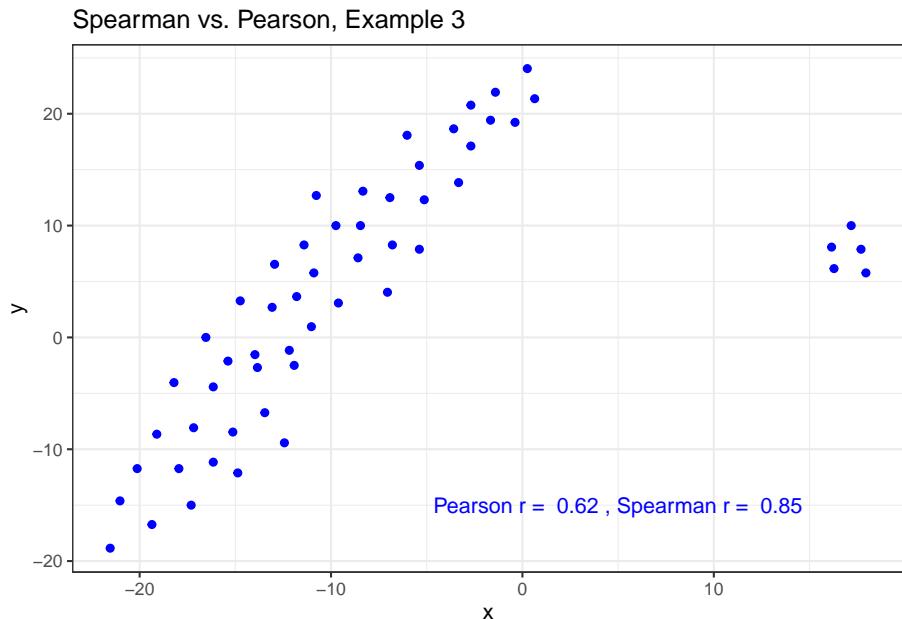
11.6.5 Spearman vs. Pearson Example 3

The Spearman correlation is less sensitive than the Pearson correlation 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(spear3, aes(x = x, y = y)) +
  geom_point(col = "blue") +
```

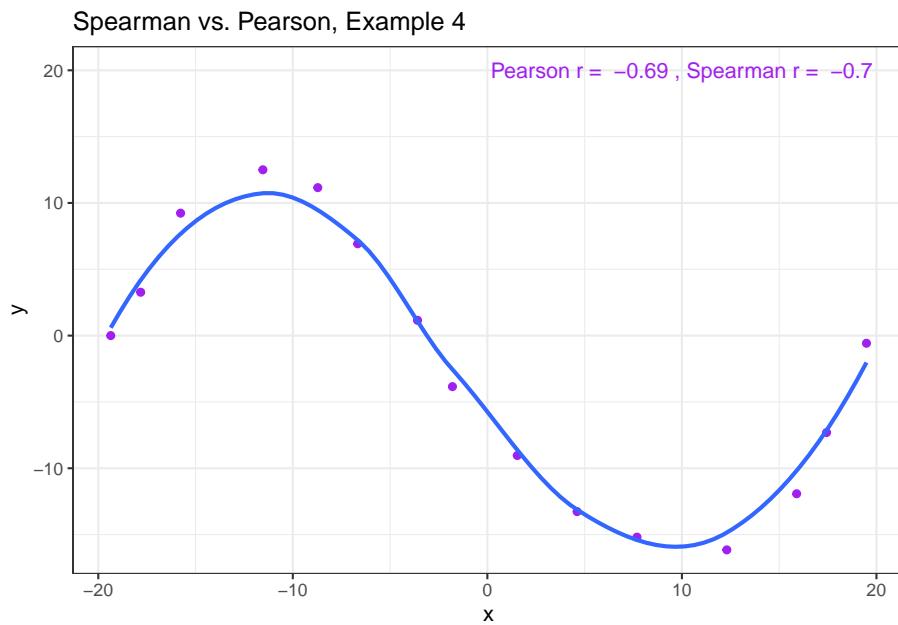
```
labs(title = "Spearman vs. Pearson, Example 3") +
  annotate("text", x = 5, y = -15, col = "blue",
    label = paste("Pearson r = ",
      signif(cor(spear3$x, spear3$y), 2),
      ", Spearman r = ",
      signif(cor(spear3$x, spear3$y, method = "spearman"), 2)))
```



11.6.6 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(spear4, aes(x = x, y = y)) +
  geom_point(col = "purple") +
  geom_smooth(method = "loess", se = FALSE) +
  labs(title = "Spearman vs. Pearson, Example 4") +
  annotate("text", x = 10, y = 20, col = "purple",
    label = paste("Pearson r = ",
      signif(cor(spear4$x, spear4$y), 2),
      ", Spearman r = ",
      signif(cor(spear4$x, spear4$y, method = "spearman"), 2)))
```



Chapter 12

Studying Crab Claws (crabs)

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).

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.

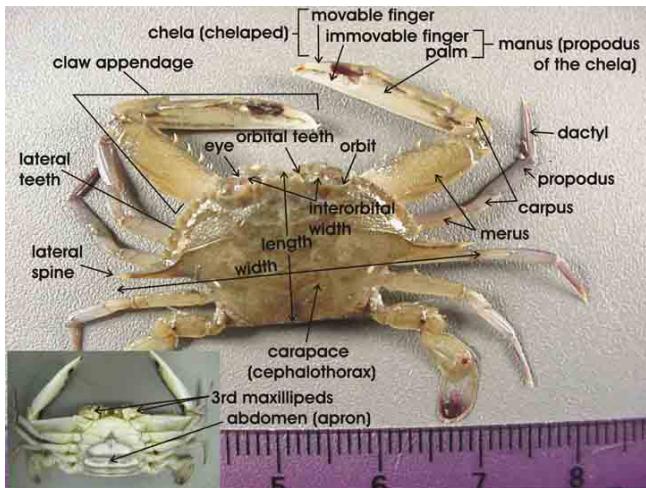


Figure 12.1: Source: <http://txmarspecies.tamug.edu/crustglossary.cfm>

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 *Hemigrapsus 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")

Parsed with column specification:
cols(
  crab = col_double(),
  species = col_character(),
  force = col_double(),
  height = col_double()
)
crabs

# A tibble: 38 x 4
  crab   species      force  height
  <dbl> <chr>        <dbl>   <dbl>
1     1 Hemigrapsus 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 Hemigrapsus nudus     3.2     5
6     6 Hemigrapsus nudus     9.5    7.9
7     7 Cancer productus    22.5    9.4
8     8 Hemigrapsus nudus     7.4    8.3
9     9 Cancer productus    14.6   11.2
10    10 Lophopanopeus bellus   8.7    8.6
# ... with 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
  Hemigrapsus 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))

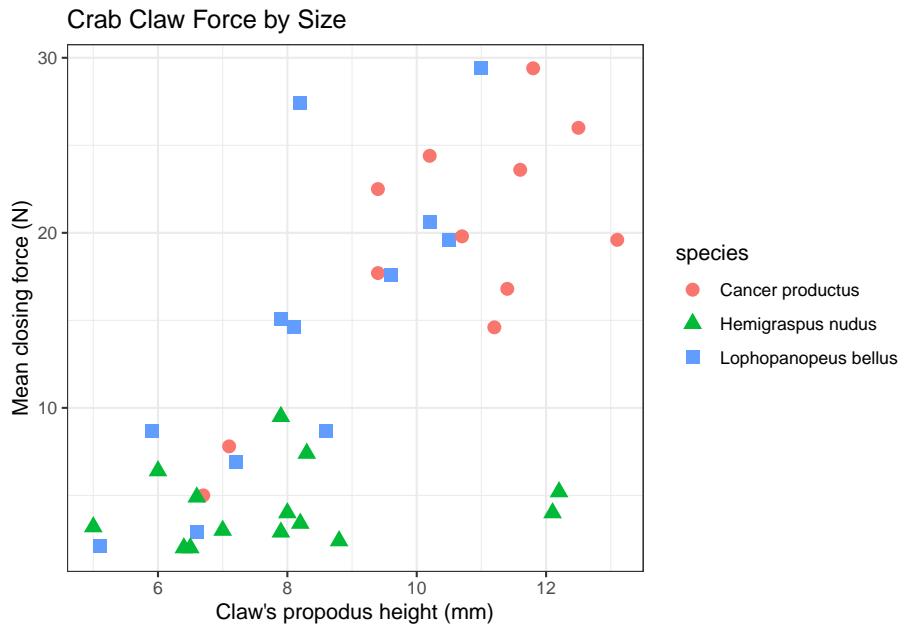
`summarise()` ungrouping output (override with `.`groups` argument)

# A tibble: 3 x 4
  species           n `median(force)` `median(height)`
  <fct>         <int>        <dbl>        <dbl>
1 Cancer productus     12        19.7       11.0
2 Hemigrapsus nudus    14        3.7        7.9
3 Lophopanopeus bellus   12        14.8       8.15
```

12.1 Association of Size and Force

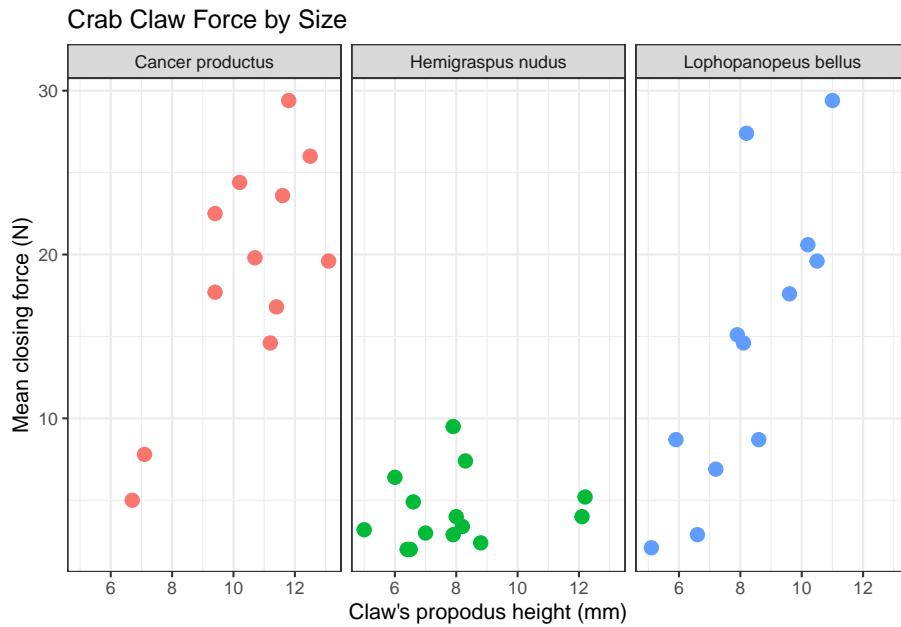
Suppose we want to describe force on the basis of height, across all 38 crabs. We'll add titles and identify the three species of crab, using shape and color.

```
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)") +
  theme_bw()
```



A faceted plot for each species really highlights the difference in force between the *Hemigrapsus 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 = FALSE) +
  labs(title = "Crab Claw Force by Size",
       x = "Claw's propodus height (mm)", y = "Mean closing force (N)") +
  theme_bw()
```

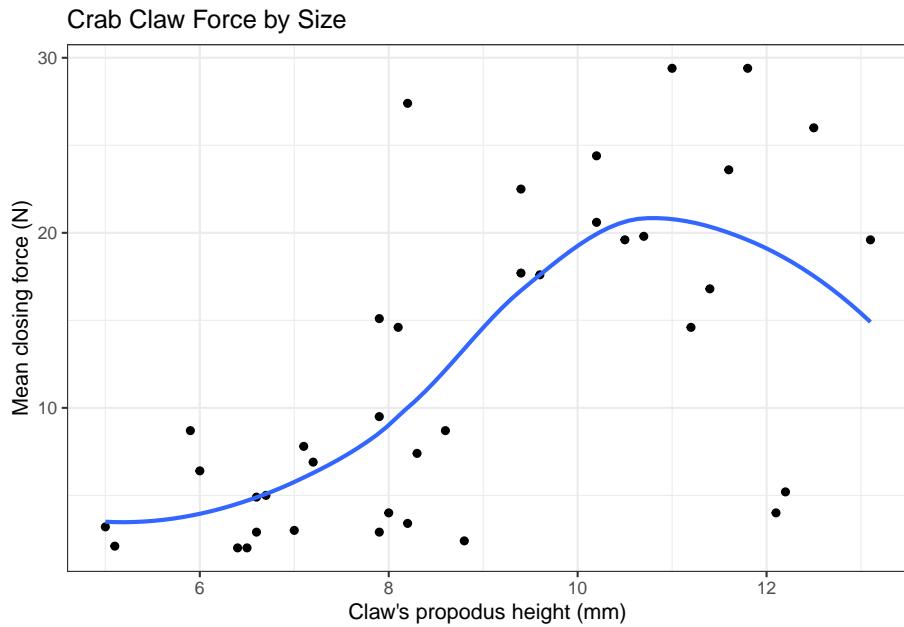


12.2 The loess smooth

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) +
  labs(title = "Crab Claw Force by Size",
       x = "Claw's propodus height (mm)", y = "Mean closing force (N)")

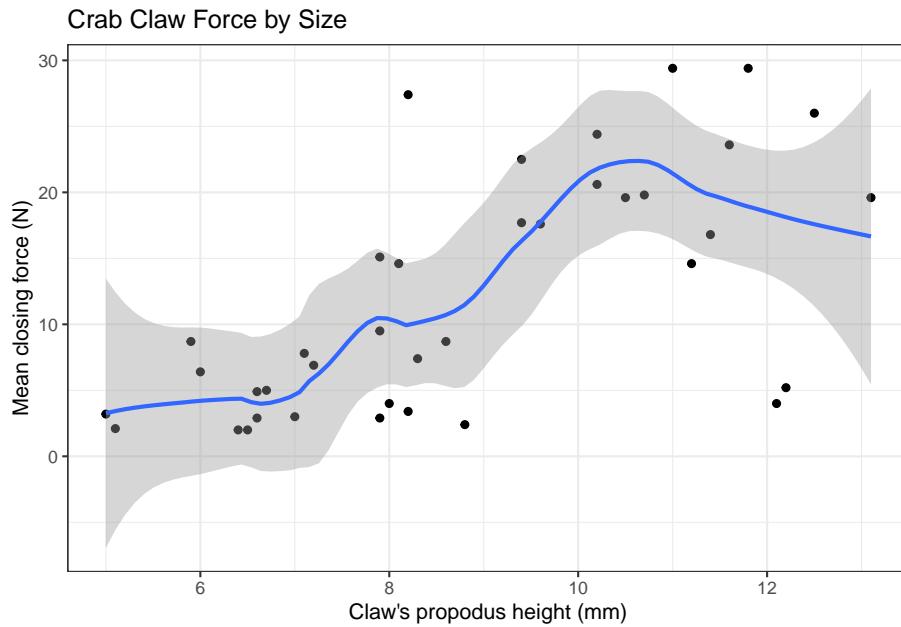
`geom_smooth()` using formula 'y ~ x'
```



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)")
```

