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

Thomas E. Love, Ph.D.

Version: 2019-09-15 14:38:02

Contents

Introduction

These Notes provide a series of examples using R to work through issues that are likely to come up in PQHS/CRSP/MPHP 431.

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. Just email us at 431-help at case dot edu, or submit a pull request.

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.

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" which we use to interface with the R language) in class.

All data and R code related to these notes are available through the course website.

Structure

The Notes, like the 431 course, fall in three main parts.

Part A is about **visualizing data and exploratory data analyses**. These Notes focus on using R to work through issues that arise in the process of exploring data, managing (cleaning and manipulating) data into a tidy format to facilitate useful work downstream, and describing those data effectively with visualizations, numerical summaries, and some simple models.

Part B is about **making comparisons** with data. The Notes discuss the use of R to address comparisons of means and of rates/proportions, primarily. The main ideas include confidence intervals, using the bootstrap and making decisions about power and sample size. We'll also discuss the value (or lack thereof) of p values for assessing hypotheses. Key ideas from Part A that have an impact here include visualizations to check the assumptions behind our inferences, and cleaning/manipulating data to facilitate our comparisons.

Part C is about **building models** with data. The Notes are primarily concerned (in 431) with linear regression models for continuous quantitative outcomes, using one or more predictors. We'll see how to use models to accomplish many of the comparisons discussed in Part B, and make heavy use of visualization and data management tools developed in Part A to assess our models.

Course Philosophy

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 answer real questions using data and equip you with the tools you need in order to answer those questions well (Çetinkaya-Rundel (2017) has more on a related teaching philosophy.)

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

- data gathering
- data wrangling
- exploratory data analysis and visualization
- multivariate modeling
- communication

It also nearly completely avoids formalism and is extremely applied - this is most definitely **not** a course in theoretical or mathematical statistics.

There's very little of this:

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

Instead, there's a lot of this:

```
nyfs1 %>%
    group_by(bmi.cat) %>%
    summarise(mean = mean(waist.circ),
              median = median(waist.circ),
              sd = sd(waist.circ))
```

The 431 course is about **getting things done**. It's not a statistics course alone, nor is it a course in computer programming alone. I think of it as a course in **data science**.

Working with this Document

1. This document is broken down into multiple chapters. Use the table of contents at left to navigate.
2. At the top of the document, you'll see 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 is updated occasionally through the semester. Check the Version information above to verify the last update time.

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.

- 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.

1.1 Why a unicorn?

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

Data Science Venn Diagram 2.0
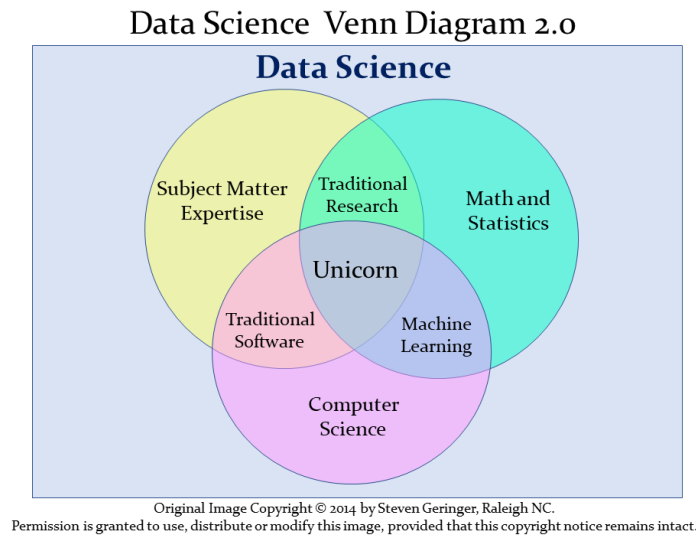
Figure 1.1: Data Science Venn Diagram from Steven Geringer

1.2   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).

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).
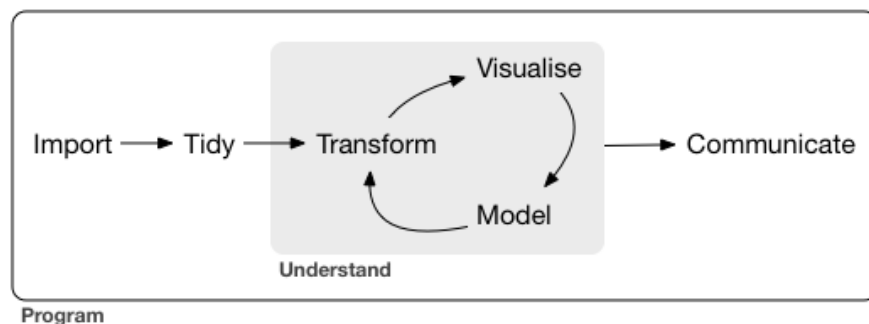
Figure 1.2: Source: R for Data Science: Introduction

1.3 What Will We Discuss in 431?

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.
- Having completed some fundamental work in Part A of the course, we then learn how to use a variety of R packages and statistical methods to accomplish specific inferential tasks (in Part B, mostly) and modeling tasks (in Part C, mostly.)

Chapter 2

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).

2.1 R Markdown

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).

2.2   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.

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

# library(boot); library(pander); library(pwr)

library(grid); library(devtools)
library(knitr); library(NHANES);
library(janitor); library(tidyverse)

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

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

2.3   Other Packages

I may also make use of functions in the following packages/libraries, but when I do so, I will explicitly specify the package name, using a command like `Hmisc::describe(x)`, rather than just `describe(x)`, so as to specify that I want the Hmisc package's version of `describe` applied to whatever x is. Those packages are:

- `aplpack` which provides `stem.leaf` and `stem.leaf.backback` for building fancier stem-and-leaf displays
- `arm` which provides a set of functions for model building and checking that are used in Gelman and Hill (2007)
- `broom` which turns the results lots of different analyses in R into more useful tidy data frames (tibbles.)
- `car` which provides some tools for building scatterplot matrices, but also many other functions described in Fox and Weisberg (2011)
- `cowplot` which is used in Part C to put multiple graphical objects in the same plot, like `gridExtra`: https://cran.r-project.org/web/packages/cowplot/vignettes/introduction.html
- `DataExplorer` for generating highly detailed profiles of a data frame
- `Epi` for 2x2 table analyses and materials for classical epidemiology: http://BendixCarstensen.com/Epi/

- `GGally` for scatterplot and correlation matrix visualizations: http://ggobi. github.io/ggally/
- `ggridges` which is used to make ridgeline plots
- `gridExtra` which includes a variety of functions for manipulating graphs: https://github.com/baptiste/gridextra
- `Hmisc` from Frank Harrell at Vanderbilt U., for its version of `describe` and for many regression modeling functions we'll use in 432. Details on Hmisc are at http://biostat.mc.vanderbilt.edu/wiki/Main/Hmisc. Frank has written several books - the most useful of which for 431 students is probably Harrell and Slaughter (2019)
- `mice`, which we'll use (a little) in 431 for multiple imputation to deal with missing data: http://www.stefvanbuuren.nl/mi/
- `mosaic`, mostly for its `favstats` summary, but Project MOSAIC is a community of educators you might be interested in: http://mosaic-web.org/
- `naniar`, for wrangling and visualizing missingness, and for checking imputations. See http://naniar.njtierney.com/.
- `psych` for its own version of `describe`, but other features are described at http://personality-project.org/r/psych/
- `skimr` for its ability to provide a "skimmed" descriptive analysis of a data set

We'll also use some packages that get loaded via `devtools` and Github by the code in these notes, including:

- `xda` for two functions called `numSummary` and `charSummary`
- `visdat` for two functions called vis_miss`and`vis_dat`
- `patchwork`, which is a framework for composing `ggplot2` objects.

Several other packages are included below, even though they are not used in these Notes, because they will be used in class sessions or in 432.

When compiling the Notes from the original code files, these packages will need to be installed (but not loaded) in R, or an error will be thrown when compiling this document. To install all of the packages used within these Notes, type in (or copy and paste) the following commands and run them in the R Console. Again, you only need to install a package once, but you need to reload it every time you start a new session.

```
pkgs <- c("aplpack", "arm", "babynames", "boot", "broom", "car", "cowplot",
          "DataExplorer", "devtools", "Epi", "faraway", "fivethirtyeight",
          "foreign", "gapminder", "GGally", "ggridges", "gridExtra", "Hmisc",
          "janitor", "kableExtra", "knitr", "lme4", "magrittr", "markdown",
          "MASS", "mice", "mosaic", "multcomp", "naniar", "NHANES", "pander",
          "psych", "pwr", "qcc", "rmarkdown", "rmdformats", "rms",
          "sandwich", "simputation", "skimr", "survival", "tableone",
          "tidyverse", "vcd")
```

```
install.packages(pkgs)
```

Part A. Exploring Data

Chapter 3

Visualizing Data

Part A of these Notes is designed to ease your transition into working effectively with data, so that you can better understand it. 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 subjects[1]. Some of the motivation for this example came from a Figure in Baumer, Kaplan, and Horton (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
```

---

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

```
    <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 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).
```
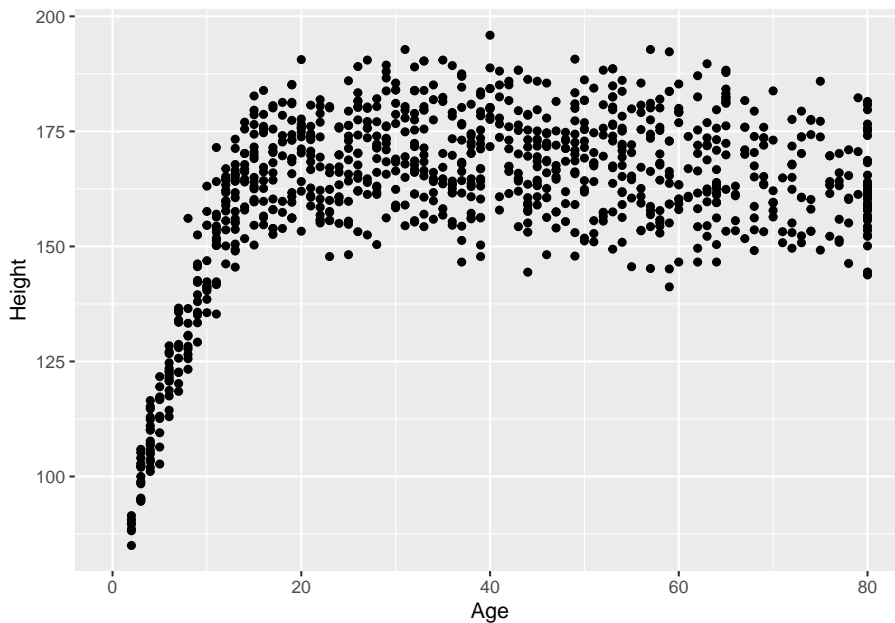
We note several interesting results here.

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 at 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.   : 2.0   Min.   : 85
 1st Qu.:57034   male  :479   1st Qu.:19.0   1st Qu.:156
 Median :62056                Median :37.0   Median :165
 Mean   :61967                Mean   :38.3   Mean   :162
 3rd Qu.:67269                3rd Qu.:56.0   3rd Qu.:174
 Max.   :71875                Max.   :80.0   Max.   :196
```

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.

---

[2]If 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. . .

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

summary(nh_dat2)
```

```
      ID              Sex           Age            Height
 Min.   :51624   female:484   Min.   : 2.0   Min.   : 85
 1st Qu.:57034   male  :479   1st Qu.:19.0   1st Qu.:156
 Median :62056                Median :37.0   Median :165
 Mean   :61967                Mean   :38.3   Mean   :162
 3rd Qu.:67269                3rd Qu.:56.0   3rd Qu.:174
 Max.   :71875                Max.   :80.0   Max.   :196
```

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.

```r
dim(nh_dat2)
```

```
[1] 963   4
```

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

```r
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.")
```
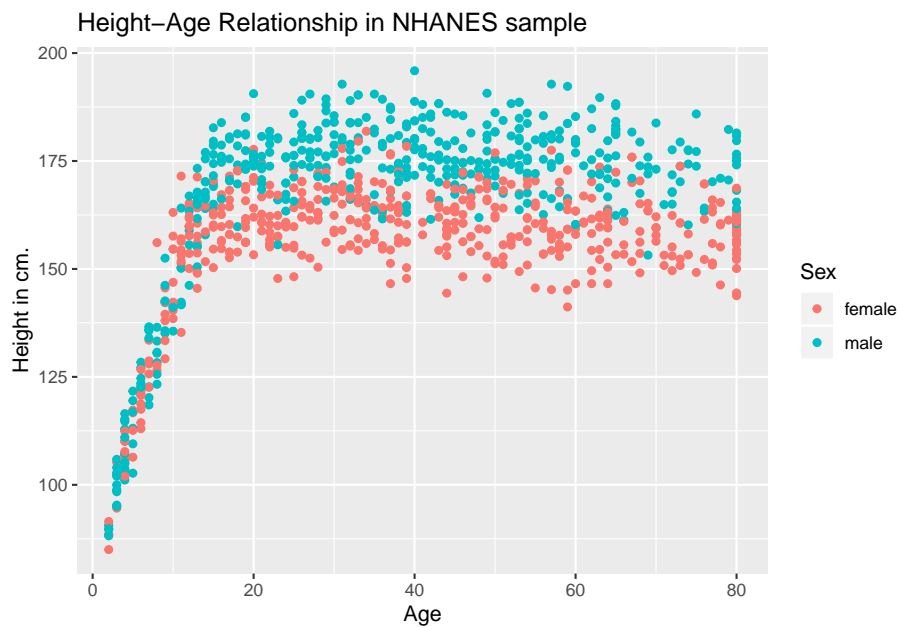

3.4.1 Can we show the Female and Male relationships in separate panels?

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)
```

Height–Age Relationship in NHANES sample
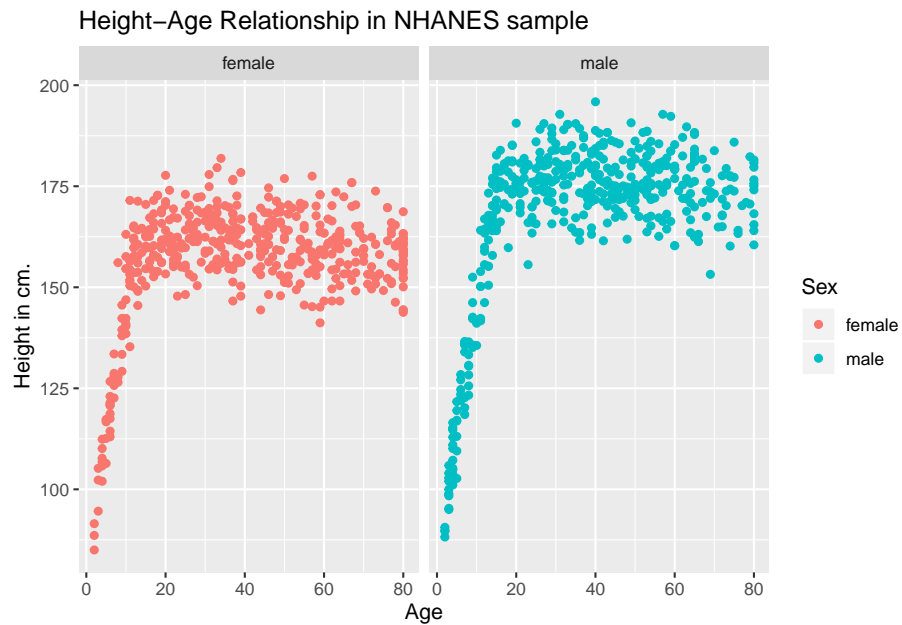
3.4.2 Can we add a smooth curve to show the relationship in each plot?

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)
```

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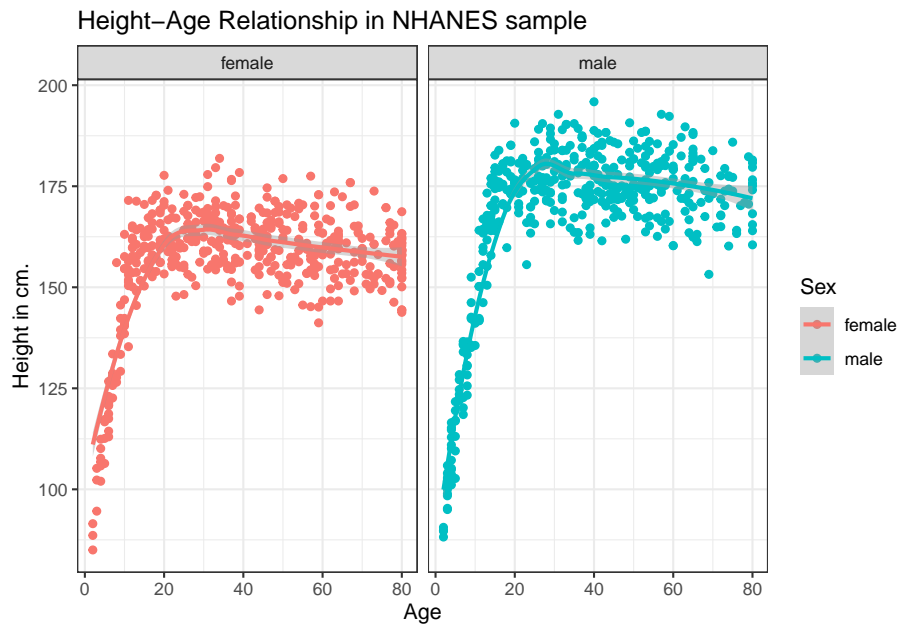
3.4.3 What if we want to assume straight line relationships?

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)
```

Height–Age Relationship in NHANES sample
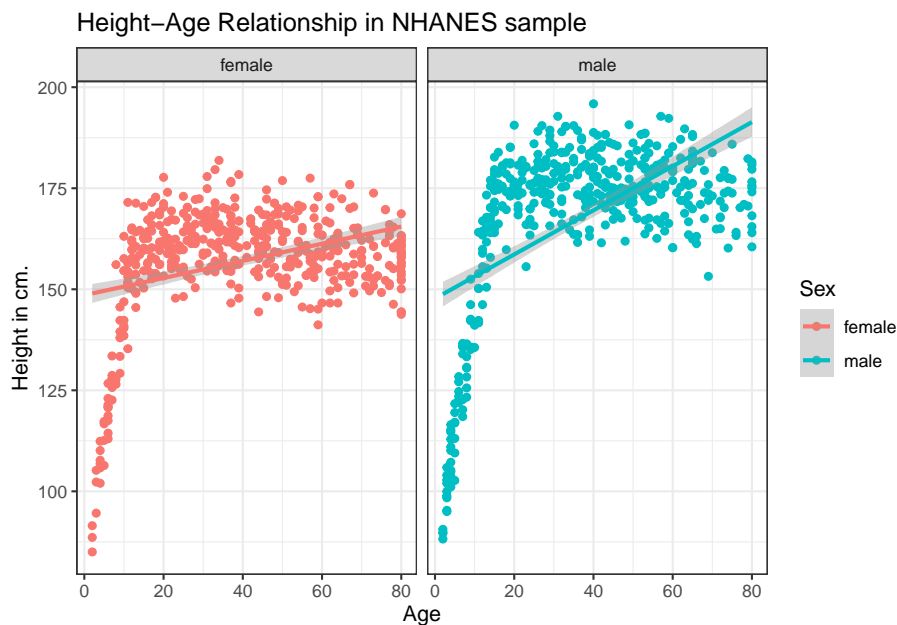
3.5   A New Subset: Ages 21-79

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 a seed for the random sampling from the original NHANES data, so we get the same original sample of 1000 people we started with earlier.
2. Select 1000 people from the NHANES data.
3. Filter the sample to only those people whose age is more than 20 and less than 80 years.
4. Select the variables we will be using in the remainder of this chapter, which will be:
   - `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.

5. Rename **Gender** as **Sex**, to more accurately describe what is being measured.
6. Omit subjects with any missingness on *any* of the variables we've selected.

Can you see how the code below accomplishes these tasks?

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

nh_data_2179 <- 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_data_2179
```

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

3.6 Distribution of Heights

What is the distribution of height in this new sample?

```r
ggplot(data = nh_data_2179, aes(x = Height)) +
    geom_histogram()
```

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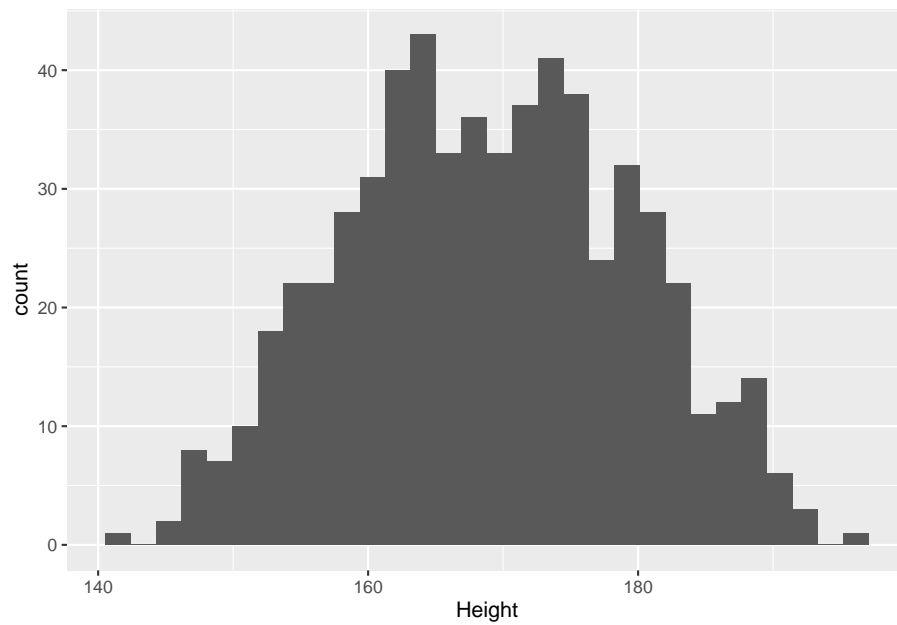
We can do several things to clean this up.

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

```
ggplot(data = nh_data_2179, aes(x = Height)) +
    geom_histogram(bins = 25, col = "yellow", fill = "blue") +
    labs(title = "Height of NHANES subjects ages 21-79",
         x = "Height in cm.")
```

Height of NHANES subjects ages 21–79
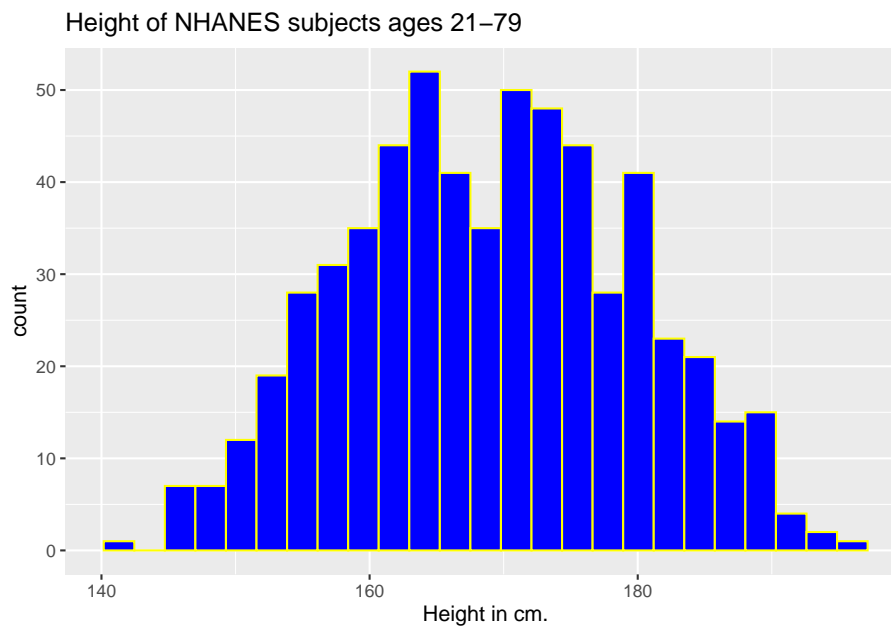
3.6.1 Changing a Histogram's Fill and Color

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_data_2179, 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()
```

Height of NHANES subjects ages 21–79
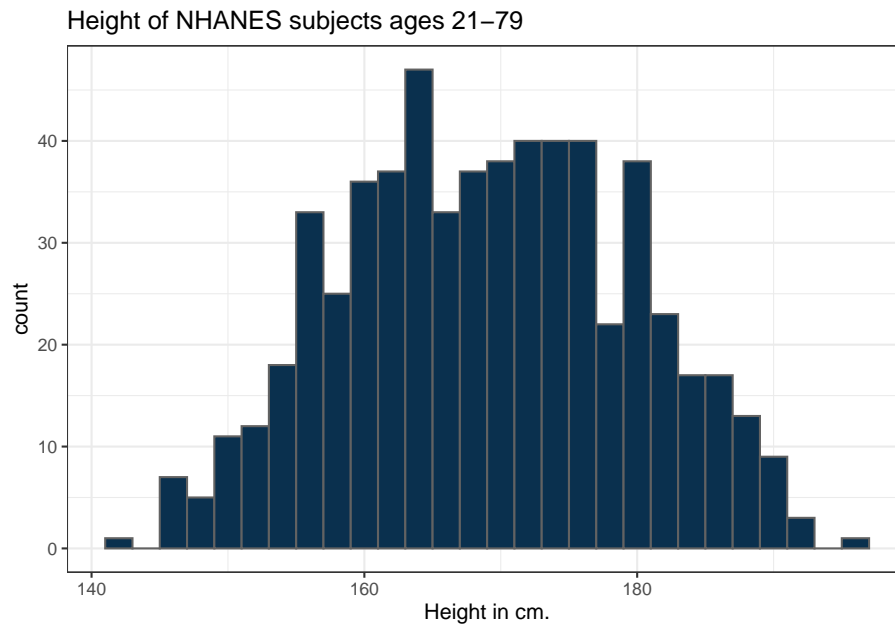
Note the other changes to the graph above.

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_data_2179, aes(x = Sex, y = Height, color = Sex)) +
    geom_point() +
    labs(title = "Height by Sex for NHANES subjects ages 21-79",
        y = "Height in cm.")
```

Height by Sex for NHANES subjects ages 21–79
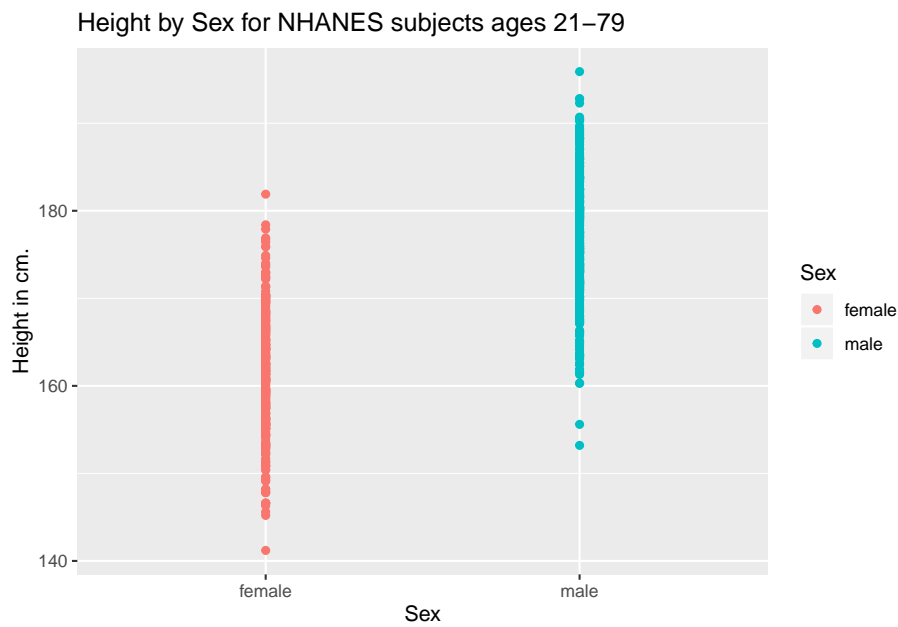
This plot isn't so useful. We can improve things a little by jittering the points horizontally, so that the overlap is reduced.

