

Notes for 431

Thomas E. Love, Ph.D.

2022-08-25

Table of contents

| | |
|--|-----------|
| Working with These Notes | 11 |
| What You'll Find Here | 11 |
| The 431 Course online | 12 |
| Setting Up R | 13 |
| R Markdown | 13 |
| R Packages | 13 |
| The <code>Love-boost.R</code> script | 14 |
| Packages Used in these Notes | 14 |
| The <code>tidyverse</code> | 15 |
| 1 Data Science and 431 | 17 |
| 1.1 Data Science Project Cycle | 18 |
| 1.2 Data Science and the 431 Course | 18 |
| 1.3 What The Course Is and Isn't | 19 |
| I Part A. Exploring Data | 21 |
| 2 The Palmer Penguins | 22 |
| 2.1 Setup: Packages Used Here | 22 |
| 2.2 Viewing a Data Set | 23 |
| 2.3 Create <code>newpenguins</code> : Eliminating Missing Data | 23 |
| 2.4 Counting Things and Making Tables | 24 |
| 2.5 Creating a Scatterplot | 26 |
| 2.6 Six Ways To "Improve" This Graph | 28 |
| 2.7 A Little Reflection | 29 |
| 3 Summarizing Penguins | 31 |
| 3.1 Setup: Packages Used Here | 31 |
| 3.2 Our Data Set | 31 |
| 3.3 Numerical Summaries for a Tibble | 32 |
| 3.3.1 Using <code>summary()</code> | 32 |
| 3.3.2 Using <code>inspect()</code> from <code>mosaic</code> | 33 |
| 3.3.3 Using <code>describe()</code> from <code>psych</code> | 33 |
| 3.3.4 Using <code>describe()</code> from <code>Hmisc</code> | 34 |

| | | |
|----------|---|-----------|
| 3.3.5 | Using <code>tbl_summary()</code> from <code>gtsummary</code> | 36 |
| 3.3.6 | Visualizing with <code>visdat</code> functions | 37 |
| 3.4 | Histograms for a Variable | 39 |
| 3.5 | Comparing Penguins by Species Numerically | 44 |
| 3.5.1 | Using <code>tbl_summary()</code> to summarize the tibble | 45 |
| 3.6 | Comparing Penguins by Species Graphically | 46 |
| 3.6.1 | Faceting Histograms | 46 |
| 3.6.2 | Using <code>facet_grid()</code> | 47 |
| 3.6.3 | Boxplots | 48 |
| 3.6.4 | Adding Violins | 49 |
| 3.6.5 | Letter-Value Plots (Boxplots for Large Data) | 50 |
| 4 | NHANES Data | 52 |
| 4.1 | Setup: Packages Used Here | 52 |
| 4.2 | The NHANES data: A First Sample | 52 |
| 4.3 | Sampling NHANES Adults | 53 |
| 4.3.1 | Creating a Temporary, Cleaner Data Frame | 53 |
| 4.3.2 | Sampling <code>nh_temp</code> to obtain our <code>nh_adult750</code> sample | 55 |
| 4.3.3 | Summarizing the Data's Structure | 56 |
| 4.3.4 | What are the variables? | 57 |
| 4.4 | Counting Missing Values | 59 |
| 4.5 | Sampling 500 Complete Cases | 62 |
| 4.6 | Saving our Samples in <code>.Rds</code> files | 64 |
| 4.7 | Coming Up | 64 |
| 5 | Types of Data | 65 |
| 5.1 | Setup: Packages Used Here | 65 |
| 5.2 | Data require structure and context | 65 |
| 5.3 | Loading in the “Complete Cases” Sample | 66 |
| 5.4 | Quantitative Variables | 66 |
| 5.5 | Quantitative Variables in <code>nh_500cc</code> | 67 |
| 5.5.1 | A look at BMI (Body-Mass Index) | 68 |
| 5.5.2 | Types of Quantitative Variables | 69 |
| 5.6 | Qualitative (Categorical) Variables | 69 |
| 5.6.1 | Nominal vs. Ordinal Categories | 70 |
| 5.7 | Tabulating Binary Variables | 71 |
| 5.8 | Tabulating Multi-Categorical Variables | 72 |
| 5.9 | Coming Up | 74 |
| 6 | More NHANES Summaries | 75 |
| 6.1 | Setup: Packages Used Here | 75 |
| 6.2 | Re-Loading the “Complete Cases” Sample | 75 |

| | | |
|----------|---|------------|
| 6.3 | Distribution of Heights | 75 |
| 6.3.1 | Changing a Histogram's Fill and Color | 77 |
| 6.3.2 | Using a frequency polygon | 78 |
| 6.3.3 | Using a dotplot | 79 |
| 6.4 | Height and Sex | 80 |
| 6.4.1 | A Boxplot of Height by Sex | 82 |
| 6.4.2 | Adding a violin plot | 83 |
| 6.4.3 | Histograms of Height by Sex | 86 |
| 6.5 | Visualizing Age and Height's Relationship, by Sex | 88 |
| 6.5.1 | Adding Color to the plot | 89 |
| 6.5.2 | Can we show the Female and Male relationships in separate panels? | 90 |
| 6.5.3 | Can we add a smooth curve to show the relationship in each plot? | 91 |
| 6.5.4 | What if we want to assume straight line relationships? | 92 |
| 6.6 | Combining Plots with <code>patchwork</code> | 93 |
| 6.7 | Looking at Pulse Rate | 95 |
| 6.7.1 | Pulse Rate and Physical Activity | 97 |
| 6.7.2 | Pulse by Sleeping Trouble | 99 |
| 6.7.3 | Pulse and HealthGen | 100 |
| 6.7.4 | Pulse Rate and Systolic Blood Pressure | 101 |
| 6.7.5 | Sleep Trouble vs. No Sleep Trouble? | 102 |
| 6.8 | General Health Status | 103 |
| 6.8.1 | Bar Chart for Categorical Data | 104 |
| 6.9 | Two-Way Tables | 107 |
| 6.10 | SBP by General Health Status | 109 |
| 6.10.1 | SBP by Physical Activity and General Health Status | 110 |
| 6.10.2 | SBP by Sleep Trouble and General Health Status | 111 |
| 6.11 | Conclusions | 112 |
| 7 | Summarizing Quantities | 113 |
| 7.1 | Setup: Packages Used Here | 113 |
| 7.2 | Working with the <code>nh_750</code> data | 113 |
| 7.3 | The <code>summary</code> function for Quantitative data | 114 |
| 7.4 | Measuring the Center of a Distribution | 115 |
| 7.4.1 | The Mean and The Median | 115 |
| 7.4.2 | Dealing with Missingness | 117 |
| 7.4.3 | The Mode of a Quantitative Variable | 118 |
| 7.5 | Measuring the Spread of a Distribution | 119 |
| 7.5.1 | The Range and the Interquartile Range (IQR) | 119 |
| 7.5.2 | The Variance and the Standard Deviation | 121 |
| 7.5.3 | Obtaining the Variance and Standard Deviation in R | 121 |
| 7.5.4 | Defining the Variance and Standard Deviation | 122 |
| 7.5.5 | Interpreting the SD when the data are Normally distributed | 122 |
| 7.5.6 | Chebyshev's Inequality: One Interpretation of the Standard Deviation . | 124 |

| | | |
|-----------|---|------------|
| 7.6 | Measuring the Shape of a Distribution | 124 |
| 7.6.1 | Multimodal vs. Unimodal distributions | 124 |
| 7.6.2 | Skew | 125 |
| 7.6.3 | Kurtosis | 126 |
| 7.7 | Numerical Summaries for Quantitative Variables | 128 |
| 7.7.1 | <code>favstats</code> in the <code>mosaic</code> package | 128 |
| 7.7.2 | <code>describe</code> in the <code>psych</code> package | 130 |
| 7.7.3 | The <code>Hmisc</code> package's version of <code>describe</code> | 131 |
| 7.7.4 | Other options | 132 |
| 8 | Summarizing Categories | 133 |
| 8.1 | Setup: Packages Used Here | 133 |
| 8.2 | Using the <code>nh_adult750</code> data again | 133 |
| 8.3 | The <code>summary</code> function for Categorical data | 134 |
| 8.4 | Tables to describe One Categorical Variable | 134 |
| 8.5 | Constructing Tables Well | 136 |
| 8.5.1 | Alabama First! | 136 |
| 8.5.2 | ALL is different and important | 137 |
| 8.6 | The Mode of a Categorical Variable | 137 |
| 8.7 | <code>describe</code> in the <code>Hmisc</code> package | 138 |
| 8.8 | Cross-Tabulations of Two Variables | 140 |
| 8.9 | Cross-Classifying Three Categorical Variables | 144 |
| 8.10 | Gaining Control over Tables in R: the <code>gt</code> package | 146 |
| 9 | Missing Data and Single Imputation | 147 |
| 9.1 | Setup: Packages Used Here | 147 |
| 9.2 | A Simulated Example with 15 subjects | 147 |
| 9.3 | Identifying missingness with <code>naniar</code> functions | 148 |
| 9.4 | Missing-data mechanisms | 152 |
| 9.5 | Options for Dealing with Missingness | 152 |
| 9.6 | Complete Case (and Available Case) analyses | 153 |
| 9.7 | Single Imputation | 153 |
| 9.8 | Multiple Imputation | 153 |
| 9.9 | Building a Complete Case Analysis | 154 |
| 9.10 | Single Imputation with the Mean or Mode | 154 |
| 9.11 | Doing Single Imputation with <code>simputation</code> | 155 |
| 10 | National Youth Fitness Survey | 158 |
| 10.1 | Setup: Packages Used Here | 158 |
| 10.2 | What is the NHANES NYFS? | 158 |
| 10.3 | The Variables included in <code>nnyfs</code> | 159 |
| 10.3.1 | From the NNYFS Demographic Component | 159 |
| 10.3.2 | From the NNYFS Dietary Component | 159 |

| | | |
|---------|--|-----|
| 10.3.3 | From the NNYFS Examination Component | 160 |
| 10.3.4 | From the NNYFS Questionnaire Component | 160 |
| 10.4 | Looking over A Few Variables | 162 |
| 10.4.1 | <code>SEQN</code> | 164 |
| 10.4.2 | <code>sex</code> | 164 |
| 10.4.3 | <code>age_child</code> | 165 |
| 10.4.4 | <code>race_eth</code> | 166 |
| 10.4.5 | <code>income_pov</code> | 167 |
| 10.4.6 | <code>bmi</code> | 169 |
| 10.4.7 | <code>bmi_cat</code> | 170 |
| 10.4.8 | <code>waist</code> | 171 |
| 10.4.9 | <code>triceps_skinfold</code> | 172 |
| 10.5 | Additional Numeric Summaries | 173 |
| 10.5.1 | The Five Number Summary, Quantiles and IQR | 173 |
| 10.6 | Additional Summaries from <code>favstats</code> | 175 |
| 10.7 | The Histogram | 175 |
| 10.7.1 | Freedman-Diaconis Rule to select bin width | 176 |
| 10.7.2 | A Note on Colors | 178 |
| 10.8 | The Frequency Polygon | 180 |
| 10.9 | Plotting the Probability Density Function | 181 |
| 10.10 | The Boxplot | 182 |
| 10.10.1 | Drawing a Boxplot for One Variable in <code>ggplot2</code> | 182 |
| 10.10.2 | About the Boxplot | 183 |
| 10.11A | Simple Comparison Boxplot | 184 |
| 10.12 | Using <code>describe</code> in the <code>psych</code> library | 187 |
| 10.12.1 | The Trimmed Mean | 188 |
| 10.12.2 | The Median Absolute Deviation | 188 |
| 10.13 | Assessing Skew | 188 |
| 10.13.1 | Non-parametric Skewness | 189 |
| 10.14 | Assessing Kurtosis (Heavy-Tailedness) | 189 |
| 10.14.1 | The Standard Error of the Sample Mean | 190 |
| 10.15 | The <code>describe</code> function in the <code>Hmisc</code> package | 190 |
| 10.16 | Summarizing data within subgroups | 192 |
| 10.17 | Another Example | 194 |
| 10.18 | Boxplots to Relate an Outcome to a Categorical Predictor | 196 |
| 10.18.1 | Augmenting the Boxplot with the Sample Mean | 198 |
| 10.19 | Building a Violin Plot | 199 |
| 10.19.1 | Adding Notches to a Boxplot | 201 |
| 10.20 | Using Multiple Histograms to Make Comparisons | 204 |
| 10.21 | Using Multiple Density Plots to Make Comparisons | 205 |
| 10.22A | Ridgeline Plot | 208 |
| 10.23 | What Summaries to Report | 211 |

| | |
|---|------------|
| 11 Assessing Normality | 212 |
| 11.1 Setup: Packages Used Here | 212 |
| 11.2 Introduction | 212 |
| 11.3 Empirical Rule Interpretation of the Standard Deviation | 213 |
| 11.4 Describing Outlying Values with Z Scores | 214 |
| 11.4.1 Fences and Z Scores | 214 |
| 11.5 Comparing a Histogram to a Normal Distribution | 214 |
| 11.5.1 Histogram of <code>energy</code> with Normal model (with Counts) | 215 |
| 11.6 Does a Normal model work well for the <code>waist</code> circumference? | 217 |
| 11.7 The Normal Q-Q Plot | 219 |
| 11.8 Interpreting the Normal Q-Q Plot | 219 |
| 11.8.1 Data from a Normal distribution shows up as a straight line in a Normal Q-Q plot | 220 |
| 11.8.2 Skew is indicated by monotonic curves in the Normal Q-Q plot | 221 |
| 11.8.3 Direction of Skew | 224 |
| 11.8.4 Outlier-proneness is indicated by “s-shaped” curves in a Normal Q-Q plot | 224 |
| 11.9 Can a Normal Distribution Fit the <code>nnyfs energy</code> data Well? | 228 |
| 11.10 The Ladder of Power Transformations | 232 |
| 11.11 Using the Ladder | 233 |
| 11.12 Protein Consumption in the NNYFS data | 233 |
| 11.12.1 Using <code>patchwork</code> to compose plots | 235 |
| 11.13 Can we transform the <code>protein</code> data? | 236 |
| 11.13.1 The Square Root | 236 |
| 11.13.2 The Logarithm | 238 |
| 11.13.3 This course uses Natural Logarithms, unless otherwise specified | 240 |
| 11.14 What if we considered all 9 available transformations? | 240 |
| 11.15 A Simulated Data Set | 243 |
| 11.16 What if we considered all 9 available transformations? | 247 |
| 12 Straight Line Models | 250 |
| 12.1 Setup: Packages Used Here | 250 |
| 12.2 Assessing A Scatterplot | 250 |
| 12.2.1 Highlighting an unusual point | 251 |
| 12.2.2 Adding a Scatterplot Smooth using <code>loess</code> | 253 |
| 12.2.3 What Line Does R Fit? | 255 |
| 12.3 Correlation Coefficients | 257 |
| 12.4 The Pearson Correlation Coefficient | 258 |
| 12.5 Studying Correlation through 6 Examples | 258 |
| 12.5.1 Data Set Alex | 259 |
| 12.5.2 Data Set Bonnie | 262 |
| 12.5.3 Correlations for All Six Data Sets in the <code>Correx1</code> Example | 264 |
| 12.5.4 Data Set Colin | 265 |
| 12.5.5 Draw the Picture! | 265 |

| | |
|---|------------|
| 12.6 Estimating Correlation from Scatterplots | 267 |
| 12.7 The Spearman Rank Correlation | 271 |
| 12.7.1 Spearman Formula | 272 |
| 12.7.2 Comparing Pearson and Spearman Correlations | 272 |
| 12.7.3 Spearman vs. Pearson Example 1 | 272 |
| 12.7.4 Spearman vs. Pearson Example 2 | 274 |
| 12.7.5 Spearman vs. Pearson Example 3 | 275 |
| 12.7.6 Spearman vs. Pearson Example 4 | 276 |
| 13 Linearizing Transformations | 278 |
| 13.1 Setup: Packages Used Here | 278 |
| 13.2 “Linearize” The Association between Quantitative Variables | 278 |
| 13.3 The Box-Cox Plot | 278 |
| 13.3.1 A Few Caveats | 279 |
| 13.4 A Simulated Example | 279 |
| 13.5 Checking on a Transformation or Re-Expression | 282 |
| 13.5.1 Checking the Correlation Coefficients | 283 |
| 13.5.2 Using the <code>testTransform</code> function | 283 |
| 13.5.3 Comparing the Residual Plots | 284 |
| 13.6 An Example from the NNYFS data | 285 |
| 13.6.1 Pearson correlation and scatterplot | 286 |
| 13.6.2 Plotting the Residuals | 287 |
| 13.6.3 Using the Box-Cox approach to identify a transformation | 289 |
| 13.6.4 Plots after Inverse Transformation | 289 |
| 14 Studying Crab Claws | 292 |
| 14.1 Setup: Packages Used Here | 292 |
| 14.2 The Data | 292 |
| 14.3 Association of Size and Force | 295 |
| 14.4 The <code>loess</code> smooth | 297 |
| 14.4.1 Smoothing within Species | 299 |
| 14.5 Fitting a Linear Regression Model | 301 |
| 14.6 Is a Linear Model Appropriate? | 303 |
| 14.6.1 The log-log model | 304 |
| 14.6.2 How does this compare to our original linear model? | 305 |
| 14.7 Making Predictions with a Model | 306 |
| 14.7.1 Predictions After a Transformation | 307 |
| 14.7.2 Comparing Model Predictions | 308 |
| 15 Dehydration Recovery | 310 |
| 15.1 Setup: Packages Used Here | 310 |
| 15.2 The Data | 310 |
| 15.3 A Scatterplot Matrix | 311 |

| | | |
|-----------|--|------------|
| 15.4 | Are the recovery scores well described by a Normal model? | 312 |
| 15.5 | Simple Regression: Using Dose to predict Recovery | 314 |
| 15.6 | The Scatterplot, with fitted Linear Model | 314 |
| 15.7 | The Fitted Linear Model | 315 |
| 15.7.1 | Confidence Intervals | 315 |
| 15.8 | Coefficient Plots | 316 |
| 15.9 | The Summary Output | 317 |
| 15.9.1 | Model Specification | 318 |
| 15.9.2 | Residual Summary | 318 |
| 15.9.3 | Coefficients Output | 319 |
| 15.9.4 | Correlation and Slope | 319 |
| 15.9.5 | Coefficient Testing | 320 |
| 15.9.6 | Summarizing the Quality of Fit | 321 |
| 15.9.7 | ANOVA F test | 323 |
| 15.10 | Viewing the complete ANOVA table | 323 |
| 15.11 | Using <code>glance</code> to summarize the model's fit | 324 |
| 15.12 | Plotting Residuals vs. Fitted Values | 325 |
| 16 | The WCGS | 328 |
| 16.1 | Setup: Packages Used Here | 328 |
| 16.2 | The Western Collaborative Group Study (<code>wcgs</code>) data set | 328 |
| 16.2.1 | Structure of <code>wcgs</code> | 329 |
| 16.2.2 | Codebook for <code>wcgs</code> | 330 |
| 16.2.3 | Quick Summary | 331 |
| 16.3 | Are the SBPs Normally Distributed? | 332 |
| 16.4 | Identifying and Describing SBP outliers | 335 |
| 16.5 | Does Weight Category Relate to SBP? | 337 |
| 16.6 | Re-Leveling a Factor | 337 |
| 16.6.1 | SBP by Weight Category | 338 |
| 16.7 | Are Weight and SBP Linked? | 340 |
| 16.8 | SBP and Weight by Arcus Senilis groups? | 341 |
| 16.9 | Linear Model for SBP-Weight Relationship: subjects without Arcus Senilis | 343 |
| 16.10 | Linear Model for SBP-Weight Relationship: subjects with Arcus Senilis | 344 |
| 16.11 | Including Arcus Status in the model | 346 |
| 16.12 | Predictions from these Linear Models | 347 |
| 16.13 | Scatterplots with Facets Across a Categorical Variable | 348 |
| 16.14 | Scatterplot and Correlation Matrices | 348 |
| 16.14.1 | Displaying a Correlation Matrix | 350 |
| 16.14.2 | Using the <code>GGally</code> package | 350 |

| | |
|---|------------|
| II Part B. Comparing Summaries | 352 |
| 17 Part B placeholder | 353 |
| III Part C. Building Models | 354 |
| 18 Part C placeholder | 355 |
| Appendices | 355 |
| A Getting Data Into R | 356 |
| Using data from an R package | 356 |
| Using <code>read_rds</code> to read in an R data set | 356 |
| Using <code>read_csv</code> to read in a comma-separated version of a data file | 357 |
| Converting Character Variables into Factors | 360 |
| Converting Data Frames to Tibbles | 360 |
| For more advice | 361 |
| B References | 362 |

Working with These Notes

1. This document is broken down into multiple chapters. Use the table of contents on the left side of the screen to navigate between chapters, or use the right side to navigate within the current chapter.
2. You can also search the document, using an automated index.
3. Any of the code provided in the document can be copied to the clipboard using the Copy icon at the top right of the code block.
4. The document is not yet complete, although a more complete draft should be in place by the start of class. Newer versions will appear unpredictably throughout the semester.

What You'll Find Here

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

While these Notes share some of the features of a textbook, they are neither comprehensive nor completely original. The main purpose is to give 431 students a set of common materials on which to draw during the course. In class, we will sometimes:

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

but what we don't do is follow these notes very precisely. We assume instead that you will read the materials and try to learn from them, just as you will attend classes and try to learn from them. We welcome feedback of all kinds on this document or anything else.

The 431 Course online

The **online** home for Dr. Love's 431 course in Fall 2022 is

<https://thomaselove.github.io/431-2022/>.

Go there for all information related to the course.

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

By the end of the semester, you will also have access to the Quarto files which generate everything in the document, including all of the R results. [Quarto](#) is a souped-up version of [R Markdown](#), which you will use during the semester to complete your assignments, and which I used to develop previous versions of these notes. We will demonstrate the use of R Markdown and [RStudio](#) (the “program” we use to interface with the R language) in class.

Setting Up R

These Notes make extensive use of

- the statistical [software language R](#), and
- the development environment [RStudio](#),

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

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

R Markdown

These notes were written using [Quarto](#), which is an amplification of R Markdown (which we'll learn in 431.) [R Markdown](#), like R and RStudio and Quarto, 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 [RStudio's introduction to R Markdown](#) which provides 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 Wickham and Grolemund (2022).

R Packages

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

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

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

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

theme_set(theme_bw())
```

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

The Love-boost.R script

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

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

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

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

Packages Used in these Notes

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

| Package | Parts | Key functions in the Package |
|--------------------|---------|--|
| <code>boot</code> | B | — |
| <code>broom</code> | A, B, C | <code>tidy</code> , <code>augment</code> , <code>glance</code> |
| <code>car</code> | A, C | <code>boxCox</code> , <code>powerTransform</code> , <code>testTransform</code> |
| <code>Epi</code> | B | <code>twoby2</code> |

| Package | Parts | Key functions in the Package |
|------------------------|-------------------|---|
| fivethirtyeight | Appendix | source of data |
| GGally | A, C | ggpairs |
| ggrepel | C | — |
| ggridges | A, B | — |
| gt | A | for presenting tables |
| gtsummary | A | tbl_summary |
| Hmisc | A, B, C | describe and others |
| janitor | A, B, C | tabyl and others |
| kableExtra | A | kbl , kable_stylings |
| knitr | A, B, C | kable |
| lvplot | A | geom_lv |
| mice | C | — |
| mosaic | A, B, C | favstats , inspect |
| naniar | A | n_miss , miss_case_table , gg_miss_var |
| NHANES | A | source of data |
| palmerpenguins | A | source of data |
| patchwork | A, B, C | for combining/annotating plots |
| psych | A, B | describe |
| pwr | B | — |
| rms | C | — |
| imputation | A | various imputation functions |
| tidyverse | A, B, C, Appendix | dozens of functions |
| vcd | B | — |
| visdat | A | vis_dat , vis_miss |

The tidyverse

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

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

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

Installing the tidyverse also installs several other useful packages on your machine. Read more about the `tidyverse` at <https://www.tidyverse.org/>

1 Data Science and 431

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

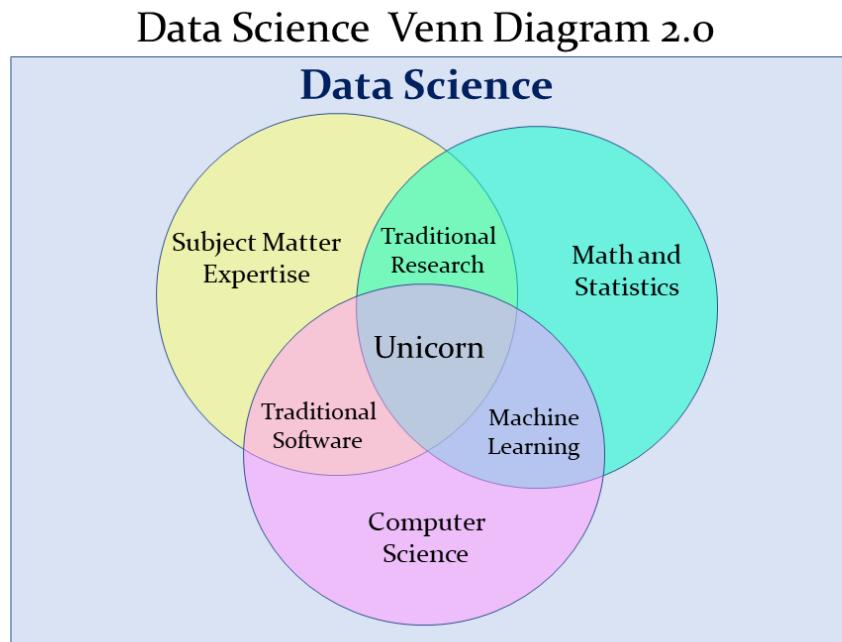


Figure 1.1: Data Science Venn Diagram from Steven Geringer

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

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

[The individual who is truly expert in all three key areas (mathematics/statistics, computer science and subject-matter knowledge) is] a mythical beast with magical powers who's rumored to exist but is never actually seen in the wild.

<http://www.kdnuggets.com/2016/10/battle-data-science-venn-diagrams.html>

1.1 Data Science Project Cycle

A typical data science project can be modeled as follows, which comes from the introduction to the amazing book **R for Data Science**, by Garrett Grolemund and Hadley Wickham, which is a key text for this course (Wickham and Grolemund 2022).

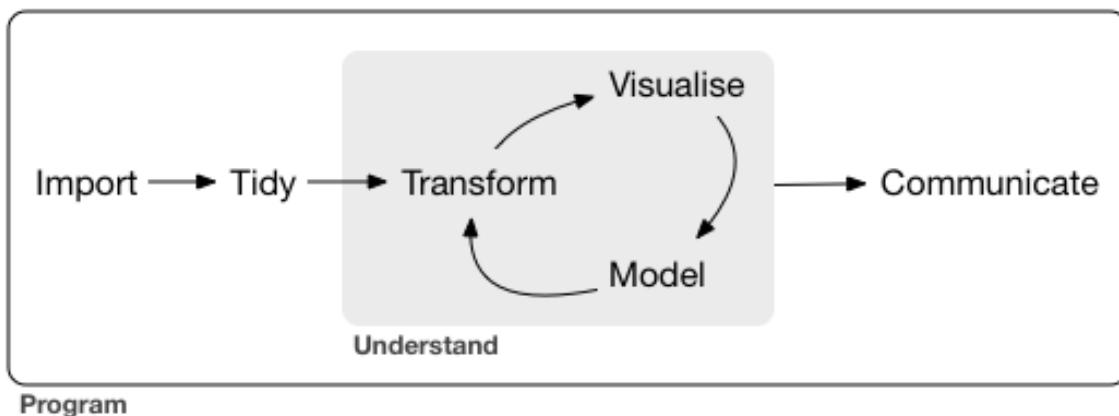


Figure 1.2: Source: R for Data Science: Introduction

This diagram is sometimes referred to as the Krebs Cycle of Data Science. For more on the steps of a data science project, we encourage you to read the Introduction of Wickham and Grolemund (2022).

1.2 Data Science and the 431 Course

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

- We get people working with R and R Studio and R Markdown, even if they are completely new to coding. A gentle introduction is provided at Ismay and Kim (2022)

- We learn how to use the **tidyverse** (<http://www.tidyverse.org/>), an array of tools in R (mostly developed by Hadley Wickham and his colleagues at R Studio) which share an underlying philosophy to make data science faster, easier, more reproducible and more fun. A critical text for understanding the tidyverse is Wickham and Grolemund (2022). Tidyverse tools facilitate:
 - **importing** data into R, which can be the source of intense pain for some things, but is really quite easy 95% of the time with the right tool.
 - **tidying** data, that is, storing it in a format that includes one row per observation and one column per variable. This is harder, and more important, than you might think.
 - **transforming** data, perhaps by identifying specific subgroups of interest, creating new variables based on existing ones, or calculating summaries.
 - **visualizing** data to generate actual knowledge and identify questions about the data - this is an area where R really shines, and we'll start with it in class.
 - **modeling** data, taking the approach that modeling is complementary to visualization, and allows us to answer questions that visualization helps us identify.
 - and last, but definitely not least, **communicating** results, models and visualizations to others, in a way that is reproducible and effective.
- Some programming/coding is an inevitable requirement to accomplish all of these aims. If you are leery of coding, you'll need to get past that, with the help of this course and our stellar teaching assistants. Getting started is always the most challenging part, but our experience is that most of the pain of developing these new skills evaporates by early October.

1.3 What The Course Is and Isn't

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

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

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

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

There's very little of the mathematical underpinnings here:

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

Instead, these notes (and the course) focus on how we get R to do the things we want to do, and how we interpret the results of our work. Our next Chapter provides a first example.

Part I

Part A. Exploring Data

2 The Palmer Penguins

The data in the `palmerpenguins` package in R includes information on several measurements of interest for adult foraging penguins observed on islands in the Palmer Archipelago near Palmer Station, Antarctica. Dr. Kristen Gorman and the Palmer Station Long Term Ecological Research (LTER) Program collected the data and made it available¹. The data describe three species of penguins, called Adelie, Chinstrap and Gentoo.

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

2.1 Setup: Packages Used Here

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

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

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

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

theme_set(theme_bw())
```

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

2.2 Viewing a Data Set

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

```
penguins

# A tibble: 344 x 8
  species island   bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex   year
  <fct>   <fct>        <dbl>        <dbl>          <dbl>        <dbl> <fct> <int>
1 Adelie  Torgersen     39.1         18.7           181      3750 male   2007
2 Adelie  Torgersen     39.5         17.4           186      3800 female 2007
3 Adelie  Torgersen     40.3         18              195      3250 female 2007
4 Adelie  Torgersen     NA            NA             NA       NA <NA>  2007
5 Adelie  Torgersen     36.7         19.3           193      3450 female 2007
6 Adelie  Torgersen     39.3         20.6           190      3650 male   2007
7 Adelie  Torgersen     38.9         17.8           181      3625 female 2007
8 Adelie  Torgersen     39.2         19.6           195      4675 male   2007
9 Adelie  Torgersen     34.1         18.1           193      3475 <NA>  2007
10 Adelie Torgersen      42            20.2          190      4250 <NA>  2007
# ... with 334 more rows, and abbreviated variable names
#   1: flipper_length_mm,
#   2: body_mass_g
# i Use `print(n = ...)` to see more rows
```

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

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

2.3 Create newpenguins: Eliminating Missing Data

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

Note that the code below:

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

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

new_penguins

# A tibble: 333 x 8
  species island   bill_length_mm bill_depth_mm flipper_~1 body_~2 sex     year
  <fct>   <fct>        <dbl>        <dbl>      <int>    <int> <fct> <int>
1 Adelie  Torgersen     39.1       18.7       181     3750 male   2007
2 Adelie  Torgersen     39.5       17.4       186     3800 fema~  2007
3 Adelie  Torgersen     40.3        18         195     3250 fema~  2007
4 Adelie  Torgersen     36.7       19.3       193     3450 fema~  2007
5 Adelie  Torgersen     39.3       20.6       190     3650 male   2007
6 Adelie  Torgersen     38.9       17.8       181     3625 fema~  2007
7 Adelie  Torgersen     39.2       19.6       195     4675 male   2007
8 Adelie  Torgersen     41.1       17.6       182     3200 fema~  2007
9 Adelie  Torgersen     38.6       21.2       191     3800 male   2007
10 Adelie  Torgersen     34.6       21.1       198     4400 male   2007
# ... with 323 more rows, and abbreviated variable names 1: flipper_length_mm,
#   2: body_mass_g
# i Use `print(n = ...)` to see more rows
```

2.4 Counting Things and Making Tables

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

```
nrow(new_penguins)
```

```
[1] 333
```

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

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

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

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

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

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

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

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

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

)

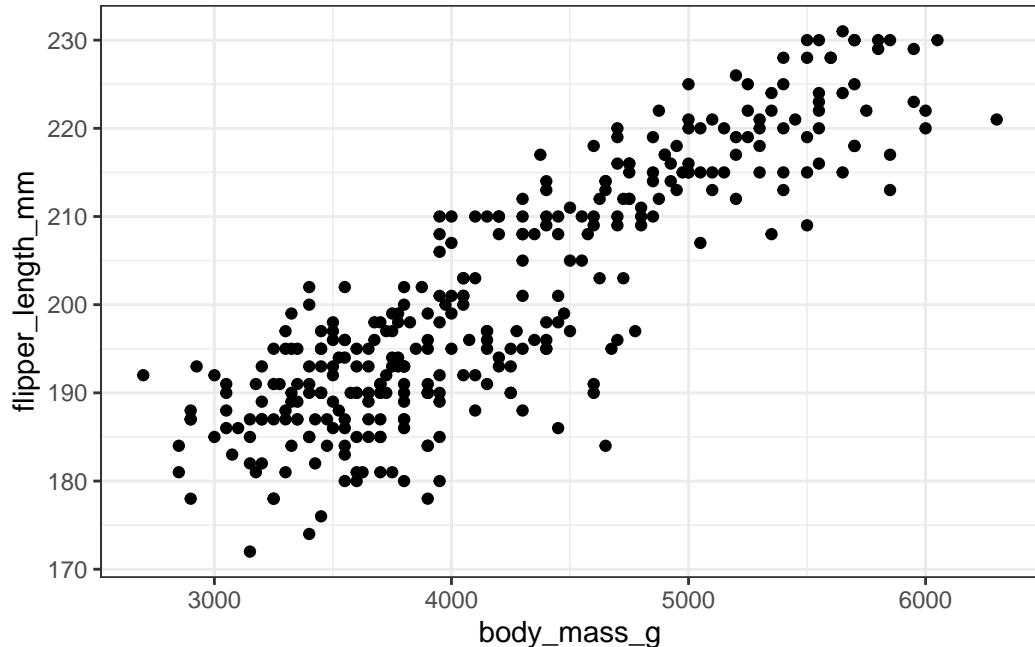
Palmer Penguins in `newpenguins` Comparing sexes by species

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

2.5 Creating a Scatterplot

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

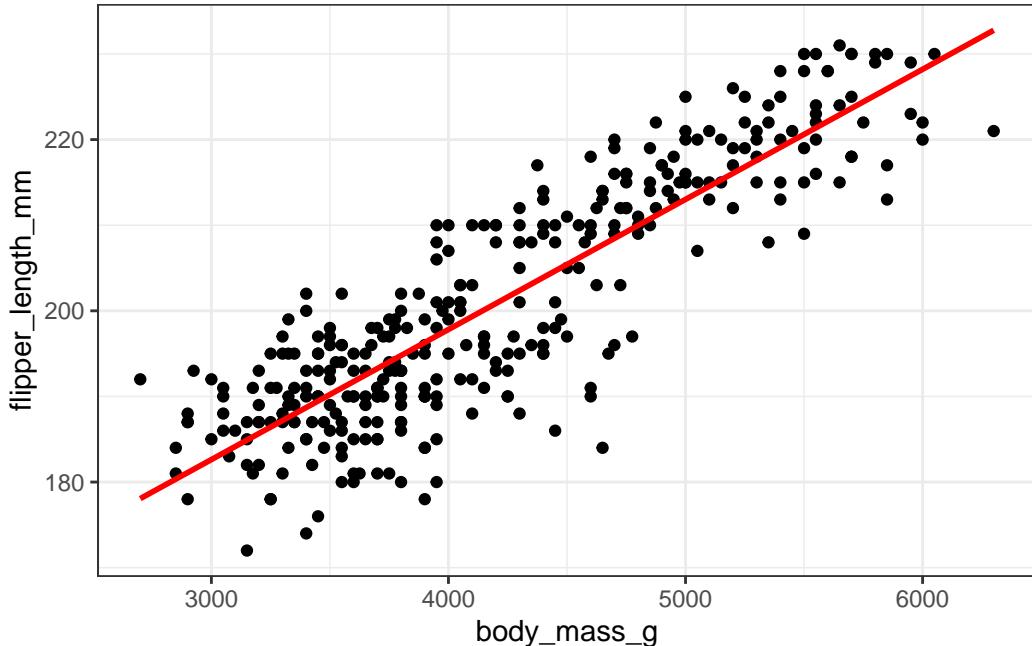
```
ggplot(new_penguins, aes(x = body_mass_g, y = flipper_length_mm)) +  
  geom_point()
```



Some of you may want to include a straight-line model (fit by a classical linear regression) to

this plot. One way to do that in R involves the addition of a single line of code, like this:

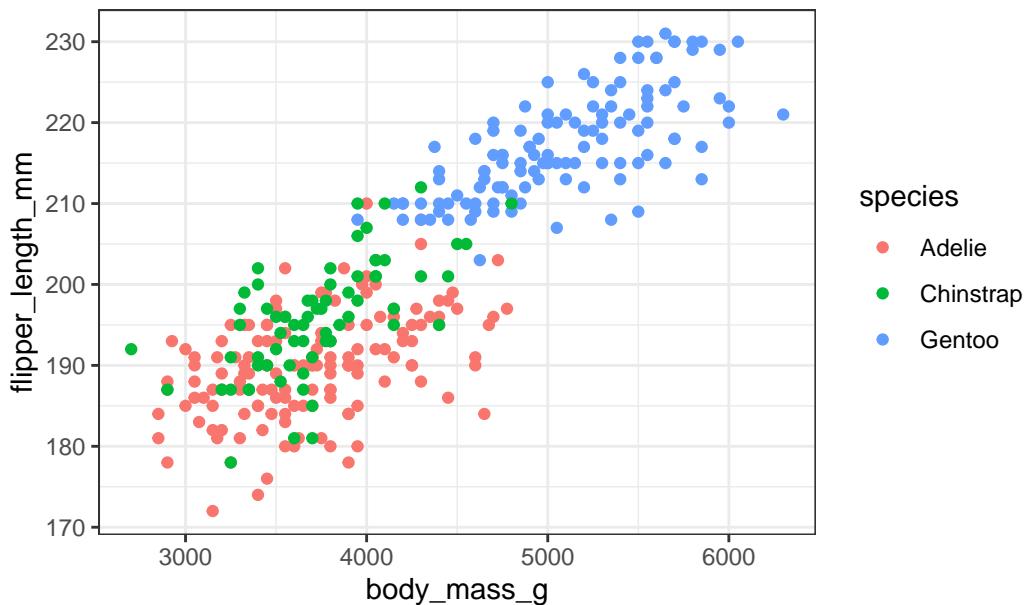
```
ggplot(new_penguins, aes(x = body_mass_g, y = flipper_length_mm)) +  
  geom_point() +  
  geom_smooth(method = "lm", formula = y ~ x,  
              col = "red", se = FALSE)
```



Whenever we build a graph for ourselves, these default choices may be sufficient. But I'd like to see a prettier version if I was going to show it to someone else. So, I might use a different color for each species, and I might add a title, like this.

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

Flipper Length and Body Mass for 333 of the Palmer Penguins



2.6 Six Ways To “Improve” This Graph

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

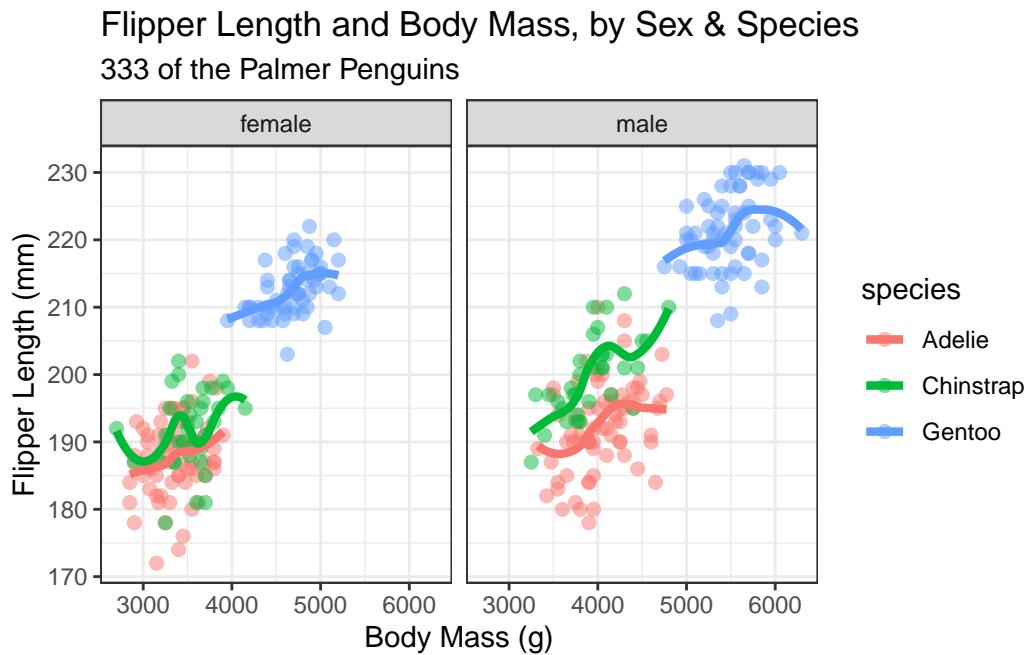
1. plot the relationship between body mass and flipper length in light of both Sex and Species
2. increase the size of the points and add a little transparency so we can see if points overlap,
3. add some smooth curves to summarize the relationships between the two quantities (body mass and flipper length)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

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

```

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

```



2.7 A Little Reflection

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

- What do these plots suggest about the center of the distribution of each quantity (body mass and flipper length) overall, and within each combination of Sex and Species?
- What does the final plot suggest about the spread of the distribution of each of those quantities in each combination of Sex and Species?
- What do the plots suggest about the association of body mass and flipper length across the complete set of penguins?
- How does the shape and nature of this body mass - flipper length relationship change based on Sex and Species?
- Do you think it would be helpful to plot a straight-line relationship (rather than a smooth curve) within each combination of Sex and Species in the final plot? Why or why not? (Also, what would we have to do to the code to accomplish this?)

- How was the R code for the plot revised to accomplish each of the six “wants” specified above?

3 Summarizing Penguins

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

3.1 Setup: Packages Used Here

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

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

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

theme_set(theme_bw())
```

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

3.2 Our Data Set

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

```
penguins
```

```

# A tibble: 344 x 8
  species island   bill_length_mm bill_depth_mm flipper_~1 body_~2 sex     year
  <fct>   <fct>           <dbl>        <dbl>      <int>      <int> <fct> <int>
1 Adelie  Torgersen       39.1         18.7       181      3750 male    2007
2 Adelie  Torgersen       39.5         17.4       186      3800 fema~  2007
3 Adelie  Torgersen       40.3          18        195      3250 fema~  2007
4 Adelie  Torgersen        NA          NA          NA       NA <NA>  2007
5 Adelie  Torgersen       36.7         19.3       193      3450 fema~  2007
6 Adelie  Torgersen       39.3         20.6       190      3650 male   2007
7 Adelie  Torgersen       38.9         17.8       181      3625 fema~  2007
8 Adelie  Torgersen       39.2         19.6       195      4675 male   2007
9 Adelie  Torgersen       34.1         18.1       193      3475 <NA>  2007
10 Adelie Torgersen        42          20.2       190      4250 <NA>  2007
# ... with 334 more rows, and abbreviated variable names 1: flipper_length_mm,
#   2: body_mass_g
# i Use `print(n = ...)` to see more rows

```

3.3 Numerical Summaries for a Tibble

3.3.1 Using `summary()`

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

```

penguins |>
  summary()

```

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

```

Max.    :231.0      Max.    :6300      Max.    :2009
NA's     :2          NA's     :2

```

3.3.2 Using inspect() from mosaic

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

```

penguins |>
  mosaic::inspect()

```

categorical variables:

| | name | class | levels | n | missing | | distribution |
|---|---------|--------|--------|-----|---------|--|--------------|
| 1 | species | factor | 3 | 344 | 0 | | |
| 2 | island | factor | 3 | 344 | 0 | | |
| 3 | sex | factor | 2 | 333 | 11 | | |

1 Adelie (44.2%), Gentoo (36%) ...

2 Biscoe (48.8%), Dream (36%) ...

3 male (50.5%), female (49.5%)

quantitative variables:

| | name | class | min | Q1 | median | Q3 | max | mean |
|---|-------------------|---------|--------|----------|---------|--------|--------|------------|
| 1 | bill_length_mm | numeric | 32.1 | 39.225 | 44.45 | 48.5 | 59.6 | 43.92193 |
| 2 | bill_depth_mm | numeric | 13.1 | 15.600 | 17.30 | 18.7 | 21.5 | 17.15117 |
| 3 | flipper_length_mm | integer | 172.0 | 190.000 | 197.00 | 213.0 | 231.0 | 200.91520 |
| 4 | body_mass_g | integer | 2700.0 | 3550.000 | 4050.00 | 4750.0 | 6300.0 | 4201.75439 |
| 5 | year | integer | 2007.0 | 2007.000 | 2008.00 | 2009.0 | 2009.0 | 2008.02907 |

| | sd | n | missing |
|---|-------------|-----|---------|
| 1 | 5.4595837 | 342 | 2 |
| 2 | 1.9747932 | 342 | 2 |
| 3 | 14.0617137 | 342 | 2 |
| 4 | 801.9545357 | 342 | 2 |
| 5 | 0.8183559 | 344 | 0 |

3.3.3 Using describe() from psych

We can use the `describe()` function from the `psych` package to get some additional summaries, if we're interested, and here we also demonstrate the use of the `kbl()` and `kable_styling()` functions from the `kableExtra` package to make the table look appealing in HTML.

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

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

3.3.4 Using `describe()` from `Hmisc`

One approach Frank Harrell has developed that I find helpful is the `describe()` function within his `Hmisc` package, which produces these results.

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

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

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

  Value       Biscoe     Dream Torgersen
```

| | | | |
|--|---------|----------|-------|
| Frequency | 168 | 124 | 52 |
| Proportion | 0.488 | 0.360 | 0.151 |
| <hr/> | | | |
| bill_length_mm | | | |
| n | missing | distinct | Info |
| 342 | 2 | 164 | 1 |
| .25 | .50 | .75 | .90 |
| 39.23 | 44.45 | 48.50 | 50.80 |
| | | | 51.99 |
| lowest : 32.1 33.1 33.5 34.0 34.1, highest: 55.1 55.8 55.9 58.0 59.6 | | | |
| <hr/> | | | |
| bill_depth_mm | | | |
| n | missing | distinct | Info |
| 342 | 2 | 80 | 1 |
| .25 | .50 | .75 | .90 |
| 15.6 | 17.3 | 18.7 | 19.5 |
| | | | 20.0 |
| lowest : 13.1 13.2 13.3 13.4 13.5, highest: 20.7 20.8 21.1 21.2 21.5 | | | |
| <hr/> | | | |
| flipper_length_mm | | | |
| n | missing | distinct | Info |
| 342 | 2 | 55 | 0.999 |
| .25 | .50 | .75 | .90 |
| 190.0 | 197.0 | 213.0 | 220.9 |
| | | | 225.0 |
| lowest : 172 174 176 178 179, highest: 226 228 229 230 231 | | | |
| <hr/> | | | |
| body_mass_g | | | |
| n | missing | distinct | Info |
| 342 | 2 | 94 | 1 |
| .25 | .50 | .75 | .90 |
| 3550 | 4050 | 4750 | 5400 |
| | | | 5650 |
| lowest : 2700 2850 2900 2925 2975, highest: 5850 5950 6000 6050 6300 | | | |
| <hr/> | | | |
| sex | | | |
| n | missing | distinct | |
| 333 | 11 | 2 | |
| | | | |
| Value | female | male | |
| Frequency | 165 | 168 | |
| Proportion | 0.495 | 0.505 | |
| <hr/> | | | |

| year | n | missing | distinct | Info | Mean | Gmd |
|------------|-------|---------|----------|------|--------|-----|
| 344 | 0 | 3 | 0.888 | 2008 | 0.8919 | |
| <hr/> | | | | | | |
| Value | 2007 | 2008 | 2009 | | | |
| Frequency | 110 | 114 | 120 | | | |
| Proportion | 0.320 | 0.331 | 0.349 | | | |

3.3.5 Using `tbl_summary()` from `gtsummary`

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

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

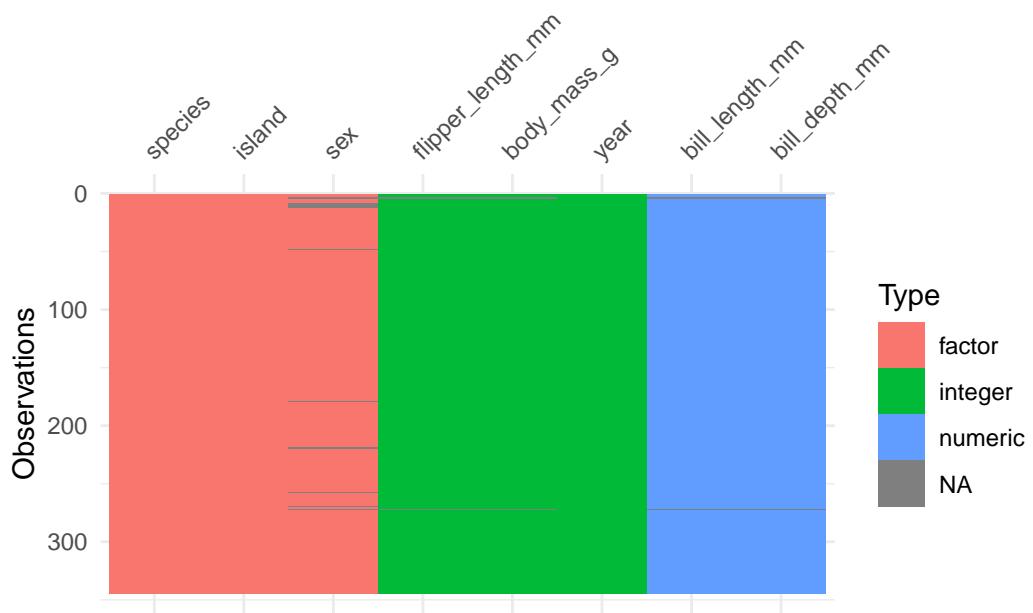
Table printed with `knitr::kable()`, not `{gt}`. Learn why at
<https://www.danielsgjoberg.com/gtsummary/articles/rmarkdown.html>
To suppress this message, include ``message = FALSE`` in code chunk header.

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

3.3.6 Visualizing with visdat functions

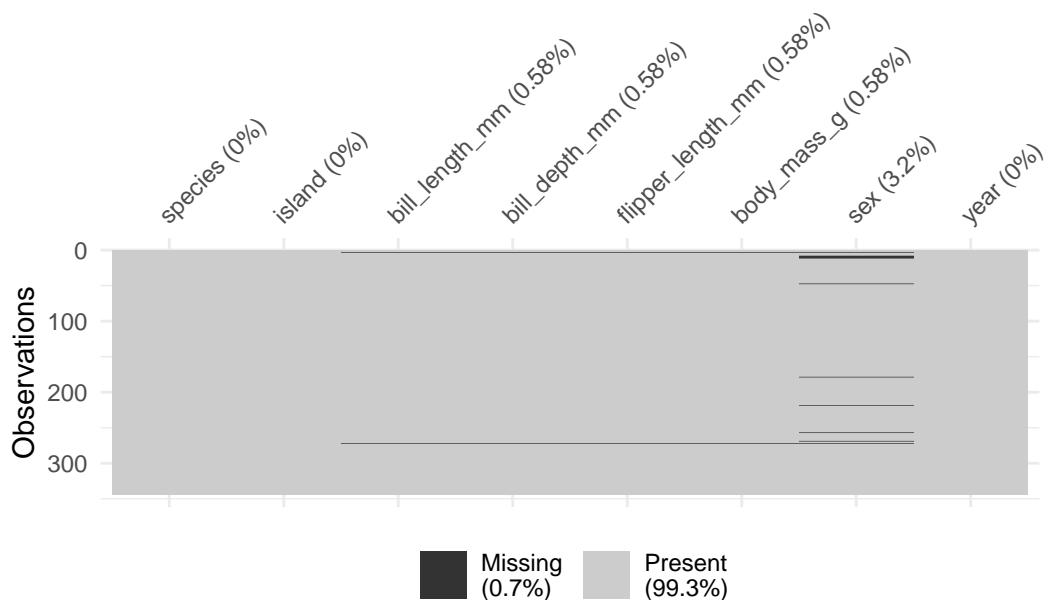
The `vis_dat()` function from the `visdat` package shows something about the types of variables, providing visual clues about what's inside. The picture below identifies variables types, and missing values.

```
vis_dat(penguins)
```



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

```
vis_miss(penguins)
```



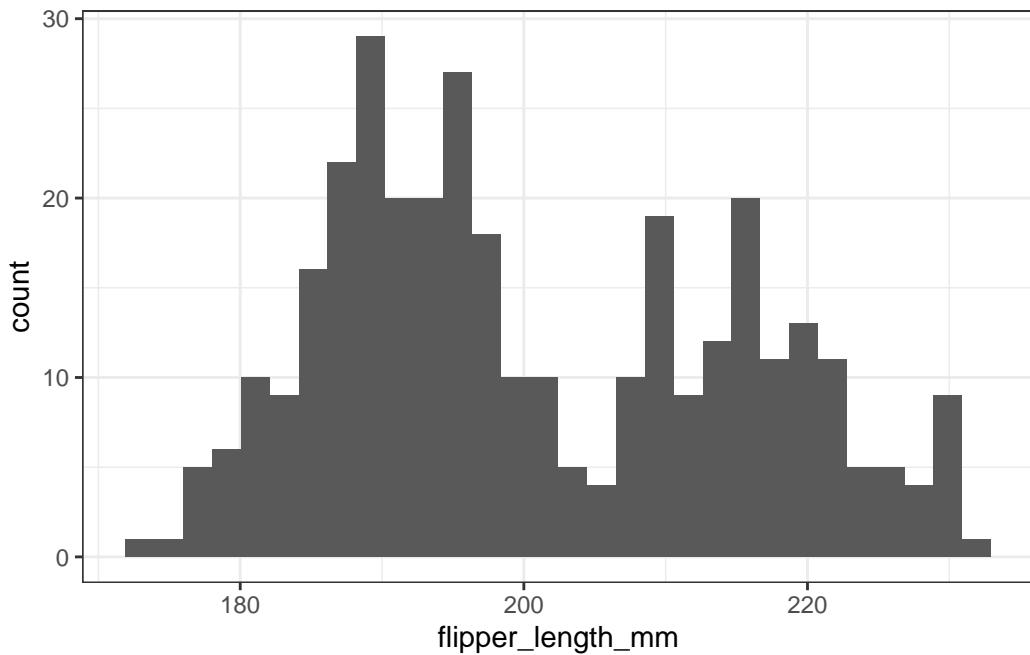
3.4 Histograms for a Variable

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

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

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

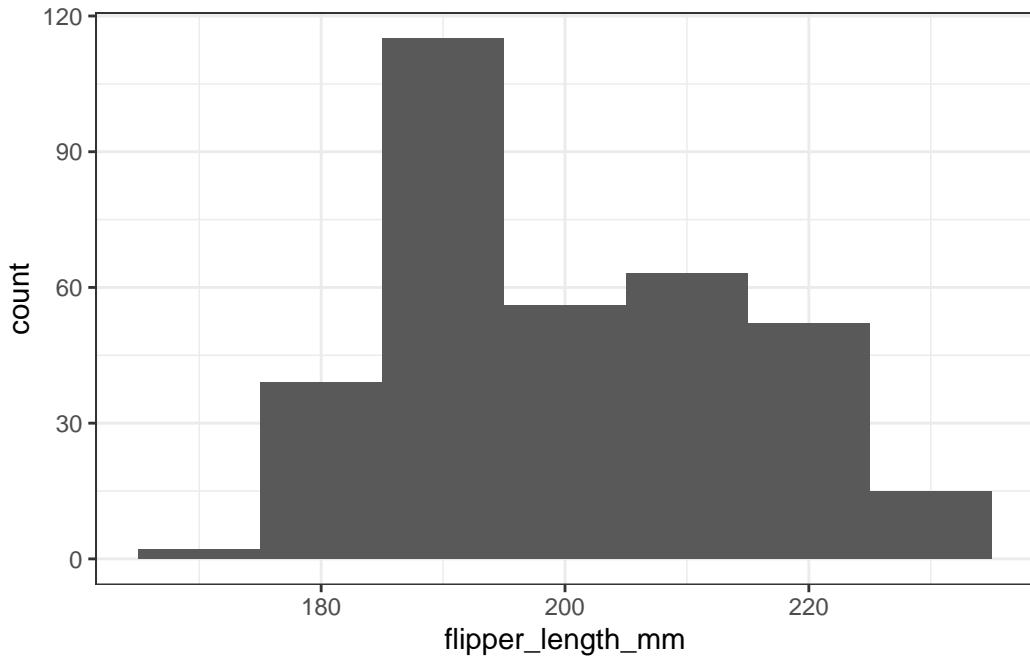
Warning: Removed 2 rows containing non-finite values (stat_bin).
```



This approach produces two messages that alert us to potential concerns, and a fairly unattractive plot.

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

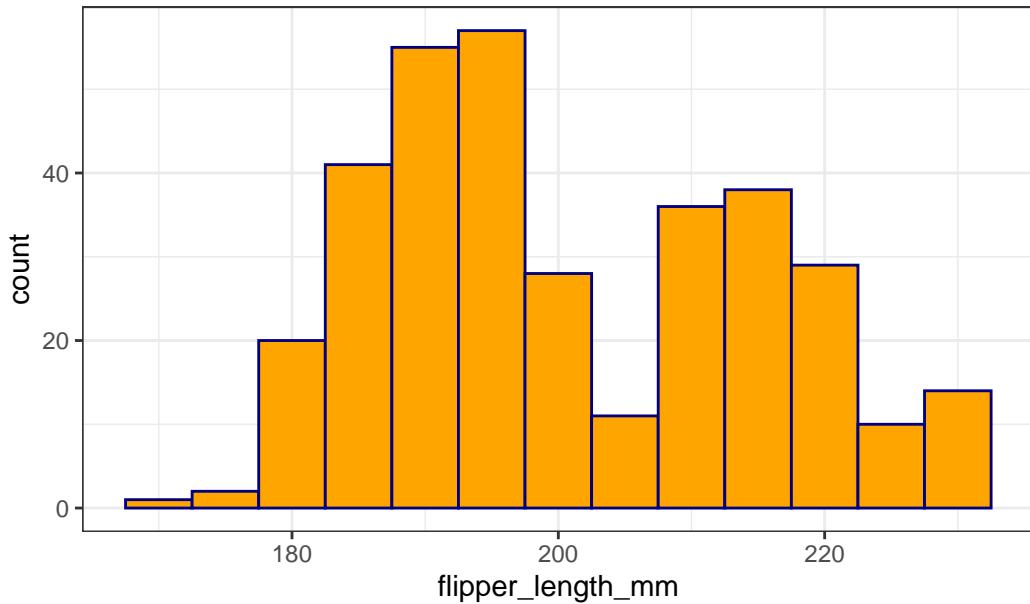
```
penguins2 <-  
  penguins |>  
  filter(complete.cases(flipper_length_mm))  
  
ggplot(data = penguins2, aes(x = flipper_length_mm)) +  
  geom_histogram(binwidth = 10)
```



Now we've eliminated the messages, but it would be nice to have some more granularity in the bars (so we'd like a smaller binwidth) and I'd also like to make the bars more clearly separated with colors. I'd also like to add a title. Like this:

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

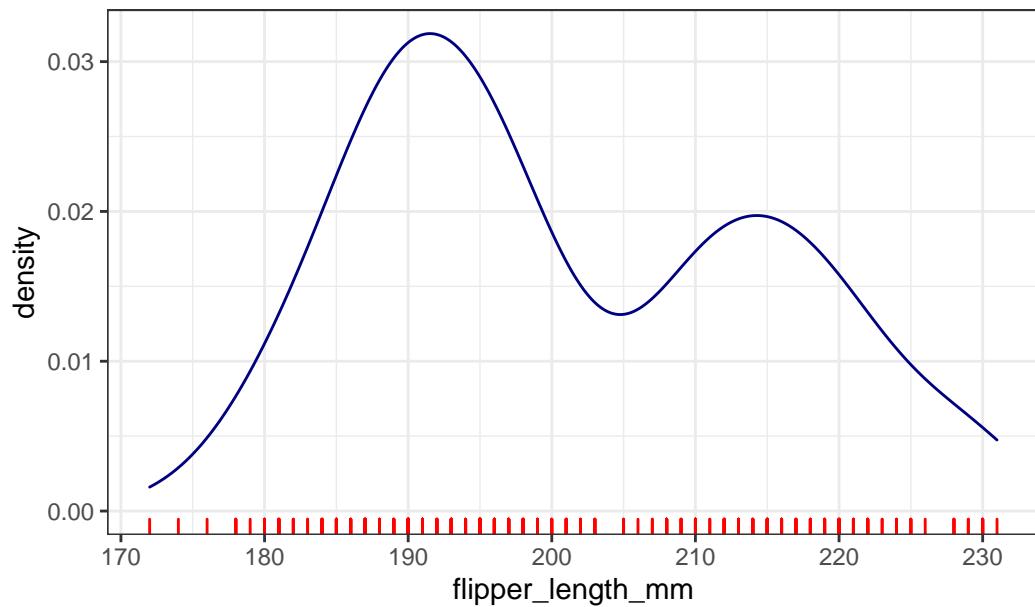
Distribution of Flipper Length in 342 Palmer Penguins



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

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

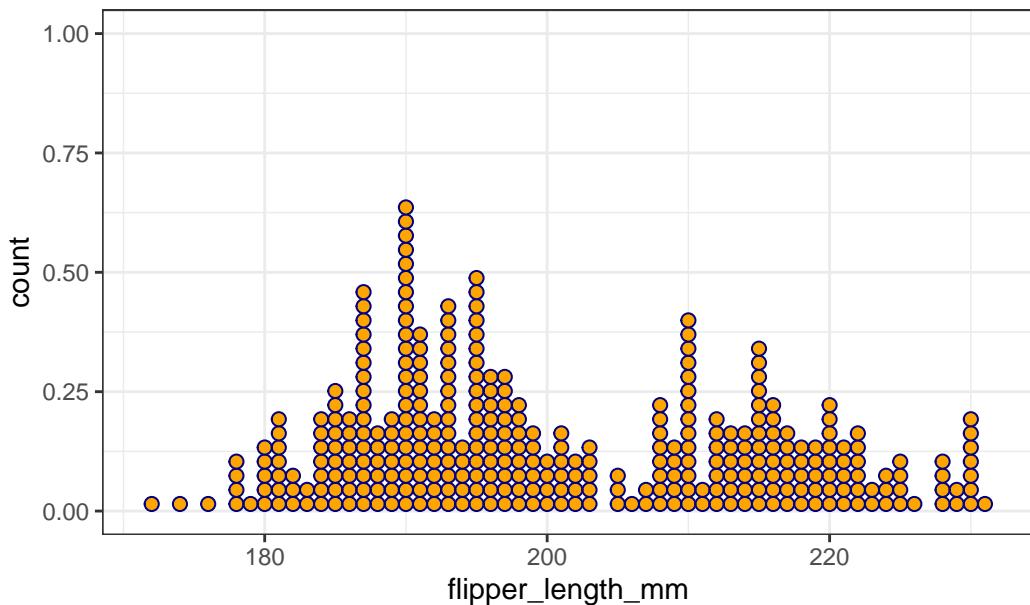
Density and Rug Plot of Flipper Length in 342 Palmer Penguin



Or perhaps a dotplot would provide a useful look...

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

Dot Plot of Flipper Length in 342 Palmer Penguins



We'll learn about several other approaches to summarizing the distribution of a variable graphically later in the course.