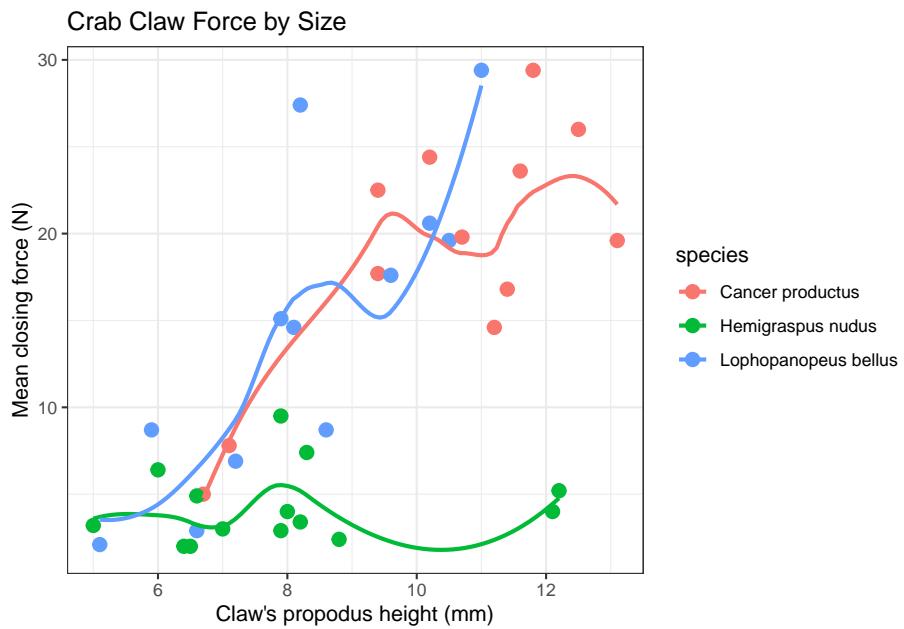


By reducing the size of the span, our resulting picture shows a much less smooth function that we generated previously.

12.2.1 Smoothing within Species

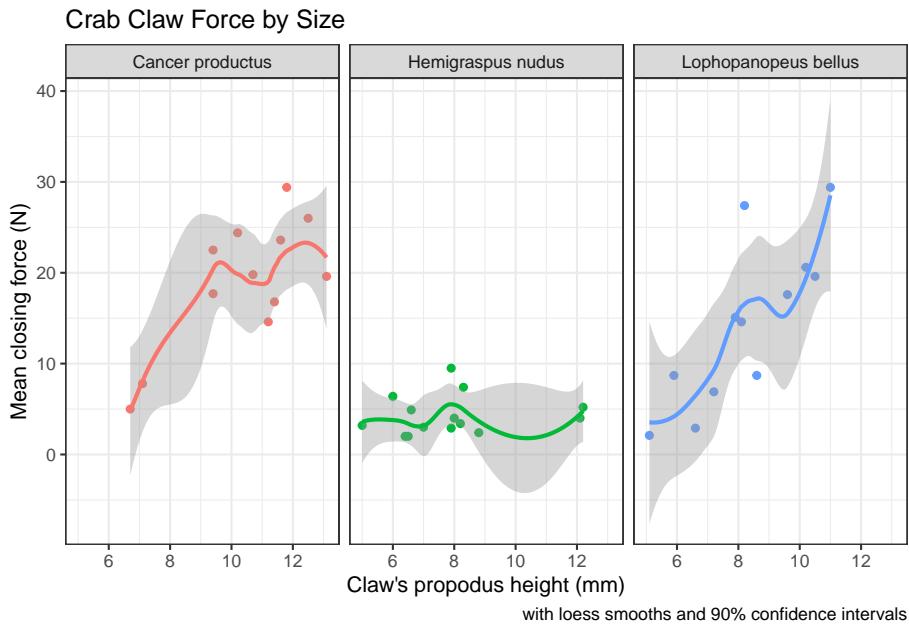
We can, of course, produce the plot above with separate smooths for each of the three species of crab.

```
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)")
```



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 = FALSE) +
  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)
```

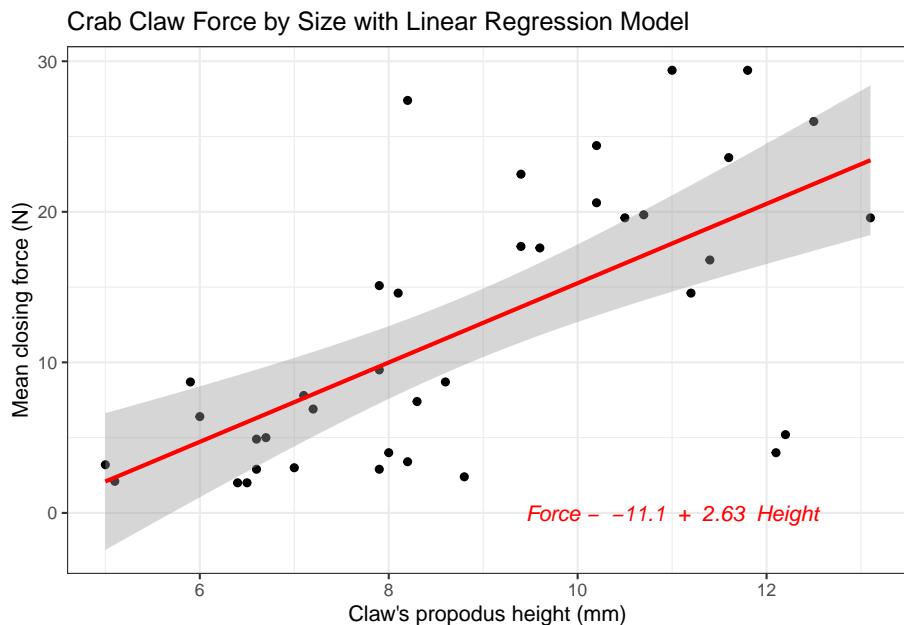


More on these and other confidence intervals later, especially in part B.

12.3 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.



```
rm(mod)
```

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 = $-11.1 + 2.63 \text{ height}$.
- 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 2.63 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.

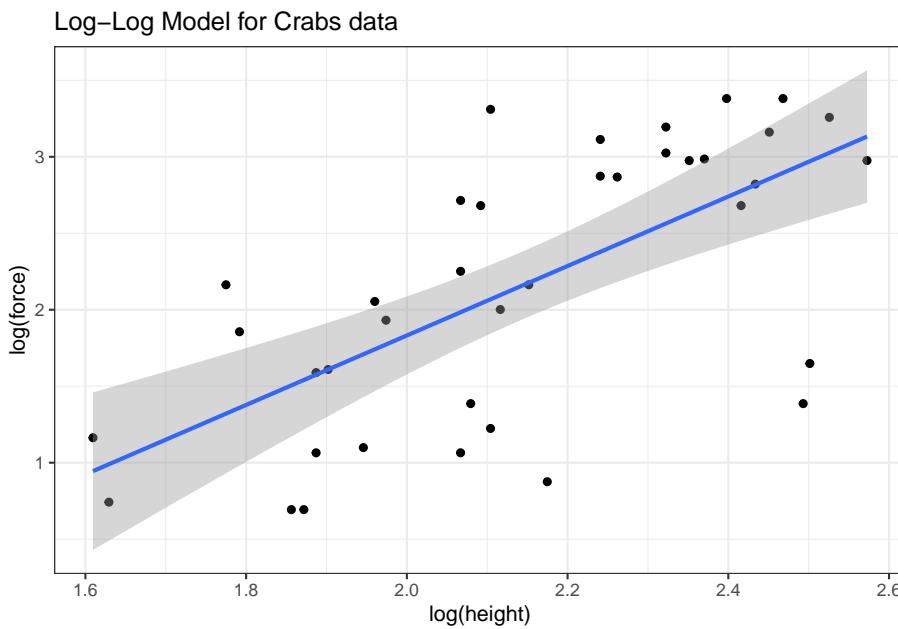
12.4 Is a Linear Model Appropriate?

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")
```



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

12.4.1 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 $\log(\text{force}) = -2.71 + 2.27 \log(\text{height})$.

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...

- $\log(\text{force}) = -2.71 + 2.27 \log(10)$
- $\log(\text{force}) = -2.71 + 2.27 \times 2.3025851$
- $\log(\text{force}) = 2.5190953$
- and so predicted force = $\exp(2.5190953) = 12.4173582$ Newtons, which, naturally, we would round to 12.4 Newtons to match the data set's level of precision.

12.4.2 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 = -11.1 + 2.63 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 15.3 Newtons.

So, it looks like the two models give meaningfully different predictions.

12.5 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.

12.5.1 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.

12.5.2 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 7
`log(force)` `log(height)` .fitted .std.resid .hat .sigma .cooksdi
<dbl>       <dbl>     <dbl>      <dbl>   <dbl>   <dbl>    <dbl>
1       1.39      2.08     2.01     -0.941   0.0280  0.676 1.28e- 2
2       2.71      2.07     1.98      1.10    0.0287  0.673 1.79e- 2
3       1.61      1.90     1.61     -0.000175 0.0499  0.684 8.06e-10
```

```

4      1.06      1.89    1.58   -0.778    0.0530   0.679 1.69e- 2
5      1.16      1.61    0.945   0.349    0.142    0.683 1.01e- 2
6      2.25      2.07    1.98    0.402    0.0287   0.683 2.39e- 3
7      3.11      2.24    2.38    1.11     0.0301   0.673 1.90e- 2
8      2.00      2.12    2.10   -0.142    0.0266   0.684 2.75e- 4
9      2.68      2.42    2.78   -0.146    0.0561   0.684 6.30e- 4
10     2.16      2.15    2.18   -0.0199   0.0263   0.684 5.34e- 6
# ... with 28 more rows

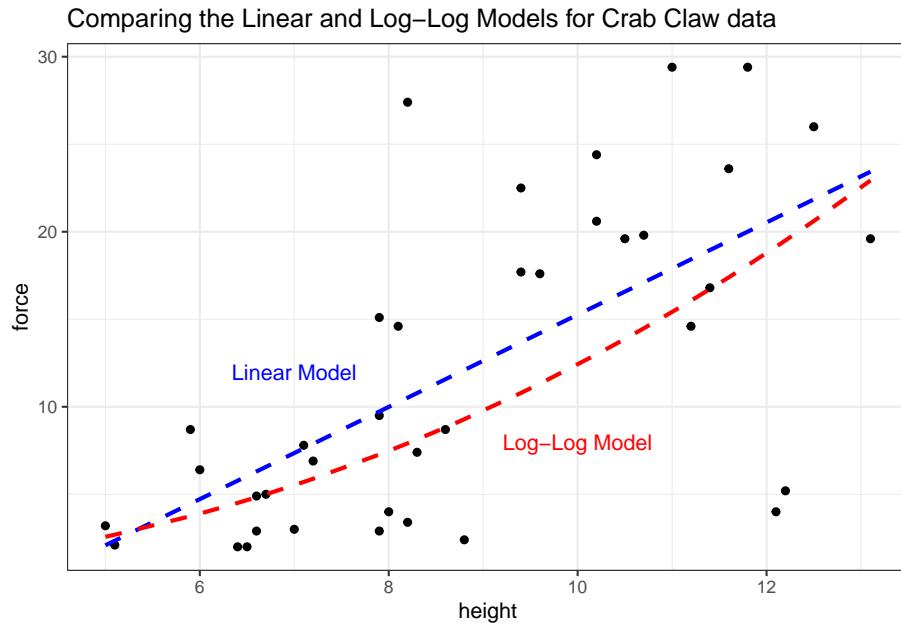
```

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, 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")
`geom_smooth()` using formula 'y ~ x'

```



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.

```
rm(loglogdat, crab_linear, crab_loglog)
```


Chapter 13

The Western Collaborative Group Study

13.1 The Western Collaborative Group Study (`wcgs`) data set

Vittinghoff et al. (2012) explore data from the Western Collaborative Group Study (WCGS) in great detail¹. We'll touch lightly on some key issues in this Chapter.

```
wcgs <- read_csv("data/wcgs.csv")  
  
wcgs  
  
# A tibble: 3,154 x 22  
  id age agec height weight lnwght wghtcat bmi sbp lnsbp dbp chol  
  <dbl> <dbl> <chr> <dbl> <dbl> <dbl> <chr> <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
```

¹For more on the WCGS, you might look at <http://www.epi.umn.edu/cvdepi/study-synopsis/western-collaborative-group-study/>

```
# ... with 3,144 more rows, and 10 more variables: behpat <chr>, dibpat <chr>,
#   smoke <chr>, ncigs <dbl>, arcus <dbl>, chd69 <chr>, typchd69 <dbl>,
#   time169 <dbl>, t1 <dbl>, uni <dbl>
```

Here, we have 3154 rows (subjects) and 22 columns (variables). Since I used `read.csv` to import the data, and then converted to a tibble, all variables containing character data will appear as factors.