```
ggplot(data = nh_data_2179, 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.")
```

Height by Sex (jittered) for NHANES subjects ages 21–79
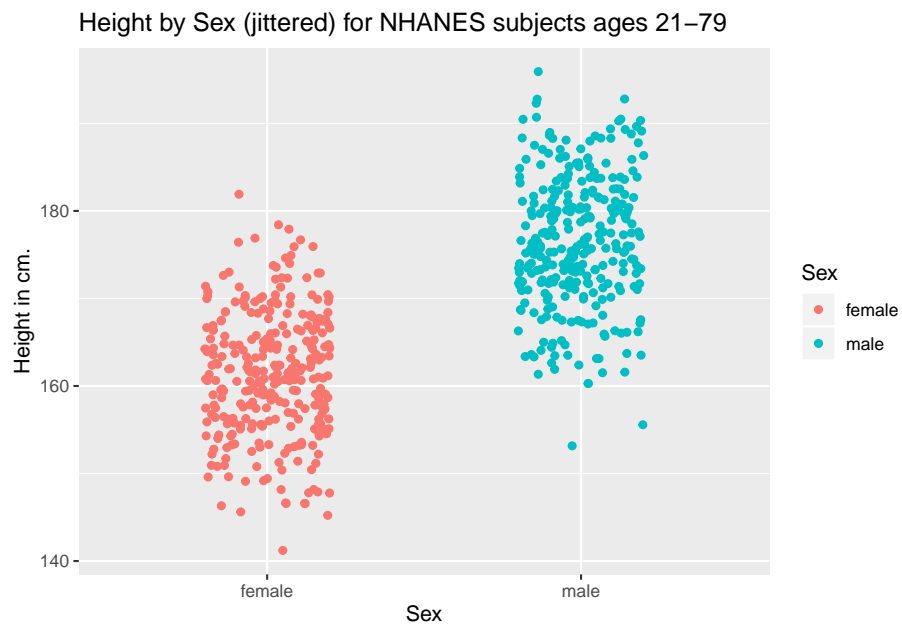
Perhaps it might be better to summarise the distribution in a different way. We might consider a boxplot of the data.

3.7.1   A Boxplot of Height by Sex

```
ggplot(data = nh_data_2179, 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.")
```

Boxplot of Height by Sex for NHANES subjects ages 21–79
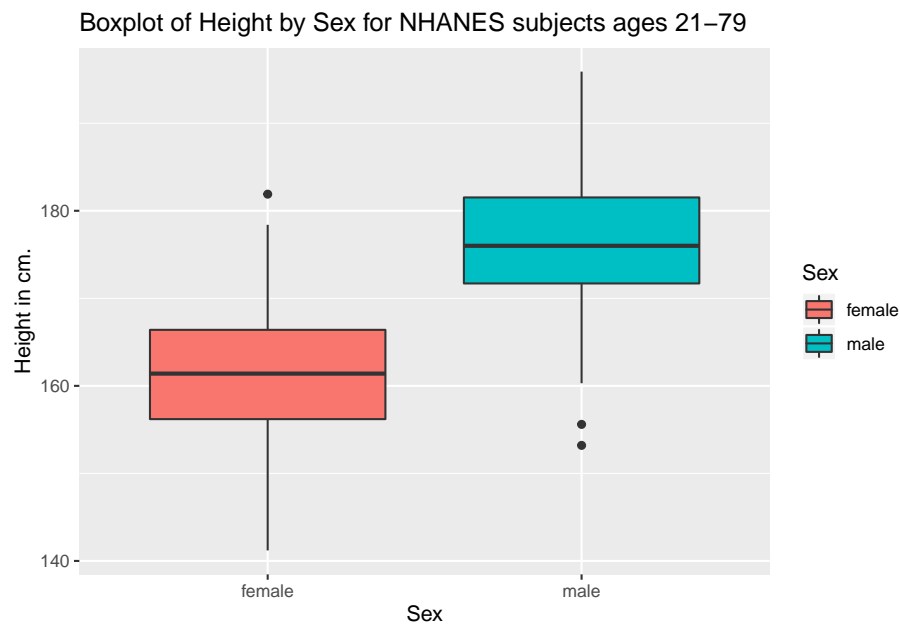
Or perhaps we'd like to see a pair of histograms?

3.7.2 Histograms of Height by Sex

```
ggplot(data = nh_data_2179, 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)
```

Histogram of Height by Sex for NHANES subjects ages 21–79
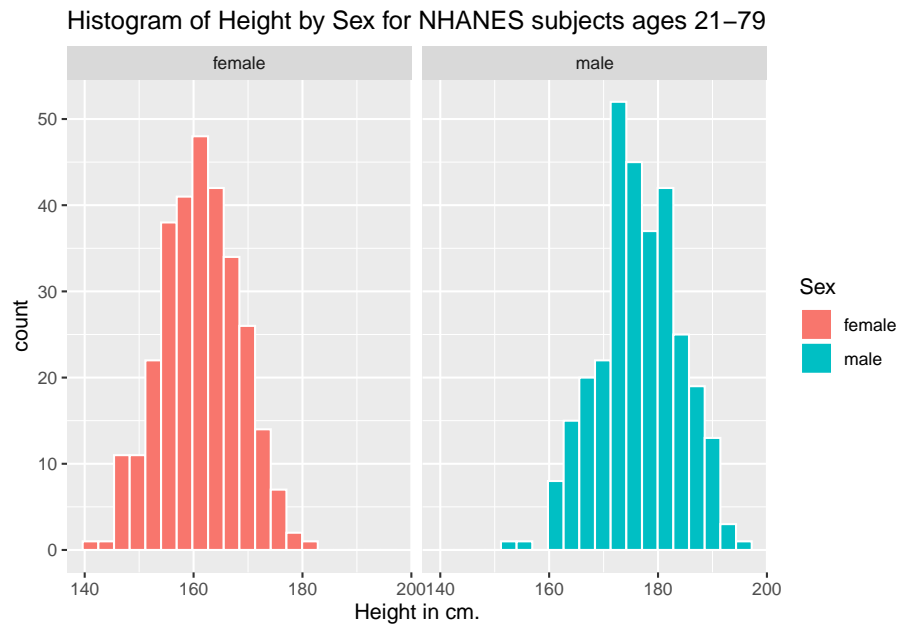
Can we redraw these histograms so that they are a little more comparable, and to get rid of the unnecessary legend?

```
ggplot(data = nh_data_2179, 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 ~ .)
```

Histogram of Height by Sex for NHANES subjects ages 21–79 (Revised)
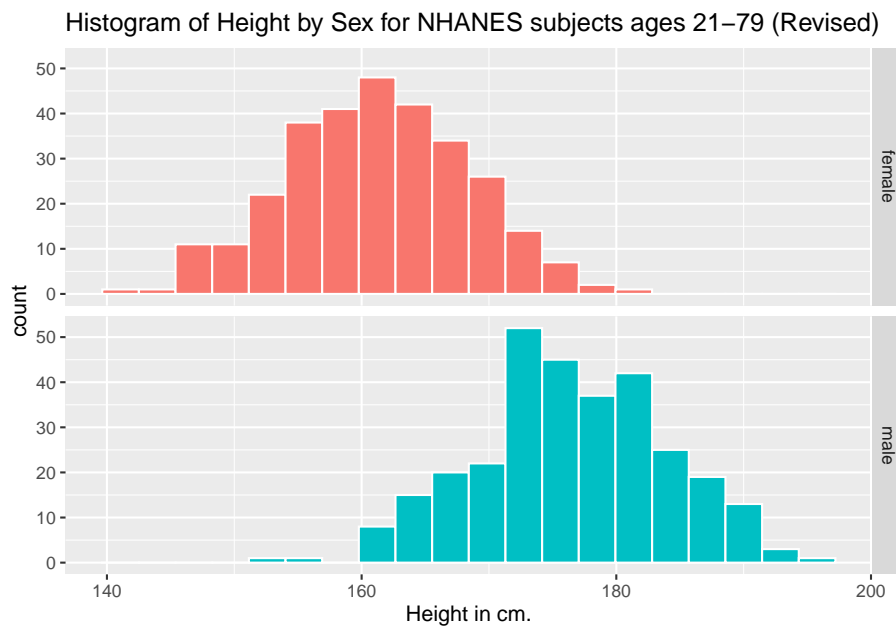
3.8 Looking at Pulse Rate

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_data_2179, aes(x = Pulse)) +
    geom_histogram(binwidth = 1, fill = cwru.blue, col = cwru.gray) +
    labs(title = "Histogram of Pulse Rate: NHANES subjects ages 21-79",
        x = "Pulse Rate (beats per minute)")
```

Histogram of Pulse Rate: NHANES subjects ages 21–79
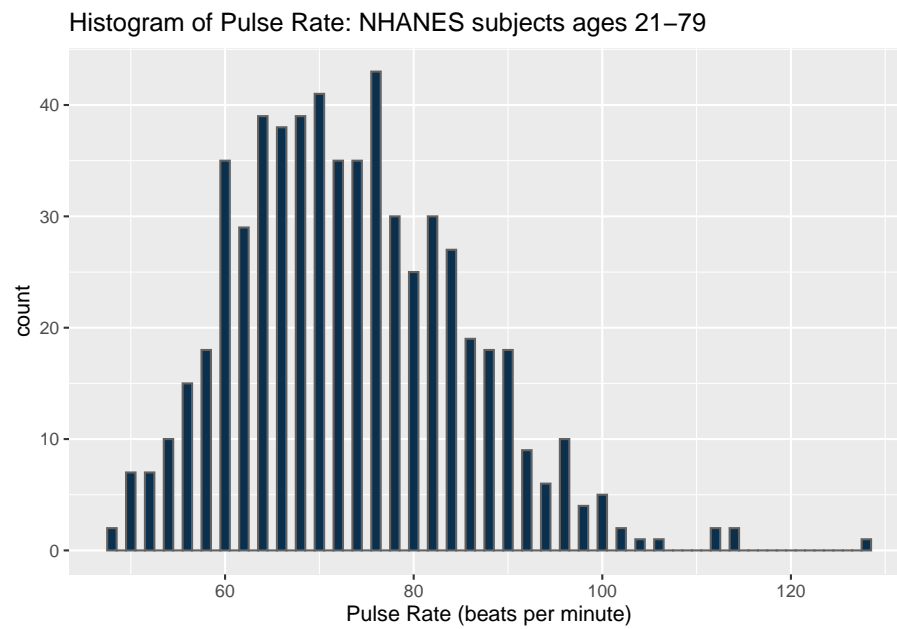
Suppose we instead bin up groups of 5 beats per minute together as we plot the Pulse rates.

```
ggplot(data = nh_data_2179, aes(x = Pulse)) +
    geom_histogram(binwidth = 5, fill = cwru.blue, col = cwru.gray) +
    labs(title = "Histogram of Pulse Rate: NHANES subjects ages 21-79",
        x = "Pulse Rate (beats per minute)")
```

Histogram of Pulse Rate: NHANES subjects ages 21–79
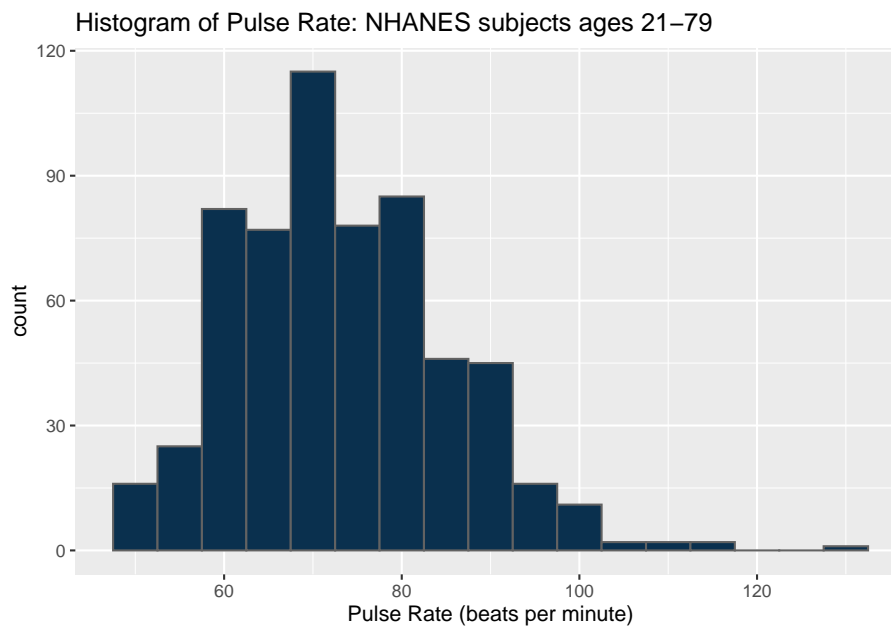
Which is the more useful representation will depend a lot on what questions you're trying to answer.

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_data_2179, aes(y = Pulse, x = PhysActive, fill = PhysActive)) +
    geom_boxplot() +
    labs(title = "Pulse Rate by Physical Activity Status for NHANES ages 21-79")
```

Pulse Rate by Physical Activity Status for NHANES ages 21−79
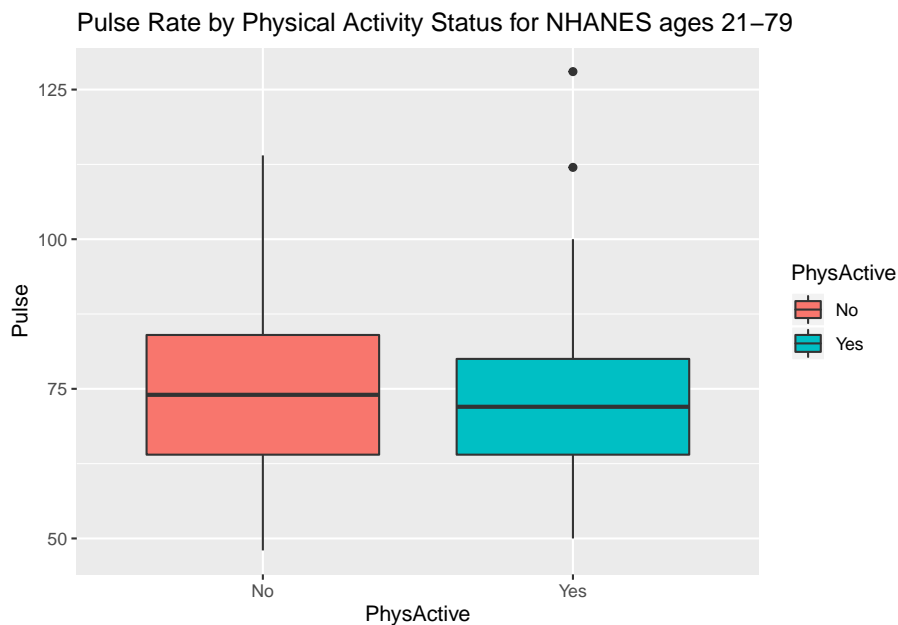
As an accompanying numerical summary, we might ask how many people fall into each of these `PhysActive` categories, and what is their "average" `Pulse` rate.

```
nh_data_2179 %>%
    group_by(PhysActive) %>%
    summarise(count = n(), mean(Pulse), median(Pulse)) %>%
    knitr::kable(digits = 2)
```

PhysActive	count	mean(Pulse)	median(Pulse)
No	293	74.2	74
Yes	310	72.4	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_data_2179, 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-
         x = "Pulse Rate (beats per minute)") +
    guides(fill = FALSE) +
    facet_grid(SleepTrouble ~ ., labeller = "label_both")
```

Histogram of Pulse Rate by Sleep Trouble for NHANES subjects ages 21–79
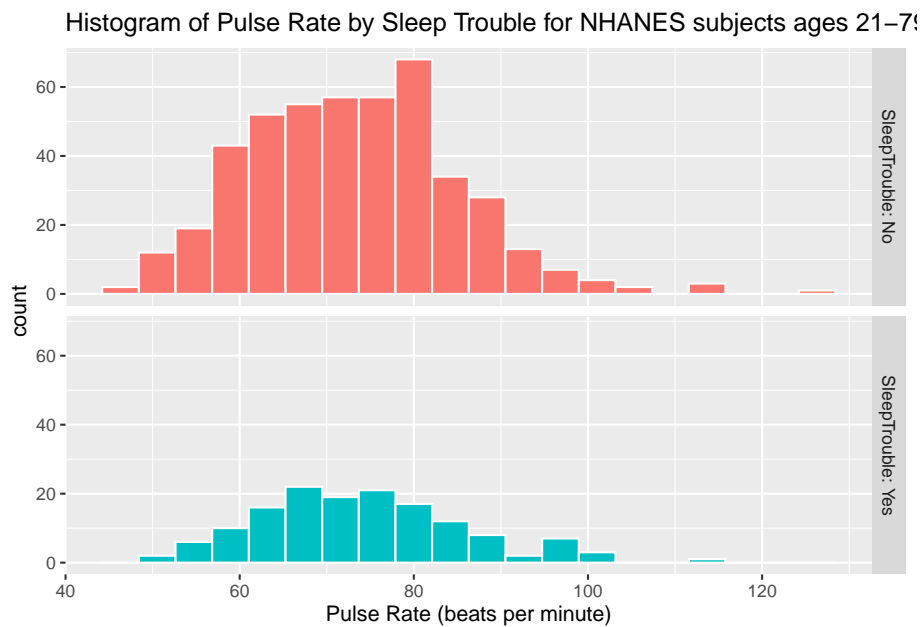
How many people fall into each of these `SleepTrouble` categories, and what is their "average" Pulse rate?

```
nh_data_2179 %>%
    group_by(SleepTrouble) %>%
    summarise(count = n(), mean(Pulse), median(Pulse)) %>%
    knitr::kable(digits = 2)
```

SleepTrouble	count	mean(Pulse)	median(Pulse)
No	457	73	72
Yes	146	74	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_data_2179, aes(x = HealthGen, y = Pulse, fill = HealthGen)) +
    geom_boxplot() +
    labs(title = "Pulse by Self-Reported Overall Health for NHANES ages 21-79",
        x = "Pulse Rate") +
    guides(fill = FALSE)
```

Pulse by Self−Reported Overall Health for NHANES ages 21−79
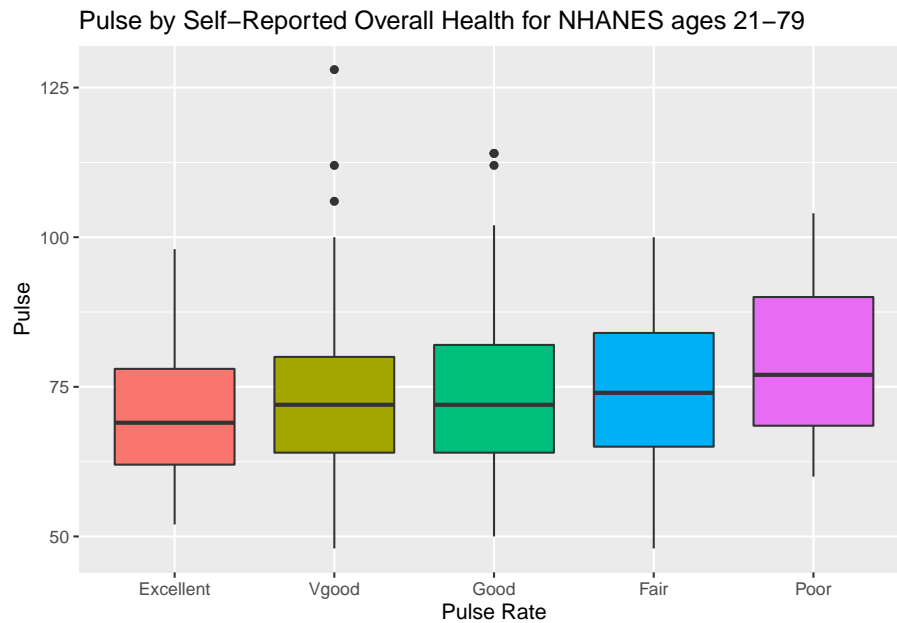
How many people fall into each of these `HealthGen` categories, and what is their "average" Pulse rate?

```
nh_data_2179 %>%
    group_by(HealthGen) %>%
    summarise(count = n(), mean(Pulse), median(Pulse)) %>%
    knitr::kable(digits = 2)
```

HealthGen	count	mean(Pulse)	median(Pulse)
Excellent	64	70.0	69
Vgood	196	72.8	72
Good	238	73.7	72
Fair	83	74.2	74
Poor	22	79.1	77

3.8.4   Pulse Rate and Systolic Blood Pressure

```
ggplot(data = nh_data_2179, aes(x = SBP, y = Pulse)) +
    geom_point() +
    geom_smooth(method = "loess") +
    labs(title = "SBP vs. Pulse rate for NHANES subjects, ages 21-79")
```

SBP vs. Pulse rate for NHANES subjects, ages 21–79
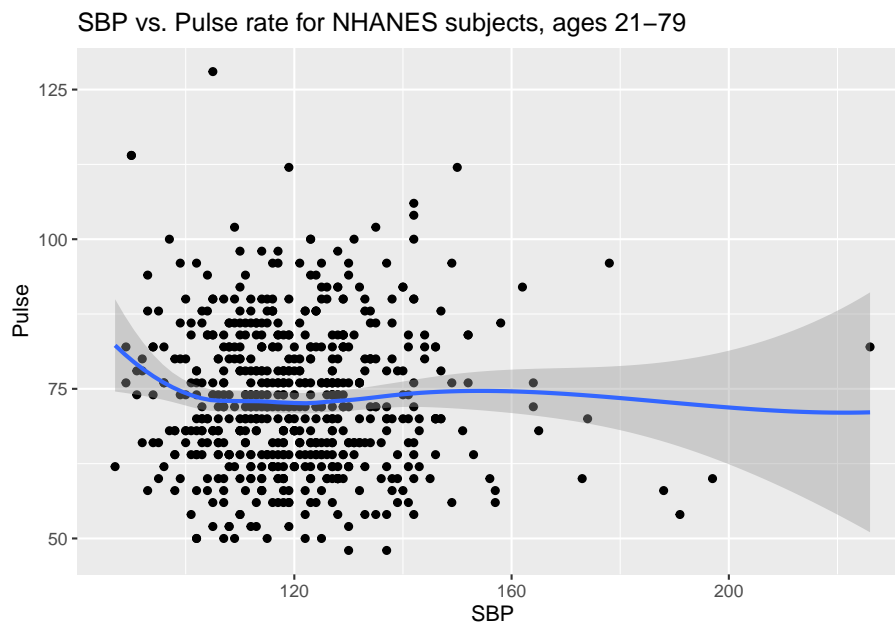
3.8.5 Sleep Troubls vs. No Sleep Trouble?

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_data_2179,
       aes(x = SBP, y = Pulse,
           color = SleepTrouble, shape = SleepTrouble)) +
    geom_point() +
    geom_smooth(method = "loess") +
    labs(title = "SBP vs. Pulse rate for NHANES subjects, ages 21-79")
```
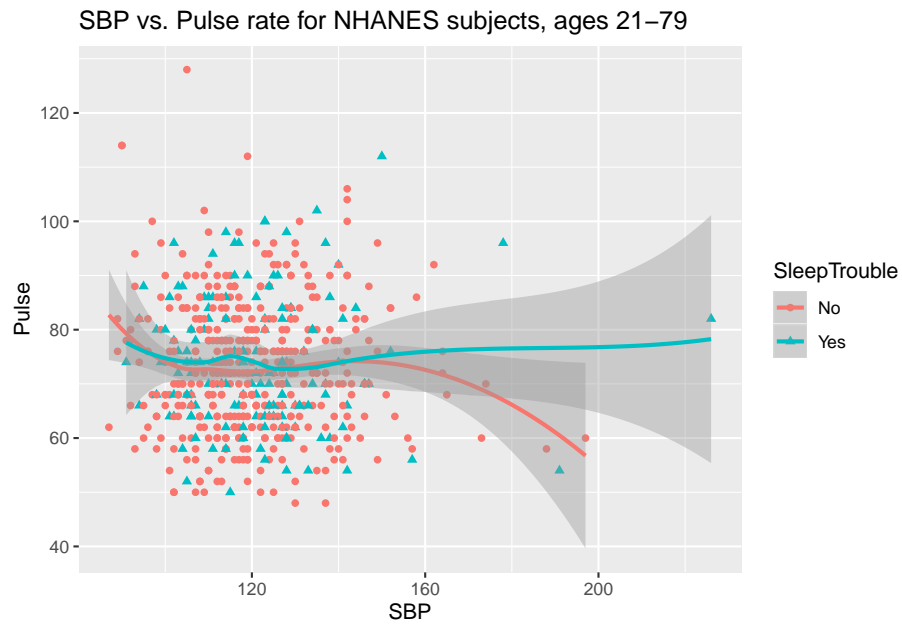
This plot might be easier to interpret if we faceted by `SleepTrouble`, as well.

```
ggplot(data = nh_data_2179,
       aes(x = SBP, y = Pulse,
           color = SleepTrouble, shape = SleepTrouble)) +
    geom_point() +
    geom_smooth(method = "loess") +
    labs(title = "SBP vs. Pulse rate for NHANES subjects, ages 21-79") +
    facet_wrap(~ SleepTrouble, labeller = "label_both")
```

SBP vs. Pulse rate for NHANES subjects, ages 21–79
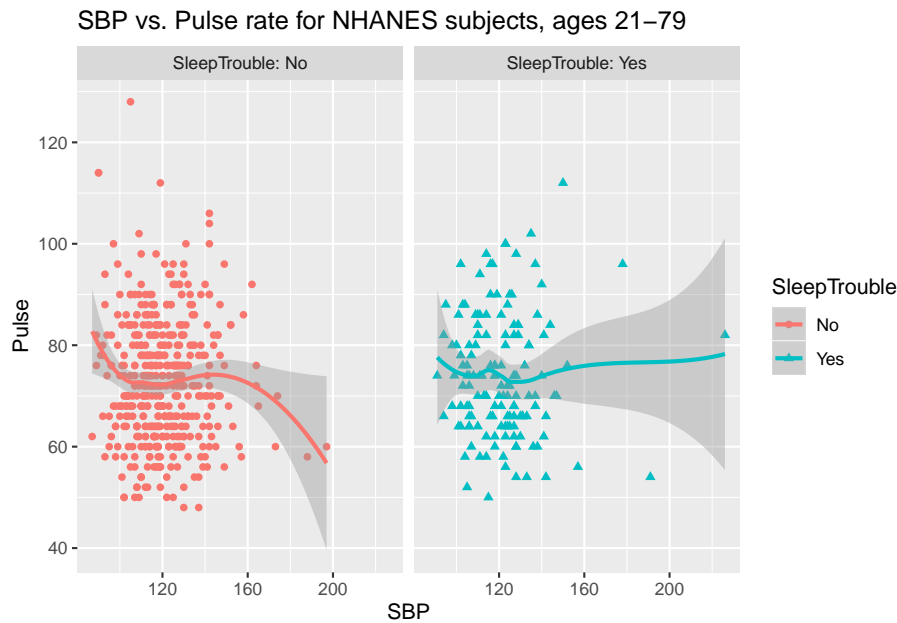
3.9 General Health Status

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_data_2179 %>%
    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_data_2179 %>%
    tabyl(HealthGen)
```

```
HealthGen    n percent
Excellent   64  0.1061
    Vgood  196  0.3250
     Good  238  0.3947
     Fair   83  0.1376
     Poor   22  0.0365
```

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.

```r
nh_data_2179 %>%
    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.

```r
nh_data_2179 %>%
    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.

```r
ggplot(data = nh_data_2179, aes(x = HealthGen)) +
    geom_bar()
```
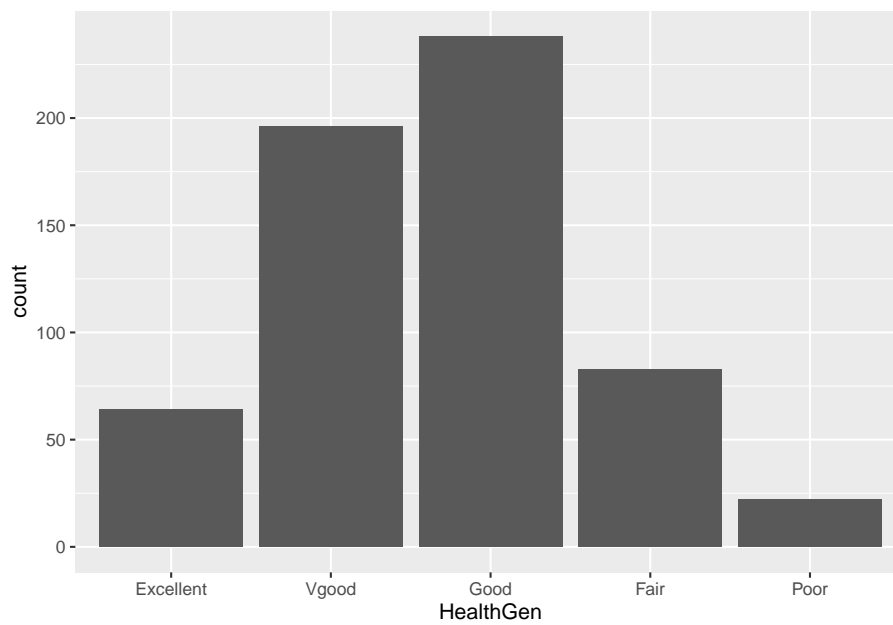
There are lots of things we can do to make this plot fancier.

```
ggplot(data = nh_data_2179, 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")
```
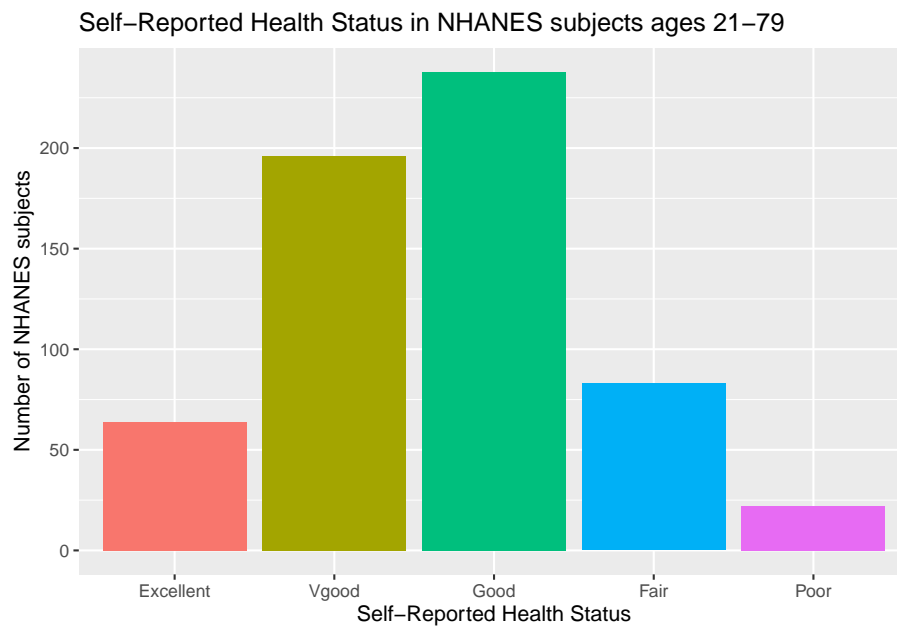
Or, we can really go crazy...