3.5 Comparing Penguins by Species Numerically

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

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

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

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

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

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

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

3.5.1 Using `tbl_summary()` to summarize the tibble

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

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

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

| **Characteristic** | **Adelie**, N = 152 | **Chinstrap**, N = 68 | **Gentoo**, N = 124 |
|--------------------|----------------------|-----------------------|----------------------|
| island | | | |
| Biscoe | 44 (29%) | 0 (0%) | 124 (100%) |
| Dream | 56 (37%) | 68 (100%) | 0 (0%) |
| Torgersen | 52 (34%) | 0 (0%) | 0 (0%) |
| bill_length_mm | 38.8 (36.8, 40.8) | 49.5 (46.3, 51.1) | 47.3 (45.3, 49.5) |
| Unknown | 1 | 0 | 1 |
| bill_depth_mm | 18.40 (17.50, 19.00) | 18.45 (17.50, 19.40) | 15.00 (14.20, 15.70) |
| Unknown | 1 | 0 | 1 |
| flipper_length_mm | 190 (186, 195) | 196 (191, 201) | 216 (212, 221) |
| Unknown | 1 | 0 | 1 |
| body_mass_g | 3,700 (3,350, 4,000) | 3,700 (3,488, 3,950) | 5,000 (4,700, 5,500) |
| Unknown | 1 | 0 | 1 |
| sex | | | |
| female | 73 (50%) | 34 (50%) | 58 (49%) |
| male | 73 (50%) | 34 (50%) | 61 (51%) |
| Unknown | 6 | 0 | 5 |
| year | | | |
| 2007 | 50 (33%) | 26 (38%) | 34 (27%) |
| 2008 | 50 (33%) | 18 (26%) | 46 (37%) |
| 2009 | 52 (34%) | 24 (35%) | 44 (35%) |

3.6 Comparing Penguins by Species Graphically

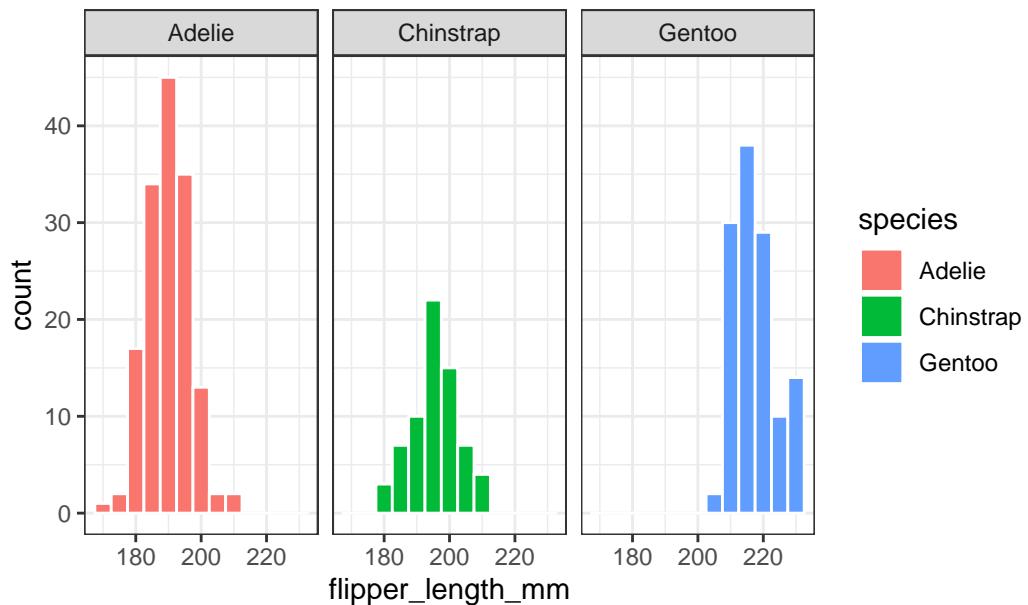
3.6.1 Faceting Histograms

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

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

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

Distribution of Flipper Length in Palmer Penguins, by Species



We might add in the command

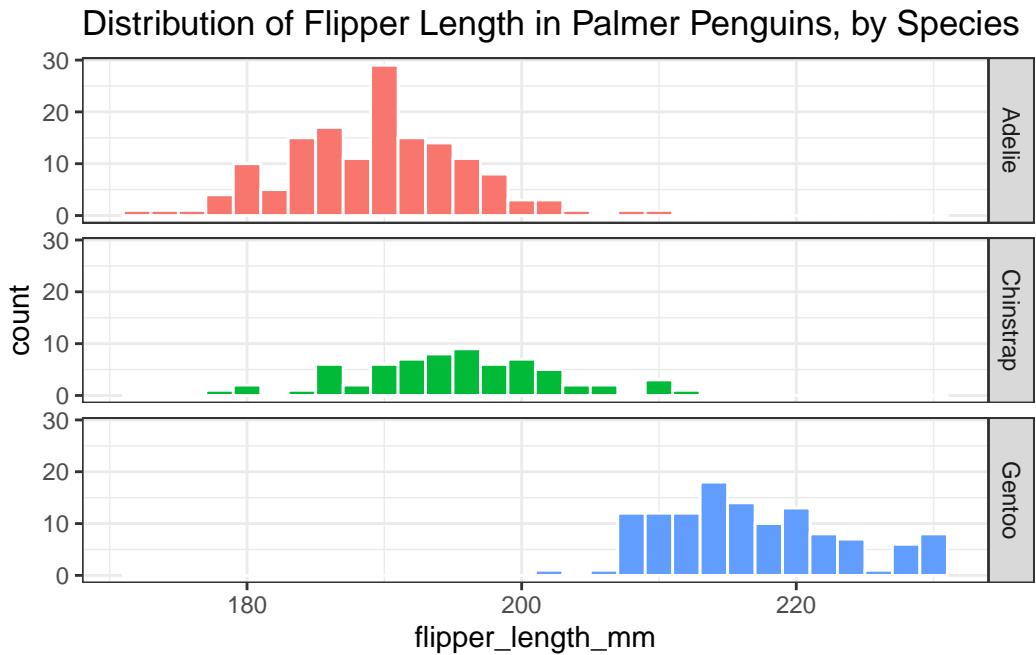
```
guides(fill = "none") +
```

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

3.6.2 Using `facet_grid()`

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

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

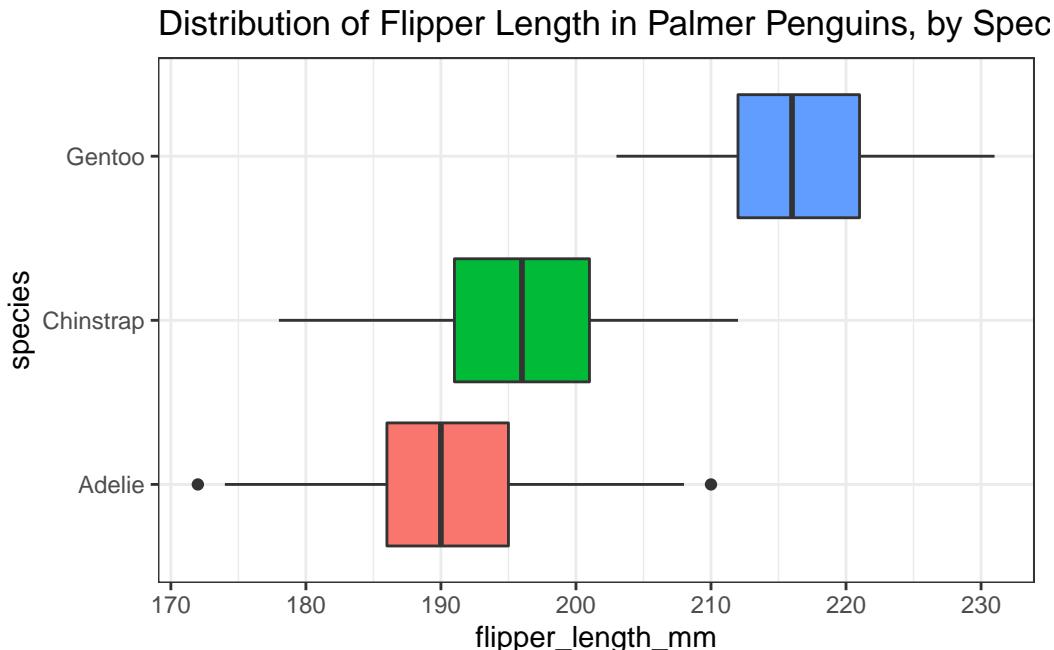


We'll use facets like this all the time in what follows.

3.6.3 Boxplots

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

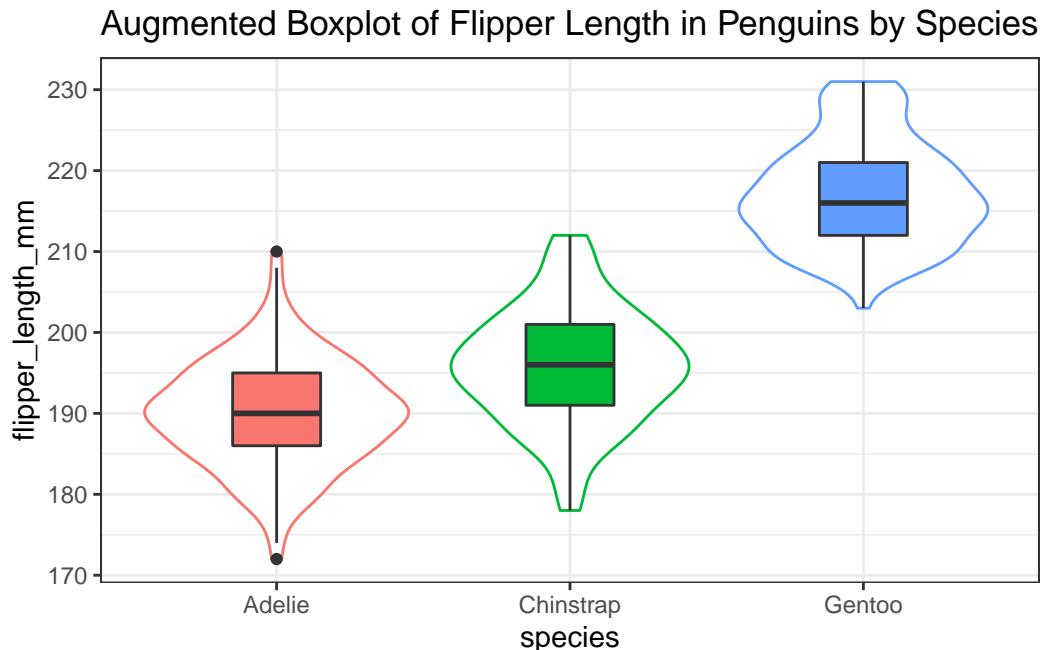
```
ggplot(data = penguins3, aes(x = flipper_length_mm, y = species,
                             fill = species)) +
  geom_boxplot() +
  guides(fill = "none") +
  labs(title = "Distribution of Flipper Length in Palmer Penguins, by Species")
```



3.6.4 Adding Violins

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

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

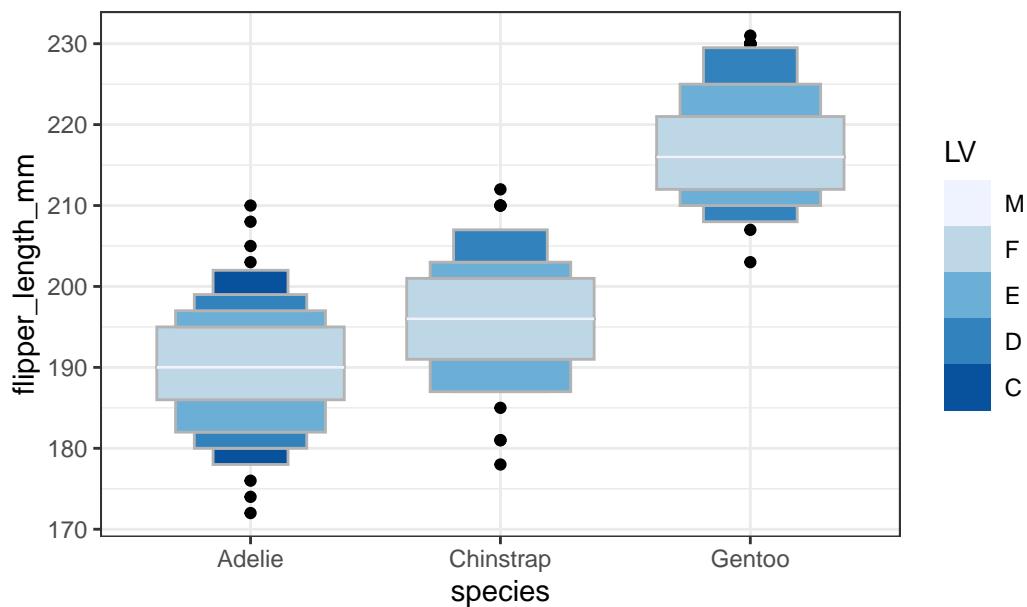


3.6.5 Letter-Value Plots (Boxplots for Large Data)

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

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

Letter–Value Plot of Flipper Length in Penguins by Species



You're probably tiring of the penguins now. Let's look at a larger data set next.

4 NHANES Data

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

4.1 Setup: Packages Used Here

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

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

theme_set(theme_bw())
```

4.2 The NHANES data: A First Sample

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

NHANES

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

```

6 51638 2009_10 male      9 " 0-9"          115 White <NA>  <NA>      <NA>
7 51646 2009_10 male      8 " 0-9"          101 White <NA>  <NA>      <NA>
8 51647 2009_10 female    45 " 40-49"        541 White <NA>  College ~ Married
9 51647 2009_10 female    45 " 40-49"        541 White <NA>  College ~ Married
10 51647 2009_10 female   45 " 40-49"        541 White <NA>  College ~ Married
# ... with 9,990 more rows, 66 more variables: HHIncome <fct>,
#   HHIncomeMid <int>, Poverty <dbl>, HomeRooms <int>, HomeOwn <fct>,
#   Work <fct>, Weight <dbl>, Length <dbl>, HeadCirc <dbl>, Height <dbl>,
#   BMI <dbl>, BMICatUnder20yrs <fct>, BMI_WHO <fct>, Pulse <int>,
#   BPSysAve <int>, BPDiaAve <int>, BPSys1 <int>, BPDia1 <int>, BPSys2 <int>,
#   BPDia2 <int>, BPSys3 <int>, BPDia3 <int>, Testosterone <dbl>,
#   DirectChol <dbl>, TotChol <dbl>, UrineVol1 <int>, UrineFlow1 <dbl>, ...
# i Use `print(n = ...)` to see more rows, and `colnames()` to see all variable names

```

We see that we have 10000 rows and 76 columns in the NHANES tibble.

4.3 Sampling NHANES Adults

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

4.3.1 Creating a Temporary, Cleaner Data Frame

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

1. We'll **filter** the original NHANES data frame to include only the responses from the 2011-12 administration of the survey. This will cut the sample in half, from 10,000 rows to 5,000.
2. We'll then **filter** again to restrict the sample to adults whose age is at least 21 and also less than 65. I'll do this because I want to avoid problems with including both children and adults in my sample, and because I also want to focus on the population of people in the US who are usually covered by private insurance from their job, or by Medicaid insurance from the government, rather than those covered by Medicare.
3. What is listed in the NHANES 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.

4. We'll also rename three other variables, specifically we'll use `Race` to describe the `Race3` variable in the original NHANES data frame, as well as `SBP` to refer to the average systolic blood pressure, which is specified as `BPSysAve`, and `DBP` to refer to the average diastolic blood pressure, which is specified as `BPDiaAve`.
5. Having accomplished the previous four steps, we'll then *select* the variables we want to keep in the sample. (We use `select` for choosing variables or columns in the data frame, and `filter` for selecting subjects or rows.) The sixteen variables we will select are: `ID`, `Sex`, `Age`, `Height`, `Weight`, `Race`, `Education`, `BMI`, `SBP`, `DBP`, `Pulse`, `PhysActive`, `Smoke100`, `SleepTrouble`, `MaritalStatus` and `HealthGen`.
6. The original NHANES data frame includes some subjects (rows) multiple times in an effort to incorporate some of the sampling weights used in most NHANES analyses. For our purposes, though, we'd like to only include each subject one time. We use the `distinct()` function to limit the data frame to completely unique subjects (so that, for example, we don't wind up with two or more rows that have the same `ID` number.)

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

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

The resulting `nh_temp` data frame has 1700 rows and 16 columns.

```
nh_temp
```

| | ID | Sex | Age | Height | Weight | Race | Educa~1 | BMI | SBP | DBP | Pulse | PhysA~2 |
|---|-------|-------|-----|--------|--------|-------|---------|------|-----|-----|-------|---------|
| 1 | 62172 | fema~ | 43 | 172 | 98.6 | Black | High S~ | 33.3 | 103 | 72 | 80 | No |
| 2 | 62176 | fema~ | 34 | 172. | 68.7 | White | Colleg~ | 23.3 | 107 | 69 | 92 | Yes |
| 3 | 62180 | male | 35 | 179. | 89 | White | Colleg~ | 27.9 | 107 | 66 | 66 | No |
| 4 | 62199 | male | 57 | 186 | 96.9 | White | Colleg~ | 28 | 110 | 65 | 84 | Yes |
| 5 | 62205 | male | 28 | 171. | 84.8 | White | Colleg~ | 28.9 | 122 | 87 | 70 | Yes |
| 6 | 62206 | fema~ | 35 | 167. | 81.5 | White | Some C~ | 29.1 | 106 | 50 | 58 | No |
| 7 | 62208 | male | 38 | 169. | 63.2 | Hisp~ | Some C~ | 22.2 | 105 | 59 | 52 | Yes |

```

8 62209 fema~    62    143.   53.5 Mexi~ 8th Gr~  26     108    57    72 No
9 62220 fema~    31    167.   113. Black Colleg~ 40.4   120    71    62 Yes
10 62222 male     32    179    80.1 White Colleg~ 25     104    73    78 No
# ... with 1,690 more rows, 4 more variables: Smoke100 <fct>,
# SleepTrouble <fct>, MaritalStatus <fct>, HealthGen <fct>, and abbreviated
# variable names 1: Education, 2: PhysActive
# i Use `print(n = ...)` to see more rows, and `colnames()` to see all variable names

```

4.3.2 Sampling nh_temp to obtain our nh_adult750 sample

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

- We will use the `set.seed()` function in R to set a random numerical seed to ensure that if you redo this work, you will obtain the same sample.
 - Setting a seed is an important part of being able to replicate the work later when sampling is involved.
- Then we will use the `slice_sample()` function to actually draw the random sample, without replacement.
 - “Without replacement” means that once we’ve selected a particular subject, we won’t select them again.

```

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

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

nh_adult750

# A tibble: 750 x 16
   ID Sex     Age Height Weight Race Educa~1 BMI   SBP   DBP Pulse PhysA~2
   <int> <fct> <int>  <dbl>  <dbl> <fct> <fct>  <dbl> <int> <int> <int> <fct>
1 68648 fema~    30    181.   67.1 White Colleg~ 20.4   103    59    78 No
2 67200 male     30    180.   86.6 White Colleg~ 26.7   113    68    70 Yes
3 66404 fema~    35    160.   71.1 White Colleg~ 27.8   116    80    68 Yes
4 70535 male     40    177.   82    White Colleg~ 26.3   130    79    68 No
5 65308 fema~    54    151.   60.6 Mexi~ 8th Gr~ 26.6   130    64    48 No
6 67392 male     41    171.   90.7 Hisp~ Colleg~ 31.2   124    82    68 Yes
7 63218 male     35    163.   81    Mexi~ 8th Gr~ 30.3   128    96    82 No
8 65879 fema~    32    160.   66.4 Mexi~ Colleg~ 25.9   104    70    78 Yes

```

```

 9 63617 male      29    189.   83.3 White Colleg~ 23.2   105    72    76 Yes
10 64720 male      29    174.   62.3 Black Colleg~ 20.6   127    60    84 Yes
# ... with 740 more rows, 4 more variables: Smoke100 <fct>, SleepTrouble <fct>,
#   MaritalStatus <fct>, HealthGen <fct>, and abbreviated variable names
#   1: Education, 2: PhysActive
# i Use `print(n = ...)` to see more rows, and `colnames()` to see all variable names

```

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

4.3.3 Summarizing the Data's Structure

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

```
dim(nh_adult750)
```

```
[1] 750 16
```

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

```
str(nh_adult750)
```

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

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

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

# A tibble: 5 x 16
# ... with 4 more variables: Smoke100 <fct>, SleepTrouble <fct>,
#   MaritalStatus <fct>, HealthGen <fct>, and abbreviated variable names
#   1: Education, 2: PhysActive
# i Use `colnames()` to see all variable names
```

4.3.4 What are the variables?

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

```
glimpse(nh_adult750)

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

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

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

The levels for the multi-categorical variables are:

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

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

```
summary(nh_adult750)
```

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

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

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

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

4.4 Counting Missing Values

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

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

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

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

```
miss_var_table(nh_adult750)

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

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

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

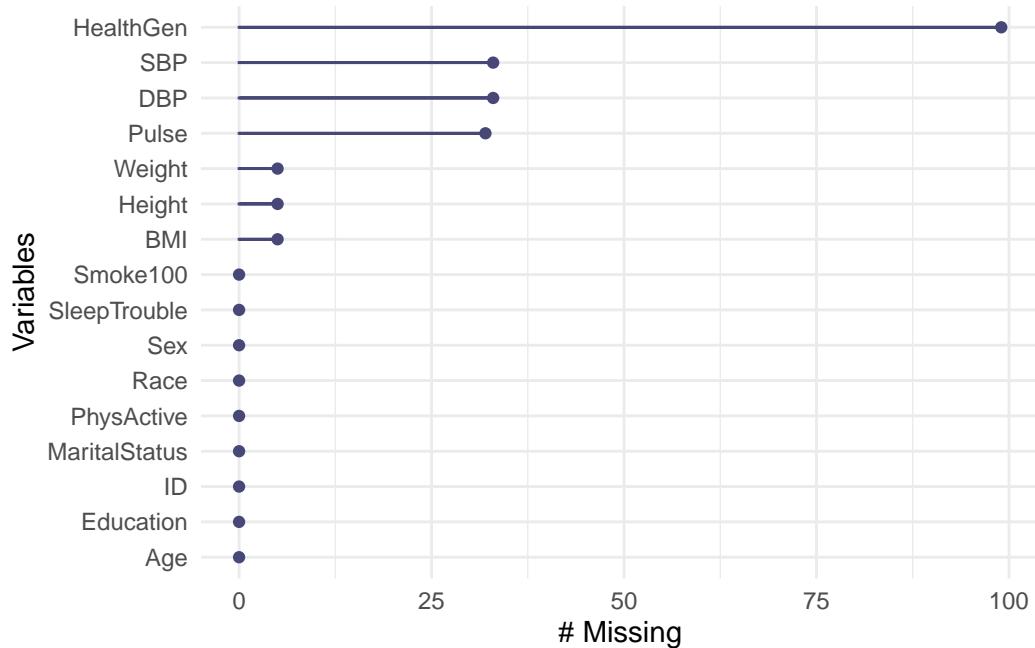
```
miss_var_summary(nh_adult750) |>
  kbl() |>
  kable_styling(full_width = FALSE, position = "center")
```

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

A graph of this information is available, as well.

```
gg_miss_var(nh_adult750)
```

| variable | n_miss | pct_miss |
|---------------|--------|------------|
| HealthGen | 99 | 13.2000000 |
| SBP | 33 | 4.4000000 |
| DBP | 33 | 4.4000000 |
| Pulse | 32 | 4.2666667 |
| Height | 5 | 0.6666667 |
| Weight | 5 | 0.6666667 |
| BMI | 5 | 0.6666667 |
| ID | 0 | 0.0000000 |
| Sex | 0 | 0.0000000 |
| Age | 0 | 0.0000000 |
| Race | 0 | 0.0000000 |
| Education | 0 | 0.0000000 |
| PhysActive | 0 | 0.0000000 |
| Smoke100 | 0 | 0.0000000 |
| SleepTrouble | 0 | 0.0000000 |
| MaritalStatus | 0 | 0.0000000 |



I'll note that there are also functions to count the number of missing observations by case (observation) rather than variable. For example, we can use `miss_case_table`.

```
miss_case_table(nh_adult750)

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

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

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

```
miss_case_summary(nh_adult750)

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

4.5 Sampling 500 Complete Cases

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

```

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

```

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

```
miss_var_table(nh_temp2)
```

| | n_miss_in_var | n_vars | pct_vars |
|-------|---------------|--------|----------|
| <int> | <int> | <dbl> | |
| 1 | 0 | 16 | 100 |

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

```

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

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

nh_adult500cc

# A tibble: 500 x 16
   ID Sex     Age Height Weight Race Educa~1   BMI    SBP    DBP Pulse PhysA~2
   <int> <fct> <int>  <dbl>  <dbl> <fct> <fct>   <dbl> <int> <int> <fct>
1 64079 fema~    25    159.   86.2 Hisp~ Some C~  34.2   120    67    84 Yes
2 64374 fema~    52    169    65.5 Asian Colleg~ 22.9    92    58    60 Yes
3 71875 male    42    182.   94.1 Black Colleg~ 28.5   102    63    76 Yes
4 66396 fema~    46    161.   107.  Asian 8th Gr~ 41.2   111    61    70 No
5 64315 fema~    52    161.   64.5 White 9 - 11~ 24.9   130    69    68 Yes
6 64015 male    32    168.   82.3 Mexi~ Some C~  29     119    79    70 No
7 63590 male    21    181.   98.3 Black Some C~ 29.9   121    67    58 Yes
8 70893 fema~    30    171.   65.7 White 9 - 11~ 22.5   104    75    74 Yes

```

```
9 70828 male      26   178.  100. White Some C~  31.5   119     77     66 No
10 67930 male     59   172.  91.7 Mexi~ Colleg~  31     127     85     66 No
# ... with 490 more rows, 4 more variables: Smoke100 <fct>, SleepTrouble <fct>,
#   MaritalStatus <fct>, HealthGen <fct>, and abbreviated variable names
#   1: Education, 2: PhysActive
# i Use `print(n = ...)` to see more rows, and `colnames()` to see all variable names
```

4.6 Saving our Samples in .Rds files

We'll save the `nh_adult750` and `nh_adult500cc` samples to use in later parts of the notes. To do this, we'll save them as `.Rds` files, which will have some advantages for us later on.

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

You will also find these `.Rds` files as part of the [431-data repository](#) for the course.

4.7 Coming Up

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

5 Types of Data

5.1 Setup: Packages Used Here

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

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

theme_set(theme_bw())
```

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

5.2 Data require structure and context

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

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

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

In general, each row of a data frame corresponds to an individual (respondent, experimental unit, record, or observation) about whom some characteristics are gathered in columns (and these characteristics may be called variables, factors or data elements.) Every column / variable

should have a name that indicates *what* it is measuring, and every row / observation should have a name that indicates *who* is being measured.

5.3 Loading in the “Complete Cases” Sample

Let’s begin by loading into the `nh_500cc` data frame the information from the `nh_adult500cc.Rds` file we created in Section 4.5..

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

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

5.4 Quantitative Variables

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

- It does little good to be told the price of something if you don’t know the currency being used.
- You might be surprised to see someone whose age is 72 listed in a database on childhood diseases until you find out that age is measured in months.
- Often just seeking the units can reveal a variable whose definition is challenging - just how do we measure “friendliness”, or “success,” for example.
- Quantitative variables may also be classified by whether they are **continuous** or can only take on a **discrete** set of values. Continuous data may take on any value, within a defined range. Suppose we are measuring height. While height is really continuous, our measuring stick usually only lets us measure with a certain degree of precision. If our measurements are only trustworthy to the nearest centimeter with the ruler we have, we might describe them as discrete measures. But we could always get a more precise ruler. The measurement divisions we make in moving from a continuous concept to a discrete measurement are usually fairly arbitrary. Another way to think of this, if you enjoy music, is that, as suggested in Norman and Streiner (2014), a piano is a *discrete*

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

instrument, but a violin is a *continuous* one, enabling finer distinctions between notes than the piano is capable of making. Sometimes the distinction between continuous and discrete is important, but usually, it's not.

5.5 Quantitative Variables in nh_500cc

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

```
names(nh_500cc)

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

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

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

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

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

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

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

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

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

| **Characteristic** | **N = 500** |
|--------------------|---------------------------------------|
| Age | 42 (13): [21, 30, 42, 52, 64] |
| Height | 169 (10): [145, 162, 169, 176, 200] |
| Weight | 83 (21): [42, 68, 80, 94, 184] |
| BMI | 29 (7): [17, 24, 28, 32, 63] |
| SBP | 119 (15): [84, 110, 119, 127, 221] |
| DBP | 72 (11): [0, 66, 72, 79, 110] |
| Pulse | 74 (12): [48, 66, 74, 82, 114] |

5.5.1 A look at BMI (Body-Mass Index)

The definition of BMI (*body-mass index*) for adult subjects (which is expressed in units of kg/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 follows: a BMI from 18.5 up to 25 may indicate optimal weight, a BMI lower than 18.5 suggests the person is underweight, a number from 25 up to 30 may indicate the person is overweight, and a number from 30 upwards suggests the person is obese.

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

5.5.2 Types of Quantitative Variables

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

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

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

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

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

5.6 Qualitative (Categorical) Variables

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

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

```

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

```

| | Sex | PhysActive | Smoke100 | SleepTrouble | Race |
|----------------|-----------|--------------|---------------|---------------|------|
| female:236 | No :216 | No :291 | No :380 | Asian : 51 | |
| male :264 | Yes:284 | Yes:209 | Yes:120 | Black : 81 | |
| | | | | Hispanic: 37 | |
| | | | | Mexican : 48 | |
| | | | | White :262 | |
| | | | | Other : 21 | |
| | Education | | MaritalStatus | HealthGen | |
| 8th Grade | : 26 | Divorced | : 47 | Excellent: 52 | |
| 9 - 11th Grade | : 59 | LivePartner | : 46 | Vgood :167 | |
| High School | : 89 | Married | :256 | Good :204 | |
| Some College | :153 | NeverMarried | :125 | Fair : 65 | |
| College Grad | :173 | Separated | : 17 | Poor : 12 | |
| | | Widowed | : 9 | | |

5.6.1 Nominal vs. Ordinal Categories

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

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

- The alternative categorical variable (where order matters) is called **ordinal**, and includes variables that are sometimes thought of as falling right in between quantitative and qualitative variables.

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

- Answers to questions like “How is your overall physical health?” with available responses Excellent, Very Good, Good, Fair or Poor, which are often coded as 1-5, certainly provide a perceived *order*, but a group of people with average health status 4 (Very Good) is not necessarily twice as healthy as a group with average health status of 2 (Fair).

- Sometimes we treat the values from ordinal variables as sufficiently scaled to permit us to use quantitative approaches like means, quantiles, and standard deviations to summarize and model the results, and at other times, we'll treat ordinal variables as if they were nominal, with tables and percentages our primary tools.
- Note that all binary variables may be treated as either ordinal, or nominal.

Binary variables in the `nh_500cc` data include `Sex`, `PhysActive`, `Smoke100`, `SleepTrouble`. Each can be thought of as either ordinal or nominal.

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

5.7 Tabulating Binary Variables

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

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

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

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

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

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

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

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

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

| | | Smoke100 | |
|------------|-----|----------|-----|
| | | No | Yes |
| PhysActive | No | 111 | 105 |
| | Yes | 180 | 104 |

5.8 Tabulating Multi-Categorical Variables

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

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

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

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

```
nh_500cc |>
  tabyl(Education, HealthGen) |>
  adorn_totals(where = c("row", "col")) |>
  adorn_title()
```

| | | HealthGen | | | | | | |
|----------------|--------------|-----------|-----------|-------|------|------|------|-------|
| | | Education | Excellent | Vgood | Good | Fair | Poor | Total |
| | 8th Grade | | 2 | 3 | 9 | 9 | 3 | 26 |
| 9 - 11th Grade | | | 4 | 12 | 28 | 12 | 3 | 59 |
| | High School | | 9 | 26 | 36 | 18 | 0 | 89 |
| | Some College | | 11 | 46 | 75 | 17 | 4 | 153 |

| | | | | | | |
|--------------|----|-----|-----|----|----|-----|
| College Grad | 26 | 80 | 56 | 9 | 2 | 173 |
| Total | 52 | 167 | 204 | 65 | 12 | 500 |

5.9 Coming Up

In our next Chapter, we'll look at several approaches to building tabular and graphical summaries for these data, beyond the introduction provided here.

6 More NHANES Summaries

6.1 Setup: Packages Used Here

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

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

theme_set(theme_bw())
```

6.2 Re-Loading the “Complete Cases” Sample

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

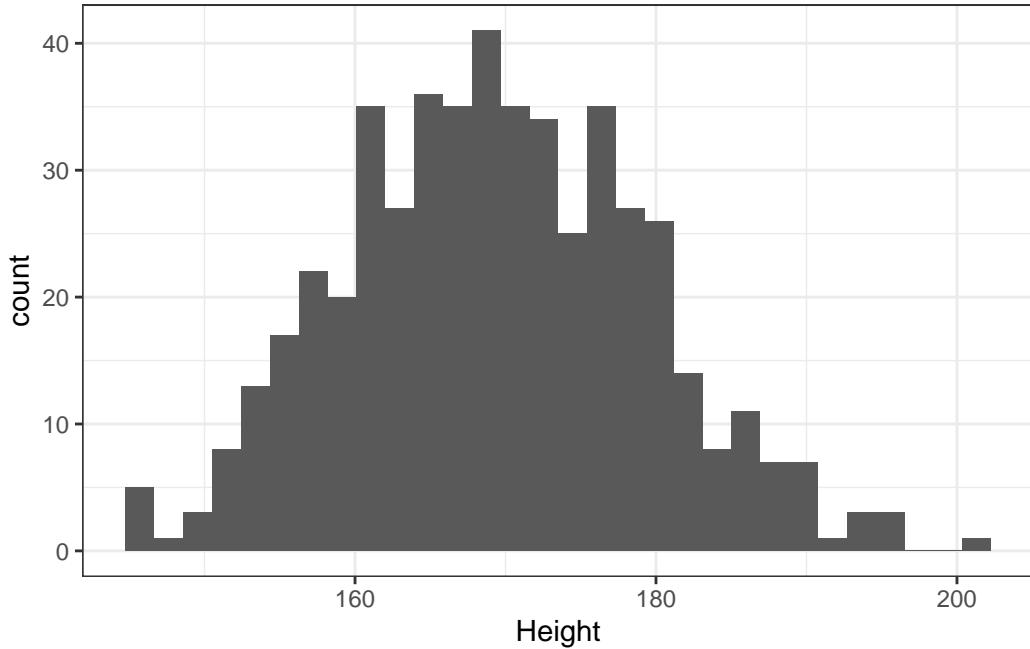
```
nh_500cc <- read_rds("data/nh_adult500cc.Rds")
```

6.3 Distribution of Heights

What is the distribution of height in this new sample?

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

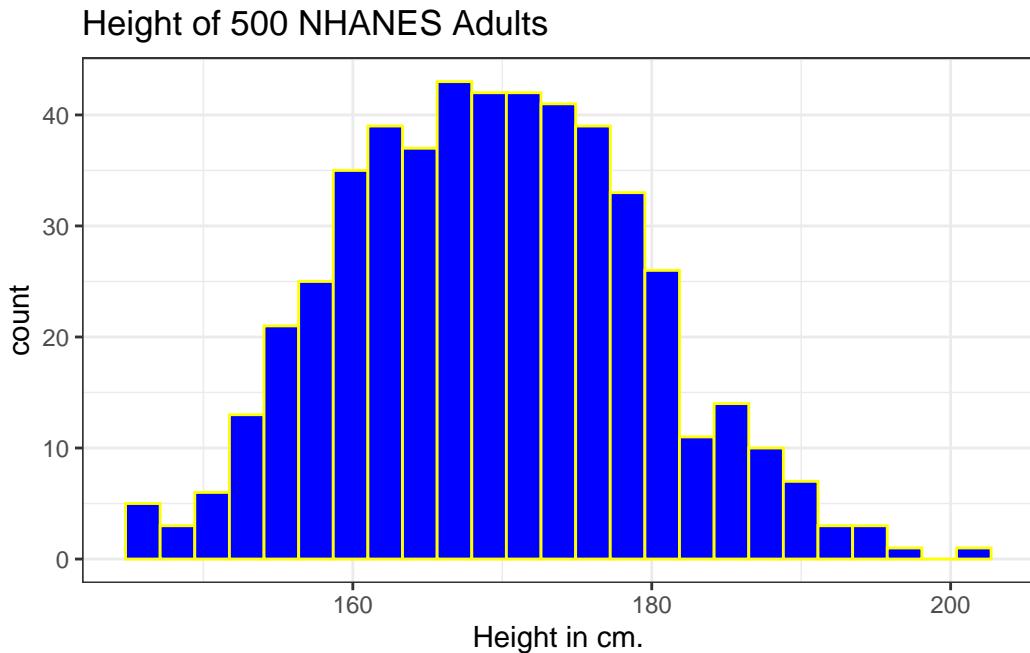
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



We can do several things to clean this up.

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

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



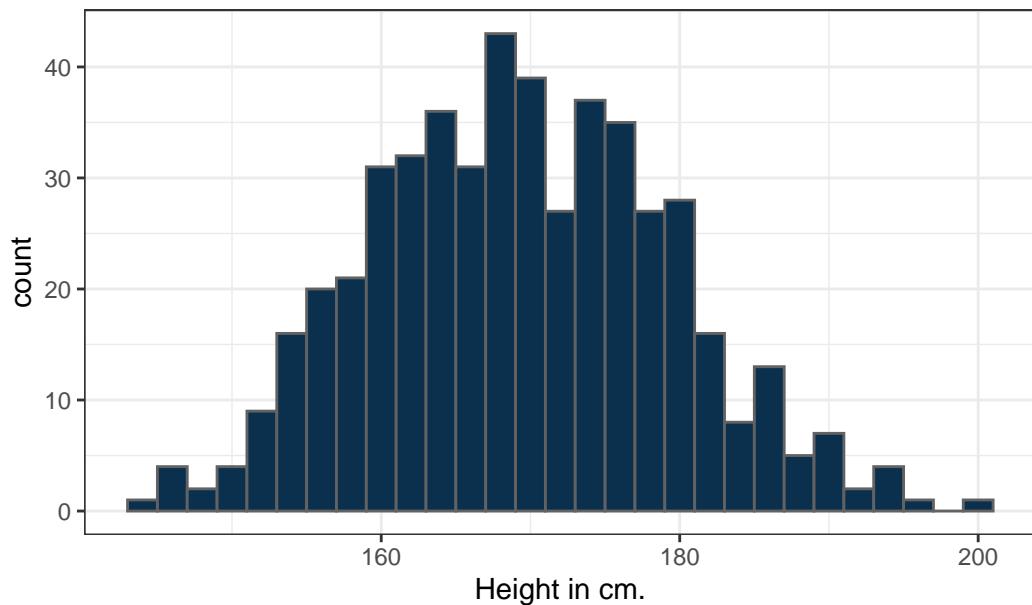
6.3.1 Changing a Histogram's Fill and Color

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

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

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

Height of 500 NHANES Adults

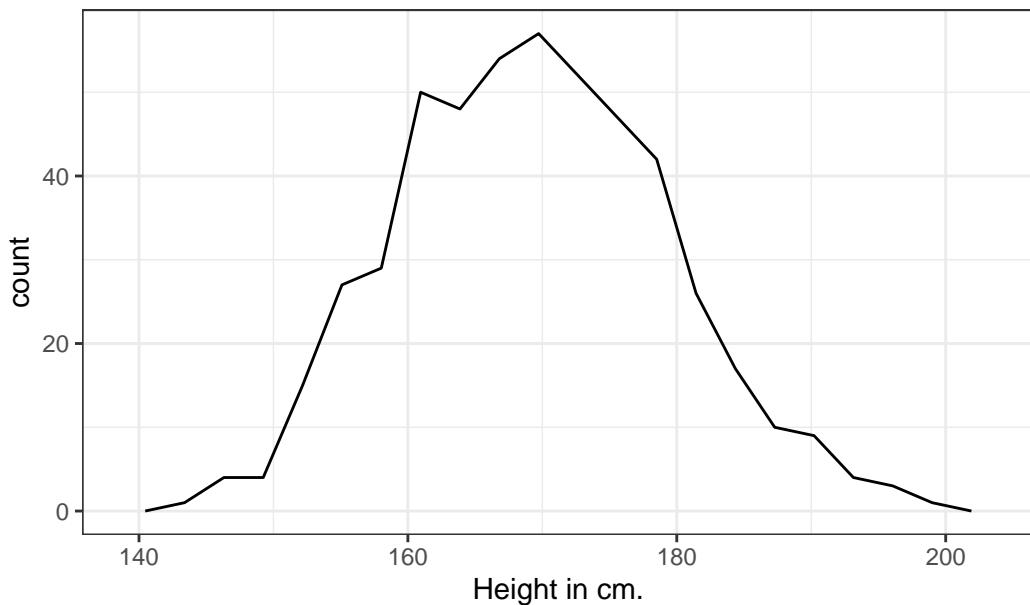


6.3.2 Using a frequency polygon

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

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

Height of 500 NHANES Adults

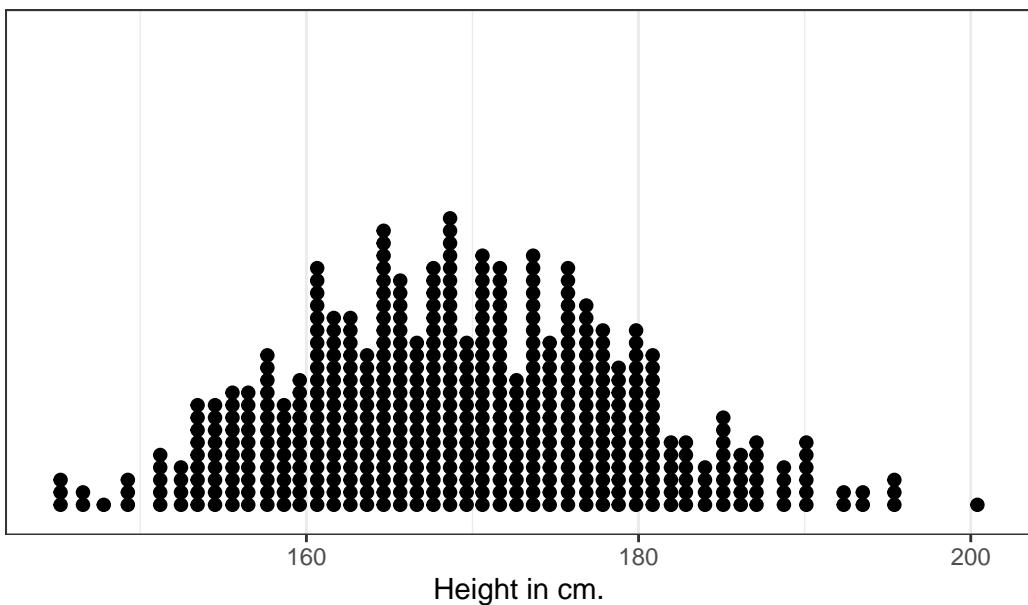


6.3.3 Using a dotplot

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

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

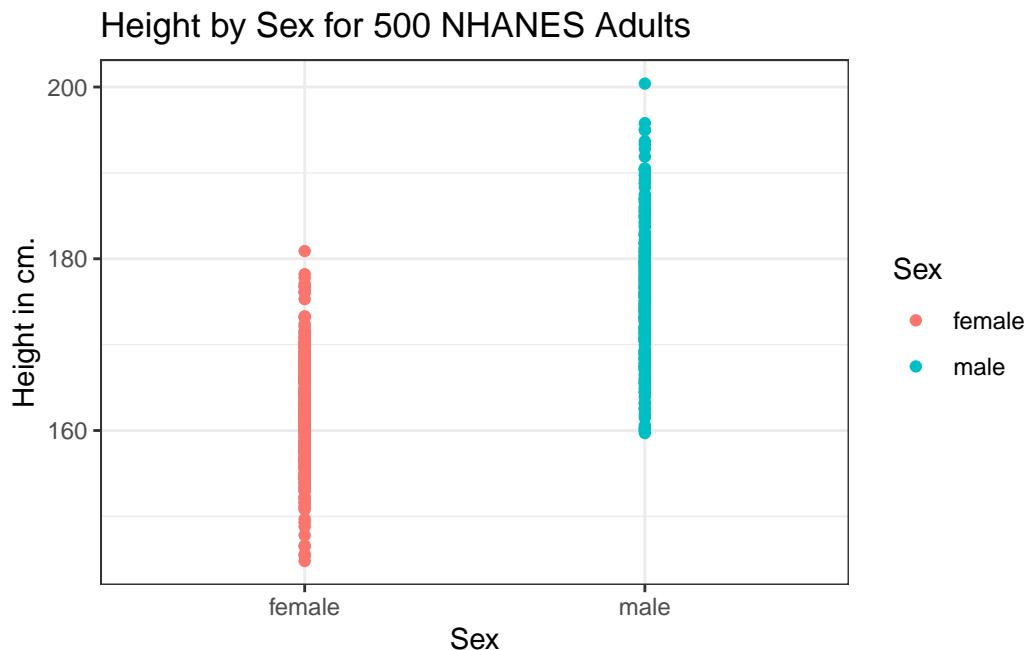
Height of 500 NHANES Adults



6.4 Height and Sex

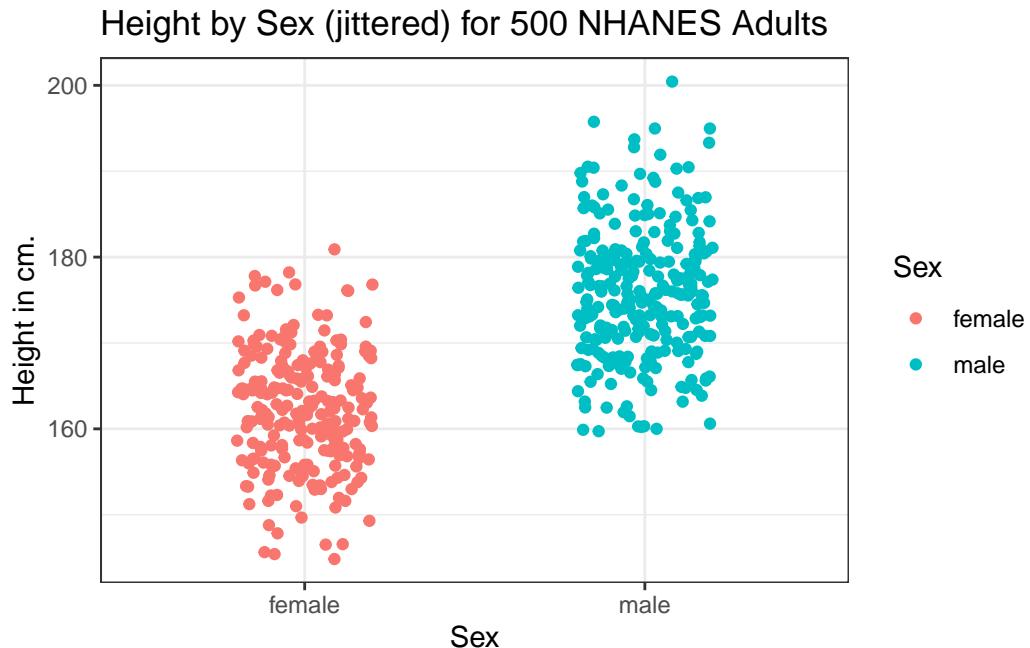
Let's look again at the impact of a respondent's sex on their height, but now within our sample of adults.

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



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

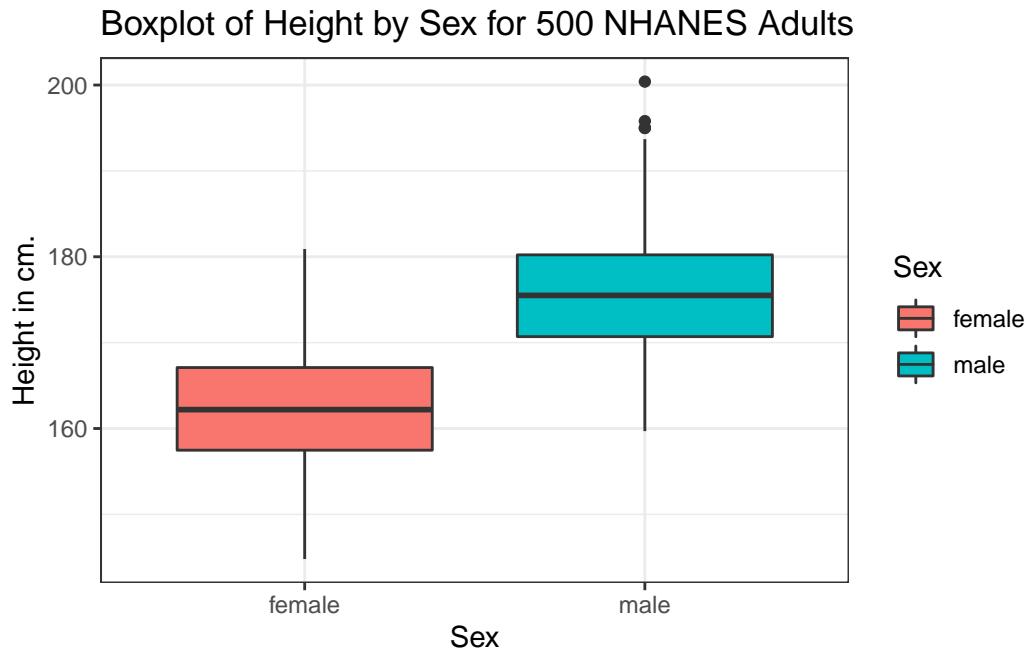
```
ggplot(data = nh_500cc, aes(x = Sex, y = Height, color = Sex)) +  
  geom_jitter(width = 0.2) +  
  labs(title = "Height by Sex (jittered) for 500 NHANES Adults",  
       y = "Height in cm.")
```



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

6.4.1 A Boxplot of Height by Sex

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



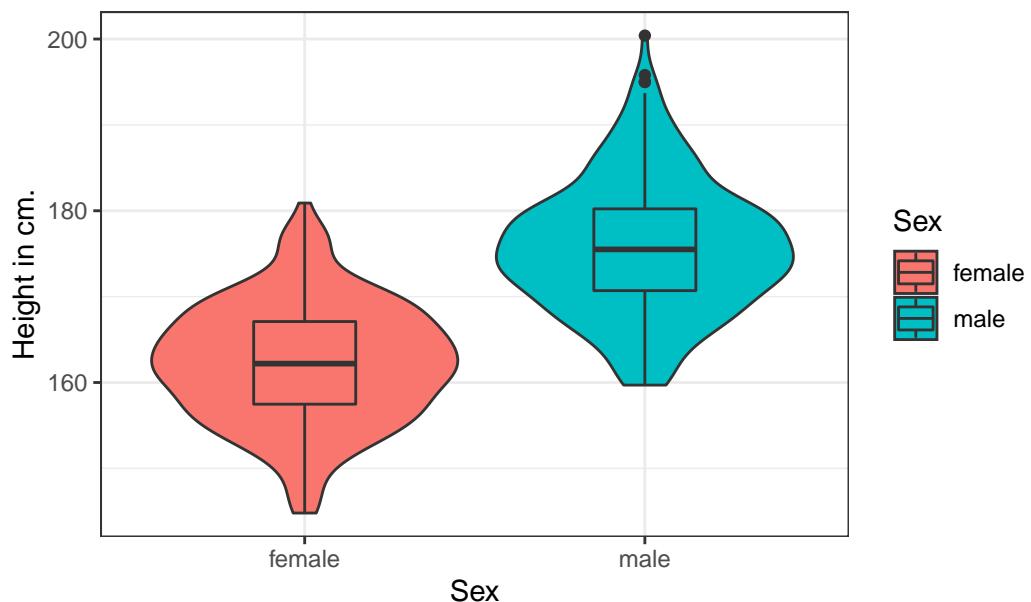
The boxplot shows some summary statistics based on percentiles. The boxes in the middle show the data values that include the middle half of the data once its been sorted. The 25th percentile (value that exceeds 1/4 of the data) is indicated by the bottom of the box, while the top of the box is located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

6.4.2 Adding a violin plot

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

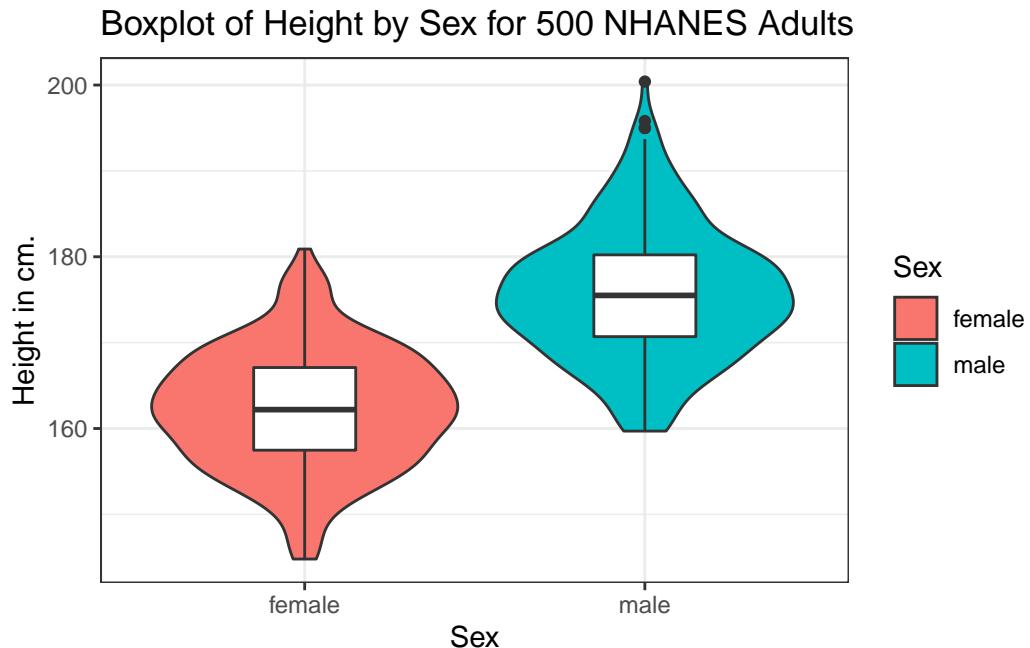
```
ggplot(data = nh_500cc, aes(x = Sex, y = Height, fill = Sex)) +
  geom_violin() +
  geom_boxplot(width = 0.3) +
  labs(title = "Boxplot of Height by Sex for 500 NHANES Adults",
       y = "Height in cm.")
```

Boxplot of Height by Sex for 500 NHANES Adults



This usually works better if the boxes are given a different fill than the violins, as shown in the following figure.

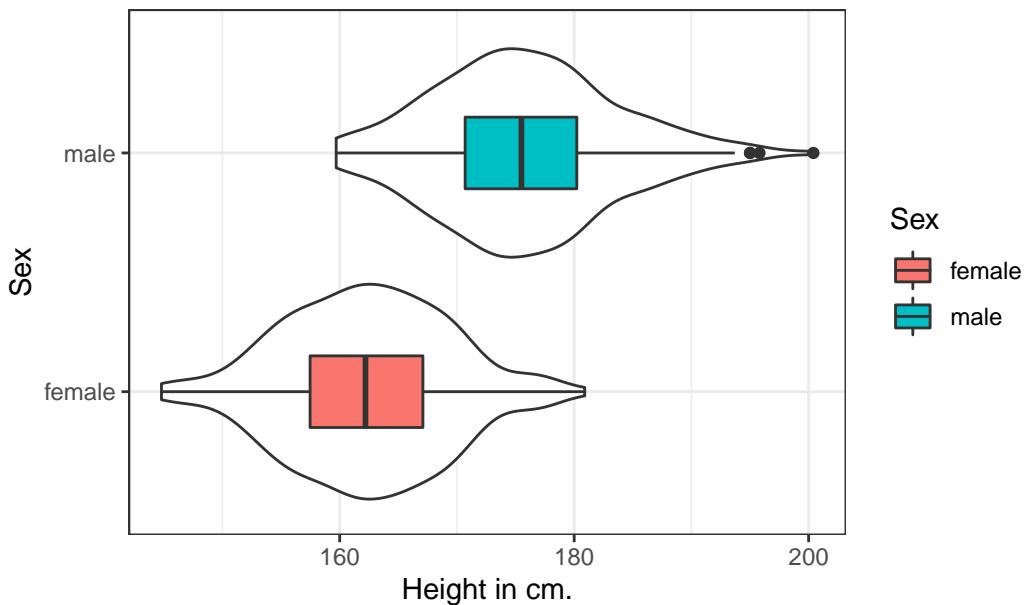
```
ggplot(data = nh_500cc, aes(x = Sex, y = Height)) +  
  geom_violin(aes(fill = Sex)) +  
  geom_boxplot(width = 0.3) +  
  labs(title = "Boxplot of Height by Sex for 500 NHANES Adults",  
       y = "Height in cm.")
```



We can also flip the boxplots on their side, using `coord_flip()`.

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

Boxplot of Height by Sex for 500 NHANES Adults

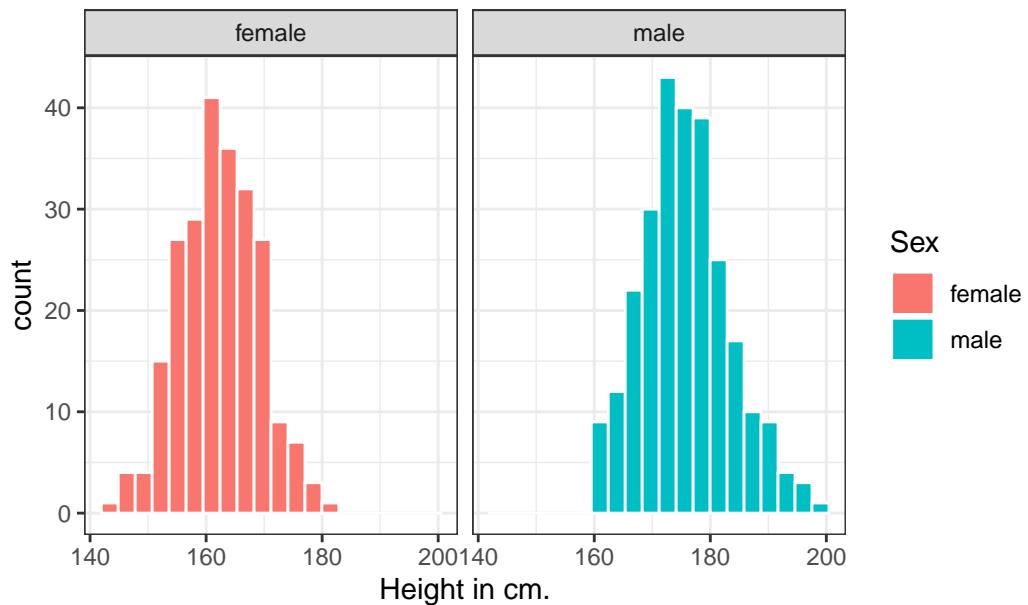


6.4.3 Histograms of Height by Sex

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

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

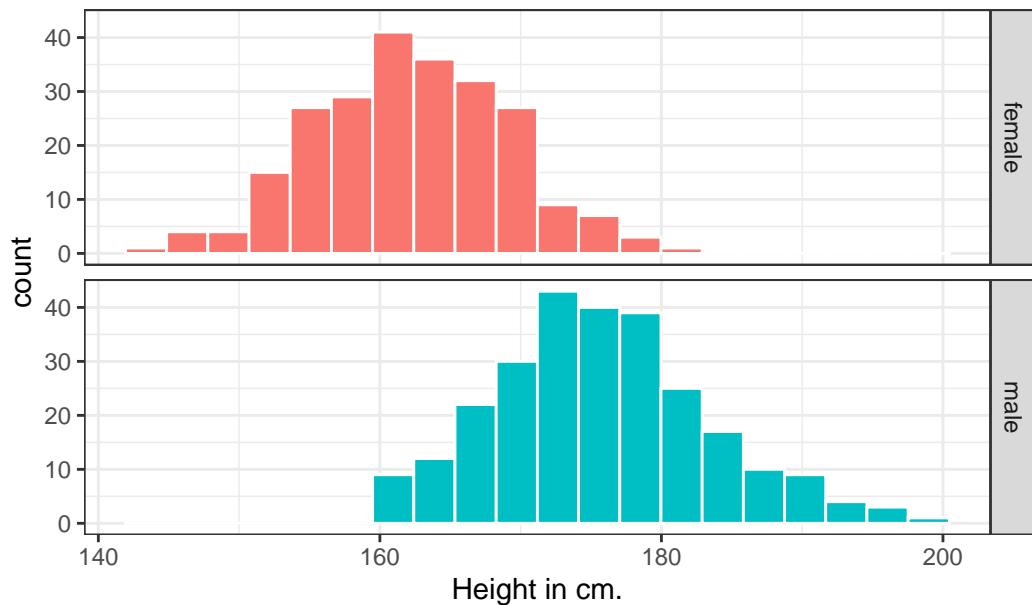
Histogram of Height by Sex for 500 NHANES Adults



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

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

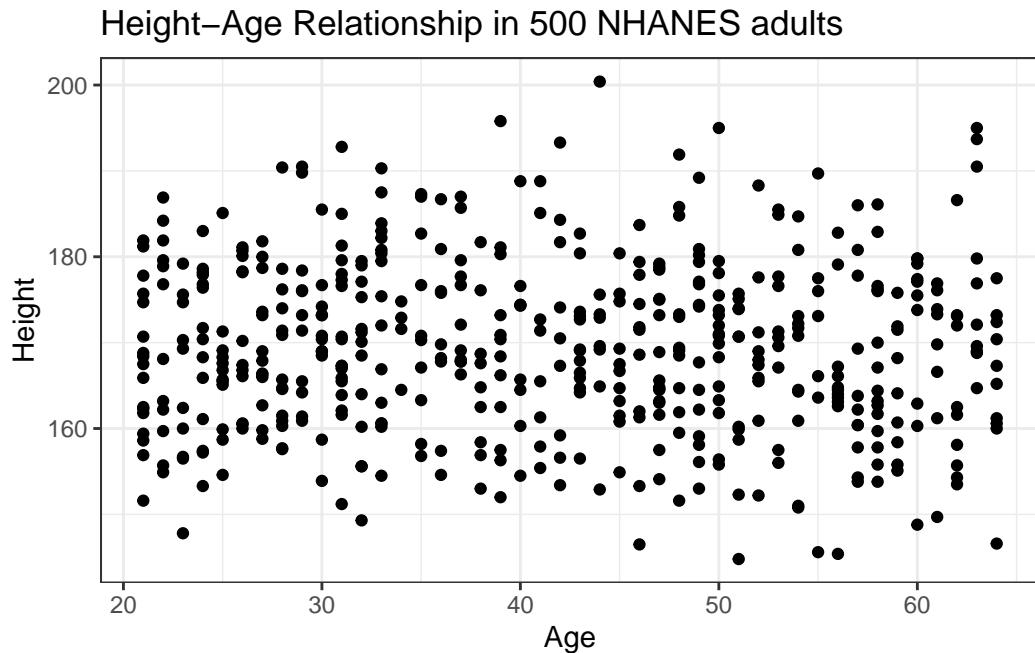
Histogram of Height by Sex for 500 NHANES Adults



6.5 Visualizing Age and Height's Relationship, by Sex

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

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

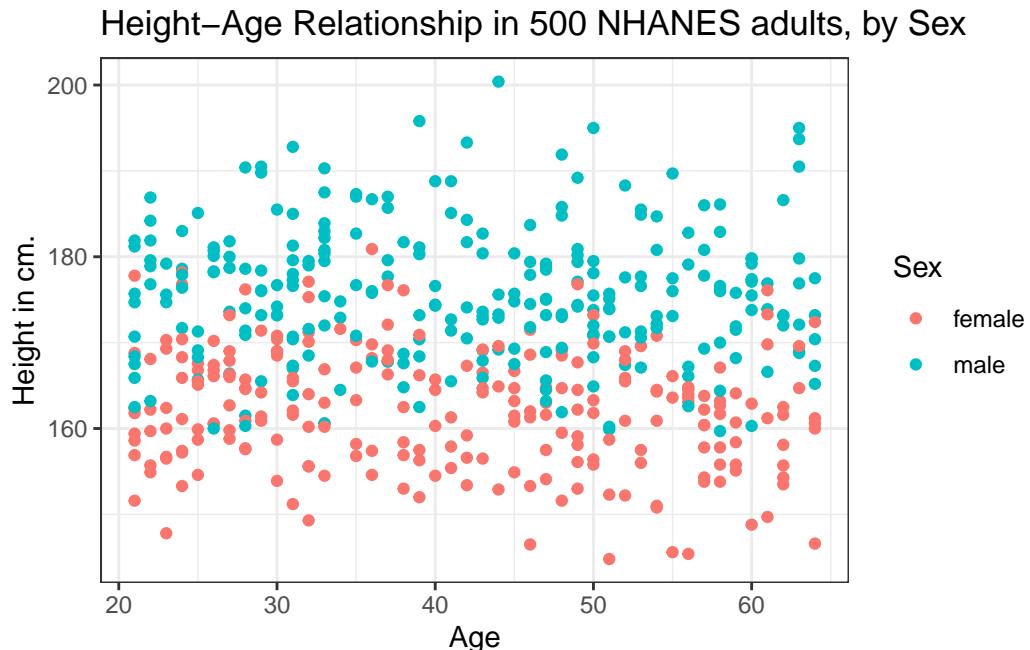


Is there a meaningful difference in what this relationship looks like, depending on Sex?

6.5.1 Adding Color to the plot

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

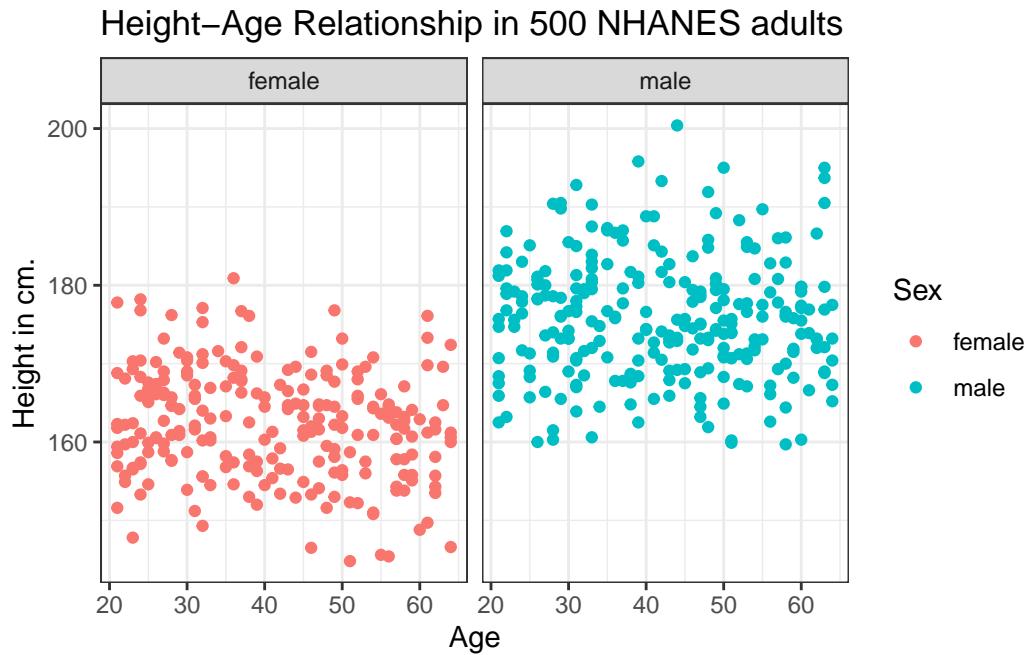
```
ggplot(data = nh_500cc, aes(x = Age, y = Height, color = Sex)) +
  geom_point() +
  labs(title = "Height–Age Relationship in 500 NHANES adults, by Sex",
       y = "Height in cm.")
```



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

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

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

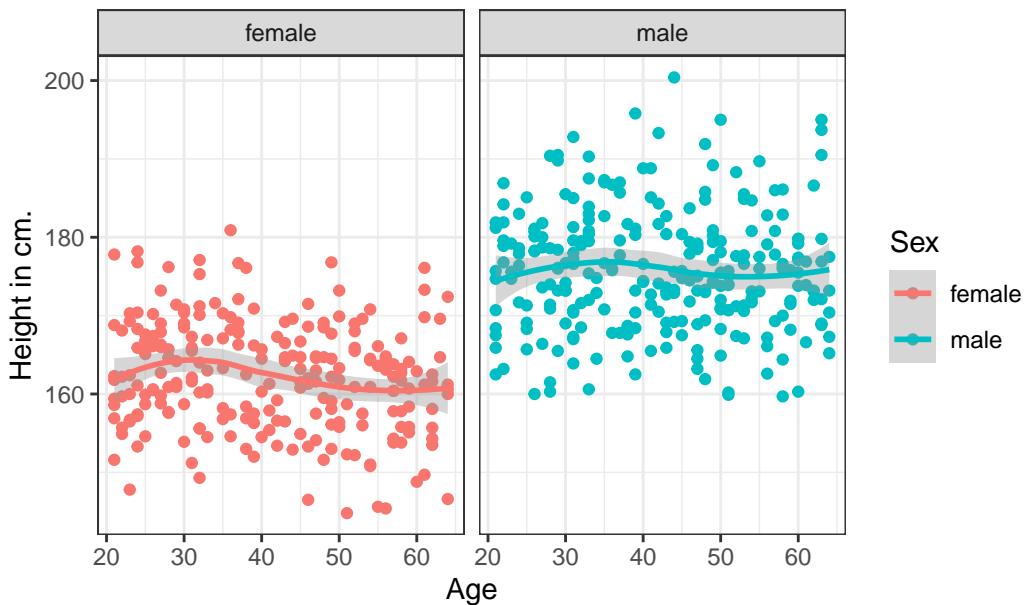


6.5.3 Can we add a smooth curve to show the relationship in each plot?

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

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

Height–Age Relationship in NHANES sample

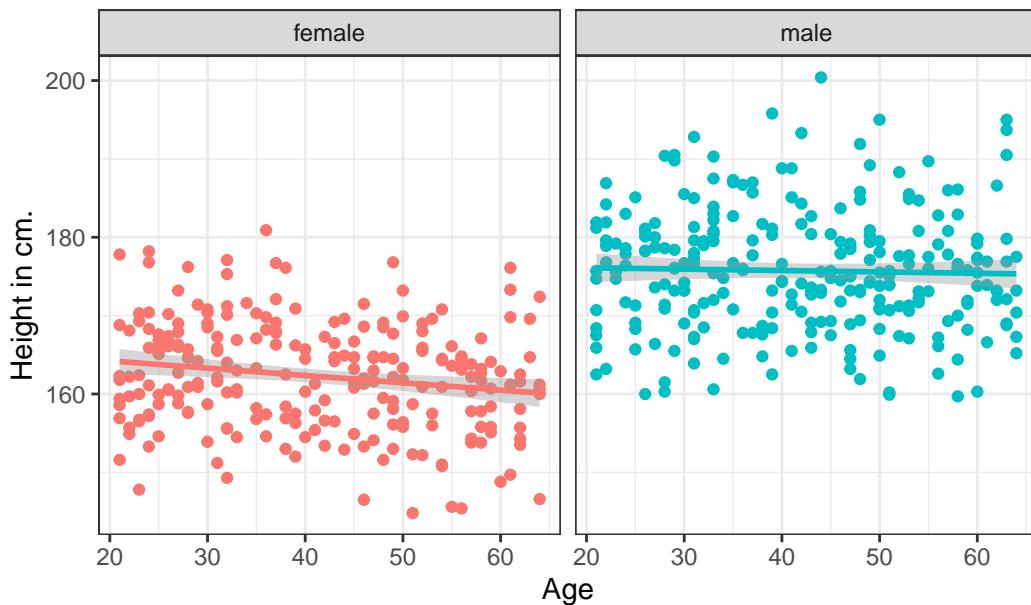


6.5.4 What if we want to assume straight line relationships?

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

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

Height–Age Relationship in NHANES sample



6.6 Combining Plots with patchwork

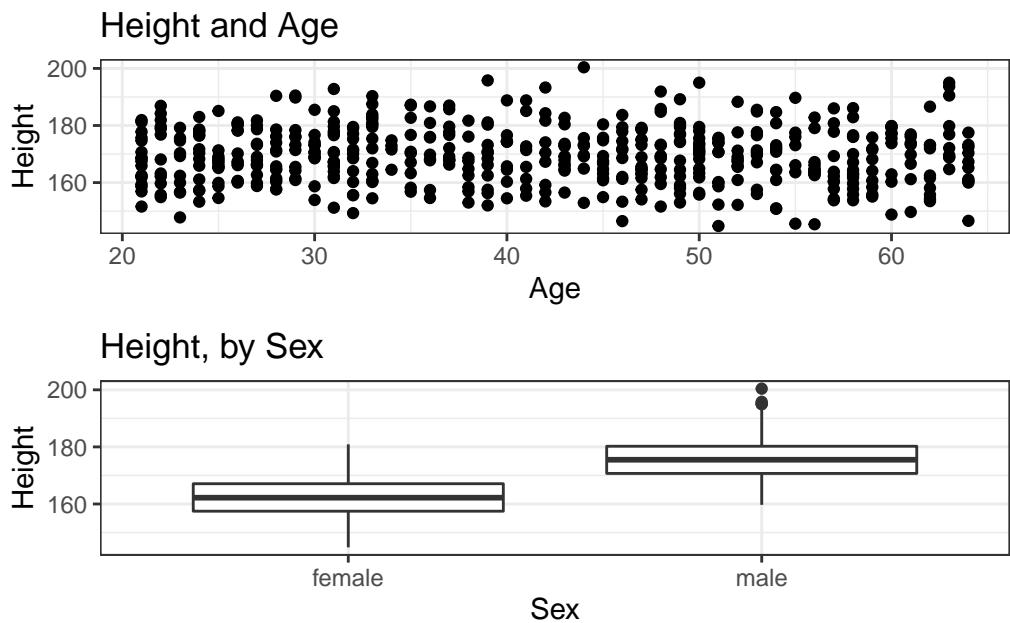
The `patchwork` package in R allows us to use some simple commands to put two plots together.

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