13.1.1 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 elements of particular interest.

```
str(wcgs)
```

```
tibble [3,154 x 22] (S3: spec_tbl_df/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    : chr [1:3154] "46-50" "51-55" "56-60" "51-55" ...
$ 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: chr [1:3154] "170-200" "170-200" "170-200" "140-170" ...
$ 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  : chr [1:3154] "A1" "A1" "A1" "A1" ...
$ dibpat  : chr [1:3154] "Type A" "Type A" "Type A" "Type A" ...
$ smoke   : chr [1:3154] "Yes" "Yes" "No" "No" ...
$ 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   : chr [1:3154] "No" "No" "No" "No" ...
$ typchd69: num [1:3154] 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 ...
- attr(*, "spec")=
.. cols(
..   id = col_double(),
..   age = col_double(),
..   agec = col_character(),
..   height = col_double(),
..   weight = col_double(),
```

13.1. THE WESTERN COLLABORATIVE GROUP STUDY (*wcgs*) DATA SET235

```

..   lnwght = col_double(),
..   wghtcat = col_character(),
..   bmi = col_double(),
..   sbp = col_double(),
..   lnsbp = col_double(),
..   dbp = col_double(),
..   chol = col_double(),
..   behpat = col_character(),
..   dibpat = col_character(),
..   smoke = col_character(),
..   ncigs = col_double(),
..   arcus = col_double(),
..   chd69 = col_character(),
..   typchd69 = col_double(),
..   time169 = col_double(),
..   t1 = col_double(),
..   uni = col_double()
.. )

```

13.1.2 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: $703 * \text{weight in lb} / (\text{height in in})^2$
sbp	integer	quantitative	systolic blood pressure, in mm Hg
lnsbp	number	quantitative	natural logarithm of sbp
dbp	integer	quantitative	diastolic blood pressure, mm Hg
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

Name	Stored As	Type	Details (units, levels, etc.)
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)

13.1.3 Quick Summary

```
summary(wcgs)
```

id	age	agec	height
Min. : 2001	Min. :39.00	Length:3154	Min. :60.00
1st Qu.: 3741	1st Qu.:42.00	Class :character	1st Qu.:68.00
Median :11406	Median :45.00	Mode :character	Median :70.00
Mean :10478	Mean :46.28		Mean :69.78
3rd Qu.:13115	3rd Qu.:50.00		3rd Qu.:72.00
Max. :22101	Max. :59.00		Max. :78.00
weight	lnwght	wghtcat	bmi
Min. : 78	Min. :4.357	Length:3154	Min. :11.19
1st Qu.:155	1st Qu.:5.043	Class :character	1st Qu.:22.96
Median :170	Median :5.136	Mode :character	Median :24.39
Mean :170	Mean :5.128		Mean :24.52
3rd Qu.:182	3rd Qu.:5.204		3rd Qu.:25.84
Max. :320	Max. :5.768		Max. :38.95
sbp	lnsbp	dbp	chol
Min. : 98.0	Min. :4.585	Min. : 58.00	Min. :103.0
1st Qu.:120.0	1st Qu.:4.787	1st Qu.: 76.00	1st Qu.:197.2
Median :126.0	Median :4.836	Median : 80.00	Median :223.0
Mean :128.6	Mean :4.850	Mean : 82.02	Mean :226.4
3rd Qu.:136.0	3rd Qu.:4.913	3rd Qu.: 86.00	3rd Qu.:253.0
Max. :230.0	Max. :5.438	Max. :150.00	Max. :645.0
			NA's :12
behpatt	dibpat	smoke	ncigs
Length:3154	Length:3154	Length:3154	Min. : 0.0
Class :character	Class :character	Class :character	1st Qu.: 0.0
Mode :character	Mode :character	Mode :character	Median : 0.0
			Mean :11.6
			3rd Qu.:20.0
			Max. :99.0
arcus	chd69	typchd69	time169

```

Min. :0.0000  Length:3154      Min. :0.0000  Min. : 18
1st Qu.:0.0000  Class :character  1st Qu.:0.0000  1st Qu.:2842
Median :0.0000  Mode  :character  Median :0.0000  Median :2942
Mean   :0.2985          Mean   :0.1363  Mean   :2684
3rd Qu.:1.0000          3rd Qu.:0.0000  3rd Qu.:3037
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::favstats`, etc.

13.2 Are the SBPs Normally Distributed?

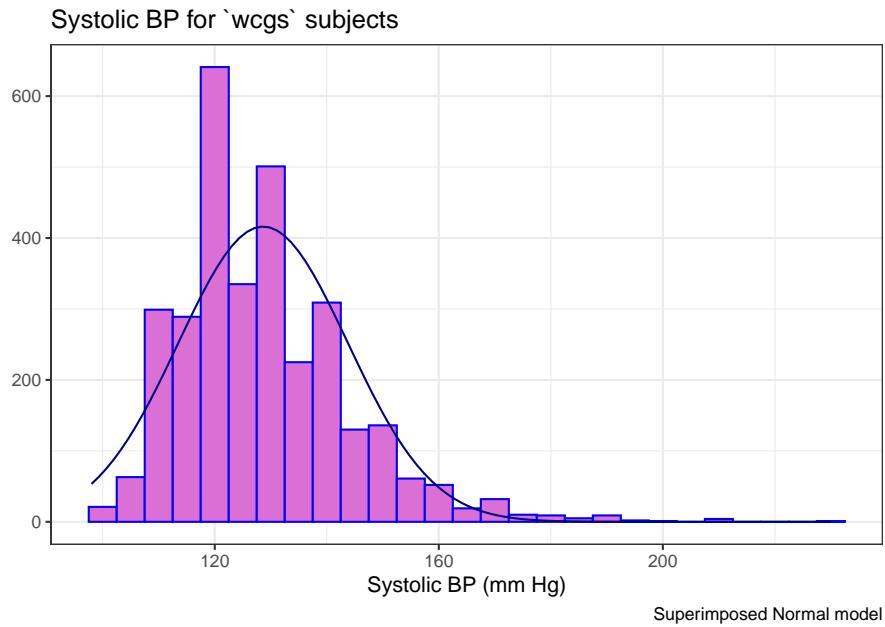
Consider the question of whether the distribution of the systolic blood pressure results is well-approximated by the Normal.

```

res <- mosaic::favstats(~ sbp, data = wcgs)
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")

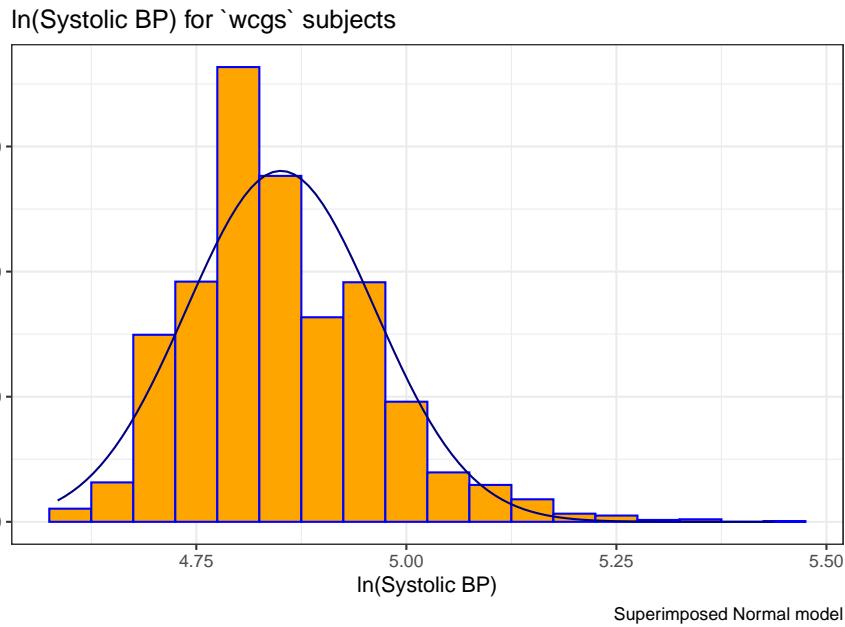
```



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")
```



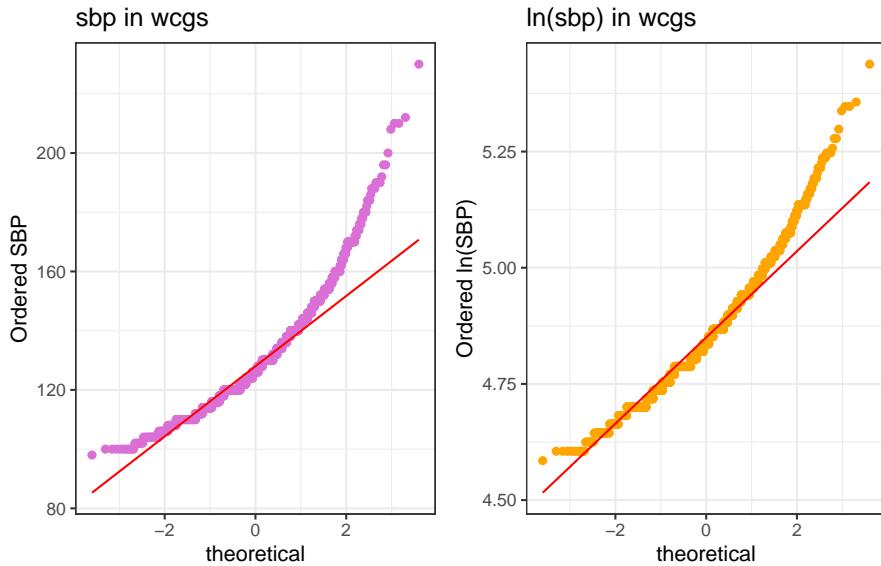
We can also look at Normal Q-Q plots, for instance...

```
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 $\ln(\text{SBP})$ in wcgs

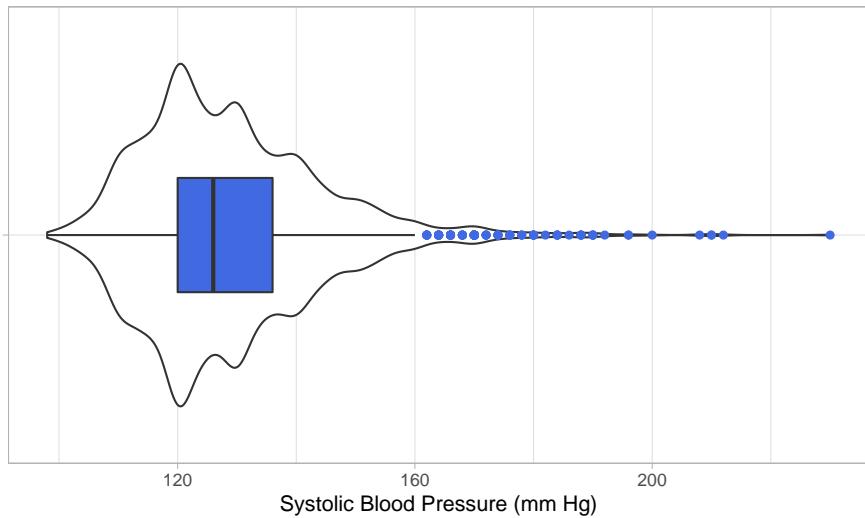
There's at best a small improvement from `sbp` to `lnsbp` in terms of approximation by a Normal distribution.

13.3 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 = "") +
  theme_light() +
  coord_flip()
```

Boxplot with Violin of SBP in `wcgs` data



```
wcgs %$% Hmisc::describe(sbp)
```

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.

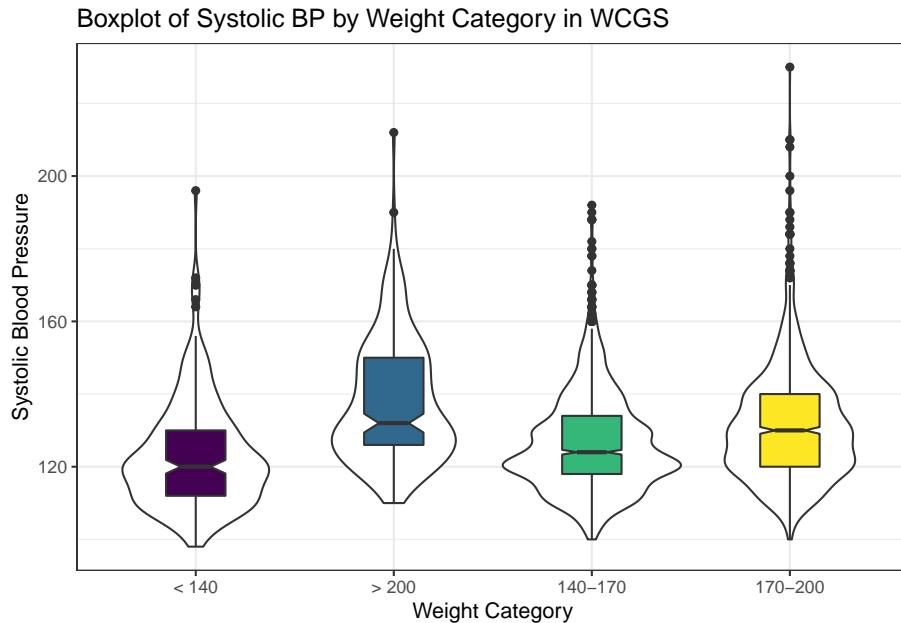
A negative Z score would indicate a point below the mean, while a positive Z score indicates, as we've seen, a point above the mean. The minimum systolic blood pressure, 98 is 2.03 standard deviations *below* the mean, so it has a Z score of -2.

Recall that the Empirical Rule 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?

13.4 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 = FALSE) +
  labs(title = "Boxplot of Systolic BP by Weight Category in WCGS",
       x = "Weight Category", y = "Systolic Blood Pressure")
```



13.5 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
< 140	232	0.07355739
> 200	213	0.06753329
140-170	1538	0.48763475
170-200	1171	0.37127457

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.