```r
nh_data_2179 %>%
    count(HealthGen) %>%
    ungroup() %>%
    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, '%')),  # prettify
            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()
```

Self−Reported Health Status in NHANES subjects ages 21−79
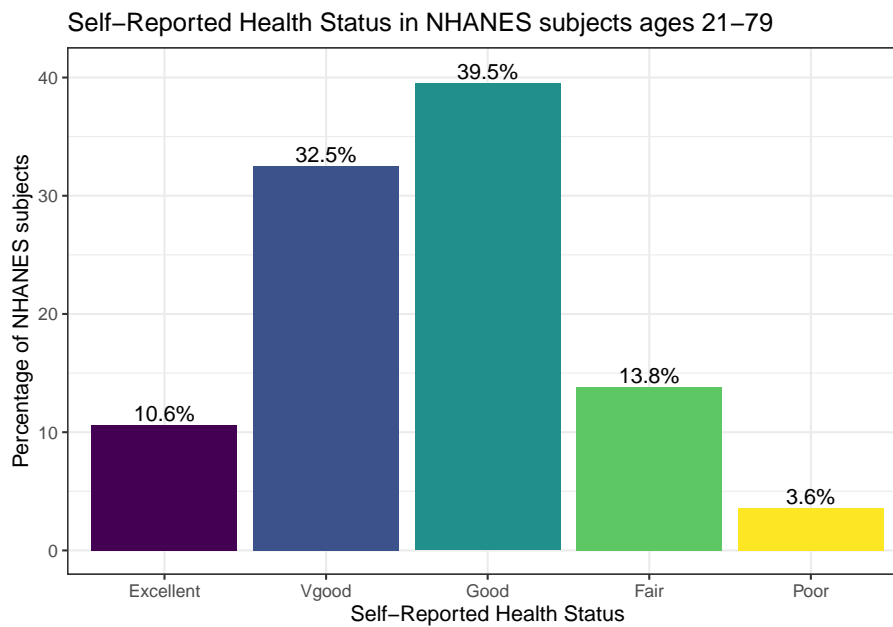
3.9.2 Working with Tables

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

```
nh_data_2179 %>%
    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_data_2179 %>%
    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_data_2179 %>%
    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_data_2179 %>%
    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_data_2179, 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)
```

SBP by Health Status, Overall Health for NHANES ages 21–79
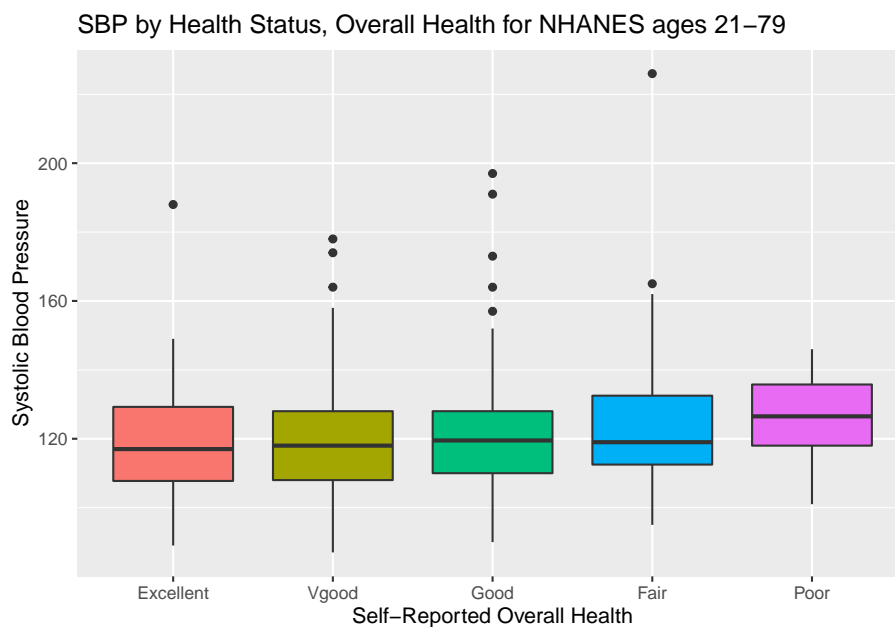
We can see that not too many people self-identify with the "Poor" health category.

```
nh_data_2179 %>%
    group_by(HealthGen) %>%
    summarise(count = n(), mean(SBP), median(SBP)) %>%
    knitr::kable()
```

HealthGen	count	mean(SBP)	median(SBP)
Excellent	64	119	117
Vgood	196	119	118
Good	238	120	120
Fair	83	124	119
Poor	22	126	126

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

```
ggplot(data = nh_data_2179, aes(x = HealthGen, y = SBP, fill = HealthGen)) +
    geom_boxplot() +
    labs(title = "SBP by Health Status, Overall Health for NHANES ages 21-79",
```

```
        y = "Body-mass index", x = "Self-Reported Overall Health") +
guides(fill = FALSE) +
facet_wrap(~ PhysActive, labeller = "label_both") +
coord_flip()
```
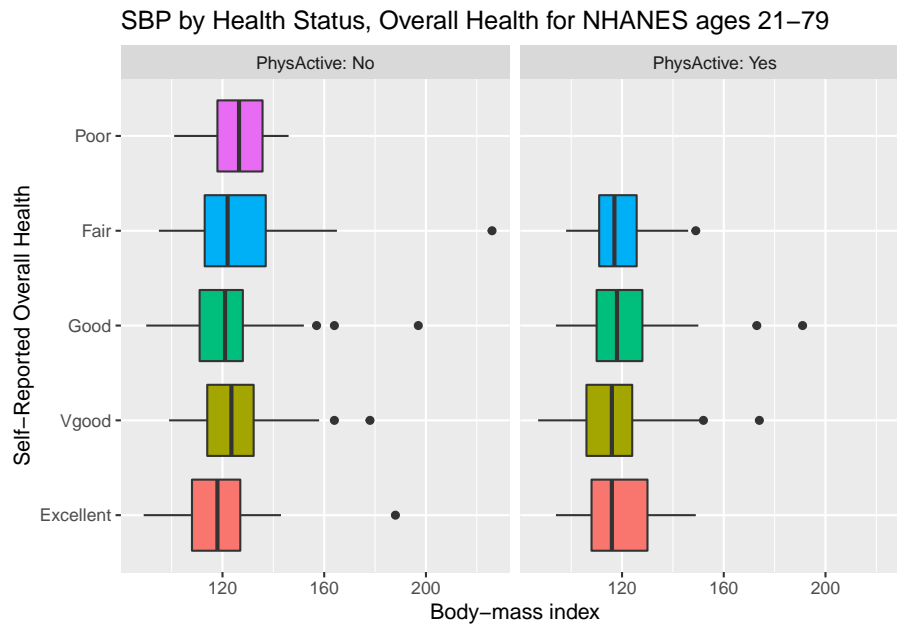

3.9.5   SBP by Sleep Trouble and General Health Status

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_data_2179, aes(x = SBP, fill = Sex)) +
    geom_histogram(color = "white", bins = 20) +
    labs(title = "SBP by Sex, Overall Health for NHANES ages 21-79",
         x = "Body-mass index") +
    guides(fill = FALSE) +
    facet_grid(HealthGen ~ Sex)
```

SBP by Sex, Overall Health for NHANES ages 21–79
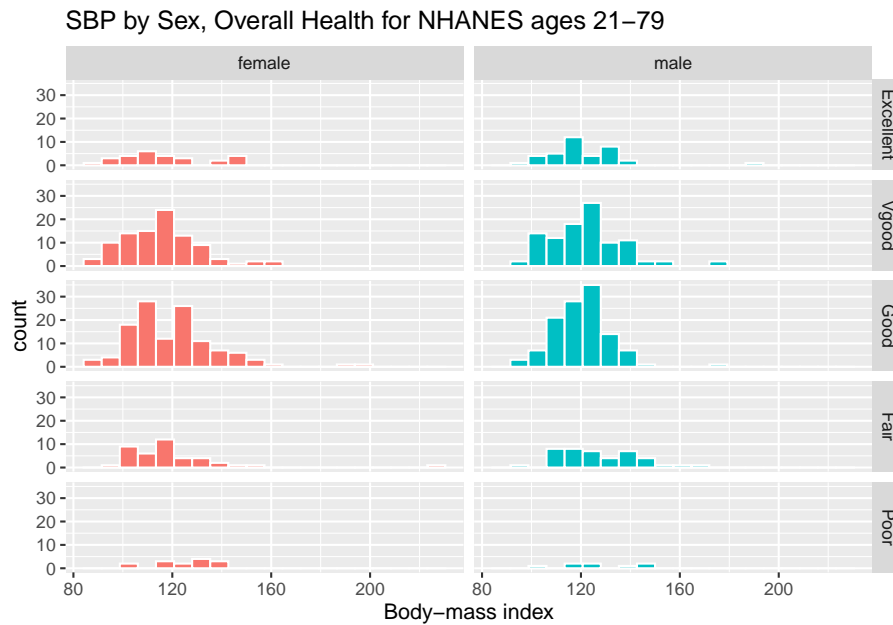
3.10 Conclusions

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, Velleman, and De Veaux (2004) point out, the truly critical notion, of course, is that data values, no matter what kind, are useless without their contexts. The Five W's (Who, What [and in what units], When, Where, Why, and often How) are just as useful for establishing the context of data as they are in journalism. If you can't answer Who and What, in particular, you don't have any useful information.

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

4.2   A New NHANES Adult Sample

In previous work, 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`.

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

```
#   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)
```

```
Classes 'tbl_df', 'tbl' and 'data.frame':   500 obs. of  14 variables:
 $ ID           : int  71531 68613 67064 63924 62840 68058 68936 71189 69936 70687 ...
 $ Sex          : Factor w/ 2 levels "female","male": 2 1 2 1 2 2 1 2 1 2 ...
 $ Age          : int  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  22.4 27.7 26.6 41.9 35.8 30.6 30.5 25.6 21.8 25.5 ...
 $ SBP          : int  143 119 110 98 127 NA 119 112 126 149 ...
 $ DBP          : int  90 86 76 56 0 NA 69 70 80 89 ...
 $ Pulse        : int  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
  <int> <fct> <int> <fct> <fct>      <dbl> <int> <int> <int> <fct>
1 66770 fema~    22 White Some Col~   44.6   100    90    92 Yes
2 68754 male     57 White Some Col~   23.2   124    85    82 No
3 70911 male     59 White College ~   24.5   118    57    76 No
4 71393 male     27 White High Sch~   25.7   116    61    88 Yes
5 70458 fema~    35 Black 9 - 11th~   21.9   115    64    84 No
```

```
# ... with 4 more variables: Smoke100 <fct>, 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)
```

```
Observations: 500
Variables: 14
$ ID            <int> 71531, 68613, 67064, 63924, 62840, 68058, 68936,...
$ Sex           <fct> male, female, male, female, male, male, female, ...
$ Age           <int> 35, 61, 31, 29, 60, 50, 36, 51, 54, 59, 59, 27, ...
$ Race          <fct> White, White, White, Black, White, White, Black,...
$ Education     <fct> Some College, Some College, College Grad, High S...
$ BMI           <dbl> 22.4, 27.7, 26.6, 41.9, 35.8, 30.6, 30.5, 25.6, ...
$ SBP           <int> 143, 119, 110, 98, 127, NA, 119, 112, 126, 149, ...
$ DBP           <int> 90, 86, 76, 56, 0, NA, 69, 70, 80, 89, 75, 78, 6...
$ Pulse         <int> 84, 112, 86, 74, 110, NA, 60, 54, 78, 62, 82, 68...
$ PhysActive    <fct> Yes, No, Yes, No, No, No, No, Yes, Yes, Yes, No,...
$ 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, No, ...
$ MaritalStatus <fct> NeverMarried, Widowed, Married, Separated, Marri...
$ HealthGen     <fct> Good, Vgood, Good, Fair, Poor, Good, Good, NA, G...
```

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

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

---

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

Variable	Description	Sample Values
MaritalStatus	Marital Status	Married, Divorced
HealthGen	Self-report general health rating (5 lev.)	Vgood, Good

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.

A more detailed summary can be obtained using either the `summary` function applied to our data frame, or perhaps the `skim` function from the `skimr` package.

```
summary(nh_adults)
```

```
      ID             Sex           Age              Race
 Min.   :62199   female:221   Min.   :21.0   Asian   : 42
 1st Qu.:64522   male  :279   1st Qu.:31.0   Black   : 63
 Median :67192                Median :42.0   Hispanic: 26
 Mean   :67122                Mean   :41.9   Mexican : 38
 3rd Qu.:69654                3rd Qu.:53.0   White   :313
 Max.   :71911                Max.   :64.0   Other   : 18


          Education        BMI            SBP            DBP
 8th Grade      : 24   Min.   :17.3   Min.   : 84   Min.   :  0.0
 9 - 11th Grade: 60   1st Qu.:23.8   1st Qu.:110   1st Qu.: 66.0
 High School    : 81   Median :27.5   Median :118   Median : 72.0
 Some College  :153   Mean   :28.5   Mean   :119   Mean   : 72.1
 College Grad  :182   3rd Qu.:31.6   3rd Qu.:127   3rd Qu.: 78.0
                      Max.   :63.3   Max.   :209   Max.   :103.0
                      NA's   :5      NA's   :15    NA's   :15
     Pulse        PhysActive Smoke100  SleepTrouble    MaritalStatus
 Min.   : 40.0   No :215    No :297   No :380      Divorced    : 51
 1st Qu.: 64.0   Yes:285    Yes:203   Yes:120      LivePartner : 51
 Median : 72.0                                     Married     :259
 Mean   : 73.4                                     NeverMarried:112
 3rd Qu.: 82.0                                     Separated   : 16
 Max.   :112.0                                     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.

```
skimr::skim(nh_adults)
```

```
Skim summary statistics
 n obs: 500
 n variables: 14

-- Variable type:factor -------------------------------------
      variable missing complete   n n_unique
     Education       0      500 500        5
      HealthGen      49      451 500        5
 MaritalStatus       0      500 500        6
    PhysActive       0      500 500        2
          Race       0      500 500        6
           Sex       0      500 500        2
  SleepTrouble       0      500 500        2
     Smoke100       0      500 500        2
                        top_counts ordered
 Col: 182, Som: 153, Hig: 81, 9 -: 60    FALSE
 Goo: 184, Vgo: 154, Exc: 50, Fai: 49    FALSE
 Mar: 259, Nev: 112, Div: 51, Liv: 51    FALSE
           Yes: 285, No: 215, NA: 0    FALSE
 Whi: 313, Bla: 63, Asi: 42, Mex: 38    FALSE
           mal: 279, fem: 221, NA: 0    FALSE
           No: 380, Yes: 120, NA: 0    FALSE
           No: 297, Yes: 203, NA: 0    FALSE

-- Variable type:integer ------------------------------------
variable missing complete   n     mean       sd    p0      p25    p50
     Age       0      500 500    41.91    12.35    21      31     42
     DBP      15      485 500    72.13     11.1     0      66     72
      ID       0      500 500 67121.85 2871.51 62199 64521.75 67192
   Pulse      15      485 500    73.41    12.01    40      64     72
     SBP      15      485 500   119.25    15.26    84     110    118
    p75  p100      hist
    53     64 <U+2587><U+2587><U+2587><U+2587><U+2587><U+2587><U+2587><U+2586>
    78    103 <U+2581><U+2581><U+2581><U+2581><U+2582><U+2587><U+2583><U+2581>
```

```
69653.75 71911 <U+2586><U+2585><U+2585><U+2586><U+2585><U+2586><U+2587><U+2585>
      82       112 <U+2581><U+2582><U+2585><U+2587><U+2585><U+2582><U+2581><U+2581>
     127       209 <U+2582><U+2587><U+2587><U+2583><U+2581><U+2581><U+2581><U+2581>

-- Variable type:numeric -----------------------------------
 variable missing complete   n mean  sd  p0  p25  p50  p75 p100    hist
      BMI       5        495 500 28.48 6.3 17.3 23.8 27.5 31.6 63.3 <U+2583><U+2587><U+2586><U+2581
```

4.3 Types of Variables

4.3.1 Quantitative Variables

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

- It does little good to be 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/m², 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 are measurements are fairly crude (this is certainly true for `Age`, measured in years) although BMI is probably *continuous* in most settings, even though it is a function of two other measures (`Height` and `Weight`) which are rounded off to integer numbers of centimeters and kilograms, respectively.

- It is also possible to separate out quantitative variables into **ratio** variables or **interval** variables. An interval variable has equal distances between values, but the zero point is arbitrary. A ratio variable has equal intervals between values, and a meaningful zero point. For example, weight is an example of a ratio variable, while IQ is an example of an interval variable. We all know what zero weight is. An intelligence score like IQ is a different matter. We say that the average IQ is 100, but that's only by convention. We could just as easily have decided to add 400 to every IQ value and make the average 500 instead. Because IQ's intervals are equal, the difference between and IQ of 70 and an IQ of 80 is the same as the difference between 120 and 130. However, an IQ of 100 is not twice as high as an IQ of 50. The point is that if the zero point is artificial and 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.

4.3.1.1   A look at BMI (Body-Mass Index)

The definition of BMI (*body-mass index*) for adult subjects (which is expressed in units of kg/m²) is:

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

[BMI is essentially] . . . a measure of a person's *thinness* or *thickness*. . . BMI was designed for use as a simple means of classifying

average sedentary (physically inactive) populations, with an average body composition. For these individuals, the current value recommendations are as follow: a BMI from 18.5 up to 25 may indicate optimal weight, a BMI lower than 18.5 suggests the person is underweight, a number from 25 up to 30 may indicate the person is overweight, and a number from 30 upwards suggests the person is obese.

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

4.3.2 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.

4.4   Creating a Data Report Quickly

The `DataExplorer` package has a function called `create_report` that you might like. It is easy to run, but the results can be a bit overwhelming, so I'll just show the code, but not run the report here.

```
DataExplorer::create_report(nh_adults)
```

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.   :21.0    Min.   :17.3    Min.   : 84    Min.   :  0.0
 1st Qu.:31.0    1st Qu.:23.8    1st Qu.:110    1st Qu.: 66.0
 Median :42.0    Median :27.5    Median :118    Median : 72.0
 Mean   :41.9    Mean   :28.5    Mean   :119    Mean   : 72.1
 3rd Qu.:53.0    3rd Qu.:31.6    3rd Qu.:127    3rd Qu.: 78.0
 Max.   :64.0    Max.   :63.3    Max.   :209    Max.   :103.0
                 NA's   :5       NA's   :15     NA's   :15
     Pulse
```

```
 Min.   : 40.0
 1st Qu.: 64.0
 Median : 72.0
 Mean   : 73.4
 3rd Qu.: 82.0
 Max.   :112.0
 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** (25$^{\text{th}}$ percentile) for each variable - for example, 25% of the subjects were Age 31 or younger.
- `Median` = the **median** (50$^{\text{th}}$ 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** (75$^{\text{th}}$ 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
```

---

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

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 21 22 22 22 22 22 22 22 22 22 22 22 22 23 23 23
 [24] 23 23 23 23 23 23 23 23 24 24 24 24 24 24 24 24 24 24 24 24 24 24 24
 [47] 25 25 25 25 25 25 25 25 25 26 26 26 26 26 26 26 26 26 26 26 26 26 27
 [70] 27 27 27 27 27 27 27 27 27 27 27 27 27 27 28 28 28 28 28 28 28 28 28
 [93] 28 28 28 28 29 29 29 29 29 29 29 30 30 30 30 30 30 30 30 30 30 30 30
[116] 30 30 30 31 31 31 31 31 31 31 31 31 31 31 31 32 32 32 32 32 32 32 32
[139] 32 32 32 32 32 32 33 33 33 33 33 33 33 33 33 33 33 33 33 34 34 34 34
[162] 34 34 34 35 35 35 35 35 35 35 35 35 36 36 36 36 36 36 36 37 37 37 37
[185] 37 37 37 37 37 37 37 37 37 37 37 37 37 37 38 38 38 38 38 38 38 38 38
[208] 38 38 38 39 39 39 39 39 39 39 39 39 39 39 39 39 40 40 40 40 40 40 40
[231] 40 40 40 40 41 41 41 41 41 41 41 41 41 41 41 41 42 42 42 42 42 42 42
[254] 42 42 42 42 42 43 43 43 43 43 43 43 43 43 43 43 43 43 43 43 44 44 44
[277] 44 44 44 44 44 44 44 44 45 45 45 45 45 45 45 45 45 45 45 45 45 45 45
[300] 46 46 46 46 46 46 46 47 47 47 47 47 47 47 47 47 47 48 48 48 48 48 48
```

```
[323] 49 49 49 49 49 49 49 49 49 49 49 49 49 49 49 49 49 50 50 50 50 50 50
[346] 50 50 50 50 50 50 50 50 50 51 51 51 51 51 51 51 51 51 52 52 52 52 52
[369] 52 52 52 53 53 53 53 53 53 53 53 53 53 53 53 53 53 54 54 54 54 54 54
[392] 54 54 54 54 54 54 55 55 55 55 55 55 55 55 55 55 55 56 56 56 56 56 56
[415] 56 56 56 56 56 56 56 56 56 57 57 57 57 57 57 57 57 57 57 57 58 58 58
[438] 58 58 58 58 58 58 59 59 59 59 59 59 59 59 59 59 59 59 59 59 59 60 60
[461] 60 60 60 60 60 60 60 60 60 61 61 61 61 61 61 61 61 61 61 61 62 62 62
[484] 62 62 63 63 63 63 63 63 63 63 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 values[2].

```
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
```

---

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

```
`mean(Pulse, na.rm = TRUE)` `median(Pulse, na.rm = TRUE)`
                      <dbl>                          <int>
1                      73.4                             72
```

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))
```

```
# 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]

---

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

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 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.   :21.0   Min.   :17.3   Min.   : 84   Min.   :  0.0
 1st Qu.:31.0   1st Qu.:23.8   1st Qu.:110   1st Qu.: 66.0
 Median :42.0   Median :27.5   Median :118   Median : 72.0
 Mean   :41.9   Mean   :28.5   Mean   :119   Mean   : 72.1
 3rd Qu.:53.0   3rd Qu.:31.6   3rd Qu.:127   3rd Qu.: 78.0
 Max.   :64.0   Max.   :63.3   Max.   :209   Max.   :103.0
                NA's   :5      NA's   :15    NA's   :15
     Pulse
 Min.   : 40.0
 1st Qu.: 64.0
 Median : 72.0
 Mean   : 73.4
 3rd Qu.: 82.0
 Max.   :112.0
 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  12.3  6.30  15.3  11.1  12.0
```

5.3.4   Defining the Variance and Standard Deviation

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

In thinking about spread, we might consider how far each data value is from the mean. Such a difference is called a *deviation*. We could just average the deviations, but the positive and negative differences always cancel out, leaving an average deviation of zero, so that's not helpful. Instead, we *square* each deviation to obtain non-negative values, and to emphasize larger differences. When we add up these squared deviations and find their mean (almost), this yields the **variance**.

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

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

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

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

5.3.5   Empirical Rule Interpretation of the Standard Deviation

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

---

[4]When we divide by n-1 as we calculate the sample variance, the average of the sample variances for all possible samples is equal to the population variance. If we instead divided by n, the average sample variance across all possible samples would be a little smaller than the population variance.
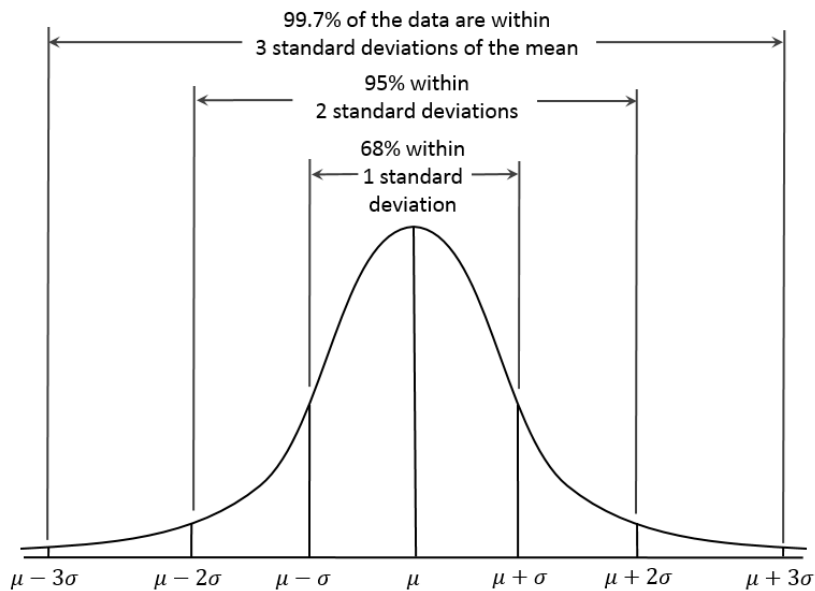
Figure 5.1: The Normal Distribution and the Empirical Rule

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

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

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

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
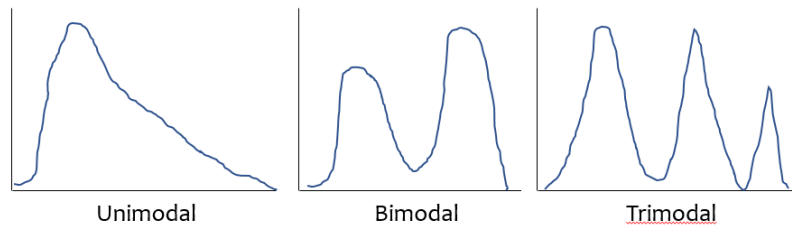
Figure 5.2: Unimodal and Multimodal Sketches

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.

Truly multimodal distributions are usually described that way in terms of shape. For unimodal distributions, skewness and kurtosis become useful ideas.
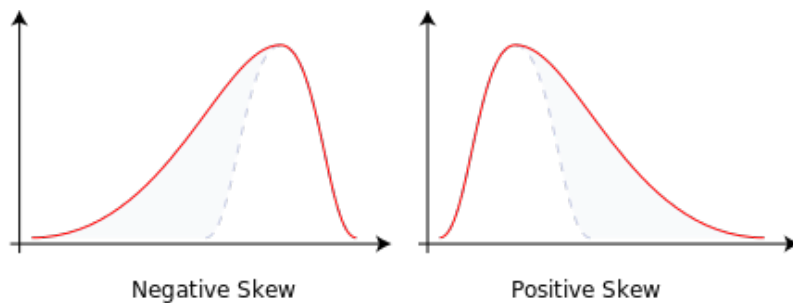
Figure 5.3: Negative (Left) Skew and Positive (Right) Skew

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.*

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 distribution, with lighter tails (fewer observations far from the center) than we'd expect from a Normal distribution.
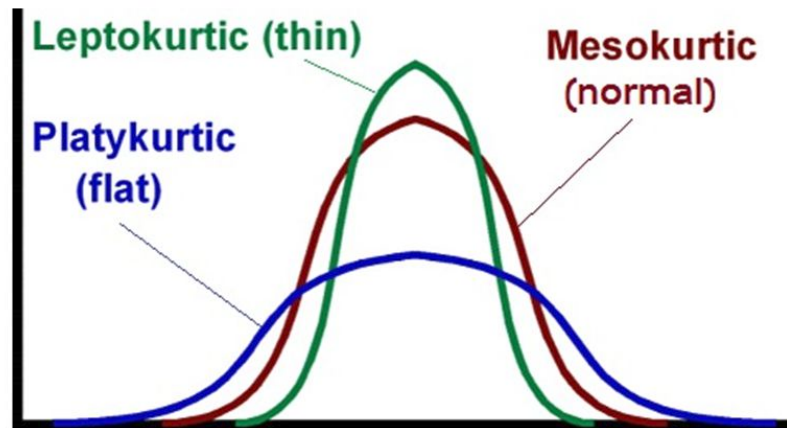
Figure 5.4: The Impact of Kurtosis

- A platykurtic distribution is a flatter distribution, with heavier tails (more observations far from the center) than we'd expect from a Normal distribution.

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 15.3 485      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>   <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    122 132 167  124 18.9  22       2
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       4
4    Some College  85 110    119 128 165  119 13.5 149       4
5    College Grad  84 109    118 126 171  118 14.3 177       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)))
```

```
# A tibble: 5 x 10
  Education         min    Q1 median    Q3   max  mean    sd     n missing
  <fct>           <int> <dbl>  <dbl> <dbl> <int> <dbl> <dbl> <int>   <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 skim in the skimr package

```
nh_adults %>%
    select(Pulse) %>%
    skimr::skim()
```

```
Skim summary statistics
```

```
 n obs: 500
 n variables: 1

-- Variable type:integer ------------------------------------
 variable missing complete   n  mean     sd p0 p25 p50 p75 p100      hist
    Pulse      15        485 500 73.41 12.01 40  64  72  82  112 <U+2581><U+2582><U+2585>
```

5.5.3 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  41.9 12.3   42.0    41.9 16.31 21.0  64.0    43  0.03
BMI       2 495  28.5  6.3   27.5    27.8  5.63 17.3  63.3    46  1.35
SBP       3 485 119.2 15.3  118.0   118.2 13.34 84.0 209.0   125  1.27
DBP       4 485  72.1 11.1   72.0    72.3  8.90  0.0 103.0   103 -0.58
Pulse     5 485  73.4 12.0   72.0    73.0 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 $Y_1, Y_2, ..., Y_n$, then the mad is defined as $median(|Y_i - median(Y_i)|)$.
  - To find the mad for a set of numbers, find the median, subtract the median from each value and find the absolute value of that difference, and then find the median of those absolute differences.
  - For non-normal data with a skewed shape but tails well approximated by the Normal, the mad is likely to be a better (more robust) estimate of the spread than is the standard deviation.
- 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 `kurtosis`, which refers to how outlier-prone, or heavy-tailed the shape of the distribution is, mainly 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.

5.5.4 `describe` in the `Hmisc` package

```
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.

5.5.5   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.

As mentioned previously, `DataExplorer` can be used for more automated exploratory data analyses (and don't forget about `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 (@ref(createnh_adults)), 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))
```

```
Warning: Factor `HealthGen` contains implicit NA, consider using
`forcats::fct_explicit_na`
```

```
# 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

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

```
nh_adults %>% select(Sex, Race, Education, PhysActive, Smoke100, SleepTrouble, HealthGe

 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

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

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

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

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

Value       Separated     Widowed
Frequency          16          11
Proportion      0.032       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?