```
p1 <- ggplot(data = nh_500cc, aes(x = Age, y = Height)) +  
  geom_point() +  
  labs(title = "Height and Age")  
  
p2 <- ggplot(data = nh_500cc, aes(x = Sex, y = Height)) +  
  geom_boxplot() +  
  labs(title = "Height, by Sex")
```

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

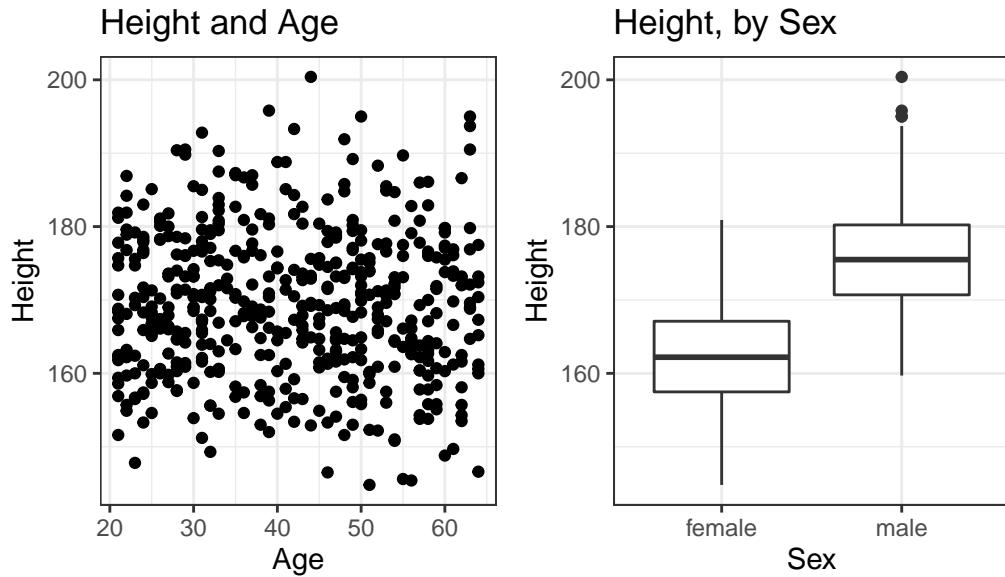
```
p1 / p2
```



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

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

Our Combined Plots



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

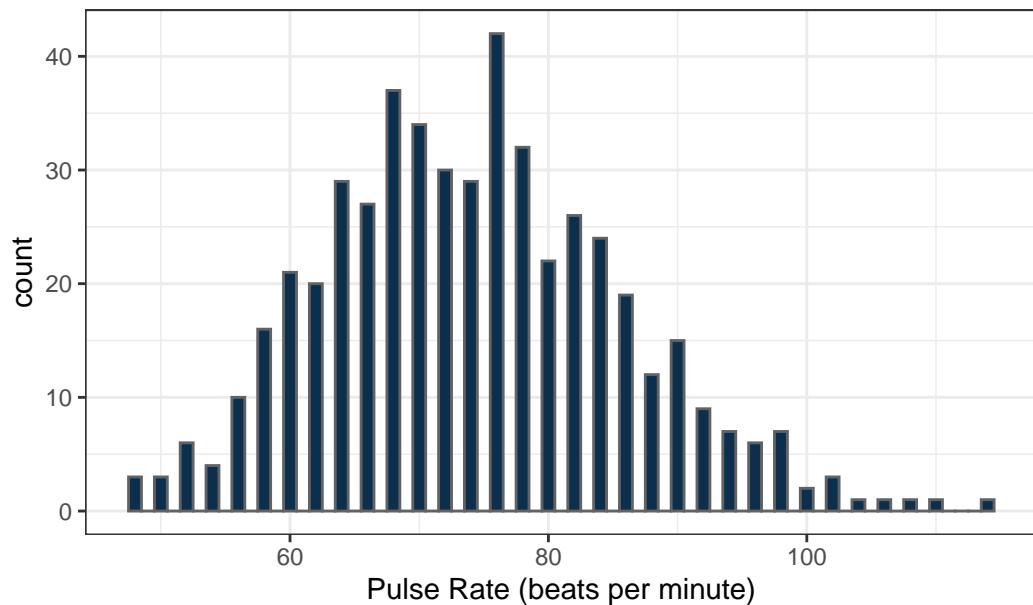
6.7 Looking at Pulse Rate

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

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

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

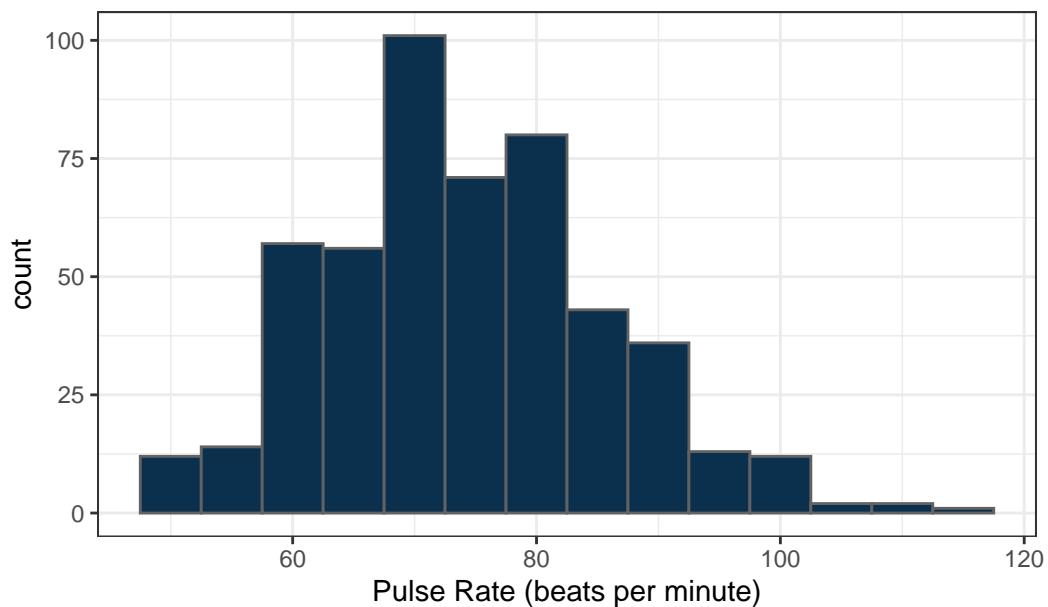
Histogram of Pulse Rate: NHANES Adults



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

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

Histogram of Pulse Rate: NHANES Adults



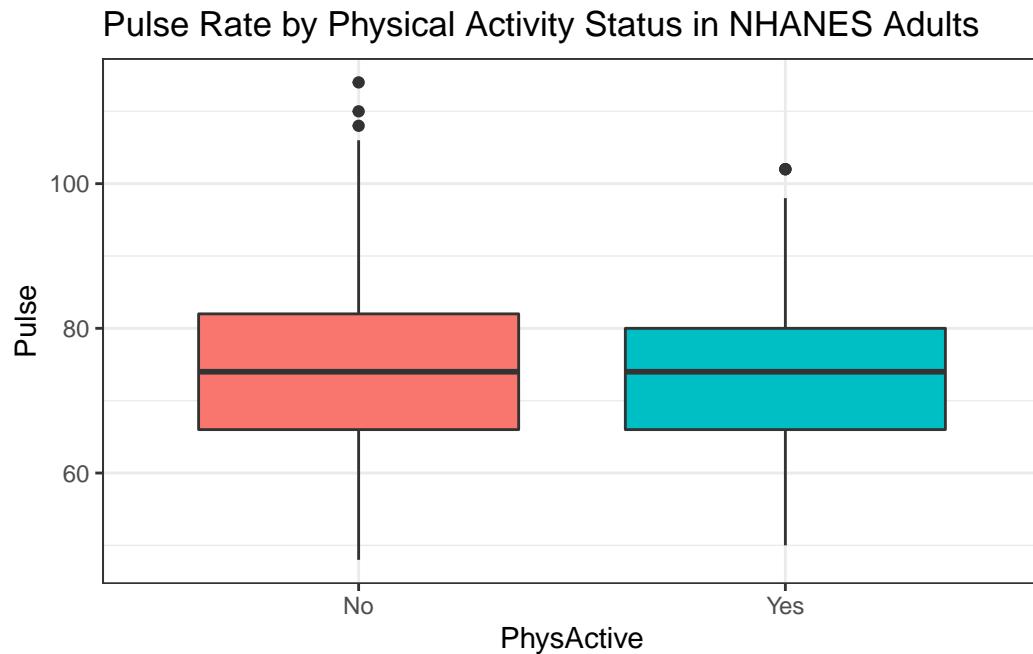
Which is the more useful representation will depend a lot on what questions you're trying to answer.

6.7.1 Pulse Rate and Physical Activity

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

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

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



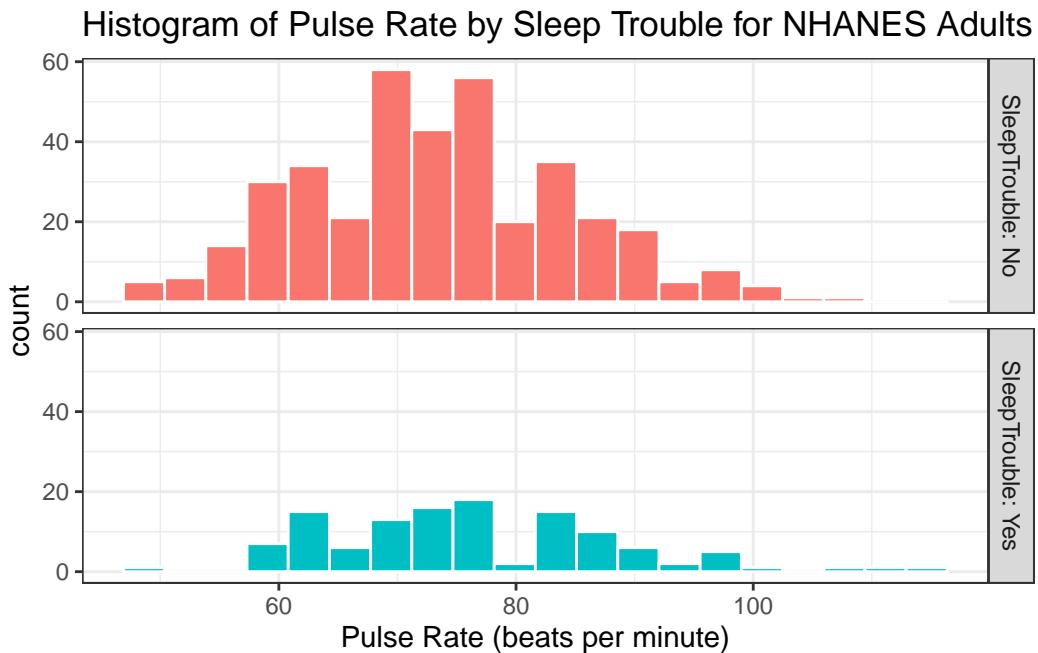
As an accompanying numerical summary, we might ask how many people fall into each of these `PhysActive` categories, and what is their “average” `Pulse` rate.

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

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

6.7.2 Pulse by Sleeping Trouble

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



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

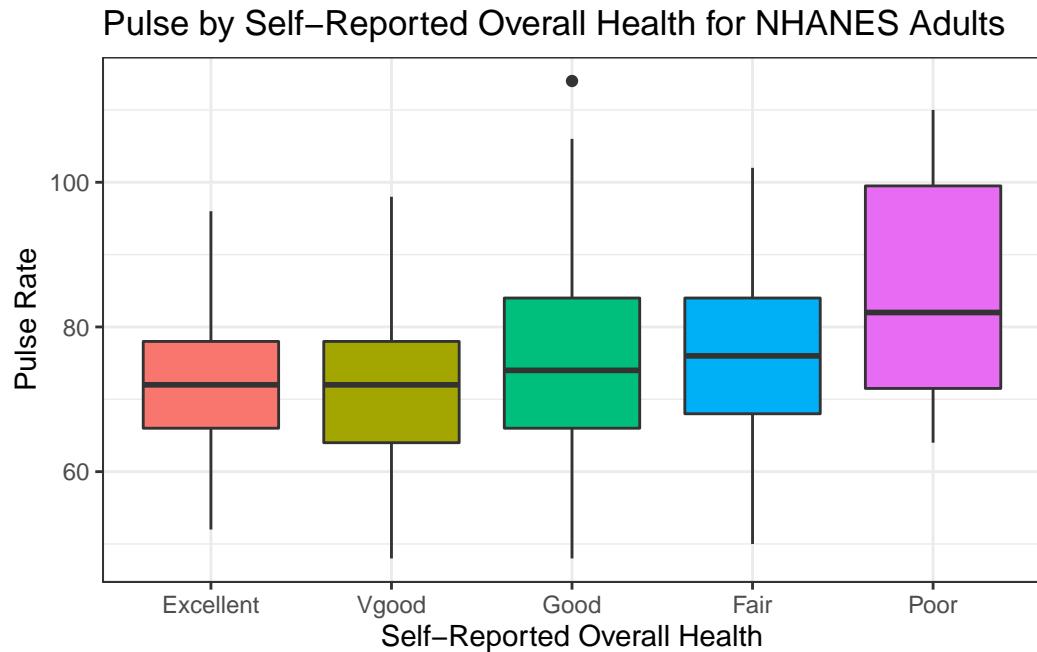
```
nh_500cc |>  
  group_by(SleepTrouble) |>  
  summarise(count = n(), mean(Pulse), median(Pulse)) |>  
  kbl(digits = 2) |>  
  kable_styling(full_width = F)
```

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

6.7.3 Pulse and HealthGen

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

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



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

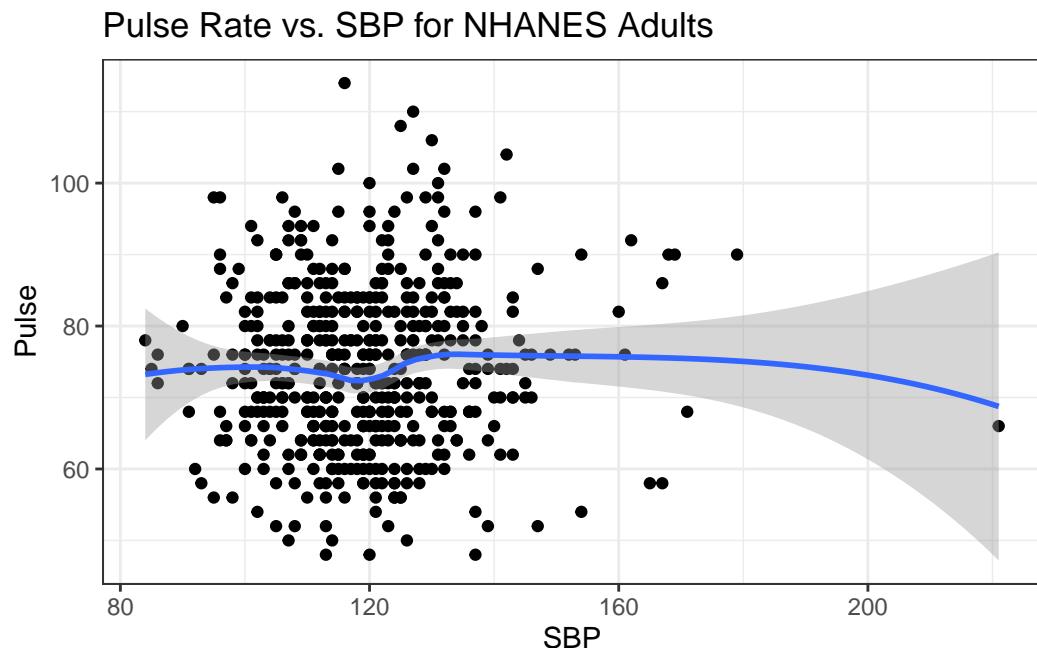
```
nh_500cc |>
  group_by(HealthGen) |>
  summarise(count = n(), mean(Pulse), median(Pulse)) |>
```

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

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

6.7.4 Pulse Rate and Systolic Blood Pressure

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

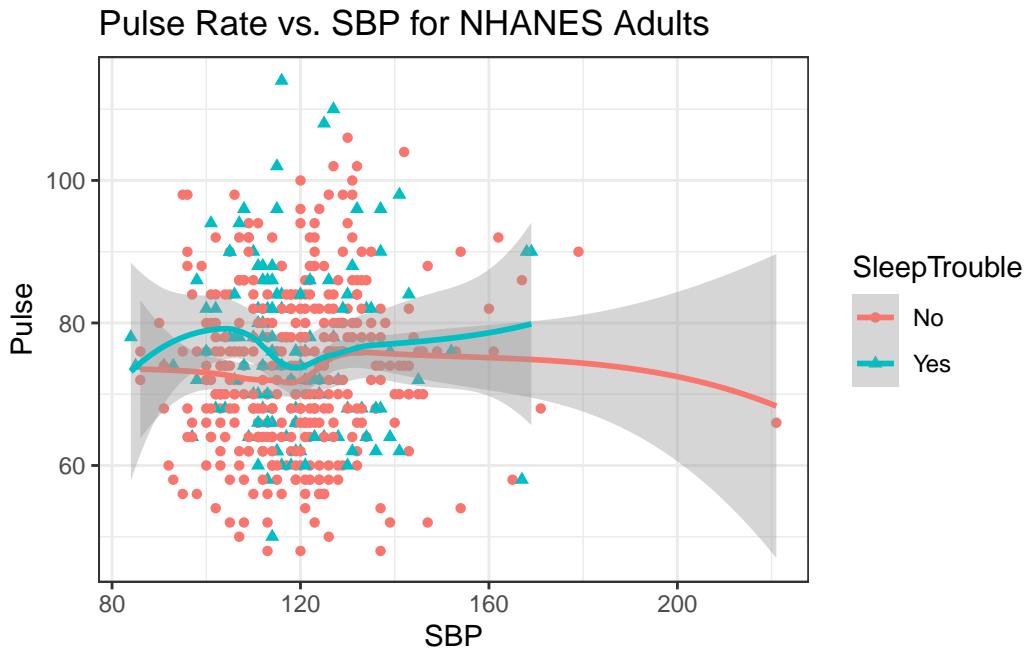


6.7.5 Sleep Trouble vs. No Sleep Trouble?

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

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

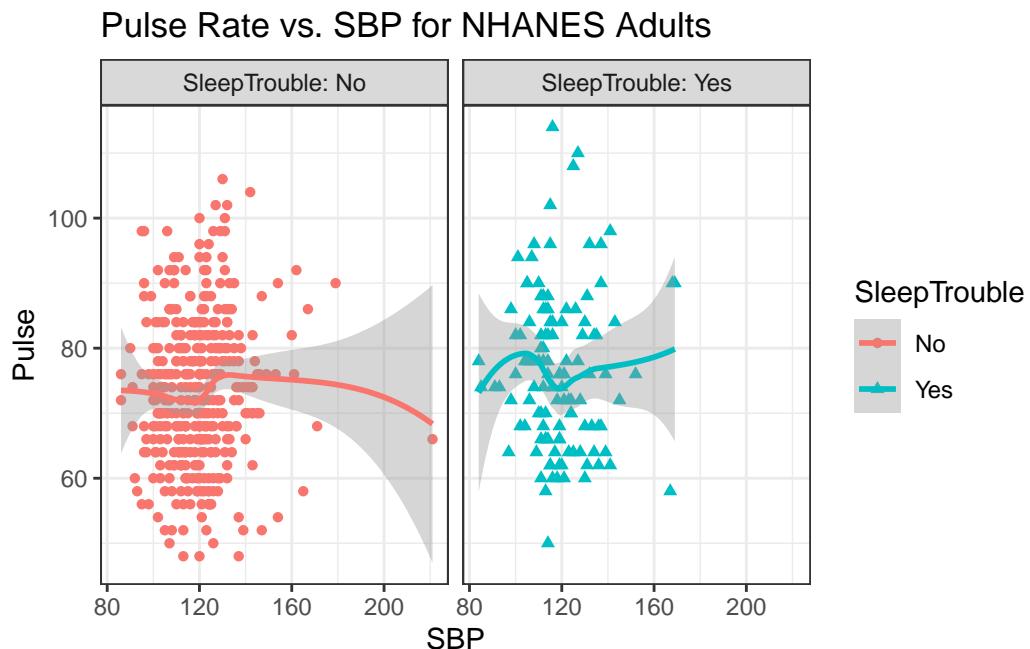
```
ggplot(data = nh_500cc,
       aes(x = SBP, y = Pulse,
           color = SleepTrouble, shape = SleepTrouble)) +
  geom_point() +
  geom_smooth(method = "loess", formula = y ~ x) +
  labs(title = "Pulse Rate vs. SBP for NHANES Adults")
```



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

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

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



6.8 General Health Status

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

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

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

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

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

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

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

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

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

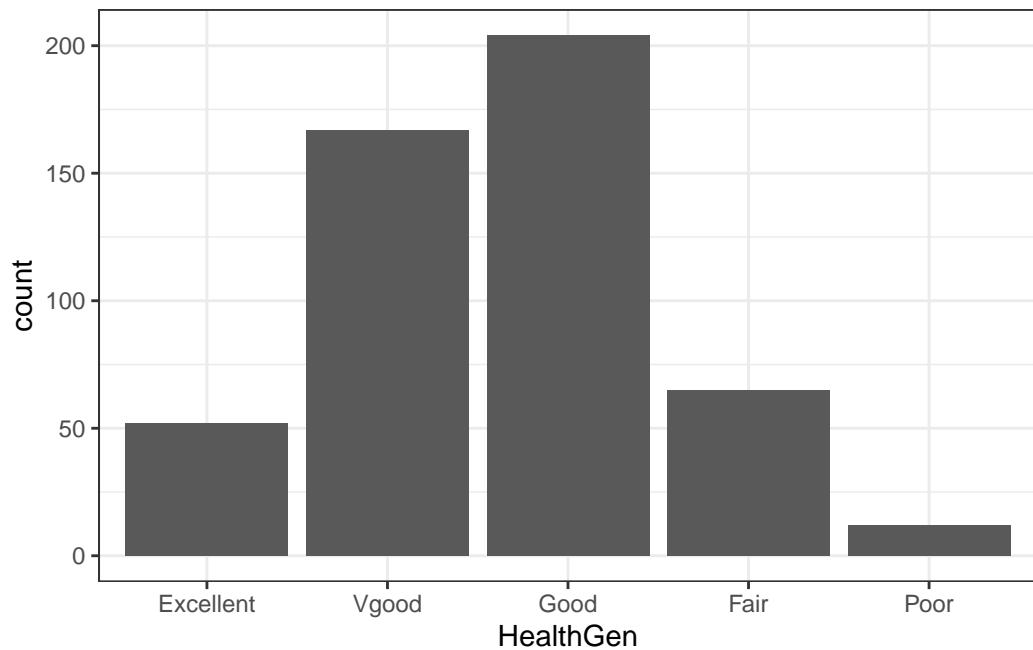
```
nh_500cc |>
  tabyl(MaritalStatus) |>
  adorn_totals() |>
  adorn_pct_formatting()
```

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

6.8.1 Bar Chart for Categorical Data

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

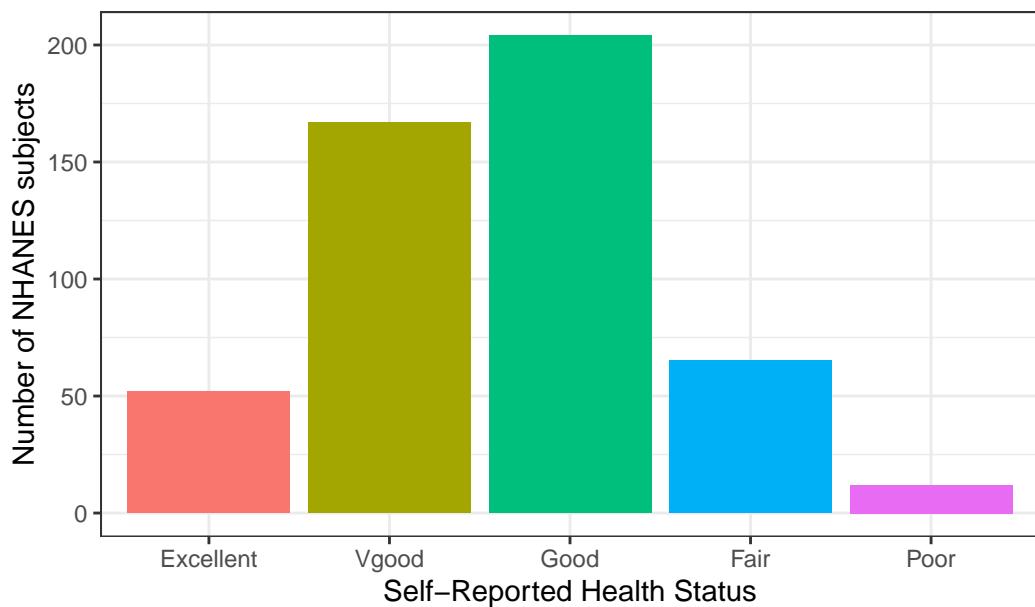
```
ggplot(data = nh_500cc, aes(x = HealthGen)) +  
  geom_bar()
```



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

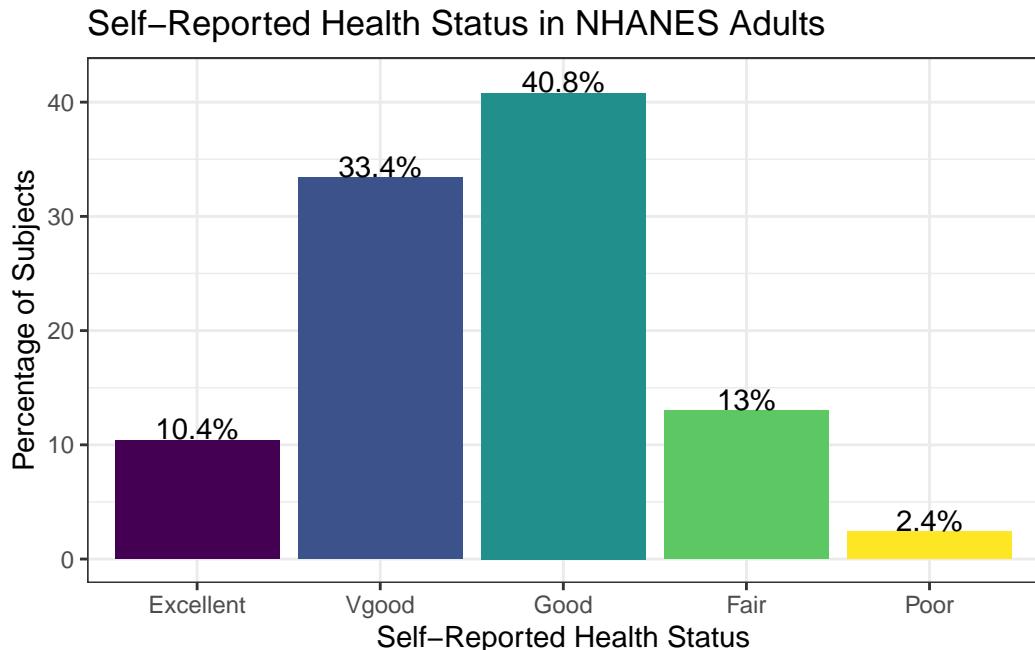
```
ggplot(data = nh_500cc, aes(x = HealthGen, fill = HealthGen)) +  
  geom_bar() +  
  guides(fill = "none") +  
  labs(x = "Self-Reported Health Status",  
       y = "Number of NHANES subjects",  
       title = "Self-Reported Health Status in NHANES Adults")
```

Self-Reported Health Status in NHANES Adults



Or, we can really go crazy...

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



6.9 Two-Way Tables

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

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

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

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

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

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

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

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

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

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

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

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

```

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

```

6.10 SBP by General Health Status

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

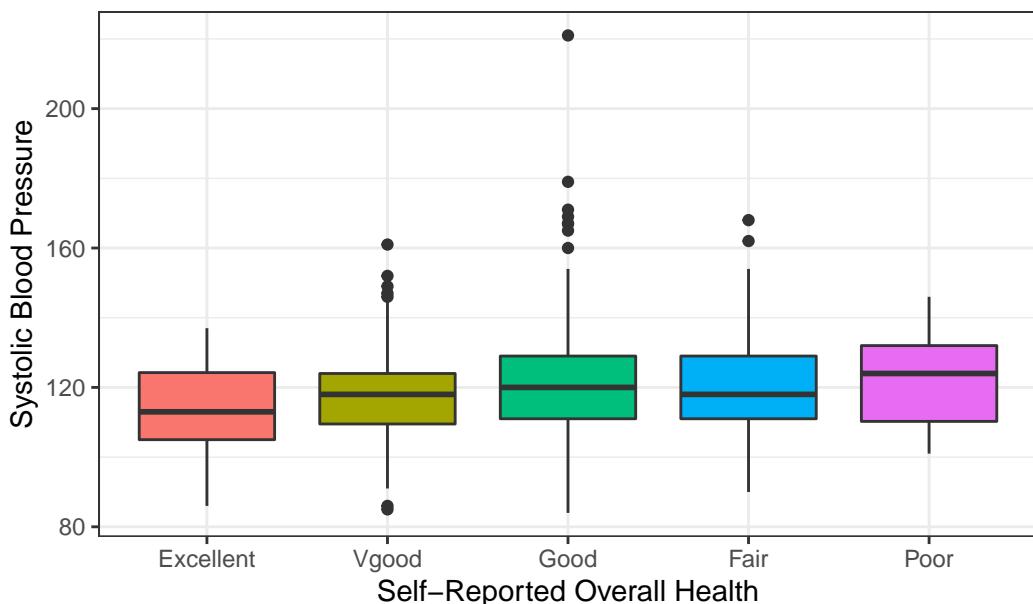
```

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

```

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

SBP by Health Status, Overall Health for NHANES Adults



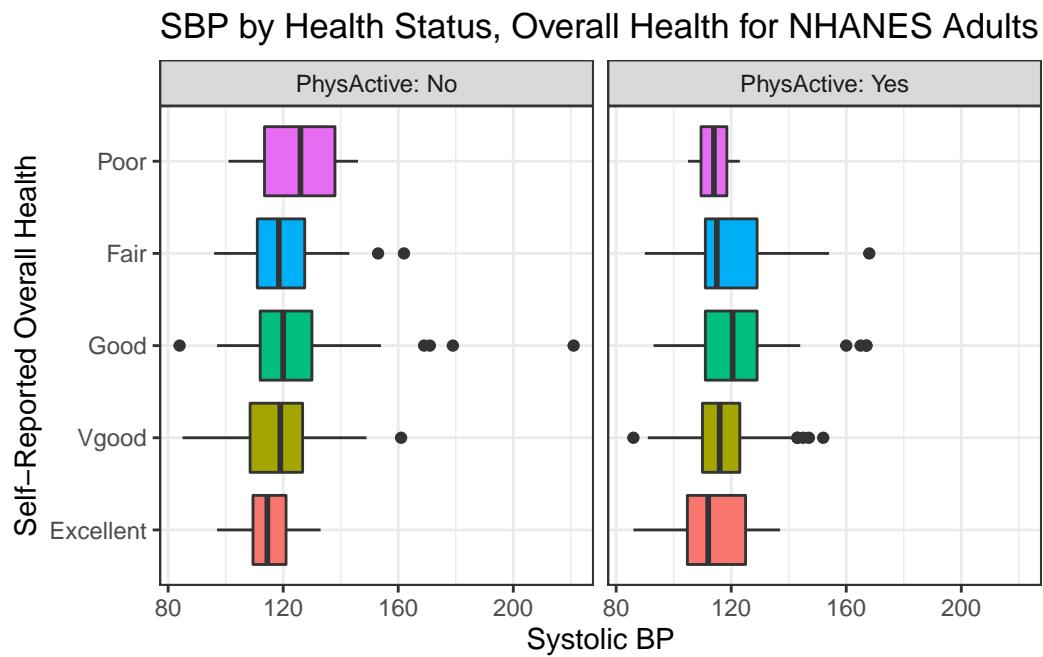
We can see that not too many people self-identify with the “Poor” health category.

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

6.10.1 SBP by Physical Activity and General Health Status

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

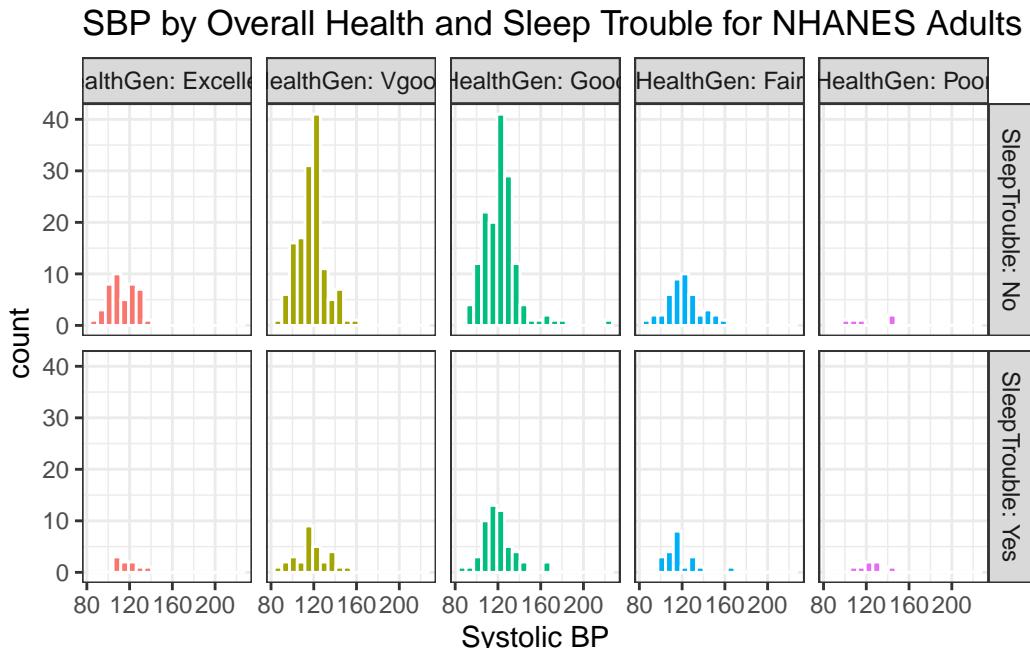
```
ggplot(data = nh_500cc, aes(x = HealthGen, y = SBP, fill = HealthGen)) +
  geom_boxplot() +
  labs(title = "SBP by Health Status, Overall Health for NHANES Adults",
       y = "Systolic BP", x = "Self-Reported Overall Health") +
  guides(fill = "none") +
  facet_wrap(~ PhysActive, labeller = "label_both")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



6.10.2 SBP by Sleep Trouble and General Health Status

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

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



6.11 Conclusions

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

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

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

7 Summarizing Quantities

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

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

7.1 Setup: Packages Used Here

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

library(patchwork)
library(tidyverse)

theme_set(theme_bw())
```

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

7.2 Working with the nh_750 data

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

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

7.3 The summary function for Quantitative data

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

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

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

| | Pulse |
|---------|---------|
| Min. | : 40.00 |
| 1st Qu. | : 66.00 |
| Median | : 72.00 |
| Mean | : 73.53 |
| 3rd Qu. | : 80.00 |
| Max. | :124.00 |
| NA's | :32 |

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

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

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

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

7.4 Measuring the Center of a Distribution

7.4.1 The Mean and The Median

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

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

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

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

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

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

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

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

```
9      21  
10     21  
# ... with 740 more rows  
# i Use `print(n = ...)` to see more rows
```

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

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

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

```
[726] 63 63 63 63 63 63 63 63 63 63 63 63 64 64 64 64 64 64 64 64 64 64 64 64 64 64
```

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

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

```
[1] 33 33
```

7.4.2 Dealing with Missingness

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

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

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

This fails because we have some missing values in the Pulse data. We can address this by either omitting the data with missing values before we run the `summarise()` function, or tell the mean and median summary functions to remove missing values².

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

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

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

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

```

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

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

```

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

7.4.3 The Mode of a Quantitative Variable

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

```

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

# A tibble: 44 x 2
  Age   count
  <int> <int>
1 32     28
2 36     26
3 50     26
4 30     24
5 33     24
6 24     23
7 21     22
8 22     22
9 23     22
10 28    20
# ... with 34 more rows
# i Use `print(n = ...)` to see more rows

```

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

Note the use of three different “verbs” in our function there - for more explanation of this strategy, visit Wickham and Grolemund (2022). The `group_by` function here is very useful. It converts the `nh_750` data frame into a new grouped tibble where operations are performed on the groups. Here, this means that it groups the data by Age before counting observations, and then sorting the groups (the Ages) by their frequencies.

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

7.5 Measuring the Spread of a Distribution

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

7.5.1 The Range and the Interquartile Range (IQR)

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

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

```
[1] 21 64
```

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

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

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

```
[1] 16.7 80.6
```

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

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

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

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

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

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

| | Pulse |
|---------|---------|
| Min. | : 40.00 |
| 1st Qu. | : 66.00 |
| Median | : 72.00 |
| Mean | : 73.53 |
| 3rd Qu. | : 80.00 |
| Max. | :124.00 |
| NA's | :32 |

7.5.2 The Variance and the Standard Deviation

The IQR is always a reasonable summary of spread, just as the median is always a reasonable summary of the center of a distribution. Yet, most people are inclined to summarize a batch of data using two numbers: the **mean** and the **standard deviation**. This is really only a sensible thing to do if you are willing to assume the data follow a Normal distribution: a bell-shaped, symmetric distribution without substantial outliers.

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

7.5.3 Obtaining the Variance and Standard Deviation in R

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

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

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

And here are the standard deviations of those same variables.

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

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

7.5.4 Defining the Variance and Standard Deviation

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

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

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

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

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

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

7.5.5 Interpreting the SD when the data are Normally distributed

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

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

⁴When we divide by $n-1$ as we calculate the sample variance, the average of the sample variances for all possible samples is equal to the population variance. If we instead divided by n , the average sample variance across all possible samples would be a little smaller than the population variance.

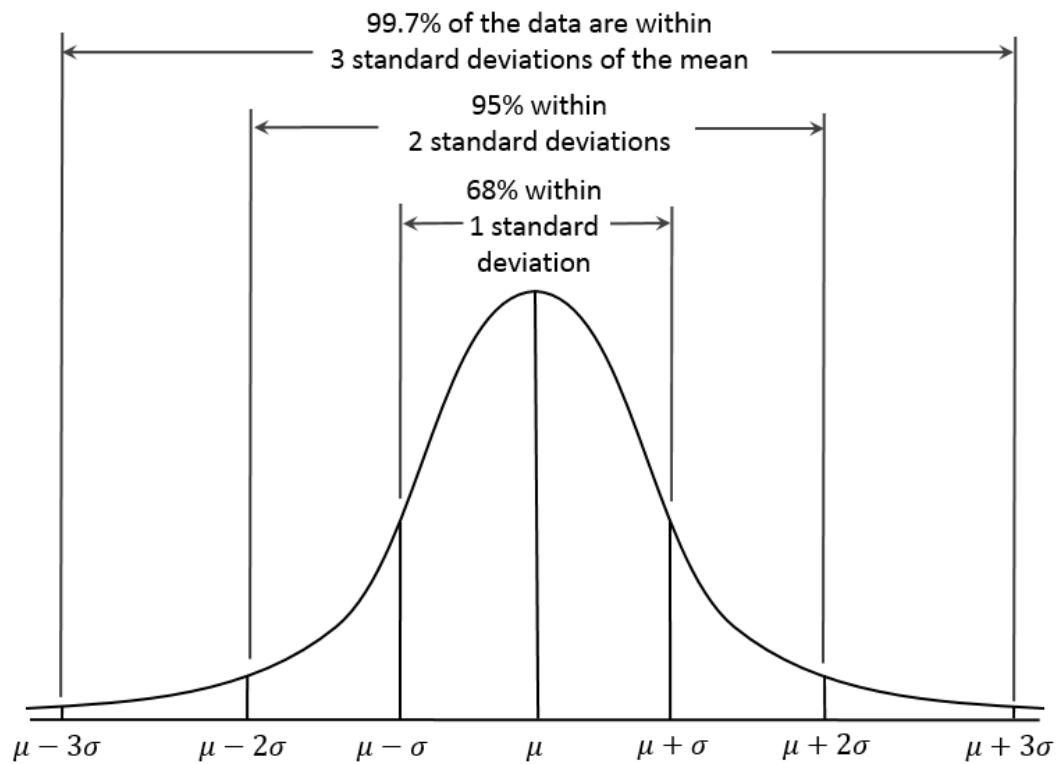


Figure 7.1: The Normal Distribution and the Empirical Rule

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

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

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

7.6 Measuring the Shape of a Distribution

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

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

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

7.6.1 Multimodal vs. Unimodal distributions

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

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

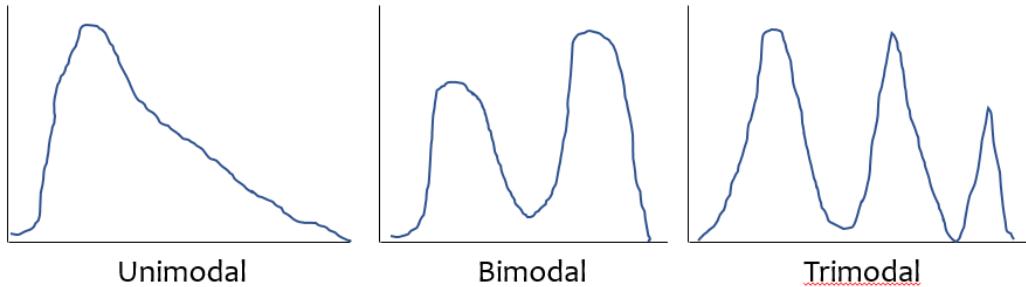


Figure 7.2: Unimodal and Multimodal Sketches

7.6.2 Skew

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

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

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

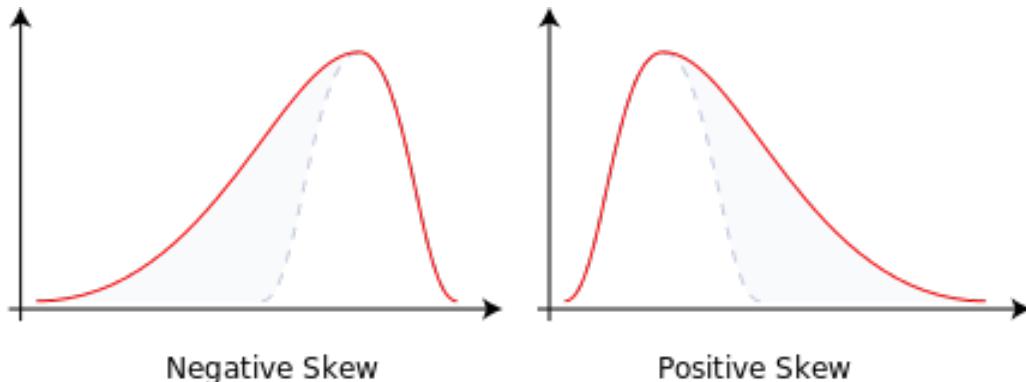


Figure 7.3: Negative (Left) Skew and Positive (Right) Skew

7.6.3 Kurtosis

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

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

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

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

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

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

```

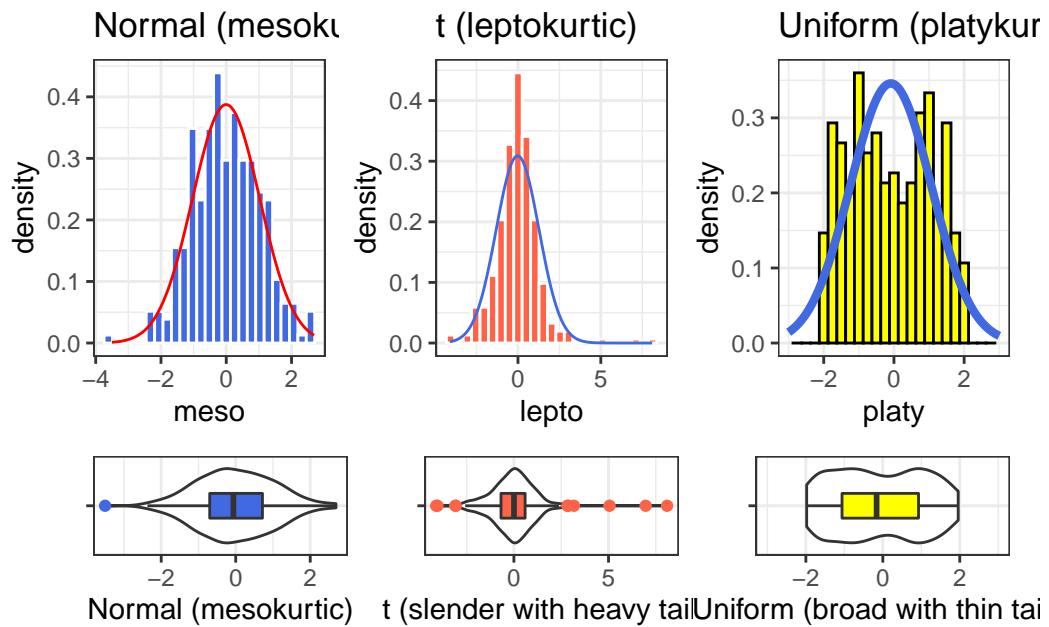
p2a <- ggplot(sims_kurt, aes(x = lepto, y = "")) +
  geom_violin() +
  geom_boxplot(fill = "tomato", outlier.color = "tomato", width = 0.3) +
  labs(y = "", x = "t (slender with heavy tails)")

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

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

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

```



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

7.7 Numerical Summaries for Quantitative Variables

7.7.1 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_750)

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

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

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

```

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

# A tibble: 1 x 9
  min     Q1 median     Q3   max   mean     sd     n miss
  <int> <dbl> <int> <dbl> <int> <dbl> <dbl> <int> <int>
1    83    108    118    127    209   119.   15.1    717     0

```

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

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

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

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

```

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

# A tibble: 5 x 10
  Education      min     Q1 median     Q3   max   mean     sd     n miss
  <fct>       <int> <dbl> <dbl> <dbl> <int> <dbl> <dbl> <int> <int>
1 8th Grade      96    110.    120.    130.    167   122.   16.3    46     0

```

| | | | | | | | | | |
|------------------|----|------|------|-----|-----|------|------|-----|---|
| 2 9 - 11th Grade | 85 | 108. | 116 | 127 | 191 | 119. | 15.8 | 76 | 0 |
| 3 High School | 84 | 112. | 120. | 129 | 209 | 121. | 16.5 | 136 | 0 |
| 4 Some College | 85 | 108 | 117 | 126 | 186 | 119. | 14.3 | 232 | 0 |
| 5 College Grad | 83 | 107 | 117 | 125 | 171 | 117. | 14.4 | 227 | 0 |

7.7.2 describe in the psych package

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

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

| | vars | n | mean | sd | median | trimmed | mad | min | max | range | skew |
|-------|------|----------|--------|-------|--------|---------|-------|------|-------|-------|-------|
| Age | 1 | 750 | 40.82 | 12.54 | 40.0 | 40.53 | 14.83 | 21.0 | 64.0 | 43.0 | 0.16 |
| BMI | 2 | 745 | 29.08 | 7.24 | 27.9 | 28.31 | 5.93 | 16.7 | 80.6 | 63.9 | 1.72 |
| SBP | 3 | 717 | 118.79 | 15.14 | 118.0 | 117.88 | 13.34 | 83.0 | 209.0 | 126.0 | 0.96 |
| DBP | 4 | 717 | 72.69 | 11.34 | 73.0 | 72.65 | 10.38 | 0.0 | 108.0 | 108.0 | -0.28 |
| Pulse | 5 | 718 | 73.53 | 11.65 | 72.0 | 73.11 | 11.86 | 40.0 | 124.0 | 84.0 | 0.48 |
| | | kurtosis | se | | | | | | | | |
| Age | | -1.15 | 0.46 | | | | | | | | |
| BMI | | 6.16 | 0.27 | | | | | | | | |
| SBP | | 3.10 | 0.57 | | | | | | | | |
| DBP | | 2.59 | 0.42 | | | | | | | | |
| Pulse | | 0.73 | 0.43 | | | | | | | | |

The additional statistics presented here are:

- `trimmed` = a trimmed mean (by default in this function, this removes the top and bottom 10% from the data, then computes the mean of the remaining values - the middle 80% of the full data set.)
- `mad` = the median absolute deviation (from the median), which can be used in a manner similar to the standard deviation or IQR to measure spread.
 - If the data are Y_1, Y_2, \dots, Y_n , then the `mad` is defined as $\text{median}(|Y_i - \text{median}(Y_i)|)$.
 - To find the `mad` for a set of numbers, find the median, subtract the median from each value and find the absolute value of that difference, and then find the median of those absolute differences.
 - For non-normal data with a skewed shape but tails well approximated by the Normal, the `mad` is likely to be a better (more robust) estimate of the spread than is the standard deviation.

- a measure of `skew`, which refers to how much asymmetry is present in the shape of the distribution. The measure is not the same as the *nonparametric skew* measure that we will usually prefer. The [Wikipedia page on skewness][<https://en.wikipedia.org/wiki/Skewness>] is very detailed.
- a measure of excess `kurtosis`, which refers to how outlier-prone, or heavy-tailed the shape of the distribution is, as compared to a Normal distribution.
- `se` = the standard error of the sample mean, equal to the sample `sd` divided by the square root of the sample size.

7.7.3 The Hmisc package's version of `describe`

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

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

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

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

lowest : 16.7 17.6 17.8 17.9 18.0, highest: 59.1 62.8 63.3 69.0 80.6
-----
SBP
  n   missing  distinct    Info     Mean     Gmd     .05     .10
  717        33       81  0.999  118.8   16.36   98.0   102.0
  .25       .50       .75    .90     .95
  108.0    118.0    127.0  137.0  144.2

lowest : 83 84 85 86 89, highest: 171 179 186 191 209
```

```

-----  

DBP  

      n   missing  distinct    Info     Mean     Gmd     .05     .10
    717       33       66  0.999  72.69  12.43    55     59
    .25       .50       .75    .90    .95
    66       73       80     86    91  

lowest :  0  25  41  42  44, highest: 104 105 106 107 108
-----  

Pulse  

      n   missing  distinct    Info     Mean     Gmd     .05     .10
    718       32       37  0.997  73.53  12.95    56     60
    .25       .50       .75    .90    .95
    66       72       80     88    94  

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

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

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

7.7.4 Other options

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

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

The `df_stats` function available when the `mosaic` package is loaded allows you to run `favstats` for multiple outcome variables simultaneously.

8 Summarizing Categories

8.1 Setup: Packages Used Here

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

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

theme_set(theme_bw())
```

8.2 Using the nh_adult750 data again

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

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

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

| Variable | Description | Levels | Type |
|------------|-----------------------------|--------|---------|
| Sex | sex of subject | 2 | binary |
| Race | subject's race | 6 | nominal |
| Education | subject's educational level | 5 | ordinal |
| PhysActive | Participates in sports? | 2 | binary |
| Smoke100 | Smoked 100+ cigarettes? | 2 | binary |

| Variable | Description | Levels | Type |
|--------------|--------------------|--------|---------|
| SleepTrouble | Trouble sleeping? | 2 | binary |
| HealthGen | Self-report health | 5 | ordinal |

8.3 The summary function for Categorical data

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

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

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

8.4 Tables to describe One Categorical Variable

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

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

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

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

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

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

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

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

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

| MaritalStatus | n | percent |
|---------------|-----|---------|
| Divorced | 78 | 10.4% |
| LivePartner | 70 | 9.3% |
| Married | 388 | 51.7% |
| NeverMarried | 179 | 23.9% |
| Separated | 19 | 2.5% |
| Widowed | 16 | 2.1% |
| Total | 750 | 100.0% |

8.5 Constructing Tables Well

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

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

8.5.1 Alabama First!

Which of these Tables is more useful to you?

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

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

or ...

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

| State | % Obese | 95% CI | Sample Size |
|-------|---------|-------------|-------------|
| Utah | 6.4 | (4.8 - 8.5) | 2,136 |

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

8.5.2 ALL is different and important

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

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

8.6 The Mode of a Categorical Variable

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

```

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

# A tibble: 6 x 2
  HealthGen count
  <fct>     <int>
1 Good       252
2 Vgood      197
3 Fair        104
4 <NA>        99
5 Excellent   84
6 Poor        14

```

8.7 describe in the Hmisc package

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

select(nh_750, Sex, Race, Education, PhysActive, Smoke100, SleepTrouble, HealthGen, MaritalStatus)

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

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

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

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

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

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

| | |
|------------|--------------|
| Value | College Grad |
| Frequency | 240 |
| Proportion | 0.320 |

PhysActive

| | | |
|-----|---------|----------|
| n | missing | distinct |
| 750 | 0 | 2 |

| | | |
|------------|-------|-------|
| Value | No | Yes |
| Frequency | 326 | 424 |
| Proportion | 0.435 | 0.565 |

Smoke100

| | | |
|-----|---------|----------|
| n | missing | distinct |
| 750 | 0 | 2 |

| | | |
|------------|-------|-------|
| Value | No | Yes |
| Frequency | 453 | 297 |
| Proportion | 0.604 | 0.396 |

SleepTrouble

| | | |
|-----|---------|----------|
| n | missing | distinct |
| 750 | 0 | 2 |

| | | |
|------------|------|------|
| Value | No | Yes |
| Frequency | 555 | 195 |
| Proportion | 0.74 | 0.26 |

HealthGen

| | | |
|-----|---------|----------|
| n | missing | distinct |
| 651 | 99 | 5 |

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

| | | | | | |
|------------|-----------|-------|-------|-------|-------|
| Value | Excellent | Vgood | Good | Fair | Poor |
| Frequency | 84 | 197 | 252 | 104 | 14 |
| Proportion | 0.129 | 0.303 | 0.387 | 0.160 | 0.022 |

MaritalStatus

| | | |
|-----|---------|----------|
| n | missing | distinct |
| 750 | 0 | 6 |

```

lowest : Divorced      LivePartner   Married       NeverMarried Separated
highest: LivePartner   Married       NeverMarried Separated     Widowed

Value          Divorced  LivePartner        Married  NeverMarried    Separated
Frequency      78        70                388      179            19
Proportion    0.104    0.093            0.517    0.239          0.025

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

8.8 Cross-Tabulations of Two Variables

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

```

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

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

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

```

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

| Education/SleepTrouble | No | Yes | Total |
|------------------------|----|-----|-------|
|------------------------|----|-----|-------|

| | | | | |
|----------------|--------------|-----|-----|-----|
| | 8th Grade | 40 | 10 | 50 |
| 9 - 11th Grade | 52 | 24 | 76 | |
| | High School | 102 | 41 | 143 |
| | Some College | 173 | 68 | 241 |
| | College Grad | 188 | 52 | 240 |
| | Total | 555 | 195 | 750 |

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

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

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

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

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

| Education/SleepTrouble | No | Yes | Total |
|------------------------|-------|-------|-------|
| 8th Grade | 7.2% | 5.1% | 6.7% |
| 9 - 11th Grade | 9.4% | 12.3% | 10.1% |
| High School | 18.4% | 21.0% | 19.1% |

```
Some College 31.2% 34.9% 32.1%
College Grad 33.9% 26.7% 32.0%
```

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

```
nh_750 |>
  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 | 5.3% | 1.3% | 6.7% |
| 9 - 11th Grade | 6.9% | 3.2% | 10.1% |
| High School | 13.6% | 5.5% | 19.1% |
| Some College | 23.1% | 9.1% | 32.1% |
| College Grad | 25.1% | 6.9% | 32.0% |
| Total | 74.0% | 26.0% | 100.0% |

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

```
nh_750 |>
  tabyl(MaritalStatus, HealthGen) |>
  adorn_totals(where = c("row")) |>
  adorn_percentages(denominator = "row") |>
  adorn_pct_formatting() |>
  adorn_ns(position = "front") |>
  adorn_title(placement = "combined") |>
  knitr::kable()
```

| MaritalStatus/HealthGen | Excellent | Vgood | Good | Fair | Poor | NA_ |
|-------------------------|---------------|----------------|----------------|---------------|----------|---------------|
| Divorced | 7 (9.0%) | 19 (24.4%) | 29 (37.2%) | 11 (14.1%) | 3 (3.8%) | 9 (11.5%) |
| LivePartner | 4 (5.7%) | 19 (27.1%) | 25 (35.7%) | 18 (25.7%) | 0 (0.0%) | 4 (5.7%) |
| Married | 46 (11.9%) | 101 (26.0%) | 130 (33.5%) | 41 (10.6%) | 6 (1.5%) | 64 (16.5%) |

| MaritalStatus/HealthGen | Excellent | Vgood | Good | Fair | Poor | NA_ |
|-------------------------|---------------|----------------|----------------|----------------|--------------|---------------|
| NeverMarried | 25 (14.0%) | 52 (29.1%) | 56 (31.3%) | 24 (13.4%) | 3 (1.7%) | 19 (10.6%) |
| Separated | 2 (10.5%) | 3 (15.8%) | 4 (21.1%) | 8 (42.1%) | 0 (0.0%) | 2 (10.5%) |
| Widowed | 0 (0.0%) | 3 (18.8%) | 8 (50.0%) | 2 (12.5%) | 2 (12.5%) | 1 (6.2%) |
| Total | 84 (11.2%) | 197 (26.3%) | 252 (33.6%) | 104 (13.9%) | 14 (1.9%) | 99 (13.2%) |

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

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

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

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

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

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

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

8.9 Cross-Classifying Three Categorical Variables

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

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

\$No

| Smoke100 | PhysActive | |
|----------|------------|-----|
| | No | Yes |
| No | 137 | 219 |
| Yes | 93 | 106 |

\$Yes

| Smoke100 | PhysActive | |
|----------|------------|-----|
| | No | Yes |
| No | 41 | 56 |
| Yes | 55 | 43 |

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

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

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

```
, , SleepTrouble = No
```

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

```
, , SleepTrouble = Yes
```

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

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

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

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

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

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

```
, , SleepTrouble = No
```

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

```
, , SleepTrouble = Yes
```

```
    PhysActive
```

| Smoke | | 100 | No | Yes |
|-------|--|-----|----|-----|
| | | No | 41 | 56 |
| | | Yes | 55 | 43 |

8.10 Gaining Control over Tables in R: the `gt` package

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

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

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

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

| Cross-Tabulation from nh_750 | | | | | | |
|--------------------------------------|-----------|-------|------|------|------|--|
| Physical Activity vs. Overall Health | | | | | | |
| PhysActive | Excellent | Vgood | Good | Fair | Poor | |
| No | 24 | 66 | 126 | 59 | 10 | |
| Yes | 60 | 131 | 126 | 45 | 4 | |

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

9 Missing Data and Single Imputation

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

9.1 Setup: Packages Used Here

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

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

theme_set(theme_bw())
```

We'll focus on tools from the `naniar` and `simputation` packages in this chapter. This chapter also requires that the `mosaic` package is loaded on your machine, but that package is not included with `library()` above.

9.2 A Simulated Example with 15 subjects

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

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

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

```

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

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

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

sbp_example

```

```

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

```

9.3 Identifying missingness with naniar functions

The `naniar` package has many useful functions.

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

```
n_miss(sbp_example)
```

```
[1] 7
```

2. How many variables have missing values, overall?

```
n_var_miss(sbp_example)
```

```
[1] 3
```

3. Which variables contain missing values?

```
miss_var_which(sbp_example)
```

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

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

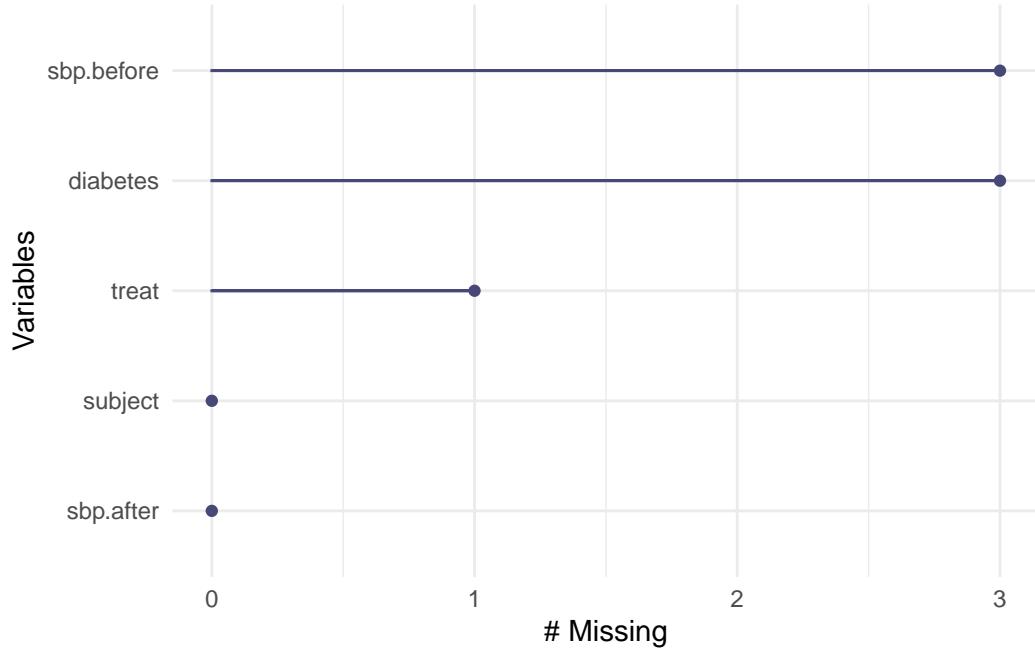
```
miss_var_summary(sbp_example)
```

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

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

5. Can we plot missingness, by variable?