```
wcgs <- wcgs %>%
  mutate(weight_f = fct_relevel(wghtcat, "< 140", "140-170", "170-200", "> 200"))

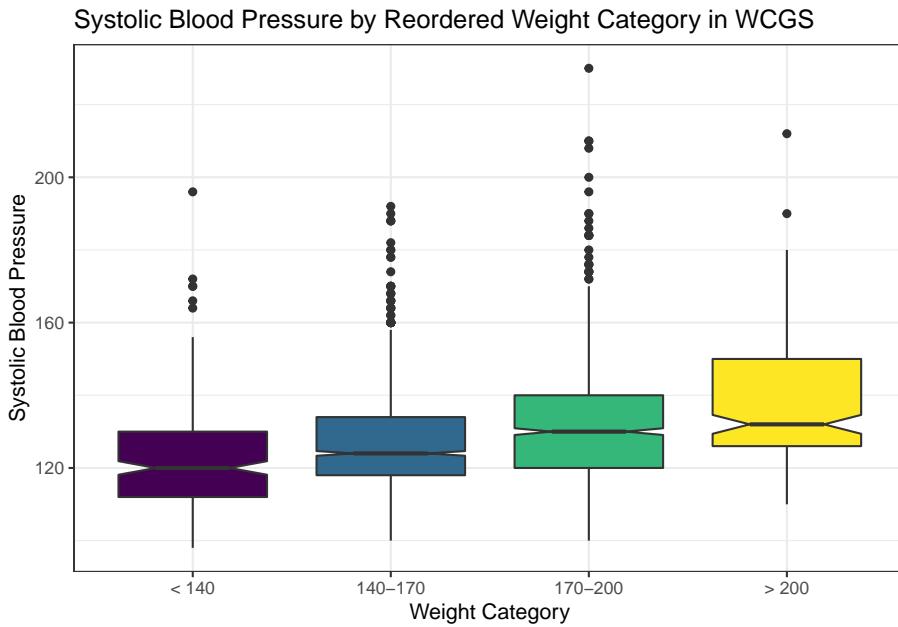
wcgs %>% tabyl(weight_f)
```

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 Grolemund and Wickham (2019), especially the Section on Factors.

13.5.1 SBP by Weight Category

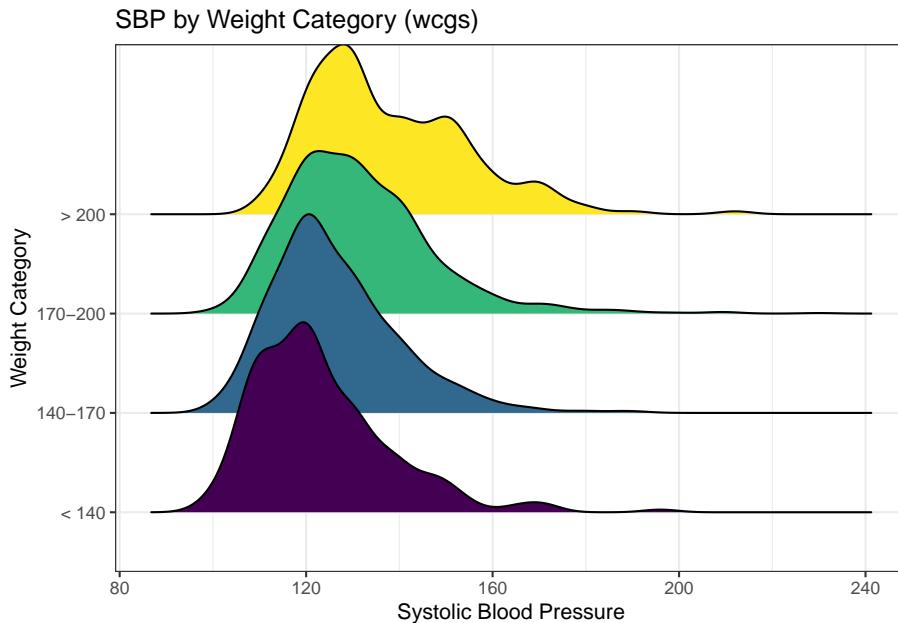
```
ggplot(wcgs, aes(x = weight_f, y = sbp, fill = weight_f)) +
  geom_boxplot(notch = TRUE) +
  scale_fill_viridis_d() +
  guides(fill = FALSE) +
  labs(title = "Systolic Blood Pressure by Reordered Weight Category in WCGS",
       x = "Weight Category", y = "Systolic Blood Pressure")
```



We might see some details well with a **ridgeline plot**, too.

```
wcgs %>%
  ggplot(aes(x = sbp, y = weight_f, fill = weight_f, height = ..density..)) +
  ggridges::geom_density_ridges(scale = 2) +
  scale_fill_viridis_d() +
  guides(fill = FALSE) +
  labs(title = "SBP by Weight Category (wcgs)",
       x = "Systolic Blood Pressure",
       y = "Weight Category") +
  theme_bw()
```

Picking joint bandwidth of 3.74



As the plots suggest, patients in the heavier groups generally had higher systolic blood pressures.

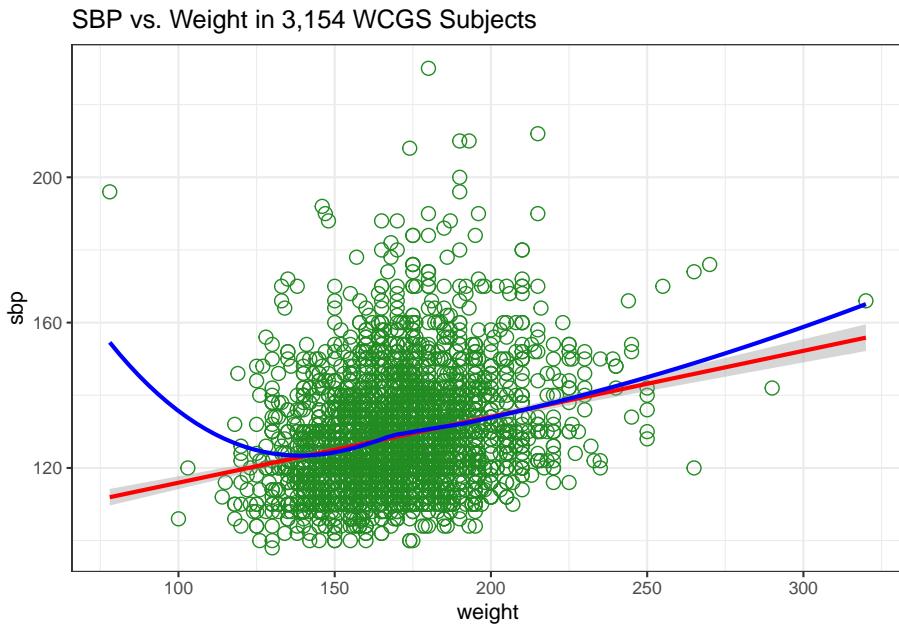
```
mosaic::favstats(sbp ~ weight_f, data = wcgs)
```

	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

13.6 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
  stat_smooth(method=lm, color="red") + ## add se=FALSE to hide conf. interval
  stat_smooth(method=loess, se=FALSE, color="blue") +
  ggtitle("SBP vs. Weight in 3,154 WCGS Subjects") +
  theme_bw()
```



- 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.

13.7 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 arcus senilis and those who do 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 %>%
  filter(complete.cases(arcus_f, sbp, weight)) %>%
  ggplot(aes(x = weight, y = sbp, group = arcus_f)) +
  geom_point(shape = 1) +
  stat_smooth(method=lm, color="red") +
  stat_smooth(method=loess, se=FALSE, color="blue") +
  labs(title = "SBP vs. Weight by Arcus Senilis status",
       caption = "3,152 Western Collaborative Group Study subjects with known arcus senilis status",
       facet_wrap(~ arcus_f) +
  theme_bw()
```



13.8 Linear Model for SBP-Weight Relationship: subjects without Arcus Senilis

```
model.noarcus <-
  lm(sbp ~ weight, data = filter(wcgs, arcus == 0))

tidy(model.noarcus) %>% kable(digits = 2)



| term        | estimate | std.error | statistic | p.value |
|-------------|----------|-----------|-----------|---------|
| (Intercept) | 95.92    | 2.56      | 37.54     | 0       |
| weight      | 0.19     | 0.01      | 12.77     | 0       |



glance(model.noarcus) %>% select(r.squared:p.value) %>% kable(digits = 3)



| r.squared | adj.r.squared | sigma  | statistic | p.value | AIC      |
|-----------|---------------|--------|-----------|---------|----------|
| 0.069     | 0.068         | 14.799 | 162.959   | 0       | 18193.78 |



summary(model.noarcus)
```

Call:
`lm(formula = sbp ~ weight, data = filter(wcgs, arcus == 0))`

Residuals:

Min	1Q	Median	3Q	Max
-----	----	--------	----	-----

13.9. LINEAR MODEL FOR SBP-WEIGHT RELATIONSHIP: SUBJECTS WITH ARCUS SENILIS

```

-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.

13.9 Linear Model for SBP-Weight Relationship: subjects with Arcus Senilis

```

model.witharcus <-
  lm(sbp ~ weight, data = filter(wcgs, arcus == 1))

tidy(model.witharcus) %>% kable(digits = 2)



| term        | estimate | std.error | statistic | p.value |
|-------------|----------|-----------|-----------|---------|
| (Intercept) | 101.88   | 3.76      | 27.13     | 0       |
| weight      | 0.16     | 0.02      | 7.39      | 0       |


glance(model.witharcus) %>% select(r.squared:p.value, AIC) %>% kable(digits = 3)



| r.squared | adj.r.squared | sigma  | statistic | p.value | AIC      |
|-----------|---------------|--------|-----------|---------|----------|
| 0.055     | 0.054         | 14.192 | 54.583    | 0       | 7666.828 |


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
 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 $101.88 - 0.163 \text{ weight}$, for those patients with Arcus Senilis.

13.10 Including Arcus Status in the model

```

model3 <- lm(sbp ~ weight * arcus, data = filter(wcgs, !is.na(arcus)))

tidy(model3) %>% kable(digits = 2)



| 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    |



glance(model3) %>% select(r.squared:p.value, AIC) %>% kable(digits = 3)



| r.squared | adj.r.squared | sigma | statistic | p.value | AIC      |
|-----------|---------------|-------|-----------|---------|----------|
| 0.066     | 0.065         | 14.62 | 74.094    | 0       | 25860.96 |



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 of weight and that 1/0 indicator.

- 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 $SBP = 95.92 + 0.19 \text{ weight} + 5.96 (1) - 0.028 \text{ weight} (1) = 101.88 + 0.162 \text{ weight}$.
 - For a patient without arcus senilis, the regression equation is $SBP = 95.92 + 0.19 \text{ weight} + 5.96 (0) - 0.028 \text{ weight} (0) = 95.92 + 0.19 \text{ weight}$.

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%.

13.11 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 = $95.9 + 0.19 \times \text{weight}$
- *With* Arcus Senilis, linear model for SBP = $101.9 + 0.16 \times \text{weight}$

So the predictions for a 175 pound subject are:

- $95.9 + 0.19 \times 175 = 129 \text{ mm Hg}$ without Arcus Senilis, and
- $101.9 + 0.16 \times 175 = 130 \text{ mm Hg}$ with Arcus Senilis.

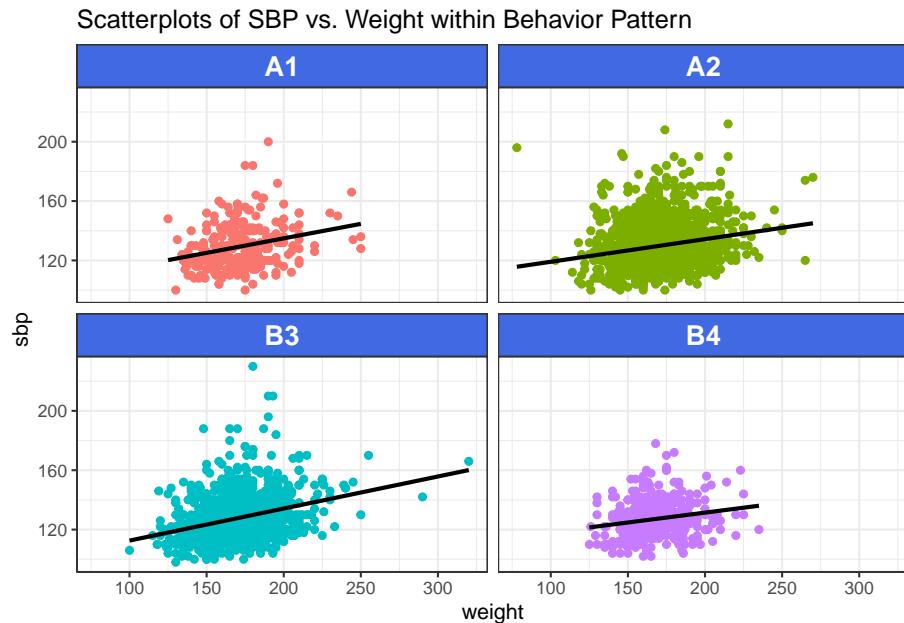
And thus, the predictions for a 200 pound subject are:

- $95.9 + 0.19 \times 200 = 134 \text{ mm Hg}$ without Arcus Senilis, and
- $101.9 + 0.16 \times 200 = 134.4 \text{ mm Hg}$ with Arcus Senilis.

13.12 Scatterplots with Facets Across a Categorical Variable

We can use facets in `ggplot2` to show scatterplots across the levels of a categorical variable, like `behpatt`.