```r
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...

```r
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:

```r
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 (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
Widowed	1 (9.1%)	3 (27.3%)	2 (18.2%)	1 (9.1%)	3 (27.3%)	1 (9.
Total	50 (10.0%)	154 (30.8%)	184 (36.8%)	49 (9.8%)	14 (2.8%)	49 (9

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 Health-Gen:

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

```
          HealthGen
Race        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         No                     99  62
           Yes                    21  33
Yes        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)	1,167
Arizona	10.7	(8.3-13.6)	1,520
Arkansas	17.8	(15.7-20.1)	1,470
Connecticut	12.3	(10.2-14.7)	2,270
Delaware	14.2	(12.9-15.6)	2,475
Florida	11.6	(10.5-12.8)	5,491
. . .			
Wisconsin	11.6	(9.7-13.9)	2,771
Wyoming	10.7	(9.4-12.2)	2,910

or . . .

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

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://wwwn.cdc.gov/nchs/nhanes/search/nnyfs12.aspx.

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

- Demographics
- Dietary
- Examination and
- Questionnaire

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

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_skinfold	BMXCALFF	Calf skinfold (mm)
triceps_skinfold	BMXTRI	Triceps skinfold (mm)
subscapular_skinfold	BMXSUB	Subscapular skinfold (mm)

From the `Y_PLX` file of Plank test results:

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

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 ($\geq$ 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

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	Generall 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 the Data Set

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

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

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

```
[1] 1518    45
```

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

`nnyfs` *# 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
   <dbl> <chr>      <dbl> <chr>         <dbl> <chr>          <dbl>
 1 71917 Fema~         15 3_Black~          9 English        28299.
 2 71918 Fema~          8 3_Black~          2 English        15127.
 3 71919 Fema~         14 2_White~          8 English        29977.
 4 71920 Fema~         15 2_White~          8 English        80652.
 5 71921 Male           3 2_White~         NA English        55592.
 6 71922 Male          12 1_Hispa~          6 English        27365.
 7 71923 Male          12 2_White~          5 English        86673.
 8 71924 Fema~          8 4_Other~          2 English        39549.
 9 71925 Male           7 1_Hispa~          0 English        42333.
10 71926 Male           8 3_Black~          2 English        15307.
# ... with 1,508 more rows, and 38 more variables: income_pov <dbl>,
#   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)`

```
Classes 'tbl_df', 'tbl' and 'data.frame':    1518 obs. of  45 variables:
 $ SEQN                : num  71917 71918 71919 71920 71921 ...
 $ sex                 : chr  "Female" "Female" "Female" "Female" ...
 $ age_child           : num  15 8 14 15 3 12 12 8 7 8 ...
 $ race_eth            : chr  "3_Black Non-Hispanic" "3_Black Non-Hispanic" "2_White Non-Hispanic
 $ educ_child          : num  9 2 8 8 NA 6 5 2 0 2 ...
 $ language            : chr  "English" "English" "English" "English" ...
 $ sampling_wt         : num  28299 15127 29977 80652 55592 ...
 $ income_pov          : num  0.21 5 5 0.87 4.34 5 5 2.74 0.46 1.57 ...
 $ age_adult           : num  46 46 42 53 31 42 39 31 45 56 ...
```

```
$ educ_adult         : chr  "2_9-11th Grade" "3_High School Graduate" "5_College Grad
$ respondent         : chr  "Child" "Mom" "Child" "Child" ...
$ salt_used          : chr  "Yes" "Yes" "Yes" "Yes" ...
$ energy             : num  2844 1725 2304 1114 1655 ...
$ protein            : num  169.1 55.2 199.3 14 50.6 ...
$ sugar              : num  128.2 118.7 81.4 119.2 90.3 ...
$ fat                : num  127.9 63.7 86.1 36 53.3 ...
$ diet_yesterday     : chr  "2_Usual" "2_Usual" "2_Usual" "2_Usual" ...
$ water              : num  607 178 503 859 148 ...
$ plank_time         : num  NA 45 121 45 11 107 127 44 184 58 ...
$ height             : num  NA 131.6 172 167.1 90.2 ...
$ weight             : num  NA 38.6 58.7 92.5 12.4 66.4 56.7 22.2 20.9 28.3 ...
$ bmi                : num  NA 22.3 19.8 33.1 15.2 25.9 22.5 14.4 15.9 17 ...
$ bmi_cat            : chr  NA "4_Obese" "2_Normal" "4_Obese" ...
$ arm_length         : num  NA 27.7 38.4 35.9 18.3 34.2 33 26.5 24.2 26 ...
$ waist              : num  NA 71.9 79.4 96.4 46.8 90 72.3 56.1 54.5 59.7 ...
$ arm_circ           : num  NA 25.4 26 37.9 15.1 29.5 27.9 17.6 17.7 19.9 ...
$ calf_circ          : num  NA 32.3 35.3 46.8 19.4 36.9 36.8 24 24.3 27.3 ...
$ calf_skinfold      : num  NA 22 18.4 NA 8.4 22 18.3 7 7.2 8.2 ...
$ triceps_skinfold   : num  NA 19.9 15 20.6 8.6 22.8 20.5 12.9 6.9 8.8 ...
$ subscapular_skinfold: num  NA 17.4 9.8 22.8 5.7 24.4 12.6 6.8 4.8 6.1 ...
$ active_days        : num  3 5 3 3 7 2 5 3 7 7 ...
$ tv_hours           : num  2 2 1 3 2 3 0 4 2 2 ...
$ computer_hours     : num  1 2 3 3 0 1 0 3 1 1 ...
$ physical_last_week : chr  "No" "No" "Yes" "Yes" ...
$ enjoy_recess       : chr  "1_Strongly Agree" "1_Strongly Agree" "3_Neither Agree n
$ meals_out          : num  0 2 3 2 1 1 2 1 0 2 ...
$ insured            : chr  "Has Insurance" "Has Insurance" "Has Insurance" "Has Insu
$ phys_health        : chr  "1_Excellent" "3_Good" "1_Excellent" "3_Good" ...
$ access_to_care     : chr  "Has Usual Care Source" "Has Usual Care Source" "Has Usua
$ care_source        : chr  "Clinic or Health Center" "Doctor's Office" "Doctor's Of
$ asthma_ever        : chr  "Never Had Asthma" "History of Asthma" "Never Had Asthma
$ asthma_now         : chr  "No Asthma Now" "Asthma Now" "No Asthma Now" "Asthma Now
$ med_use            : chr  "No Medications" "Had Medication" "No Medications" "Had
$ med_count          : num  0 1 0 2 0 0 0 0 0 0 ...
$ insurance          : chr  "State Sponsored" "State Sponsored" "Private" "State Spo
```

There are a lot of variables here. Let's run through the first few rather slowly,
and then we'll speed up.

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*.

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

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

Obviously, we don't actually need more than a decimal place here for any real purpose.

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.

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

```
 age_child    n percent
         3  110    7.2%
         4  112    7.4%
         5  114    7.5%
         6  129    8.5%
         7  123    8.1%
         8  112    7.4%
         9   99    6.5%
        10  124    8.2%
        11  111    7.3%
```

```
12 137    9.0%
13 119    7.8%
14 130    8.6%
15  98    6.5%
```

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

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

```
   age_child
 Min.   : 3.00
 1st Qu.: 6.00
 Median : 9.00
 Mean   : 9.03
 3rd Qu.:12.00
 Max.   :15.00
```

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)
```

```
 min Q1 median Q3 max mean   sd    n missing
   3  6      9 12  15 9.03 3.71 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 8.79 3.73 450       0
2   2_White Non-Hispanic  3 6.00    9.0 12  15 9.14 3.80 610       0
3   3_Black Non-Hispanic  3 6.00    9.0 12  15 9.04 3.58 338       0
4 4_Other Race/Ethnicity  3 7.00    9.5 12  15 9.38 3.43 120       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.

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

```
   income_pov
 Min.   :0.0
 1st Qu.:0.9
 Median :1.7
 Mean   :2.2
 3rd Qu.:3.5
 Max.   :5.0
 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).
```
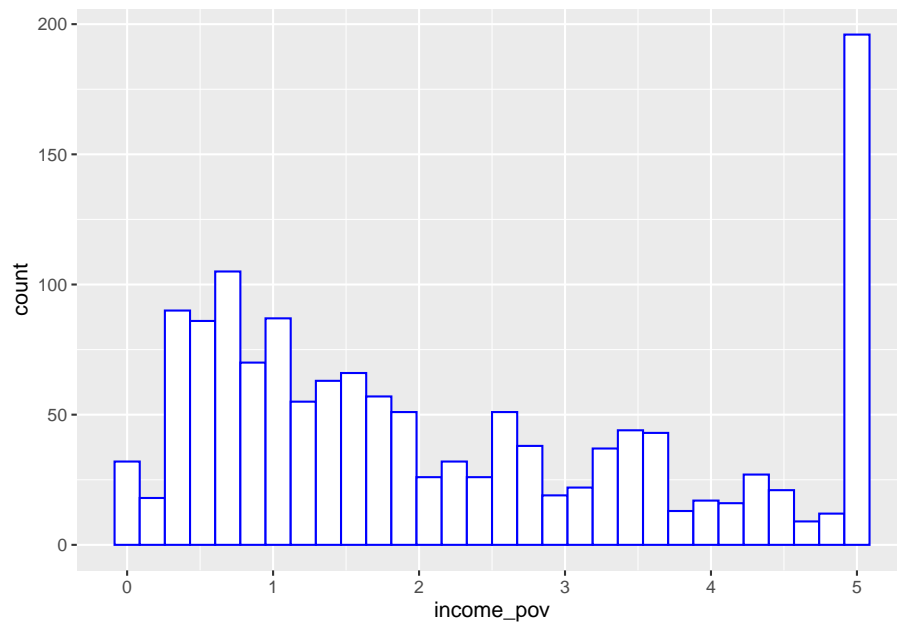
The histogram shows us that the values are truncated at 5, so that children whose actual family income is above 5 times the poverty line are listed as 5. We also see a message reminding us that some of the data are missing for this variable.

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

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

	race_eth	min	Q1	median	Q3	max	mean	sd	n	missing
1	1_Hispanic	0	0.56	0.96	1.74	5	1.34	1.10	409	41
2	2_White Non-Hispanic	0	1.52	2.96	4.52	5	2.92	1.62	588	22
3	3_Black Non-Hispanic	0	0.78	1.57	2.82	5	1.97	1.49	328	10
4	4_Other Race/Ethnicity	0	1.17	2.74	4.58	5	2.85	1.67	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).
```
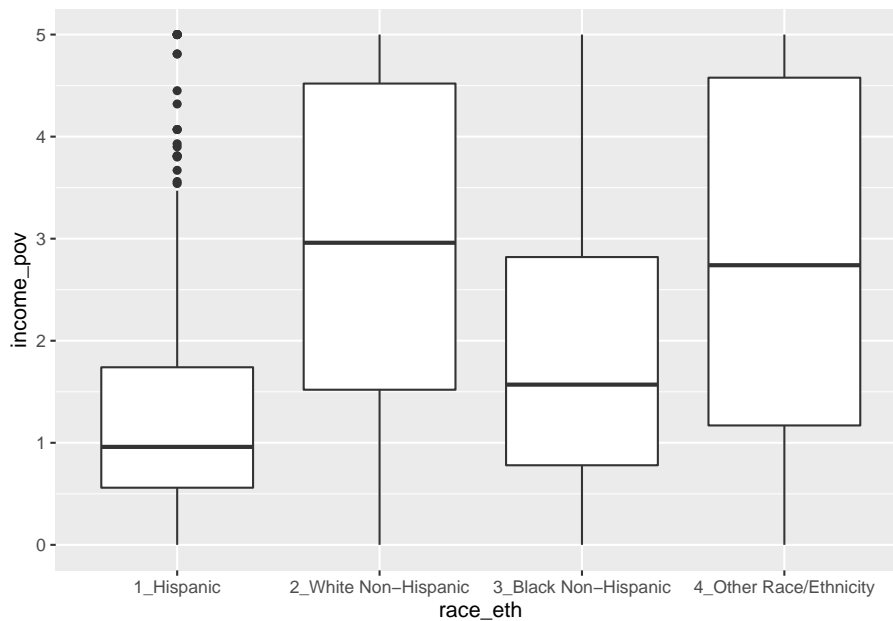
7.2.6 `bmi`

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, BMI = weight in kg / (height in m)^2. This is a continuous concept, measured to as many decimal places as you like, and it has a meaningful zero point, so it's a ratio variable.

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

```
      bmi
 Min.   :11.9
 1st Qu.:15.9
 Median :18.1
 Mean   :19.6
 3rd Qu.:21.9
 Max.   :48.3
 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 $\geq$ 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)
```

```
         bmi_cat  min   Q1 median   Q3  max mean   sd   n missing
1 1_Underweight 11.9 13.4   13.7 15.0 16.5 14.1 1.10  41       0
2      2_Normal 13.5 15.4   16.5 18.7 24.0 17.2 2.30 920       0
3  3_Overweight 16.9 18.3   21.4 23.4 27.9 21.2 2.92 258       0
4       4_Obese 17.9 22.3   26.2 30.2 48.3 26.7 5.72 295       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 76.6 145 67.7 15.2 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).
```
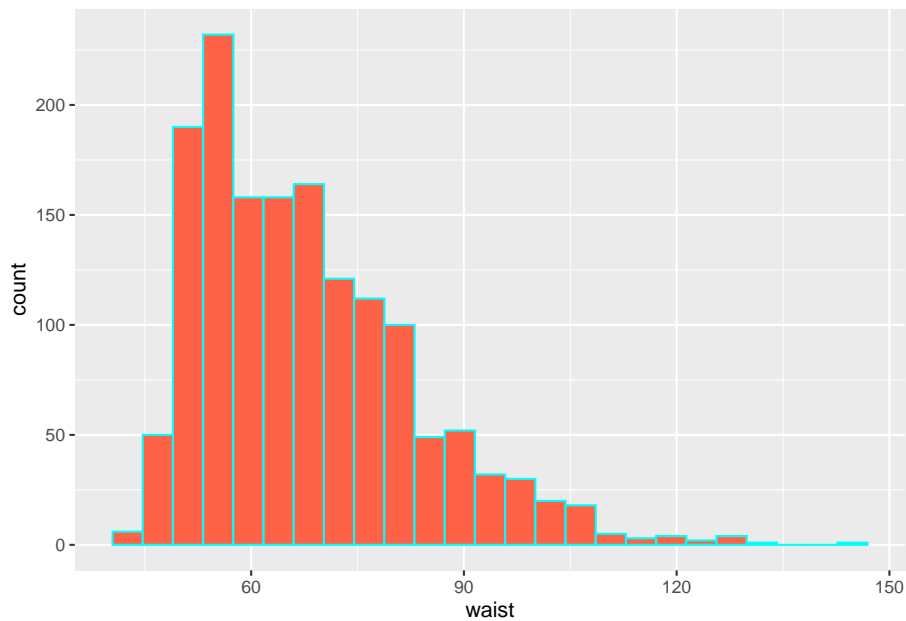
7.2.9 `triceps_skinfold`

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 14.4 6.76 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).
```
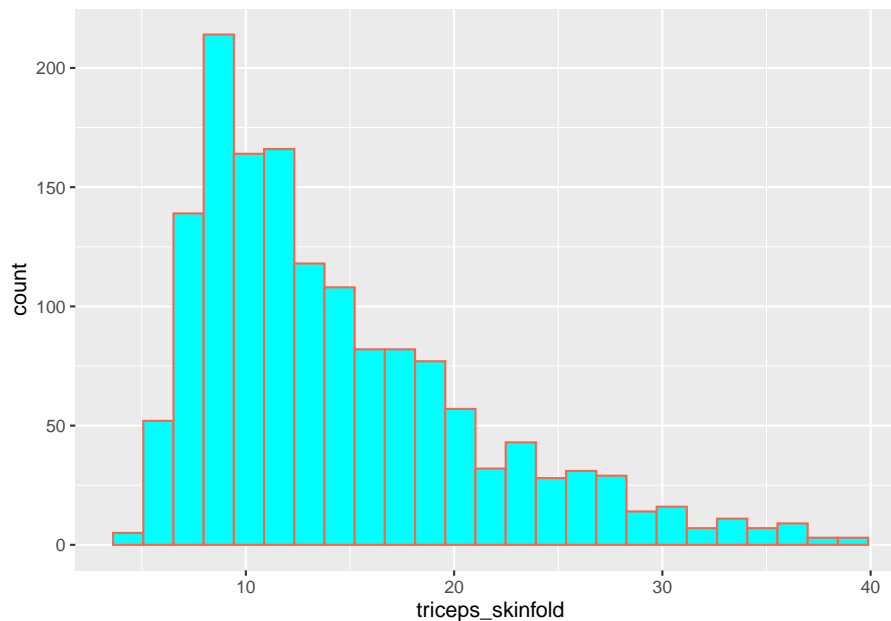
OK. We've seen a few variables, and we'll move on now to look more seriously at the data.

7.3 Basic Numerical 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.   : 42.5   Min.   : 257   Min.   :  1
 1st Qu.: 55.6   1st Qu.:1368   1st Qu.: 83
 Median : 64.8   Median :1794   Median :117
 Mean   : 67.7   Mean   :1877   Mean   :124
 3rd Qu.: 76.6   3rd Qu.:2306   3rd Qu.:157
 Max.   :144.7   Max.   :5265   Max.   :405
 NA's   :6
```

As an alternative, we can use the `$` notation to indicate the variable we wish to study inside a data set, and we can use the `fivenum` function to get the five numbers used in developing a box plot. We'll focus for a little while on the number of kilocalories consumed by each child, according to the dietary recall questionnaire. That's the `energy` variable.

```
fivenum(nnyfs$energy)
```

```
[1]  257 1367 1794 2306 5265
```

- 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
```

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%
 567 4052
```

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 1368   1794 2306 5265 1877 722 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")
```
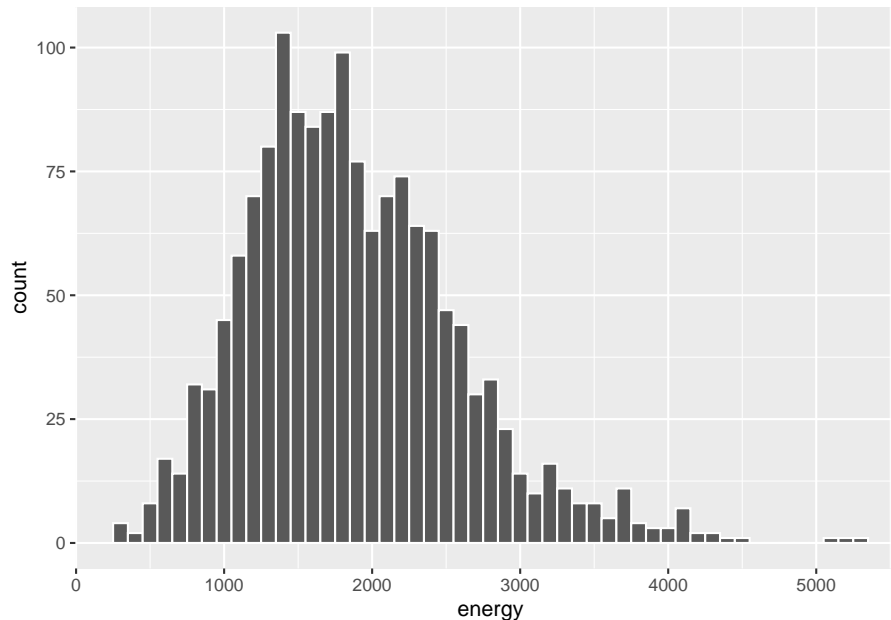

7.5.1   Freedman-Diaconis Rule to select bin width

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 163.32, or, realistically, 163.
- And so the number of bins would be 30.664, or, realistically 31.

Here, we'll draw the graph again, using the Freedman-Diaconis rule to identify the number of bins, and also play around a bit with the fill and color of the bars.

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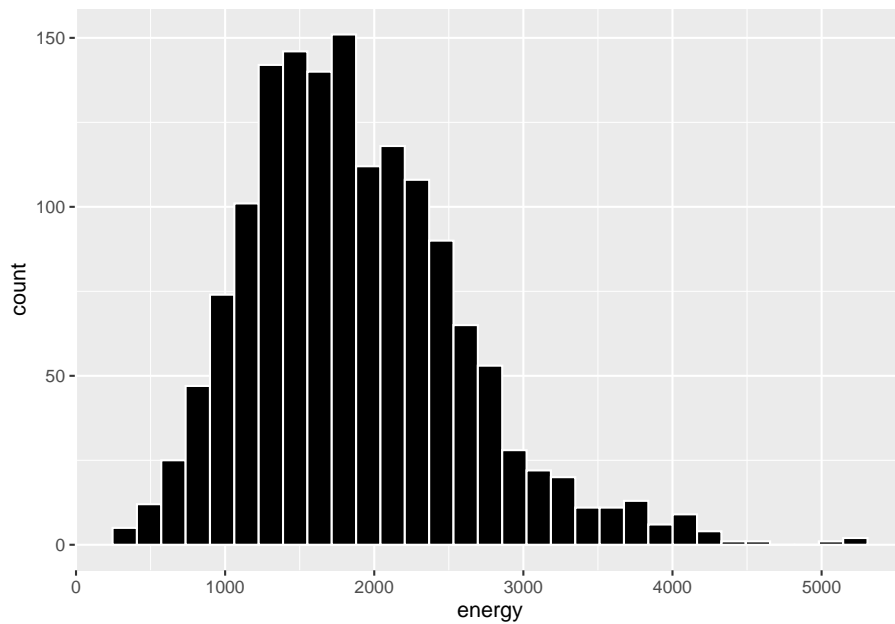

This is a nice start, but it is by no means a finished graph.

Let's improve the axis labels, add a title, and fill in the bars with a distinctive blue and use a black outline around each bar. I'll just use 25 bars, because I like how that looks in this case, and optimizing the number of bins is rarely important.

```
ggplot(data = nnyfs, aes(x = energy)) +
    geom_histogram(bins=25, color = "black", fill = "dodgerblue") +
    labs(title = "Histogram of Body-Mass Index Results in the nnyfs data",
        x = "Energy Consumed (kcal)", y = "# of Subjects")
```

Histogram of Body–Mass Index Results in the nnyfs data
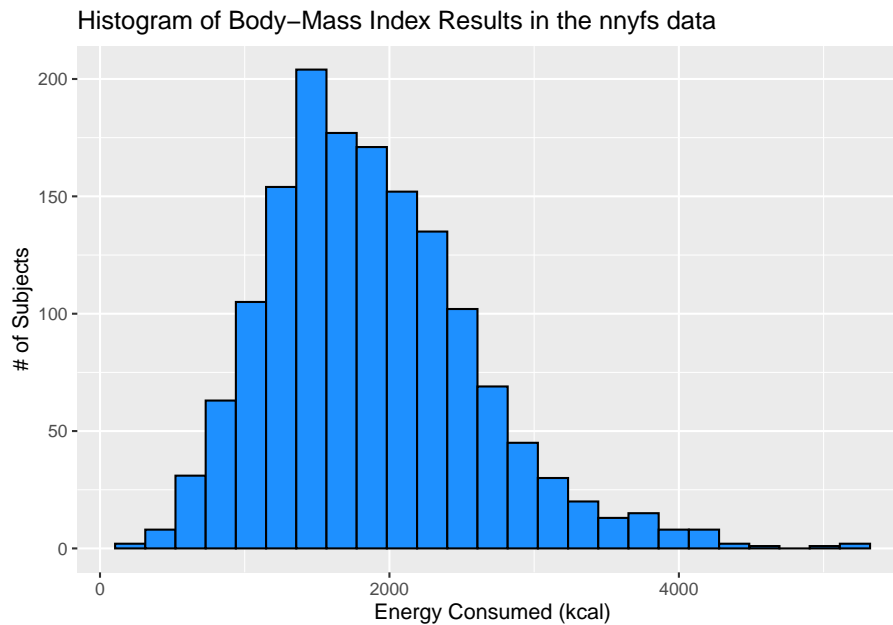
7.5.2 A Note on Colors

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)
```
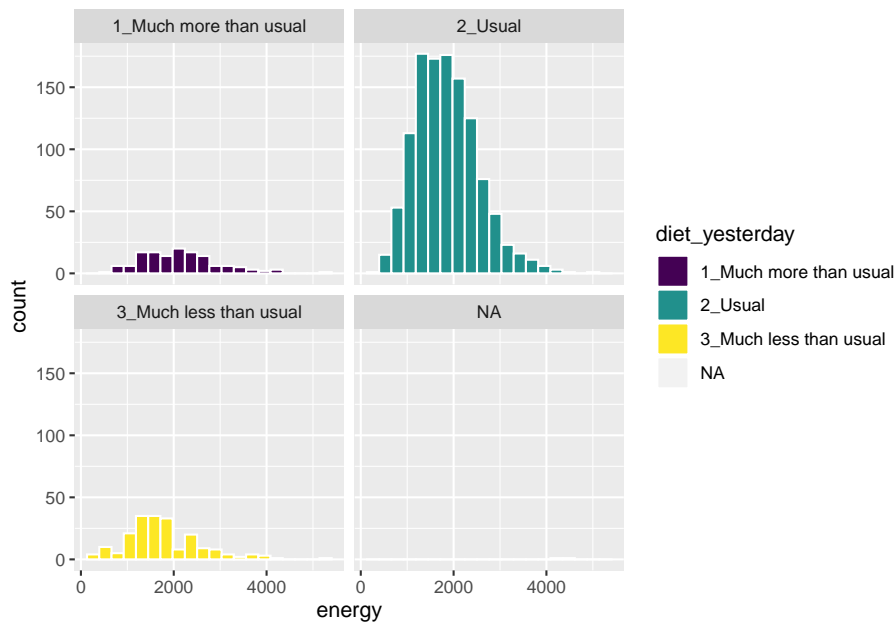
We don't really need the legend here, and perhaps we should restrict the plot to participants who responded to the `diet_yesterday` question, and put in a title and better axis labels?

```
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")
```

Energy Consumption and How Typical Was Yesterday's Eating
NHANES National Youth Fitness Survey, no survey weighting
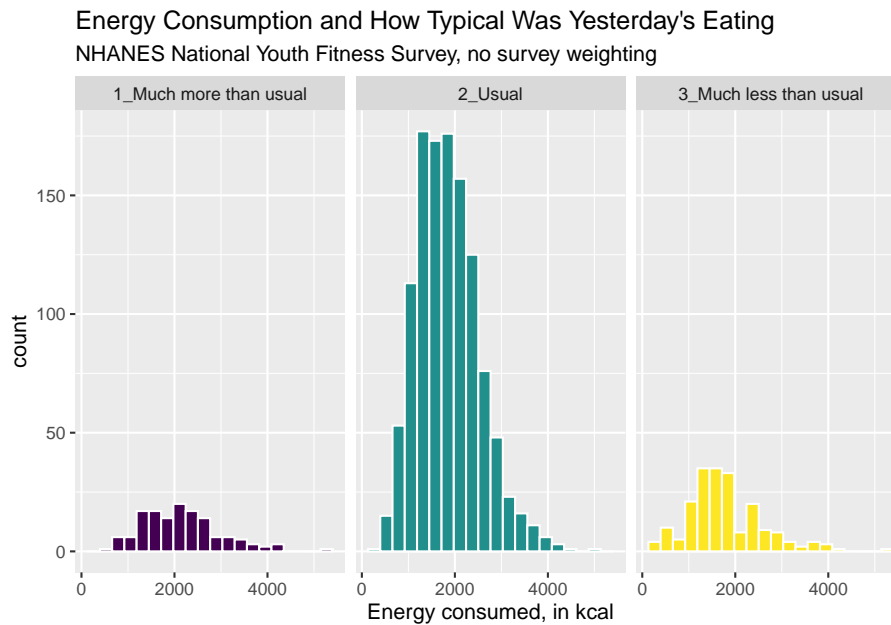
7.6   The Stem-and-Leaf

We might consider a **stem-and-leaf display** (a John Tukey invention) to show
the actual data values while retaining the shape of a histogram. The `scale`
parameter can help expand the size of the diagram, so you can see more of the
values. Stem and leaf displays are usually used for relatively small samples,
perhaps with 10-200 observations, so we'll first take a sample of 150 of the BMI
values from the complete set gathered in the `nnyfs` tibble.

```
set.seed(431) # set a seed for the random sampling so we can replicate the results

sampleA <- sample_n(nnyfs, 150, replace = FALSE) # draw a sample of 150 unique rows from nnyfs

stem(sampleA$bmi) # build a stem-and-leaf for those 150 sampled BMI values
```

```
  The decimal point is at the |

  12 | 936678999
  14 | 0124578800112335555566777789999
  16 | 00122333333355566888900012223355888889
  18 | 03455778991112556889
  20 | 02344678912378
  22 | 00013455779125
```

```
24 | 346978
26 | 03799446
28 | 25
30 | 2
32 | 065
34 | 14
36 | 31
38 |
40 |
42 |
44 |
46 |
48 | 3
```

We can see that the minimum BMI value in this small sample is NA and the maximum BMI value is NA.

```
summary(sampleA$bmi)
```

```
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  12.9    15.9    17.8    19.7    22.0    48.3       1
```

If we really wanted to, we could obtain a stem-and-leaf of all of the BMI values in the entire nnyfs data. The scale parameter lets us see some more of the values.

```
stem(nnyfs$bmi, scale = 2)
```

```
  The decimal point is at the |

  11 | 9
  12 | 679
  13 | 0012344455556666677778888899999999999
  14 | 0000000111111111111122222222233333333333334444444444455555555555666666667+48
  15 | 00000000000001111111111111122222222222222222233333333333333333333+134
  16 | 00000000000000000000001111111111111122222222222222222233333333333333+114
  17 | 0000000000000000000001111111111111122222222222222222233333333333333+75
  18 | 0000000000000011111111111111122222222222233333333334444444455555555555+39
  19 | 00000111111111111111222222222222222233333333333333444444444555555555555+36
  20 | 0000000000001111111111222222233333333333334444444455566666666666777778+3
  21 | 00000001111112222223333333333444444444455555555556666666677777778888+7
  22 | 0000000000001111111222222233333333344444444455555666666777777788999999
  23 | 000000011112222222233333334444444555667778888899999
  24 | 0000001122222333344444555555666666677788888899999
  25 | 0001112222223334444455566666666677777888899999
  26 | 000112222233455556677899999
  27 | 00223344445556667999
```

```
28 | 11122344456667778999
29 | 0112222333567788
30 | 1122222344556788999
31 | 0013445567788
32 | 0023344669
33 | 1255
34 | 01224456679
35 | 0179
36 | 369
37 | 01458
38 | 138
39 |
40 | 0
41 | 6
42 | 8
43 | 0
44 |
45 |
46 | 9
47 |
48 | 23
```

Note that some of the rows extend far beyond what is displayed in the data (as indicated by the + sign, followed by a count of the number of unshown data values.)