```
gg_miss_var(sbp_example)
```



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

```
n_case_miss(sbp_example)
```

[1] 6

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

```
n_case_complete(sbp_example)
```

[1] 9

8. Can we tabulate missingness by case?

```
miss_case_table(sbp_example)
```

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

9. Which cases have missing values?

```
miss_case_summary(sbp_example)
```

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

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

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

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

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

9.4 Missing-data mechanisms

My source for this description of mechanisms is Chapter 25 of Gelman and Hill (2007), and that chapter is [available at this link](#).

1. **MCAR = Missingness completely at random.** A variable is missing completely at random if the probability of missingness is the same for all units, for example, if for each subject, we decide whether to collect the `diabetes` status by rolling a die and refusing to answer if a “6” shows up. If data are missing completely at random, then throwing out cases with missing data does not bias your inferences.
2. **Missingness that depends only on observed predictors.** A more general assumption, called **missing at random** or **MAR**, is that the probability a variable is missing depends only on available information. Here, we would have to be willing to assume that the probability of nonresponse to `diabetes` depends only on the other, fully recorded variables in the data. It is often reasonable to model this process as a logistic regression, where the outcome variable equals 1 for observed cases and 0 for missing. When an outcome variable is missing at random, it is acceptable to exclude the missing cases (that is, to treat them as NA), as long as the regression controls for all the variables that affect the probability of missingness.
3. **Missingness that depends on unobserved predictors.** Missingness is no longer “at random” if it depends on information that has not been recorded and this information also predicts the missing values. If a particular treatment causes discomfort, a patient is more likely to drop out of the study. This missingness is not at random (unless “discomfort” is measured and observed for all patients). If missingness is not at random, it must be explicitly modeled, or else you must accept some bias in your inferences.
4. **Missingness that depends on the missing value itself.** Finally, a particularly difficult situation arises when the probability of missingness depends on the (potentially missing) variable itself. For example, suppose that people with higher earnings are less likely to reveal them.

Essentially, situations 3 and 4 are referred to collectively as **non-random missingness**, and cause more trouble for us than 1 and 2.

9.5 Options for Dealing with Missingness

There are several available methods for dealing with missing data that are MCAR or MAR, but they basically boil down to:

- Complete Case (or Available Case) analyses
- Single Imputation
- Multiple Imputation

9.6 Complete Case (and Available Case) analyses

In **Complete Case** analyses, rows containing NA values are omitted from the data before analyses commence. This is the default approach for many statistical software packages, and may introduce unpredictable bias and fail to include some useful, often hard-won information.

- A complete case analysis can be appropriate when the number of missing observations is not large, and the missing pattern is either MCAR (missing completely at random) or MAR (missing at random.)
- Two problems arise with complete-case analysis:
 1. If the units with missing values differ systematically from the completely observed cases, this could bias the complete-case analysis.
 2. If many variables are included in a model, there may be very few complete cases, so that most of the data would be discarded for the sake of a straightforward analysis.
- A related approach is *available-case* analysis where different aspects of a problem are studied with different subsets of the data, perhaps identified on the basis of what is missing in them.

9.7 Single Imputation

In **single imputation** analyses, NA values are estimated/replaced *one time* with *one particular data value* for the purpose of obtaining more complete samples, at the expense of creating some potential bias in the eventual conclusions or obtaining slightly *less* accurate estimates than would be available if there were no missing values in the data.

- A single imputation can be just a replacement with the mean or median (for a quantity) or the mode (for a categorical variable.) However, such an approach, though easy to understand, underestimates variance and ignores the relationship of missing values to other variables.
- Single imputation can also be done using a variety of models to try to capture information about the NA values that are available in other variables within the data set.
- The **simputation** package can help us execute single imputations using a wide variety of techniques, within the pipe approach used by the **tidyverse**. Another approach I have used in the past is the **mice** package, which can also perform single imputations.

9.8 Multiple Imputation

Multiple imputation, where NA values are repeatedly estimated/replaced with multiple data values, for the purpose of obtaining more complete samples *and* capturing details of the

variation inherent in the fact that the data have missingness, so as to obtain *more* accurate estimates than are possible with single imputation.

- We'll postpone further discussion of multiple imputation to later in the semester.

9.9 Building a Complete Case Analysis

We can drop all of the missing values from a data set with `drop_na` or with `na.omit` or by filtering for `complete.cases`. Any of these approaches produces the same result - a new data set with 9 rows (after dropping the six subjects with any NA values) and 5 columns.

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

9.10 Single Imputation with the Mean or Mode

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

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

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

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

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

Here, suppose we decide to impute

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

- `treat` with its more common value, or mode (A)

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

si.1
```

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

9.11 Doing Single Imputation with `simputation`

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

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

Suppose we wanted to use:

- a robust linear model to predict `sbp.before` missing values, on the basis of `sbp.after` and `diabetes` status, and

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

```
set.seed(50001)

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

```
imp.2
```

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

Details on the many available methods in `simputation` are provided [in its manual](#). These include:

- `impute_cart` uses a Classification and Regression Tree approach for numerical or categorical data. There is also an `impute_rf` command which uses Random Forests for imputation.
- `impute_pmm` is one of several “hot deck” options for imputation, this one is predictive mean matching, which can be used with numeric data (only). Missing values are first imputed using a predictive model. Next, these predictions are replaced with the observed

values which are nearest to the prediction. Other imputation options in this group include random hot deck, sequential hot deck and k-nearest neighbor imputation.

- `impute_rlm` is one of several regression imputation methods, including linear models, robust linear models (which use what is called M-estimation to impute numerical variables) and lasso/elastic net/ridge regression models.

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

10 National Youth Fitness Survey

10.1 Setup: Packages Used Here

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

library(janitor)
library(knitr)
library(patchwork)
library(tidyverse)

theme_set(theme_bw())
```

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

10.2 What is the NHANES NYFS?

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

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

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

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

- Demographics
- Dietary
- Examination and

- Questionnaire

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

10.3 The Variables included in nnyfs

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

10.3.1 From the NNYFS Demographic Component

All of these come from the Y_DEMO file.

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

10.3.2 From the NNYFS Dietary Component

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

| In nnyfs | In Y_DR1TOT | Description |
|------------|-------------|---|
| respondent | DR1MNRSR | 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) |

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

10.3.3 From the NNYFS Examination Component

From the Y_BMX file of Body Measures:

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

From the Y_PLX file of Plank test results:

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

10.3.4 From the NNYFS Questionnaire Component

From the Y_PAQ file of Physical Activity questions:

| In nnyfs | In Y_PAQ | Description |
|--------------------|----------|---|
| active_days | PAQ706 | Days physically active (≥ 60 min.) in past week |
| tv_hours | PAQ710 | Average hours watching TV/videos past 30d |
| computer_hours | PAQ715 | Average hours on computer past 30d |
| physical_last_week | PAQ722 | Any physical activity outside of school past week |

| In nnyfs | In Y_PAQ | Description |
|--------------|----------|----------------------------------|
| enjoy_recess | PAQ750 | Enjoy participating in PE/recess |

From the Y_DBQ file of Diet Behavior and Nutrition questions:

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

From the Y_HIQ file of Health Insurance questions:

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

From the Y_HUQ file of Access to Care questions:

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

From the Y_MCQ file of Medical Conditions questions:

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

From the Y_RXQ_RX file of Prescription Medication questions:

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

10.4 Looking over A Few Variables

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

```
nnyfs <- readRDS("data/nnyfs.Rds")  
  
## size of the tibble  
dim(nnyfs)
```

```
[1] 1518 45
```

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

```
nnyfs # this is a tibble, has some nice features in a print-out like this  
  
# A tibble: 1,518 x 45  
# ... with 1,508 more rows, 35 more variables: respondent <fct>,  
#   salt_used <fct>, energy <dbl>, protein <dbl>, sugar <dbl>, fat <dbl>,  
#   diet_yesterday <fct>, water <dbl>, plank_time <dbl>, height <dbl>,  
#   weight <dbl>, bmi <dbl>, bmi_cat <fct>, arm_length <dbl>, waist <dbl>,  
#   arm_circ <dbl>, calf_circ <dbl>, calf_skinfold <dbl>,  
#   triceps_skinfold <dbl>, subscapular_skinfold <dbl>, active_days <dbl>,  
#   tv_hours <dbl>, computer_hours <dbl>, physical_last_week <fct>, ...  
# i Use `print(n = ...)` to see more rows, and `colnames()` to see all variable names
```

Tibbles are a modern reimagining of the main way in which people have stored data in R, called a data frame. Tibbles were developed to keep what time has proven to be effective, and throwing out what is not. We can learn something about the structure of the tibble from such functions as `str` or `glimpse`.

```
str(nnyfs)
```

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

```

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

```

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

10.4.1 SEQN

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

10.4.2 sex

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

```

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

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

```

10.4.3 age_child

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

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

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

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.

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

```
age_child  
Min.   : 3.000  
1st Qu.: 6.000  
Median : 9.000  
Mean   : 9.033  
3rd Qu.:12.000  
Max.   :15.000
```

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

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

We could get the standard deviation and a count of missing and non-missing observations with `favstats` from the `mosaic` package.

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

Registered S3 method overwritten by 'mosaic':

```
method                  from  
fortify.SpatialPolygonsDataFrame ggplot2
```

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

10.4.4 race_eth

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

```
nnyfs |> tabyl(race_eth) |>  
  adorn_pct_formatting() |>  
  knitr::kable()
```

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

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

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

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

10.4.5 income_pov

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

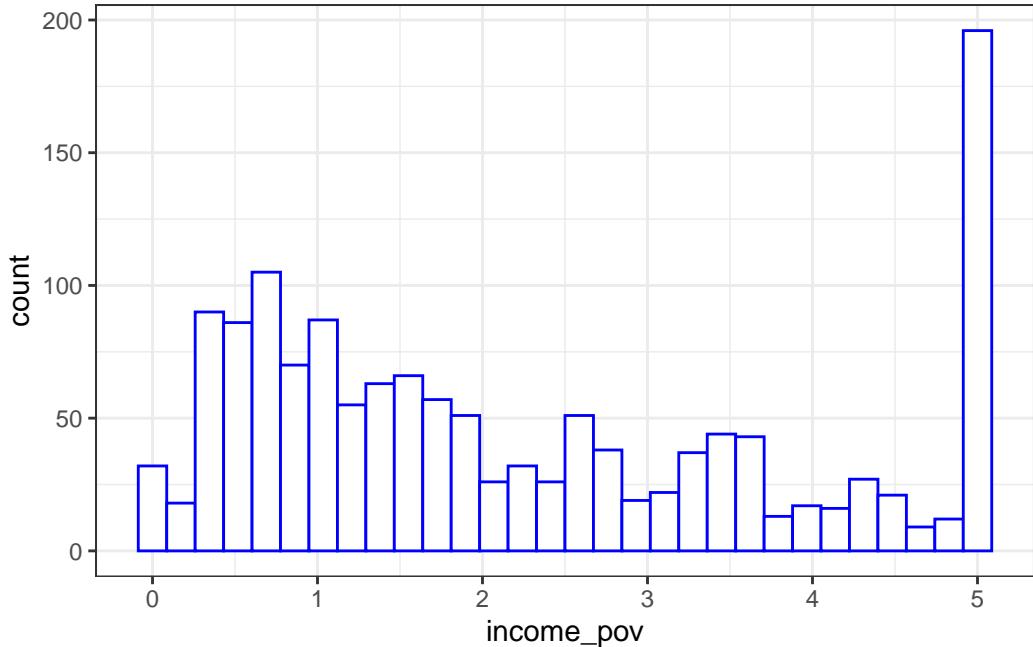
```
nnysf |> select(income_pov) |> summary()
```

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

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

```
ggplot(nnysf, 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?

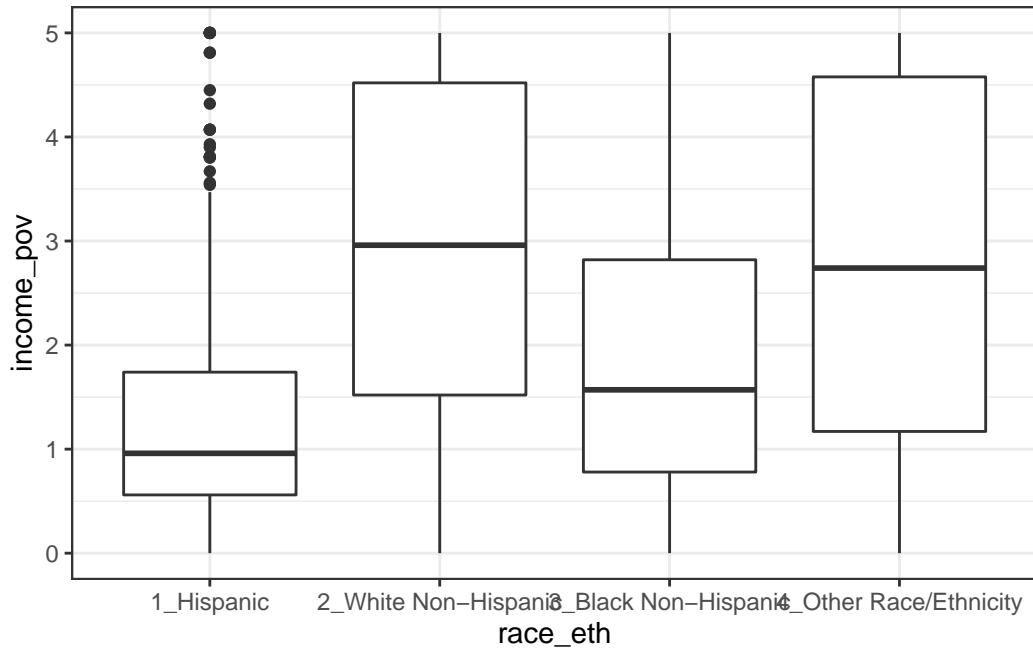
```
mosaic::favstats(income_pov ~ race_eth, data = nnyfs) |>
  kable(digits = 1)
```

| race_eth | min | Q1 | median | Q3 | max | mean | sd | n | missing |
|----------------------|-----|-----|--------|-----|-----|------|-----|-----|---------|
| 1_Hispanic | 0 | 0.6 | 1.0 | 1.7 | 5 | 1.3 | 1.1 | 409 | 41 |
| 2_White Non-Hispanic | 0 | 1.5 | 3.0 | 4.5 | 5 | 2.9 | 1.6 | 588 | 22 |
| 3_Black Non-Hispanic | 0 | 0.8 | 1.6 | 2.8 | 5 | 2.0 | 1.5 | 328 | 10 |
| 4_Other | 0 | 1.2 | 2.7 | 4.6 | 5 | 2.8 | 1.7 | 104 | 16 |
| Race/Ethnicity | | | | | | | | | |

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).



10.4.6 bmi

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

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

```
bmi
Min.   :11.90
1st Qu.:15.90
Median  :18.10
Mean    :19.63
3rd Qu.:21.90
Max.   :48.30
NA's   :4
```

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

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

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

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

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

10.4.7 `bmi_cat`

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

```
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 ($\text{BMI} < 5\text{th percentile}$)
- Normal weight ($\text{BMI } 5\text{th to } < 85\text{th percentile}$)
- Overweight ($\text{BMI } 85\text{th to } < 95\text{th percentile}$)
- Obese ($\text{BMI} \geq 95\text{th percentile}$)

Note how I've used labels in the `bmi_cat` variable that include a number at the start so that the table results are sorted in a rational way. R sorts tables alphabetically, in general. We'll use the `forcats` package to work with categorical variables that we store as *factors* eventually, but for now, we'll keep things relatively simple.

Note that the `bmi_cat` data don't completely separate out the raw `bmi` data, because the calculation of percentiles requires different tables for each combination of `age` and `sex`.

```
mosaic::favstats(bmi ~ bmi_cat, data = nnyfs) |>
  kable(digits = 1)
```

| bmi_cat | min | Q1 | median | Q3 | max | mean | sd | n | missing |
|---------------|------|------|--------|------|------|------|-----|-----|---------|
| 1_Underweight | 11.9 | 13.4 | 13.7 | 15.0 | 16.5 | 14.1 | 1.1 | 41 | 0 |
| 2_Normal | 13.5 | 15.4 | 16.5 | 18.7 | 24.0 | 17.2 | 2.3 | 920 | 0 |
| 3_Overweight | 16.9 | 18.3 | 21.4 | 23.4 | 27.9 | 21.2 | 2.9 | 258 | 0 |
| 4_Obese | 17.9 | 22.3 | 26.2 | 30.2 | 48.3 | 26.7 | 5.7 | 295 | 0 |

10.4.8 waist

Let's also look briefly at `waist`, which is the circumference of the child's waist, in centimeters. Again, this is a numeric variable, so perhaps we'll stick to the simple summary, rather than obtaining a table of observed values.

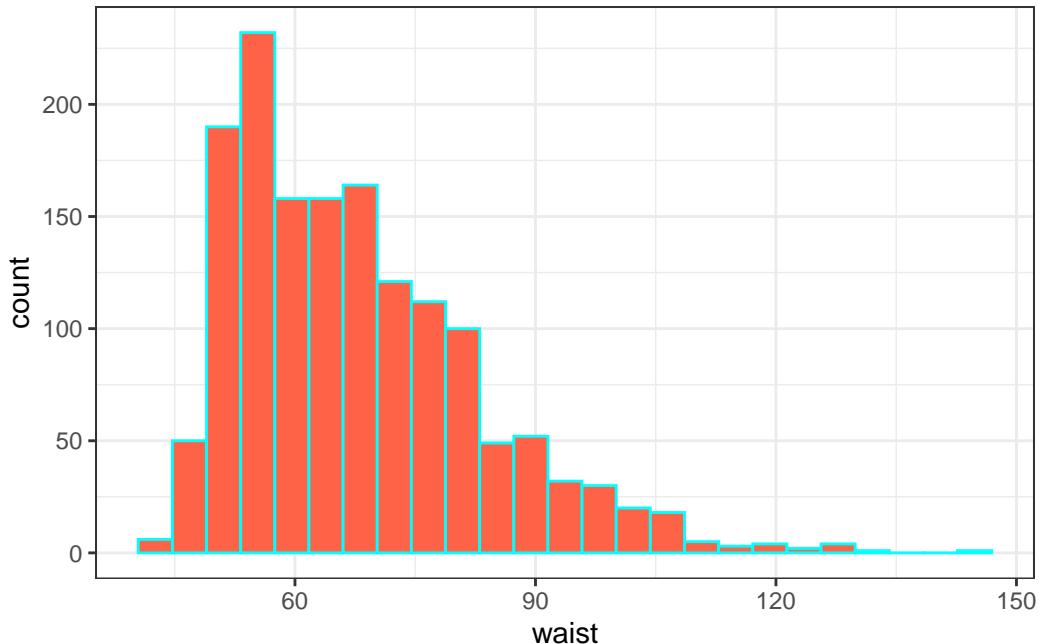
```
mosaic::favstats(~ waist, data = nnyfs)
```

| | min | Q1 | median | Q3 | max | mean | sd | n | missing |
|--|------|------|--------|------|-------|----------|----------|------|---------|
| | 42.5 | 55.6 | 64.8 | 76.6 | 144.7 | 67.70536 | 15.19809 | 1512 | 6 |

Here's a histogram of the waist circumference data.

```
ggplot(nnyfs, aes(x = waist)) +
  geom_histogram(bins = 25, fill = "tomato", color = "cyan")
```

Warning: Removed 6 rows containing non-finite values (stat_bin).



10.4.9 triceps_skinfold

The last variable I'll look at for now is `triceps_skinfold`, which is measured in millimeters. This is one of several common locations used for the assessment of body fat using skinfold calipers, and is a frequent part of growth assessments in children. Again, this is a numeric variable according to R.

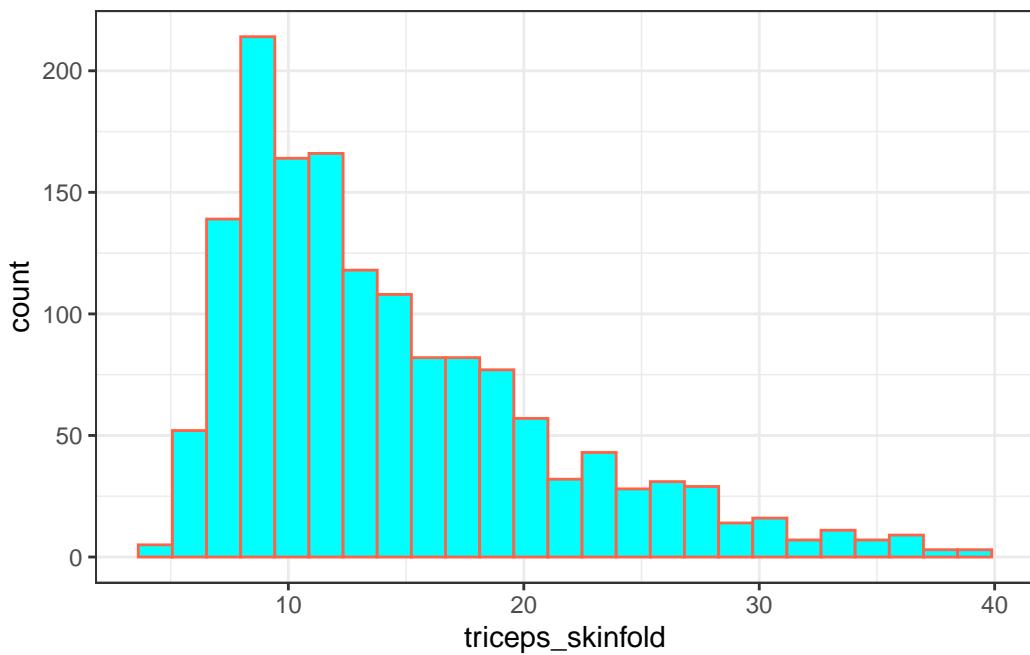
```
mosaic::favstats(~ triceps_skinfold, data = nnyfs)
```

| min | Q1 | median | Q3 | max | mean | sd | n | missing |
|-----|-----|--------|----|------|----------|----------|------|---------|
| 4 | 9.1 | 12.4 | 18 | 38.8 | 14.35725 | 6.758825 | 1497 | 21 |

And here's a histogram of the triceps skinfold data, with the fill and color flipped from what we saw in the plot of the waist circumference data a moment ago.

```
ggplot(nnyfs, aes(x = triceps_skinfold)) +
  geom_histogram(bins = 25, fill = "cyan", color = "tomato")
```

Warning: Removed 21 rows containing non-finite values (stat_bin).



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

10.5 Additional Numeric Summaries

10.5.1 The Five Number Summary, Quantiles and IQR

The **five number summary** is most famous when used to form a box plot - it's the minimum, 25th percentile, median, 75th percentile and maximum. For numerical and integer variables,

the **summary** function produces the five number summary, plus the mean, and a count of any missing values (NA's).

```
nnyfs |>
  select(waist, energy, sugar) |>
  summary()
```

| | waist | energy | sugar |
|---------|---------|--------------|----------------|
| Min. | : 42.50 | Min. : 257 | Min. : 1.00 |
| 1st Qu. | : 55.60 | 1st Qu.:1368 | 1st Qu.: 82.66 |
| Median | : 64.80 | Median :1794 | Median :116.92 |
| Mean | : 67.71 | Mean :1877 | Mean :124.32 |
| 3rd Qu. | : 76.60 | 3rd Qu.:2306 | 3rd Qu.:157.05 |
| Max. | :144.70 | Max. :5265 | Max. :405.49 |
| NA's | :6 | | |

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

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

```
[1] 257.0 1367.0 1794.5 2306.0 5265.0
```

- As mentioned in @ref(rangeandiqr), the **inter-quartile range**, or IQR, is sometimes used as a competitor for the standard deviation. It's the difference between the 75th percentile and the 25th percentile. The 25th percentile, median, and 75th percentile are referred to as the quartiles of the data set, because, together, they split the data into quarters.

```
IQR(nnyfs$energy)
```

```
[1] 938.5
```

We can obtain **quantiles** (percentiles) as we like - here, I'm asking for the 1st and 99th:

```
quantile(nnyfs$energy, probs=c(0.01, 0.99))
```

```
1%      99%
566.85 4051.75
```

10.6 Additional Summaries from favstats

If we're focusing on a single variable, the `favstats` function in the `mosaic` package can be very helpful. Rather than calling up the entire `mosaic` library here, I'll just specify the function within the library.

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

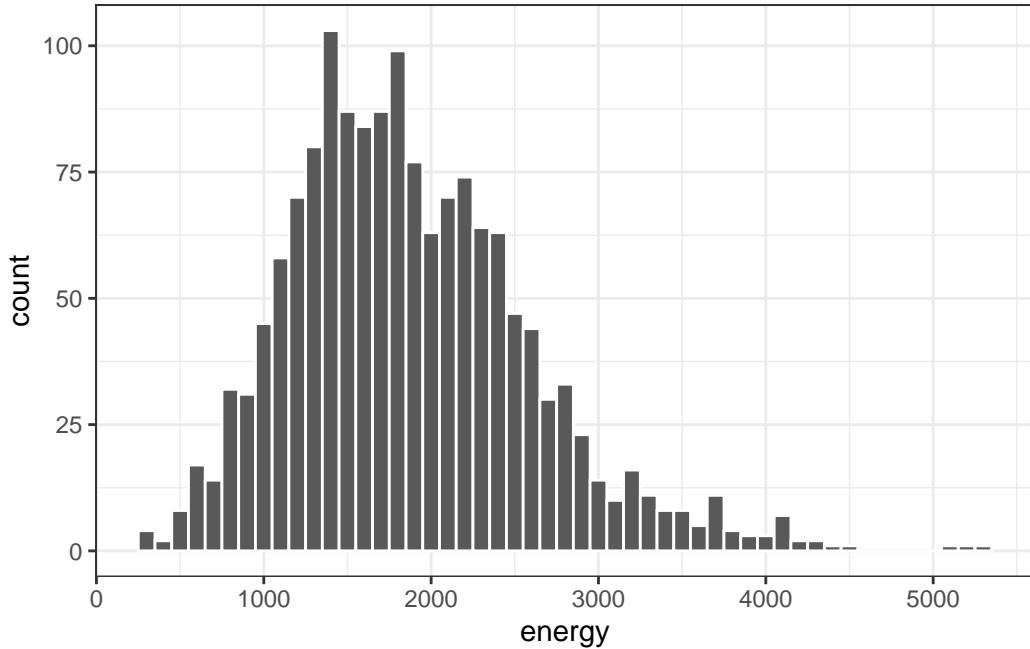
min      Q1 median      Q3   max      mean        sd      n missing
257 1367.5 1794.5 2306 5265 1877.157 722.3537 1518          0
```

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

10.7 The Histogram

Obtaining a basic **histogram** of, for example, the energy (kilocalories consumed) in the `nnyfs` data is pretty straightforward.

```
ggplot(data = nnyfs, aes(x = energy)) +
  geom_histogram(binwidth = 100, col = "white")
```



10.7.1 Freedman-Diaconis Rule to select bin width

If we like, we can suggest a particular number of cells for the histogram, instead of accepting the defaults. In this case, we have $n = 1518$ observations. The **Freedman-Diaconis rule** can be helpful here. That rule suggests that we set the bin-width to

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

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

For the `energy` data in the `nnyfs` tibble, we have

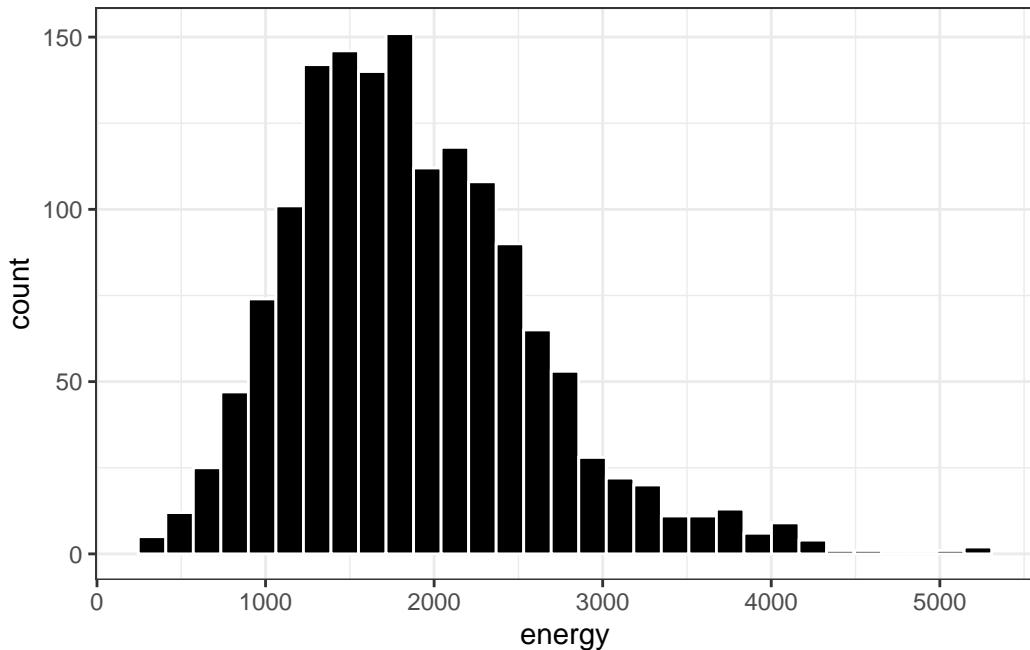
- IQR of 938.5, $n = 1518$ and range = 5008
- Thus, by the Freedman-Diaconis rule, the optimal binwidth h is 163.3203676, or, realistically, 163.
- And so the number of bins would be 30.6636586, or, realistically 31.

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

```

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

```



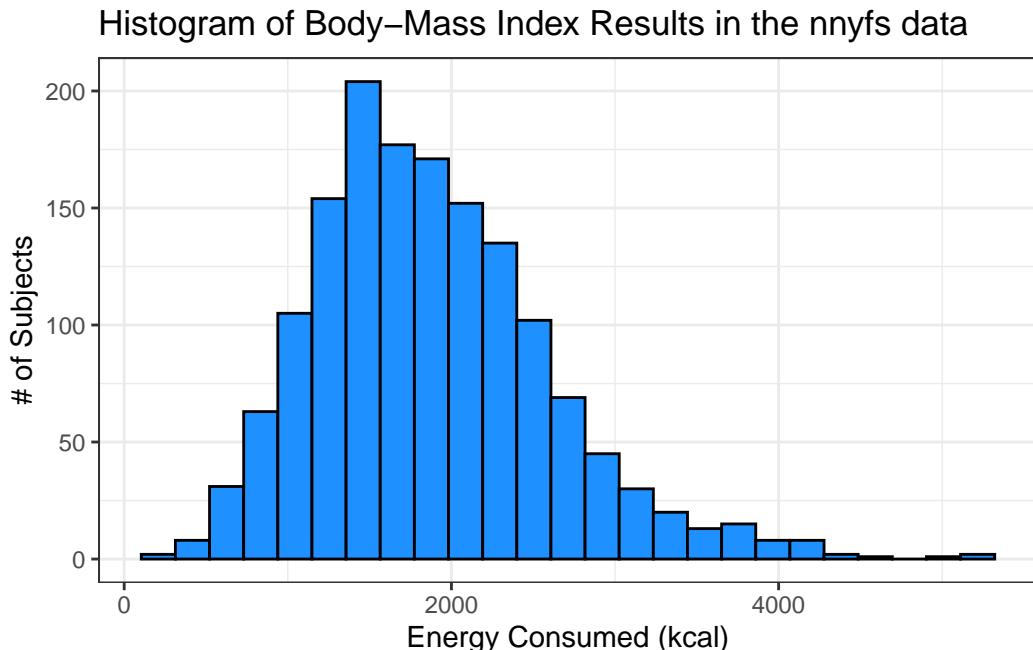
This is a nice start, but it is by no means a finished graph.

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

```

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

```



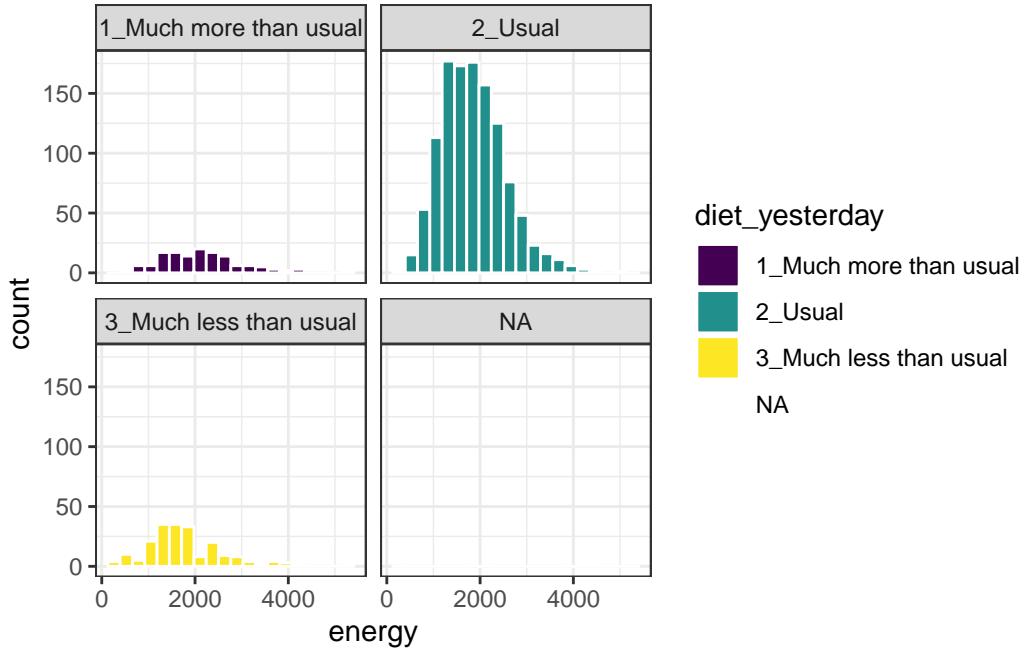
10.7.2 A Note on Colors

The simplest way to specify a color is with its name, enclosed in parentheses. My favorite list of R colors is <http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf>. In a pinch, you can usually find it by googling **Colors in R**. You can also type `colors()` in the R console to obtain a list of the names of the same 657 colors.

When using colors to make comparisons, you may be interested in using a scale that has some nice properties. The [viridis package vignette](#) describes four color scales (viridis, magma, plasma and inferno) that are designed to be colorful, robust to colorblindness and gray scale printing, and perceptually uniform, which means (as the package authors describe it) that values close to each other have similar-appearing colors and values far away from each other have more different-appearing colors, consistently across the range of values. We can apply these colors with special functions within `ggplot`.

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

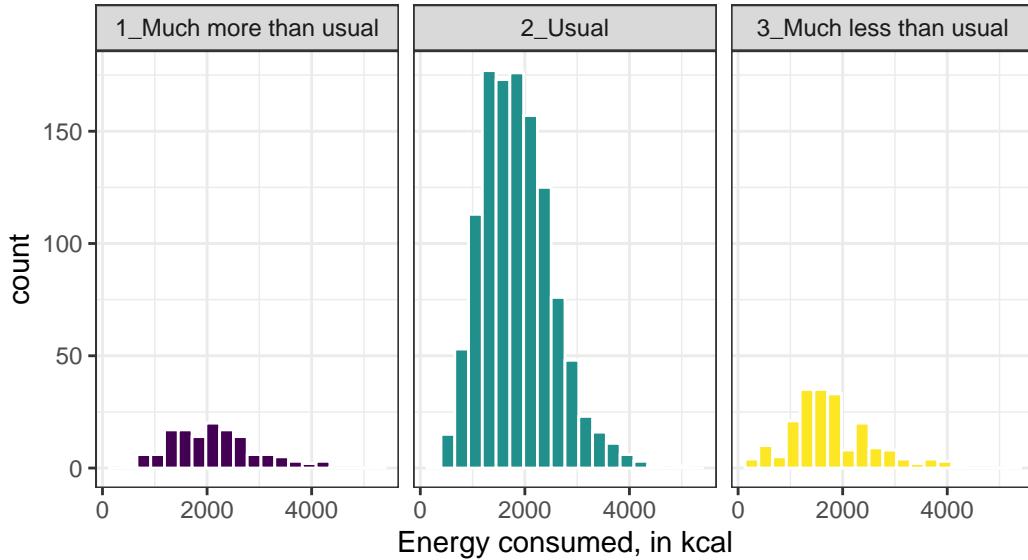
```
ggplot(data = nnyfs, aes(x = energy, fill = diet_yesterday)) +
  geom_histogram(bins = 20, col = "white") +
  scale_fill_viridis_d() +
  facet_wrap(~ diet_yesterday)
```



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

```
nnyfs |> filter(!is.na(energy), !is.na(diet_yesterday)) %>%
  ggplot(data = ., aes(x = energy, fill = diet_yesterday)) +
  geom_histogram(bins = 20, col = "white") +
  scale_fill_viridis_d() +
  guides(fill = "none") +
  facet_wrap(~ diet_yesterday) +
  labs(x = "Energy consumed, in kcal",
       title = "Energy Consumption and How Typical Was Yesterday's Eating",
       subtitle = "NHANES National Youth Fitness Survey, no survey weighting")
```

Energy Consumption and How Typical Was Yesterday's Eating NHANES National Youth Fitness Survey, no survey weighting

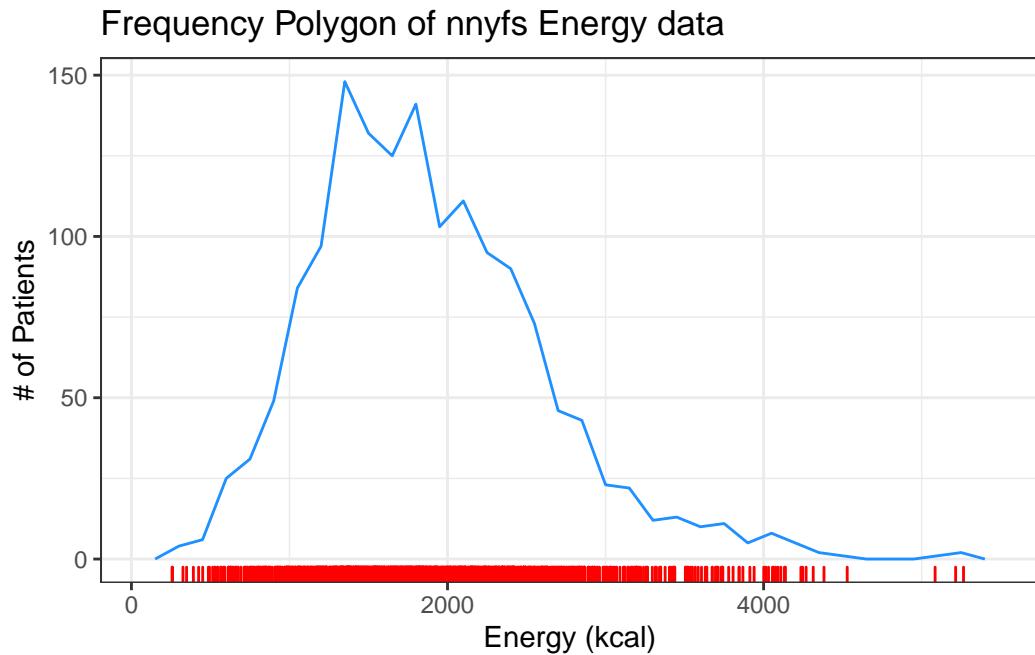


Note the use of the `%>%` pipe here. I need to write more about this.

10.8 The Frequency Polygon

As we've seen, we can also plot the distribution of a single continuous variable using the `freqpoly` geom. We can also add a *rug plot*, which places a small vertical line on the horizontal axis everywhere where an observation appears in the data.

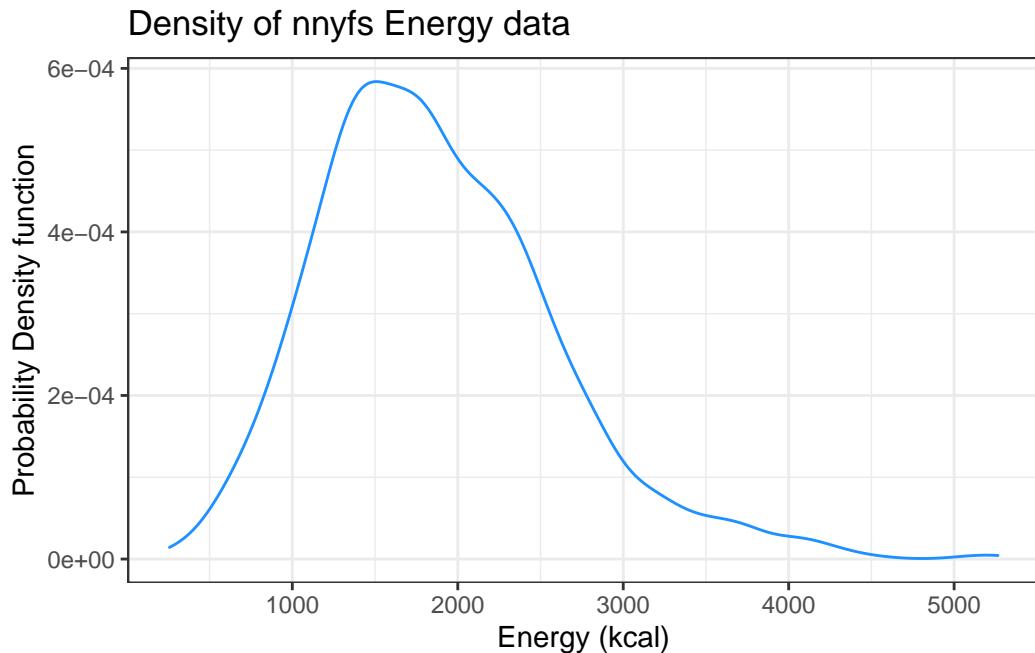
```
ggplot(data = nnyfs, aes(x = energy)) +  
  geom_freqpoly(binwidth = 150, color = "dodgerblue") +  
  geom_rug(color = "red") +  
  labs(title = "Frequency Polygon of nnyfs Energy data",  
       x = "Energy (kcal)", y = "# of Patients")
```



10.9 Plotting the Probability Density Function

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

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



So, what's a density function?

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

10.10 The Boxplot

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

10.10.1 Drawing a Boxplot for One Variable in ggplot2

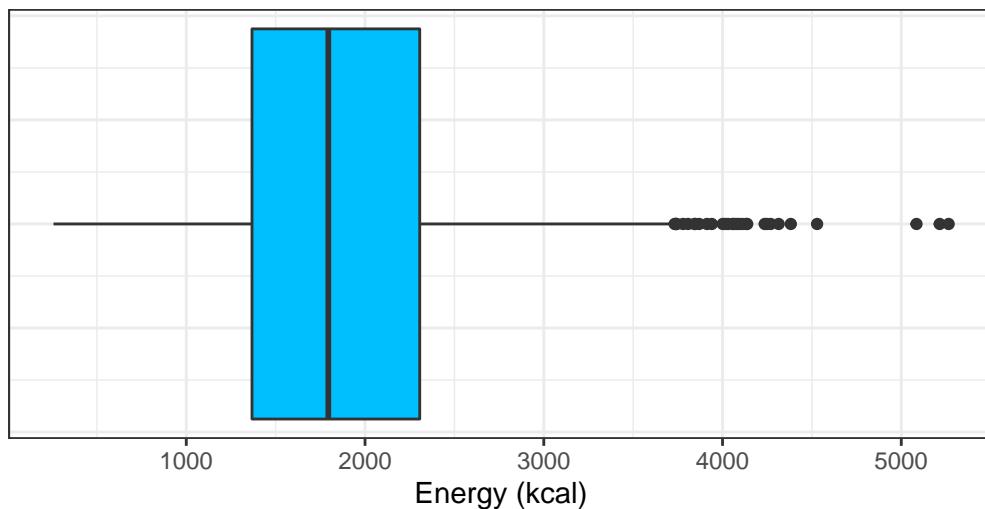
The `ggplot2` library easily handles comparison boxplots for multiple distributions, as we'll see in a moment. However, building a boxplot for a single distribution requires a little trickiness.

```

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

```

Boxplot of Energy for kids in the NNYFS



10.10.2 About the Boxplot

The boxplot is another John Tukey invention.

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

The boxplot's **whiskers** that are drawn from the first and third quartiles (i.e. the 25th and 75th percentiles) out to the most extreme points in the data that do not meet the standard

of “candidate outliers.” An outlier is simply a point that is far away from the center of the data - which may be due to any number of reasons, and generally indicates a need for further investigation.

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

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

So for these `energy` data,

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

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

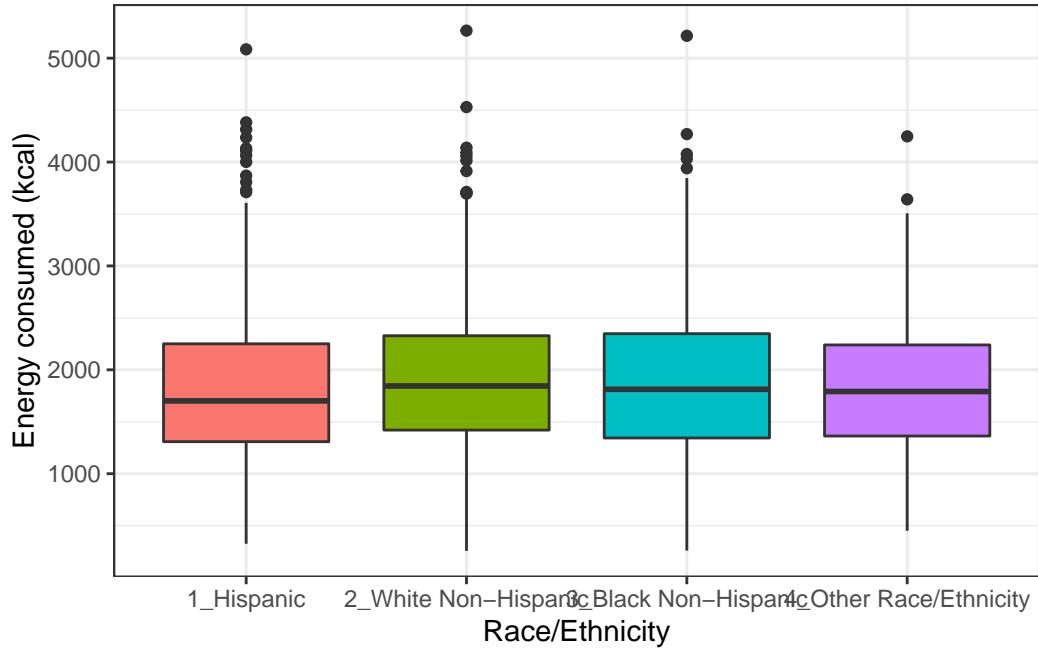
- This plot is indicating clearly that there is some asymmetry (skew) in the data, specifically right skew.
- The standard R uses is to indicate as outliers any points that are more than 1.5 inter-quartile ranges away from the edges of the box.

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

10.11 A Simple Comparison Boxplot

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

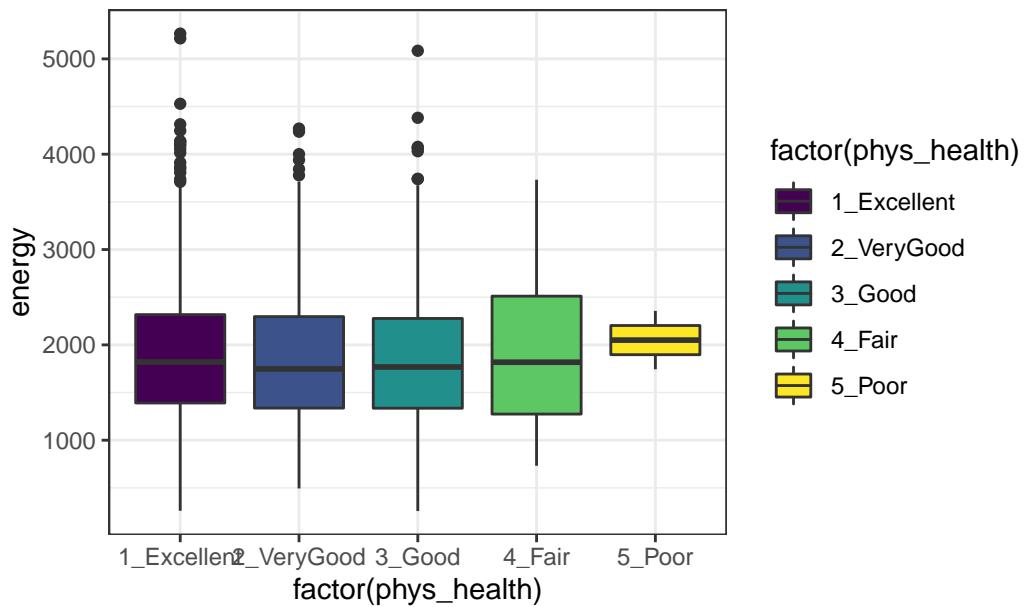
```
ggplot(nnyfs, aes(x = factor(race_eth), y = energy, fill=factor(race_eth))) +
  geom_boxplot() +
  guides(fill = "none") +
  labs(y = "Energy consumed (kcal)", x = "Race/Ethnicity")
```



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

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

Energy by Self-Reported Physical Health, in nnyfs data

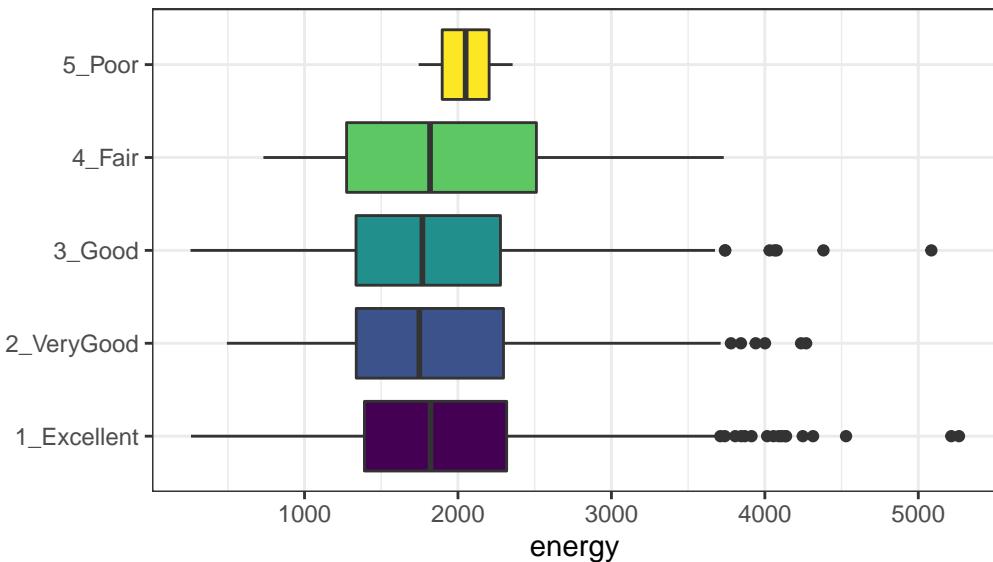


As a graph, that's not bad, but what if we want to improve it further?

Let's turn the boxes in the horizontal direction, and get rid of the perhaps unnecessary `phys_health` labels.

```
ggplot(nnyfs, aes(x = factor(phys_health), y = energy, fill = factor(phys_health))) +  
  geom_boxplot() +  
  scale_fill_viridis_d() +  
  coord_flip() +  
  guides(fill = "none") +  
  labs(title = "Energy Consumed by Self-Reported Physical Health",  
       subtitle = "NHANES National Youth Fitness Survey, unweighted",  
       x = "")
```

Energy Consumed by Self-Reported Physical Health
NHANES National Youth Fitness Survey, unweighted



10.12 Using describe in the psych library

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

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

```
vars     n      mean       sd median trimmed      mad min   max range skew kurtosis
X1      1 1518 1877.16 722.35 1794.5 1827.1 678.29 257 5265 5008  0.8      1.13
      se
X1 18.54
```

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

- `n` = the sample size
- `mean` = the sample mean
- `sd` = the sample standard deviation
- `median` = the median, or 50th percentile
- `trimmed` = mean of the middle 80% of the data
- `mad` = median absolute deviation
- `min` = minimum value in the sample

- `max` = maximum value in the sample
- `range` = max - min
- `skew` = skewness measure, described below (indicates degree of asymmetry)
- `kurtosis` = kurtosis measure, described below (indicates heaviness of tails, degree of outlier-proneness)
- `se` = standard error of the sample mean = `sd` / square root of sample size, useful in inference

10.12.1 The Trimmed Mean

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

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

```
[1] 1827.1
```

10.12.2 The Median Absolute Deviation

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

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

```
[1] 678.2895
```

10.13 Assessing Skew

A relatively common idea is to assess **skewness**, several measures of which are available. Many models assume a Normal distribution, where, among other things, the data are symmetric around the mean.

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

10.13.1 Non-parametric Skewness

A simpler measure of skew, sometimes called the **nonparametric skew** and closely related to Pearson's notion of median skewness, falls between -1 and +1 for any distribution. It is just the difference between the mean and the median, divided by the standard deviation.

- Values greater than +0.2 are sometimes taken to indicate fairly substantial right skew, while values below -0.2 indicate fairly substantial left skew.

```
(mean(nnyfs$energy) - median(nnyfs$energy))/sd(nnyfs$energy)
```

```
[1] 0.114427
```

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

10.14 Assessing Kurtosis (Heavy-Tailedness)

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

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

Here's a table:

| Fewer outliers than a Normal | Approximately Normal | More outliers than a Normal |
|---|---|---|
| Light-tailed <i>platykurtic</i> (kurtosis < 0) | “Normalish” <i>mesokurtic</i> (kurtosis = 0) | Heavy-tailed <i>leptokurtic</i> (kurtosis > 0) |

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

```
[1] 1.130539
```

Note that the `kurtosi()` function is strangely named, and is part of the `psych` package.

10.14.1 The Standard Error of the Sample Mean

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

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

```
[1] 18.54018
```

10.15 The `describe` function in the `Hmisc` package

The `Hmisc` package has lots of useful functions. It's named for its main developer, Frank Harrell. The `describe` function in `Hmisc` knows enough to separate numerical from categorical variables, and give you separate (and detailed) summaries for each.

- For a categorical variable, it provides counts of total observations (n), the number of missing values, and the number of unique categories, along with counts and percentages falling in each category.
- For a numerical variable, it provides:
 - counts of total observations (n), the number of missing values, and the number of unique values
 - an Info value for the data, which indicates how continuous the variable is (a score of 1 is generally indicative of a completely continuous variable with no ties, while scores near 0 indicate lots of ties, and very few unique values)
 - the sample Mean

- Gini's mean difference, which is a robust measure of spread, with larger values indicating greater dispersion in the data. It is defined as the mean absolute difference between any pairs of observations.
- many sample percentiles (quantiles) of the data, specifically (5, 10, 25, 50, 75, 90, 95, 99)
- either a complete table of all observed values, with counts and percentages (if there are a modest number of unique values), or
- a table of the five smallest and five largest values in the data set, which is useful for range checking

```
nnyfs |>
  select(waist, energy, bmi) |>
  Hmisc::describe()
```

```
select(nnyfs, waist, energy, bmi)
```

```
3 Variables      1518 Observations
```

waist

| | n | missing | distinct | Info | Mean | Gmd | .05 | .10 |
|-------|---|---------|----------|-------|-------|------|-------|-------|
| 1512 | | 6 | 510 | 1 | 67.71 | 16.6 | 49.40 | 51.40 |
| .25 | | .50 | .75 | .90 | .95 | | | |
| 55.60 | | 64.80 | 76.60 | 88.70 | 96.84 | | | |

```
lowest : 42.5 43.4 44.1 44.4 44.5, highest: 125.8 126.0 127.0 132.3 144.7
```

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](#)

10.16 Summarizing data within subgroups

Suppose we want to understand how the subjects whose diet involved consuming much more than usual yesterday compare to those who consumer their usual amount, or to those who consumed much less than usual, in terms of the energy they consumed, as well as the protein. We might start by looking at the medians and means.

```
nnyfs |>
  group_by(diet_yesterday) |>
  select(diet_yesterday, energy, protein) |>
  summarise_all(list(median = median, mean = mean))

# A tibble: 4 x 5
  diet_yesterday      energy_median protein_median energy_mean protein_mean
  <fct>                <dbl>        <dbl>       <dbl>        <dbl>
1 1_Much more than usual     2098       69.4      2150.       75.1
2 2_Usual                  1794       61.3      1858.       67.0
3 3_Much less than usual    1643       53.9      1779.       60.1
4 <NA>                     4348       155.      4348        155.
```

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

```
nnyfs |>
  filter(complete.cases(diet_yesterday)) |>
  group_by(diet_yesterday) |>
  select(diet_yesterday, energy, protein, sugar, water) |>
  summarise_all(list(median))

# A tibble: 3 x 5
  diet_yesterday      energy protein sugar water
  <fct>                <dbl>   <dbl> <dbl> <dbl>
1 1_Much more than usual     2098    69.4  137.  500
2 2_Usual                  1794    61.3  114.  385.
3 3_Much less than usual    1643    53.9  115.  311.
```

It looks like the children in the “Much more than usual” category consumed more energy, protein, sugar and water than the children in the other two categories. Let’s draw a picture of this.

```
temp_dat <- nnyfs |>
  filter(complete.cases(diet_yesterday)) |>
  mutate(diet_yesterday = fct_recode(diet_yesterday,
    "Much more" = "1_Much more than usual",
    "Usual diet" = "2_Usual",
    "Much less" = "3_Much less than usual"))

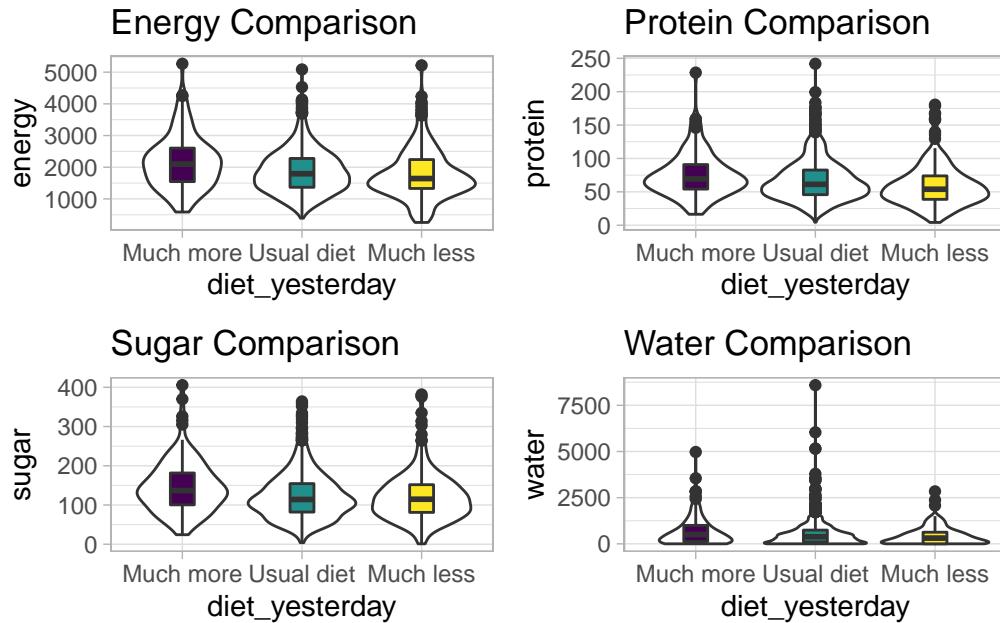
p1 <- ggplot(temp_dat, aes(x = diet_yesterday, y = energy)) +
  geom_violin() +
  geom_boxplot(aes(fill = diet_yesterday), width = 0.2) +
  theme_light() +
  scale_fill_viridis_d() +
  guides(fill = "none") +
  labs(title = "Energy Comparison")

p2 <- ggplot(temp_dat, aes(x = diet_yesterday, y = protein)) +
  geom_violin() +
  geom_boxplot(aes(fill = diet_yesterday), width = 0.2) +
  theme_light() +
  scale_fill_viridis_d() +
  guides(fill = "none") +
  labs(title = "Protein Comparison")

p3 <- ggplot(temp_dat, aes(x = diet_yesterday, y = sugar)) +
  geom_violin() +
  geom_boxplot(aes(fill = diet_yesterday), width = 0.2) +
  theme_light() +
  scale_fill_viridis_d() +
  guides(fill = "none") +
  labs(title = "Sugar Comparison")

p4 <- ggplot(temp_dat, aes(x = diet_yesterday, y = water)) +
  geom_violin() +
  geom_boxplot(aes(fill = diet_yesterday), width = 0.2) +
  theme_light() +
  scale_fill_viridis_d() +
  guides(fill = "none") +
  labs(title = "Water Comparison")
```

p1 + p2 + p3 + p4



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

10.17 Another Example

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

```
nyfs |>
  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
<fct>    <dbl>   <dbl>  <dbl>   <dbl>
```

```

<fct>      <dbl> <dbl> <dbl> <dbl>
1 1_Underweight 55.2 7.58 54.5 0.09
2 2_Normal      NA   NA    NA   NA
3 3_Overweight   72.3 11.9 74   -0.14
4 4_Obese        NA   NA    NA   NA
5 <NA>          NA   NA    NA   NA

```

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

```

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

# A tibble: 4 x 6
  bmi_cat     count   mean     sd median skew_1
<fct>       <int> <dbl> <dbl> <dbl>   <dbl>
1 1_Underweight 41    55.2  7.58  54.5  0.09
2 2_Normal      917   61.2  9.35  59.5  0.19
3 3_Overweight   258   72.3  11.9   74   -0.14
4 4_Obese        294   85.6  17.1   86.8 -0.07

```

Or, we could use something like `favstats` from the `mosaic` package, which automatically accounts for missing data, and omits it when calculating summary statistics within each group.

```

mosaic::favstats(waist ~ bmi_cat, data = nnyfs) |>
  kable(digits = 1)

```

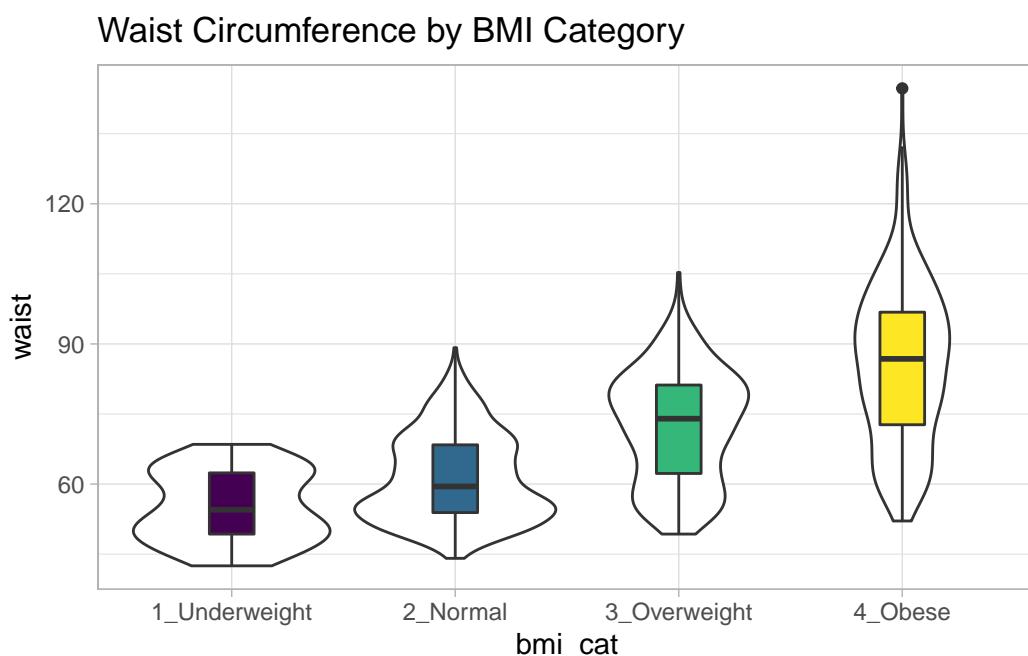
| bmi_cat | min | Q1 | median | Q3 | max | mean | sd | n | missing |
|---------------|------|------|--------|------|-------|------|------|-----|---------|
| 1_Underweight | 42.5 | 49.3 | 54.5 | 62.4 | 68.5 | 55.2 | 7.6 | 41 | 0 |
| 2_Normal | 44.1 | 53.9 | 59.5 | 68.4 | 89.2 | 61.2 | 9.4 | 917 | 3 |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

While patients in the heavier groups generally had higher waist circumferences, the standard deviations suggest there may be some meaningful overlap. Let's draw the picture, in this case a comparison boxplot accompanying a violin plot.