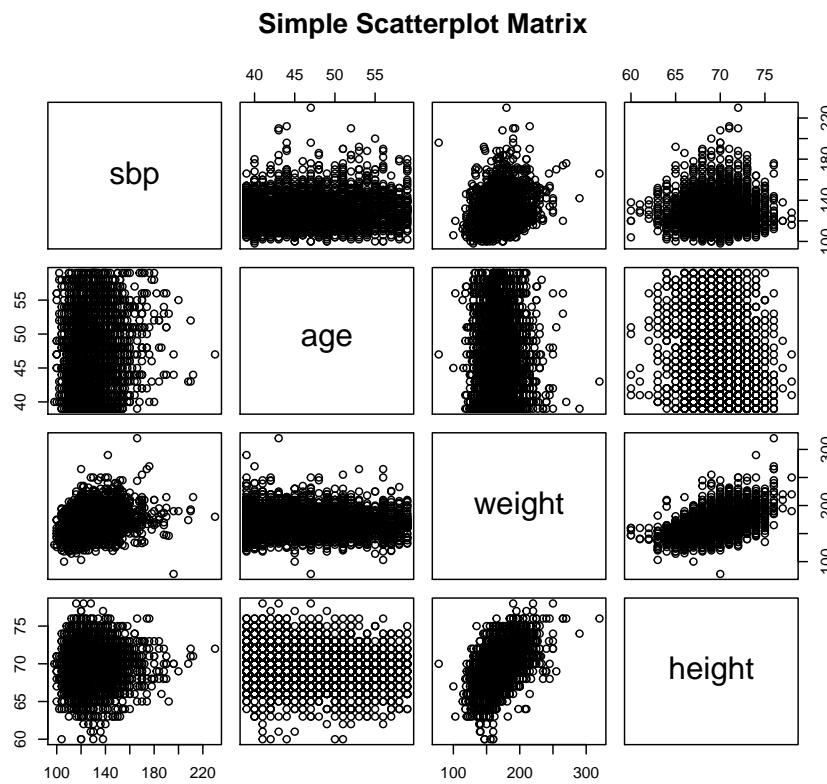
```
ggplot(wcgs, aes(x = weight, y = sbp, col = behpatt)) +
  geom_point() +
  facet_wrap(~ behpatt) +
  geom_smooth(method = "lm", se = FALSE, col = "black") +
  guides(color = FALSE) +
  theme(strip.text = element_text(face="bold", size=rel(1.25), color="white"),
        strip.background = element_rect(fill="royalblue")) +
  labs(title = "Scatterplots of SBP vs. Weight within Behavior Pattern")
```



13.13 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")
```



13.13.1 Displaying a Correlation Matrix

```
wcgs %>%
  dplyr::select(sbp, age, weight, height) %>%
  cor() %>% # obtain correlation coefficients for this subgroup
  signif(., 3) # round them off to three significant figures before printing
```

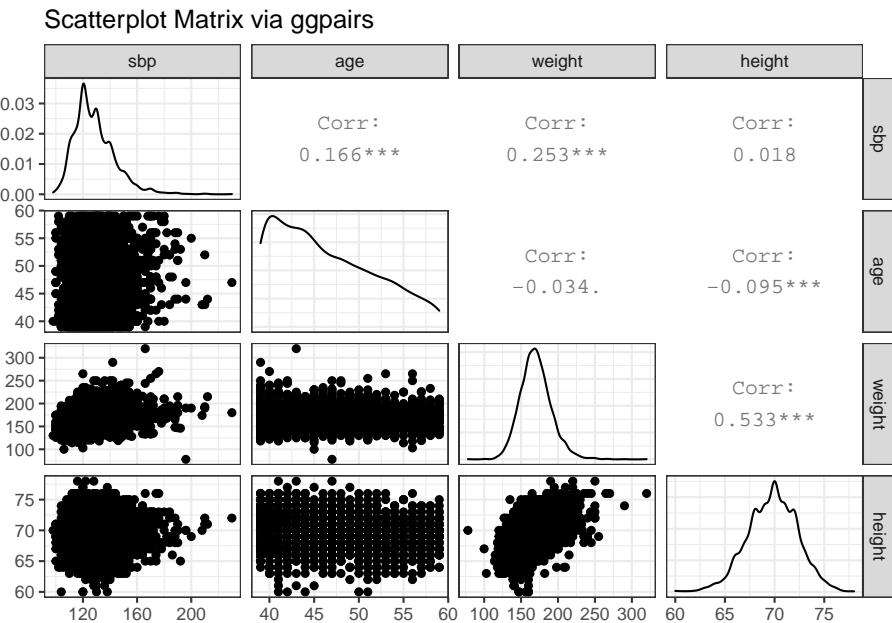
	sbp	age	weight	height
sbp	1.0000	0.1660	0.2530	0.0184
age	0.1660	1.0000	-0.0344	-0.0954
weight	0.2530	-0.0344	1.0000	0.5330
height	0.0184	-0.0954	0.5330	1.0000

13.13.2 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.

```
GGally::ggpairs(wcgs %>% select(sbp, age, weight, height),
                 title = "Scatterplot Matrix via ggpairs")
```



Chapter 14

Re-Expression, Tukey’s Ladder & Box-Cox Plot

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 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 A New Tool: 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.

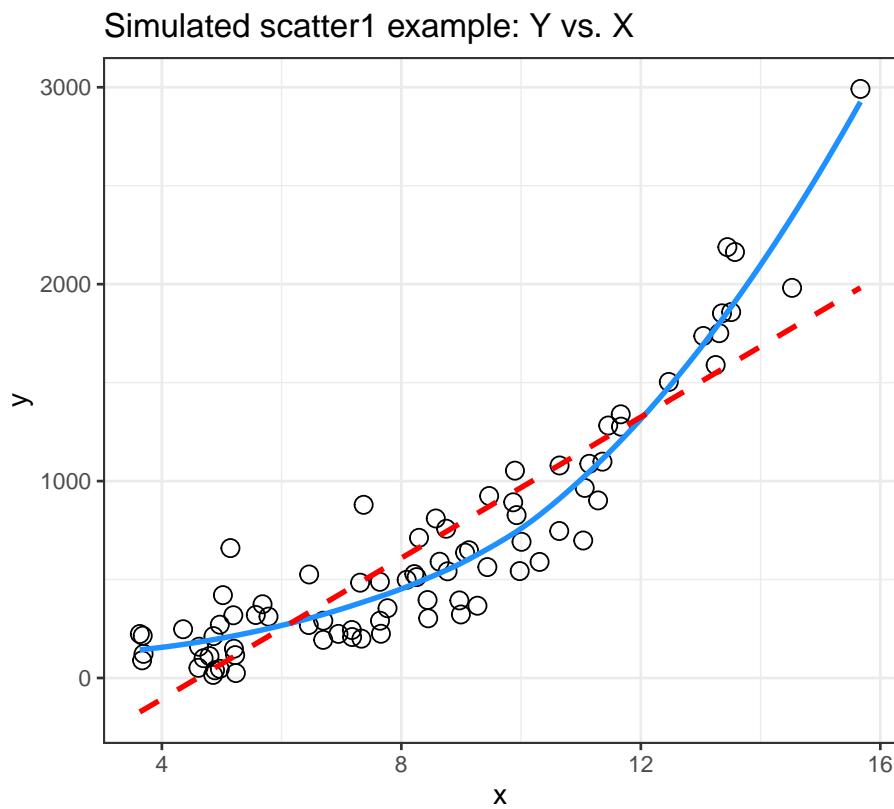
14.2.1 A Few Caveats

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.

14.3 A Simulated Example

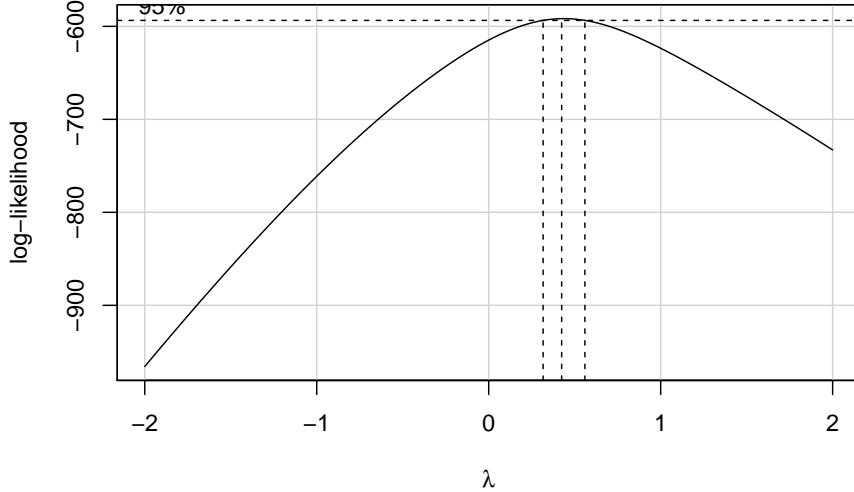
```
set.seed(999); x.rand <- rbeta(80, 2, 5) * 20 + 3
set.seed(1000); y.rand <- abs(50 + 0.75*x.rand^(3) - 0.65*x.rand + rnorm(80, 0, 200))
scatter1 <- data.frame(x = x.rand, y = y.rand) %>% tbl_df
rm(x.rand, y.rand)

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")
```



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.

```
library(car)
boxCox(scatter1$y ~ scatter1$x)
```



```
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")

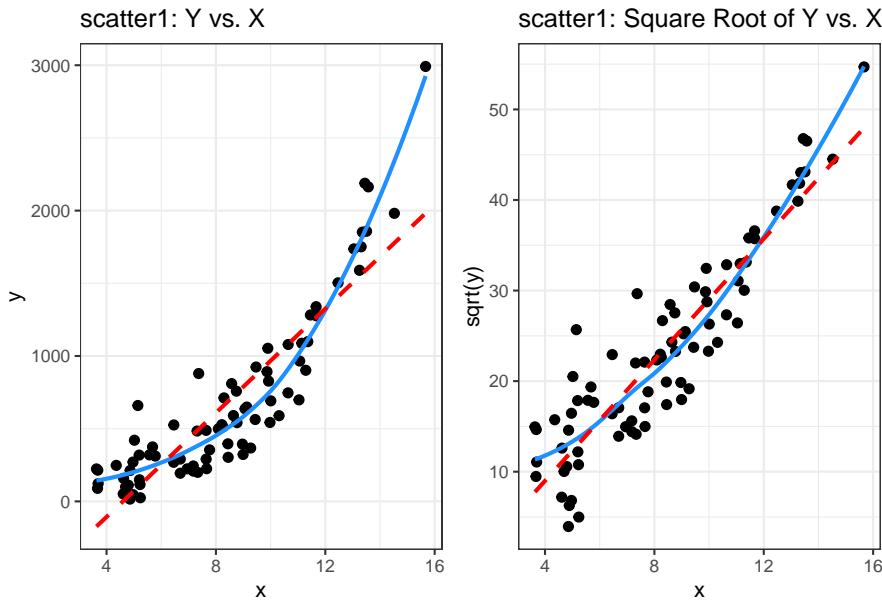
p2 <- ggplot(scatter1, aes(x = x, y = sqrt(y))) +
  geom_point(size = 2) +
```

```

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.

14.4 Checking on a Transformation or Re-Expression

We can do three more things to check on our transformation.

1. We can calculate the correlation of our original and re-expressed associations.
2. We can use the `testTransform` function in the `car` library in R to perform a statistical test comparing the optimal choice of power ($\lambda = 0.44$) to various other transformations.
3. 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.

Option 3 is by far the most important in practice, and it's the one we'll focus on going forward, but we'll demonstrate all three here.

14.4.1 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.

14.4.2 Using the `testTransform` function

Here, we use the `testTransform` function (also from the `car` package) to compare the optimal choice determined by the `powerTransform` function (here $\lambda = 0.44$) to $\lambda = 0$ (logarithm), 0.5 (square root) and 1 (no transformation).

```
testTransform(powerTransform(scatter1$y ~ scatter1$x), 0)
LRT df      pval
LR test, lambda = (0) 46.17947 1 1.079e-11
testTransform(powerTransform(scatter1$y ~ scatter1$x), 0.5)

LRT df      pval
LR test, lambda = (0.5) 1.024888 1 0.31136
testTransform(powerTransform(scatter1$y ~ scatter1$x), 1)

LRT df      pval
LR test, lambda = (1) 63.75953 1 1.4433e-15
```

- It looks like only the square root ($\lambda = 0.5$) of these three options is not significantly worse by the log-likelihood criterion applied here than the optimal choice.
- That's because it's the only one with a p value larger than our usual standard for statistical significance, of 0.05.

14.4.3 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.

```

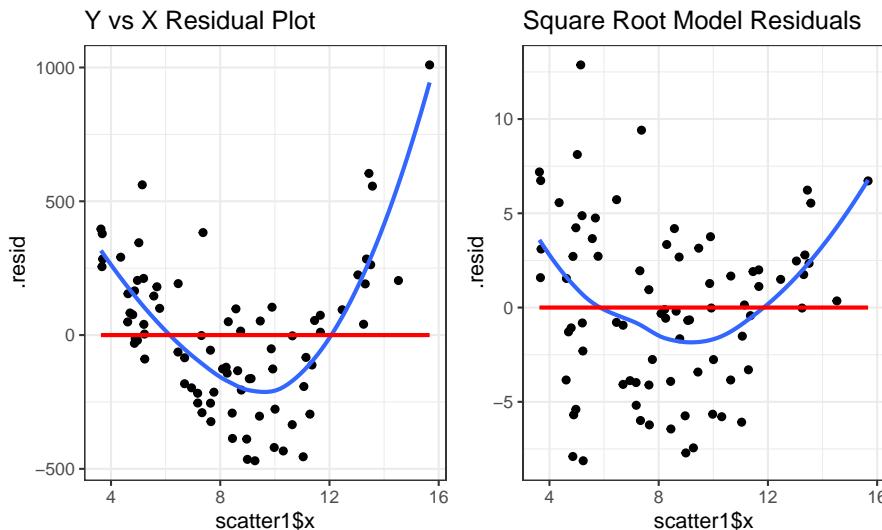
model.orig <- lm(scatter1$y ~ scatter1$x)
model.sqrt <- lm(sqrt(scatter1$y) ~ scatter1$x)

p1 <- augment(model.orig) %>%
  ggplot(., 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 <- augment(model.sqrt) %>%
  ggplot(., 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.

Chapter 15

Dehydration Recovery in Kids: A Small Study

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)
```

	<code>id</code>	<code>recov.score</code>	<code>dose</code>	<code>age</code>
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

	<code>weight</code>
Min.	:22.00
1st Qu.	:34.50
Median	:47.50
Mean	:46.89

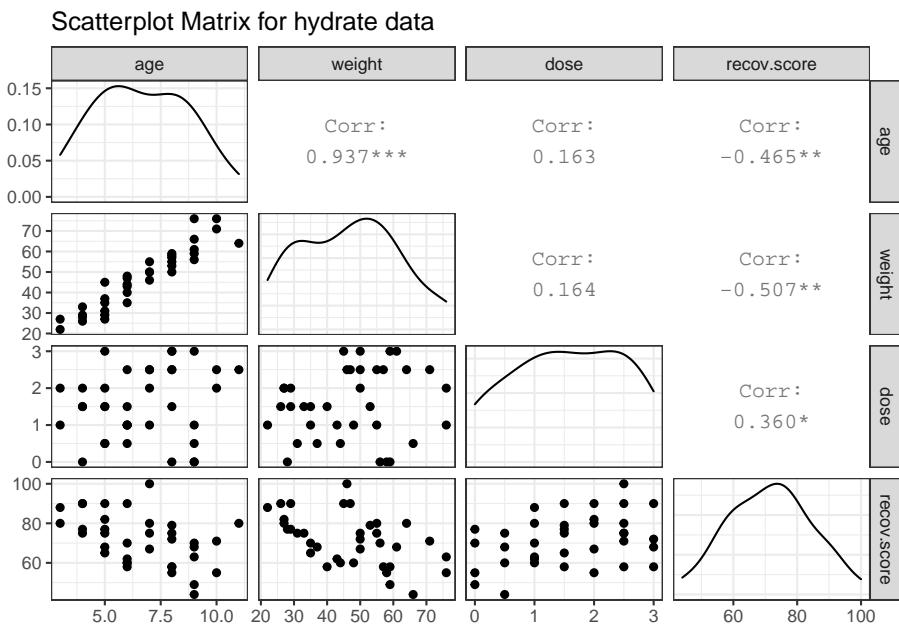
3rd Qu.: 57.25
 Max. : 76.00

There are no missing values, and all of the ranges make sense. There are no especially egregious problems to report.