7.6.1 A Fancier Stem-and-Leaf Display

We can use the `stem.leaf` function in the `aplpack` package to obtain a fancier version of the stem-and-leaf plot, that identifies outlying values. Below, we display this new version for the random sample of 150 BMI observations we developed earlier.

```
aplpack::stem.leaf(sampleA$bmi)
```

```
1 | 2: represents 1.2
 leaf unit: 0.1
            n: 149
    1     12 | 9
    9     13 | 36678999
   17     14 | 01245788
   39     15 | 0011233555566777789999
   60     16 | 001223333333555668889
  (16)    17 | 0001222355888889
   73     18 | 0345577899
   63     19 | 1112556889
```

```
  53      20 |  023446789
  44      21 |  12378
  39      22 |  00013455779
  28      23 |  125
  25      24 |  3469
  21      25 |  78
  19      26 |  03799
  14      27 |  446
          28 |
  11      29 |  25
   9      30 |  2
HI: 32 32.6 33.5 34.1 34.4 36.3 37.1 48.3
NA's: 1
```

We can also produce back-to-back stem and leaf plots to compare, for instance, body-mass index by sex.

```
samp.F <- filter(sampleA, sex=="Female")
samp.M <- filter(sampleA, sex=="Male")


aplpack::stem.leaf.backback(samp.F$bmi, samp.M$bmi)
```

```
    _____
      1 | 2: represents 1.2, leaf unit: 0.1
              samp.F$bmi       samp.M$bmi

    _____
      1                  9| 12 |
      5               9863| 13 |6799               4
      7                 10| 14 |245788            10
     19       999875532100| 15 |1355667779        20
     25             863330| 16 |012233335556889   35
     29               8210| 17 |002235588889     (12)
     (4)              9750| 18 |345789            38
     31             985211| 19 |1568              32
     25               943| 20 |024678            28
     22              8731| 21 |2                 22
     18               000| 22 |13455779          21
     15                51| 23 |2                 13
     13                63| 24 |49                12
     11                87| 25 |
      9                 0| 26 |3799              10
      8               644| 27 |
                        | 28 |
      5                 2| 29 |5                  6
      4                 2| 30 |
                        | 31 |

    _____
```

```
HI: 33.5 37.1 48.3         HI: 32 32.6 34.1 34.4
                              36.3
n:                  65     85
NAs:                 1     0
-------------------------------------------------
```

7.7 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
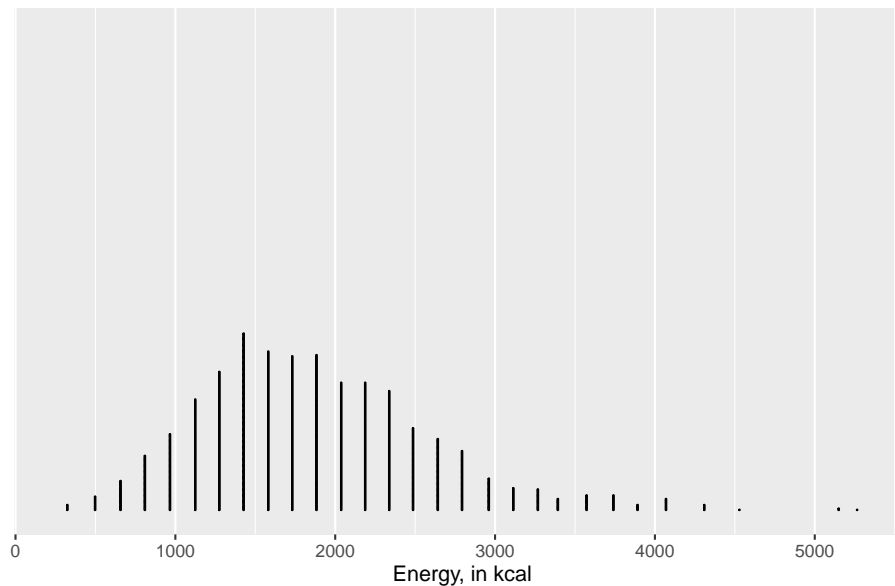
7.8 The Frequency Polygon

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")
```

Frequency Polygon of nnyfs Energy data
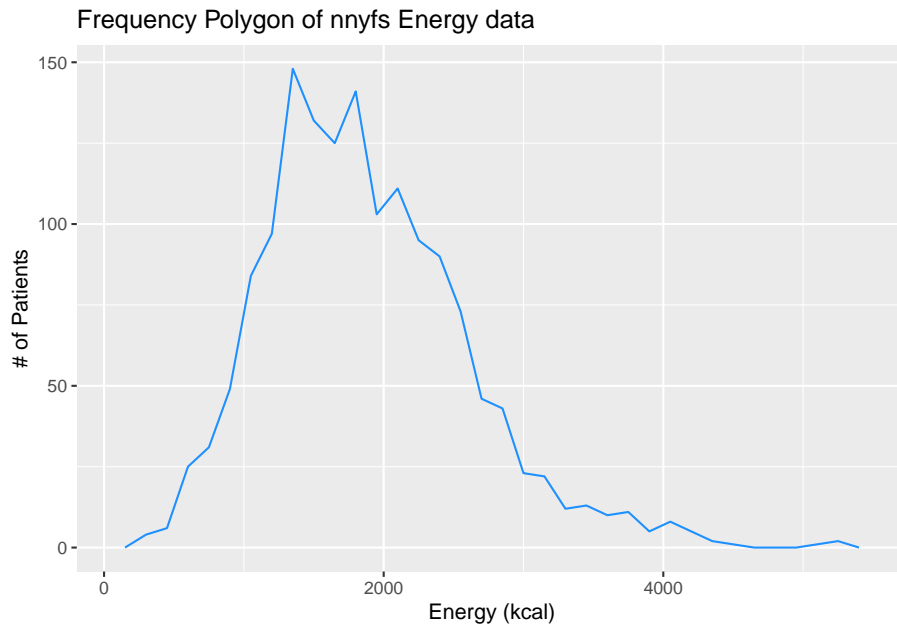
7.9 Plotting the Probability Density Function

We can also produce a density function, which has the effect of smoothing out the bumps in a histogram or frequency polygon, while also changing what is plotted on the y-axis.

```
ggplot(data = nnyfs, aes(x = energy)) +
    geom_density(kernel = "gaussian", color = "dodgerblue") +
    labs(title = "Density of nnyfs Energy data",
         x = "Energy (kcal)", y = "Probability Density function")
```

Density of nnyfs Energy data
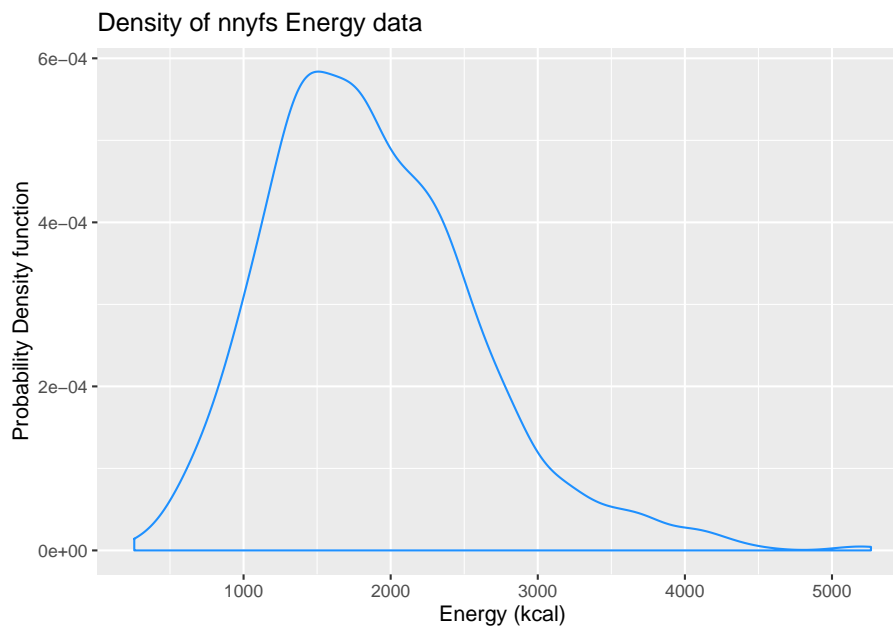
So, what's a density function?

- A probability density function is a function of a continuous variable, x, that represents the probability of x falling within a given range. Specifically, the integral over the interval (a,b) of the density function gives the probability that the value of x is within (a,b).
- If you're interested in exploring more on the notion of density functions for continuous (and discrete) random variables, some nice elementary material is available at Khan Academy.

7.10 The Boxplot

Sometimes, it's helpful to picture the five-number summary of the data in such a way as to get a general sense of the distribution. One approach is a **boxplot**, sometimes called a box-and-whisker plot.

7.10.1 Drawing a Boxplot for One Variable in `ggplot2`

The `ggplot2` library easily handles comparison boxplots for multiple distributions, as we'll see in a moment. However, building a boxplot for a single distribution requires a little trickiness.

```
ggplot(nnyfs, aes(x = 1, y = energy)) +
    geom_boxplot(fill = "deepskyblue") +
    coord_flip() +
    labs(title = "Boxplot of Energy for kids in the NNYFS",
         y = "Energy (kcal)",
         x = "") +
    theme(axis.text.y = element_blank(),
          axis.ticks.y = element_blank())
```

Boxplot of Energy for kids in the NNYFS
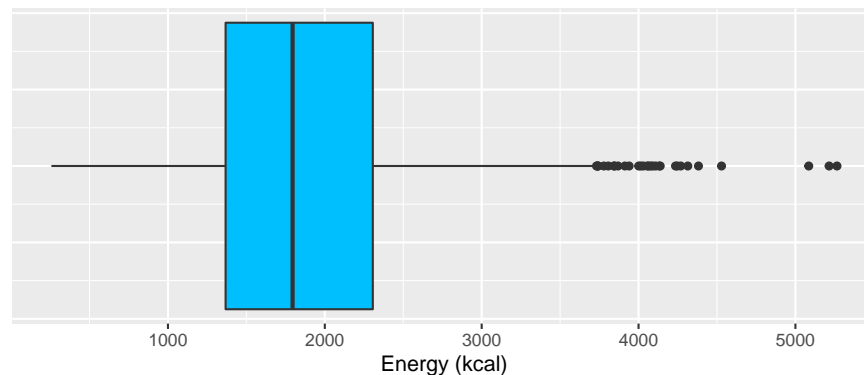
7.10.2   About the Boxplot

The boxplot is another John Tukey invention.

- R draws the box (here in yellow) so that its edges of the box fall at the 25th and 75th percentiles of the data, and the thick line inside the box falls at the median (50th percentile).
- The whiskers then extend out to the largest and smallest values that are not classified by the plot as candidate *outliers.*
- An outlier is an unusual point, far from the center of a distribution.
- Note that I've used the `horizontal` option to show this boxplot in this direction. Most comparison boxplots, as we'll see below, are oriented vertically.

The boxplot's **whiskers** that are drawn from the first and third quartiles (i.e. the 25th and 75th percentiles) out to the most extreme points in the data that do not meet the standard of "candidate outliers." An outlier is simply a point that is far away from the center of the data - which may be due to any number of reasons, and generally indicates a need for further investigation.

Most software, including R, uses a standard proposed by Tukey which describes a "candidate outlier" as any point above the ***upper fence*** or below the ***lower fence***. The definitions of the fences are based on the inter-quartile range (IQR).

If IQR = 75th percentile - 25th percentile, then the upper fence is 75th percentile + 1.5 IQR, and the lower fence is 25th percentile - 1.5 IQR.

So for these `energy` data,

- the upper fence is located at 2306 + 1.5(938.5) = 3713.75
- the lower fence is located at 1367 - 1.5(938.5) = -40.75

In this case, we see no points identified as outliers in the low part of the distribution, but quite a few identified that way on the high side. This tends to identify about 5% of the data as a candidate outlier, *if* the data follow a Normal distribution.

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

The horizontal orientation I've chosen here clarifies the relationship of direction of skew to the plot. A plot like this, with multiple outliers on the right side is indicative of a long right tail in the distribution, and hence, positive or right skew - with the mean being larger than the median. Other indications of skew include having one side of the box being substantially wider than the other, or one side of the whiskers being substantially longer than the other. More on skew later.

7.11 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")
```
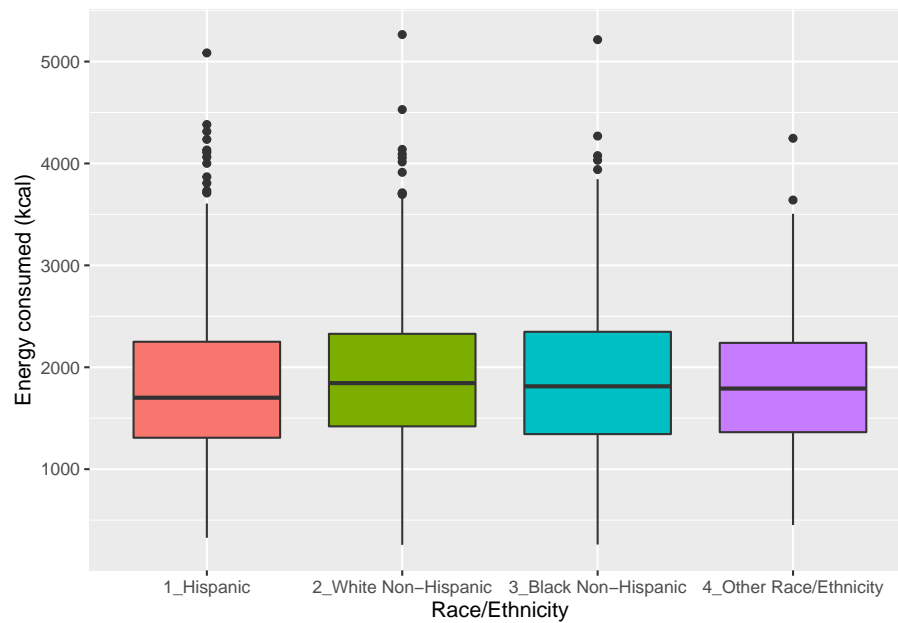
Let's look at the comparison of observed `energy` levels across the five categories
in our `phys_health` variable, now making use of the `viridis` color scheme.

```
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")
```
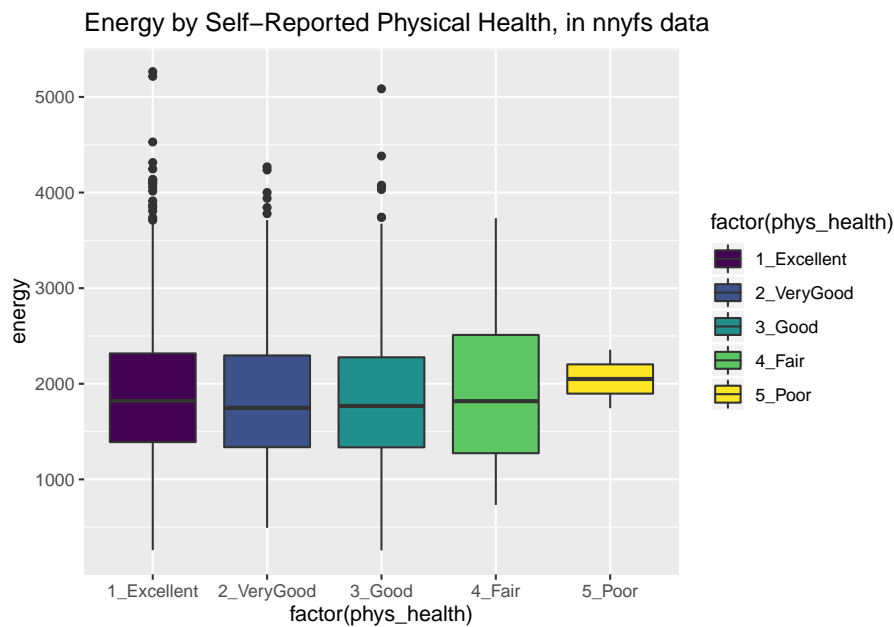
As a graph, that's not bad, but what if we want to improve it further?

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 = "")
```
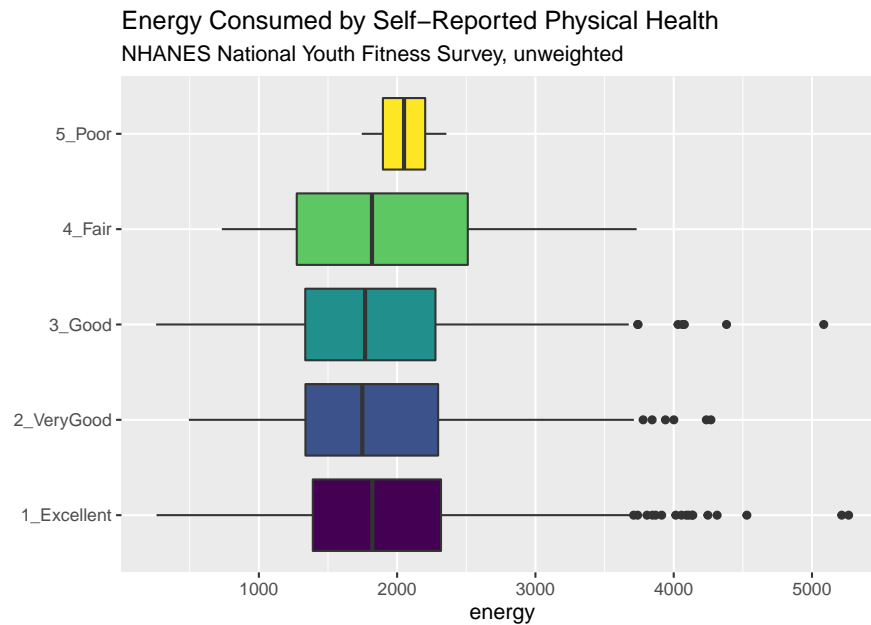
7.12 Using `describe` in the `psych` library

For additional numerical summaries, one option would be to consider using the `describe` function from the `psych` library.

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

```
     vars    n mean  sd median trimmed mad min  max range skew kurtosis   se
X1     1 1518 1877 722   1794    1827 678 257 5265  5008  0.8     1.13 18.5
```

This package provides, in order, the following...

- `n` = the sample size
- `mean` = the sample mean
- `sd` = the sample standard deviation
- `median` = the median, or 50th percentile
- `trimmed` = mean of the middle 80% of the data
- `mad` = median absolute deviation
- `min` = minimum value in the sample
- `max` = maximum value in the sample
- `range` = max - min
- `skew` = skewness measure, described below (indicates degree of asymmetry)
- `kurtosis` = kurtosis measure, described below (indicates heaviness of tails, degree of outlier-proneness)

- `se` = standard error of the sample mean = sd / square root of sample size, useful in inference

7.12.1 The Trimmed Mean

The **trimmed mean** trim value in R indicates proportion of observations to be trimmed from each end of the outcome distribution before the mean is calculated. The `trimmed` value provided by the `psych::describe` package describes what this particular package calls a 20% trimmed mean (bottom and top 10% of `energy` values are removed before taking the mean - it's the mean of the middle 80% of the data.) I might call that a 10% trimmed mean in some settings, but that's just me.

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

```
[1] 1827
```

7.12.2 The Median Absolute Deviation

An alternative to the IQR that is fancier, and a bit more robust, is the **median absolute deviation**, which, in large sample sizes, for data that follow a Normal distribution, will be (in expectation) equal to the standard deviation. The MAD is the median of the absolute deviations from the median, multiplied by a constant (1.4826) to yield asymptotically normal consistency.

```
mad(nnyfs$energy)
```

```
[1] 678
```

7.13 Assessing Skew

A relatively common idea is to assess **skewness**, several measures of which (including the one below, sometimes called type 3 skewness, or Pearson's moment coefficient of skewness) 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 - left skew (mean < median) is indicated by negative skewness values, while right skew (mean > median) is indicated by positive values. The skew value will be near zero for data that follow a Normal distribution.

7.13.1 Non-parametric Skew via skew$_1$

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] 0.114
```

The Wikipedia page on skewness, from which some of this material is derived, provides definitions for several other skewness measures.

7.14 Assessing Kurtosis (Heavy-Tailedness)

Another measure of a distribution's shape that can be found in the `psych` library is the **kurtosis**. Kurtosis is an indicator of whether the distribution is heavy-tailed or light-tailed as compared to a Normal distribution. Positive kurtosis means more of the variance is due to outliers - unusual points far away from the mean relative to what we might expect from a Normally distributed data set with the same standard deviation.

- A Normal distribution will have a kurtosis value near 0, a distribution with similar tail behavior to what we would expect from a Normal is said to be *mesokurtic*
- Higher kurtosis values (meaningfully higher than 0) indicate that, as compared to a Normal distribution, the observed variance is more the result of extreme outliers (i.e. heavy tails) as opposed to being the result of more modest sized deviations from the mean. These heavy-tailed, or outlier prone, distributions are sometimes called *leptokurtic.*
- Kurtosis values meaningfully lower than 0 indicate light-tailed data, with fewer outliers than we'd expect in a Normal distribution. Such distributions are sometimes referred to as *platykurtic*, and include distributions without outliers, like the Uniform distribution.

Here's a table:

Fewer outliers than a Normal	Approximately Normal	More outliers than a Normal
Light-tailed *platykurtic* (kurtosis < 0)	"Normalish" *mesokurtic* (kurtosis $= 0$)	Heavy-tailed *leptokurtic* (kurtosis > 0)

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

```
[1] 1.13
```

7.14.1 The Standard Error of the Sample Mean

The **standard error** of the sample mean, which is the standard deviation divided by the square root of the sample size:

```
sd(nnyfs$energy)/sqrt(length(nnyfs$energy))
```

```
[1] 18.5
```

7.15 The `describe` function in the `Hmisc` library

The `Hmisc` library 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
- 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

7.16 `xda` from GitHub for numerical summaries for exploratory data analysis

```
## next two commands needed if xda is not already installed

library(devtools)
install_github("ujjwalkarn/xda")
```

```
Skipping install of 'xda' from a github remote, the SHA1 (86cf14db) has not changed sir
  Use `force = TRUE` to force installation
```

```
xda::numSummary(nnyfs)
```

```
                       n      mean       sd      max      min    range
SEQN                1518  7.27e+04 4.57e+02  73492.0 71917.00   1575.0
age_child           1518  9.03e+00 3.71e+00     15.0     3.00     12.0
educ_child          1181  4.26e+00 2.82e+00      9.0     0.00      9.0
```

```
sampling_wt            1518 3.41e+04 1.60e+04 104673.9  9412.87 95261.0
income_pov             1429 2.24e+00 1.61e+00      5.0     0.00     5.0
age_adult              1518 4.00e+01 9.60e+00     80.0    18.00    62.0
energy                 1518 1.88e+03 7.22e+02   5265.0   257.00  5008.0
protein                1518 6.69e+01 3.10e+01    241.8     4.18   237.7
sugar                  1518 1.24e+02 5.90e+01    405.5     1.00   404.5
fat                    1518 6.74e+01 3.37e+01    235.2     1.70   233.5
water                  1518 5.43e+02 6.50e+02   8591.2     0.00  8591.2
plank_time             1384 6.11e+01 4.58e+01    450.0     1.00   449.0
height                 1514 1.37e+02 2.30e+01    188.9    89.70    99.2
weight                 1514 3.98e+01 2.08e+01    136.9    12.30   124.6
bmi                    1514 1.96e+01 5.08e+00     48.3    11.90    36.4
arm_length             1511 2.90e+01 5.63e+00     42.5    17.30    25.2
waist                  1512 6.77e+01 1.52e+01    144.7    42.50   102.2
arm_circ               1513 2.29e+01 5.50e+00     46.8    13.70    33.1
calf_circ              1509 2.98e+01 6.19e+00     55.9    18.30    37.6
calf_skinfold          1390 1.37e+01 6.81e+00     39.9     3.70    36.2
triceps_skinfold       1497 1.44e+01 6.76e+00     38.8     4.00    34.8
subscapular_skinfold   1451 1.07e+01 6.31e+00     38.0     3.70    34.3
active_days            1513 5.39e+00 2.12e+00      7.0     0.00     7.0
tv_hours               1514 1.90e+00 1.37e+00      5.0     0.00     5.0
computer_hours         1515 1.04e+00 1.35e+00      5.0     0.00     5.0
meals_out              1510 1.79e+00 2.13e+00     20.0     0.00    20.0
med_count              1518 2.96e-01 7.56e-01      6.0     0.00     6.0
                     nunique nzeros      iqr lowerbound upperbound
SEQN                    1518      0   792.00   71121.25   74288.75
age_child                 13      0     6.00      -3.00      21.00
educ_child                11    146     5.00      -5.50      14.50
sampling_wt              845      0 19987.74   -6664.16   73223.34
income_pov               238     15     2.65      -3.10       7.49
age_adult                 55      0    12.00      15.00      63.00
energy                  1137      0   939.00     -41.00    3714.50
protein                 1482      0    37.25     -10.55     138.44
sugar                   1491      0    74.35     -28.86     268.57
fat                     1474      0    42.48     -20.57     149.34
water                    342    326   636.94    -851.73    1696.05
plank_time               190      0    55.00     -55.50     164.75
height                   694      0    38.45      60.67     214.45
weight                   615      0    30.55     -23.30      98.90
bmi                      226      0     6.00       6.90      30.90
arm_length               222      0     9.60       9.80      48.20
waist                    511      0    21.00      24.10     108.10
arm_circ                 240      0     7.80       6.70      37.90
calf_circ                259      0     9.57      10.34      48.66
calf_skinfold            253      0     8.40      -4.00      29.60
triceps_skinfold         263      0     8.90      -4.25      31.35
```

subscapular_skinfold	230	0	7.00	-4.30	23.70
active_days	9	68	3.00	-0.50	11.50
tv_hours	7	254	2.00	-2.00	6.00
computer_hours	7	759	2.00	-3.00	5.00
meals_out	17	425	2.00	-3.00	5.00
med_count	7	1241	0.00	0.00	0.00

	noutlier	kurtosis	skewness	mode	miss	miss%
SEQN	0	-1.2050	0.000814	71917.0	0	0.000
age_child	0	-1.2317	-0.025225	12.0	0	0.000
educ_child	0	-1.2195	0.010719	NA	337	22.200
sampling_wt	37	1.9218	1.123744	53977.6	0	0.000
income_pov	0	-1.1165	0.501565	5.0	89	5.863
age_adult	20	0.8409	0.631691	40.0	0	0.000
energy	30	1.1305	0.797210	1915.0	0	0.000
protein	45	2.0114	1.121048	82.5	0	0.000
sugar	30	1.4265	0.921237	94.7	0	0.000
fat	43	1.6609	1.024582	76.7	0	0.000
water	79	24.9830	3.486903	0.0	0	0.000
plank_time	40	5.3557	1.538689	NA	134	8.827
height	0	-1.0648	-0.057423	154.0	4	0.264
weight	18	1.0902	1.031371	22.0	4	0.264
bmi	63	3.3768	1.600300	15.5	4	0.264
arm_length	0	-1.0848	-0.023759	34.0	7	0.461
waist	25	1.1911	1.041341	55.4	6	0.395
arm_circ	19	0.5714	0.862535	17.0	5	0.329
calf_circ	6	-0.1366	0.535805	33.0	9	0.593
calf_skinfold	49	0.7705	1.114848	NA	128	8.432
triceps_skinfold	39	0.8936	1.135255	8.0	21	1.383
subscapular_skinfold	77	2.2220	1.573722	NA	67	4.414
active_days	0	0.0351	-1.095239	7.0	5	0.329
tv_hours	0	-0.2414	0.538424	2.0	4	0.264
computer_hours	0	0.8827	1.293648	0.0	3	0.198
meals_out	70	12.4583	2.823921	0.0	8	0.527
med_count	277	14.3174	3.409516	0.0	0	0.000

	1%	5%	25%	50%	75%	95%
SEQN	7.19e+04	71993.85	72309.25	72704.50	73100.75	73415.2
age_child	3.00e+00	3.00	6.00	9.00	12.00	15.0
educ_child	0.00e+00	0.00	2.00	4.00	7.00	9.0
sampling_wt	1.07e+04	13352.52	23317.44	31605.91	43241.74	62215.7
income_pov	2.80e-03	0.31	0.87	1.74	3.52	5.0
age_adult	2.12e+01	26.00	33.00	40.00	45.00	56.0
energy	5.67e+02	849.00	1367.50	1794.50	2306.00	3195.4
protein	1.67e+01	26.53	45.33	61.25	82.57	125.7
sugar	2.41e+01	44.55	82.66	116.91	157.05	234.0
fat	1.16e+01	22.74	43.16	61.98	85.62	129.9
water	0.00e+00	0.00	103.69	375.00	740.64	1718.2

```
plank_time              2.00e+00     5.00    27.00    54.00    82.25   144.0
height                  9.45e+01   100.73   118.35   137.60   156.78   172.2
weight                  1.36e+01    15.70    22.52    34.95    53.08    77.7
bmi                     1.35e+01    14.30    15.90    18.10    21.90    30.2
arm_length              1.88e+01    20.00    24.20    29.00    33.80    37.5
waist                   4.61e+01    49.40    55.60    64.80    76.60    96.8
arm_circ                1.50e+01    16.10    18.40    22.00    26.20    33.3
calf_circ               1.98e+01    21.40    24.70    29.20    34.30    40.2
calf_skinfold           4.89e+00     5.95     8.60    12.00    17.00    27.7
triceps_skinfold        5.60e+00     6.80     9.10    12.40    18.00    28.0
subscapular_skinfold    4.20e+00     4.80     6.20     8.20    13.20    24.1
active_days             0.00e+00     1.00     4.00     7.00     7.00     7.0
tv_hours                0.00e+00     0.00     1.00     2.00     3.00     5.0
computer_hours          0.00e+00     0.00     0.00     0.00     2.00     4.0
meals_out               0.00e+00     0.00     0.00     1.00     2.00     5.0
med_count               0.00e+00     0.00     0.00     0.00     0.00     2.0
                            99%
SEQN                    73476.8
age_child                  15.0
educ_child                  9.0
sampling_wt             86783.8
income_pov                  5.0
age_adult                  66.0
energy                   4051.7
protein                   165.8
sugar                     309.1
fat                       172.8
water                    2841.3
plank_time                200.9
height                    180.5
weight                    103.2
bmi                        36.9
arm_length                 39.5
waist                     112.2
arm_circ                   38.3
calf_circ                  45.3
calf_skinfold              33.8
triceps_skinfold           35.5
subscapular_skinfold       31.4
active_days                 7.0
tv_hours                    5.0
computer_hours              5.0
meals_out                  10.0
med_count                   4.0
```

Most of the elements of this `numSummary` should be familiar. Some new pieces include:

- `nunique` = number of unique values
- `nzeroes` = number of zeroes
- `noutlier` = number of outliers (using a standard that isn't entirely transparent to me)
- `miss` = number of rows with missing value
- `miss%` = percentage of total rows with missing values ((miss/n)*100)
- `5%` = 5th percentile value of that variable (value below which 5 percent of the observations may be found)

```
xda::charSummary(nnyfs)
```

	n	miss	miss%	unique
sex	1518	0	0.000	2
race_eth	1518	0	0.000	4
language	1518	0	0.000	2
educ_adult	1496	22	1.449	6
respondent	1506	12	0.791	4
salt_used	1505	13	0.856	3
diet_yesterday	1516	2	0.132	4
bmi_cat	1514	4	0.264	5
physical_last_week	1514	4	0.264	3
enjoy_recess	1240	278	18.314	6
insured	1518	0	0.000	2
phys_health	1518	0	0.000	5
access_to_care	1518	0	0.000	2
care_source	1518	0	0.000	6
asthma_ever	1518	0	0.000	2
asthma_now	1518	0	0.000	2
med_use	1518	0	0.000	2
insurance	1518	0	0.000	10

```
sex
race_eth                                2_White Non-Hispanic:610, 1_Hispanic:450, 3_
language
educ_adult           4_Some College:464, 5_College Graduate:386, 3_High School Graduate:
respondent
salt_used
diet_yesterday                                              2_Usual:1175, 3_Mu
bmi_cat                                                         2_Normal:92
physical_last_week
enjoy_recess                1_Strongly Agree:903, 2_Agree:257, 3_Neither Agree nor
insured
phys_health                                                   1_Excellent:74
access_to_care                                                          Ha
```

```
care_source            Doctor's Office:1164, Clinic or Health Center:297, No Usual Care Source:27
asthma_ever                                                                               Never
asthma_now
med_use
insurance                                                          Private:736, Medicaid:470, Stat
```

The `top5levels:count` provides the top 5 unique values for each variable, sorted by their counts.