```

nnyfs |>
  filter(complete.cases(bmi_cat, waist)) %>%
  ggplot(., aes(x = bmi_cat, y = waist)) +
  geom_violin() +
  geom_boxplot(aes(fill = bmi_cat), width = 0.2) +
  theme_light() +
  scale_fill_viridis_d() +
  guides(fill = "none") +
  labs(title = "Waist Circumference by BMI Category")

```



Note the use of the `%>%` pipe here. I need to write more about this.

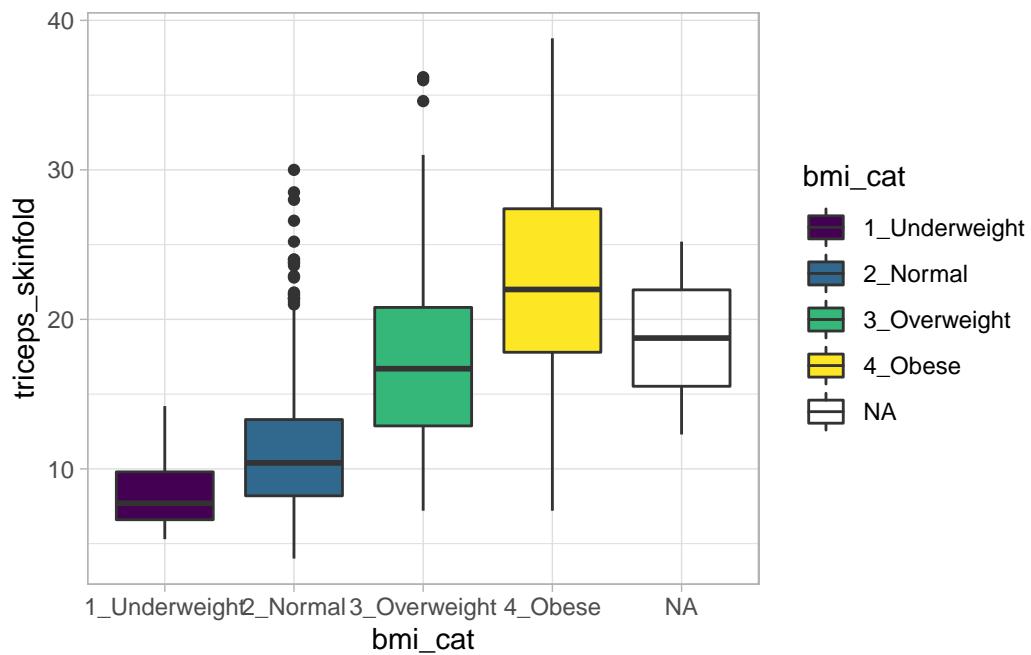
The data transformation with dplyr cheat sheet found under the Help menu in RStudio is a great resource. And, of course, for more details, visit Wickham and Grolemund (2022).

10.18 Boxplots to Relate an Outcome to a Categorical Predictor

Boxplots are much more useful when comparing samples of data. For instance, consider this comparison boxplot describing the triceps skinfold results across the four levels of BMI category.

```
ggplot(nnyfs, aes(x = bmi_cat, y = triceps_skinfold,
                  fill = bmi_cat)) +
  geom_boxplot() +
  scale_fill_viridis_d() +
  theme_light()
```

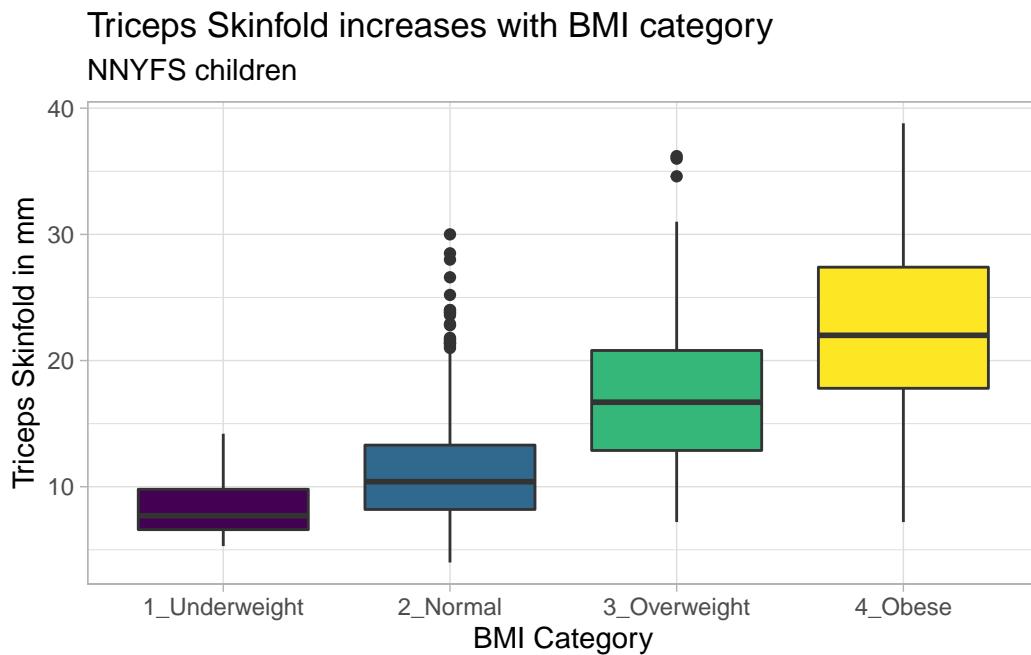
Warning: Removed 21 rows containing non-finite values (stat_boxplot).



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

```
nnyfs |>
  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 = "none") +
  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.

10.18.1 Augmenting the Boxplot with the Sample Mean

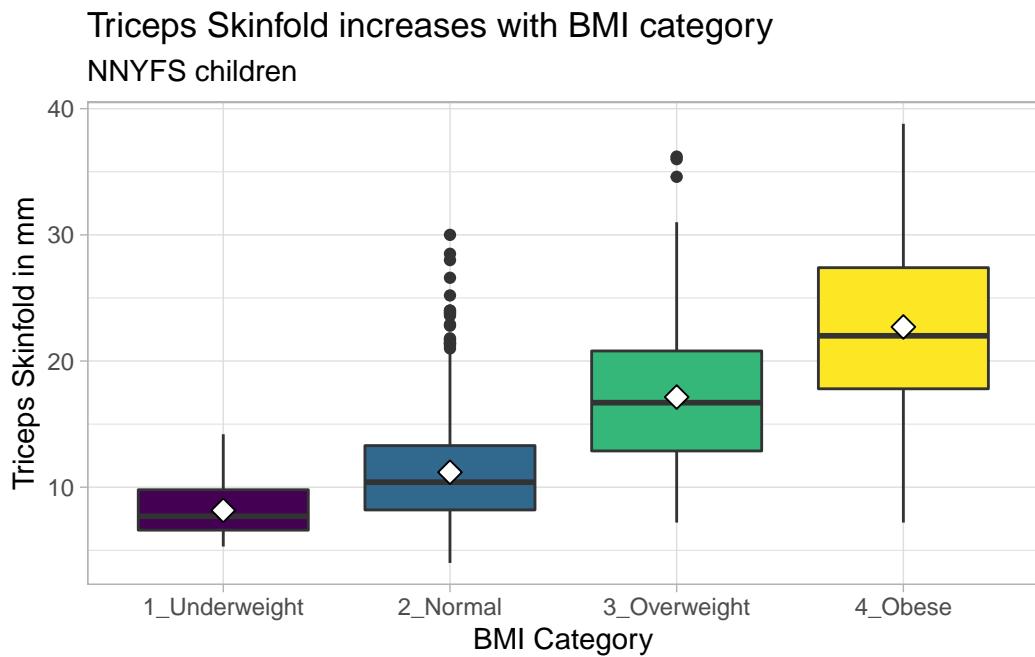
Often, we want to augment such a plot, perhaps by adding a little diamond to show the **sample mean** within each category, so as to highlight skew (in terms of whether the mean is meaningfully different from the median.)

```
nnyfs |>
  filter(complete.cases(bmi_cat, triceps_skinfold)) %>%
  ggplot(., aes(x = bmi_cat, y = triceps_skinfold,
                fill = bmi_cat)) +
  geom_boxplot() +
  stat_summary(fun="mean", geom="point",
              shape=23, size=3, fill="white") +
  scale_fill_viridis_d()
```

```

guides(fill = "none") +
theme_light() +
labs(x = "BMI Category", y = "Triceps Skinfold in mm",
title = "Triceps Skinfold increases with BMI category",
subtitle = "NNYFS children")

```



10.19 Building a Violin Plot

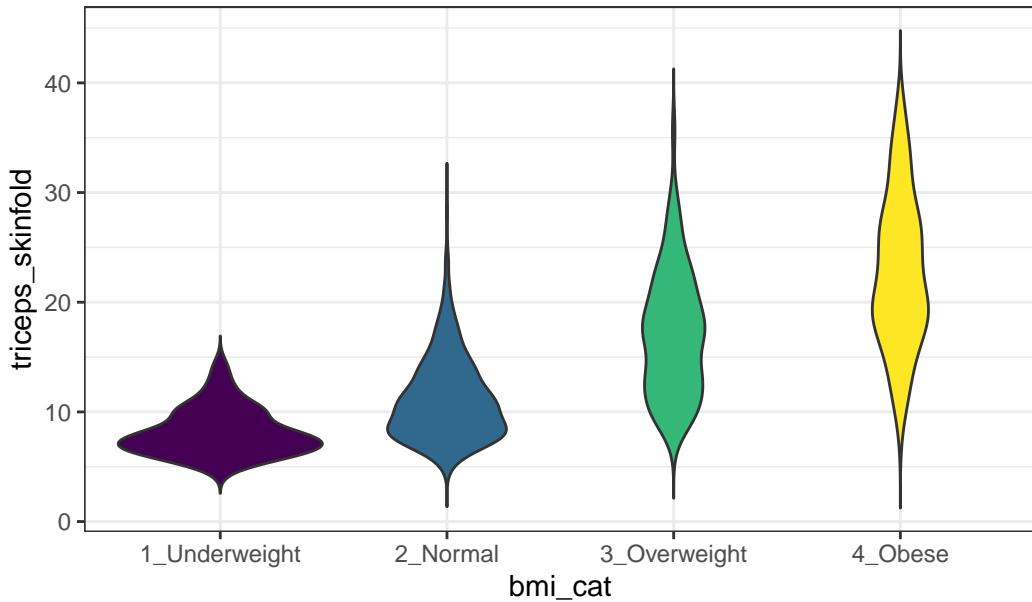
There are a number of other plots which compare distributions of data sets. An interesting one is called a **violin plot**. A violin plot is a kernel density estimate, mirrored to form a symmetrical shape.

```

nnyfs |>
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")

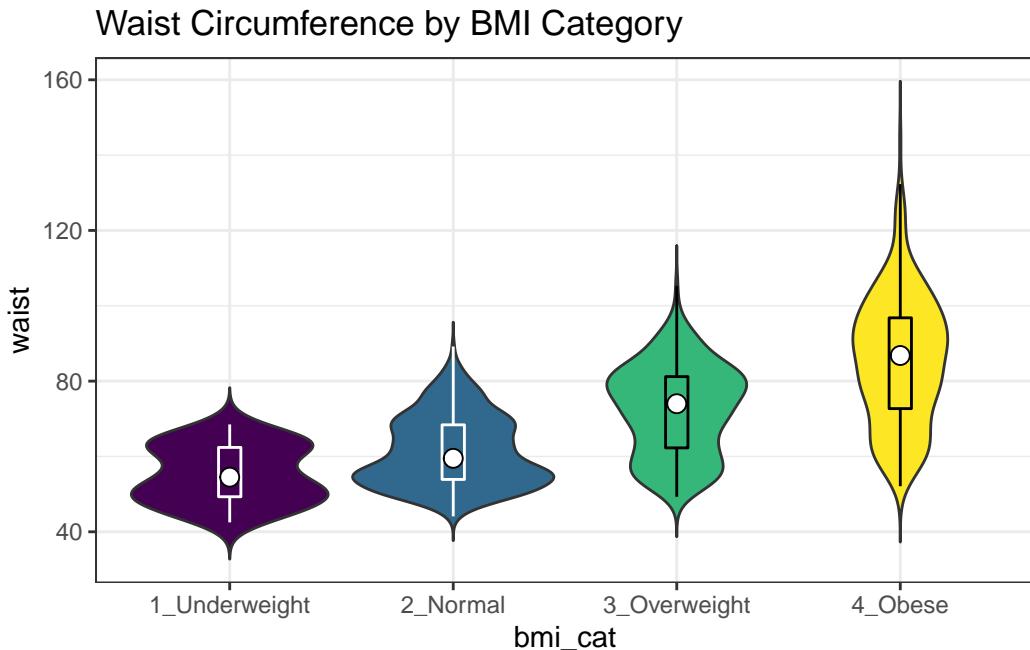
```

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

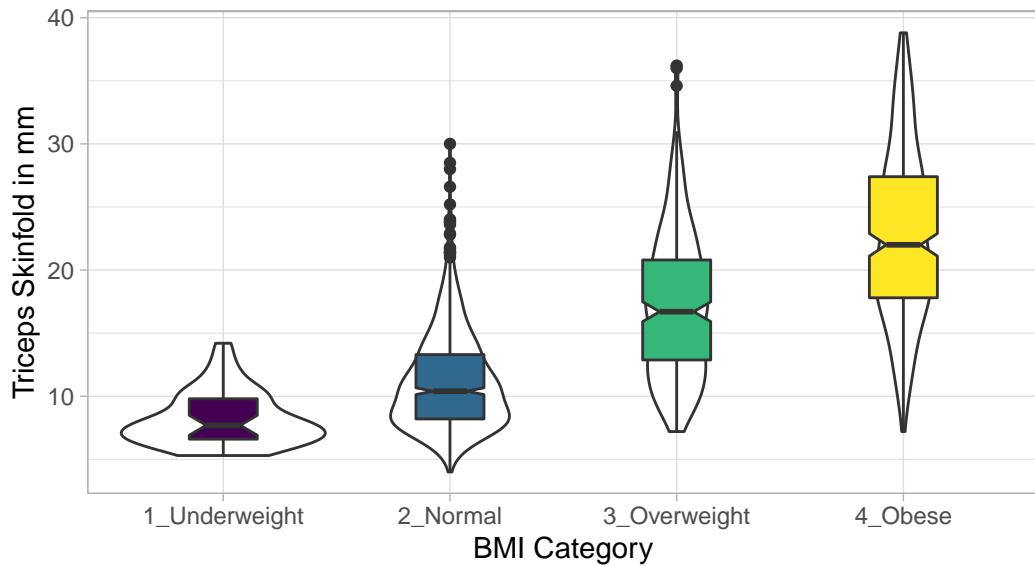


10.19.1 Adding Notches to a Boxplot

Notches are used in boxplots to help visually assess whether the medians of the distributions across the various groups actually differ to a statistically detectable extent. Think of them as confidence regions around the medians. If the notches do not overlap, as in this situation, this provides some evidence that the medians in the populations represented by these samples may be different.

```
nnyfs |>
  filter(complete.cases(bmi_cat, triceps_skinfold)) %>%
  ggplot(., aes(x = bmi_cat, y = triceps_skinfold)) +
  geom_violin() +
  geom_boxplot(aes(fill = bmi_cat), width = 0.3, notch = TRUE) +
  scale_fill_viridis_d() +
  guides(fill = "none") +
  theme_light() +
  labs(x = "BMI Category", y = "Triceps Skinfold in mm",
       title = "Triceps Skinfold increases with BMI category",
       subtitle = "NNYFS children")
```

Triceps Skinfold increases with BMI category NNYFS children

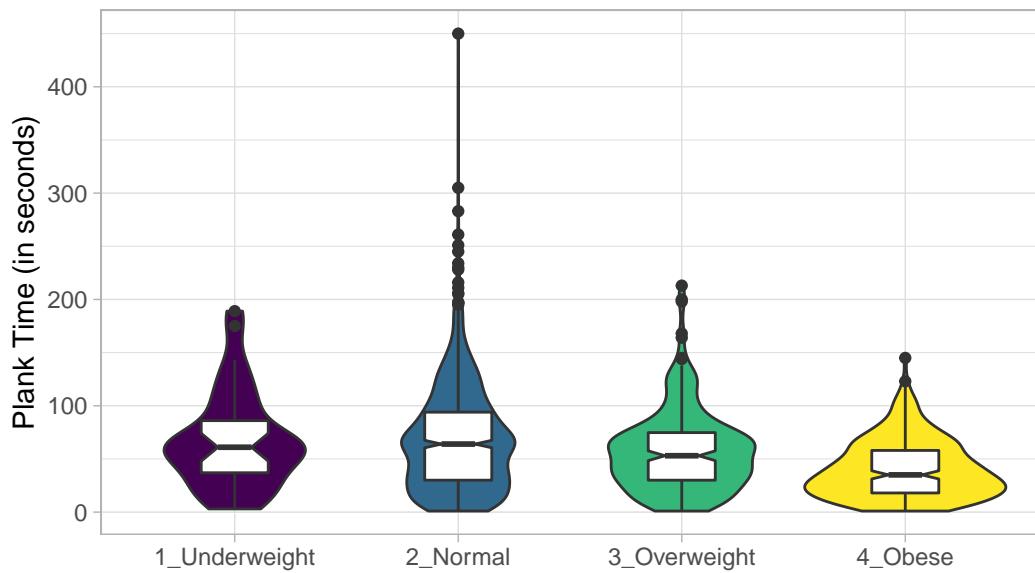


There is no overlap between the notches for each of the four categories, so we might reasonably conclude that the true median triceps skinfold values across the four categories are statistically significantly different.

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

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

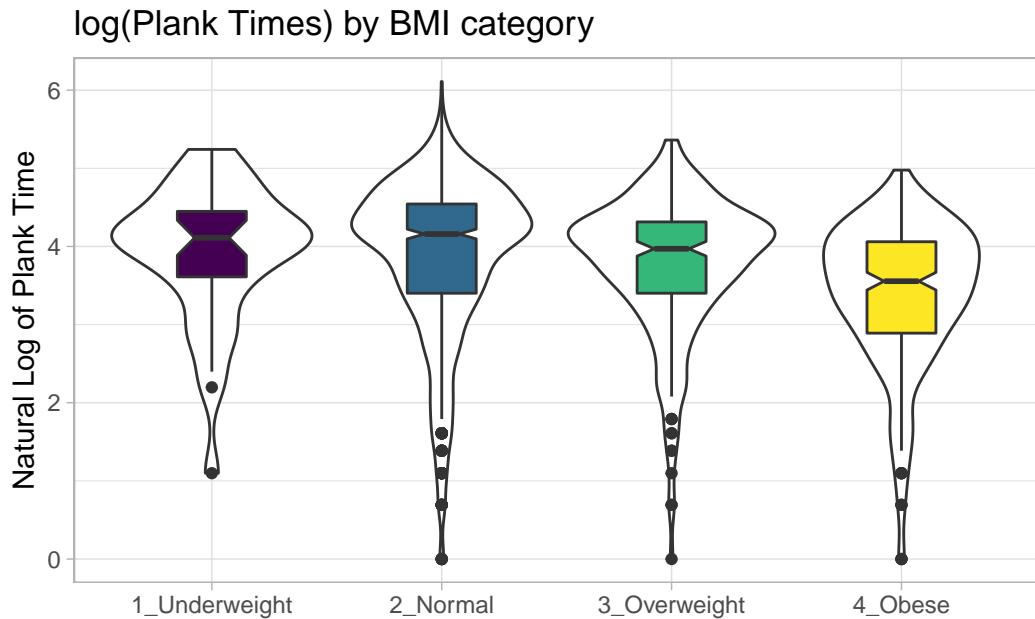
Plank Times by BMI category



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))) +
  geom_violin() +
  geom_boxplot(aes(fill = bmi_cat), width = 0.3, notch=TRUE) +
  scale_fill_viridis_d() +
  guides(fill = "none") +
  theme_light() +
  labs(title = "log(Plank Times) by BMI category",
       x = "", y = "Natural Log of Plank Time")
```

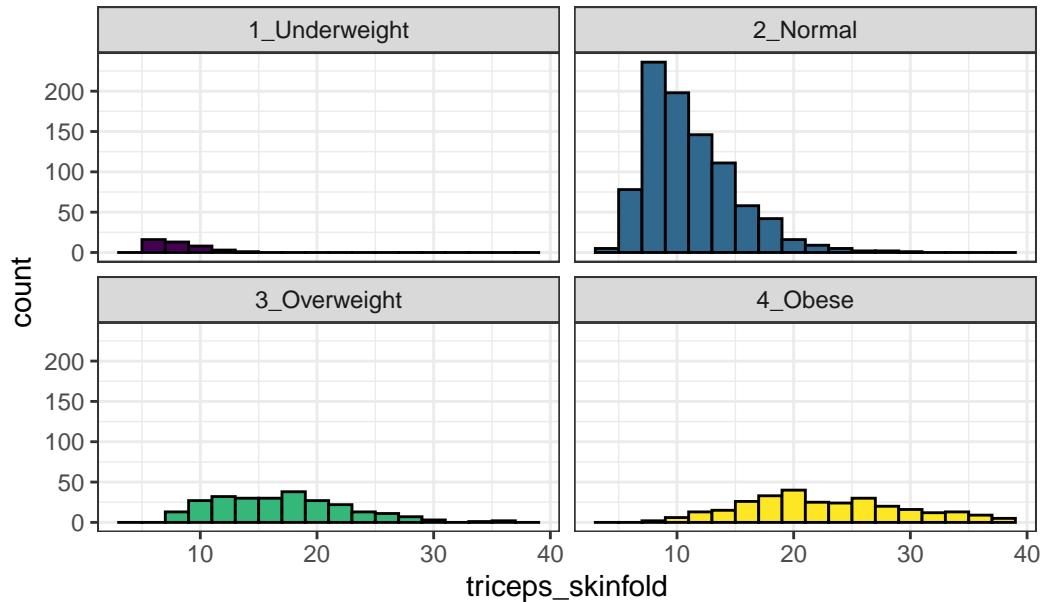


10.20 Using Multiple Histograms to Make Comparisons

We can make an array of histograms to describe multiple groups of data, using `ggplot2` and the notion of **faceting** our plot.

```
nnysf |>
  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")
```

Triceps Skinfold by BMI Category

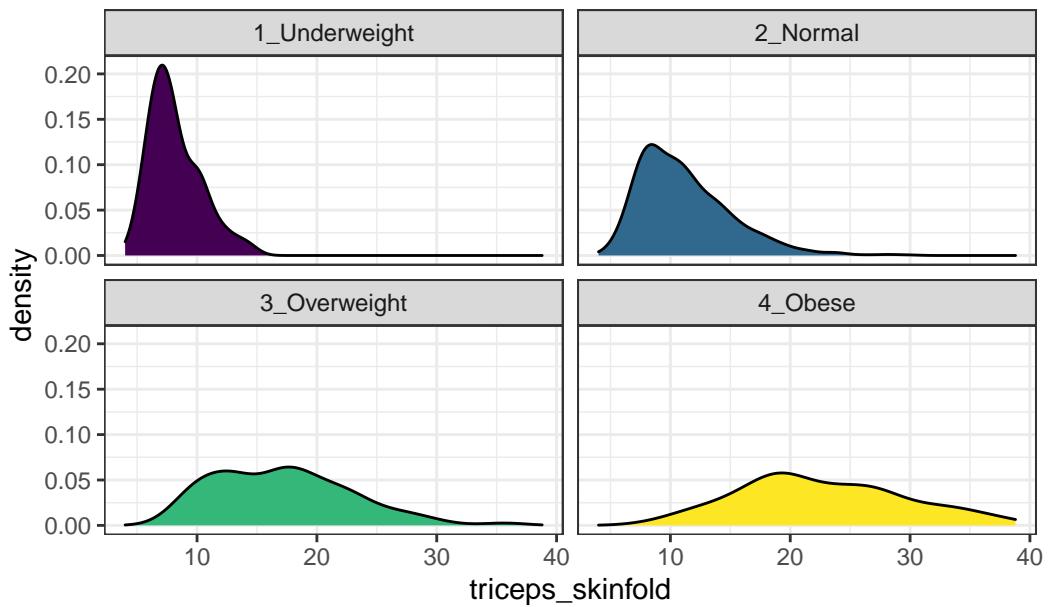


10.21 Using Multiple Density Plots to Make Comparisons

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

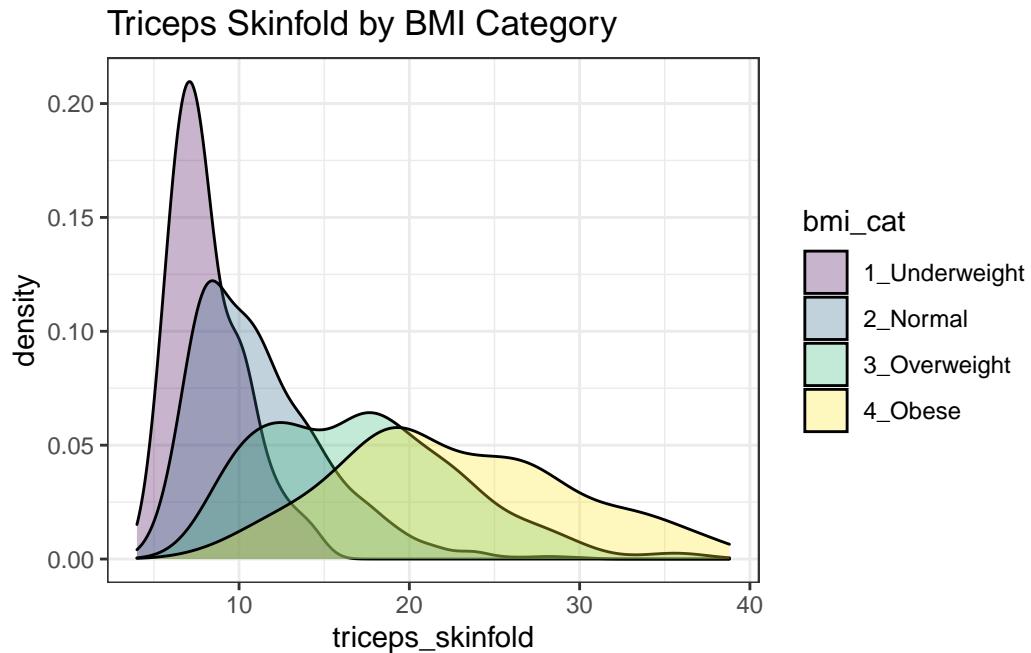
```
nyfs |>
  filter(complete.cases(triceps_skinfold, bmi_cat)) %>%
  ggplot(., aes(x=triceps_skinfold, fill = bmi_cat)) +
  geom_density(color = "black") +
  facet_wrap(~ bmi_cat) +
  scale_fill_viridis_d() +
  guides(fill = "none") +
  labs(title = "Triceps Skinfold by BMI Category")
```

Triceps Skinfold by BMI Category



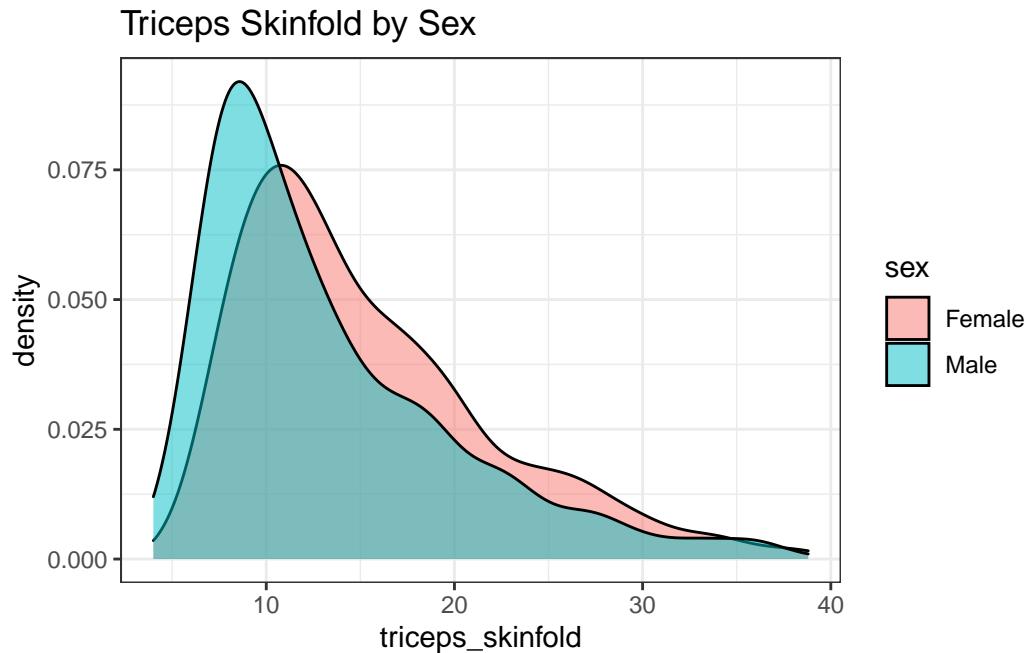
Or, we can plot all of the densities on top of each other with semi-transparent fills.

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



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

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



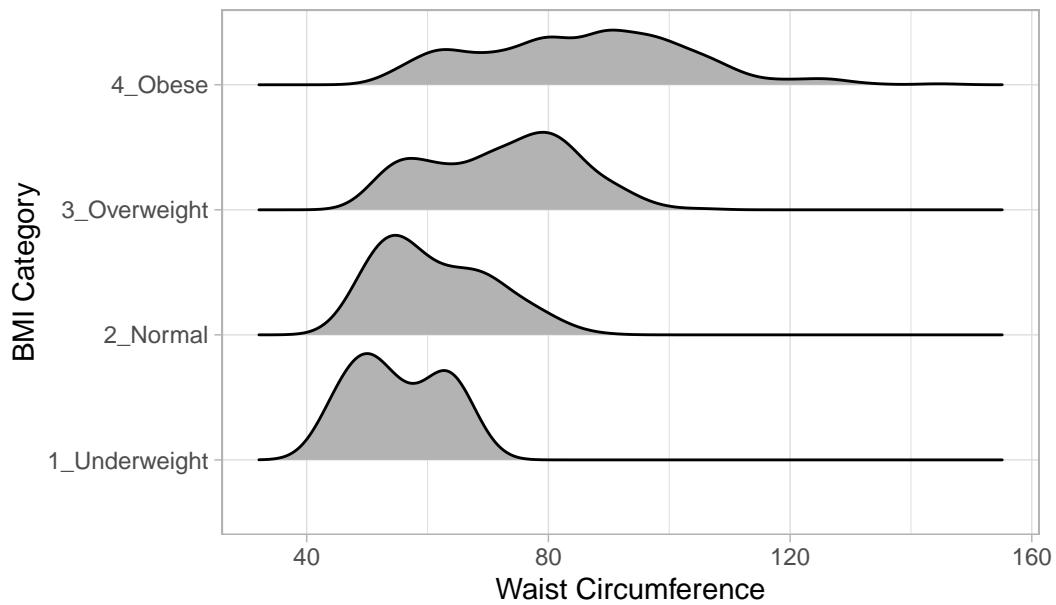
10.22 A Ridgeline Plot

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

```
nnyfs |>
  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

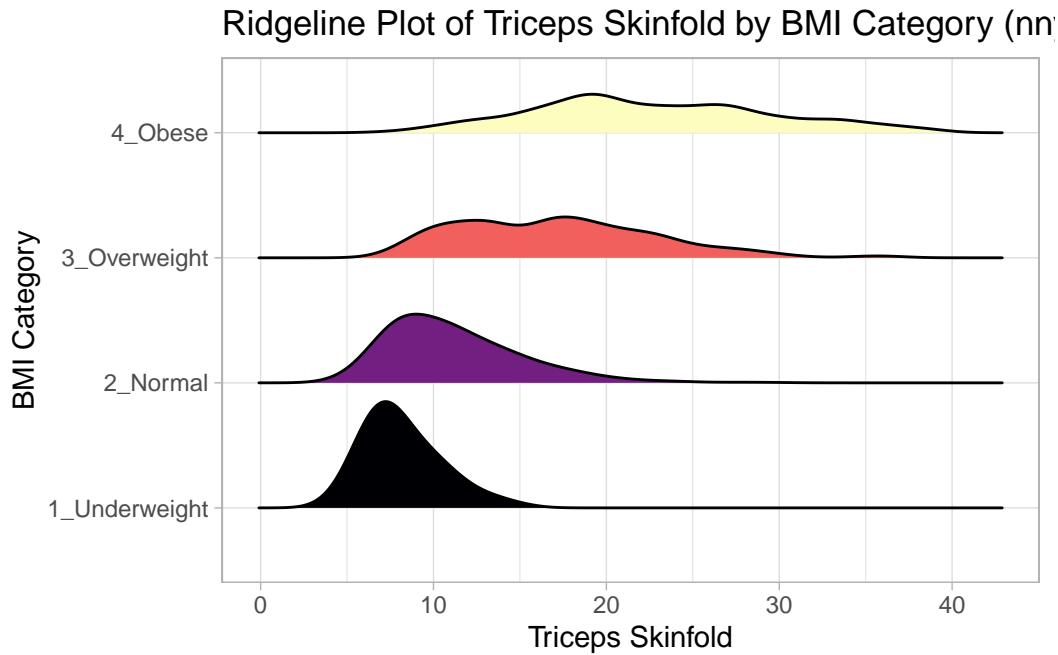
Ridgeline Plot of Waist Circumference by BMI category



And here's a ridgeline plot for the triceps skinfolds. We'll start by sorting the subgroups by the median value of our outcome (triceps skinfold) in this case, though it turns out not to matter. We'll also add some color.

```
nnyfs |>
  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 = "none") +
  labs(title = "Ridgeline Plot of Triceps Skinfold by BMI Category (nnyfs)",
       x = "Triceps Skinfold", y = "BMI Category") +
  theme_light()
```

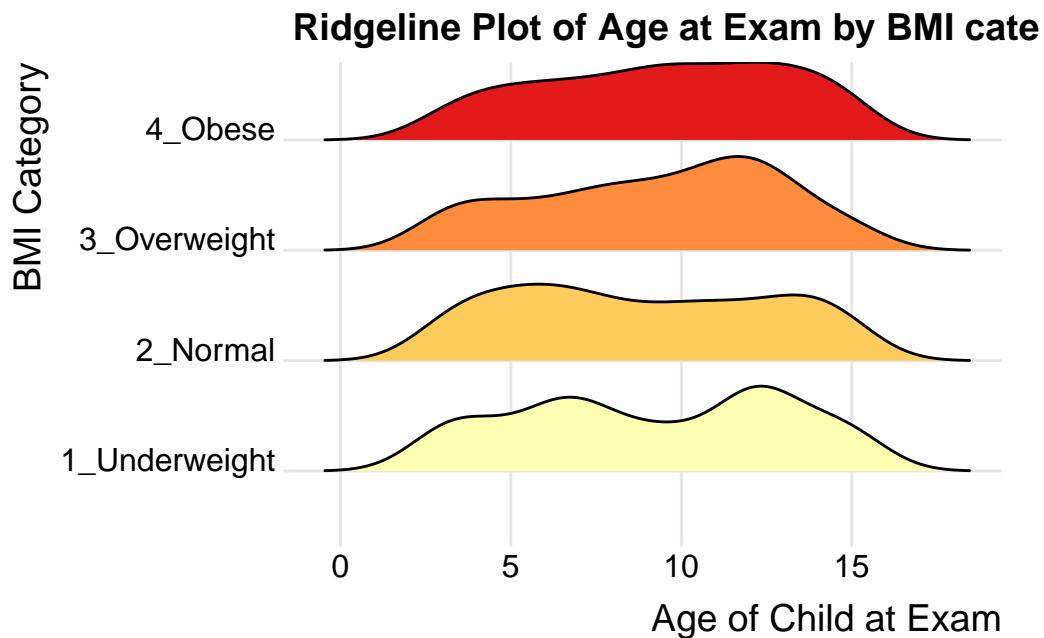
Picking joint bandwidth of 1.37



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 = "none") +
  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



10.23 What Summaries to Report

It is usually helpful to focus on the shape, center and spread of a distribution. Bock, Velleman and DeVeaux provide some useful advice:

- If the data are skewed, report the median and IQR (or the three middle quantiles). You may want to include the mean and standard deviation, but you should point out why the mean and median differ. The fact that the mean and median do not agree is a sign that the distribution may be skewed. A histogram will help you make that point.
- If the data are symmetric, report the mean and standard deviation, and possibly the median and IQR as well.
- If there are clear outliers and you are reporting the mean and standard deviation, report them with the outliers present and with the outliers removed. The differences may be revealing. The median and IQR are not likely to be seriously affected by outliers.

11 Assessing Normality

11.1 Setup: Packages Used Here

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

library(patchwork)
library(tidyverse)

theme_set(theme_bw())
```

We also use the `favstat` function from the `mosaic` package in this chapter, but do not load the whole packages.

11.2 Introduction

Data are well approximated by a Normal distribution if the shape of the data's distribution is a good match for a Normal distribution with mean and standard deviation equal to the sample statistics.

- the data are symmetrically distributed about a single peak, located at the sample mean
- the spread of the distribution is well characterized by a Normal distribution with standard deviation equal to the sample standard deviation
- the data show outlying values (both in number of candidate outliers, and size of the distance between the outliers and the center of the distribution) that are similar to what would be predicted by a Normal model.

We have several tools for assessing Normality of a single batch of data, including:

- a histogram with superimposed Normal distribution
- histogram variants (like the boxplot) which provide information on the center, spread and shape of a distribution
- the Empirical Rule for interpretation of a standard deviation

- a specialized *normal Q-Q plot* (also called a normal probability plot or normal quantile-quantile plot) designed to reveal differences between a sample distribution and what we might expect from a normal distribution of a similar number of values with the same mean and standard deviation

11.3 Empirical Rule Interpretation of the Standard Deviation

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

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

Again, most data sets do not follow a Normal distribution. We will occasionally think about transforming or re-expressing our data to obtain results which are better approximated by a Normal distribution, in part so that a standard deviation can be more meaningful.

For the energy data we have been studying, here again are some summary statistics...

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

mosaic::favstats(nnyfs$energy)

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

The mean is 1877 and the standard deviation is 722, so if the data really were Normally distributed, we'd expect to see:

- About 68% of the data in the range (1155, 2600). In fact, 1085 of the 1518 energy values are in this range, or 71.5%.
- About 95% of the data in the range (432, 3322). In fact, 1450 of the 1518 energy values are in this range, or 95.5%.
- About 99.7% of the data in the range (-290, 4044). In fact, 1502 of the 1518 energy values are in this range, or 98.9%.

So, based on this Empirical Rule approximation, do the energy data seem to be well approximated by a Normal distribution?

11.4 Describing Outlying Values with Z Scores

The maximum energy consumption value here is 5265. One way to gauge how extreme this is (or how much of an outlier it is) uses that observation's **Z score**, the number of standard deviations away from the mean that the observation falls.

Here, the maximum value, 5265 is 4.69 standard deviations above the mean, and thus has a Z score of 4.7.

A negative Z score would indicate a point below the mean, while a positive Z score indicates, as we've seen, a point above the mean. The minimum body-mass index, 257 is 2.24 standard deviations *below* the mean, so it has a Z score of -2.2.

Recall that the Empirical Rule suggests that if a variable follows a Normal distribution, it would have approximately 95% of its observations falling inside a Z score of (-2, 2), and 99.74% falling inside a Z score range of (-3, 3).

11.4.1 Fences and Z Scores

Note the relationship between the fences (Tukey's approach to identifying points which fall within the whiskers of a boxplot, as compared to candidate outliers) and the Z scores.

The upper inner fence in this case falls at 3713.75, which indicates a Z score of 2.5, while the lower inner fence falls at -40.25, which indicates a Z score of -2.7. It is neither unusual nor inevitable for the inner fences to fall at Z scores near -2.0 and +2.0.

11.5 Comparing a Histogram to a Normal Distribution

Most of the time, when we want to understand whether our data are well approximated by a Normal distribution, we will use a graph to aid in the decision.

One option is to build a histogram with a Normal density function (with the same mean and standard deviation as our data) superimposed. This is one way to help visualize deviations between our data and what might be expected from a Normal distribution.

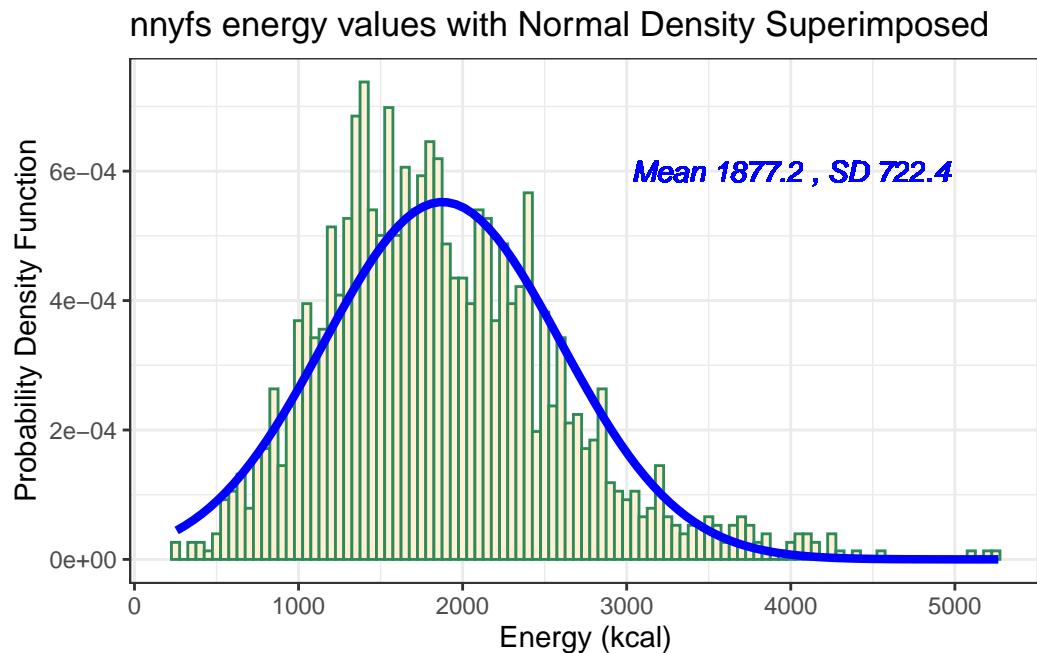
```
res <- mosaic::favstats(~ energy, data = nnyfs)
bin_w <- 50 # specify binwidth

ggplot(nnyfs, aes(x=energy)) +
  geom_histogram(aes(y = ..density..), binwidth = bin_w,
                 fill = "papayawhip", color = "seagreen") +
  stat_function(fun = dnorm,
```

```

    args = list(mean = res$mean, sd = res$sd),
    lwd = 1.5, col = "blue") +
  geom_text(aes(label = paste("Mean", round(res$mean,1),
                             ", SD", round(res$sd,1))),
            x = 4000, y = 0.0006,
            color="blue", fontface = "italic") +
  labs(title = "nnyfs energy values with Normal Density Superimposed",
       x = "Energy (kcal)", y = "Probability Density Function")

```



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

We'll return shortly to the questions:

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

11.5.1 Histogram of energy with Normal model (with Counts)

But first, we'll demonstrate an approach to building a histogram of counts (rather than a probability density) and then superimposing a Normal model.

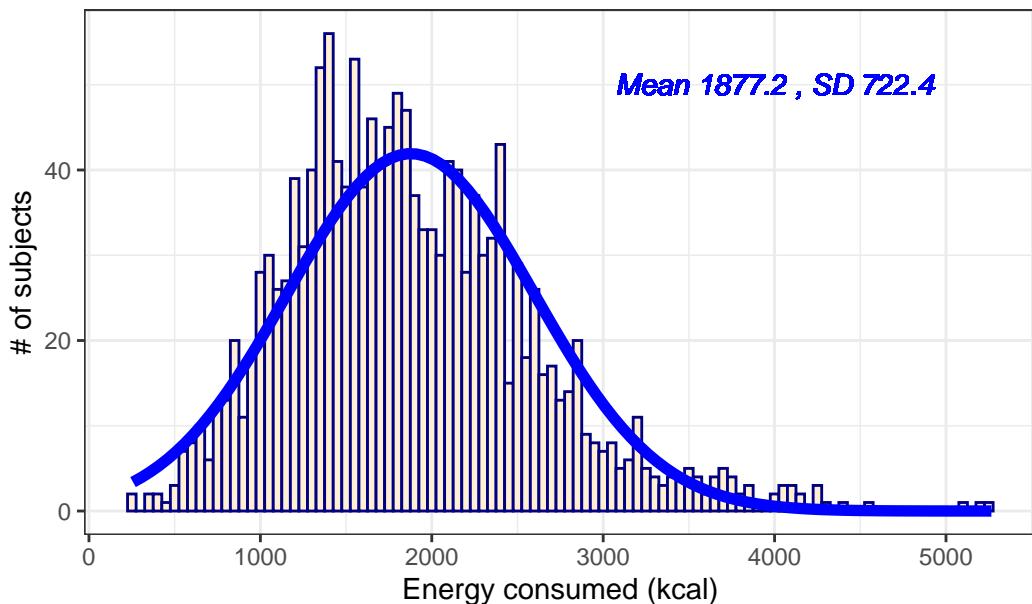
```

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

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



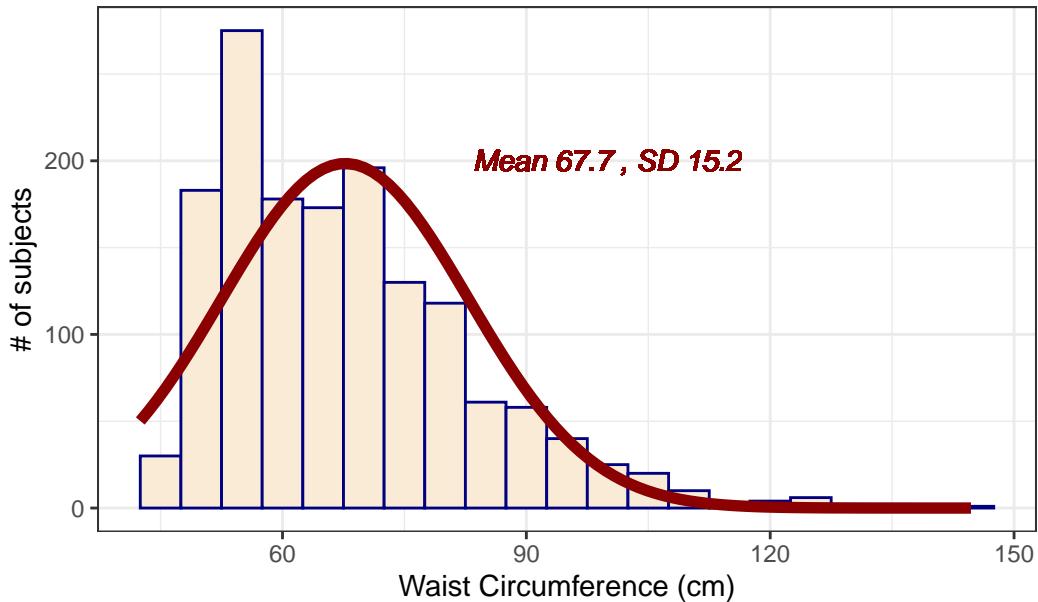
11.6 Does a Normal model work well for the waist circumference?

Now, suppose we instead look at the `waist` data, remembering to filter the data to the complete cases before plotting. Do these data appear to follow a Normal distribution?

```
res <- mosaic::favstats(~ waist, data = nnyfs)
bin_w <- 5 # specify binwidth

nnnyfs |> filter(complete.cases(waist)) %>%
  ggplot(., aes(x = waist)) +
  geom_histogram(binwidth = bin_w,
                 fill = "antiquewhite",
                 col = "navy") +
  theme_bw() +
  stat_function(
    fun = function(x) dnorm(x, mean = res$mean,
                           sd = res$sd) *
      res$n * bin_w,
    col = "darkred", size = 2) +
  geom_text(aes(label = paste("Mean", round(res$mean,1),
                            ", SD", round(res$sd,1))),
            x = 100, y = 200,
            color="darkred", fontface = "italic") +
  labs(title = "Histogram of waist, with Normal Model",
       x = "Waist Circumference (cm)", y = "# of subjects")
```

Histogram of waist, with Normal Model



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

| min | Q1 | median | Q3 | max | mean | sd | n | missing |
|------|------|--------|------|-------|----------|----------|------|---------|
| 42.5 | 55.6 | 64.8 | 76.6 | 144.7 | 67.70536 | 15.19809 | 1512 | 6 |

The mean is 67.71 and the standard deviation is 15.2 so if the `waist` data really were Normally distributed, we'd expect to see:

- About 68% of the data in the range (52.51, 82.9). In fact, 1076 of the 1512 Age values are in this range, or 71.2%.
- About 95% of the data in the range (37.31, 98.1). In fact, 1443 of the 1512 Age values are in this range, or 95.4%.
- About 99.7% of the data in the range (22.11, 113.3). In fact, 1500 of the 1512 Age values are in this range, or 99.2%.

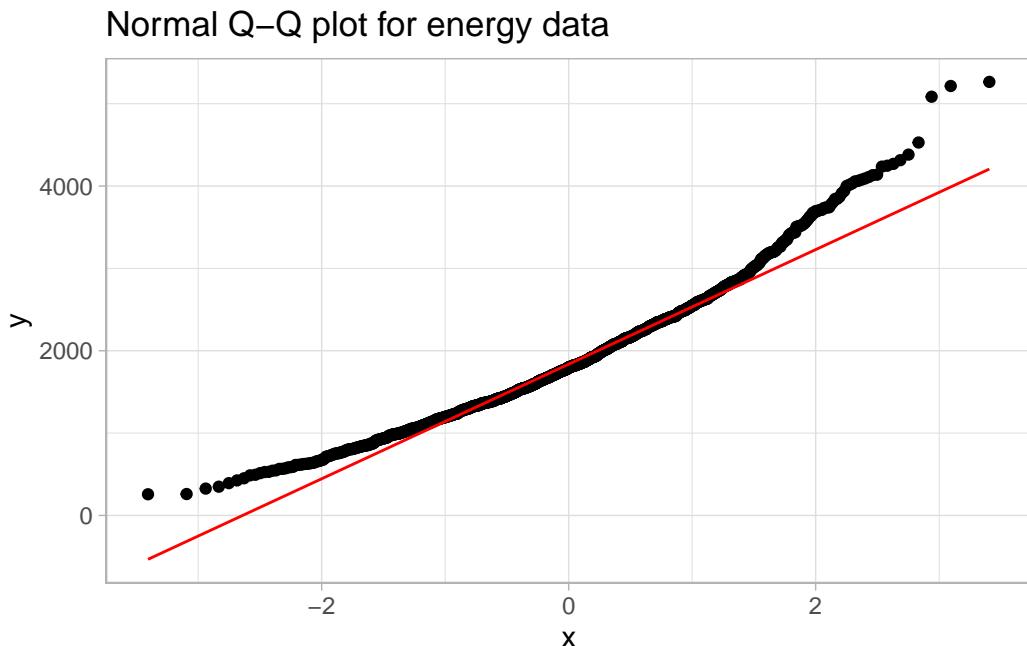
How does the Normal approximation work for waist circumference, according to the Empirical Rule?

11.7 The Normal Q-Q Plot

A normal probability plot (or normal quantile-quantile plot) of the energy results from the `nnyfs` data, developed using `ggplot2` is shown below. In this case, this is a picture of 1518 energy consumption assessments. The idea of a normal Q-Q plot is that it plots the observed sample values (on the vertical axis) and then, on the horizontal, the expected or theoretical quantiles that would be observed in a standard normal distribution (a Normal distribution with mean 0 and standard deviation 1) with the same number of observations.

A Normal Q-Q plot will follow a straight line when the data are (approximately) Normally distributed. When the data have a different shape, the plot will reflect that.

```
ggplot(nnyfs, aes(sample = energy)) +  
  geom_qq() + geom_qq_line(col = "red") +  
  theme_light() +  
  labs(title = "Normal Q-Q plot for energy data")
```



11.8 Interpreting the Normal Q-Q Plot

The purpose of a Normal Q-Q plot is to help point out distinctions from a Normal distribution. A Normal distribution is symmetric and has certain expectations regarding its tails. The

Normal Q-Q plot can help us identify data as well approximated by a Normal distribution, or not, because of:

- skew (including distinguishing between right skew and left skew)
- behavior in the tails (which could be heavy-tailed [more outliers than expected] or light-tailed)

11.8.1 Data from a Normal distribution shows up as a straight line in a Normal Q-Q plot

We'll demonstrate the looks that we can obtain from a Normal Q-Q plot in some simulations. First, here is an example of a Normal Q-Q plot, and its associated histogram, for a sample of 200 observations simulated from a Normal distribution.

```
set.seed(123431) # so the results can be replicated

# simulate 200 observations from a Normal(20, 5) distribution and place them
# in the d variable within the temp.1 data frame
temp.1 <- data.frame(d = rnorm(200, mean = 20, sd = 5))

# left plot - basic Normal Q-Q plot of simulated data
p1 <- ggplot(temp.1, aes(sample = d)) +
  geom_qq() + geom_qq_line(col = "red") +
  theme_light() +
  labs(y = "Ordered Simulated Sample Data")

# right plot - histogram with superimposed normal distribution
res <- mosaic::favstats(~ d, data = temp.1)
bin_w <- 2 # specify binwidth

p2 <- ggplot(temp.1, aes(x = d)) +
  geom_histogram(binwidth = bin_w,
                 fill = "papayawhip",
                 col = "seagreen") +
  theme_bw() +
  stat_function(
    fun = function(x) dnorm(x, mean = res$mean,
                           sd = res$sd) *
      res$n * bin_w,
    col = "blue", size = 1.5) +
  geom_text(aes(label = paste("Mean", round(res$mean, 1),
```

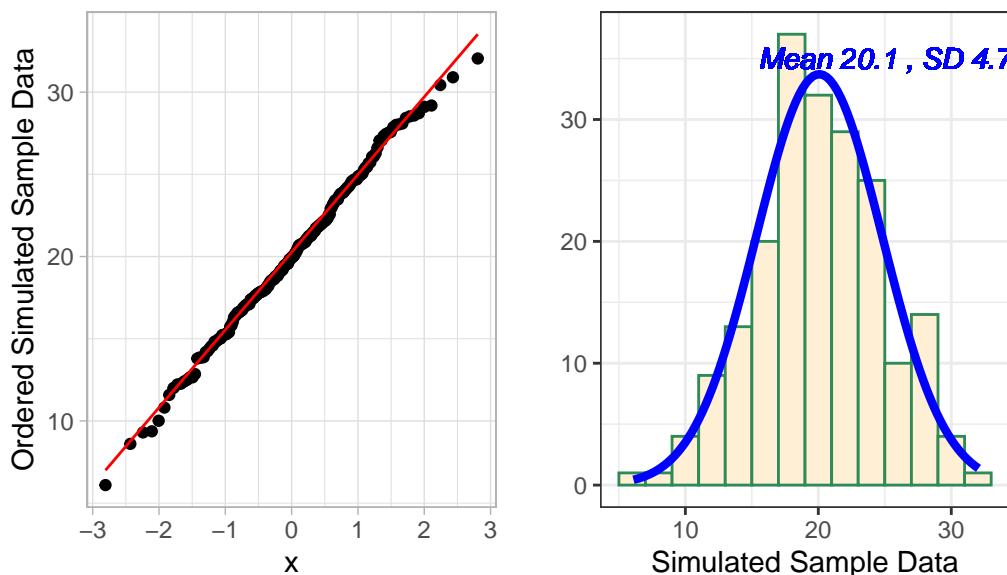
```

    ", SD", round(res$sd,1))),
x = 25, y = 35,
color="blue", fontface = "italic") +
labs(x = "Simulated Sample Data", y = "")

p1 + p2 +
plot_annotation(title = "200 observations from a simulated Normal distribution")

```

200 observations from a simulated Normal distribution



```
# uses patchwork package to combine plots
```

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

- there is no substantial curve (such as we'd see with data that were skewed)
- there is no particularly surprising behavior (curves away from the line) at either tail, so there's no obvious problem with outliers

11.8.2 Skew is indicated by monotonic curves in the Normal Q-Q plot

Data that come from a skewed distribution appear to curve away from a straight line in the Q-Q plot.

```

set.seed(123431) # so the results can be replicated

# simulate 200 observations from a beta(5, 2) distribution into the e1 variable
# simulate 200 observations from a beta(1, 5) distribution into the e2 variable
temp.2 <- data.frame(e1 = rbeta(200, 5, 2), e2 = rbeta(200, 1, 5))

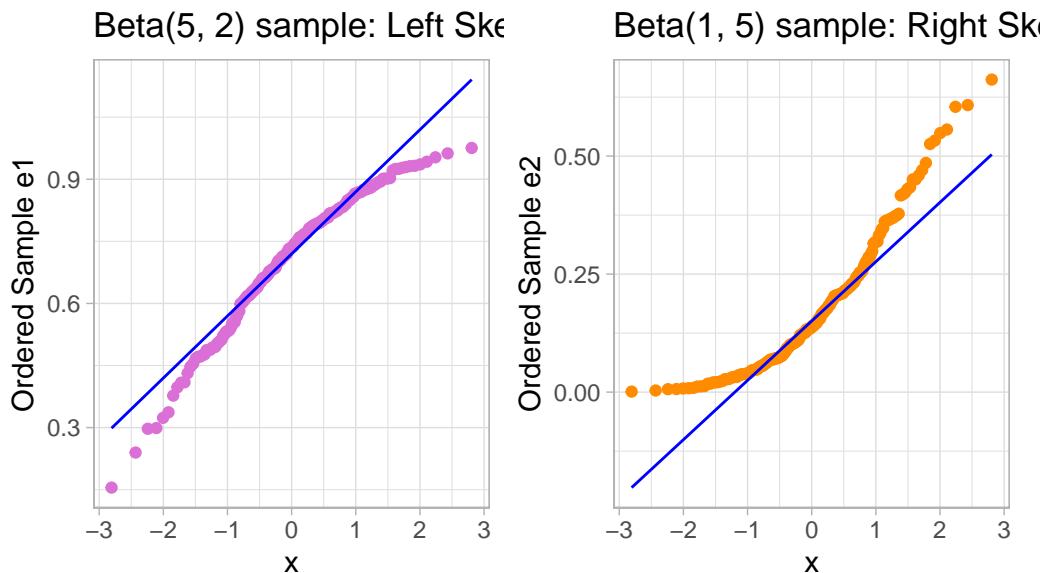
p1 <- ggplot(temp.2, aes(sample = e1)) +
  geom_qq(col = "orchid") + geom_qq_line(col = "blue") +
  theme_light() +
  labs(y = "Ordered Sample e1",
       title = "Beta(5, 2) sample: Left Skewed")

p2 <- ggplot(temp.2, aes(sample = e2)) +
  geom_qq(col = "darkorange") + geom_qq_line(col = "blue") +
  theme_light() +
  labs(y = "Ordered Sample e2",
       title = "Beta(1, 5) sample: Right Skewed")

p1 + p2 + plot_annotation(title = "200 observations from simulated Beta distributions")

```

200 observations from simulated Beta distributions



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

```

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

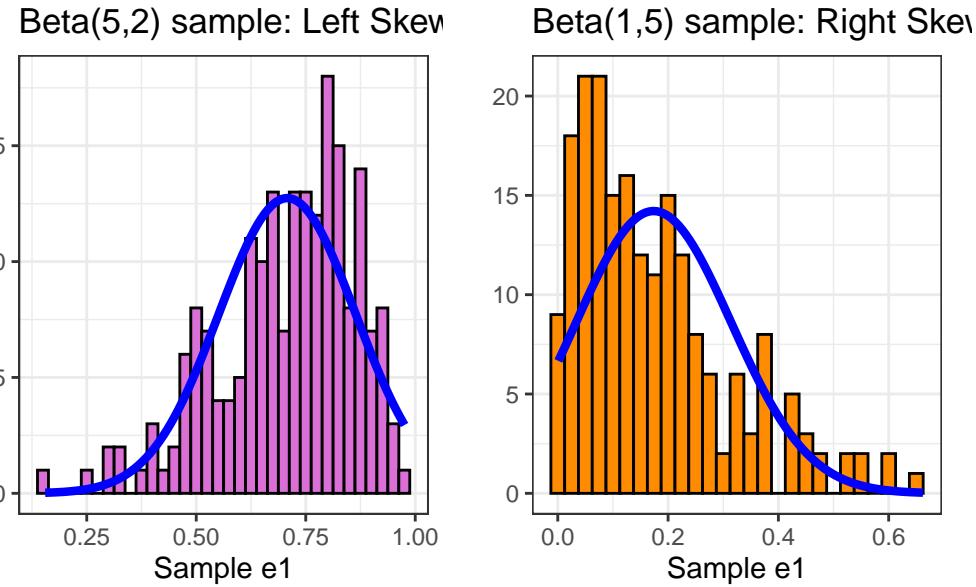
p1 <- ggplot(temp.2, aes(x = e1)) +
  geom_histogram(binwidth = bin_w1,
                 fill = "orchid",
                 col = "black") +
  theme_bw() +
  stat_function(
    fun = function(x) dnorm(x, mean = res1$mean,
                            sd = res1$sd) *
      res1$n * bin_w1,
    col = "blue", size = 1.5) +
  labs(x = "Sample e1", y = "",
       title = "Beta(5,2) sample: Left Skew")

res2 <- mosaic::favstats(~ e2, data = temp.2)
bin_w2 <- 0.025 # specify binwidth

p2 <- ggplot(temp.2, aes(x = e2)) +
  geom_histogram(binwidth = bin_w2,
                 fill = "darkorange",
                 col = "black") +
  theme_bw() +
  stat_function(
    fun = function(x) dnorm(x, mean = res2$mean,
                            sd = res2$sd) *
      res2$n * bin_w2,
    col = "blue", size = 1.5) +
  labs(x = "Sample e1", y = "",
       title = "Beta(1,5) sample: Right Skew")

p1 + p2 + plot_annotation(caption = "Histograms with Normal curve superimposed")

```



Histograms with Normal curve superimposed

11.8.3 Direction of Skew

In each of these pairs of plots, we see the same basic result.

- The left plot (for data e1) shows left skew, with a longer tail on the left hand side and more clustered data at the right end of the distribution.
- The right plot (for data e2) shows right skew, with a longer tail on the right hand side, the mean larger than the median, and more clustered data at the left end of the distribution.