15.1 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.

```
GGally::ggpairs(dplyr::select(hydrate, age, weight, dose, recov.score),
                 title = "Scatterplot Matrix for hydrate data")
```



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`. ($r = 0.16$)

15.2 Are the recovery scores well described by a Normal model?

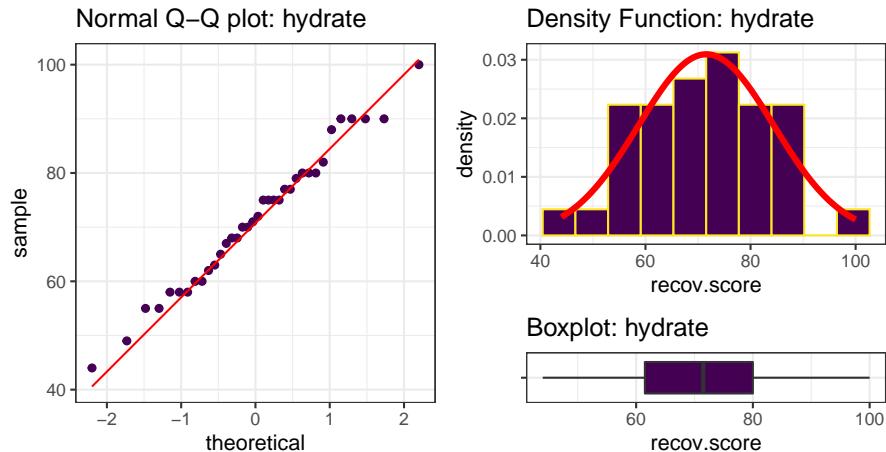
Next, we'll do a more thorough graphical summary of our outcome, recovery score.

```
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)))
```



```
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.

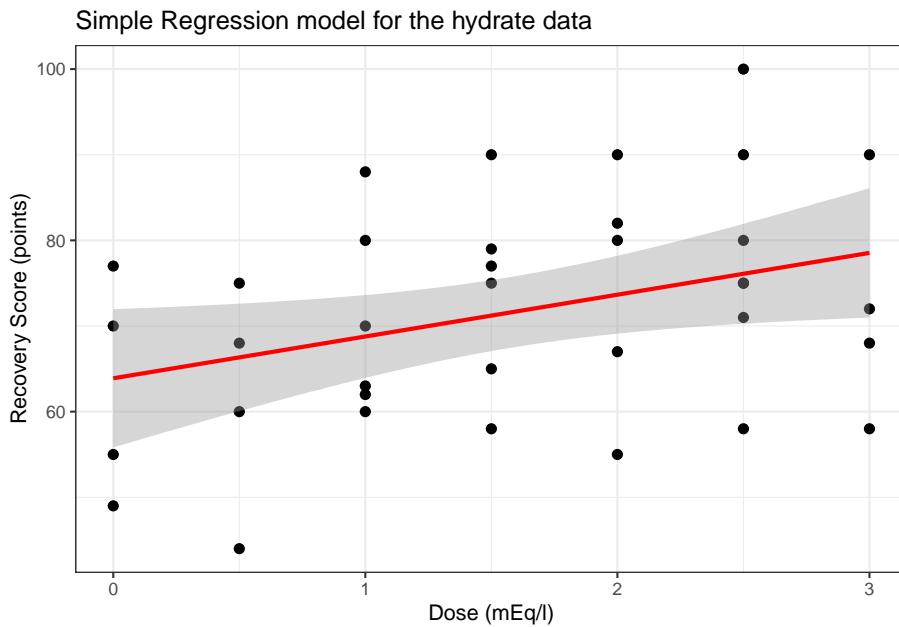
15.3 Simple Regression: Using Dose to predict Recovery

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?

15.4 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") +
```

```
theme_bw() +
  labs(title = "Simple Regression model for the hydrate data",
       x = "Dose (mEq/l)", y = "Recovery Score (points)")
```



15.5 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) %>% kable(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`.

15.5.1 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) %>% kable(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.

15.6 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
```

15.6.1 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.

15.6.2 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.

15.6.3 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

270CHAPTER 15. DEHYDRATION RECOVERY IN KIDS: A SMALL STUDY

- 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).

15.6.4 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.

```
cor(hydrate$recov.score, hydrate$dose)
```

```
[1] 0.359528
```

15.6.5 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:

- H₀: 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:

- H₀: This predictor adds no 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.

15.6.6 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.
 - $R_{adj}^2 = 1 - \frac{(1-R^2)(n-1)}{n-k}$, 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, $R_{adj}^2 = 1 - \frac{(1-0.1293)(35)}{34} = 0.1037$
 - 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.

15.6.7 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.
- H_A: 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.
- H_A: 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.

15.7 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
```

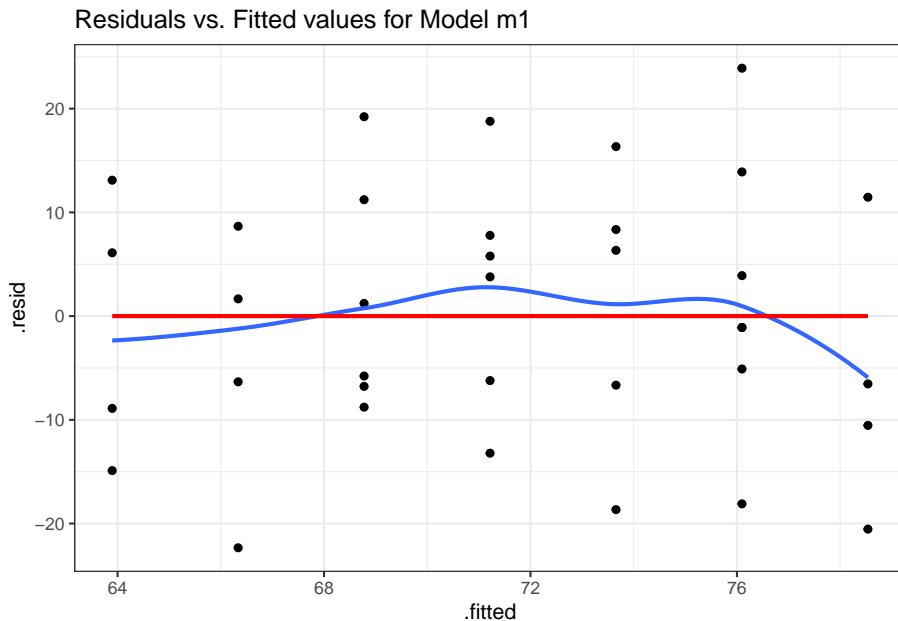
- The R-squared for our regression model is equal to the η^2 for this ANOVA model.
 - If we divide SS(`dose`) = 752.2 by the total sum of squares (752.2 + 5066.7), we’ll get the multiple R-squared [0.1293]

- 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.

15.8 Plotting Residuals vs. Fitted Values

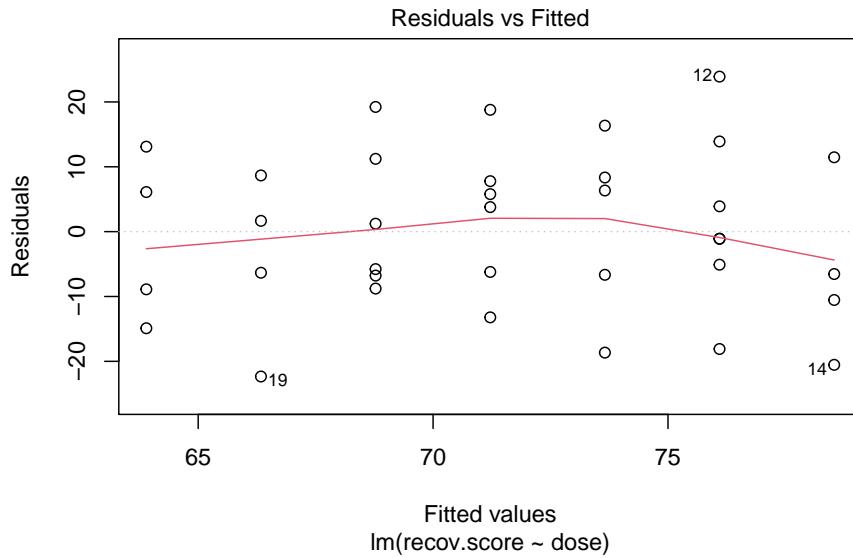
To save the residuals and predicted (fitted) values from this simple regression model, we can use the `resid` and `fitted` commands, respectively, or we can use the `augment` function in the `broom` package to obtain a tidy data set containing these objects and others.

```
augment(m1) %>%
  ggplot(., 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")
```



We can also obtain a plot of residuals vs. fitted values for `m1` using the following code from base R.

```
plot(m1, which = 1)
```



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.

Chapter 16

Highlights of What We've Seen So Far

16.1 Key Graphical Descriptive Summaries for Quantitative Data

- **Histograms** and their variants, including smooth density curves, and normal density functions based on the sample mean and sample standard deviation
- **Boxplots** and the like, including ridgeline plots and violin plots, that show more of the distribution in a compact format that is especially useful for comparisons
- **Normal QQ Plots** which are plots of the ordered data (technically, the order statistics) against certain quantiles of the Normal distribution - show curves to indicate skew, and “S” shaped arcs to indicate seriously heavy- or light-tailed distributions compared to the Normal.

16.2 Key Numerical Descriptive Summaries for Quantitative Data

- Measures of *Location* (including Central Tendency), such as the **mean**, **median**, **quantiles** and even the **mode**.
- Measures of *Spread*, including the **range**, **IQR** (which measures the variability of points near the center of the distribution), **standard deviation** (which is less appropriate as a summary measure if the data show substantial skew or heavy-tailedness), **variance**, **standard error**, **median**

absolute deviation (which is less affected by outlying values in the tails of the distribution than a standard deviation).

- I'll mention the **coefficient of variation** (ratio of the standard deviation to the mean, expressed as a percentage, note that this is only appropriate for variables that take only positive values.)
- One Key Measure of *Shape* is nonparametric skew (`skew1`), which can be used to help confirm plot-based decisions about data shape.

16.3 The Empirical Rule - Interpreting a Standard Deviation

If the data are approximately Normally distributed, then the mean and median will be very similar, and there will be minimal skew and no large outlier problem.

Should this be the case, the mean and standard deviation describe the distribution well, and the **Empirical Rule** will hold reasonably well.

If the data are (approximately) Normally distributed, then

- About 68% of the data will fall within one standard deviation of the mean
- Approximately 95% of the data will fall within two standard deviations of the mean
- Approximately 99.7% of the data will fall within three standard deviations of the mean.

16.4 Identifying “Outliers” Using Fences and/or Z Scores

- Distributions can be symmetric, but still not Normally distributed, if they are either outlier-prone (heavy-tailed) or light-tailed.
- Outliers can have an important impact on other descriptive measures.
- John Tukey described **fences** which separated non-outlier from outlier values in a distribution. Generally, the fences are set 1.5 IQR away from the 25th and 75th percentiles in a boxplot.
- Or, we can use **Z scores** to highlight the relationship between values and what we might expect if the data were normally distributed.
- The Z score for an individual value is that value minus the data's mean, all divided by the data's standard deviation.
- If the data are normally distributed, we'd expect all but 5% of its observations to have Z scores between -2 and +2, for example.

16.5 Summarizing Bivariate Associations: Scatterplots and Regression Lines

- The most important tools are various **scatterplots**, often accompanied by **regression lines** estimated by the method of least squares, and by (loess) **smooths** which permit local polynomial functions to display curved relationships.
- In a multivariate setting, we will occasionally consider plots in the form of a **scatterplot matrix** to enable simultaneous comparisons of multiple two-way associations.
- We fit linear models to our data using the **lm** function, and we evaluate the models in terms of their ability to effectively predict an outcome given a predictor, and through R-square, which is interpreted as the proportion of variation in the outcome accounted for by the model.

16.6 Summarizing Bivariate Associations With Correlations

- **Correlation coefficients**, of which 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.
- Also relevant to us is the **Spearman rank correlation coefficient**, which is obtained by using the usual formula for a Pearson correlation, but on the ranks ($1 = \text{minimum}$, $n = \text{maximum}$, with average ranks 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 monotone association.

Chapter 17

Confidence Intervals for a Mean

17.1 Love-boost.R is something we'll start using now

In this part of the course, we'll make use of a few scripts I've gathered for you.

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

17.2 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.

17.3 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.

17.4 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" values¹ are said to be 0.66-1.10 mcg/ml, which is 66 - 110 micrograms per deciliter.

```
serzinc <- read_csv("data/serzinc.csv")
```

```
Parsed with column specification:
cols(
  ID = col_character(),
  zinc = col_double()
)
```

```
summary(serzinc)
```

¹Reference values for those over the age of 10 years at <http://www.mayomedicallaboratories.com/test-catalog/Clinical+and+Interpretive/8620> , visited 2019-09-17.

```

      ID          zinc
Length:462      Min.   : 50.00
Class :character 1st Qu.: 76.00
Mode  :character Median  : 86.00
                  Mean   : 87.94
                  3rd Qu.: 98.00
                  Max.   :153.00

```

17.5 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).