7.17 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 ± Standard Deviation contains approximately 68% of the measurements;
- Mean ± 2(Standard Deviation) contains approximately 95% of the measurements;
- Mean ± 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 1368   1794 2306 5265 1877 722 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.

```r
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 = 35, y = 0.02,
              color="blue", fontface = "italic") +
    labs(title = "nnyfs energy values with Normal Density Superimposed",
         x = "Energy (kcal)", y = "Probability Density Function")
```

nnyfs energy values with Normal Density Superimposed
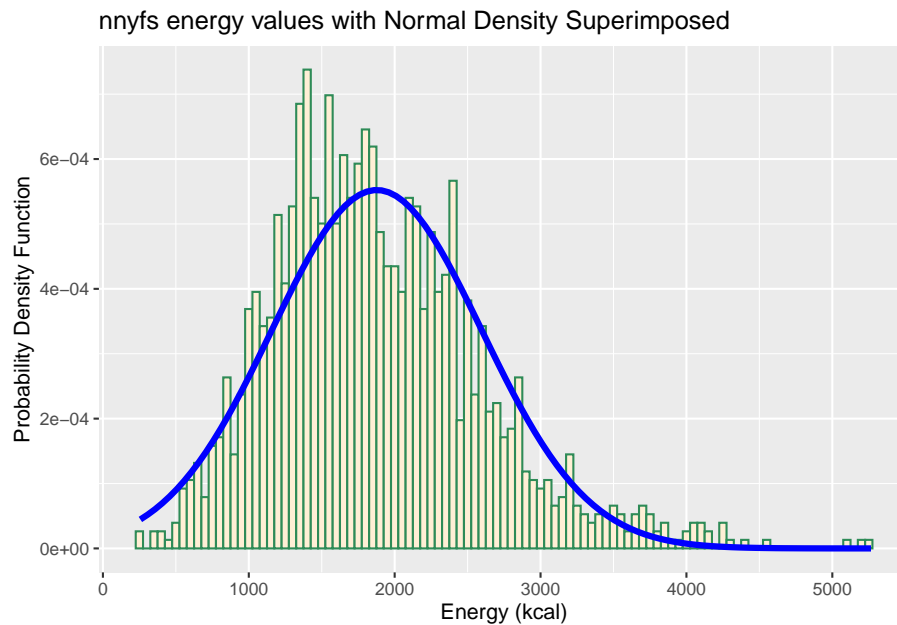
Does it seem as though the Normal model (as shown in the blue density curve) is an effective approximation to the observed distribution shown in the bars of the histogram?

We'll return shortly to the questions:

- Does a Normal distribution model fit our data well? *and*
- If the data aren't Normal, but we want to use a Normal model anyway, what should we do?

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.

```r
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")
```


Histogram of energy, with Normal Model
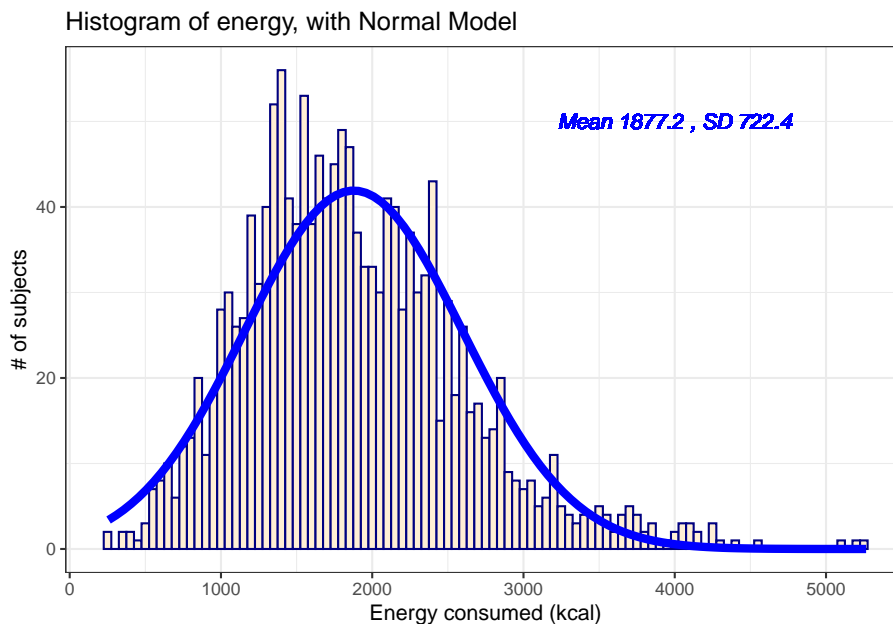
8.4 Does a Normal model work well for the `waist` circumference?

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

nnyfs %>% 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")
```
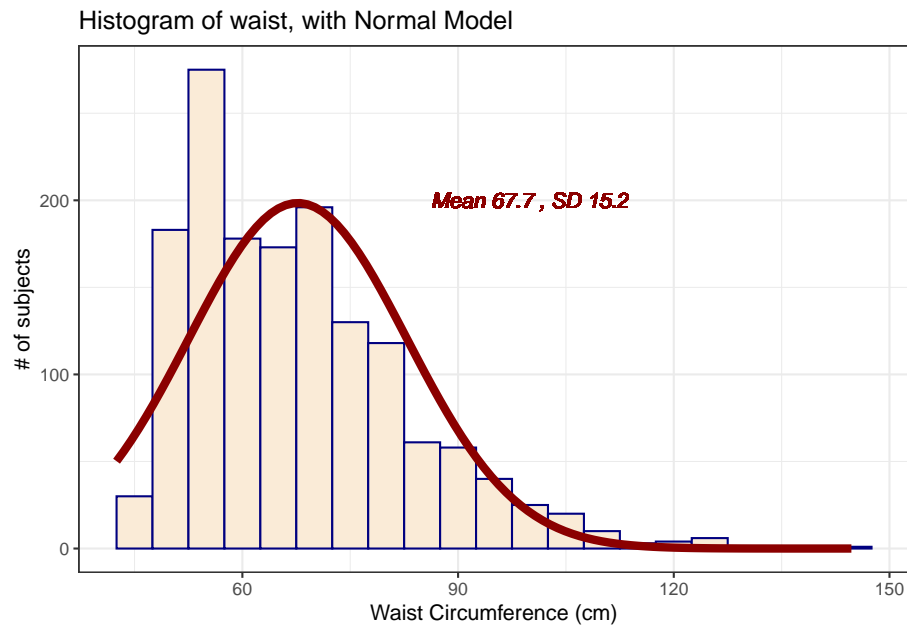
Histogram of waist, with Normal Model


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

```
  min   Q1 median   Q3 max mean   sd    n missing
 42.5 55.6   64.8 76.6 145 67.7 15.2 1512       6
```

The mean is NA and the standard deviation is NA so if the `waist` data really were Normally distributed, we'd expect to see: - About 68% of the data in the range (NA, NA). In fact, 1518 of the 1518 Age values are in this range, or 100%. - About 95% of the data in the range (NA, NA). In fact, 1518 of the 1518 Age values are in this range, or 100%. - About 99.7% of the data in the range (NA, NA). In fact, 1518 of the 1518 Age values are in this range, or 100%.

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")
```
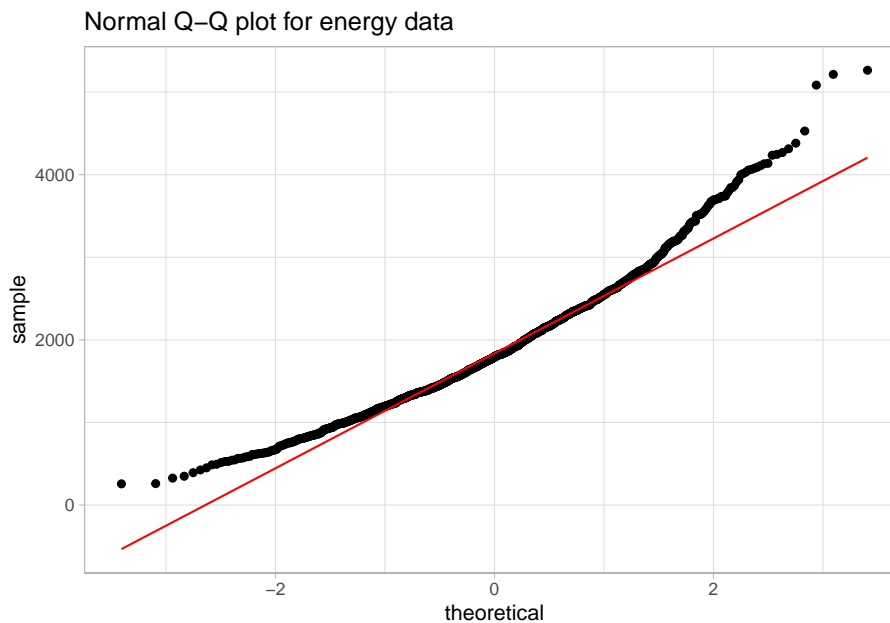
8.6   Interpreting the Normal Q-Q Plot

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.

```r
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 = 100, y = 200,
             color="darkred", fontface = "italic") +
    labs(x = "Simulated Sample Data", y = "")

gridExtra::grid.arrange(p1, p2, ncol=2,
              top ="200 observations from a simulated Normal distribution")
```


200 observations from a simulated Normal distribution
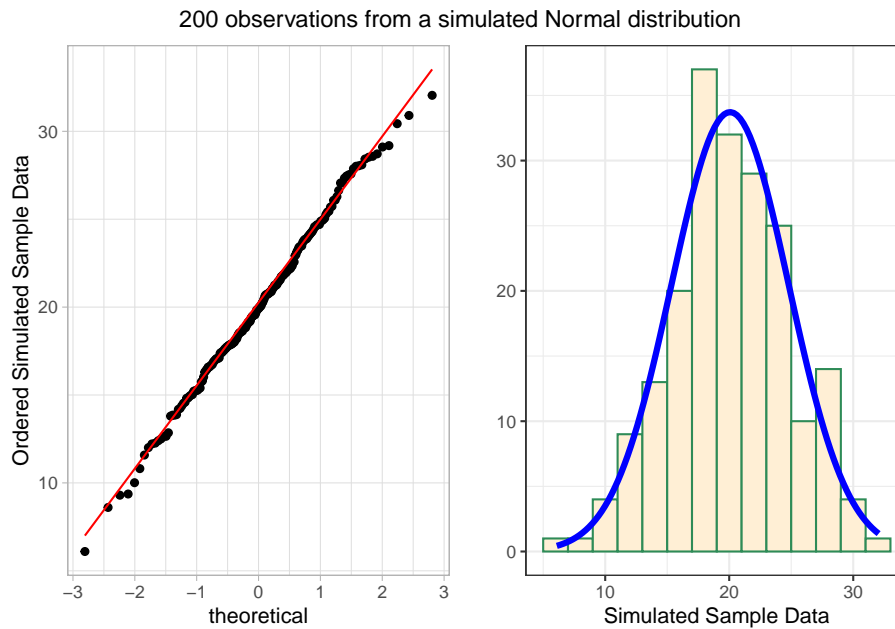
These simulated data appear to be well-modeled by the Normal distribution, because the points on the Normal Q-Q plot follow the diagonal reference line. In particular,

- 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
```

```r
# 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")

gridExtra::grid.arrange(p1, p2, ncol=2, top ="200 observations from simulated Beta dist
```
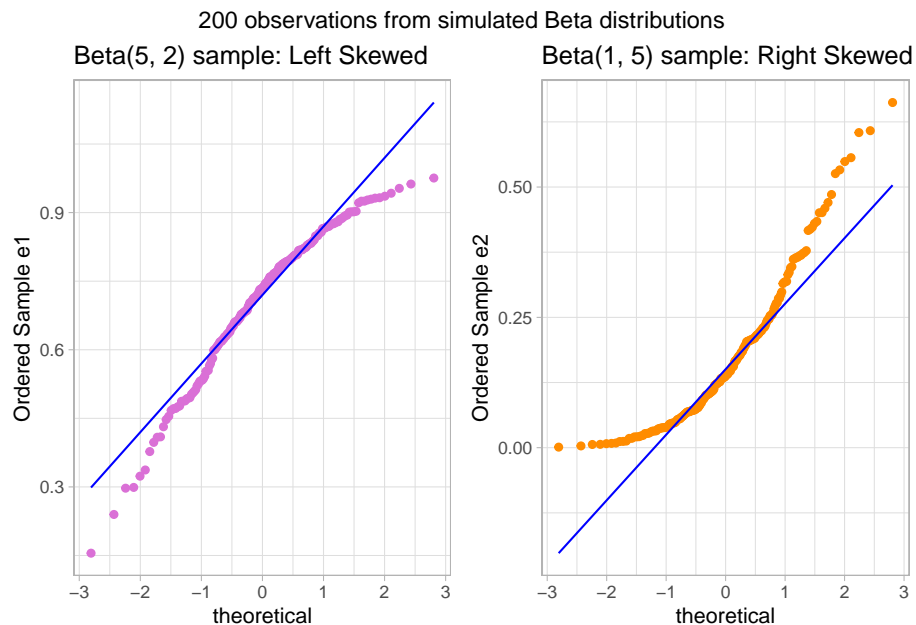

Note the bends away from a straight line in each sample. The non-Normality may be easier to see in a histogram.

```r
res1 <- mosaic::favstats(~ e1, data = temp.2)
bin_w1 <- 0.025 # specify binwidth

p1 <- ggplot(temp.2, aes(x = e1)) +
```

```r
    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")

gridExtra::grid.arrange(p1, p2, ncol=2,
bottom ="Histograms with Normal curve superimposed")
```

Histograms with Normal curve superimposed
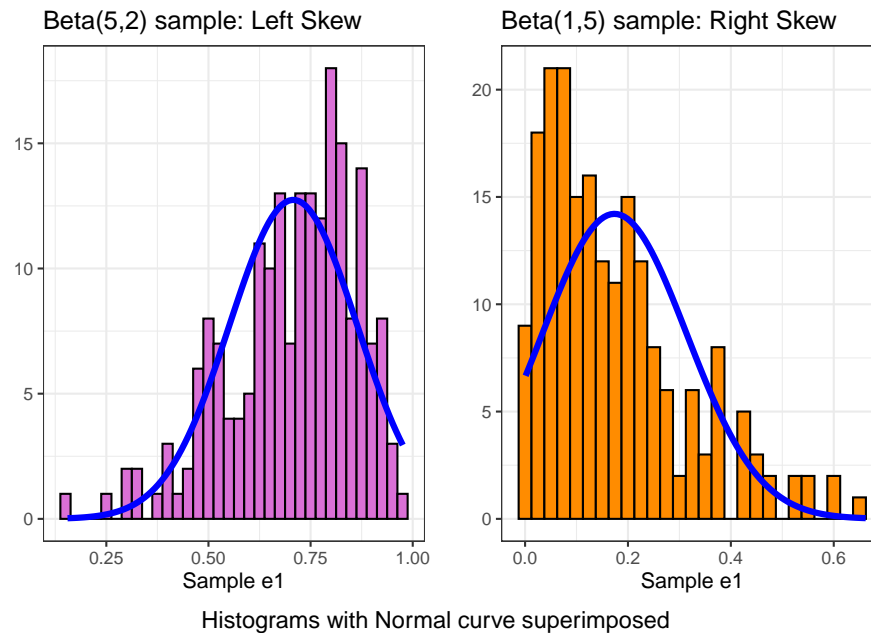
8.6.3   Direction of Skew

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")

gridExtra::grid.arrange(p1, p2, ncol=2, top ="200 observations from simulated distributions")
```
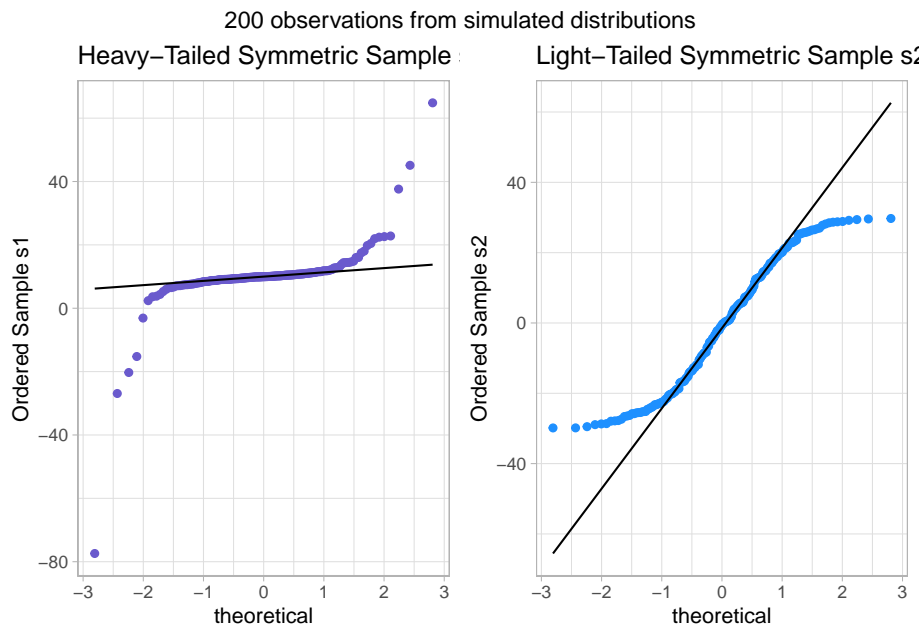

And, we can also visualize these simulations with histograms, although they're less helpful for understanding tail behavior than they are for skew.

```
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() +
```

```r
    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")

gridExtra::grid.arrange(p1, p2, ncol=2,
bottom ="Histograms with Normal curve superimposed")
```

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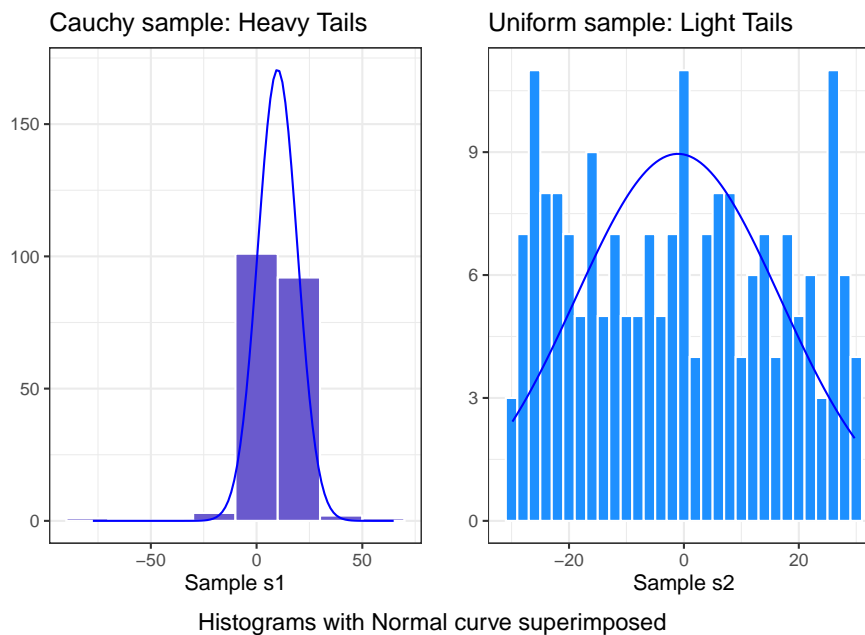
Instead, boxplots (here augmented with violin plots) can be more helpful when thinking about light-tailed vs. heavy-tailed distributions.

```
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")

gridExtra::grid.arrange(p1, p2, nrow=2, top ="200 observations from simulated distributions")
```

200 observations from simulated distributions

Heavy−Tailed Symmetric Sample s1

Light−Tailed Symmetric Sample s2
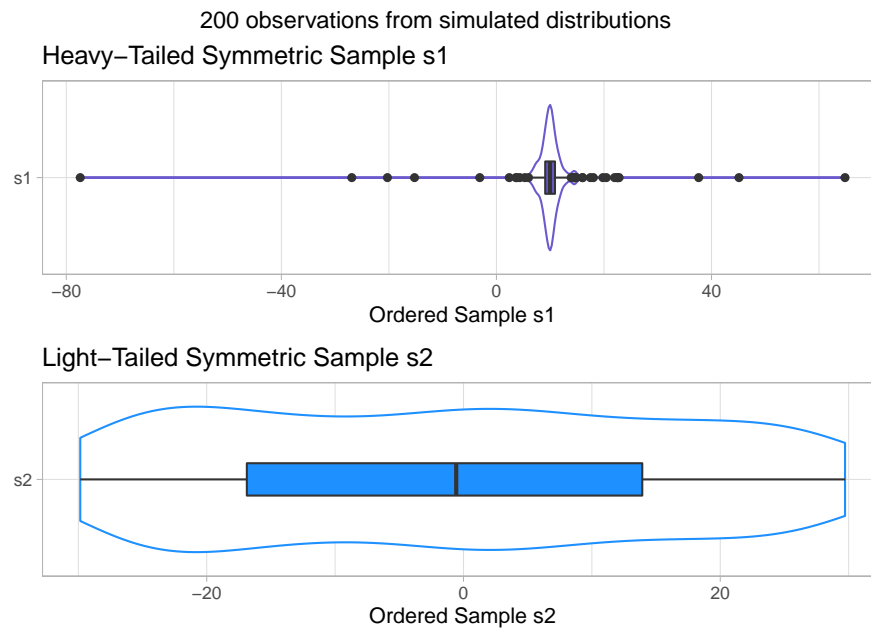

```r
rm(temp.1, temp.2, temp.3, p1, p2, res, res1, res2, bin_w, bin_w1, bin_w2) # cleaning
```

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.

```r
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")
```

```
gridExtra::grid.arrange(p1, p2, ncol = 2)
```
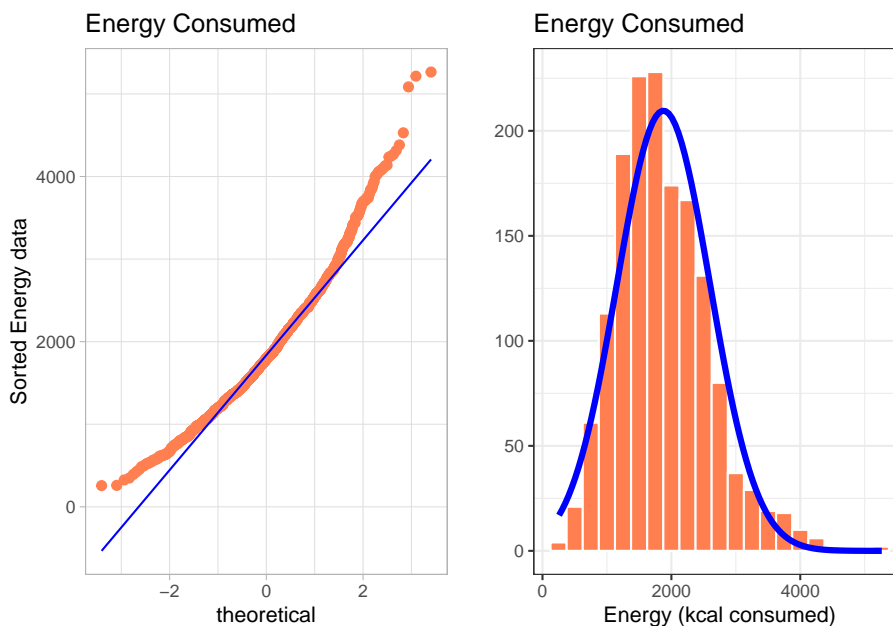

- Skewness is indicated by the curve in the Normal Q-Q plot. Curving up and away from the line in both tails suggests right skew, as does the histogram.

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")

gridExtra::grid.arrange(p1, p2, ncol = 2)
```
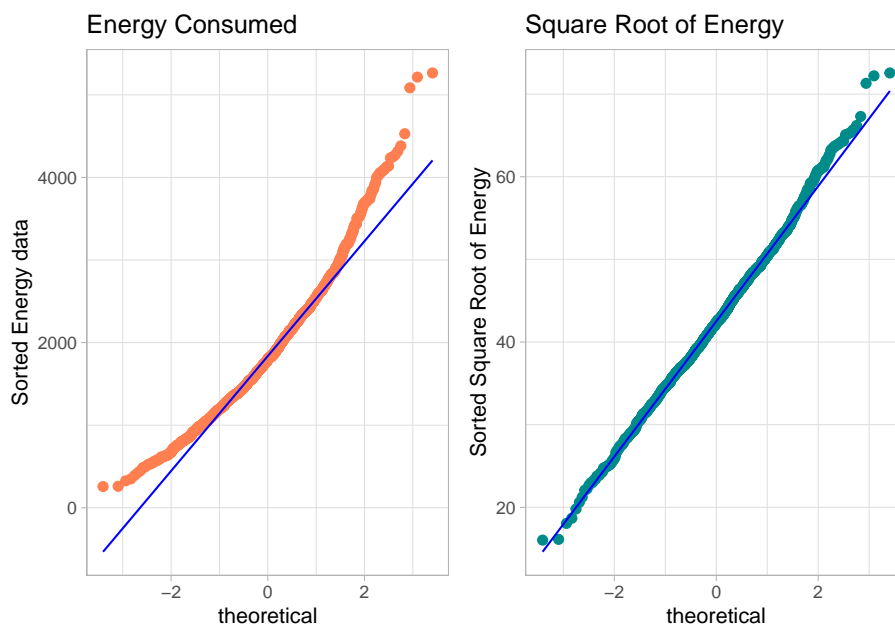

- The left plot shows substantial **right** or *positive* skew
- The right plot shows there's much less skew after the square root has been taken.

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,
```

```r
                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")


gridExtra::grid.arrange(p1, p2, ncol=2,
    top =
    textGrob("Comparing energy to sqrt(energy)",
            gp=gpar(fontsize=15)))
```

Comparing energy to sqrt(energy)
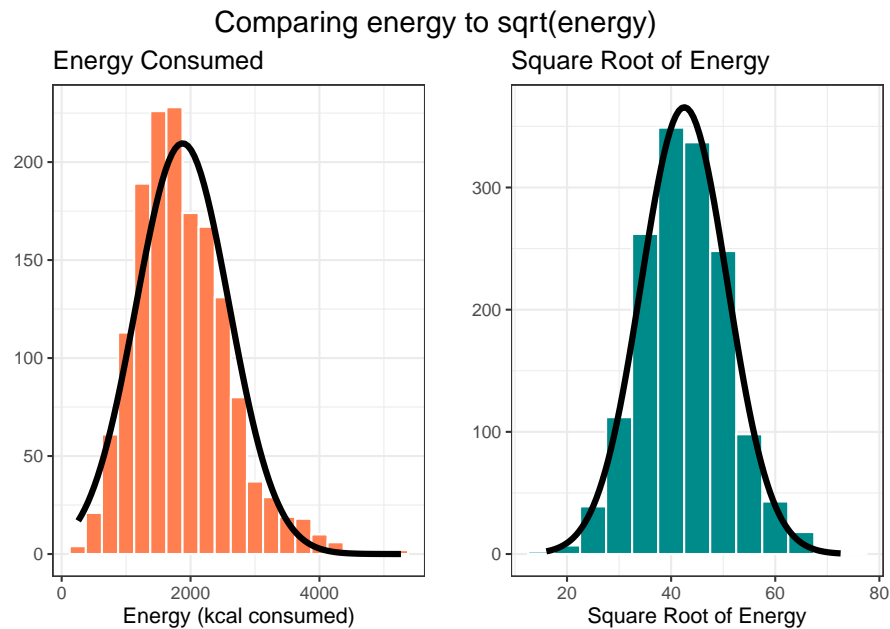
```r
rm(p1, p2, bin_w, bin_w2, res, res2) # cleanup
```

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 initial guidance about choosing between 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	$x^{-2} = 1/x^2$

Power   Transformation

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 45.3   61.3 82.6 242 66.9 31 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)")
```

```r
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)")

gridExtra::grid.arrange(p1, p2, nrow = 1)
```
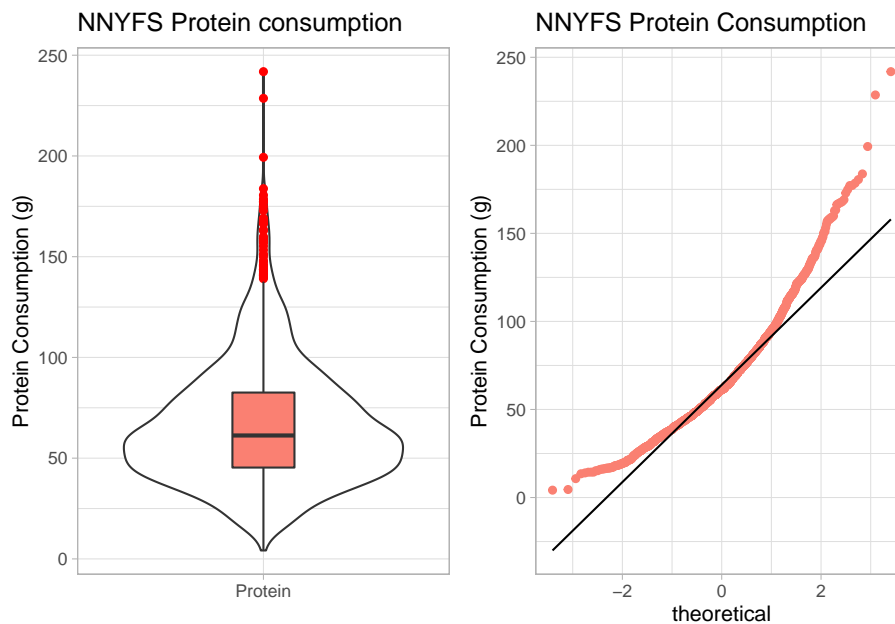

The key point here is that we see several signs of meaningful right skew, and we'll want to consider a transformation that might make a Normal model more plausible.

9.3.1 Using `cowplot::plot_grid()` to compose plots

In addition to the `grid.arrange` function in `gridExtra`, we could also use `plot_grid()` in the `cowplot` package to put our plots together.