11.8.4 Outlier-proneness is indicated by “s-shaped” curves in a Normal Q-Q plot

- Heavy-tailed but symmetric distributions are indicated by reverse “S”-shapes, as shown on the left below.
- Light-tailed but symmetric distributions are indicated by “S” shapes in the plot, as shown on the right below.

```
set.seed(4311) # so the results can be replicated

# sample 200 observations from each of two probability distributions
temp.3 <- data.frame(s1 = rcauchy(200, location=10, scale = 1),
                      s2 = runif(200, -30, 30))
```

```

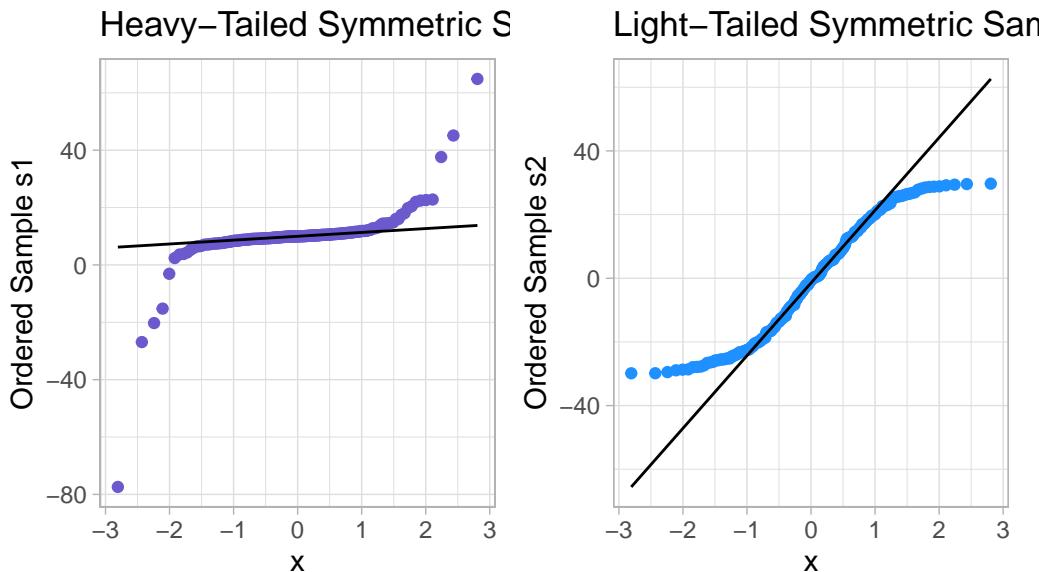
p1 <- ggplot(temp.3, aes(sample = s1)) +
  geom_qq(col = "slateblue") + geom_qq_line(col = "black") +
  theme_light() +
  labs(y = "Ordered Sample s1",
       title = "Heavy-Tailed Symmetric Sample s1")

p2 <- ggplot(temp.3, aes(sample = s2)) +
  geom_qq(col = "dodgerblue") + geom_qq_line(col = "black") +
  theme_light() +
  labs(y = "Ordered Sample s2",
       title = "Light-Tailed Symmetric Sample s2")

p1 + p2 + plot_annotation(title = "200 observations from simulated distributions")

```

200 observations from simulated distributions



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

```

res1 <- mosaic::favstats(~ s1, data = temp.3)
bin_w1 <- 20 # specify binwidth

```

```

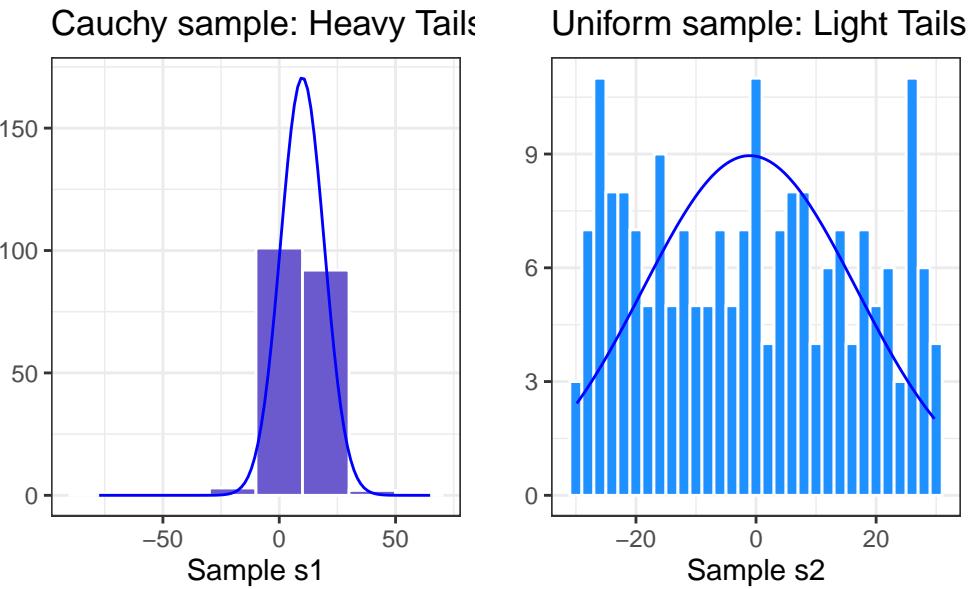
p1 <- ggplot(temp.3, aes(x = s1)) +
  geom_histogram(binwidth = bin_w1,
                 fill = "slateblue",
                 col = "white") +
  theme_bw() +
  stat_function(
    fun = function(x) dnorm(x, mean = res1$mean,
                            sd = res1$sd) *
    res1$n * bin_w1,
    col = "blue") +
  labs(x = "Sample s1", y = "",
       title = "Cauchy sample: Heavy Tails")

res2 <- mosaic::favstats(~ s2, data = temp.3)
bin_w2 <- 2 # specify binwidth

p2 <- ggplot(temp.3, aes(x = s2)) +
  geom_histogram(binwidth = bin_w2,
                 fill = "dodgerblue",
                 col = "white") +
  theme_bw() +
  stat_function(
    fun = function(x) dnorm(x, mean = res2$mean,
                            sd = res2$sd) *
    res2$n * bin_w2,
    col = "blue") +
  labs(x = "Sample s2", y = "",
       title = "Uniform sample: Light Tails")

p1 + p2 + plot_annotation(caption = "Histograms with Normal curve superimposed")

```



Histograms with Normal curve superimposed

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

```

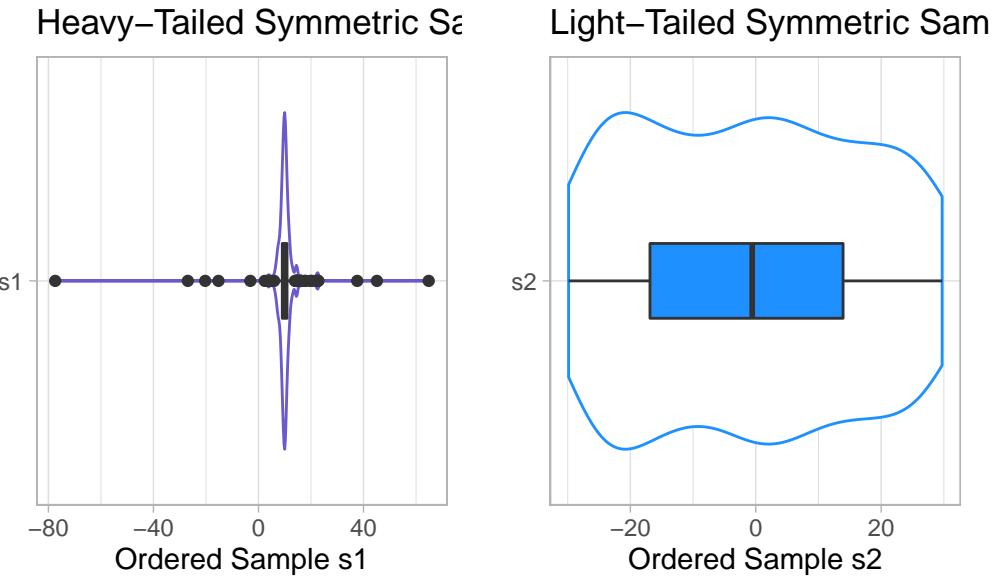
p1 <- ggplot(temp.3, aes(x = "s1", y = s1)) +
  geom_violin(col = "slateblue") +
  geom_boxplot(fill = "slateblue", width = 0.2) +
  theme_light() +
  coord_flip() +
  labs(y = "Ordered Sample s1", x = "",
       title = "Heavy-Tailed Symmetric Sample s1")

p2 <- ggplot(temp.3, aes(x = "s2", y = s2)) +
  geom_violin(col = "dodgerblue") +
  geom_boxplot(fill = "dodgerblue", width = 0.2) +
  theme_light() +
  coord_flip() +
  labs(y = "Ordered Sample s2", x = "",
       title = "Light-Tailed Symmetric Sample s2")

p1 + p2 + plot_annotation(title = "200 observations from simulated distributions")

```

200 observations from simulated distributions



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

11.9 Can a Normal Distribution Fit the nnyfs energy data Well?

The `energy` data we've been studying shows meaningful signs of right skew.

```
p1 <- ggplot(nnyfs, aes(sample = energy)) +
  geom_qq(col = "coral", size = 2) +
  geom_qq_line(col = "blue") +
  theme_light() +
  labs(title = "Energy Consumed",
       y = "Sorted Energy data")

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

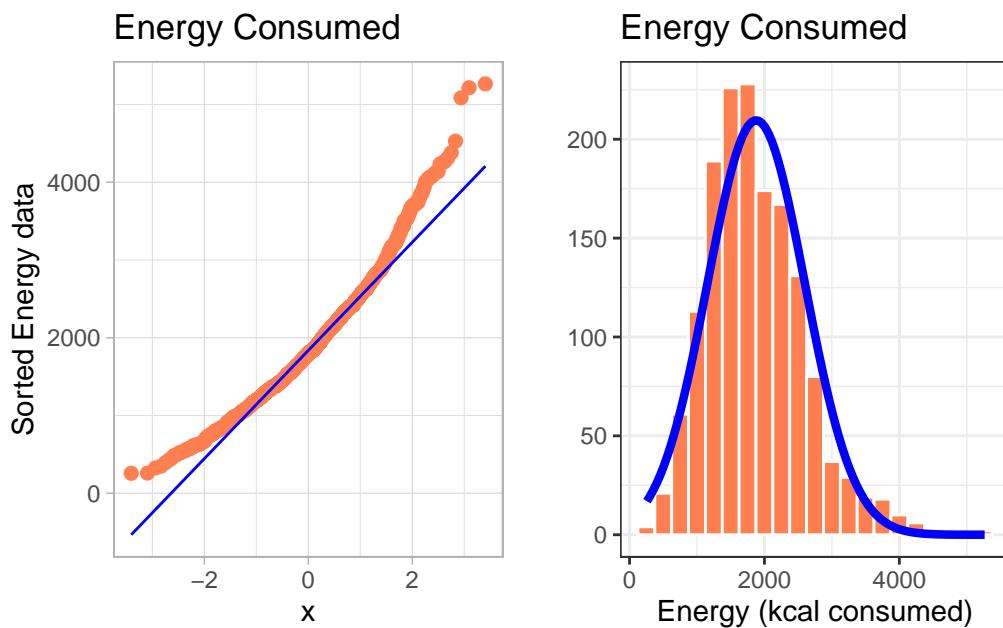
p2 <- ggplot(nnyfs, aes(x = energy)) +
  geom_histogram(binwidth = bin_w,
                 fill = "coral",
                 col = "white") +
```

```

theme_bw() +
stat_function(
  fun = function(x) dnorm(x, mean = res$mean,
                           sd = res$sd) *
  res$n * bin_w,
  col = "blue", size = 1.5) +
labs(x = "Energy (kcal consumed)", y = "",
     title = "Energy Consumed")

```

p1 + p2



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

What if we plotted not the original `energy` values (all of which are positive) but instead plotted the square roots of the `energy` values?

- Compare these two plots - the left describes the distribution of the original energy data from the NNYFS data frame, and the right plot shows the distribution of the square root of those values.

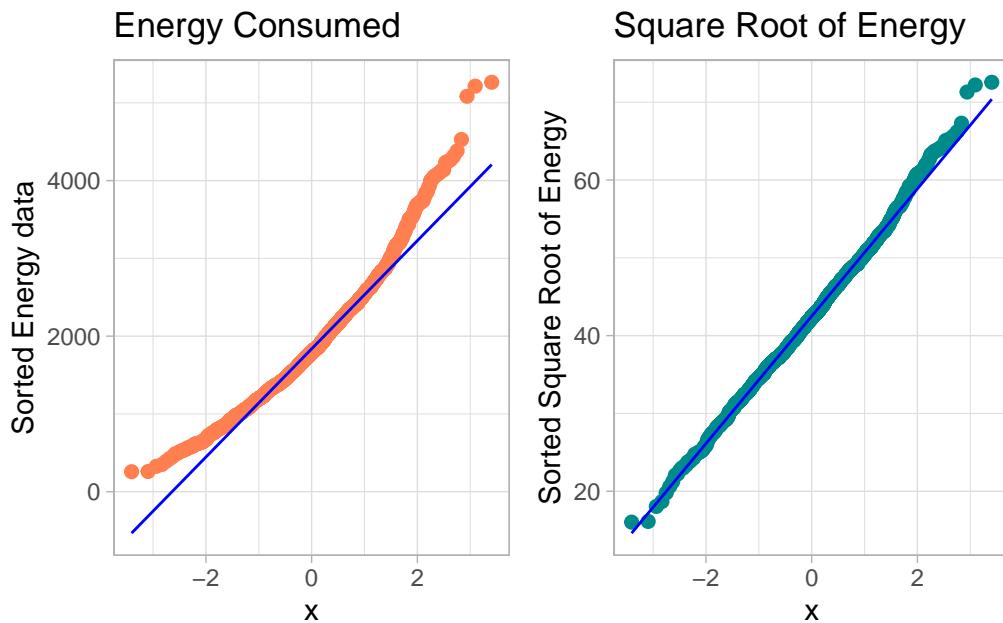
```

p1 <- ggplot(nnyfs, aes(sample = energy)) +
  geom_qq(col = "coral", size = 2) +
  geom_qq_line(col = "blue") +
  theme_light() +
  labs(title = "Energy Consumed",
       y = "Sorted Energy data")

p2 <- ggplot(nnyfs, aes(sample = sqrt(energy))) +
  geom_qq(col = "darkcyan", size = 2) +
  geom_qq_line(col = "blue") +
  theme_light() +
  labs(title = "Square Root of Energy",
       y = "Sorted Square Root of Energy")

p1 + p2

```



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

Our conclusion is that a Normal model is a far better fit to the square root of the energy values than it is to the raw energy values.

The effect of taking the square root may be clearer from the histograms below, with Normal

models superimposed.

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

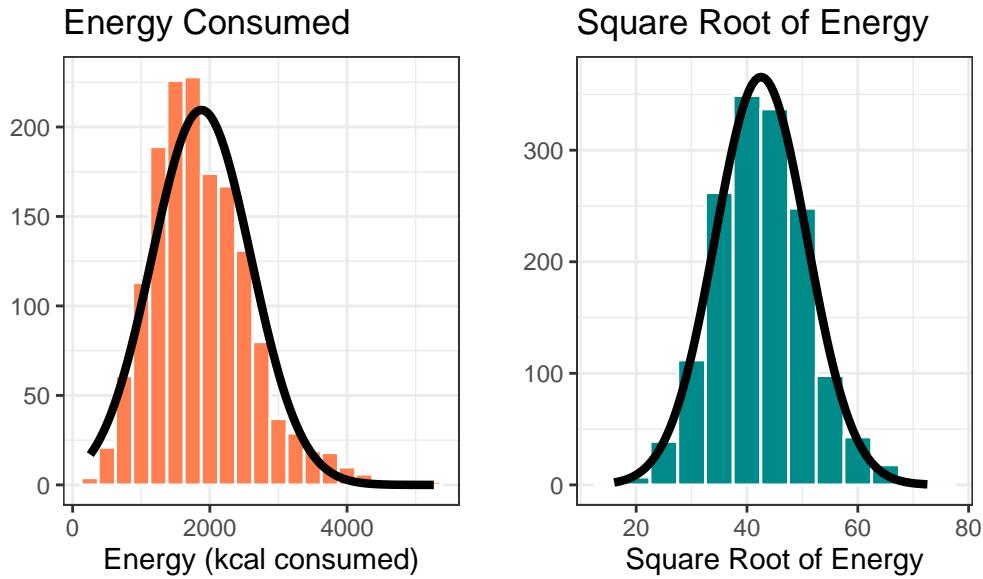
p1 <- ggplot(nnyfs, aes(x = energy)) +
  geom_histogram(binwidth = bin_w,
                 fill = "coral",
                 col = "white") +
  theme_bw() +
  stat_function(
    fun = function(x) dnorm(x, mean = res$mean,
                           sd = res$sd) *
      res$n * bin_w,
    col = "black", size = 1.5) +
  labs(x = "Energy (kcal consumed)", y = "",
       title = "Energy Consumed")

res2 <- mosaic::favstats(~ sqrt(energy), data = nnyfs)
bin_w2 <- 5 # specify binwidth

p2 <- ggplot(nnyfs, aes(x = sqrt(energy))) +
  geom_histogram(binwidth = bin_w2,
                 fill = "darkcyan",
                 col = "white") +
  theme_bw() +
  stat_function(
    fun = function(x) dnorm(x, mean = res2$mean,
                           sd = res2$sd) *
      res2$n * bin_w2,
    col = "black", size = 1.5) +
  labs(x = "Square Root of Energy", y = "",
       title = "Square Root of Energy")

p1 + p2 + plot_annotation(title = "Comparing energy to sqrt(energy)")
```

Comparing energy to $\sqrt{\text{energy}}$



```
rm(p1, p2, bin_w, bin_w2, res, res2) # cleanup
```

When we are confronted with a variable that is not Normally distributed but that we wish was Normally distributed, it is sometimes useful to consider whether working with a **transformation** of the data will yield a more helpful result, as the square root does in this instance.

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

- When we are confronted with a variable that is not Normally distributed but that we wish was Normally distributed, it is sometimes useful to consider whether working with a transformation of the data will yield a more helpful result.
- Many statistical methods, including t tests and analyses of variance, assume Normal distributions.
- We'll discuss using R to assess a range of what are called Box-Cox power transformations, via plots, mainly.

11.10 The Ladder of Power Transformations

The key notion in re-expression of a single variable to obtain a distribution better approximated by the Normal or re-expression of an outcome in a simple regression model is that of a **ladder of power transformations**, which applies to any unimodal data.

| Power | Transformation |
|-------|-------------------------|
| 3 | x^3 |
| 2 | x^2 |
| 1 | x (unchanged) |
| 0.5 | $x^{0.5} = \sqrt{x}$ |
| 0 | $\ln x$ |
| -0.5 | $x^{-0.5} = 1/\sqrt{x}$ |
| -1 | $x^{-1} = 1/x$ |
| -2 | $x^{-2} = 1/x^2$ |

11.11 Using the Ladder

As we move further away from the *identity* function (power = 1) we change the shape more and more in the same general direction.

- For instance, if we try a logarithm, and this seems like too much of a change, we might try a square root instead.
- Note that this ladder (which like many other things is due to John Tukey) uses the logarithm for the “power zero” transformation rather than the constant, which is what x^0 actually is.
- If the variable x can take on negative values, we might take a different approach. If x is a count of something that could be zero, we often simply add 1 to x before transformation.

The ladder of power transformations is particularly helpful when we are confronted with data that shows skew.

- To handle right skew (where the mean exceeds the median) we usually apply powers below 1.
- To handle left skew (where the median exceeds the mean) we usually apply powers greater than 1.

The most common transformations are the square (power 2), the square root (power 1/2), the logarithm (power 0) and the inverse (power -1), and I usually restrict myself to those options in practical work.

11.12 Protein Consumption in the NNYFS data

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

```

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

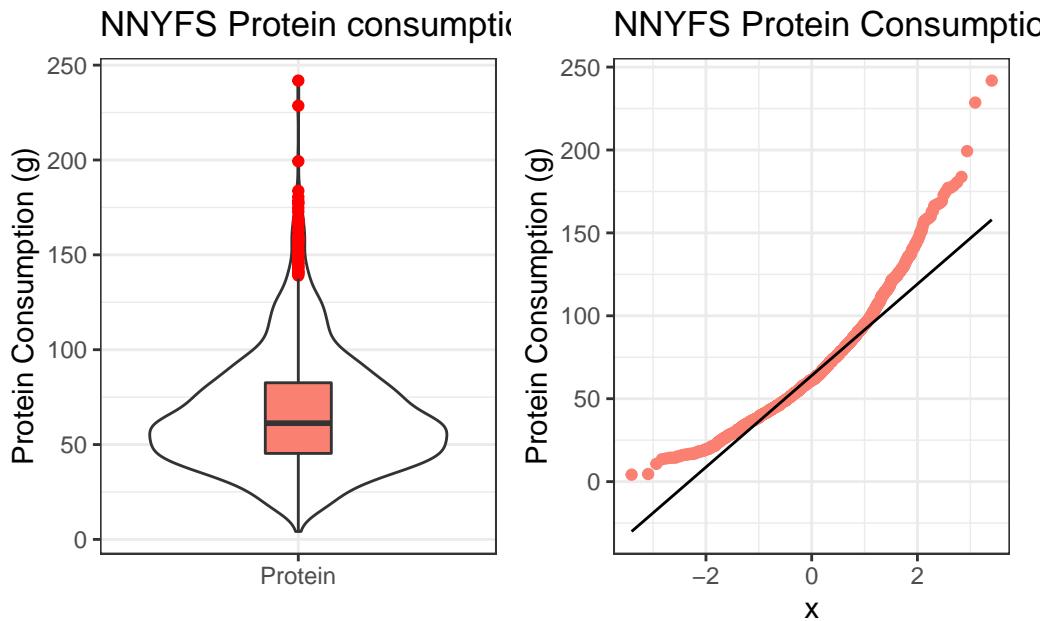
min      Q1 median      Q3    max      mean       sd     n missing
4.18  45.33 61.255 82.565 241.84 66.90148 30.96319 1518         0

p1 <- ggplot(nnyfs, aes(x = "Protein", y = protein)) +
  geom_violin() +
  geom_boxplot(width = 0.2, fill = "salmon",
                outlier.color = "red") +
  labs(title = "NNYFS Protein consumption",
       x = "", y = "Protein Consumption (g)")

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

p1 + p2

```



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

11.12.1 Using patchwork to compose plots

As we mentioned previously, I feel that the slickest approach to composing how a series of plots are placed together is available in the `patchwork` package. Here's another example.

```
res <- mosaic::favstats(~ protein, data = nnyfs)
bin_w <- 5 # specify binwidth

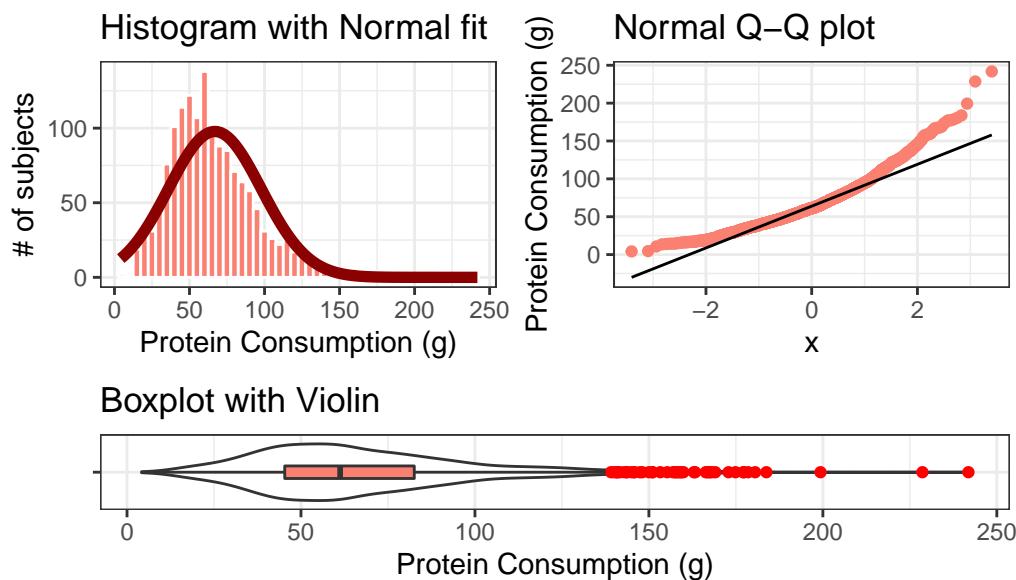
p1 <- ggplot(nnyfs, aes(x = protein)) +
  geom_histogram(binwidth = bin_w,
                 fill = "salmon",
                 col = "white") +
  stat_function(
    fun = function(x) dnorm(x, mean = res$mean,
                           sd = res$sd) *
      res$n * bin_w,
    col = "darkred", size = 2) +
  labs(title = "Histogram with Normal fit",
       x = "Protein Consumption (g)", y = "# of subjects")

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

p3 <- ggplot(nnyfs, aes(x = "", y = protein)) +
  geom_violin() +
  geom_boxplot(width = 0.2, fill = "salmon",
               outlier.color = "red") +
  coord_flip() +
  labs(title = "Boxplot with Violin",
       x = "", y = "Protein Consumption (g)")
```

```
p1 + p2 - p3 + plot_layout(ncol = 1, height = c(3, 1)) +
  plot_annotation(title = "NNYFS Protein Consumption")
```

NNYFS Protein Consumption



Again, the `patchwork` package repository at <https://patchwork.data-imaginist.com/index.html> has lots of nice examples to work from.

11.13 Can we transform the protein data?

As we've seen, the protein data are right skewed, and all of the values are strictly positive. If we want to use the tools of the Normal distribution to describe these data, we might try taking a step "down" our ladder from power 1 (raw data) to lower powers.

11.13.1 The Square Root

Would a square root applied to the protein data help alleviate that right skew?

```
res <- mosaic::favstats(~ sqrt(protein), data = nnyfs)
bin_w <- 1 # specify binwidth
```

```

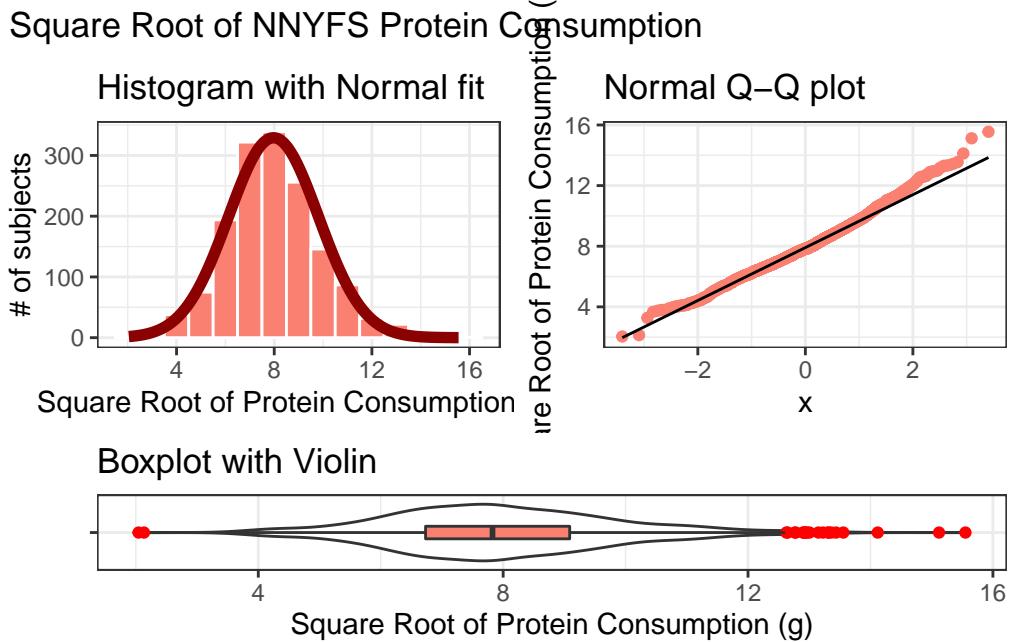
p1 <- ggplot(nnyfs, aes(x = sqrt(protein))) +
  geom_histogram(binwidth = bin_w,
                 fill = "salmon",
                 col = "white") +
  stat_function(
    fun = function(x) dnorm(x, mean = res$mean,
                            sd = res$sd) *
      res$n * bin_w,
    col = "darkred", size = 2) +
  labs(title = "Histogram with Normal fit",
       x = "Square Root of Protein Consumption (g)", y = "# of subjects")

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

p3 <- ggplot(nnyfs, aes(x = "", y = sqrt(protein))) +
  geom_violin() +
  geom_boxplot(width = 0.2, fill = "salmon",
               outlier.color = "red") +
  coord_flip() +
  labs(title = "Boxplot with Violin",
       x = "", y = "Square Root of Protein Consumption (g)")

p1 + p2 - p3 + plot_layout(ncol = 1, height = c(3, 1)) +
  plot_annotation(title = "Square Root of NNYFS Protein Consumption")

```



That looks like a more symmetric distribution, certainly, although we still have some outliers on the right side of the distribution. Should we take another step down the ladder?

11.13.2 The Logarithm

We might also try a logarithm of the energy circumference data. We can use either the natural logarithm (`log`, in R) or the base-10 logarithm (`log10`, in R) - either will have the same impact on skew.

```
res <- mosaic::favstats(~ log(protein), data = nnyfs)
bin_w <- 0.5 # specify binwidth

p1 <- ggplot(nnyfs, aes(x = log(protein))) +
  geom_histogram(binwidth = bin_w,
                 fill = "salmon",
                 col = "white") +
  stat_function(
    fun = function(x) dnorm(x, mean = res$mean,
                           sd = res$sd) *
      res$n * bin_w,
    col = "darkred", size = 2) +
```

```

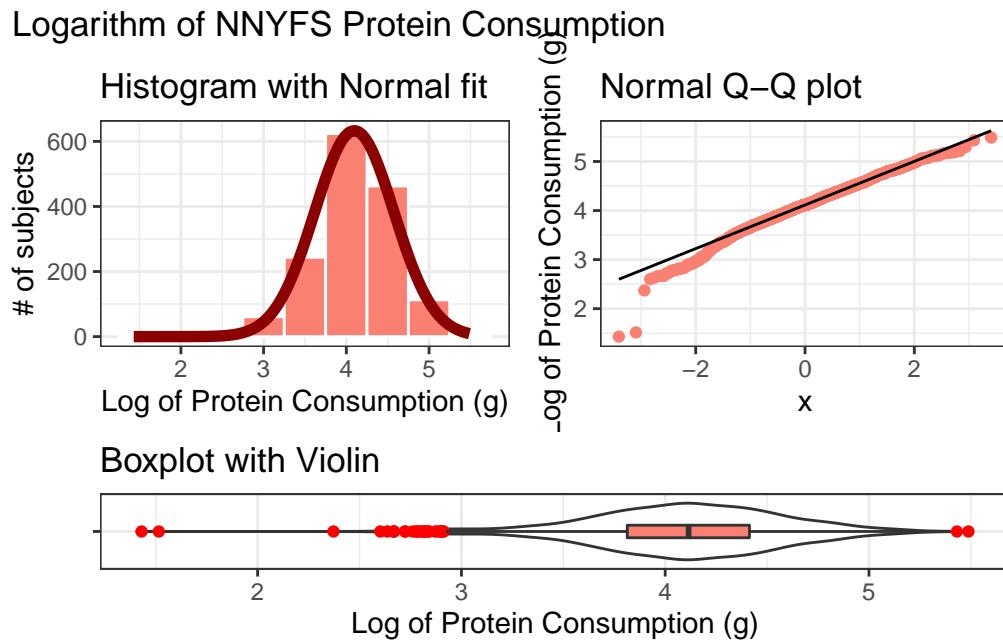
  labs(title = "Histogram with Normal fit",
       x = "Log of Protein Consumption (g)", y = "# of subjects")

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

p3 <- ggplot(nnyfs, aes(x = "", y = log(protein))) +
  geom_violin() +
  geom_boxplot(width = 0.2, fill = "salmon",
               outlier.color = "red") +
  coord_flip() +
  labs(title = "Boxplot with Violin",
       x = "", y = "Log of Protein Consumption (g)")

p1 + p2 - p3 + plot_layout(ncol = 1, height = c(3, 1)) +
  plot_annotation(title = "Logarithm of NNYFS Protein Consumption")

```



Now, it looks like we may have gone too far in the other direction. It looks like the square root

is a sensible choice to try to improve the fit of a Normal model to the protein consumption data.

11.13.3 This course uses Natural Logarithms, unless otherwise specified

In this course, we will assume the use of natural logarithms unless we specify otherwise. Following R's convention, we will use `log` for natural logarithms.

11.14 What if we considered all 9 available transformations?

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

p2 <- ggplot(nnyfs, aes(sample = protein^2)) +
  geom_qq(col = "salmon") +
  geom_qq_line(col = "black") +
  labs(title = "Square (power 2)",
       y = "Protein, Squared")

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

p4 <- ggplot(nnyfs, aes(sample = sqrt(protein))) +
  geom_qq(col = "salmon") +
  geom_qq_line(col = "black") +
  labs(title = "sqrt (power 0.5)",
       y = "Square Root of Protein")

p5 <- ggplot(nnyfs, aes(sample = log(protein))) +
  geom_qq(col = "salmon") +
  geom_qq_line(col = "black") +
  labs(title = "log (power 0)",
       y = "Natural Log of Protein")
```

```

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

p7 <- ggplot(nnyfs, aes(sample = 1/protein)) +
  geom_qq(col = "salmon") +
  geom_qq_line(col = "black") +
  labs(title = "Inverse (power -1)",
       y = "1/Protein")

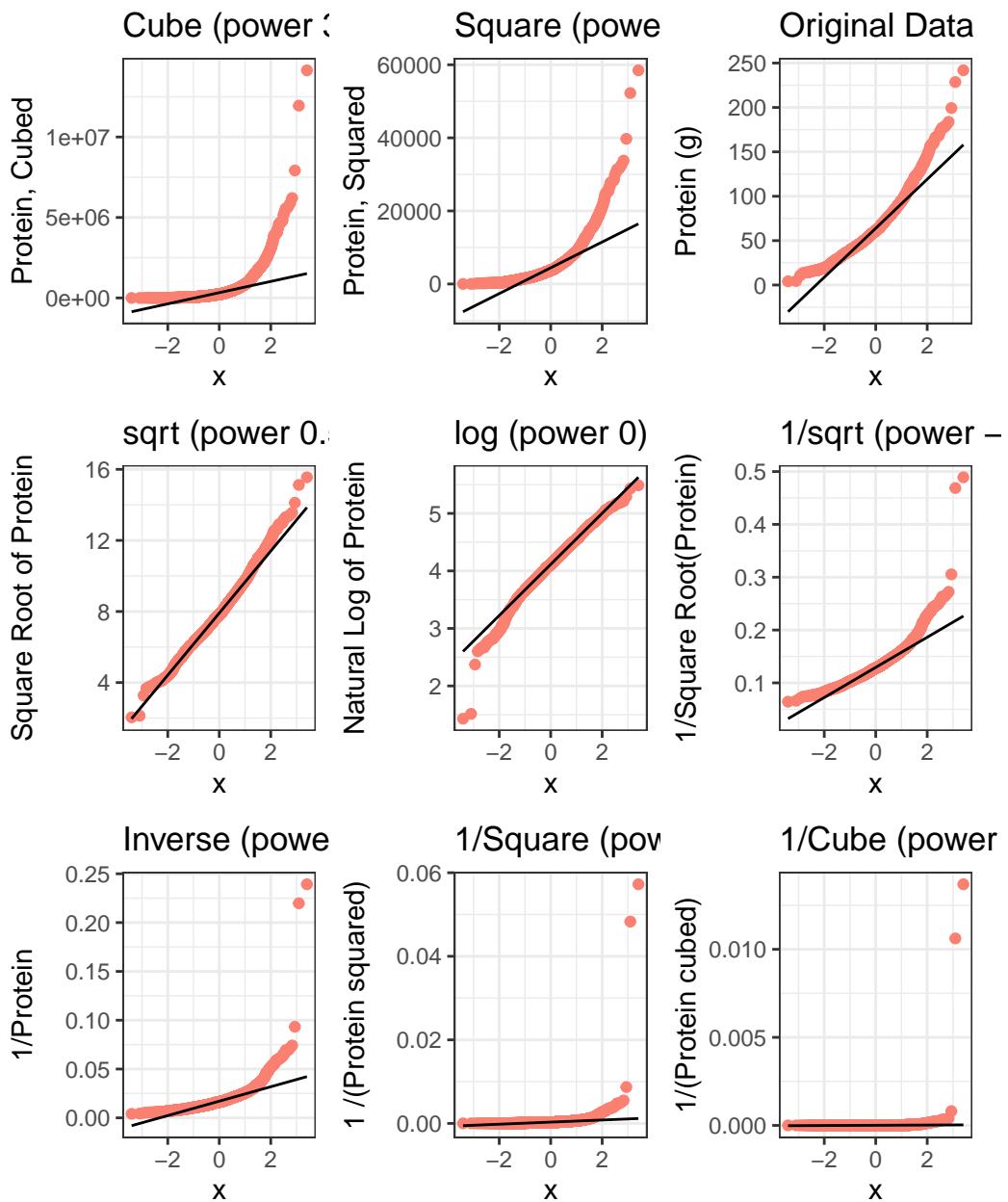
p8 <- ggplot(nnyfs, aes(sample = 1/(protein^2))) +
  geom_qq(col = "salmon") +
  geom_qq_line(col = "black") +
  labs(title = "1/Square (power -2)",
       y = "1 /(Protein squared)")

p9 <- ggplot(nnyfs, aes(sample = 1/(protein^3))) +
  geom_qq(col = "salmon") +
  geom_qq_line(col = "black") +
  labs(title = "1/Cube (power -3)",
       y = "1/(Protein cubed)")

p1 + p2 + p3 + p4 + p5 + p6 + p7 + p8 + p9 +
  plot_layout(nrow = 3) +
  plot_annotation(title = "Transformations of NNYFS Protein Consumption")

```

Transformations of NNYFS Protein Consumption



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

11.15 A Simulated Data Set

```
set.seed(431);
data2 <-
  data_frame(sample2 = 100*rbeta(n = 125, shape1 = 5, shape2 = 2))

Warning: `data_frame()` was deprecated in tibble 1.1.0.
Please use `tibble()` instead.
This warning is displayed once every 8 hours.
Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

If we'd like to transform these data so as to better approximate a Normal distribution, where should we start? What transformation do you suggest?

```
res <- mosaic::favstats(~ sample2, data = data2)
bin_w <- 4 # specify binwidth

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

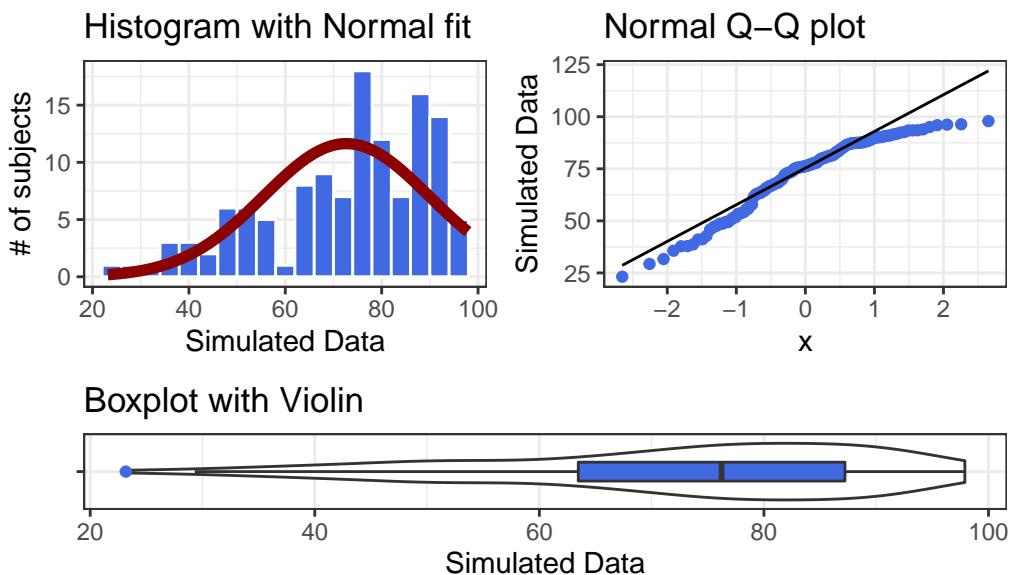
```

  labs(title = "Boxplot with Violin",
       x = "", y = "Simulated Data")

p1 + p2 - p3 + plot_layout(ncol = 1, height = c(3, 1)) +
  plot_annotation(title = "Simulated Data")

```

Simulated Data



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

```

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

p1 <- ggplot(data2, aes(x = sample2^2)) +
  geom_histogram(binwidth = bin_w,
                 fill = "royalblue",
                 col = "white") +
  stat_function(
    fun = function(x) dnorm(x, mean = res$mean,
                           sd = res$sd) *
      res$n * bin_w,
    col = "darkred", size = 2) +

```

```

  labs(title = "Histogram with Normal fit",
       x = "Squared Simulated Data", y = "# of subjects")

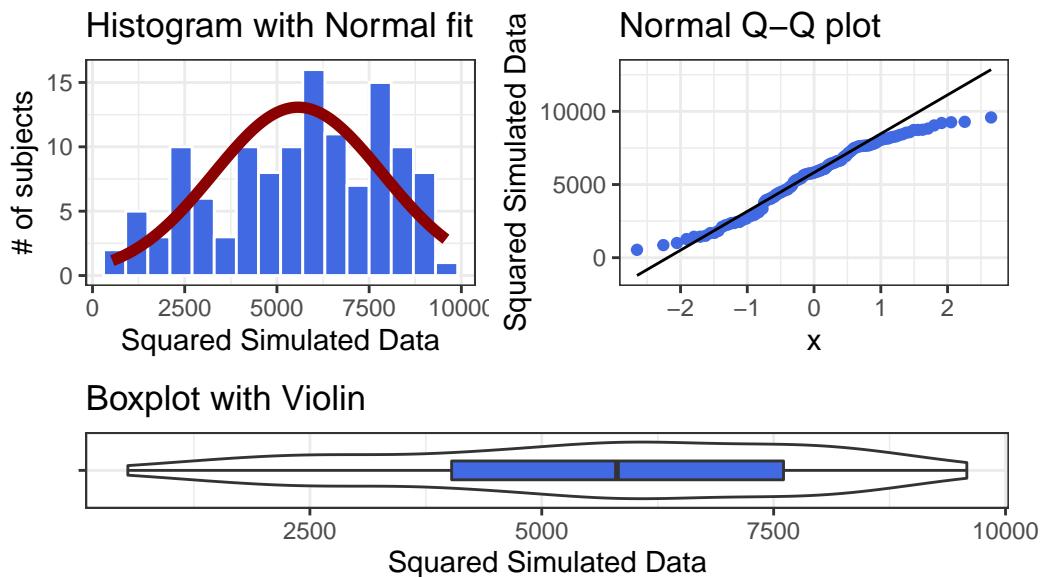
p2 <- ggplot(data2, aes(sample = sample2^2)) +
  geom_qq(col = "royalblue") +
  geom_qq_line(col = "black") +
  labs(title = "Normal Q-Q plot",
       y = "Squared Simulated Data")

p3 <- ggplot(data2, aes(x = "", y = sample2^2)) +
  geom_violin() +
  geom_boxplot(width = 0.3, fill = "royalblue",
               outlier.color = "royalblue") +
  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?

```

res <- mosaic::favstats(~ sample2^3, data = data2)
bin_w <- 100000 # specify binwidth

p1 <- ggplot(data2, aes(x = sample2^3)) +
  geom_histogram(binwidth = bin_w,
                 fill = "royalblue",
                 col = "white") +
  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")

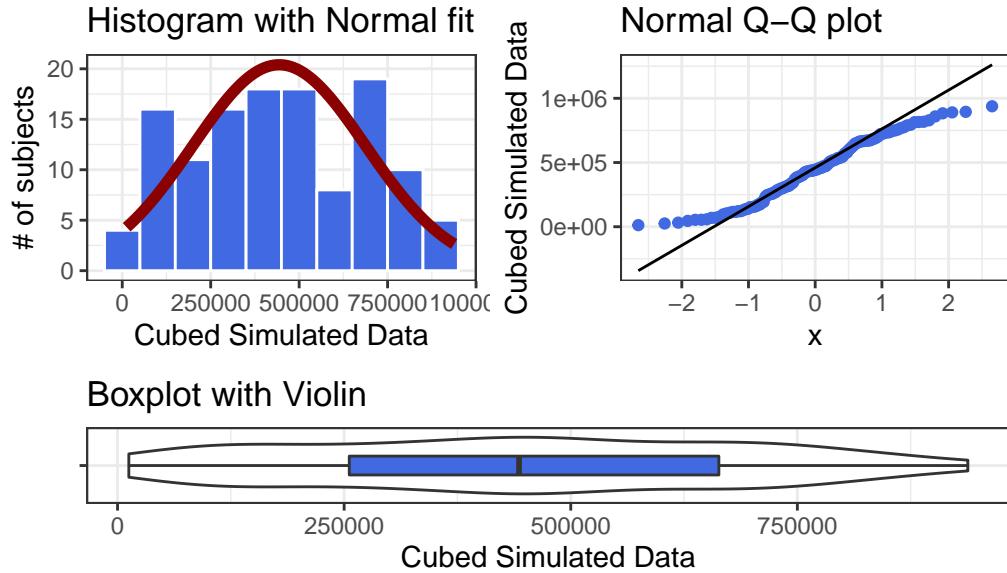
p2 <- ggplot(data2, aes(sample = sample2^3)) +
  geom_qq(col = "royalblue") +
  geom_qq_line(col = "black") +
  labs(title = "Normal Q-Q plot",
       y = "Cubed Simulated Data")

p3 <- ggplot(data2, aes(x = "", y = sample2^3)) +
  geom_violin() +
  geom_boxplot(width = 0.3, fill = "royalblue",
               outlier.color = "royalblue") +
  coord_flip() +
  labs(title = "Boxplot with Violin",
       x = "", y = "Cubed Simulated Data")

p1 + p2 - p3 + plot_layout(ncol = 1, height = c(3, 1)) +
  plot_annotation(title = "Cubed Simulated Data")

```

Cubed Simulated Data



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

11.16 What if we considered all 9 available transformations?

```
p1 <- ggplot(data2, aes(sample = sample2^3)) +  
  geom_qq(col = "royalblue") +  
  geom_qq_line(col = "black") +  
  labs(title = "Cube (power 3)")  
  
p2 <- ggplot(data2, aes(sample = sample2^2)) +  
  geom_qq(col = "royalblue") +  
  geom_qq_line(col = "black") +  
  labs(title = "Square (power 2)")  
  
p3 <- ggplot(data2, aes(sample = sample2)) +  
  geom_qq(col = "royalblue") +
```

```

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

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

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

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

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

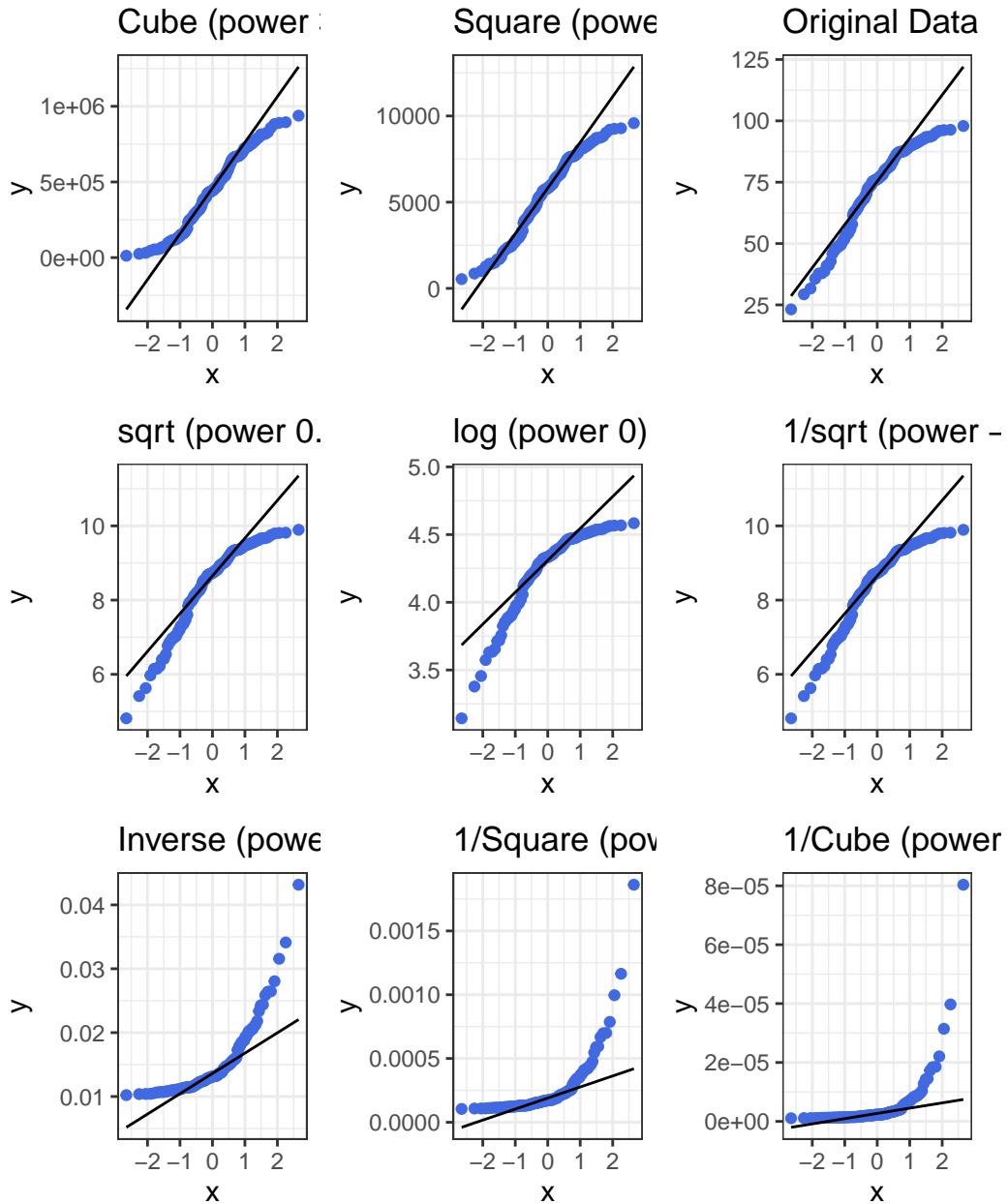
p8 <- ggplot(data2, aes(sample = 1/(sample2^2))) +
  geom_qq(col = "royalblue") +
  geom_qq_line(col = "black") +
  labs(title = "1/Square (power -2)")

p9 <- ggplot(data2, aes(sample = 1/(sample2^3))) +
  geom_qq(col = "royalblue") +
  geom_qq_line(col = "black") +
  labs(title = "1/Cube (power -3)")

p1 + p2 + p3 + p4 + p5 + p6 + p7 + p8 + p9 +
  plot_layout(nrow = 3) +
  plot_annotation(title = "Transformations of Simulated Sample")

```

Transformations of Simulated Sample



Again, either the cube or the square looks like best choice here, in terms of creating a more symmetric (albeit light-tailed) distribution.

12 Straight Line Models

12.1 Setup: Packages Used Here

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

library(broom)
library(knitr)
library(patchwork)
library(tidyverse)

theme_set(theme_bw())
```

12.2 Assessing A Scatterplot

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

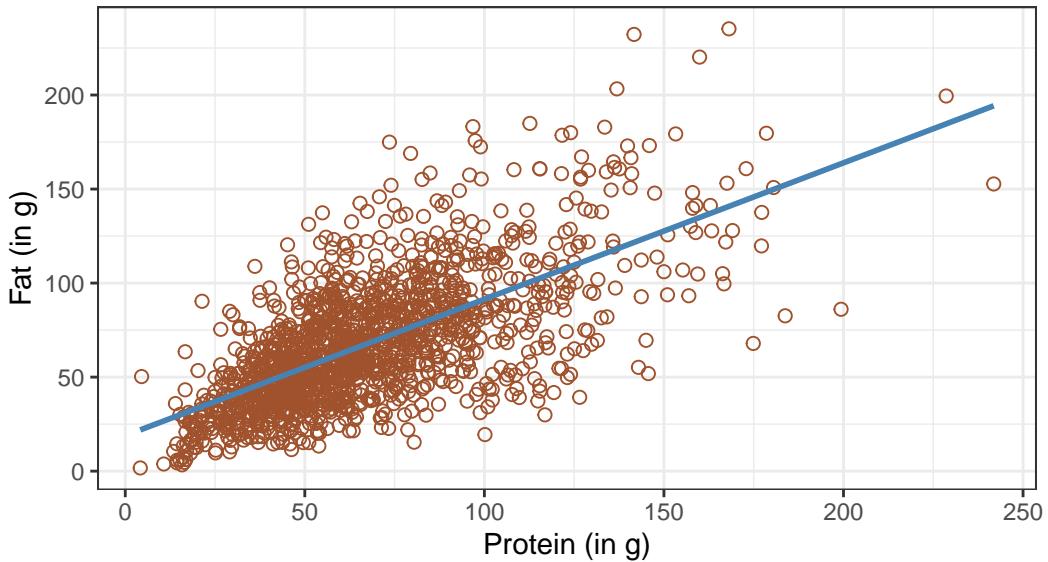
```
nnyfs <- read_rds("data/nnyfs.Rds")
```

We'll begin our investigation, as we always should, by drawing a relevant picture. For the association of two quantitative variables, a **scatterplot** is usually the right start. Each subject in the `nnyfs` data is represented by one of the points below. To the plot, I've also used `geom_smooth` to add a straight line regression model, which we'll discuss later.

```
ggplot(data = nnyfs, aes(x = protein, y = fat)) +
  geom_point(shape = 1, size = 2, col = "sienna") +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE, col = "steelblue") +
  labs(title = "Protein vs. Fat consumption",
       subtitle = "Children in the NNYFS data, with fitted straight line regression model",
       x = "Protein (in g)", y = "Fat (in g)")
```

Protein vs. Fat consumption

Children in the NNYFS data, with fitted straight line regression model



Here, I've arbitrarily decided to place `fat` on the vertical axis, and `protein` on the horizontal. Fitting a prediction model to this scatterplot will then require that we predict `fat` on the basis of `protein`.

In this case, the pattern appears to be:

1. **direct**, or positive, in that the values of the x variable (`protein`) increase, so do the values of the y variable (`fat`). Essentially, it appears that subjects who consumed more protein also consumed more fat, but we don't know cause and effect here.
2. fairly **linear** in that most of the points cluster around what appears to be a pattern which is well-fitted by a straight line.
3. moderately **strong** in that the range of values for `fat` associated with any particular value of `protein` is fairly tight. If we know someone's protein consumption, that should meaningfully improve our ability to predict their fat consumption, among the subjects in these data.
4. that we see some unusual or **outlier** values, further away from the general pattern of most subjects shown in the data.

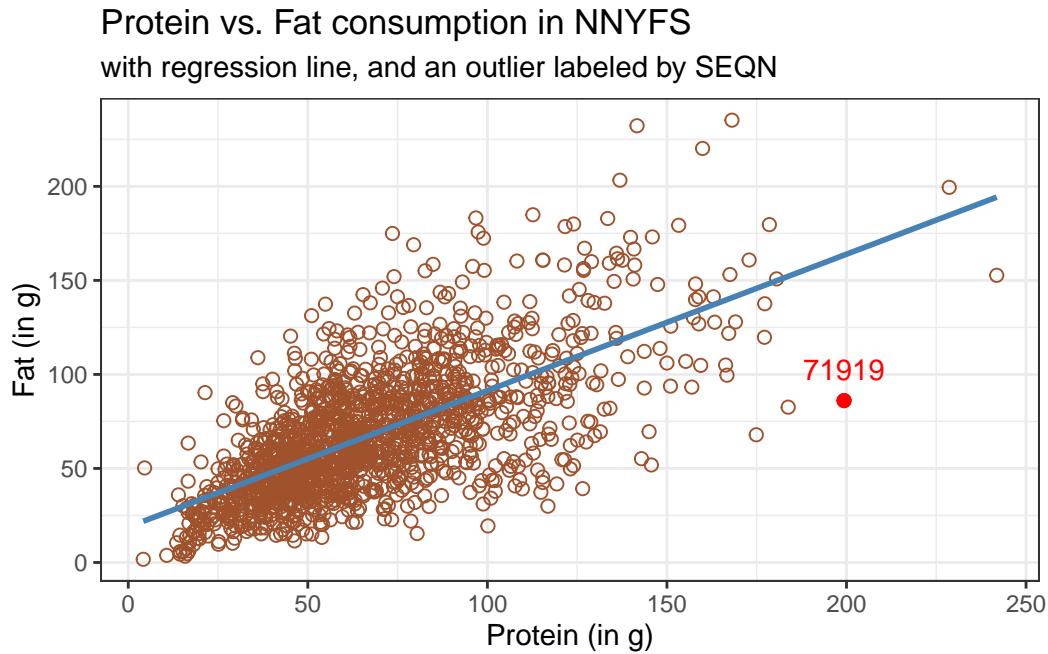
12.2.1 Highlighting an unusual point

Consider the subject with protein consumption close to 200 g, whose fat consumption is below 100 g. That's well below the prediction of the linear model for example. We can identify the subject because it is the only person with `protein` > 190 and `fat` < 100 with BMI > 35 and

`waist.circ < 70`. So I'll create a subset of the `nnyfs` data containing the point that meets that standard, and then add a red point and a label to the plot.

```
# identify outlier and place it in data frame s1
s1 <- filter(nnyfs, protein > 190 & fat < 100)

ggplot(data = nnyfs, aes(x = protein, y = fat)) +
  geom_point(shape = 1, size = 2, col = "sienna") +
  geom_smooth(method = "lm", se = FALSE, formula = y ~ x, col = "steelblue") +
  geom_point(data = s1, size = 2, col = "red") +
  geom_text(data = s1, label = s1$SEQN,
            vjust = -1, col = "red") +
  labs(title = "Protein vs. Fat consumption in NNYFS",
       subtitle = "with regression line, and an outlier labeled by SEQN",
       x = "Protein (in g)", y = "Fat (in g)")
```



While this subject is hardly the only unusual point in the data set, it is one of the more unusual ones, in terms of its vertical distance from the regression line. We can identify the subject by printing (part of) the tibble we created.

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

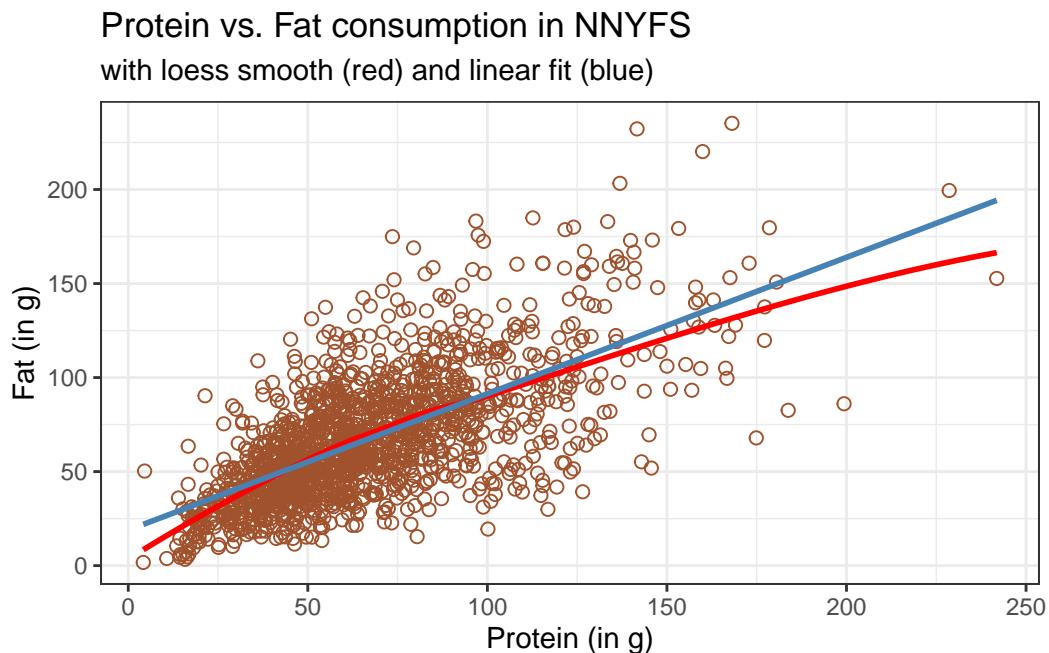
| SEQN | sex | race_eth | age_child | protein | fat |
|-------|--------|----------------------|-----------|---------|-------|
| 71919 | Female | 2_White Non-Hispanic | 14 | 199.33 | 86.08 |

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

12.2.2 Adding a Scatterplot Smooth using loess

Next, we'll use the **loess** procedure to fit a smooth curve to the data, which attempts to capture the general pattern.

```
ggplot(data = nnyfs, aes(x = protein, y = fat)) +
  geom_point(shape = 1, size = 2, col = "sienna") +
  geom_smooth(method = "loess", se = FALSE, formula = y ~ x, col = "red") +
  geom_smooth(method = "lm", se = FALSE, formula = y ~ x, col = "steelblue") +
  labs(title = "Protein vs. Fat consumption in NNYFS",
       subtitle = "with loess smooth (red) and linear fit (blue)",
       x = "Protein (in g)", y = "Fat (in g)")
```



This “loess” smooth curve is fairly close to the straight line fit, indicating that perhaps a linear regression model might fit the data well.

A **loess smooth** is a method of fitting a local polynomial regression model that R uses as its default smooth for scatterplots with fewer than 1000 observations. Think of the loess as a way of fitting a curve to data by tracking (at point x) the points within a neighborhood of point x , with more emphasis given to points near x . It can be adjusted by tweaking two specific parameters, in particular:

- a **span** parameter (defaults to 0.75) which is also called α in the literature, that controls the degree of smoothing (essentially, how large the neighborhood should be), and
- a **degree** parameter (defaults to 2) which specifies the degree of polynomial to be used. Normally, this is either 1 or 2 - more complex functions are rarely needed for simple scatterplot smoothing.

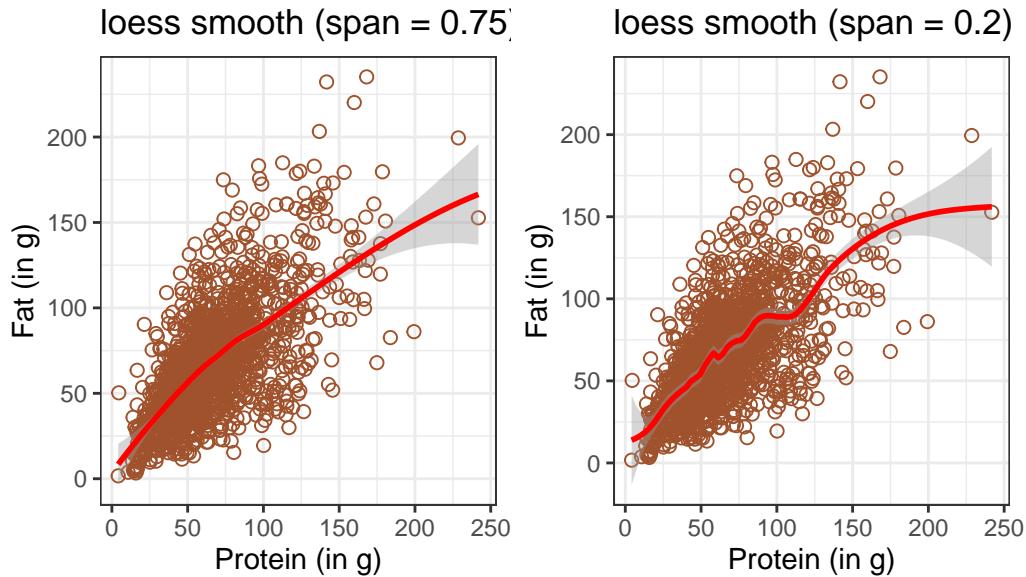
In addition to the curve, smoothing procedures can also provide confidence intervals around their main fitted line. Consider the following plot, which adjusts the span and also adds in the confidence intervals.

```
p1 <- ggplot(data = nnyfs, aes(x = protein, y = fat)) +
  geom_point(shape = 1, size = 2, col = "sienna") +
  geom_smooth(method = "loess", span = 0.75, se = TRUE,
              col = "red", formula = y ~ x) +
  labs(title = "loess smooth (span = 0.75)",
       x = "Protein (in g)", y = "Fat (in g)")

p2 <- ggplot(data = nnyfs, aes(x = protein, y = fat)) +
  geom_point(shape = 1, size = 2, col = "sienna") +
  geom_smooth(method = "loess", span = 0.2, se = TRUE,
              col = "red", formula = y ~ x) +
  labs(title = "loess smooth (span = 0.2)",
       x = "Protein (in g)", y = "Fat (in g)")

p1 + p2 +
  plot_annotation(title = "Impact of adjusting loess smooth span: NNYFS")
```

Impact of adjusting loess smooth span: NNYFS



By reducing the size of the span, the plot on the right shows a somewhat less “smooth” function than the plot on the left.

12.2.3 What Line Does R Fit?

Returning to the linear regression model, how can we, mathematically, characterize that line? As with any straight line, our model equation requires us to specify two parameters: a slope and an intercept (sometimes called the y-intercept.)

To identify the equation R used to fit this line (using the method of least squares), we use the `lm` command

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

```
Call:  
lm(formula = fat ~ protein, data = nnyfs)  
  
Coefficients:  
(Intercept)      protein  
    18.8945        0.7251
```

So the fitted line is specified as

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

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

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

```
Call:  
lm(formula = fat ~ protein, data = nnyfs)  
  
Residuals:  
    Min      1Q  Median      3Q     Max  
-77.798 -14.841 -2.449  13.601 110.597  
  
Coefficients:  
            Estimate Std. Error t value Pr(>|t|)  
(Intercept) 18.8945     1.5330   12.32 <2e-16 ***  
protein      0.7251     0.0208   34.87 <2e-16 ***  
---  
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 25.08 on 1516 degrees of freedom  
Multiple R-squared:  0.4451,    Adjusted R-squared:  0.4447  
F-statistic: 1216 on 1 and 1516 DF,  p-value: < 2.2e-16
```

The way we'll usually summarize the estimated coefficients of a linear model is to use the `broom` package's `tidy` function to put the coefficient estimates into a tibble.

```
tidy(lm(fat ~ protein, data = nnyfs),  
     conf.int = TRUE, conf.level = 0.95) |>  
kable(digits = 3)
```

| term | estimate | std.error | statistic | p.value | conf.low | conf.high |
|-------------|----------|-----------|-----------|---------|----------|-----------|
| (Intercept) | 18.895 | 1.533 | 12.325 | 0 | 15.887 | 21.902 |
| protein | 0.725 | 0.021 | 34.868 | 0 | 0.684 | 0.766 |

We can also summarize the quality of fit in a linear model using the `broom` package's `glance` function. For now, we'll focus our attention on just one of the many summaries available for a linear model from `glance`: the R-squared value.

```
glance(lm(fat ~ protein, data = nnyfs)) |> select(r.squared) |>  
kable(digits = 3)
```

| r.squared |
|-----------|
| 0.445 |

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

For now, it will suffice to understand the following:

- The outcome variable in this model is `fat`, and the predictor variable is `protein`.
- The straight line model for these data fitted by least squares is $\text{fat} = 18.9 + 0.725 \text{protein}$
- The slope of `protein` is positive, which indicates that as `protein` increases, we expect that `fat` will also increase. Specifically, we expect that for every additional gram of protein consumed, the fat consumption will be 0.725 gram larger.
- The multiple R-squared (squared correlation coefficient) is 0.445, which implies that 44.5% of the variation in `fat` is explained using this linear model with `protein`.
- This also implies that the Pearson correlation between `fat` and `protein` is the square root of 0.445, or 0.667. More on the Pearson correlation soon.

So, if we plan to use a simple (least squares) linear regression model to describe fat consumption as a function of protein consumption in the NNYFS data, does it look like a least squares (or linear regression) model will be an effective choice?

12.3 Correlation Coefficients

Two different correlation measures are worth our immediate attention.

- The one most often used is called the *Pearson* correlation coefficient, and is symbolized with the letter r or sometimes the Greek letter rho (ρ).
- Another tool is the Spearman rank correlation coefficient, also occasionally symbolized by ρ .

For the `nnyfs` data, the Pearson correlation of `fat` and `protein` can be found using the `cor()` function.

```
nnyfs |> select(fat, protein) |> cor()
```

```
      fat    protein
fat    1.0000000 0.6671209
protein 0.6671209 1.0000000
```

Note that the correlation of any variable with itself is 1, and that the correlation of `fat` with `protein` is the same regardless of whether you enter `fat` first or `protein` first.

12.4 The Pearson Correlation Coefficient

Suppose we have n observations on two variables, called X and Y . The Pearson correlation coefficient assesses how well the relationship between X and Y can be described using a linear function.

- The Pearson correlation is **dimension-free**.
- It falls between -1 and +1, with the extremes corresponding to situations where all the points in a scatterplot fall exactly on a straight line with negative and positive slopes, respectively.
- A Pearson correlation of zero corresponds to the situation where there is no linear association.
- Unlike the estimated slope in a regression line, the sample correlation coefficient is symmetric in X and Y , so it does not depend on labeling one of them (Y) the response variable, and one of them (X) the predictor.