17.6 Exploratory Data Analysis for Serum Zinc

17.6.1 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.

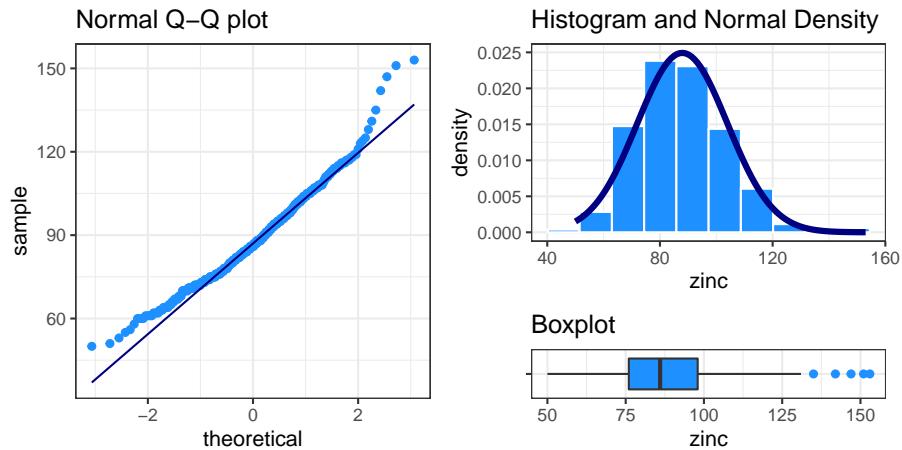
```
p1 <- ggplot(serzinc, aes(sample = zinc)) +
  geom_qq(col = "dodgerblue") + geom_qq_line(col = "navy") +
  theme(aspect.ratio = 1) +
  labs(title = "Normal Q-Q plot")

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") +
  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")
```

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.

17.6.2 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.

```
mosaic::favstats(~ zinc, data = serzinc) %>%
  kable(digits = 3)



|  | min | Q1 | median | Q3 | max | mean   | sd     | n   | missing |
|--|-----|----|--------|----|-----|--------|--------|-----|---------|
|  | 50  | 76 | 86     | 98 | 153 | 87.937 | 16.005 | 462 | 0       |


serzinc %>%
  summarize(mean(zinc), median(zinc), sd(zinc),
            skew1 = (mean(zinc) - median(zinc))/sd(zinc)) %>%
  kable(digits = 3)



| mean(zinc) | median(zinc) | sd(zinc) | skew1 |
|------------|--------------|----------|-------|
| 87.937     | 86           | 16.005   | 0.121 |


```

The skew1 value here (mean - median divided by the standard deviation) backs up our graphical assessment, that the data are slightly right skewed.

```
serzinc %$% psych::describe(zinc)

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
```

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.

17.7 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.

17.8 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.

17.9 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.

17.10 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 deviations 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*.

17.11 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.

17.11.1 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:

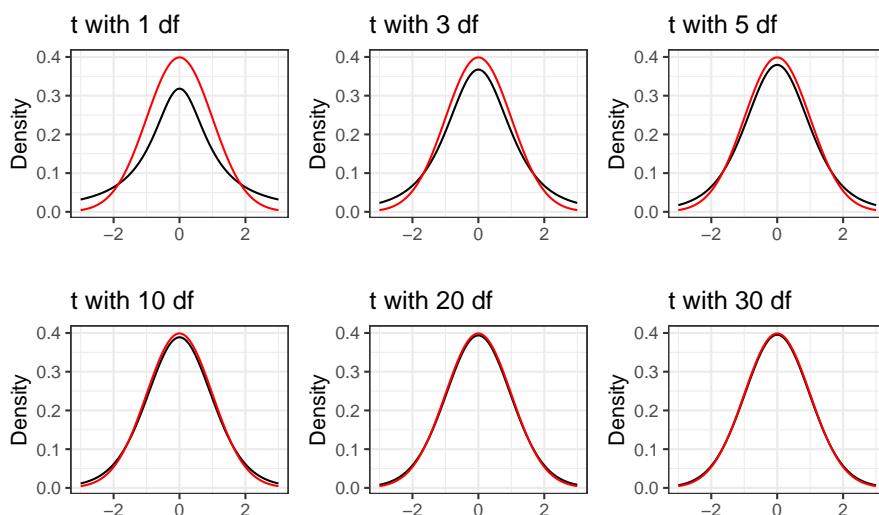
```
qt(alphaover2, df = n-1, lower.tail=FALSE)
```

17.11.2 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

Various t distributions and the Standard Normal



In each plot, the Standard Normal distribution is in red

17.12 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 $1 - \text{confidence}$ = 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.

17.13 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 `broom` to obtain the key information from a model like this:

```
tidy(model_zinc, conf.int = TRUE, conf = 0.90) %>%
  kable(digits = 2)
```

term	estimate	std.error	statistic	p.value	conf.low	conf.high
(Intercept)	87.94	0.74	118.1	0	86.71	89.16

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:

```
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.

```
# requires library(broom)
tidy(tt, conf.int = TRUE, conf = 0.90) %>%
  knitr::kable(digits = 2)
```

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

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 deciliter².

17.14 Interpreting the Result

An appropriate interpretation of the 90% two-sided confidence interval above follows:

²Since the measured zinc levels appear as integers, we should probably be rounding even further in our confidence interval, down to perhaps one decimal place.

- (86.71, 89.16) micrograms per deciliter is a 90% two-sided confidence interval for the population mean serum zinc level among teenage males.
- 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.

17.15 What if we want a 95% or 99% confidence interval instead?

We can obtain them using `tidy` and the same modeling approach.

```
tidy(model_zinc, conf.int = TRUE, conf.level = 0.95)

# 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     86.5     89.4

tidy(model_zinc, conf.int = TRUE, conf.level = 0.99)

# 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     86.0     89.9
```

17.16 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.

```
tt <- t.test(serzinc$zinc, conf.level = 0.95, alternative = "two.sided")
tidy(tt)

# A tibble: 1 x 8
  estimate statistic p.value conf.level alternative
  <dbl>     <dbl>    <dbl>     <dbl>        <chr>
1     87.9     118.      0     0.95        "two.sided"
```

```
estimate statistic p.value parameter conf.low conf.high method alternative
<dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl> <chr>    <chr>
1   87.9     118.      0     461     86.5     89.4 One Sampl~ 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.

17.16.1 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?

17.17 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.

17.18 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?

17.19 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.

17.20 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.

17.21 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

17.22 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)
serzinc %$% Hmisc::smean.cl.boot(zinc, B = 1000, conf.int = 0.90)
```

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
```

17.23 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)
serzinc %$% Hmisc::smean.cl.boot(zinc, B = 1000, conf.int = 0.90)
```

Mean	Lower	Upper
87.93723	86.76775	89.20617

```
serzinc %$% Hmisc::smean.cl.normal(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.

17.23.1 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).

17.24 Using the Bootstrap to develop other CIs

17.24.1 Changing the Confidence Level

What if we wanted to change the confidence level?

```
set.seed(431654)
serzinc %$% Hmisc::smean.cl.boot(zinc, B = 1000, conf.int = 0.95)
```

Mean	Lower	Upper
87.93723	86.51066	89.42002

```
set.seed(431321)
serzinc %$% Hmisc::smean.cl.boot(zinc, B = 1000, conf.int = 0.99)
```

Mean	Lower	Upper
87.93723	86.20657	89.68619

17.25 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)
serzinc %$% Hmisc::smean.cl.boot(zinc, B = 1000, conf.int = 0.90)

      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.

17.25.1 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 summon the `boot` package and call the `boot.ci` function.

```
set.seed(431787)
boot::boot.ci(boot::boot (serzinc$zinc, f.median, 1000), conf=0.90, type="basic")
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates
```

```
CALL :
boot::boot.ci(boot.out = boot::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)

min Q1 median Q3 max     mean        sd    n missing
```

```
50 76      86 98 153 87.93723 16.00469 462      0
```

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.

17.25.2 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::boot.ci(boot::boot (serzinc$zinc, f.IQR, 1000),
    conf=0.95, type="basic")
```

```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates
```

```
CALL :
boot::boot.ci(boot.out = boot::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
```

17.26 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.

17.26.1 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.

17.27 Wilcoxon Signed Rank-based CI in R

```
serzinc %$% wilcox.test(zinc, conf.int = TRUE, conf.level = 0.95)
```

```
Wilcoxon signed rank test with continuity correction

data: 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
```

17.27.1 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).

17.27.2 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 <- serzinc %$% wilcox.test(zinc, conf.int = TRUE, conf.level = 0.95)
tidy(wt)

# A tibble: 1 x 7
  estimate statistic p.value conf.low conf.high method      alternative
  <dbl>     <dbl>    <dbl>    <dbl>    <dbl> <chr>       <chr>
1     87.5   106953 2.00e-77    86.0     88.5 Wilcoxon signed ra~ two.sided
```

17.28 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.

Chapter 18

The Ibuprofen in Sepsis Randomized Clinical 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 group¹. 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.

¹This 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.

```
sepsis <- read_csv("data/sepsis.csv")
```

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)
```

	<code>id</code>	<code>treat</code>	<code>race</code>	<code>apache</code>
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
				Mean :15.4
				3rd Qu.:20.0
				Max. :35.0
	<code>temp_0</code>	<code>temp_drop</code>		
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		

18.1 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.

18.1.1 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.

18.2 Key Questions for Comparing with Independent Samples

18.2.1 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.

18.2.2 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.

18.2.3 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.

18.2.4 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.)

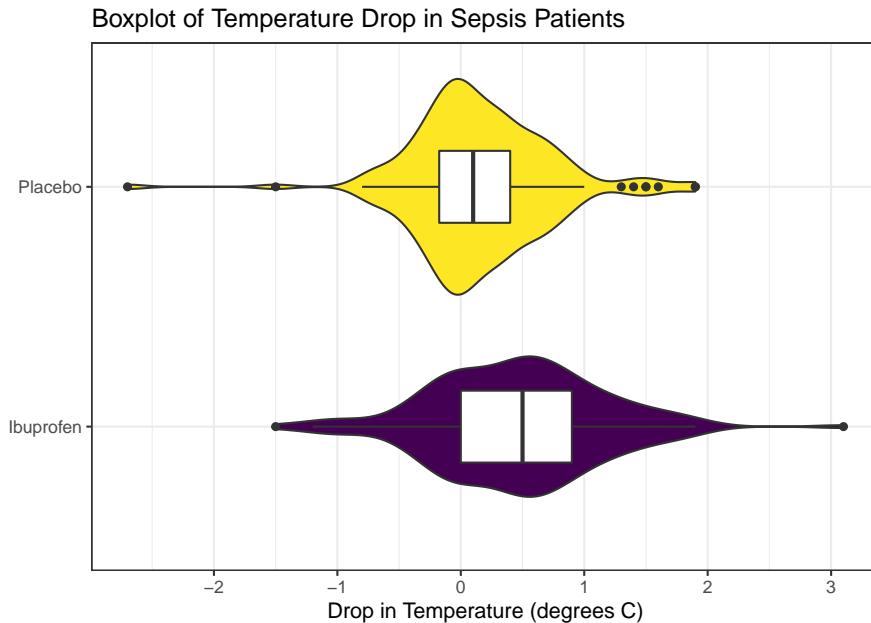
18.2.5 RCT Caveats

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.

18.3 Exploratory Data Analysis

Consider the following boxplot with violin of the `temp_drop` data within each `treat` group.

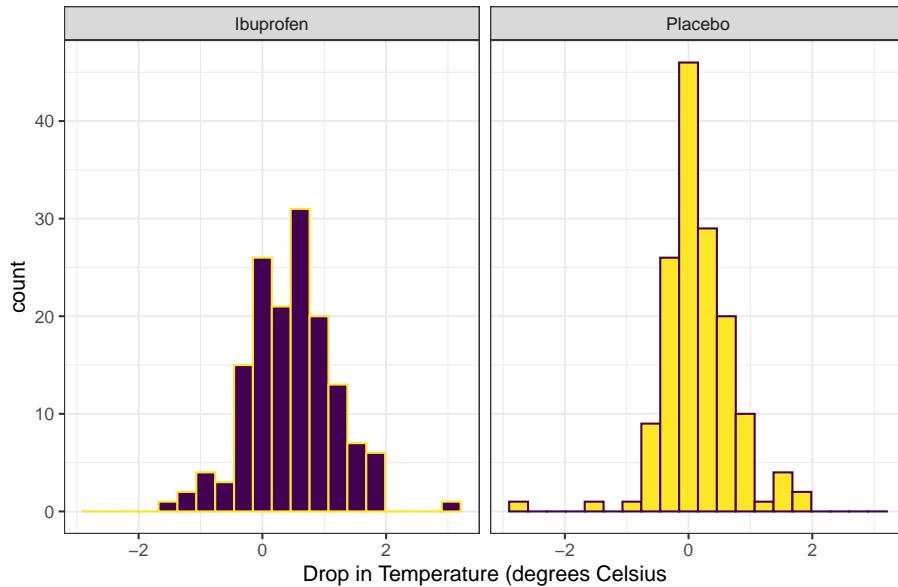
```
ggplot(sepsis, aes(x = treat, y = temp_drop, fill = treat)) +
  geom_violin() +
  geom_boxplot(width = 0.3, fill = "white") +
  scale_fill_viridis_d() +
  guides(fill = FALSE) +
  labs(title = "Boxplot of Temperature Drop in Sepsis Patients",
       x = "", y = "Drop in Temperature (degrees C)") +
  coord_flip() +
  theme_bw()
```



Next, we'll consider faceted histograms of the data.

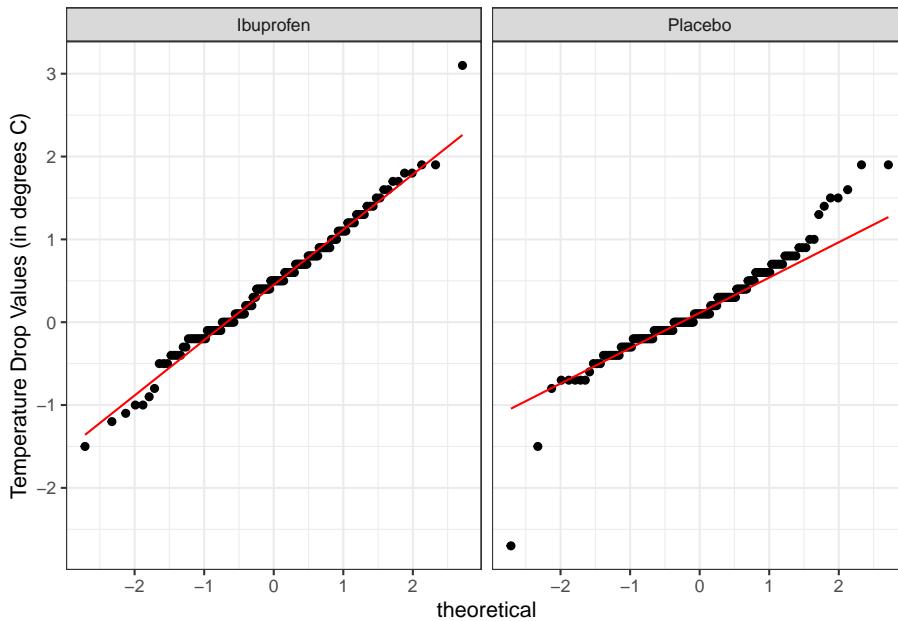
```
ggplot(sepsis, aes(x = temp_drop, fill = treat, color = treat)) +
  geom_histogram(bins = 20) +
  scale_fill_viridis_d() +
  scale_color_viridis_d(direction = -1) +
  guides(fill = FALSE, color = FALSE) +
  labs(title = "Histograms of Temperature Drop in Sepsis Patients",
       x = "Drop in Temperature (degrees Celsius)") +
  theme_bw() +
  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.