```r
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)")

cowplot::plot_grid(p1, p2, nrow = 1)
```
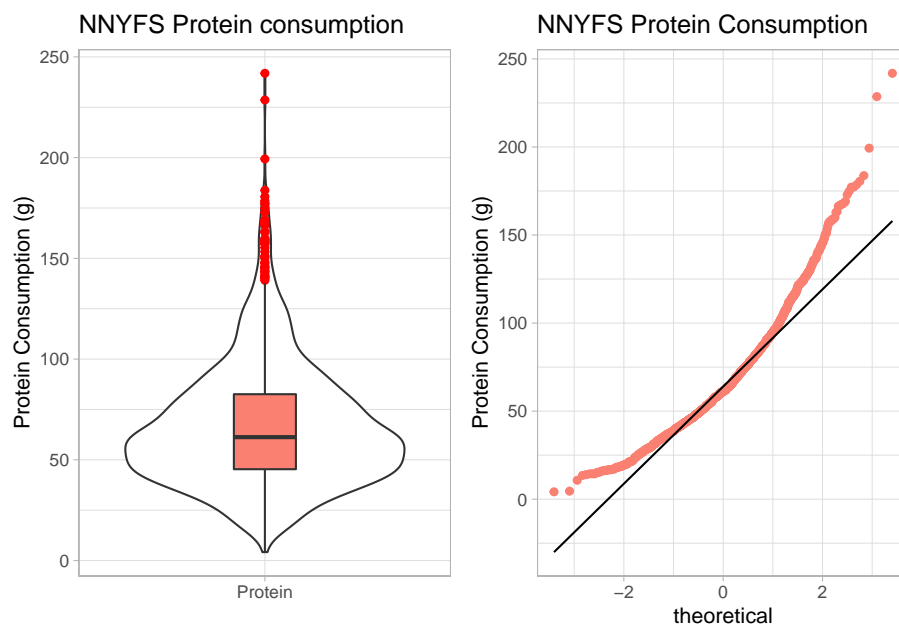

`plot_grid()` has some functionality that's easier to work with than `grid.arrange` for some people, so you might want to learn more about it. Check out its web site at https://wilkelab.org/cowplot/.

9.3.2 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, which must be installed via Github using `devtools` at this writing.

```
## must install patchwork from Github using devtools
## with devtools::install_github("thomasp85/patchwork")
```

```r
library(patchwork)

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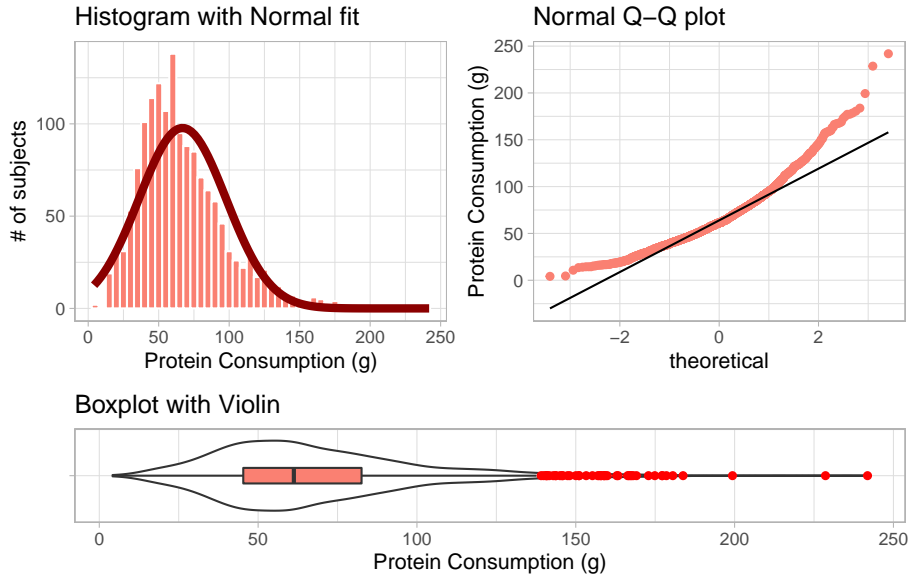
For more on the `patchwork` package, check out its repository at https://github.com/thomasp85/patchwork.

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?

```r
## must install patchwork from Github using devtools
## with devtools::install_github("thomasp85/patchwork")
## library(patchwork)

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")
```
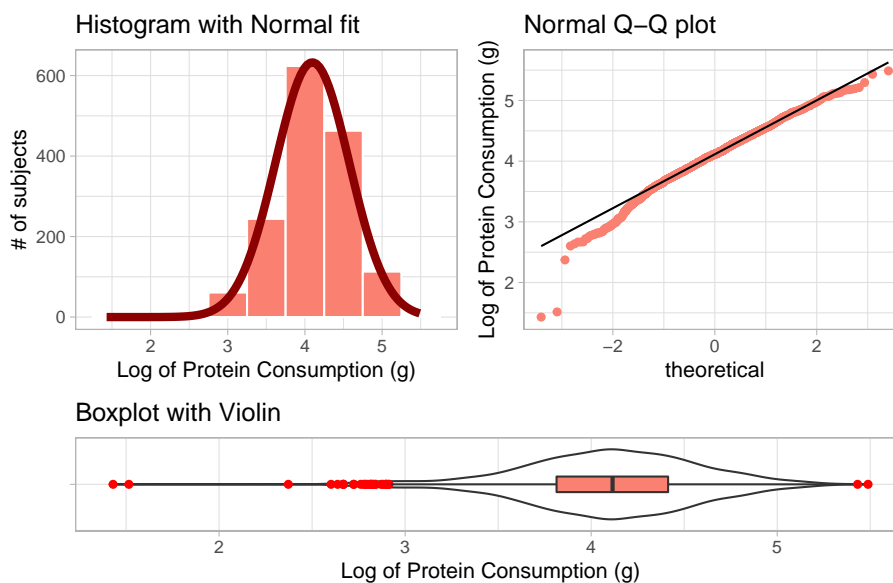
Square Root of NNYFS Protein Consumption
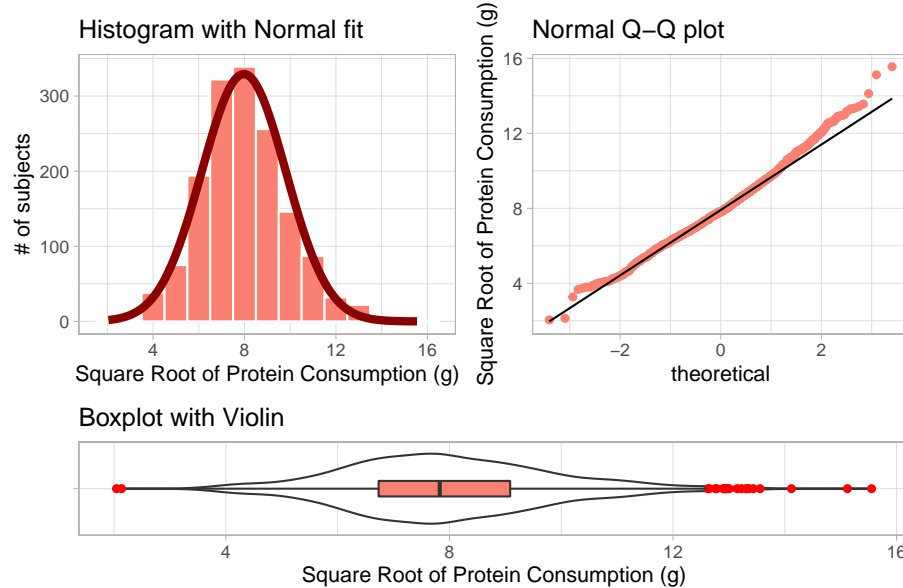
That looks like a more symmetric distribution, certainly, although we still have some outliers on the right side of the distribution. Should we take another step down the ladder?

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.

```
## must install patchwork from Github using devtools
## with devtools::install_github("thomasp85/patchwork")
## library(patchwork)

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 = 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")
```


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.

9.5 What if we considered all 9 available transformations?

```
## must install patchwork from Github using devtools
## with devtools::install_github("thomasp85/patchwork")
## library(patchwork)

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") +
```

```r
    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/(protein^2))) +
    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")
```

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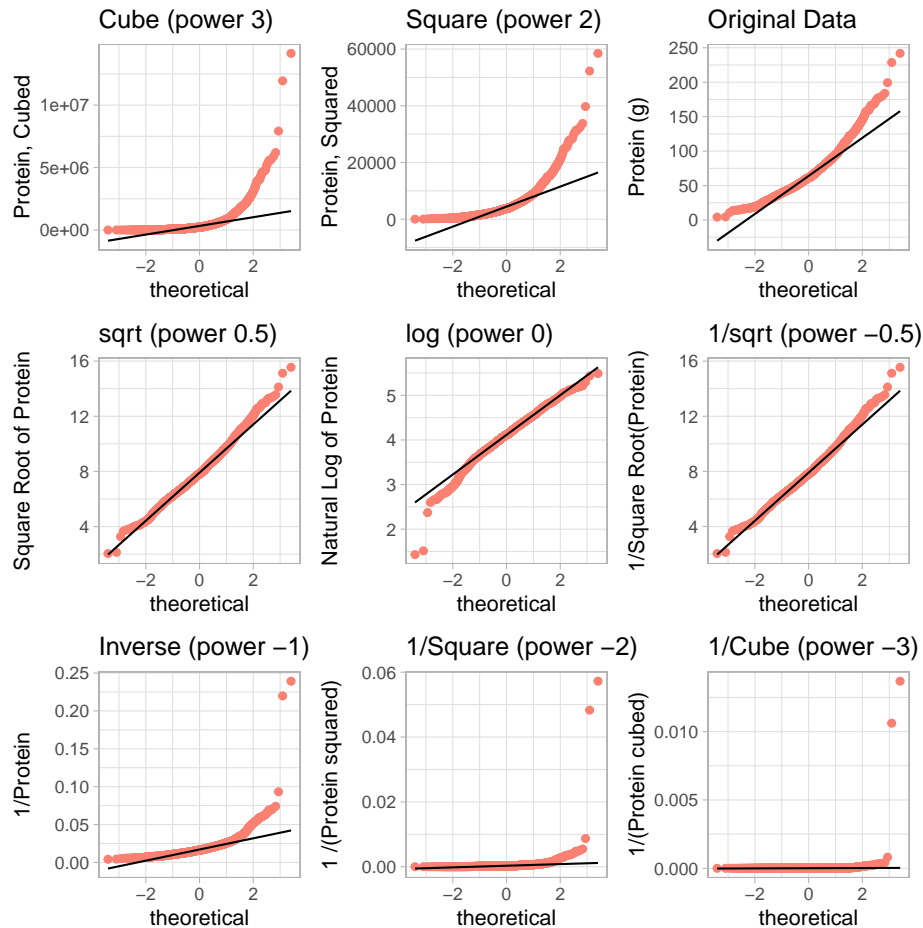
The square root still appears to be the best choice of transformation here, even after we consider all 8 transformation of the raw data.

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?

```r
## must install patchwork from Github using devtools
## with devtools::install_github("thomasp85/patchwork")
## library(patchwork)

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
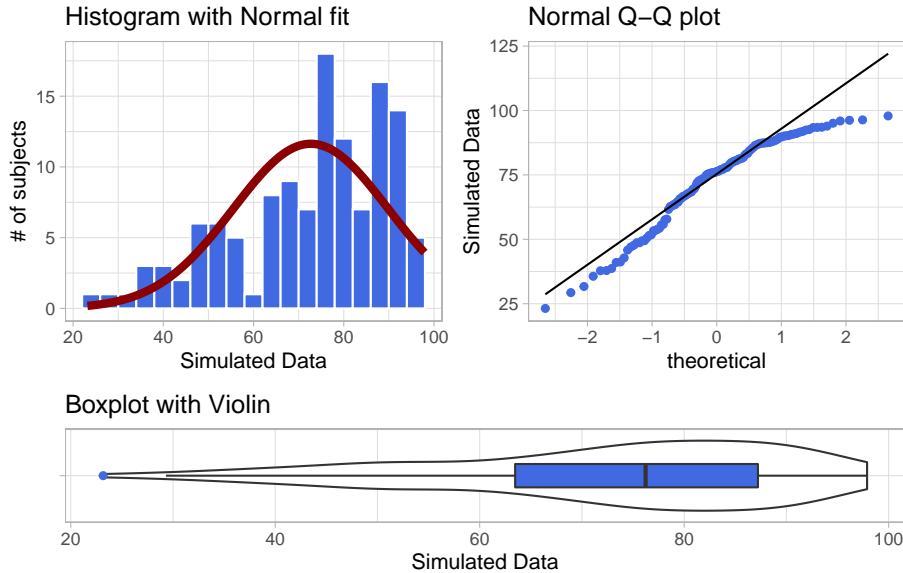
Given the left skew in the data, it looks like a step up in the ladder is warranted, perhaps by looking at the square of the data?

```r
## must install patchwork from Github using devtools
## with devtools::install_github("thomasp85/patchwork")
## library(patchwork)

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)) +
```

```r
    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")
```

Squared Simulated Data
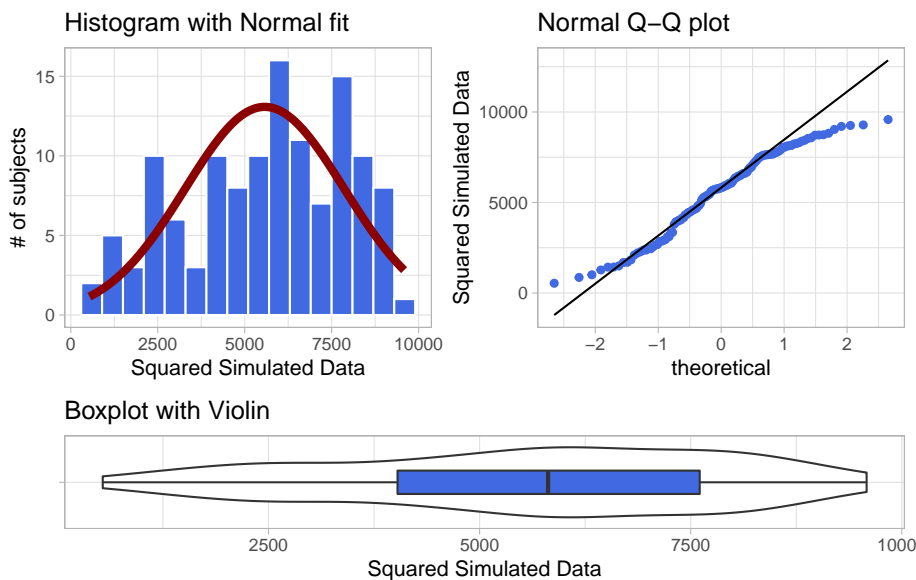
Looks like at best a modest improvement. How about cubing the data, instead?

```r
## must install patchwork from Github using devtools
## with devtools::install_github("thomasp85/patchwork")
## library(patchwork)

res <- mosaic::favstats(~ sample2^3, data = data2)
```

```r
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 = 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")
```

Cubed Simulated Data
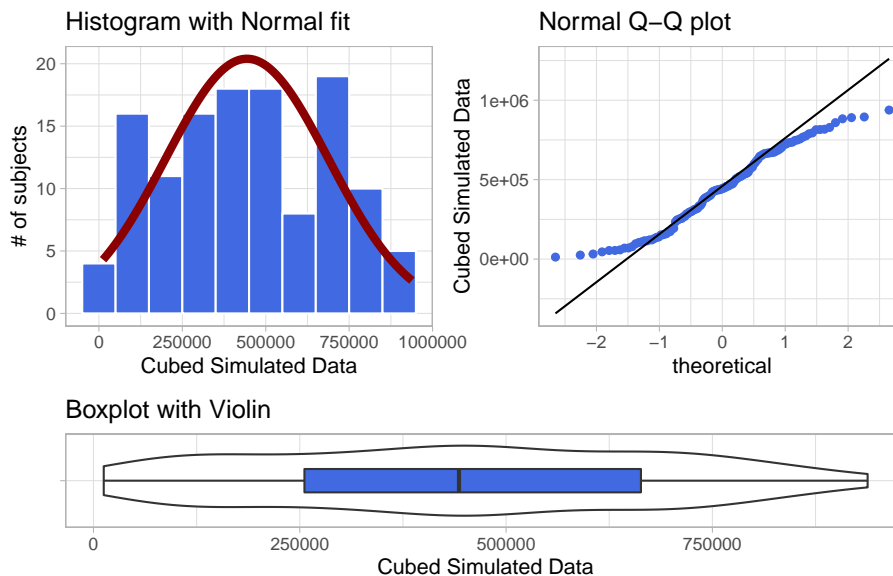
The newly transformed (cube of the) data appears more symmetric, although somewhat light-tailed. Perhaps a Normal model would be more appropriate now, although the standard deviation is likely to overstate the variation we see in the data due to the light tails. Again, I wouldn't be thrilled using a cube in practical work, as it is so hard to interpret, but it does look like a reasonable choice here.

9.6.1 Transformation Example 2 Ladder with Normal Q-Q Plots

9.7 What if we considered all 9 available transformations?

```r
## must install patchwork from Github using devtools
## with devtools::install_github("thomasp85/patchwork")
## library(patchwork)

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

```r
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") +
```
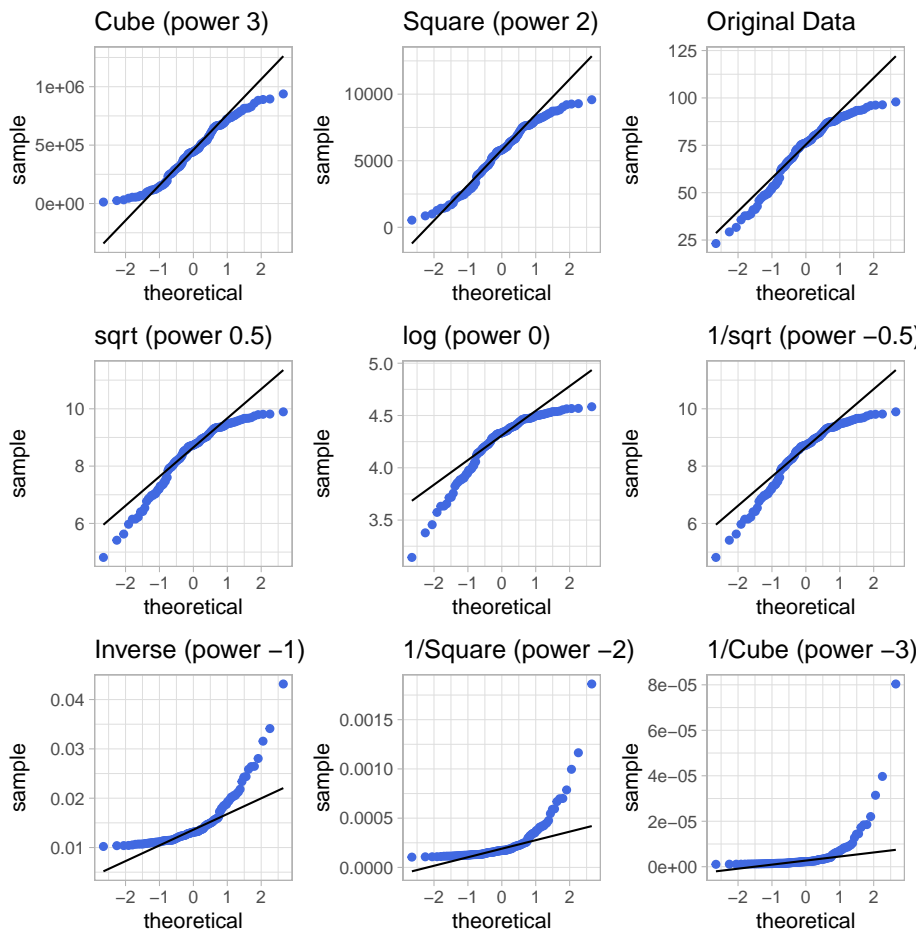
```
    theme_light() +
    labs(title = "1/Cube (power -3)")
```

```
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.

```
nnyfs %>%
    group_by(diet_yesterday) %>%
    select(diet_yesterday, energy, protein) %>%
    summarise_all(funs(median, 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 u~          2098           69.4       2150.         75.1
2 2_Usual                     1794           61.3       1858.         67.0
3 3_Much less than u~         1643           53.9       1779.         60.1
4 <NA>                        4348          155.        4348         155.
```

Perhaps we should restrict ourselves to the people who were not missing the `diet_yesterday` category, and look now at their `sugar` and `water` consumption.

```
nnyfs %>%
    filter(complete.cases(diet_yesterday)) %>%
```

```
    group_by(diet_yesterday) %>%
    select(diet_yesterday, energy, protein, sugar, water) %>%
    summarise_all(funs(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.

```
library(patchwork)

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) +
```

```r
    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
```
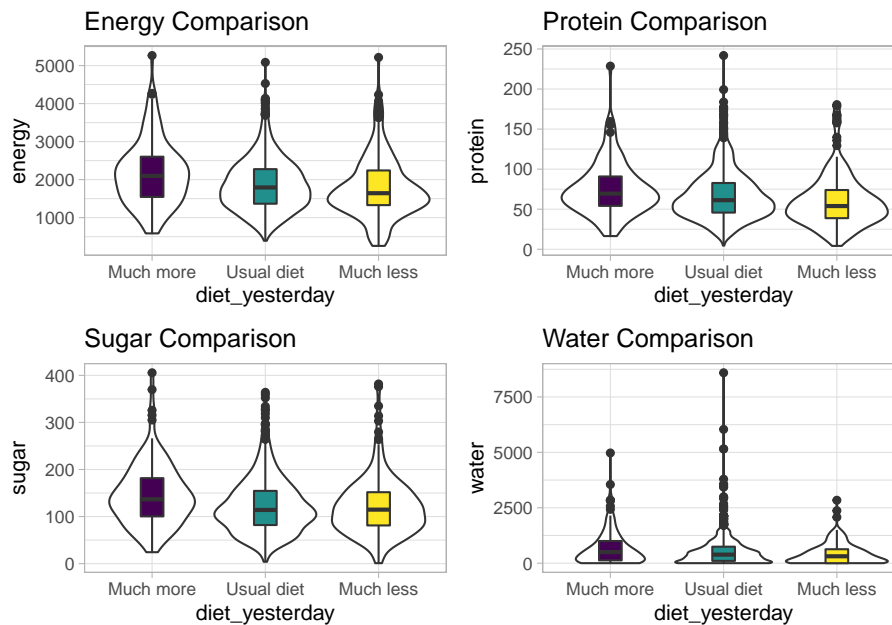

We can see that there is considerable overlap in these distributions, regardless of what we're measuring.

10.2 Another Example

Suppose now that we ask a different question. Do kids in larger categories of BMI have larger waist circumferences?

```r
nnyfs %>%
    group_by(bmi_cat) %>%
    summarise(mean = mean(waist), sd = sd(waist),
            median = median(waist),
```

```
                    skew_1 = round((mean(waist) - median(waist)) /
                                    sd(waist),2))
```

```
# A tibble: 5 x 5
  bmi_cat        mean      sd median skew_1
  <chr>         <dbl>   <dbl>  <dbl>  <dbl>
1 1_Underweight  55.2    7.58   54.5   0.09
2 2_Normal         NA     NaN     NA     NA
3 3_Overweight   72.3    11.9     74  -0.14
4 4_Obese          NA     NaN     NA     NA
5 <NA>             NA     NaN     NA     NA
```

Oops. Looks like we need to filter for cases with complete data on both BMI category and waist circumference in order to get meaningful results. We should add a count, too.

```
nnyfs %>%
    filter(complete.cases(bmi_cat, waist)) %>%
    group_by(bmi_cat) %>%
    summarise(count = n(), mean = mean(waist),
              sd = sd(waist), median = median(waist),
        skew_1 =
          round((mean(waist) - median(waist)) / sd(waist),2))
```

```
# A tibble: 4 x 6
  bmi_cat        count  mean     sd median skew_1
  <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)
```

```
        bmi_cat  min   Q1 median   Q3   max mean    sd   n missing
1 1_Underweight 42.5 49.3   54.5 62.4  68.5 55.2  7.58  41       0
2      2_Normal 44.1 53.9   59.5 68.4  89.2 61.2  9.35 917       3
3  3_Overweight 49.3 62.3   74.0 81.2 105.3 72.3 11.94 258       0
4       4_Obese 52.1 72.7   86.8 96.8 144.7 85.6 17.11 294       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")
```
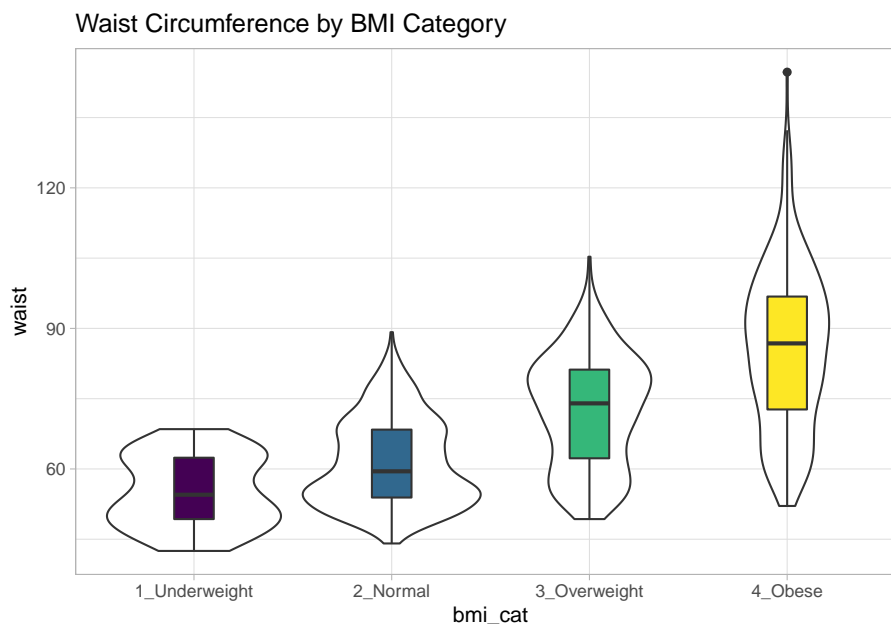

The data transformation with dplyr cheat sheet found under the Help menu in RStudio is a great resource. And, of course, for more details, visit Grolemund and Wickham (2019).

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).
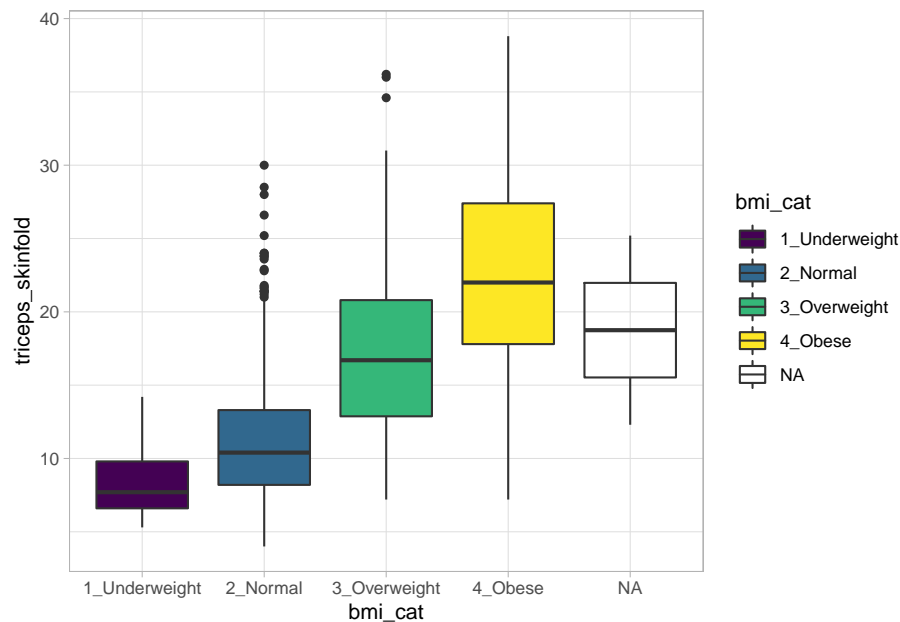
Again, we probably want to omit those missing values (both in `bmi_cat` and `triceps_skinfold`) and also eliminate the repetitive legend (guides) on the right.