Suppose we have n observations on two variables, called X and Y , where \bar{X} is the sample mean of X and s_x is the standard deviation of X . The **Pearson** correlation coefficient r_{XY} is:

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

12.5 Studying Correlation through 6 Examples

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

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

```
Rows: 277 Columns: 3
-- Column specification -----
Delimiter: ","
chr (1): set
dbl (2): x, y

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
summary(correx1)
```

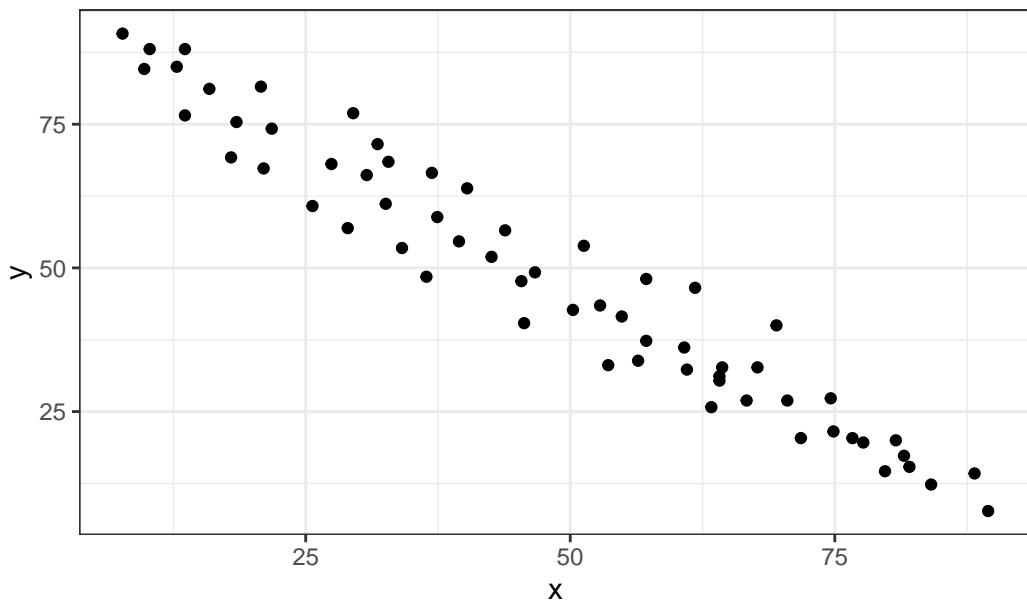
| | set | x | y |
|---------|-----------|----------------|----------------|
| Length: | 277 | Min. : 5.897 | Min. : 7.308 |
| Class : | character | 1st Qu.:29.487 | 1st Qu.:30.385 |
| Mode : | character | Median :46.154 | Median :46.923 |
| | | Mean :46.529 | Mean :49.061 |
| | | 3rd Qu.:63.333 | 3rd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

12.5.1 Data Set Alex

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

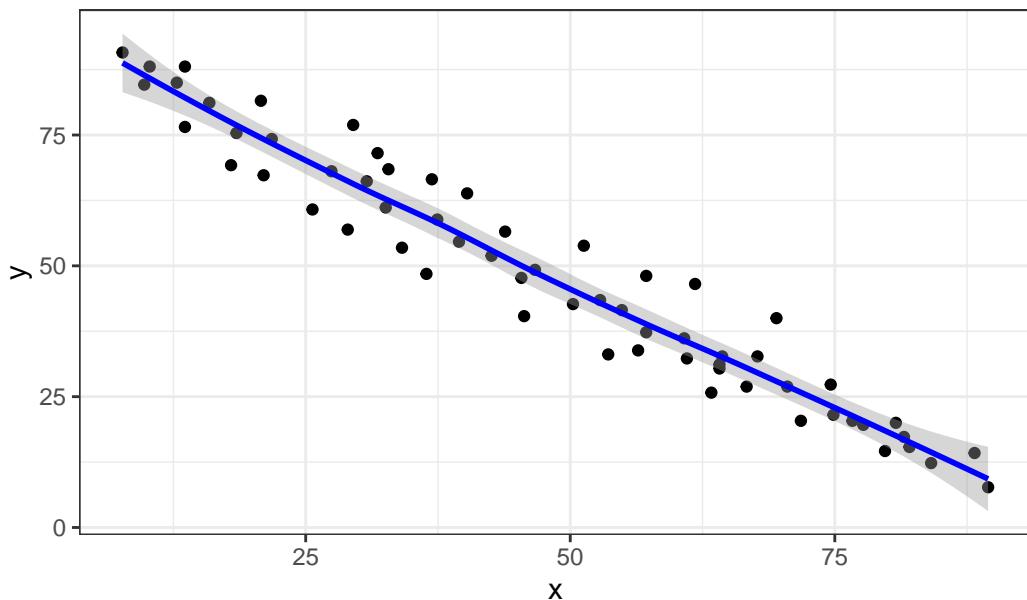
```
ggplot(filter(correx1, set == "Alex"), aes(x = x, y = y)) +
  geom_point() +
  labs(title = "correx1: Data Set Alex")
```

correx1: Data Set Alex



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

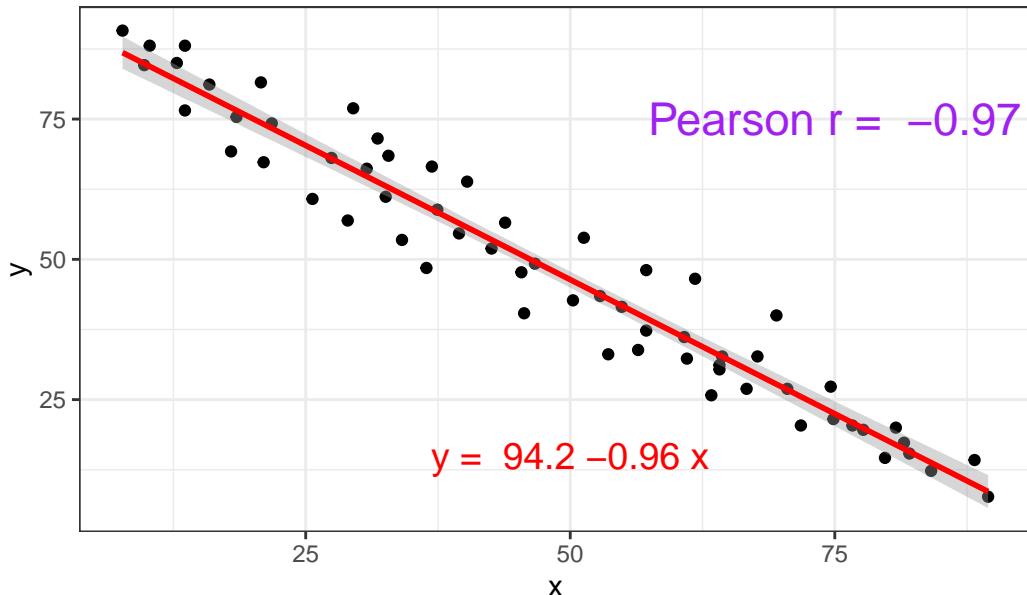
correx1: Alex, with loess smooth



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

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

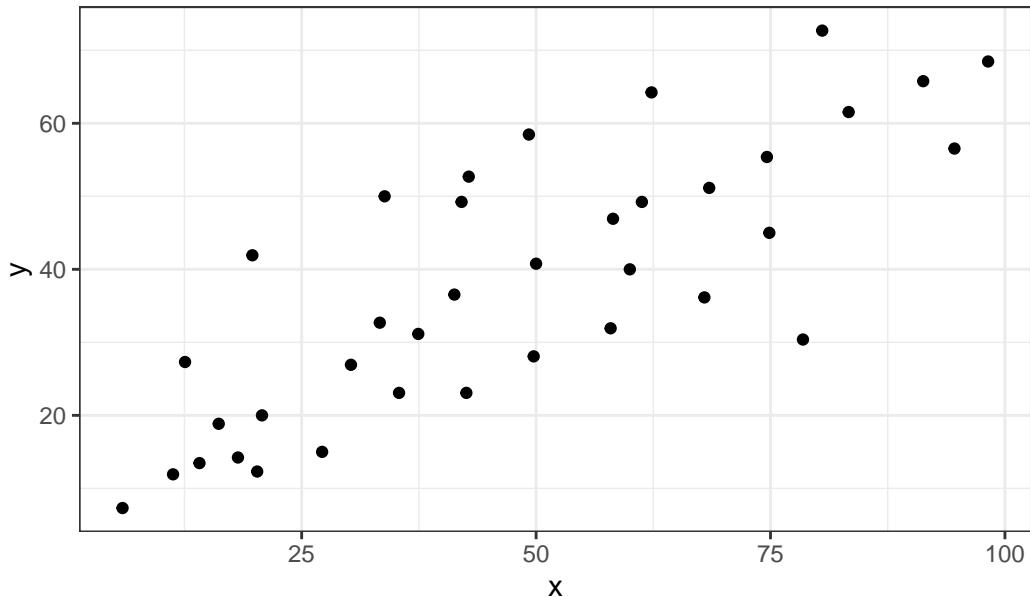
correx1: Alex, with Fitted Linear Model



12.5.2 Data Set Bonnie

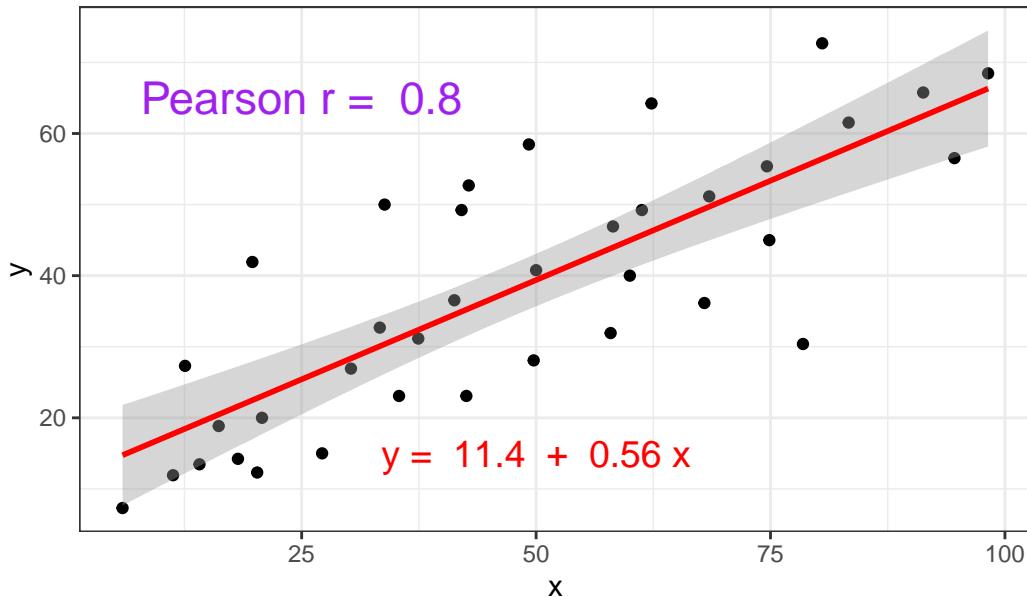
```
setB <- dplyr::filter(correx1, set == "Bonnie")  
  
ggplot(setB, aes(x = x, y = y)) +  
  geom_point() +  
  labs(title = "correx1: Data Set Bonnie")
```

correx1: Data Set Bonnie



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

correx1: Bonnie, with Fitted Linear Model



12.5.3 Correlations for All Six Data Sets in the Correx1 Example

Let's look at the Pearson correlations associated with each of the six data sets contained in the `correx1` example.

```
tab1 <- correx1 |>
  group_by(set) |>
  summarise("Pearson r" = round(cor(x, y, use="complete"), 2))

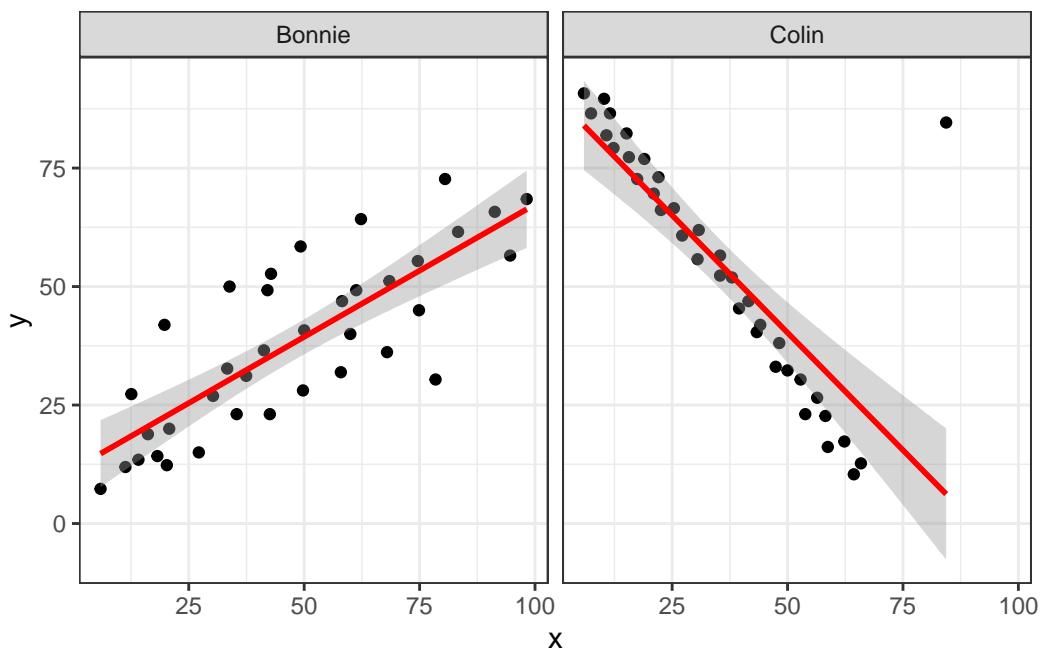
knitr::kable(tab1)
```

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

12.5.4 Data Set Colin

It looks like the picture for Colin should be very similar (in terms of scatter) to the picture for Bonnie, except that Colin will have a negative slope, rather than the positive one Bonnie has. Is that how this plays out?

```
setBC <- filter(correx1, set == "Bonnie" | set == "Colin")  
  
ggplot(setBC, aes(x = x, y = y)) +  
  geom_point() +  
  geom_smooth(method = "lm", formula = y ~ x, col = "red") +  
  facet_wrap(~ set)
```

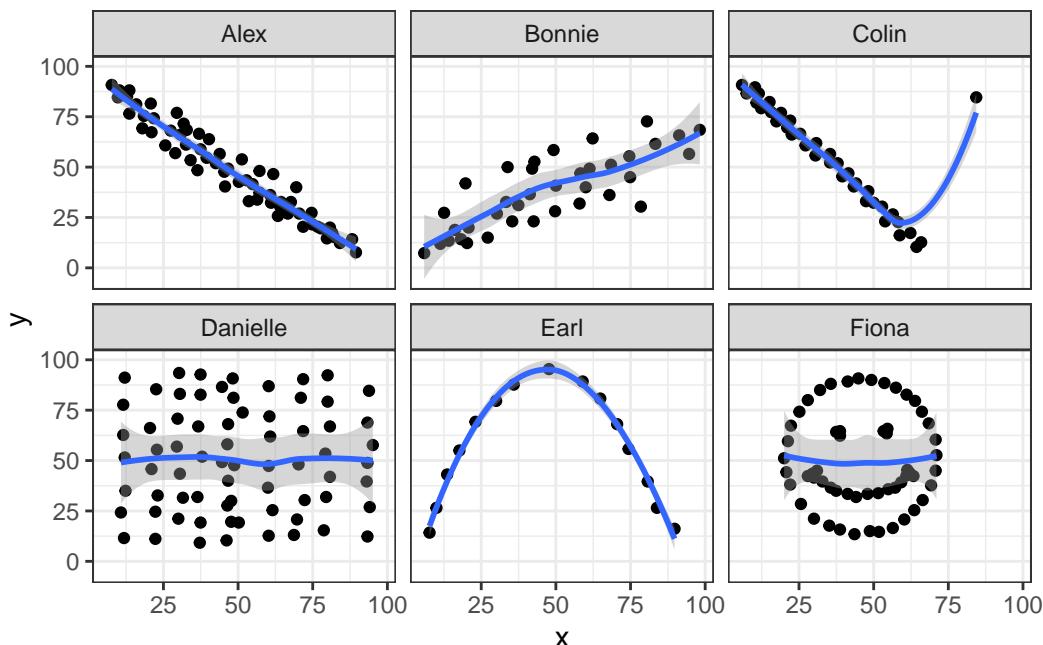


Uh, oh. It looks like the point in Colin at the top right is twisting what would otherwise be a very straight regression model with an extremely strong negative correlation. There's no better way to look for outliers than to examine the scatterplot.

12.5.5 Draw the Picture!

We've seen that Danielle, Earl and Fiona all show Pearson correlations of essentially zero. However, the three data sets look very different in a scatterplot.

```
ggplot(correx1, aes(x = x, y = y)) +
  geom_point() +
  geom_smooth(method = "loess", formula = y ~ x) +
  facet_wrap(~ set)
```



When we learn that the correlation is zero, we tend to assume we have a picture like the Danielle data set. If Danielle were our real data, we might well think that x would be of little use in predicting y .

- But what if our data looked like Earl? In the Earl data set, x is incredibly helpful in predicting y , but we can't use a straight line model - instead, we need a non-linear modeling approach.
- You'll recall that the Fiona data set also had a Pearson correlation of zero. But here, the picture is rather more interesting.

So, remember, draw the appropriate scatterplot whenever you make use of a correlation coefficient.

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

12.6 Estimating Correlation from Scatterplots

The correx2 data set is designed to help you calibrate yourself a bit in terms of estimating a correlation from a scatterplot. There are 11 data sets buried within the correx2 example, and they are labeled by their Pearson correlation coefficients, ranging from $r = 0.01$ to $r = 0.999$

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

Rows: 582 Columns: 4

-- Column specification -----

Delimiter: ","

chr (1): set

dbl (3): x, y, group

i Use `spec()` to retrieve the full column specification for this data.

i Specify the column types or set `show_col_types = FALSE` to quiet this message.

```
correx2 |>
  group_by(set) |>
  summarise(cor = round(cor(x, y, use="complete"), 3))
```

A tibble: 11 x 2

set cor

<chr> <dbl>

1 Set 01 0.01

2 Set 10 0.102

3 Set 20 0.202

4 Set 30 0.301

5 Set 40 0.403

6 Set 50 0.499

7 Set 60 0.603

8 Set 70 0.702

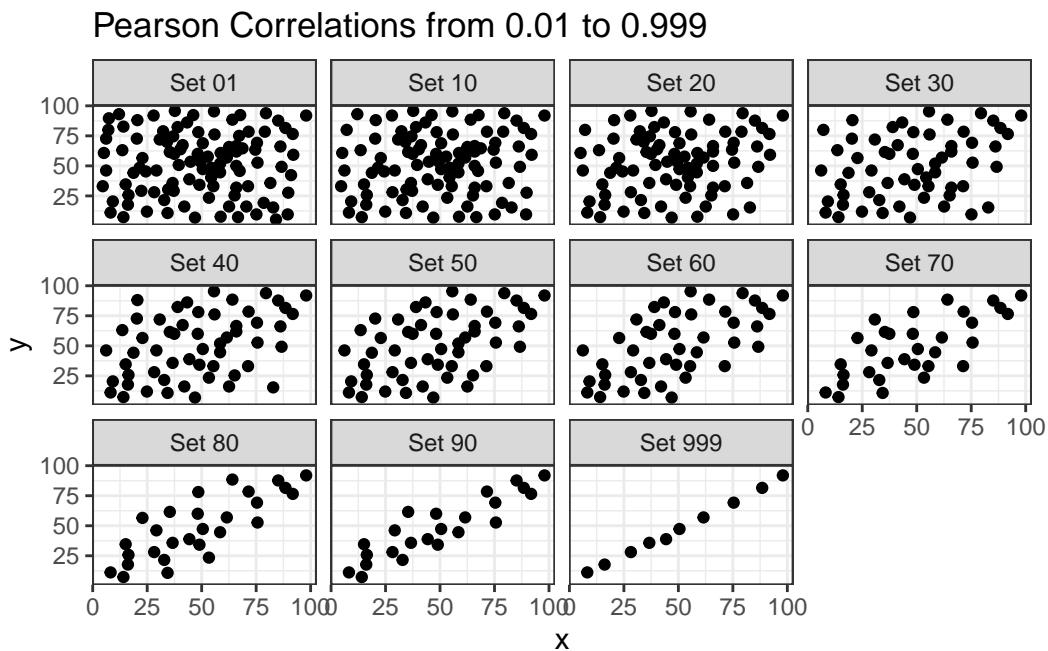
9 Set 80 0.799

10 Set 90 0.902

11 Set 999 0.999

Here is a plot of the 11 data sets, showing the increase in correlation from 0.01 (in Set 01) to 0.999 (in Set 999).

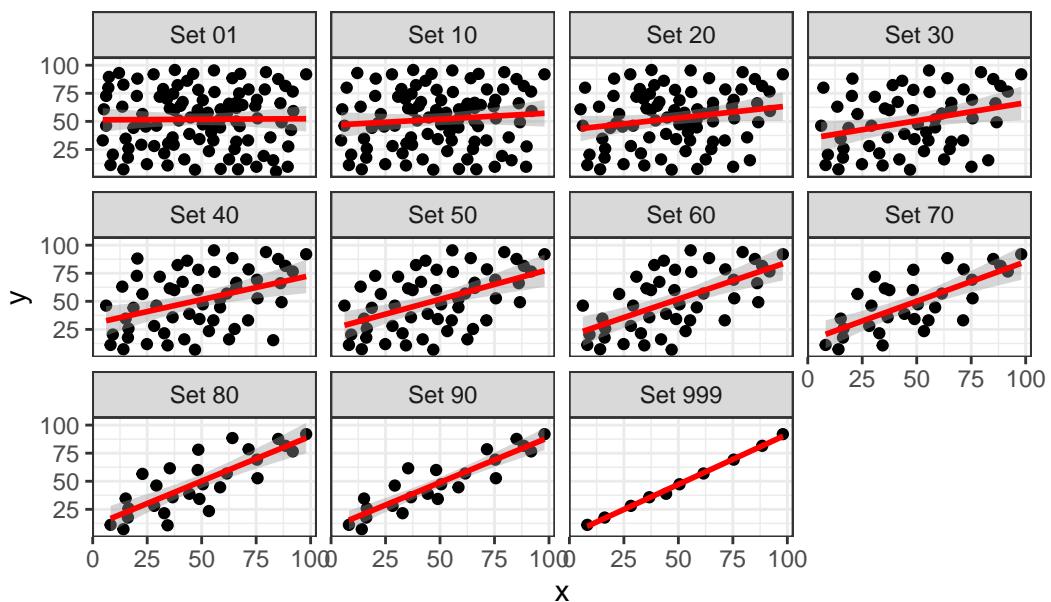
```
ggplot(correx2, aes(x = x, y = y)) +
  geom_point() +
  facet_wrap(~ set) +
  labs(title = "Pearson Correlations from 0.01 to 0.999")
```



Note that R will allow you to fit a straight line model to any of these relationships, no matter how appropriate it might be to do so.

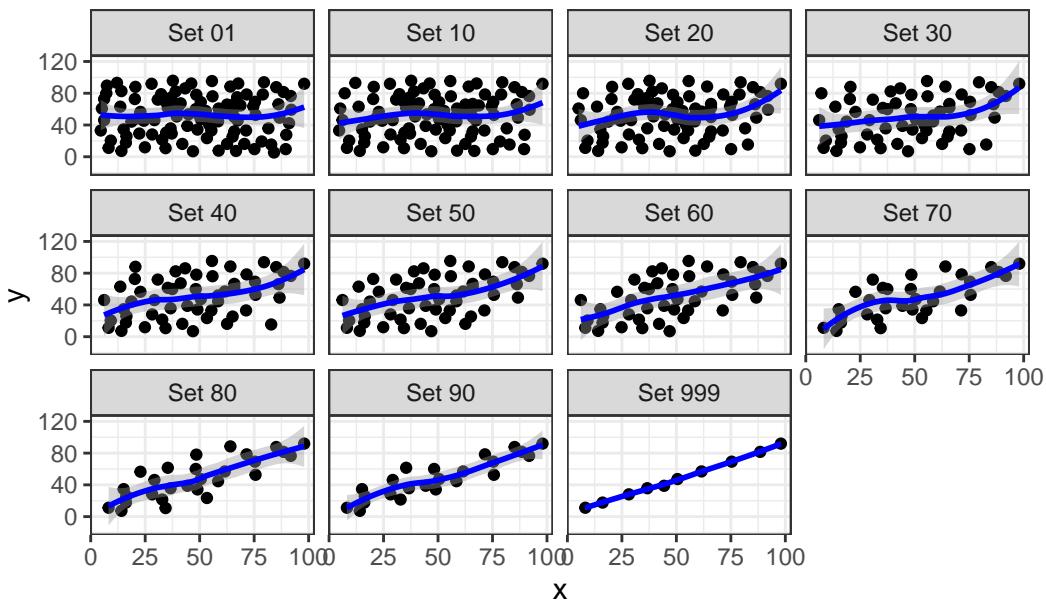
```
ggplot(correx2, aes(x = x, y = y)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, col = "red") +
  facet_wrap(~ set) +
  labs(title = "R will fit a straight line to anything.")
```

R will fit a straight line to anything.



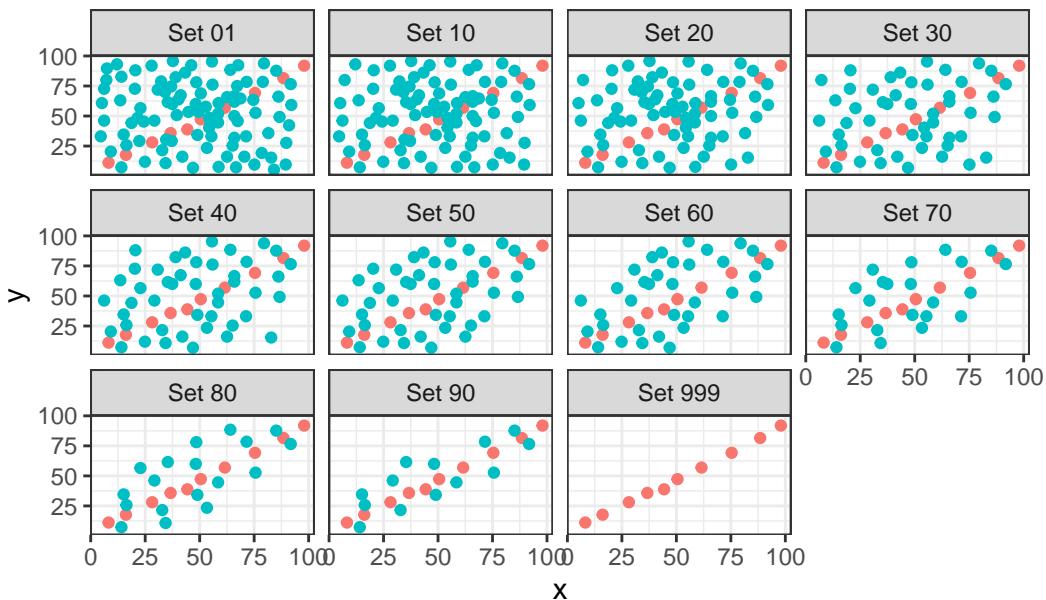
```
ggplot(correx2, aes(x = x, y = y)) +  
  geom_point() +  
  geom_smooth(col = "blue") +  
  facet_wrap(~ set) +  
  labs(title = "Even if a loess smooth suggests non-linearity.")
```

Even if a loess smooth suggests non-linearity.



```
ggplot(correx2, aes(x = x, y = y, color = factor(group))) +  
  geom_point() +  
  guides(color = "none") +  
  facet_wrap(~ set) +  
  labs(title = "Note: The same 10 points (in red) are in each plot.")
```

Note: The same 10 points (in red) are in each plot.



Note that the same 10 points are used in each of the data sets. It's always possible that a lurking subgroup of the data within a scatterplot follows a very strong linear relationship. This is why it's so important (and difficult) not to go searching for such a thing without a strong foundation of logic, theory and prior empirical evidence.

12.7 The Spearman Rank Correlation

The Spearman rank correlation coefficient is a rank-based measure of statistical dependence that assesses how well the relationship between X and Y can be described using a **monotone function** even if that relationship is not linear.

- A monotone function preserves order, that is, Y must either be strictly increasing as X increases, or strictly decreasing as X increases.
- A Spearman correlation of 1.0 indicates simply that as X increases, Y always increases.
- Like the Pearson correlation, the Spearman correlation is dimension-free, and falls between -1 and +1.
- A positive Spearman correlation corresponds to an increasing (but not necessarily linear) association between X and Y, while a negative Spearman correlation corresponds to a decreasing (but again not necessarily linear) association.

12.7.1 Spearman Formula

To calculate the Spearman rank correlation, we take the ranks of the X and Y data, and then apply the usual Pearson correlation. To find the ranks, sort X and Y into ascending order, and then number them from 1 (smallest) to n (largest). In the event of a tie, assign the average rank to the tied subjects.

12.7.2 Comparing Pearson and Spearman Correlations

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

```
nnyfs |> select(fat, protein) |> cor()

      fat    protein
fat    1.0000000 0.6671209
protein 0.6671209 1.0000000

nnyfs |> select(fat, protein) %>% cor(., method = "spearman")

      fat    protein
fat    1.0000000 0.6577489
protein 0.6577489 1.0000000
```

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

12.7.3 Spearman vs. Pearson Example 1

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

```
spear1 <- read_csv("data/spear1.csv")

Rows: 22 Columns: 2
-- Column specification -----
Delimiter: ","
dbl (2): x, y

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
spear2 <- read_csv("data/spear2.csv")
```

Rows: 90 Columns: 2

-- Column specification -----

Delimiter: ","

dbl (2): x, y

i Use `spec()` to retrieve the full column specification for this data.

i Specify the column types or set `show_col_types = FALSE` to quiet this message.

```
spear3 <- read_csv("data/spear3.csv")
```

Rows: 55 Columns: 2

-- Column specification -----

Delimiter: ","

dbl (2): x, y

i Use `spec()` to retrieve the full column specification for this data.

i Specify the column types or set `show_col_types = FALSE` to quiet this message.

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

Rows: 15 Columns: 2

-- Column specification -----

Delimiter: ","

dbl (2): x, y

i Use `spec()` to retrieve the full column specification for this data.

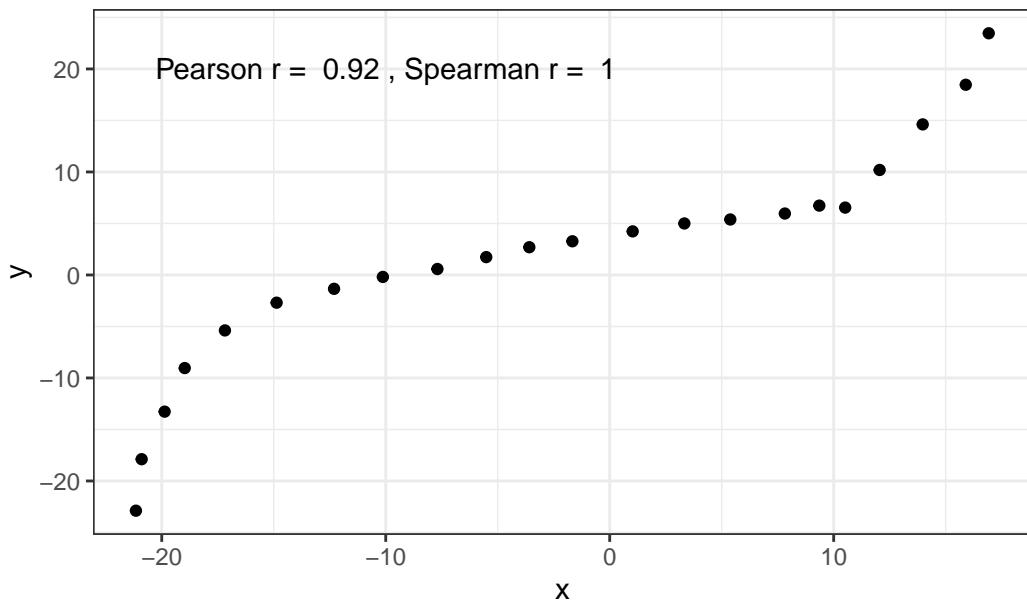
i Specify the column types or set `show_col_types = FALSE` to quiet this message.

```
# these are just toy examples with  
# two columns per data set and no row numbering
```

Example 1 shows a function where the Pearson correlation is 0.925 (a strong but not perfect linear relation), but the Spearman correlation is 1 because the relationship is monotone, even though it is not perfectly linear.

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

Spearman vs. Pearson, Example 1



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

12.7.4 Spearman vs. Pearson Example 2

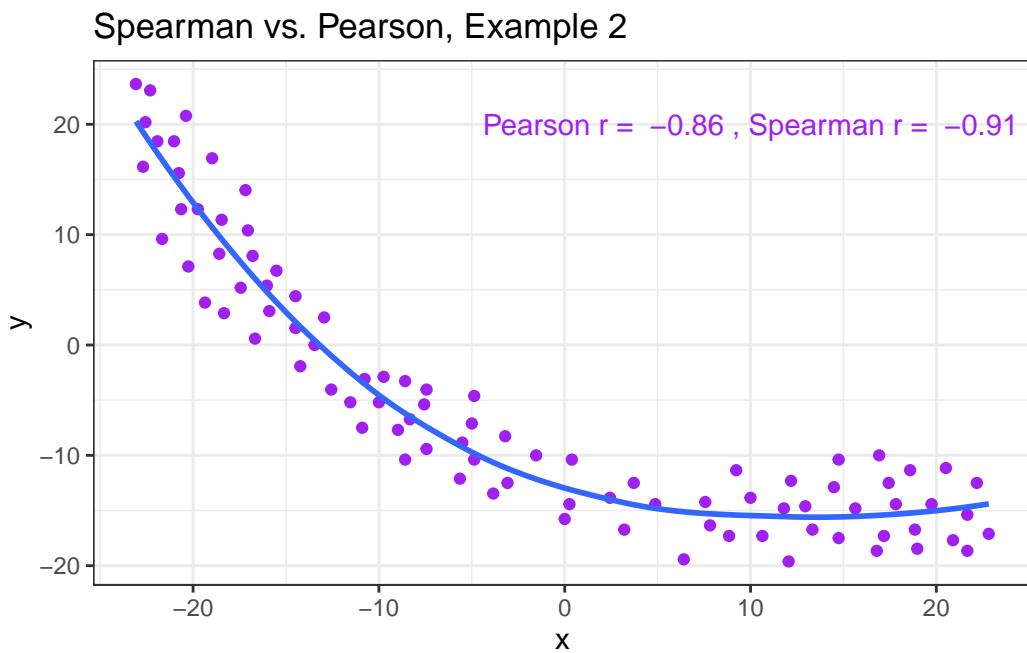
Example 2 shows that a negative Spearman correlation corresponds to a decreasing (but, again, not necessarily linear) association between x and y.

```
ggplot(spear2, aes(x = x, y = y)) +
  geom_point(col = "purple") +
  geom_smooth(method = "loess", formula = y ~ x, se = FALSE) +
```

```

labs(title = "Spearman vs. Pearson, Example 2") +
annotate("text", x = 10, y = 20, col = "purple",
label = paste("Pearson r = ",
signif(cor(spear2$x, spear2$y),2),
", Spearman r = ",
signif(cor(spear2$x, spear2$y, method = "spearman"),2)))

```



12.7.5 Spearman vs. Pearson Example 3

The Spearman correlation is less sensitive than the Pearson correlation is to strong outliers that are unusual on either the X or Y axis, or both. That is because the Spearman rank coefficient limits the outlier to the value of its rank.

In Example 3, for instance, the Spearman correlation reacts much less to the outliers around $X = 12$ than does the Pearson correlation.

```

ggplot(spear3, aes(x = x, y = y)) +
  geom_point(col = "blue") +
  labs(title = "Spearman vs. Pearson, Example 3") +
  annotate("text", x = 5, y = -15, col = "blue",
label = paste("Pearson r = ",

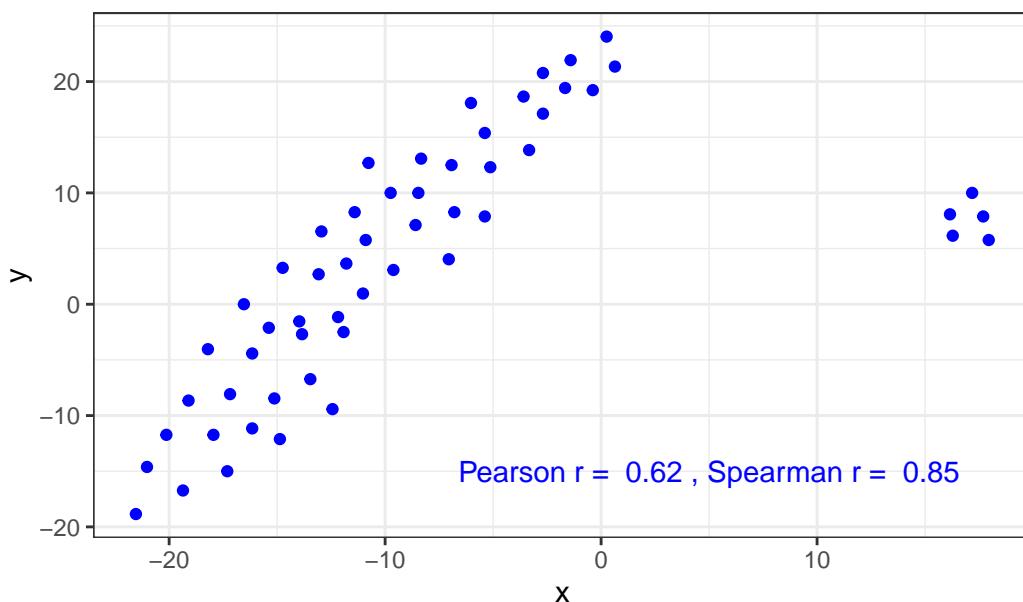
```

```

signif(cor(spear3$x, spear3$y), 2),
", Spearman r = ",
signif(cor(spear3$x, spear3$y, method = "spearman"), 2)))

```

Spearman vs. Pearson, Example 3



12.7.6 Spearman vs. Pearson Example 4

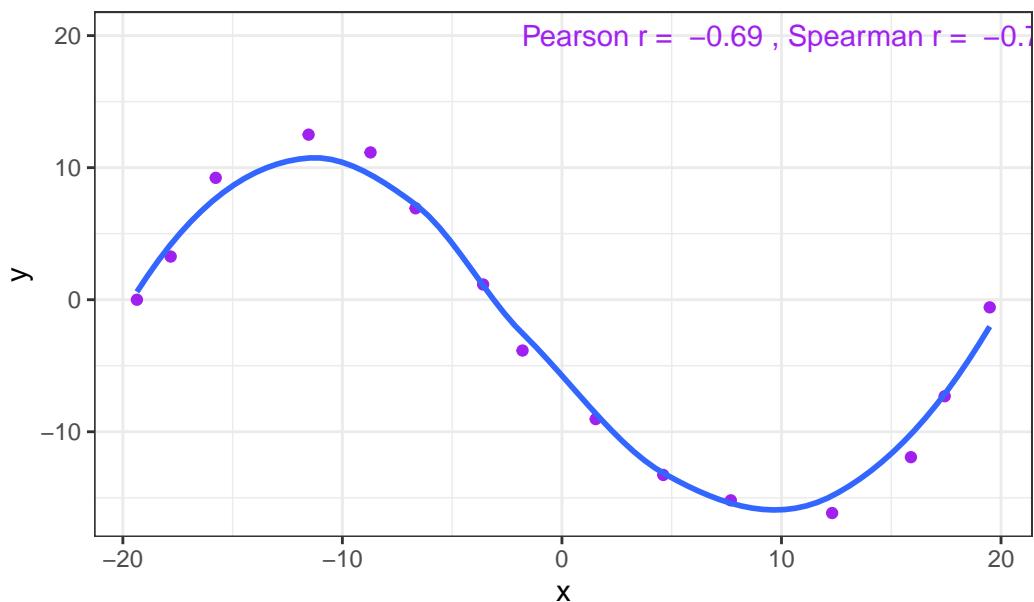
The use of a Spearman correlation is no substitute for looking at the data. For non-monotone data like what we see in Example 4, neither the Spearman nor the Pearson correlation alone provides much guidance, and just because they are (essentially) telling you the same thing, that doesn't mean what they're telling you is all that helpful.

```

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

```

Spearman vs. Pearson, Example 4



13 Linearizing Transformations

13.1 Setup: Packages Used Here

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

library(broom)
library(car)
library(patchwork)
library(tidyverse)

theme_set(theme_bw())
```

13.2 “Linearize” The Association between Quantitative Variables

Confronted with a scatterplot describing a monotone association between two quantitative variables, we may decide the data are not well approximated by a straight line, and thus, that a least squares regression may not be sufficiently useful. In these circumstances, we have at least two options, which are not mutually exclusive:

- a. Let the data be as they may, and summarize the scatterplot using tools like loess curves, polynomial functions, or cubic splines to model the relationship.
- b. Consider re-expressing the data (often we start with re-expressions of the outcome data [the Y variable]) using a transformation so that the transformed data may be modeled effectively using a straight line.

13.3 The Box-Cox Plot

As before, Tukey’s ladder of power transformations can guide our exploration.

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

The **Box-Cox plot**, from the `boxCox` function in the `car` package, sifts through the ladder of options to suggest a transformation (for Y) to best linearize the outcome-predictor(s) relationship.

13.3.1 A Few Caveats

1. These methods work well with *monotone* data, where a smooth function of Y is either strictly increasing, or strictly decreasing, as X increases.
2. Some of these transformations require the data to be positive. We can rescale the Y data by adding a constant to every observation in a data set without changing shape.
3. We can use a natural logarithm (`log` in R), a base 10 logarithm (`log10`) or even sometimes a base 2 logarithm (`log2`) to good effect in Tukey's ladder. All affect the association's shape in the same way, so we'll stick with `log` (base e).
4. Some re-expressions don't lead to easily interpretable results. Not many things that make sense in their original units also make sense in inverse square roots. There are times when we won't care, but often, we will.
5. If our primary interest is in making predictions, we'll generally be more interested in getting good predictions back on the original scale, and we can back-transform the point and interval estimates to accomplish this.

13.4 A Simulated Example

```

set.seed(999);
x.rand <- rbeta(80, 2, 5) * 20 + 3
set.seed(1000);
y.rand <- abs(50 + 0.75*x.rand^(3)
              - 0.65*x.rand + rnorm(80, 0, 200))

scatter1 <- tibble(x = x.rand, y = y.rand)
rm(x.rand, y.rand)

ggplot(scatter1, aes(x = x, y = y)) +
  geom_point(shape = 1, size = 3) +
  ## add loess smooth

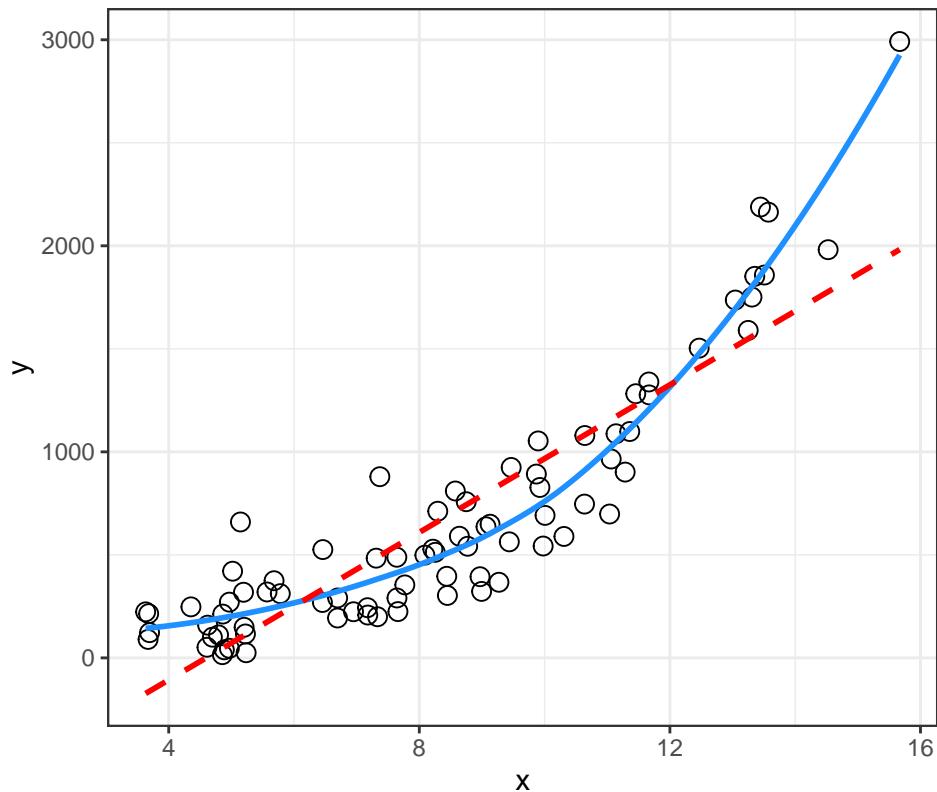
```

```

geom_smooth(method = "loess", se = FALSE,
            col = "dodgerblue", formula = y ~ x) +
## then add linear fit
geom_smooth(method = "lm", se = FALSE,
            col = "red", formula = y ~ x, linetype = "dashed") +
labs(title = "Simulated scatter1 example: Y vs. X")

```

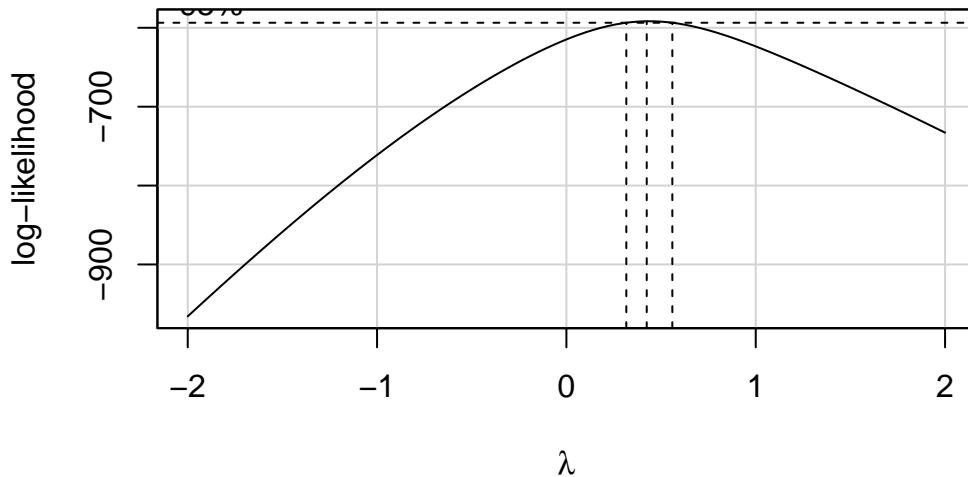
Simulated scatter1 example: Y vs. X



Having simulated data that produces a curved scatterplot, I will now use the Box-Cox plot to lead my choice of an appropriate power transformation for Y in order to “linearize” the association of Y and X.

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

Profile Log-likelihood



```
powerTransform(scatter1$y ~ scatter1$x)
```

```
Estimated transformation parameter
Y1
0.4368753
```

The Box-Cox plot peaks at the value $\lambda = 0.44$, which is pretty close to $\lambda = 0.5$. Now, 0.44 isn't on Tukey's ladder, but 0.5 is.

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

If we use $\lambda = 0.5$, on Tukey's ladder of power transformations, it suggests we look at the relationship between the square root of Y and X, as shown next.

```
p1 <- ggplot(scatter1, aes(x = x, y = y)) +
  geom_point(size = 2) +
  geom_smooth(method = "loess", se = FALSE,
              formula = y ~ x, col = "dodgerblue") +
```

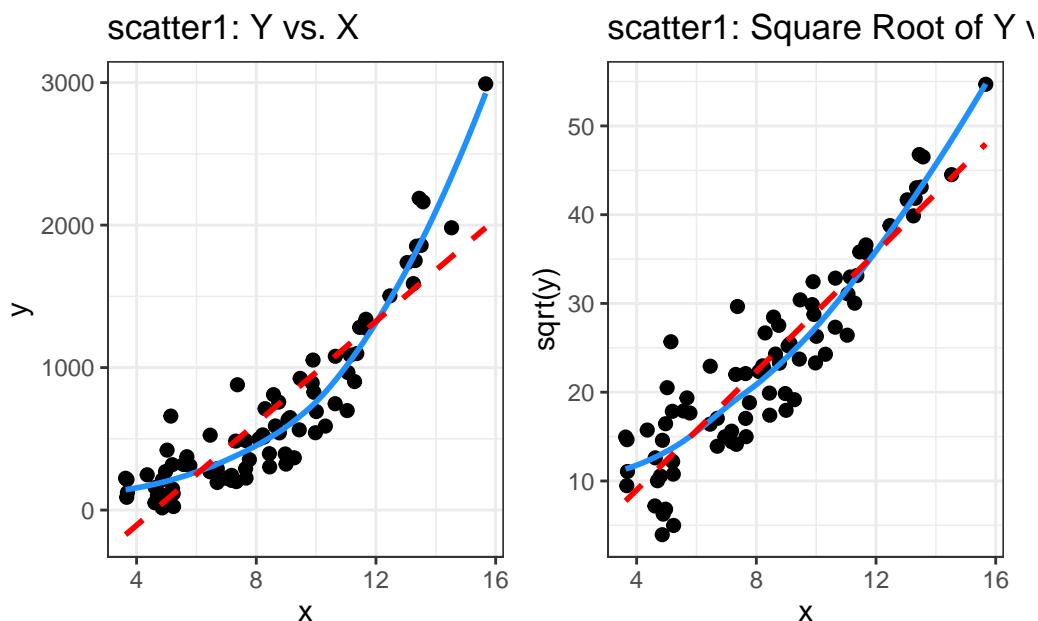
```

geom_smooth(method = "lm", se = FALSE,
            formula = y ~ x, col = "red", linetype = "dashed") +
  labs(title = "scatter1: Y vs. X")

p2 <- ggplot(scatter1, aes(x = x, y = sqrt(y))) +
  geom_point(size = 2) +
  geom_smooth(method = "loess", se = FALSE,
              formula = y ~ x, col = "dodgerblue") +
  geom_smooth(method = "lm", se = FALSE,
              formula = y ~ x, col = "red", linetype = "dashed") +
  labs(title = "scatter1: Square Root of Y vs. X")

p1 + p2

```



By eye, I think the square root plot better matches the linear fit.

13.5 Checking on a Transformation or Re-Expression

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

1. We can calculate the correlation of our original and re-expressed associations.

2. We can use the `testTransform` function in the `car` library in R to perform a statistical test comparing the optimal choice of power ($\lambda = 0.44$) to various other transformations.
3. We can go ahead and fit the regression models using each approach and compare the plots of studentized residuals against fitted values from the data to see if the re-expression reduces the curve in that residual plot, as well.

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

13.5.1 Checking the Correlation Coefficients

Here, we calculate the correlation of original and re-expressed associations.

```
cor(scatter1$y, scatter1$x)

[1] 0.891198

cor(sqrt(scatter1$y), scatter1$x)

[1] 0.9144307
```

The Pearson correlation is a little stronger after the transformation. as we'd expect.

13.5.2 Using the `testTransform` function

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

```
testTransform(powerTransform(scatter1$y ~ scatter1$x), 0)

      LRT df      pval
LR test, lambda = (0) 46.17947 1 1.079e-11

testTransform(powerTransform(scatter1$y ~ scatter1$x), 0.5)

      LRT df      pval
LR test, lambda = (0.5) 1.024888 1 0.31136
```

```
testTransform(powerTransform(scatter1$y ~ scatter1$x), 1)
```

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

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

13.5.3 Comparing the Residual Plots

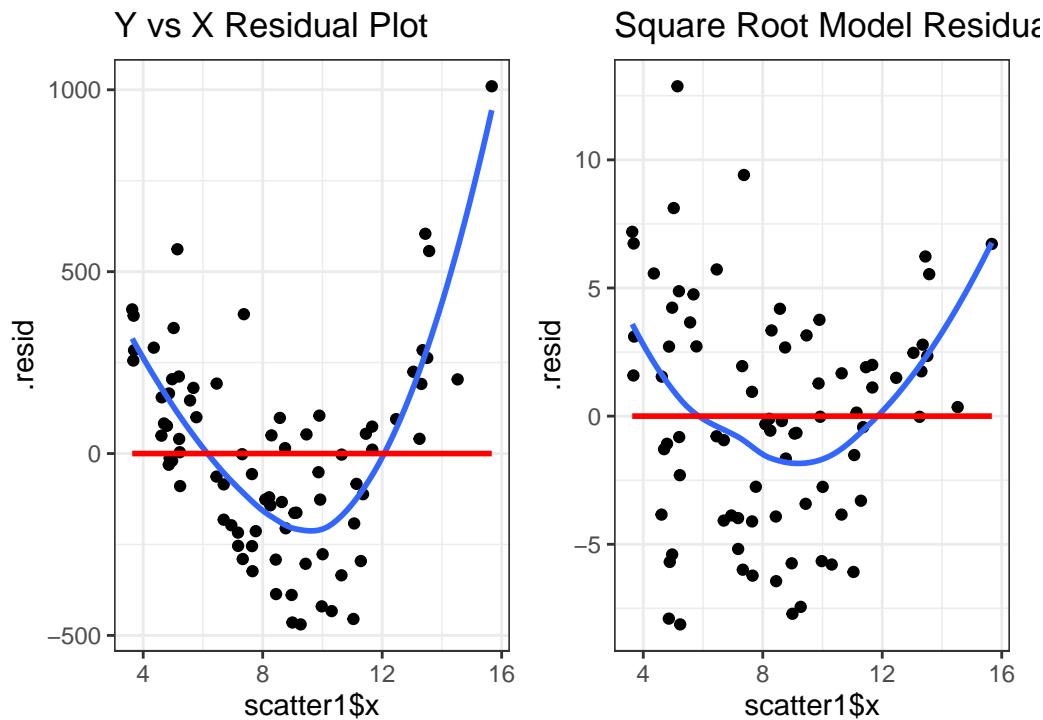
We can fit the regression models, obtain plots of residuals against fitted values, and compare them to see which one has less indication of a curve in the residuals.

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

p1 <- augment(model.orig) %>%
  ggplot(., aes(x = scatter1$x, y = .resid)) +
  geom_point() +
  geom_smooth(method = "loess", formula = y ~ x, se = FALSE) +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE, col = "red") +
  labs(title = "Y vs X Residual Plot")

p2 <- augment(model.sqrt) %>%
  ggplot(., aes(x = scatter1$x, y = .resid)) +
  geom_point() +
  geom_smooth(method = "loess", formula = y ~ x, se = FALSE) +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE, col = "red") +
  labs(title = "Square Root Model Residuals")

p1 + p2
```



What we're looking for in such a plot is the absence of a curve, among other things, we want to see “fuzzy football” shapes.

As compared to the original residual plot, the square root version, is a modest improvement in this regard. It does look a bit less curved, and a bit more like a random cluster of points, so that's nice. Usually, we can do a little better in real data, as shown in the next example from the NNYFS data we introduced in Chapter @ref(NYFS-Study).

13.6 An Example from the NNYFS data

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

Using the subjects in the `nnyfs` data with complete data on the two variables of interest, let's look at the relationship between arm circumference (the outcome, shown on the Y axis) and arm length (the predictor, shown on the X axis.)

```
nnyfs_c <- nnyfs |>  
  filter(complete.cases(arm_circ, arm_length)) |>  
  select(SEQN, arm_circ, arm_length)
```

13.6.1 Pearson correlation and scatterplot

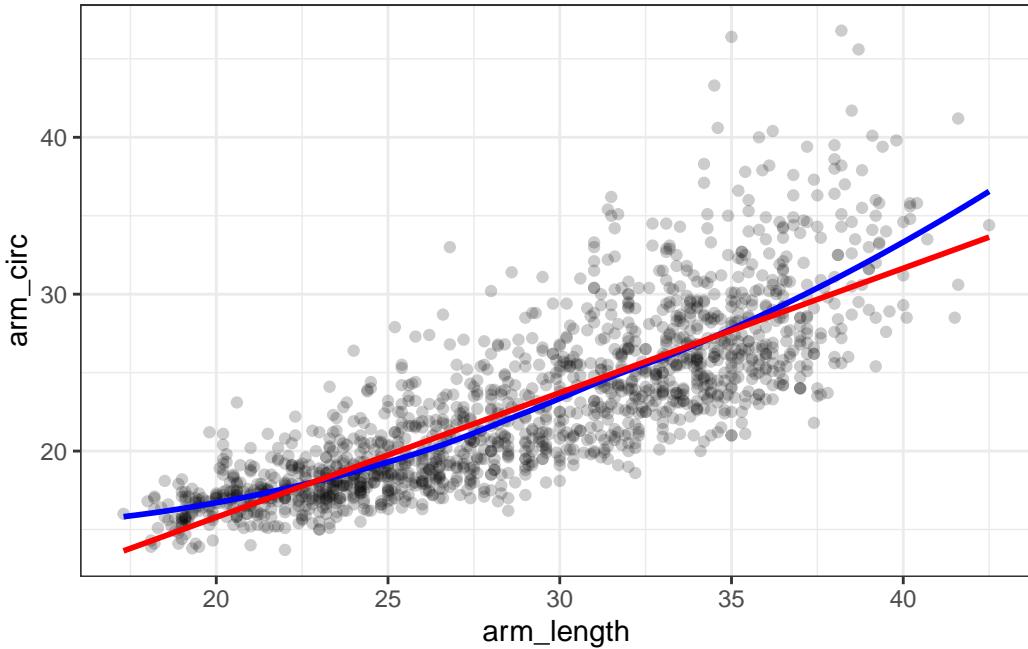
Here is the Pearson correlation between these two variables.

```
nnyfs_c |> select(arm_length, arm_circ) |> cor()
```

| | arm_length | arm_circ |
|------------|------------|-----------|
| arm_length | 1.0000000 | 0.8120242 |
| arm_circ | 0.8120242 | 1.0000000 |

Here's the resulting scatterplot.

```
ggplot(nnyfs_c, aes(x = arm_length, y = arm_circ)) +  
  geom_point(alpha = 0.2) +  
  geom_smooth(method = "loess", formula = y ~ x,  
              se = FALSE, color = "blue") +  
  geom_smooth(method = "lm", formula = y ~ x,  
              se = FALSE, color = "red")
```



While the Pearson correlation is still quite strong, note that the loess smooth (shown in blue) bends up from the straight line model (shown in red) at both the low and high end of arm length.

Note also the use of `alpha = 0.2` to show the points with greater transparency than they would be shown normally (the default setting is no transparency with `alpha = 1.`)

13.6.2 Plotting the Residuals

Now, let's build a plot of residuals from the straight line model plotted against the arm length. We can obtain these residuals using the `augment()` function from the `broom` package.

```
m1 <- lm(arm_circ ~ arm_length, data = nnyfs_c)

nnyfs_c_aug1 <- augment(m1, data = nnyfs_c)

nnyfs_c_aug1

# A tibble: 1,511 x 9
  SEQN arm_circ arm_length .fitted .resid      .hat .sigma   .cooksdi .std.resid
  <dbl>    <dbl>     <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
```

```

1 71918    25.4      27.7    21.9  3.51  0.000695  3.21 0.000416   1.09
2 71919    26        38.4    30.4 -4.38  0.00253   3.21 0.00237  -1.37
3 71920    37.9      35.9    28.4  9.50  0.00167   3.20 0.00735   2.96
4 71921    15.1      18.3    14.4  0.669 0.00304   3.21 0.0000663  0.209
5 71922    29.5      34.2    27.0  2.45  0.00124   3.21 0.000362  0.764
6 71923    27.9      33       26.1  1.80  0.00100   3.21 0.000159  0.562
7 71924    17.6      26.5    20.9 -3.34  0.000788  3.21 0.000427  -1.04
8 71925    17.7      24.2    19.1 -1.41  0.00113   3.21 0.000110  -0.441
9 71926    19.9      26       20.5 -0.642 0.000844  3.21 0.0000169  -0.200
10 71927   17.3      20       15.8  1.52  0.00234   3.21 0.000263  0.474
# ... with 1,501 more rows
# i Use `print(n = ...)` to see more rows

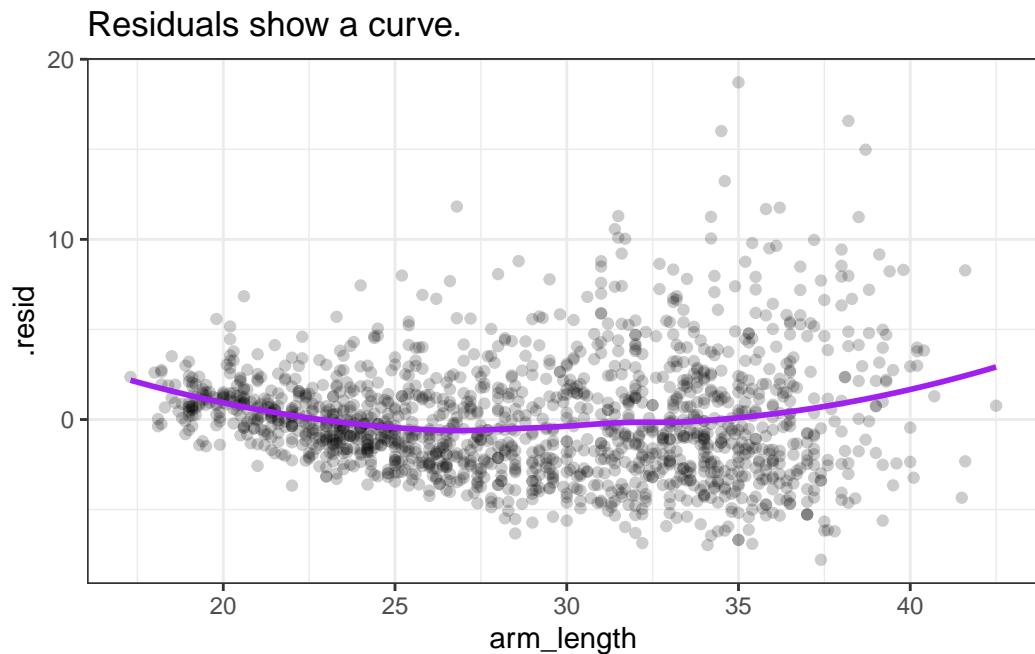
```

OK. So the residuals are now stored in the `.resid` variable. We can create a residual plot, as follows.

```

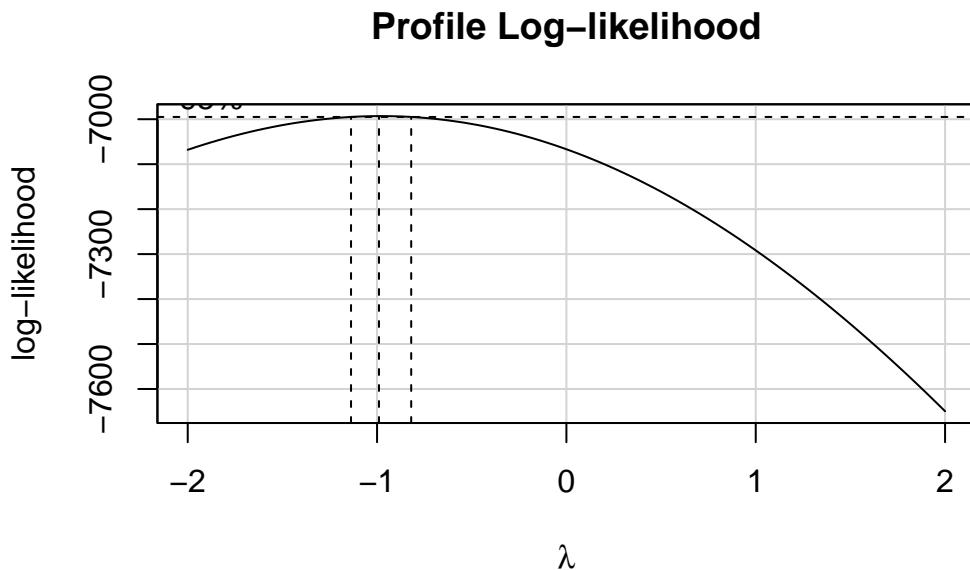
ggplot(nnyfs_c_aug1, aes(x = arm_length, y = .resid)) +
  geom_point(alpha = 0.2) +
  geom_smooth(method = "loess", col = "purple",
              formula = y ~ x, se = FALSE) +
  labs(title = "Residuals show a curve.")

```



13.6.3 Using the Box-Cox approach to identify a transformation

```
library(car)
boxCox(nnyfs_c$arm_circ ~ nnyfs_c$arm_length)
```



```
powerTransform(nnyfs_c$arm_circ ~ nnyfs_c$arm_length)
```

```
Estimated transformation parameter
Y1
-0.9783135
```

This suggests that we should transform the `arm_circ` data by taking its inverse (power = -1.) Let's take a look at that result.

13.6.4 Plots after Inverse Transformation

Let's build (on the left) the revised scatterplot and (on the right) the revised residual plot after transforming the outcome (`arm_circ`) by taking its inverse.

```

nnyfs_c <- nnyfs_c |>
  mutate(inv_arm_circ = 1/arm_circ)

p1 <- ggplot(nnyfs_c, aes(x = arm_length, y = inv_arm_circ)) +
  geom_point(alpha = 0.2) +
  geom_smooth(method = "loess", formula = y ~ x,
              se = FALSE, color = "blue") +
  geom_smooth(method = "lm", formula = y ~ x,
              se = FALSE, color = "red") +
  labs(title = "Transformation reduces curve")

m2 <- lm(inv_arm_circ ~ arm_length, data = nnyfs_c)

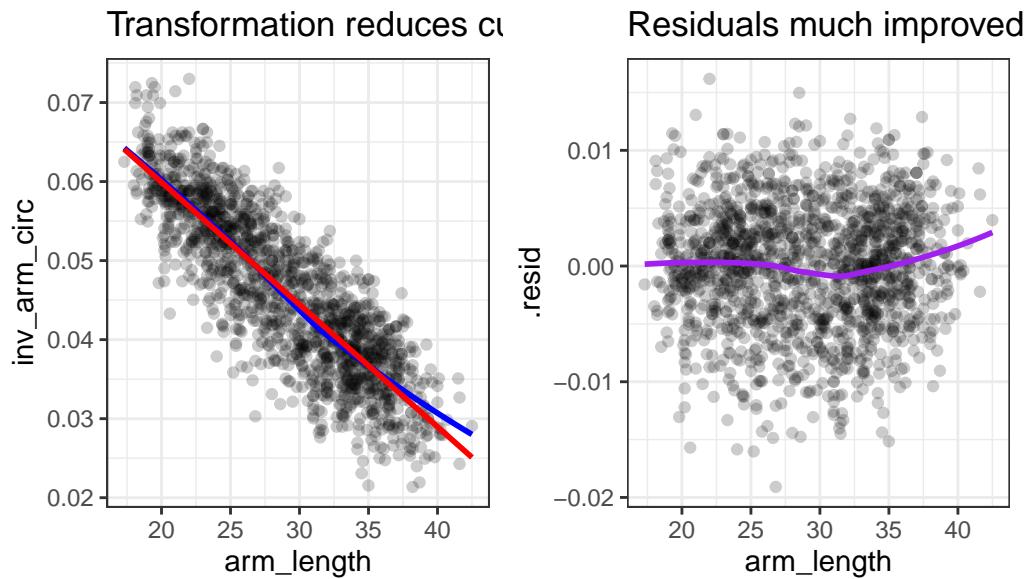
nnyfs_c_aug2 <- augment(m2, data = nnyfs_c)

p2 <- ggplot(nnyfs_c_aug2, aes(x = arm_length, y = .resid)) +
  geom_point(alpha = 0.2) +
  geom_smooth(method = "loess", col = "purple",
              formula = y ~ x, se = FALSE) +
  labs(title = "Residuals much improved")

p1 + p2 +
  plot_annotation(title = "Evaluating the Inverse Transformation")

```

Evaluating the Inverse Transformation



14 Studying Crab Claws

For our next example, we'll consider a study from zoology, specifically carcinology - the study of crustaceans. My source for these data is Chapter 7 in Ramsey and Schafer (2002) which drew the data from a figure in Yamada and Boulding (1998).