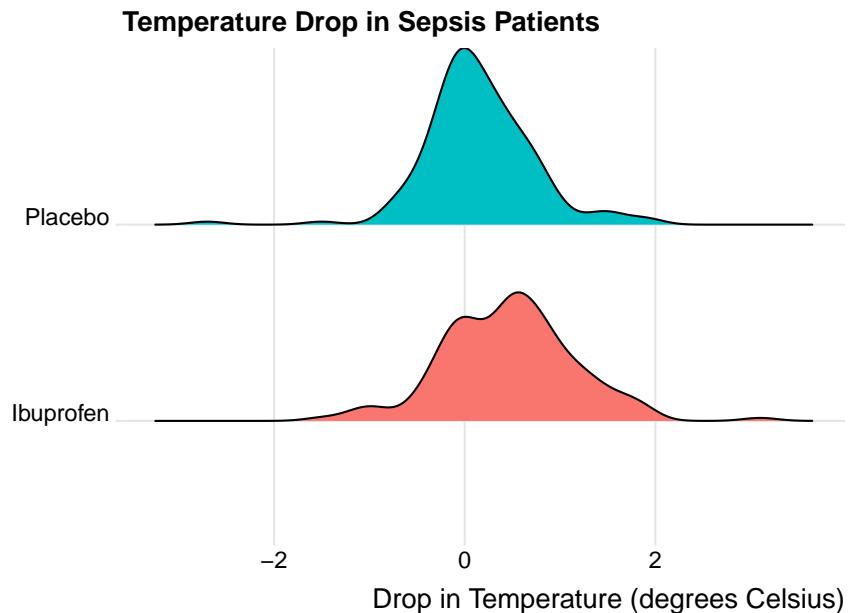
```
ggplot(sepsis, aes(sample = temp_drop)) +
  geom_qq() + geom_qq_line(col = "red") +
  theme_bw() +
  facet_wrap(~ treat) +
  labs(y = "Temperature Drop Values (in degrees C)")
```



We'll could perhaps also look at a ridgeline plot.

```
ggplot(sepsis, aes(x = temp_drop, y = treat, fill = treat)) +
  ggridges::geom_density_ridges(scale = 0.9) +
  guides(fill = FALSE) +
  labs(title = "Temperature Drop in Sepsis Patients",
       x = "Drop in Temperature (degrees Celsius)", y = "") +
  ggridges::theme_ridges()
```

Picking joint bandwidth of 0.182



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
Ibuprofen	-1.5	0.000	0.5	0.9	3.1	0.4640000	0.6877919	150	0
Placebo	-2.7	-0.175	0.1	0.4	1.9	0.1526667	0.5709637	150	0

18.4 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) %>%
  kable(digits = 3)
```

18.5. T-BASED CI FOR POPULATION MEAN1 - MEAN2 DIFFERENCE 311

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

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) %>%
  kable(digits = 3)
```

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

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.

18.5 t-based CI for population mean1 - mean2 difference

18.5.1 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.

```
sepsis %$% t.test(temp_drop ~ treat,
  conf.level = 0.90,
  alt = "two.sided",
  var.equal = TRUE)
```

Two Sample t-test

```
data: temp_drop by treat
t = 4.2656, df = 298, p-value = 2.68e-05
```

312CHAPTER 18. THE IBUPROFEN IN SEPSIS RANDOMIZED CLINICAL TRIAL

```
alternative hypothesis: true difference in means is not equal to 0
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 <- sepsis %$% t.test(temp_drop ~ treat,
                         conf.level = 0.90,
                         alt = "two.sided",
                         var.equal = TRUE)
tidy(tt1)

# A tibble: 1 x 10
  estimate estimate1 estimate2 statistic p.value parameter conf.low conf.high
  <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
1 0.311     0.464    0.153    4.27  2.68e-5     298    0.191    0.432
# ... with 2 more variables: method <chr>, alternative <chr>
```

18.5.2 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
```

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

18.5. T-BASED CI FOR POPULATION MEAN1 - MEAN2 DIFFERENCE313

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) %>% kable(digits = 3)
```

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

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

18.5.3 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 com-

pared are not assumed to have the same variance. Each population is assumed to follow a Normal distribution.

```
sepsis %$% t.test(temp_drop ~ treat,
                    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 is not equal to 0
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 <- sepsis %$% t.test(temp_drop ~ treat,
                           conf.level = 0.90,
                           alt = "two.sided")

tidy(tt0)

# A tibble: 1 x 10
  estimate estimate1 estimate2 statistic p.value parameter conf.low conf.high
  <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
1 0.311     0.464    0.153    4.27  2.71e-5    288.    0.191    0.432
# ... with 2 more variables: method <chr>, alternative <chr>
```

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.

18.6 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 <- sepsis %$% wilcox.test(temp_drop ~ treat,
                           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)

# A tibble: 1 x 7
  estimate statistic p.value conf.low conf.high method      alternative
  <dbl>     <dbl>    <dbl>    <dbl>    <dbl> <chr>      <chr>
1 0.300    14614. 7.28e-6   0.200    0.400 Wilcoxon rank sum ~ two.sided

```

18.7 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>.

18.7.1 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)
```

```
sepsis %$% bootdif(temp_drop, 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).

18.8 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?

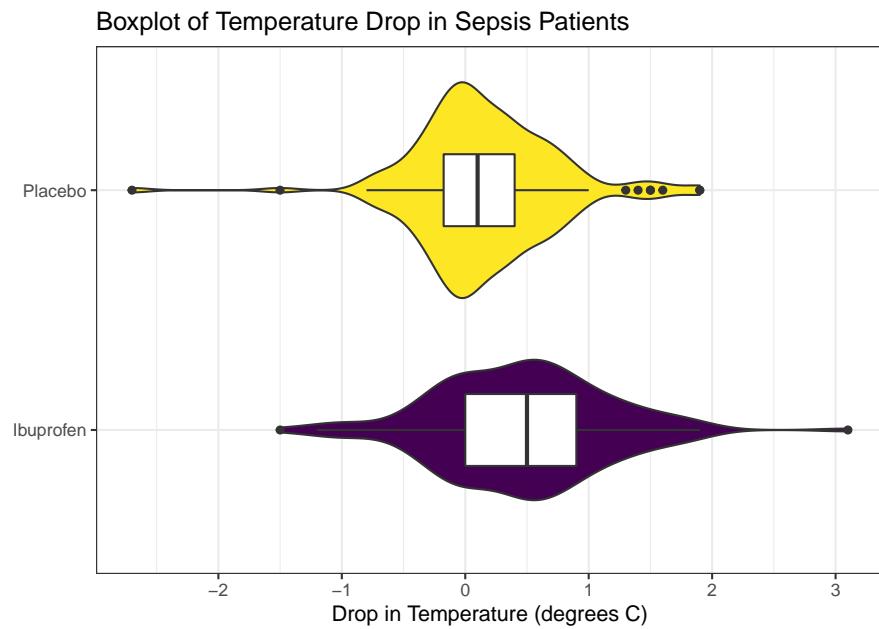
18.9 Results for the `sepsis` study

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.

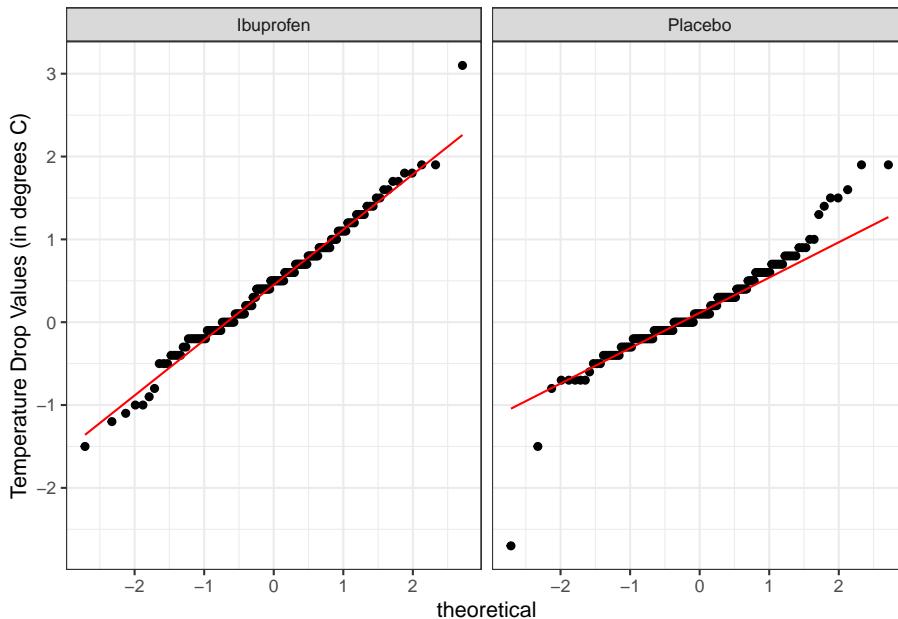
Questions 7 and 8 don't apply, because these are independent samples of data, rather than paired samples.

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") +
  scale_fill_viridis_d() +
  guides(fill = FALSE) +
  labs(title = "Boxplot of Temperature Drop in Sepsis Patients",
       x = "", y = "Drop in Temperature (degrees C)") +
  coord_flip() +
  theme_bw()
```



```
ggplot(sepsis, aes(sample = temp_drop)) +  
  geom_qq() + geom_qq_line(col = "red") +  
  theme_bw() +  
  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.

18.9.1 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 <- sepsis %$% t.test(temp_drop ~ treat,
                                      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 <- sepsis %$% t.test(temp_drop ~ treat,
                                      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 <- sepsis %$% wilcox.test(temp_drop ~ treat,
                                             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 <- sepsis %$% bootdif(temp_drop, 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?

18.10 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%)	

18.11 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.

```
sepsis %$% table(treat, dropped) %>% Epi::twoby2(alpha = 0.10)
```

2 by 2 table analysis:

322CHAPTER 18. THE IBUPROFEN IN SEPSIS RANDOMIZED CLINICAL TRIAL

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.

Bibliography

- Baumer, B. S., Kaplan, D. T., and Horton, N. J. (2017). *Modern Data Science with R*. CRC Press, Boca Raton, FL.
- Bernard, G. R., Wheeler, A. P., Russell, J. A., Schein, R., Summer, W. R., Steinberg, K. P., Fulkerson, W. J., Wright, P. E., Christman, B. W., Dupont, W. D., Higgins, S. B., and Swindell, B. B. (1997). The effects of ibuprofen on the physiology and survival of patients with sepsis. *New England Journal of Medicine*, 336:912–18.
- Bock, D. E., Velleman, P. F., and De Veaux, R. D. (2004). *Stats: Modelling the World*. Pearson Addison-Wesley, Boston MA.
- Dupont, W. D. (2002). *Statistical Modeling for Biomedical Researchers*. Cambridge University Press, New York.
- Gelman, A. and Nolan, D. (2017). *Teaching Statistics: A Bag of Tricks*. Oxford University Press, Oxford, UK, second edition.
- Grolemund, G. and Wickham, H. (2019). *R for Data Science*. O'Reilly.
- Ismay, C. and Kim, A. Y. (2019). *ModernDive: Statistical Inference via Data Science*.
- Norman, G. R. and Streiner, D. L. (2014). *Biostatistics: The Bare Essentials*. People's Medical Publishing House, fourth edition.
- Pagano, M. and Gauvreau, K. (2000). *Principles of Biostatistics*. Duxbury Press, second edition.
- Ramsey, F. L. and Schafer, D. W. (2002). *The Statistical Sleuth: A Course in Methods of Data Analysis*. Duxbury, Pacific Grove, CA, second edition.
- Vittinghoff, E., Glidden, D. V., Shiboski, S. C., and McCulloch, C. E. (2012). *Regression Methods in Biostatistics: Linear, Logistic, Survival, and Repeated Measures Models*. Springer-Verlag, Inc., second edition.
- Wainer, H. (1997). *Visual Revelations: Graphical Tales of Fate and Deception from Napoleon Bonaparte to Ross Perot*. Springer-Verlag, New York.
- Wainer, H. (2005). *Graphic Discovery: A Trout in the Milk and Other Visual Adventures*. Princeton University Press, Princeton, NJ.

- Wainer, H. (2013). *Medical Illuminations: Using Evidence, Visualization and Statistical Thinking to Improve Healthcare*. Oxford University Press, New York.
- Yamada, S. and Boulding, E. (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.