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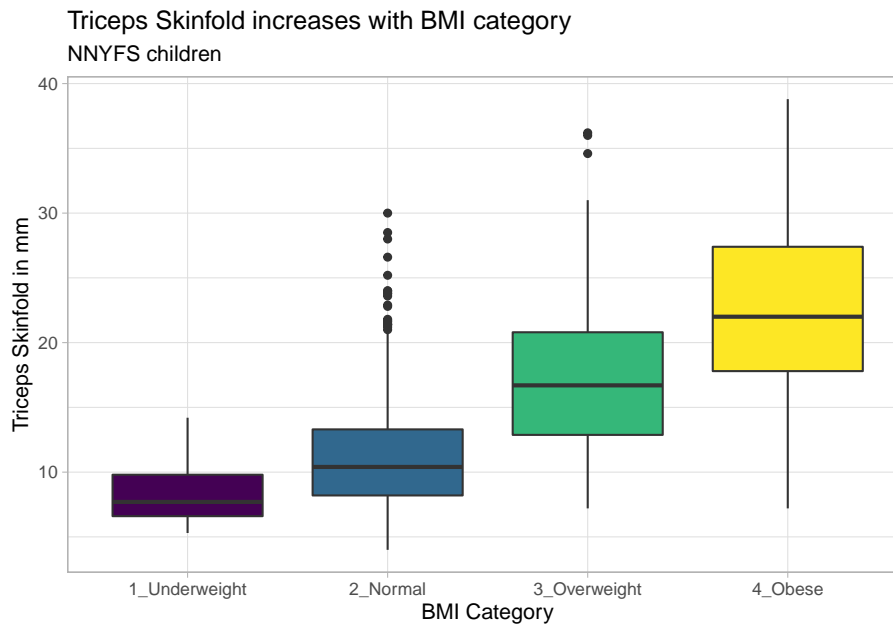
As always, the boxplot shows the five-number summary (minimum, 25th percentile, median, 75th percentile and maximum) in addition to highlighting candidate outliers.

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.y="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")
```

Triceps Skinfold increases with BMI category

NNYFS children
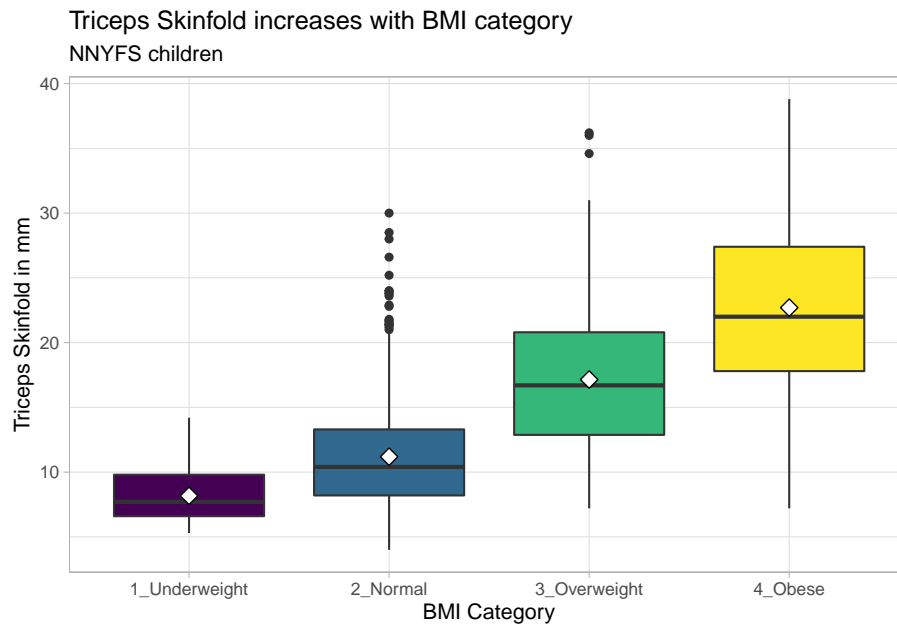
10.3.2   Adding Notches to a Boxplot

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,
                fill = bmi_cat)) +
    geom_boxplot(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")
```

Triceps Skinfold increases with BMI category

NNYFS children
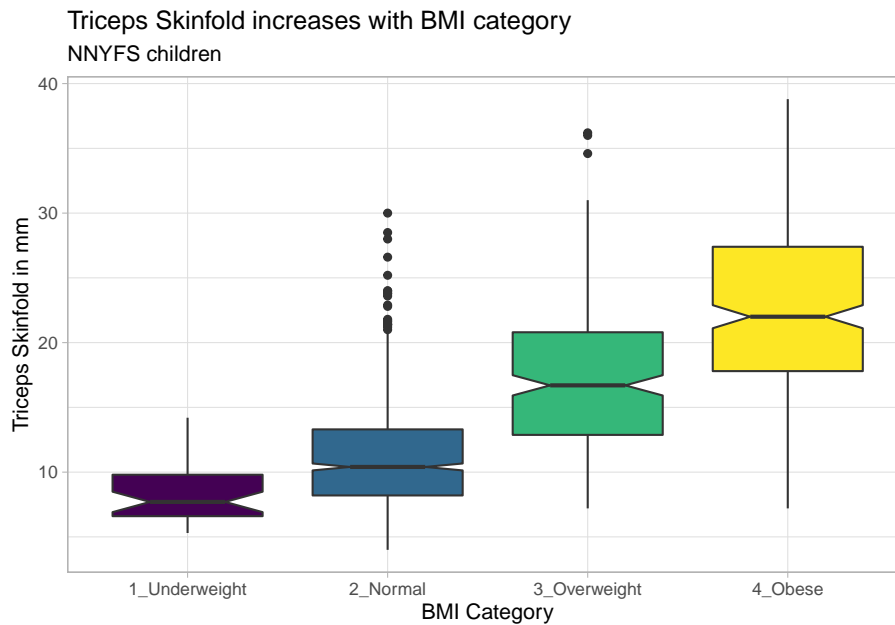
There is no overlap between the notches for each of the four categories, so we might reasonably conclude that the true median triceps skinfold values across the four categories are statistically significantly different.

For an example where the notches do overlap, consider the comparison of plank times by BMI category.

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

Plank Times by BMI category
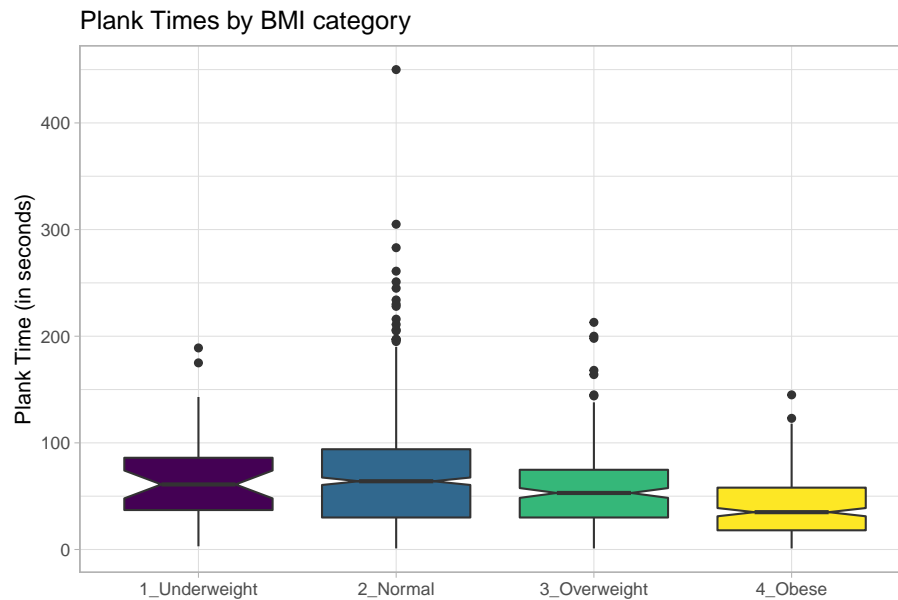
The overlap in the notches (for instance between Underweight and Normal) suggests that the median plank times in the population of interest don't necessarily differ in a meaningful way by BMI category, other than perhaps the Obese group which may have a shorter time.

These data are somewhat right skewed. Would a logarithmic transformation in the plot help us see the patterns more clearly?

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

log(Plank Times) by BMI category
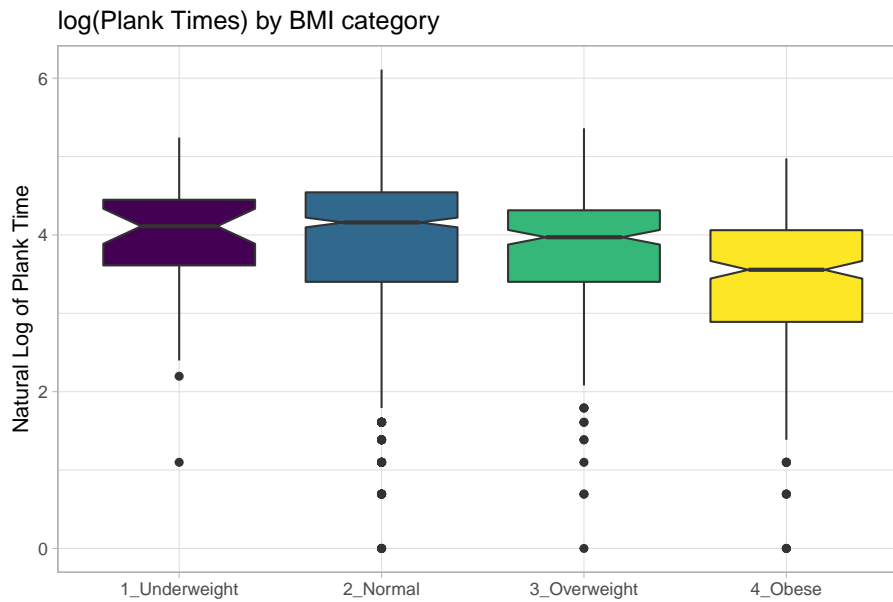
10.4 Using Multiple Histograms to Make Comparisons

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")
```
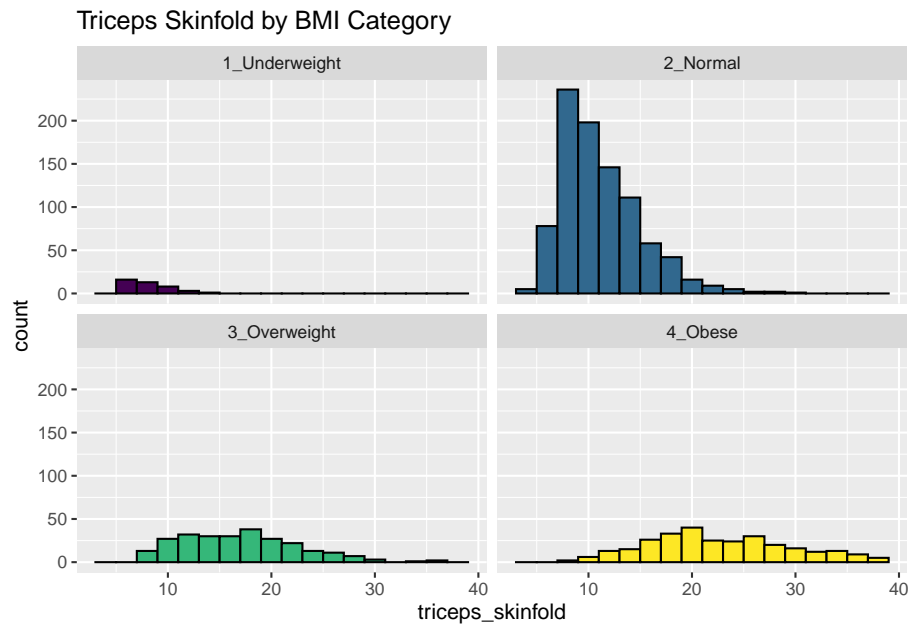
10.5 Using Multiple Density Plots to Make Comparisons

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

```
nnyfs %>%
    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")
```
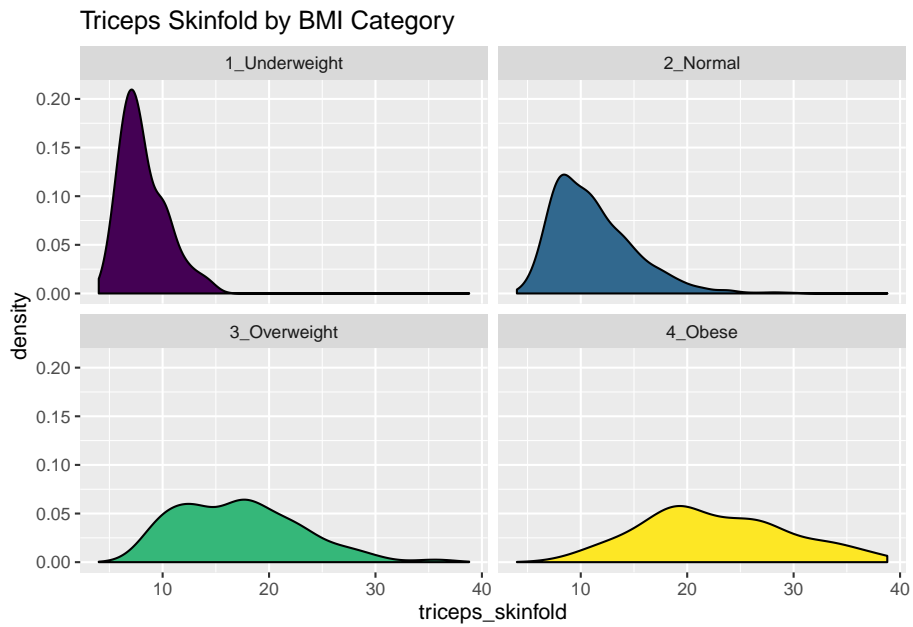
Or, we can plot all of the densities on top of each other with semi-transparent fills.

```
nnyfs %>%
    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")
```

Triceps Skinfold by BMI Category
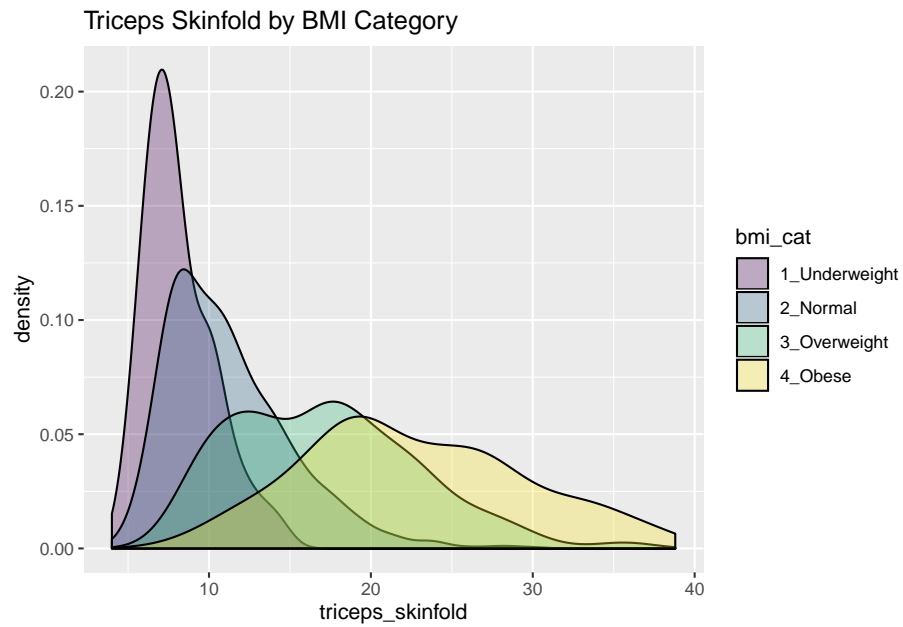
This really works better when we are comparing only two groups, like females to males.

```
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")
```

Triceps Skinfold by Sex
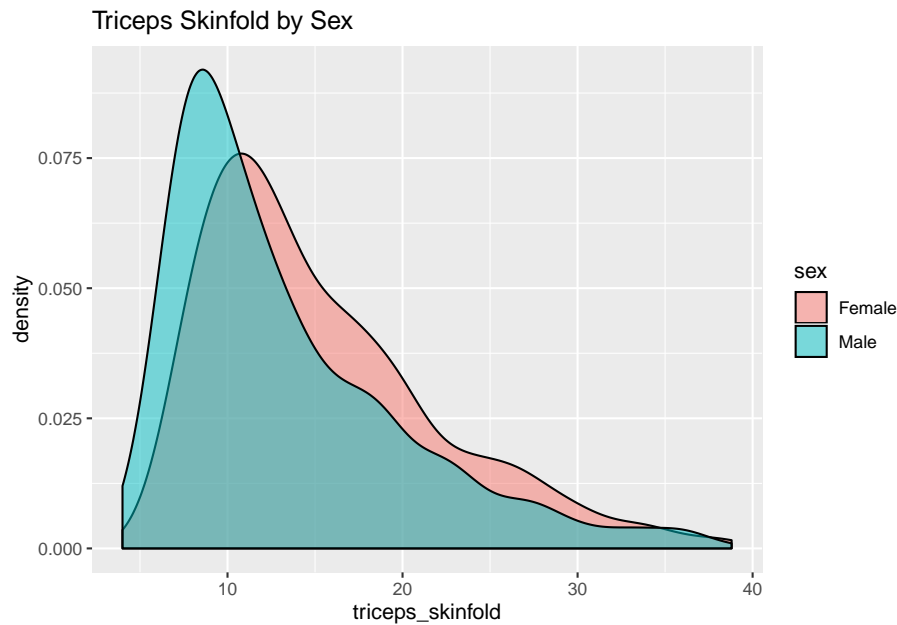

10.6 Building a Violin Plot

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")
```

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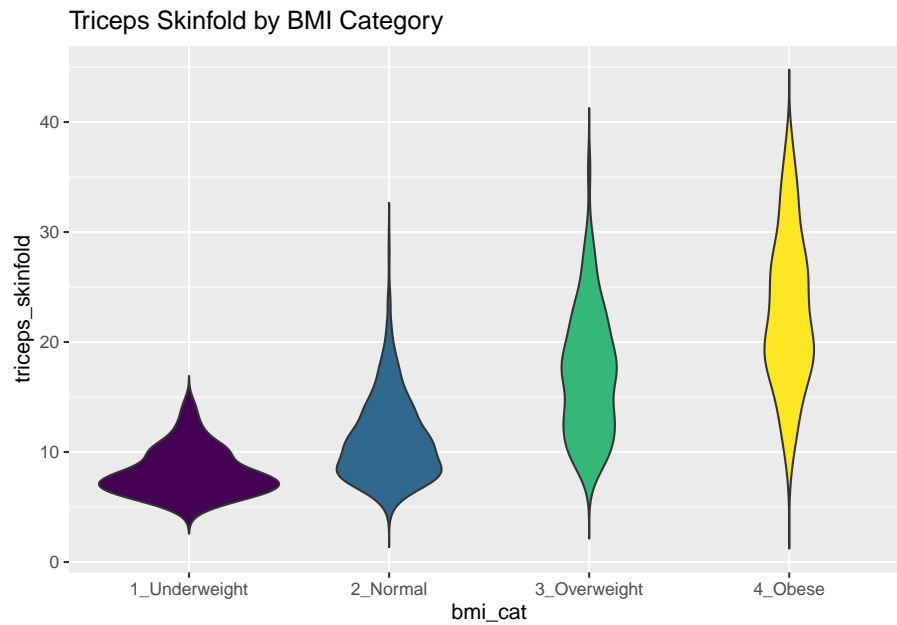
Traditionally, these plots are shown with overlaid boxplots and a white dot at the median, like this example, now looking at waist circumference again.

```
nnyfs %>%
    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.y=median, geom="point",
                 fill="white", shape=21, size=3) +
    scale_fill_viridis_d() +
    guides(fill = "none") +
    labs(title = "Waist Circumference by BMI Category")
```

Waist Circumference by BMI Category
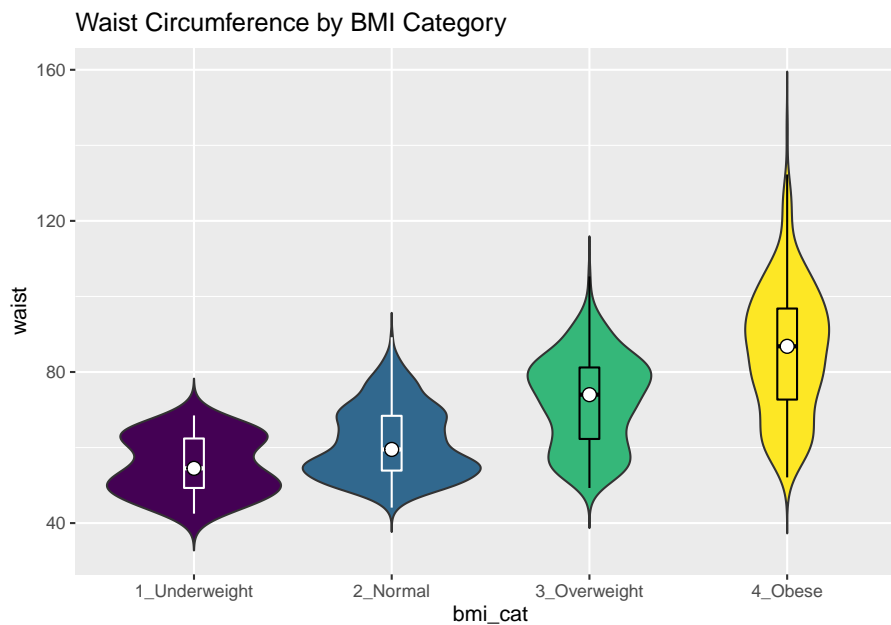
10.7 A Ridgeline Plot

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**[1].

```
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
```

---

[1]These were originally called joy plots, and the tools were contained in the `ggjoy` package
but that name and package has been deprecated in favor of `ggridges`.
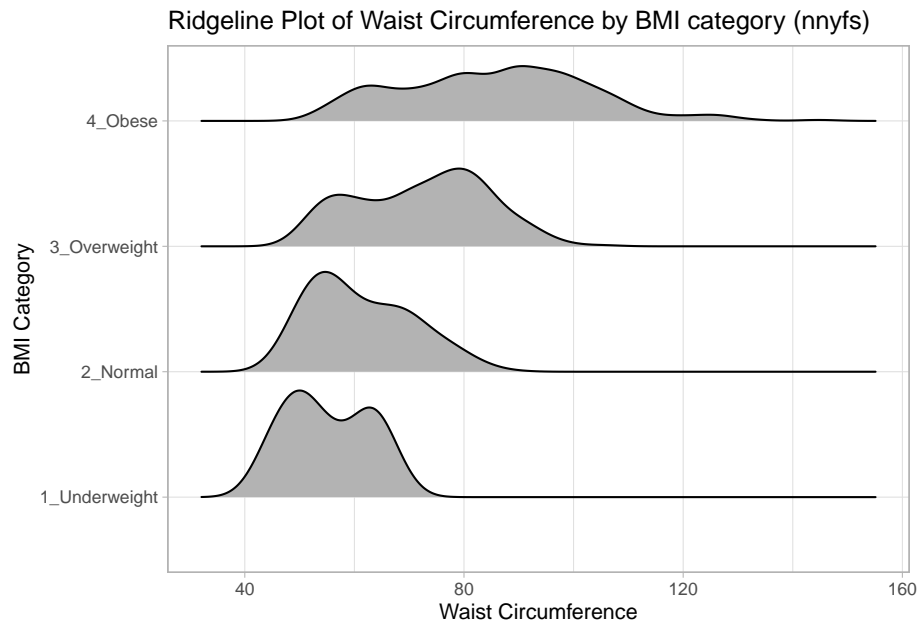
And here's a ridgeline plot for the triceps skinfolds. We'll start by sorting the subgroups by the median value of our outcome (triceps skinfold) in this case, though it turns out not to matter. We'll also add some color.

```r
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
```

Ridgeline Plot of Triceps Skinfold by BMI Category (nnyfs)
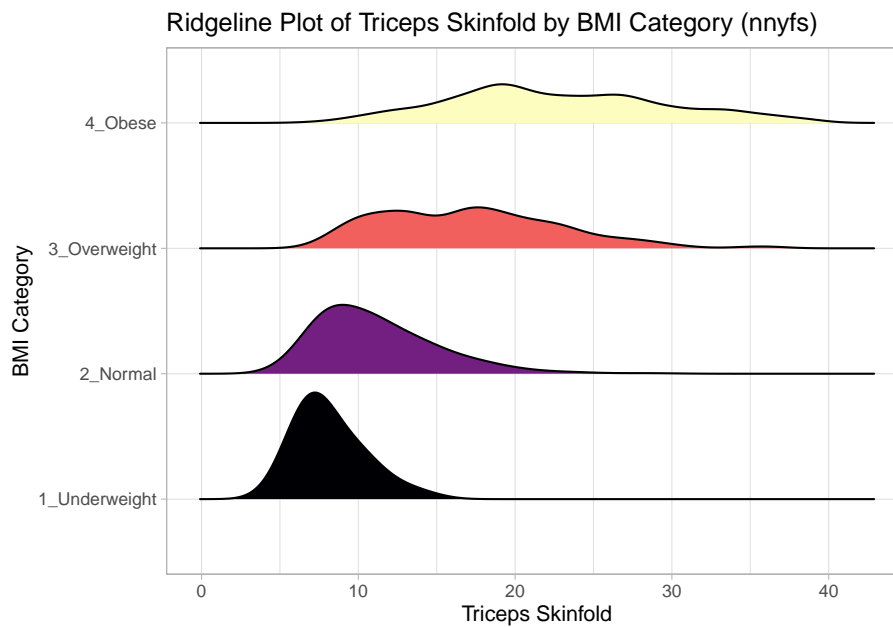
For one last example, we'll look at age by BMI category, so that sorting the BMI subgroups by the median matters, and we'll try an alternate color scheme, and a theme specially designed for the ridgeline plot.

```
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
```

Ridgeline Plot of Age at Exam by BMI category (nnyfs
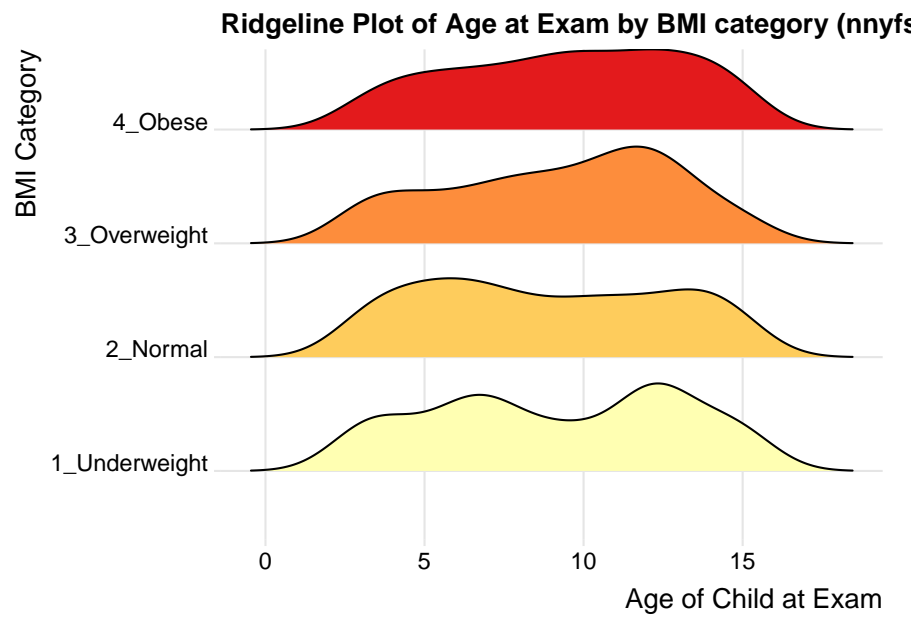
This is Work in Progress

Dr. Love is revising these notes and adding new materials regularly.

Materials will continue to be posted leading up to the start of class, as well as over the course of the Fall semester.

Baumer, Benjamin S., Daniel T. Kaplan, and Nicholas J. Horton. 2017. *Modern Data Science with R*. Boca Raton, FL: CRC Press. https://mdsr-book.github.io/.

Bock, David E., Paul F. Velleman, and Richard D. De Veaux. 2004. *Stats: Modelling the World*. Boston MA: Pearson Addison-Wesley.

Çetinkaya-Rundel, Mine. 2017. "Teaching Data Science to New useRs." bit.ly/user2017.

Fox, John, and Sanford Weisberg. 2011. *An R Companion to Applied Regression*. Second. Thousand Oaks CA: Sage. http://socserv.socsci.mcmaster.ca/jfox/Books/Companion.

Gelman, Andrew, and Jennifer Hill. 2007. *Data Analysis Using Regression and Multilevel-Hierarchical Models*. New York: Cambridge University Press. http://www.stat.columbia.edu/~gelman/arm/.

Gelman, Andrew, and Deborah Nolan. 2017. *Teaching Statistics: A Bag of Tricks*. Second. Oxford, UK: Oxford University Press.

Grolemund, Garrett, and Hadley Wickham. 2019. *R for Data Science*. O'Reilly. http://r4ds.had.co.nz/.

Harrell, Frank E., and James C. Slaughter. 2019. *Biostatistics for Biomedical Research*. Vanderbilt University School of Medicine. biostat.mc.vanderbilt.edu/ClinStat.

Ismay, Chester, and Albert Y. Kim. 2019. *ModernDive: Statistical Inference via Data Science*. http://moderndive.com/.

Norman, Geoffrey R., and David L. Streiner. 2014. *Biostatistics: The Bare Essentials*. Fourth. People's Medical Publishing House.

Vittinghoff, Eric, David V. Glidden, Stephen C. Shiboski, and Charles E. McCulloch. 2012. *Regression Methods in Biostatistics: Linear, Logistic, Sur-*

vival, and Repeated Measures Models. Second. Springer-Verlag, Inc. http://www.biostat.ucsf.edu/vgsm/.

Wainer, Howard. 1997. *Visual Revelations: Graphical Tales of Fate and Deception from Napoleon Bonaparte to Ross Perot.* New York: Springer-Verlag.

———. 2005. *Graphic Discovery: A Trout in the Milk and Other Visual Adventures.* Princeton, NJ: Princeton University Press.

———. 2013. *Medical Illuminations: Using Evidence, Visualization and Statistical Thinking to Improve Healthcare.* New York: Oxford University Press.