14.1 Setup: Packages Used Here

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

library(janitor)
library(broom)
library(knitr)
library(tidyverse)

theme_set(theme_bw())
```

We will also use the `describe` function from the `psych` package.

14.2 The Data

The available data are the mean closing forces (in Newtons) and the propodus heights (mm) of the claws on 38 crabs that came from three different species. The *propodus* is the segment of the crab's clawed leg with an immovable finger and palm.

This was part of a study of the effects that predatory intertidal crab species have on populations of snails. The three crab species under study are:

- 14 *Hemigrapsus nudus*, also called the [purple shore crab](#) (14 crabs)
- 12 *Lophopanopeus bellus*, also called the [black-clawed pebble crab](#), and
- 12 *Cancer productus*, one of several species of [red rock crabs](#) (12)

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

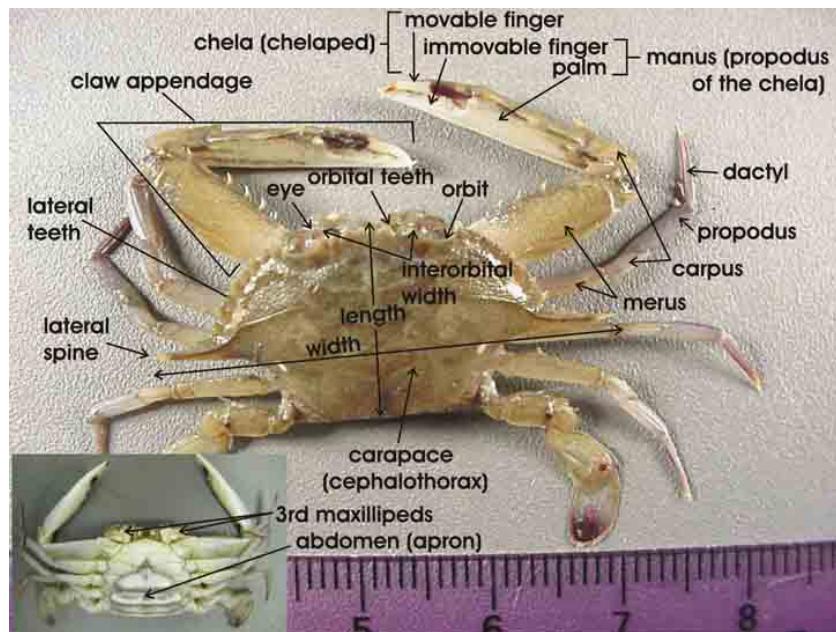


Figure 14.1: Source: <http://txmarspecies.tamug.edu/crustglossary.cfm>

Rows: 38 Columns: 4

-- Column specification -----

Delimiter: ","

chr (1): species

dbl (3): crab, force, height

i Use `spec()` to retrieve the full column specification for this data.

i Specify the column types or set `show_col_types = FALSE` to quiet this message.

crabs

| # | A tibble: 38 x 4 | crab | species | force | height |
|---|------------------------|-------|---------|-------|--------|
| | | <dbl> | <chr> | <dbl> | <dbl> |
| 1 | 1 Hemigrapsus nudus | 4 | 8 | | |
| 2 | 2 Lophopanopeus bellus | 15.1 | 7.9 | | |
| 3 | 3 Cancer productus | 5 | 6.7 | | |
| 4 | 4 Lophopanopeus bellus | 2.9 | 6.6 | | |
| 5 | 5 Hemigrapsus nudus | 3.2 | 5 | | |
| 6 | 6 Hemigrapsus nudus | 9.5 | 7.9 | | |
| 7 | 7 Cancer productus | 22.5 | 9.4 | | |

```

8      8 Hemigrapsus nudus      7.4     8.3
9      9 Cancer productus     14.6    11.2
10     10 Lophopanopeus bellus   8.7     8.6
# ... with 28 more rows
# i Use `print(n = ...)` to see more rows

```

The `species` information is stored here as a character variable. How many different crabs are we talking about in each `species`?

```
crabs |> tabyl(species)
```

| | species | n | percent |
|----------------------|---------|-----------|---------|
| Cancer productus | 12 | 0.3157895 | |
| Hemigrapsus nudus | 14 | 0.3684211 | |
| Lophopanopeus bellus | 12 | 0.3157895 | |

As it turns out, we're going to want to treat the `species` information as a **factor** with three levels, rather than as a character variable.

```
crabs <- crabs |>
  mutate(species = factor(species))
```

Here's a quick summary of the data. Take care to note the useless results for the first two variables. At least the function flags with a * those variables it thinks are non-numeric.

```
psych::describe(crabs)
```

| | vars | n | mean | sd | median | trimmed | mad | min | max | range | skew | kurtosis |
|----------|------|----|-------|-------|--------|---------|-------|-----|------|-------|------|----------|
| crab | 1 | 38 | 19.50 | 11.11 | 19.50 | 19.50 | 14.08 | 1 | 38.0 | 37.0 | 0.00 | -1.30 |
| species* | 2 | 38 | 2.00 | 0.81 | 2.00 | 2.00 | 1.48 | 1 | 3.0 | 2.0 | 0.00 | -1.50 |
| force | 3 | 38 | 12.13 | 8.98 | 8.70 | 11.53 | 9.04 | 2 | 29.4 | 27.4 | 0.47 | -1.25 |
| height | 4 | 38 | 8.81 | 2.23 | 8.25 | 8.78 | 2.52 | 5 | 13.1 | 8.1 | 0.19 | -1.14 |
| | | | se | | | | | | | | | |
| crab | | | 1.80 | | | | | | | | | |
| species* | | | 0.13 | | | | | | | | | |
| force | | | 1.46 | | | | | | | | | |
| height | | | 0.36 | | | | | | | | | |

Actually, we're more interested in these results after grouping by species.

```

crabs |>
  group_by(species) |>
  summarise(n = n(), median(force), median(height))

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

```

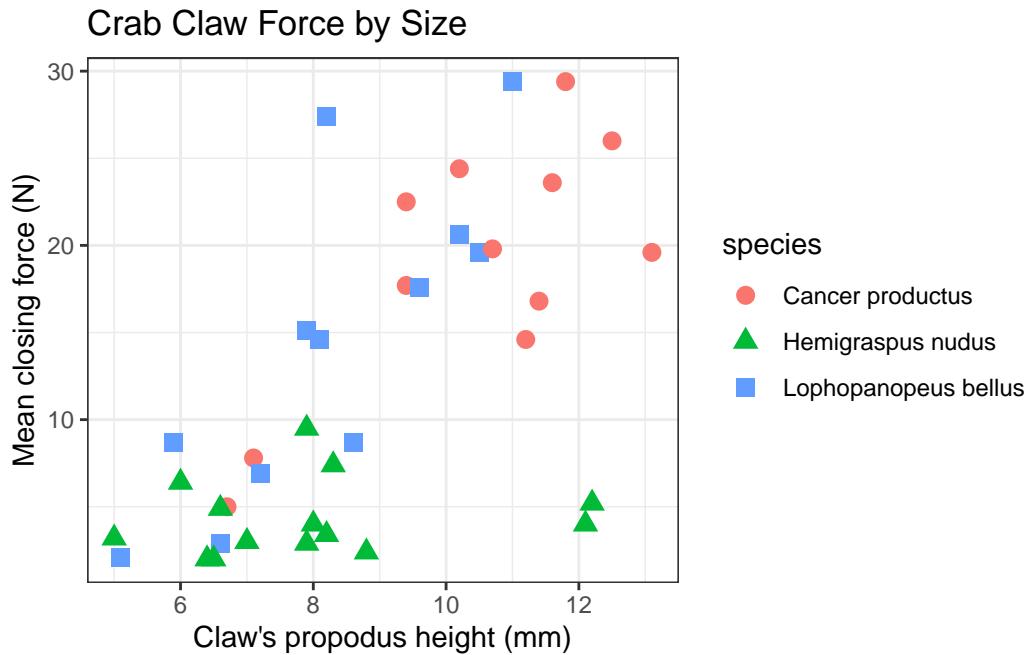
14.3 Association of Size and Force

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

```

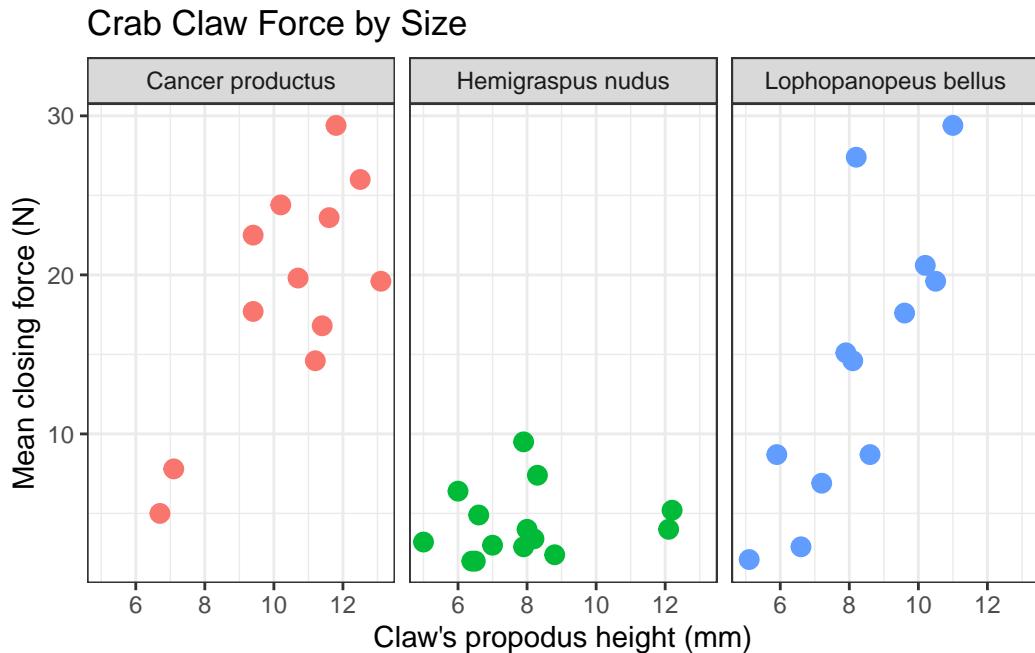
ggplot(crabs, aes(x = height, y = force, color = species, shape = species)) +
  geom_point(size = 3) +
  labs(title = "Crab Claw Force by Size",
       x = "Claw's propodus height (mm)", y = "Mean closing force (N)") +
  theme_bw()

```



A faceted plot for each species really highlights the difference in force between the *Hemigrapsus nudus* and the other two species of crab.

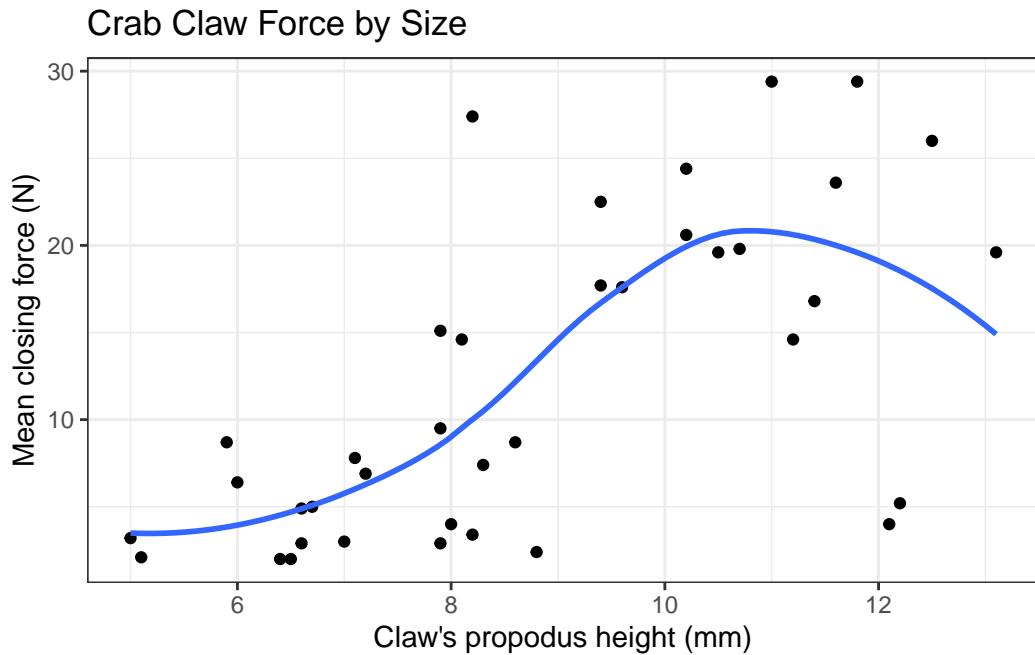
```
ggplot(crabs, aes(x = height, y = force, color = species)) +
  geom_point(size = 3) +
  facet_wrap(~ species) +
  guides(color = "none") +
  labs(title = "Crab Claw Force by Size",
       x = "Claw's propodus height (mm)", y = "Mean closing force (N)") +
  theme_bw()
```



14.4 The loess smooth

We can obtain a smoothed curve (using several different approaches) to summarize the pattern presented by the data in any scatterplot. For instance, we might build such a plot for the complete set of 38 crabs, adding in a non-linear smooth function (called a loess smooth.)

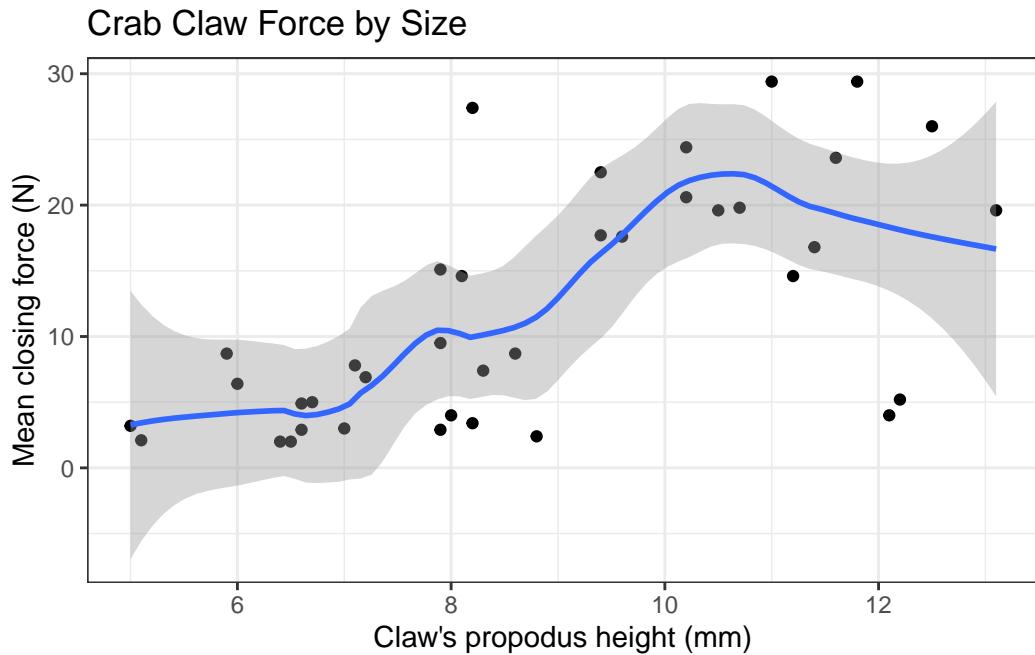
```
ggplot(crabs, aes(x = height, y = force)) +
  geom_point() +
  geom_smooth(method = "loess", se = FALSE, formula = y ~ x) +
  labs(title = "Crab Claw Force by Size",
       x = "Claw's propodus height (mm)", y = "Mean closing force (N)")
```



As we have discussed previously, a **loess smooth** fits a curve to data by tracking (at point x) the points within a neighborhood of point x , with more emphasis given to points near x . It can be adjusted by tweaking the `span` and `degree` parameters.

In addition to the curve, smoothing procedures can also provide confidence intervals around their main fitted line. Consider the following plot of the `crabs` information, which adjusts the `span` (from its default of 0.75) and also adds in the confidence intervals.

```
ggplot(crabs, aes(x = height, y = force)) +
  geom_point() +
  geom_smooth(method = "loess", formula = y ~ x, span = 0.5, se = TRUE) +
  labs(title = "Crab Claw Force by Size",
       x = "Claw's propodus height (mm)", y = "Mean closing force (N)")
```

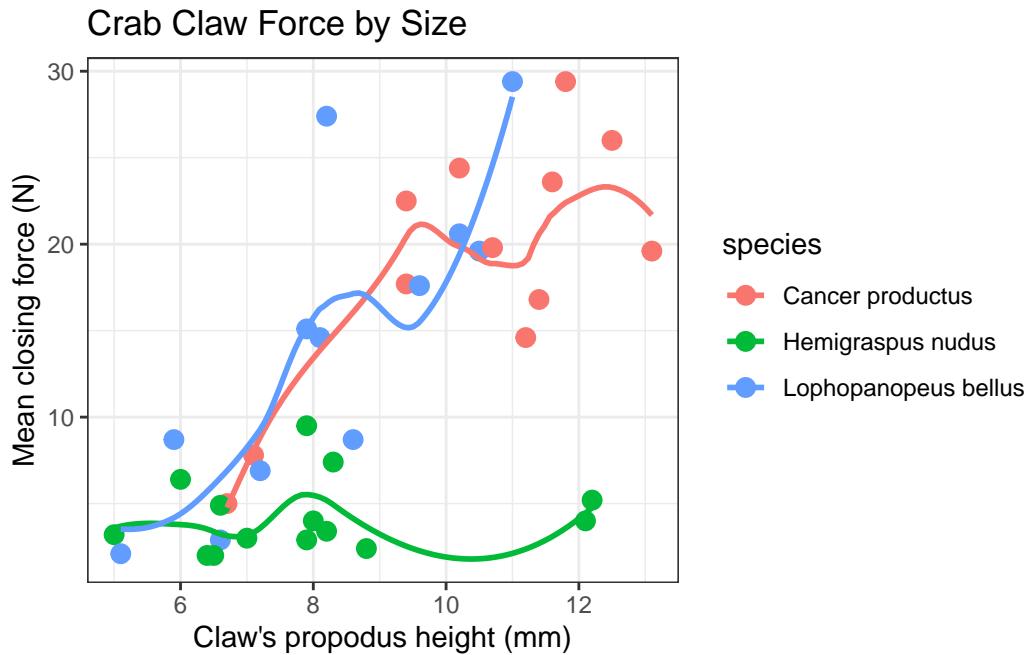


By reducing the size of the span, our resulting picture shows a much less smooth function than we generated previously.

14.4.1 Smoothing within Species

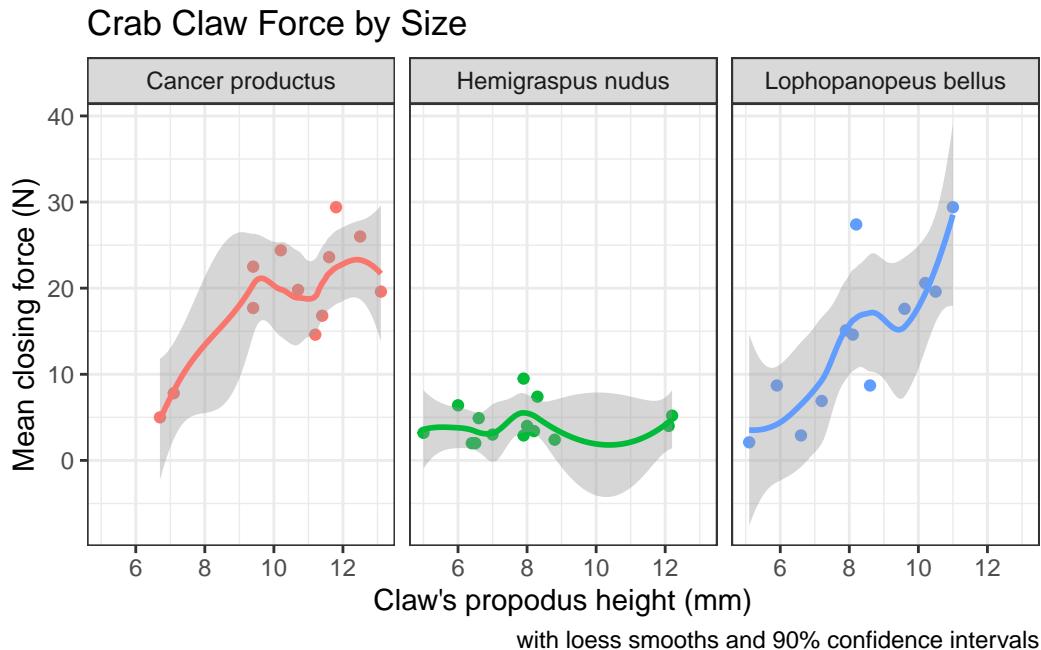
We can, of course, produce the plot above with separate smooths for each of the three species of crab.

```
ggplot(crabs, aes(x = height, y = force, group = species, color = species)) +
  geom_point(size = 3) +
  geom_smooth(method = "loess", formula = y ~ x, se = FALSE) +
  labs(title = "Crab Claw Force by Size",
       x = "Claw's propodus height (mm)", y = "Mean closing force (N)")
```



If we want to add in the confidence intervals (here I'll show them at 90% rather than the default of 95%) then this plot should be faceted. Note that by default, what is displayed when `se = TRUE` are 95% prediction intervals - the `level` function in `stat_smooth` [which can be used in place of `geom_smooth`] is used here to change the coverage percentage from 95% to 90%.

```
ggplot(crabs, aes(x = height, y = force, group = species, color = species)) +
  geom_point() +
  stat_smooth(method = "loess", formula = y ~ x, level = 0.90, se = TRUE) +
  guides(color = "none") +
  labs(title = "Crab Claw Force by Size",
       caption = "with loess smooths and 90% confidence intervals",
       x = "Claw's propodus height (mm)", y = "Mean closing force (N)") +
  facet_wrap(~ species)
```



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

14.5 Fitting a Linear Regression Model

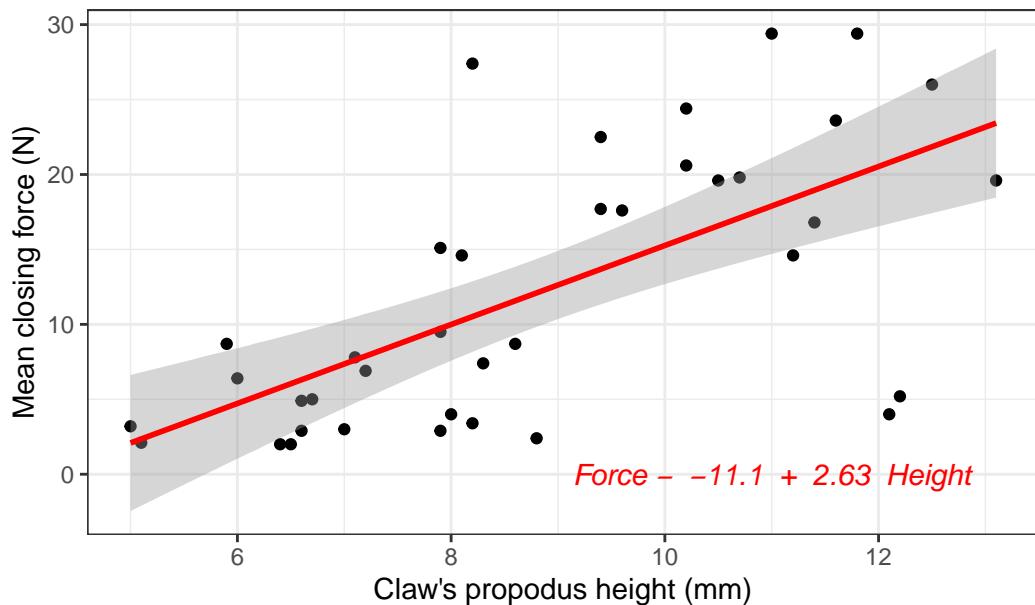
Suppose we plan to use a simple (least squares) linear regression model to describe force as a function of height. Is a least squares model likely to be an effective choice here?

The plot below shows the regression line predicting closing force as a function of propodus height. Here we annotate the plot to show the actual fitted regression line, which required fitting it with the `lm` statement prior to developing the graph.

```
mod <- lm(force ~ height, data = crabs)

ggplot(crabs, aes(x = height, y = force)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, color = "red") +
  labs(title = "Crab Claw Force by Size with Linear Regression Model",
       x = "Claw's propodus height (mm)", y = "Mean closing force (N)") +
  annotate("text", x = 11, y = 0, color = "red", fontface = "italic",
           label = paste( "Force = ", signif(coef(mod)[1],3), " + ",
                         signif(coef(mod)[2],3), " Height" ))
```

Crab Claw Force by Size with Linear Regression Model



```
rm(mod)
```

The **lm** function, again, specifies the linear model we fit to predict force using height. Here's the summary.

```
summary(lm(force ~ height, data = crabs))
```

```
Call:  
lm(formula = force ~ height, data = crabs)
```

Residuals:

| Min | 1Q | Median | 3Q | Max |
|----------|---------|---------|--------|---------|
| -16.7945 | -3.8113 | -0.2394 | 4.1444 | 16.8814 |

Coefficients:

| | Estimate | Std. Error | t value | Pr(> t) |
|-------------|----------|------------|---------|--------------|
| (Intercept) | -11.0869 | 4.6224 | -2.399 | 0.0218 * |
| height | 2.6348 | 0.5089 | 5.177 | 8.73e-06 *** |

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```
Residual standard error: 6.892 on 36 degrees of freedom
Multiple R-squared:  0.4268,    Adjusted R-squared:  0.4109
F-statistic: 26.8 on 1 and 36 DF,  p-value: 8.73e-06
```

Again, the key things to realize are:

- The outcome variable in this model is **force**, and the predictor variable is **height**.
- The straight line model for these data fitted by least squares is $\text{force} = -11.1 + 2.63 \text{ height}$.
- The slope of height is positive, which indicates that as height increases, we expect that force will also increase. Specifically, we expect that for every additional mm of height, the force will increase by 2.63 Newtons.
- The multiple R-squared (squared correlation coefficient) is 0.427, which implies that 42.7% of the variation in force is explained using this linear model with height. It also implies that the Pearson correlation between force and height is the square root of 0.427, or 0.653.

14.6 Is a Linear Model Appropriate?

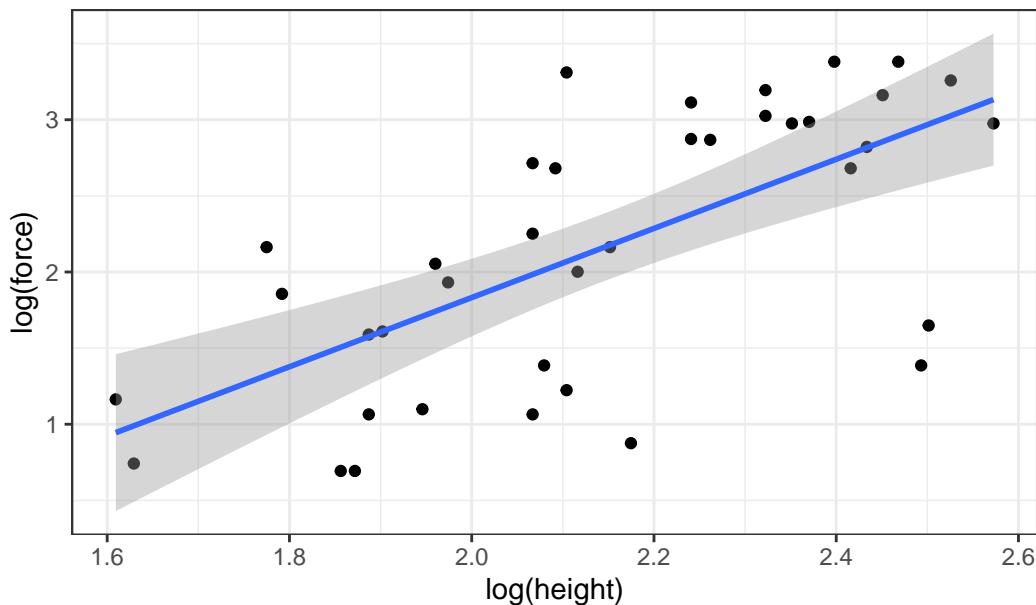
The zoology (at least as described in Ramsey and Schafer (2002)) suggests that the actual nature of the relationship would be represented by a log-log relationship, where the log of force is predicted by the log of height.

This log-log model is an appropriate model when we think that percentage increases in X (height, here) lead to constant percentage increases in Y (here, force).

To see the log-log model in action, we plot the log of force against the log of height. We could use either base 10 (`log10` in R) or natural (`log` in R) logarithms.

```
ggplot(crabs, aes(x = log(height), y = log(force))) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x) +
  labs(title = "Log-Log Model for Crabs data")
```

Log–Log Model for Crabs data



The correlations between the raw force and height and between their logarithms turn out to be quite similar, and because the log transformation is monotone in these data, there's actually no change at all in the Spearman correlations.

| Correlation of | Pearson r | Spearman r |
|--|-----------|------------|
| force and height | 0.653 | 0.657 |
| $\log(\text{force})$ and $\log(\text{height})$ | 0.662 | 0.657 |

14.6.1 The log-log model

```
crab_loglog <- lm(log(force) ~ log(height), data = crabs)
summary(crab_loglog)
```

Call:
`lm(formula = log(force) ~ log(height), data = crabs)`

Residuals:

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

```

-1.5657 -0.4450  0.1884  0.4798  1.2422

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.7104     0.9251  -2.930  0.00585 **
log(height)  2.2711     0.4284   5.302 5.96e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6748 on 36 degrees of freedom
Multiple R-squared:  0.4384,    Adjusted R-squared:  0.4228
F-statistic: 28.11 on 1 and 36 DF,  p-value: 5.96e-06

```

Our regression equation is $\log(\text{force}) = -2.71 + 2.27 \log(\text{height})$.

So, for example, if we found a crab with propodus height = 10 mm, our prediction for that crab's claw force (in Newtons) based on this log-log model would be...

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

14.6.2 How does this compare to our original linear model?

```

crab_linear <- lm(force ~ height, data = crabs)

summary(crab_linear)

```

Call:
`lm(formula = force ~ height, data = crabs)`

Residuals:

| Min | 1Q | Median | 3Q | Max |
|----------|---------|---------|--------|---------|
| -16.7945 | -3.8113 | -0.2394 | 4.1444 | 16.8814 |

Coefficients:

| | Estimate | Std. Error | t value | Pr(> t) |
|-------------|----------|------------|---------|----------|
| (Intercept) | -11.0869 | 4.6224 | -2.399 | 0.0218 * |

```

height          2.6348      0.5089    5.177 8.73e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.892 on 36 degrees of freedom
Multiple R-squared:  0.4268,   Adjusted R-squared:  0.4109
F-statistic:  26.8 on 1 and 36 DF,  p-value: 8.73e-06

```

The linear regression equation is force = $-11.1 + 2.63 \text{ height}$.

So, for example, if we found a crab with propodus height = 10 mm, our prediction for that crab's claw force (in Newtons) based on this linear model would be...

- force = $-11.0869025 + 2.6348232 \times 10$
- force = $-11.0869025 + 26.3482321$
- so predicted force = 15.2613297, which we would round to 15.3 Newtons.

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

14.7 Making Predictions with a Model

The `broom` package's `augment` function provides us with a consistent method for obtaining predictions (also called fitted values) for a new crab or for our original data. Suppose we want to predict the `force` level for two new crabs: one with height = 10 mm, and another with height = 12 mm.

```

newcrab <- tibble(crab = c("Crab_A", "Crab_B"), height = c(10, 12))

augment(crab_linear, newdata = newcrab)

# A tibble: 2 x 3
  crab    height .fitted
  <chr>    <dbl>    <dbl>
1 Crab_A     10     15.3
2 Crab_B     12     20.5

```

Should we want to obtain a prediction interval, we can use the `predict` function:

```
predict(crab_linear, newdata = newcrab, interval = "prediction", level = 0.95)
```

```
    fit      lwr      upr
1 15.26133 1.048691 29.47397
2 20.53098 5.994208 35.06774
```

We'd interpret this result as saying that the linear model's predicted force associated with a single new crab claw with propodus height 10 mm is 15.3 Newtons, and that a 95% prediction interval for the true value of such a force for such a claw is between 1.0 and 29.5 Newtons. More on prediction intervals later.

14.7.1 Predictions After a Transformation

We can also get predictions from the log-log model. The default choice is a 95% prediction interval.

```
predict(crab_loglog, newdata = newcrab, interval = "prediction")
```

```
    fit      lwr      upr
1 2.519095 1.125900 3.912291
2 2.933174 1.515548 4.350800
```

Of course, these predictions describe the `log(force)` for such a crab claw. To get the prediction in terms of simple force, we'd need to back out of the logarithm, by exponentiating our point estimate and the prediction interval endpoints.

```
exp(predict(crab_loglog, newdata = newcrab, interval = "prediction"))
```

```
    fit      lwr      upr
1 12.41736 3.082989 50.01341
2 18.78716 4.551916 77.54044
```

We'd interpret this result as saying, for the first new crab, that the log-log model's predicted force associated with a single new crab claw with propodus height 10 mm is 12.4 Newtons, and that a 95% prediction interval for the true value of such a force for such a claw is between 3.1 and 50.0 Newtons.

14.7.2 Comparing Model Predictions

Suppose we wish to build a plot of force vs height with a straight line for the linear model's predictions, and a new curve for the log-log model's predictions, so that we can compare and contrast the implications of the two models on a common scale. The `predict` function, when not given a new data frame, will use the existing predictor values that are in our `crabs` data. Such predictions are often called fitted values.

To put the two sets of predictions on the same scale despite the differing outcomes in the two models, we'll exponentiate the results of the log-log model, and build a little data frame containing the heights and the predicted forces from that model.

```
loglogdat <- tibble(height = crabs$height, force = exp(predict(crab_loglog)))
```

A cleaner way to do this might be to use the `augment` function directly from `broom`:

```
augment(crab_loglog)
```

```
# A tibble: 38 x 7
`log(force)` `log(height)` .fitted   .hat   .sigma   .cooksdi .std.resid
<dbl>        <dbl>      <dbl>  <dbl>    <dbl>     <dbl>       <dbl>
1      1.39      2.08    2.01  0.0280  0.676 1.28e- 2  -0.941
2      2.71      2.07    1.98  0.0287  0.673 1.79e- 2   1.10 
3      1.61      1.90    1.61  0.0499  0.684 8.06e-10 -0.000175
4      1.06      1.89    1.58  0.0530  0.679 1.69e- 2  -0.778
5      1.16      1.61    0.945 0.142   0.683 1.01e- 2   0.349 
6      2.25      2.07    1.98  0.0287  0.683 2.39e- 3   0.402 
7      3.11      2.24    2.38  0.0301  0.673 1.90e- 2   1.11 
8      2.00      2.12    2.10  0.0266  0.684 2.75e- 4  -0.142 
9      2.68      2.42    2.78  0.0561  0.684 6.30e- 4  -0.146 
10     2.16      2.15    2.18  0.0263  0.684 5.34e- 6  -0.0199
# ... with 28 more rows
# i Use `print(n = ...)` to see more rows
```

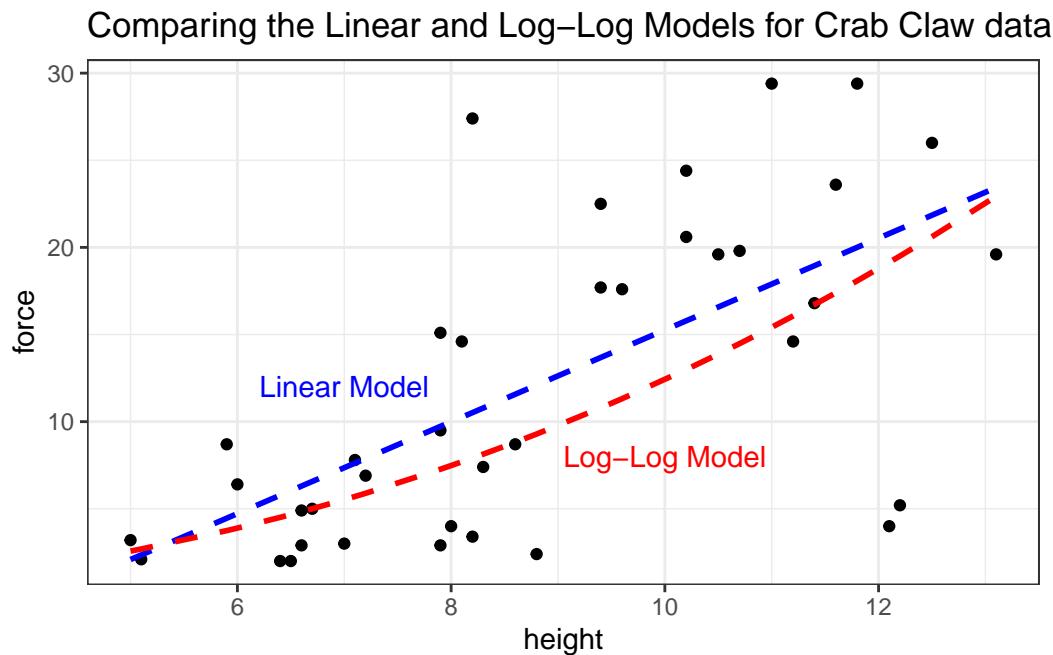
Now, we're ready to use the `geom_smooth` approach to plot the linear fit, and `geom_line` (which also fits curves) to display the log-log fit.

```
ggplot(crabs, aes(x = height, y = force)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE,
              formula = y ~ x, col="blue", linetype = 2) +
  geom_line(data = loglogdat, col = "red", linetype = 2, size = 1) +
```

```

annotate("text", 7, 12, label = "Linear Model", col = "blue") +
annotate("text", 10, 8, label = "Log-Log Model", col = "red") +
labs(title = "Comparing the Linear and Log-Log Models for Crab Claw data")

```



Based on these 38 crabs, we see some modest differences between the predictions of the two models, with the log-log model predicting generally lower closing force for a given propodus height than would be predicted by a linear model.

15 Dehydration Recovery

15.1 Setup: Packages Used Here

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

library(knitr)
library(broom)
library(patchwork)
library(tidyverse)

theme_set(theme_bw())
```

We will also use the `ggpairs` function from the `GGally` package, and the `favstats` function from the `mosaic` package.

15.2 The Data

The `hydrate` data describe the degree of recovery that takes place 90 minutes following treatment of moderate to severe dehydration, for 36 children diagnosed at a hospital's main pediatric clinic.

Upon diagnosis and study entry, patients were treated with an electrolytic solution at one of seven `dose` levels (0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0 mEq/l) in a frozen, flavored, ice popsicle. The degree of rehydration was determined using a subjective scale based on physical examination and parental input, converted to a 0 to 100 point scale, representing the percent of recovery (`recov.score`). Each child's `age` (in years) and `weight` (in pounds) are also available.

First, we'll check ranges (and for missing data) in the `hydrate` file.

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

summary(hydrate)
```

```

      id      recov.score       dose        age
Min.   : 1.00  Min.   :44.00  Min.   :0.000  Min.   : 3.000
1st Qu.: 9.75  1st Qu.:61.50  1st Qu.:1.000  1st Qu.: 5.000
Median :18.50  Median :71.50  Median :1.500  Median : 6.500
Mean   :18.50  Mean   :71.56  Mean   :1.569  Mean   : 6.667
3rd Qu.:27.25  3rd Qu.:80.00  3rd Qu.:2.500  3rd Qu.: 8.000
Max.   :36.00  Max.   :100.00  Max.   :3.000  Max.   :11.000

      weight
Min.   :22.00
1st Qu.:34.50
Median :47.50
Mean   :46.89
3rd Qu.:57.25
Max.   :76.00

```

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

15.3 A Scatterplot Matrix

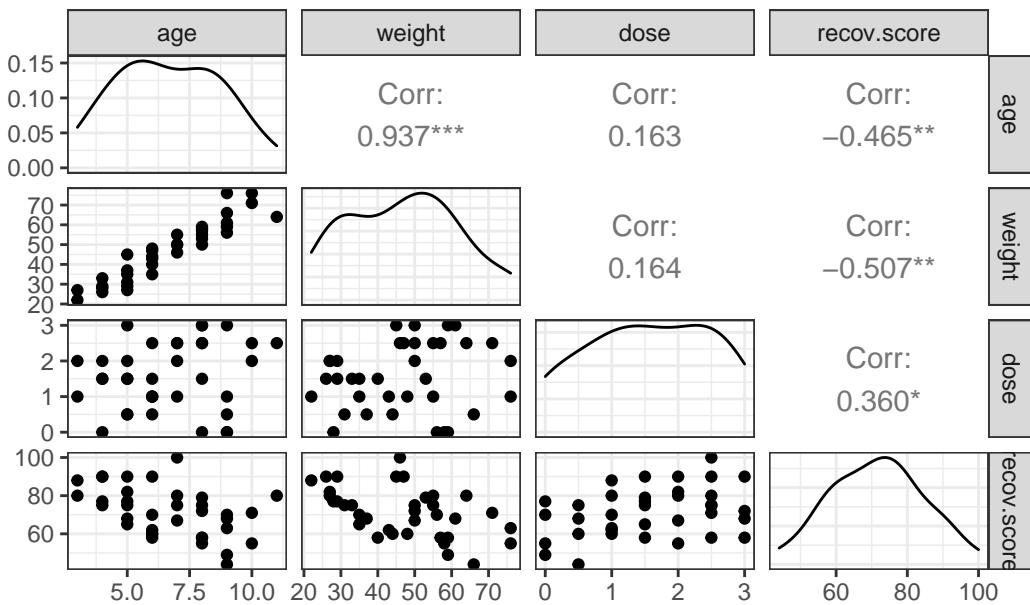
Next, we'll use a scatterplot matrix to summarize relationships between the outcome `recov.score` and the key predictor `dose` as well as the ancillary predictors `age` and `weight`, which are of less interest, but are expected to be related to our outcome. The one below uses the `ggpairs` function in the `GGally` package, as introduced in Part A of the Notes. We place the outcome in the bottom row, and the key predictor immediately above it, with `age` and `weight` in the top rows, using the `select` function within the 'ggpairs' call.

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

```
Registered S3 method overwritten by 'GGally':
```

```
  method from
+.gg   ggplot2
```

Scatterplot Matrix for hydrate data



What can we conclude here?

- It looks like `recov.score` has a moderately strong negative relationship with both `age` and `weight` (with correlations in each case around -0.5), but a positive relationship with `dose` (correlation = 0.36).
- The distribution of `recov.score` looks to be pretty close to Normal. No potential predictors (`age`, `weight` and `dose`) show substantial non-Normality.
- `age` and `weight`, as we'd expect, show a very strong and positive linear relationship, with $r = 0.94$
- Neither `age` nor `weight` shows a meaningful relationship with `dose`. ($r = 0.16$)

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

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

```
p1 <- ggplot(hydrate, aes(sample = recov.score)) +
  geom_qq(col = '#440154') + geom_qq_line(col = "red") +
  theme(aspect.ratio = 1) +
  labs(title = "Normal Q-Q plot: hydrate")

p2 <- ggplot(hydrate, aes(x = recov.score)) +
```

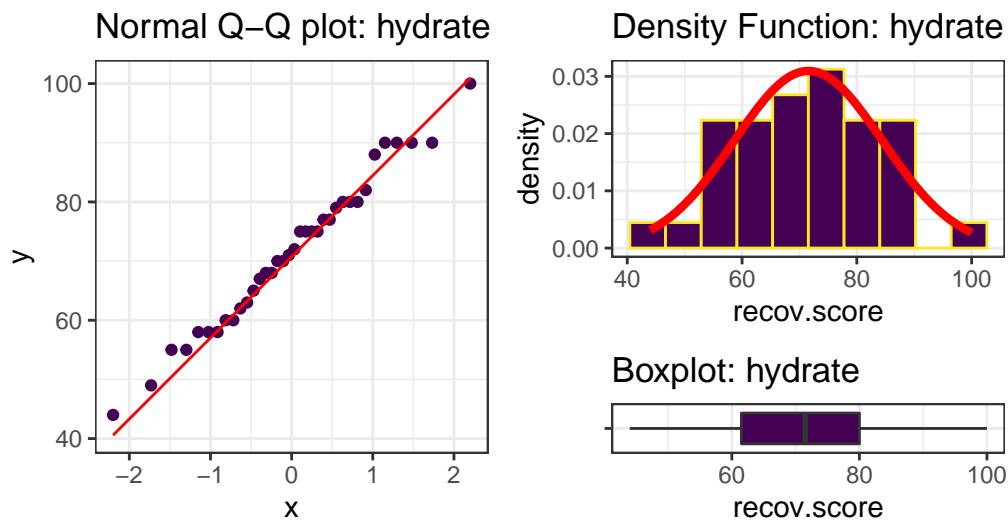
```

geom_histogram(aes(y = stat(density)),
               bins = 10, fill = '#440154', col = '#FDE725') +
stat_function(fun = dnorm,
              args = list(mean = mean(hydrate$recov.score),
                          sd = sd(hydrate$recov.score)),
              col = "red", lwd = 1.5) +
labs(title = "Density Function: hydrate")

p3 <- ggplot(hydrate, aes(x = recov.score, y = "")) +
geom_boxplot(fill = '#440154', outlier.color = '#440154') +
labs(title = "Boxplot: hydrate", y = "")

p1 + (p2 / p3 + plot_layout(heights = c(4,1)))

```



```
mosaic::favstats(~ recov.score, data = hydrate) |> kable(digits = 1)
```

```

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

```

| min | Q1 | median | Q3 | max | mean | sd | n | missing |
|-----|------|--------|----|-----|------|------|----|---------|
| 44 | 61.5 | 71.5 | 80 | 100 | 71.6 | 12.9 | 36 | 0 |

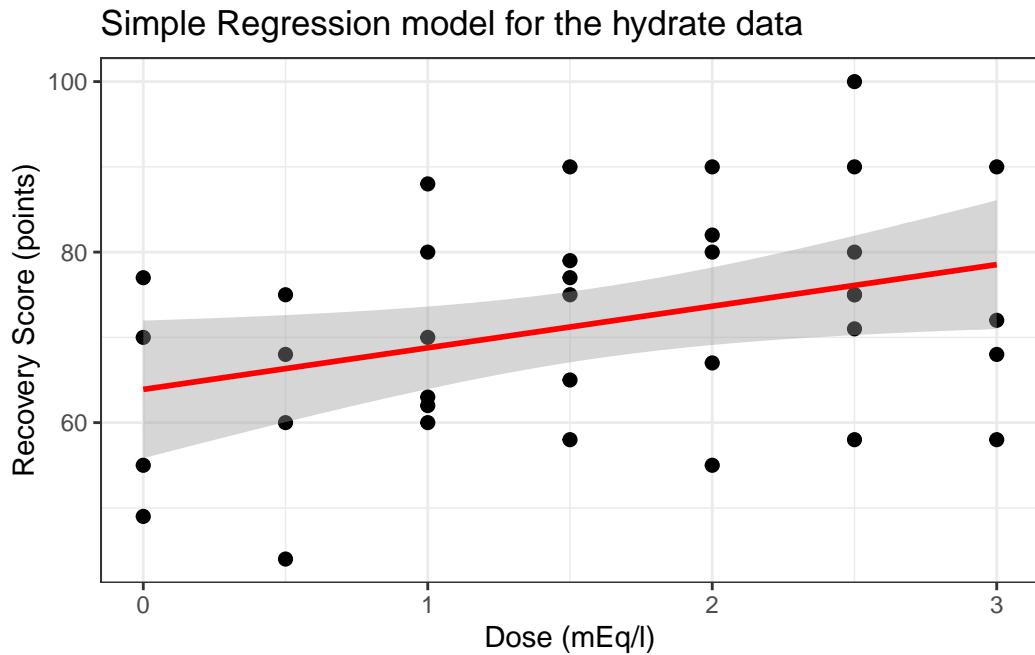
I see no serious problems with assuming Normality for these recovery scores. Our outcome variable doesn't in any way *need* to follow a Normal distribution, but it's nice when it does, because summaries involving means and standard deviations make sense.

15.5 Simple Regression: Using Dose to predict Recovery

To start, consider a simple (one predictor) regression model using `dose` alone to predict the % Recovery (`recov.score`). Ignoring the `age` and `weight` covariates, what can we conclude about this relationship?

15.6 The Scatterplot, with fitted Linear Model

```
ggplot(hydrate, aes(x = dose, y = recov.score)) +
  geom_point(size = 2) +
  geom_smooth(method = "lm", formula = y ~ x, col = "red") +
  labs(title = "Simple Regression model for the hydrate data",
       x = "Dose (mEq/l)", y = "Recovery Score (points)")
```



15.7 The Fitted Linear Model

To obtain the fitted linear regression model, we use the `lm` function:

```
m1 <- lm(recov.score ~ dose, data = hydrate)

tidy(m1) |> kable(digits = 2)
```

| term | estimate | std.error | statistic | p.value |
|-------------|----------|-----------|-----------|---------|
| (Intercept) | 63.90 | 3.97 | 16.09 | 0.00 |
| dose | 4.88 | 2.17 | 2.25 | 0.03 |

So, our fitted regression model (prediction model) is `recov.score = 63.9 + 4.88 dose`.

15.7.1 Confidence Intervals

We can obtain confidence intervals around the coefficients of our fitted model with `tidy`, too.

```
tidy(m1, conf.int = TRUE, conf.level = 0.90) |> kable(digits = 2)
```

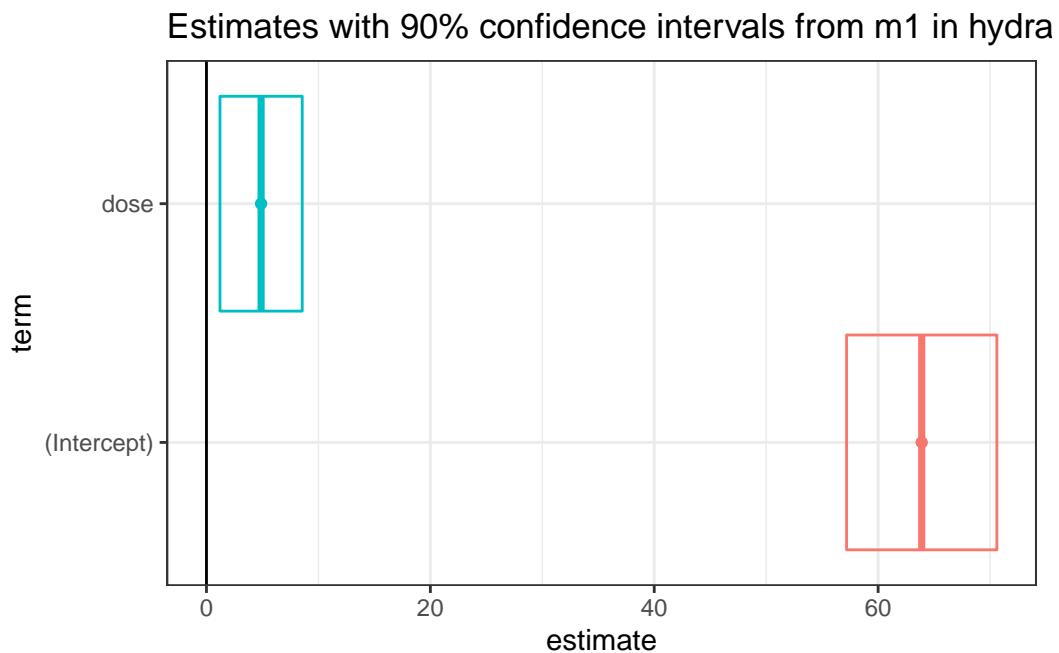
| term | estimate | std.error | statistic | p.value | conf.low | conf.high |
|-------------|----------|-----------|-----------|---------|----------|-----------|
| (Intercept) | 63.90 | 3.97 | 16.09 | 0.00 | 57.18 | 70.61 |
| dose | 4.88 | 2.17 | 2.25 | 0.03 | 1.21 | 8.55 |

So, our 90% confidence interval for the slope of dose ranges from 1.21 to 8.55.

15.8 Coefficient Plots

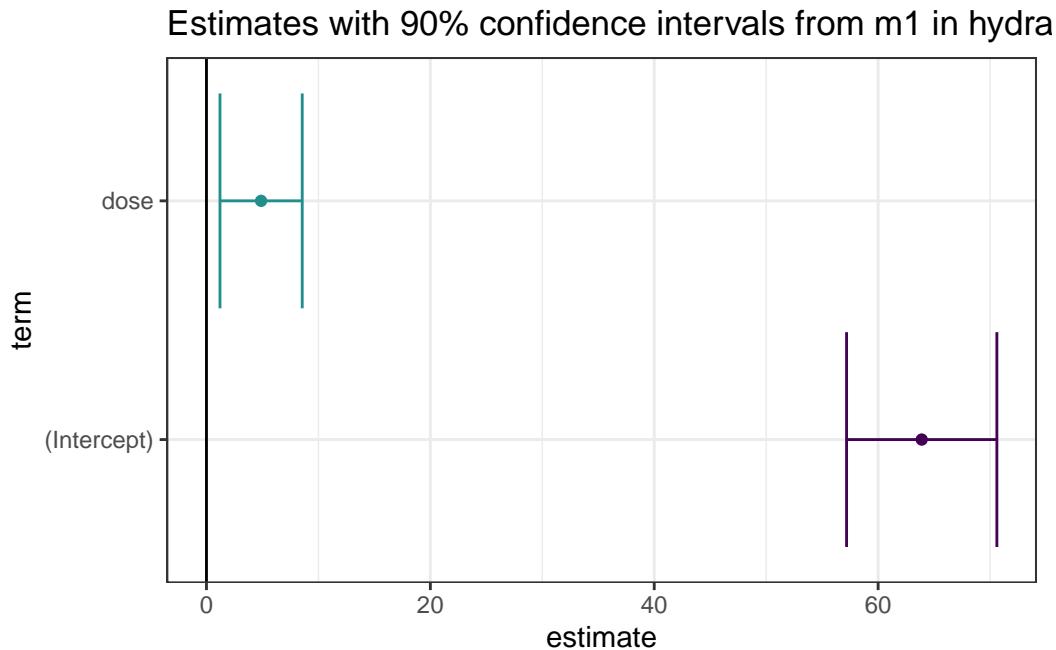
The `tidy` method makes it easy to construct coefficient plots using `ggplot2`.

```
td <- tidy(m1, conf.int = TRUE, conf.level = 0.90)
ggplot(td, aes(x = estimate, y = term, col = term)) +
  geom_point() +
  geom_crossbar(aes(xmin = conf.low, xmax = conf.high)) +
  geom_vline(xintercept = 0) +
  guides(col = "none") +
  labs(title = "Estimates with 90% confidence intervals from m1 in hydrate")
```



Another option would be to use `geom_errorbarh` in this setting, perhaps with a different color scheme...

```
td <- tidy(m1, conf.int = TRUE, conf.level = 0.90)
ggplot(td, aes(x = estimate, y = term, col = term)) +
  geom_point() +
  geom_errorbarh(aes(xmin = conf.low, xmax = conf.high)) +
  geom_vline(xintercept = 0) +
  scale_color_viridis_d(end = 0.5) +
  guides(col = "none") +
  labs(title = "Estimates with 90% confidence intervals from m1 in hydrate")
```



15.9 The Summary Output

To get a more complete understanding of the fitted model, we'll summarize it.

```
summary(lm(recov.score ~ dose, data = hydrate))
```

Call:

```

lm(formula = recov.score ~ dose, data = hydrate)

Residuals:
    Min      1Q  Median      3Q     Max 
-22.3360 -7.2763  0.0632  8.4233 23.9028 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 63.896     3.970   16.093 <2e-16 ***
dose        4.881     2.172    2.247   0.0313 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 12.21 on 34 degrees of freedom
Multiple R-squared:  0.1293,    Adjusted R-squared:  0.1037 
F-statistic: 5.047 on 1 and 34 DF,  p-value: 0.03127

```

15.9.1 Model Specification

1. The first part of the output specifies the model that has been fit.
 - Here, we have a simple regression model that predicts `recov.score` on the basis of `dose`.
 - Notice that we're treating `dose` here as a quantitative variable. If we wanted `dose` to be treated as a factor, we'd have specified that in the model.

15.9.2 Residual Summary

2. The second part of the output summarizes the regression **residuals** across the subjects involved in fitting the model.
 - The **residual** is defined as the Actual value of our outcome minus the predicted value of that outcome fitted by the model.
 - In our case, the residual for a given child is their actual `recov.score` minus the predicted `recov.score` according to our model, for that child.
 - The residual summary gives us a sense of how “incorrect” our predictions are for the `hydrate` observations.
 - A positive residual means that the observed value was higher than the predicted value from the linear regression model, so the prediction was too low.
 - A negative residual means that the observed value was lower than the predicted value from the linear regression model, so the prediction was too high.

- The residuals will center near 0 (the ordinary least squares model fitting process is designed so the mean of the residuals will always be zero)
- We hope to see the median of the residuals also be near zero, generally. In this case, the median prediction is 0.06 point too low.
- The minimum and maximum show us the largest prediction errors, made in the subjects used to fit this model.
- Here, we predicted a recovery score that was 22.3 points too high for one patient, and another of our predicted recovery scores was 23.9 points too low.
- The middle half of our predictions were between 8.4 points too low and 7.3 points too high.

15.9.3 Coefficients Output

- The **Coefficients** output begins with a table of the estimated coefficients from the regression equation.
 - Generally, we write a simple regression model as $y = \beta_0 + \beta_1 x$.
 - In the `hydrate` model, we have `recov.score = \beta_0 + \beta_1 dose`.
 - The first column of the table gives the estimated β coefficients for our model
 - Here the estimated intercept $\hat{\beta}_0 = 63.9$
 - The estimated slope of dose $\hat{\beta}_1 = 4.88$
 - Thus, our model is `recov.score = 63.9 + 4.88 dose`

We interpret these coefficients as follows:

- The intercept (63.9) is the predicted `recov.score` for a patient receiving a `dose` of 0 mEq/l of the electrolytic solution.
- The slope (4.88) of the `dose` is the predicted *change* in `recov.score` associated with a 1 mEq/l increase in the dose of electrolytic solution.
 - Essentially, if we have two children like the ones studied here, and we give Roger a popsicle with dose X and Sarah a popsicle with dose X + 1, then this model predicts that Sarah will have a recovery score that is 4.88 points higher than will Roger.
 - From the confidence interval output we saw previously with the function `confint(lm(recov.score ~ dose))`, we are 95% confident that the true slope for `dose` is between (0.47, 9.30) mEq/l. We are also 95% confident that the true intercept is between (55.8, 72.0).

15.9.4 Correlation and Slope

If we like, we can use the `cor` function to specify the Pearson correlation of `recov.score` and `dose`, which turns out to be 0.36. - Note that the `slope` in a simple regression model will

follow the sign of the Pearson correlation coefficient, in this case, both will be positive.

```
hydrate |> select(recov.score, dose) |> cor()
```

```
      recov.score      dose
recov.score    1.000000 0.359528
dose          0.359528 1.000000
```

15.9.5 Coefficient Testing

```
summary(lm(recov.score ~ dose, data = hydrate))
```

Call:

```
lm(formula = recov.score ~ dose, data = hydrate)
```

Residuals:

| Min | 1Q | Median | 3Q | Max |
|----------|---------|--------|--------|---------|
| -22.3360 | -7.2763 | 0.0632 | 8.4233 | 23.9028 |

Coefficients:

| | Estimate | Std. Error | t value | Pr(> t) |
|-------------|----------|------------|---------|------------|
| (Intercept) | 63.896 | 3.970 | 16.093 | <2e-16 *** |
| dose | 4.881 | 2.172 | 2.247 | 0.0313 * |

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 12.21 on 34 degrees of freedom

Multiple R-squared: 0.1293, Adjusted R-squared: 0.1037

F-statistic: 5.047 on 1 and 34 DF, p-value: 0.03127

Next to each coefficient in the summary regression table is its estimated standard error, followed by the coefficient's t value (the coefficient value divided by the standard error), and the associated two-tailed *p* value for the test of:

- H₀: This coefficient's β value = 0 vs.
- H_A: This coefficient's β value \neq 0.

For the slope coefficient, we can interpret this choice as:

- H₀: This predictor adds no predictive value to the model vs.
- H_A: This predictor adds some predictive value to the model.

In the `hydrate` simple regression model, by running either `tidy` with or just the `confint` function shown below, we can establish a confidence interval for each of the estimated regression coefficients.

```
tidy(m1, conf.int = TRUE, conf.level = 0.95) |> kable(digits = 2)
```

| term | estimate | std.error | statistic | p.value | conf.low | conf.high |
|-------------|----------|-----------|-----------|---------|----------|-----------|
| (Intercept) | 63.90 | 3.97 | 16.09 | 0.00 | 55.83 | 71.96 |
| dose | 4.88 | 2.17 | 2.25 | 0.03 | 0.47 | 9.30 |

```
confint(m1, level = .95)
```

| | 2.5 % | 97.5 % |
|-------------|-----------|-----------|
| (Intercept) | 55.826922 | 71.964589 |
| dose | 0.465695 | 9.295466 |

If the slope of dose was in fact zero, then this would mean that knowing the dose information would be of no additional value in predicting the outcome over just guessing the mean of `recov.score` for every subject.

So, since the confidence interval for the slope of dose does not include zero, it appears that there is at least some evidence that the model `m1` is more effective than a model that ignores the `dose` information (and simply predicts the mean of `recov.score` for each subject.) That's not saying much, actually.

15.9.6 Summarizing the Quality of Fit

4. The next part of the regression summary output is a summary of fit quality.

The **residual standard error** estimates the standard deviation of the prediction errors made by the model.

- If assumptions hold, the model will produce residuals that follow a Normal distribution with mean 0 and standard deviation equal to this residual standard error.
 - So we'd expect roughly 95% of our residuals to fall between -2(12.21) and +2(12.21), or roughly -24.4 to +24.4 and that we'd see virtually no residuals outside the range of -3(12.21) to +3(12.21), or roughly -36.6 to +36.6.

- The output at the top of the summary tells us about the observed regression residuals, and that they actually range from -22 to +24.
- In context, it's hard to know whether or not we should be happy about this. On a scale from 0 to 100, rarely missing by more than 24 seems OK to me, but not terrific.
- The **degrees of freedom** here are the same as the denominator degrees of freedom in the ANOVA to follow. The calculation is $n - k$, where n = the number of observations and k is the number of coefficients estimated by the regression (including the intercept and any slopes).
 - Here, there are 36 observations in the model, and we fit $k = 2$ coefficients; the slope and the intercept, as in any simple regression model, so $df = 36 - 2 = 34$.

The multiple R-squared value is usually just referred to as R-squared.

- This is interpreted as the proportion of variation in the outcome variable that has been accounted for by our regression model.
 - Here, we've accounted for just under 13% of the variation in % Recovery using Dose.
- The R in multiple R-squared is the Pearson correlation of `recov.score` and `dose`, which in this case is 0.3595.
 - Squaring this value gives the R-squared for this simple regression.
 - $(0.3595)^2 = 0.129$

R-squared is greedy.

- R-squared will always suggest that we make our models as big as possible, often including variables of dubious predictive value.
- As a result, there are various methods for adjusting or penalizing R-squared so that we wind up with smaller models.
- The **adjusted R-squared** is often a useful way to compare multiple models for the same response.
 - $R_{adj}^2 = 1 - \frac{(1-R^2)(n-1)}{n-k}$, where n = the number of observations and k is the number of coefficients estimated by the regression (including the intercept and any slopes).
 - So, in this case, $R_{adj}^2 = 1 - \frac{(1-0.1293)(35)}{34} = 0.1037$
 - The adjusted R-squared value is not, technically, a proportion of anything, but it is comparable across models for the same outcome.
 - The adjusted R-squared will always be less than the (unadjusted) R-squared.

15.9.7 ANOVA F test

5. The last part of the standard summary of a regression model is the overall ANOVA F test.

The hypotheses for this test are:

- H₀: Each of the coefficients in the model (other than the intercept) has $\beta = 0$ vs.
- H_A: At least one regression slope has $\beta \neq 0$

Since we are doing a simple regression with just one predictor, the ANOVA F test hypotheses are exactly the same as the t test for dose:

- H₀: The slope for `dose` has $\beta = 0$ vs.
- H_A: The slope for `dose` has $\beta \neq 0$

In this case, we have an F statistic of 5.05 on 1 and 34 degrees of freedom, yielding $p = 0.03$

This provides some evidence that “something” in our model (here, `dose` is the only predictor) predicts the outcome to a degree beyond that easily attributed to chance alone. This is not actually surprising, nor is it especially interesting. The confidence interval for the slope is definitely more interesting than this.

- In *simple regression* (regression with only one predictor), the t test for the slope (`dose`) always provides the same p value as the ANOVA F test.
 - The F test statistic in a *simple regression* is always by definition just the square of the slope’s t test statistic.
 - Here, $F = 5.047$, and this is the square of $t = 2.247$ from the Coefficients output

This test is basically just a combination of the R-squared value (13%) and the sample size. We don’t learn much from it that’s practically interesting or useful.

15.10 Viewing the complete ANOVA table

We can obtain the complete ANOVA table associated with this particular model, and the details behind this F test using the `anova` function:

```
anova(lm(recov.score ~ dose, data = hydrate))
```

Analysis of Variance Table

```
Response: recov.score
          Df Sum Sq Mean Sq F value Pr(>F)
dose       1 752.2 752.15  5.0473 0.03127 *
Residuals 34 5066.7 149.02
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- The R-squared for our regression model is equal to the η^2 for this ANOVA model.
 - If we divide $SS(dose) = 752.2$ by the total sum of squares ($752.2 + 5066.7$), we'll get the multiple R-squared [0.1293]
- Note that this is *not* the same ANOVA model we would get if we treated `dose` as a factor with seven levels, rather than as a quantitative variable.

15.11 Using `glance` to summarize the model's fit

When applied to a linear model, the `glance` function from the `broom` package summarizes 12 characteristics of the model's fit.

Let's look at the eight of these that we've already addressed.

```
glance(m1) |> select(r.squared:df, df.residual, nobs) |>
  kable(digits = c(3, 3, 1, 2, 3, 0, 0, 0))
```

| r.squared | adj.r.squared | sigma | statistic | p.value | df | df.residual | nobs |
|-----------|---------------|-------|-----------|---------|----|-------------|------|
| 0.129 | 0.104 | 12.2 | 5.05 | 0.031 | 1 | 34 | 36 |

- We've discussed the R-square value, shown in `r.squared`.
- We've also discussed the adjusted R-square value, in `adj.r.squared`
- `sigma` is the residual standard error.
- `statistic` is the ANOVA F statistic.
- `p.value` is the p value associated with the ANOVA F statistic.
- `df` is the numerator degrees of freedom (here, the `df` associated with `dose`) for the ANOVA test associated with this model.
- `df.residual` is the denominator degrees of freedom (here the `df` associated with `residual`) for that same ANOVA test.
- Remember that the F-statistic at the bottom of the summary output provides these last four statistics, as well.
- `nobs` is the number of observations (rows) used to fit the model.

Now, let's look at the remaining four summaries:

```
glance(m1) |> select(logLik:deviance) |>  
kable(digits = 1)
```

| logLik | AIC | BIC | deviance |
|--------|-------|-----|----------|
| -140.1 | 286.3 | 291 | 5066.7 |

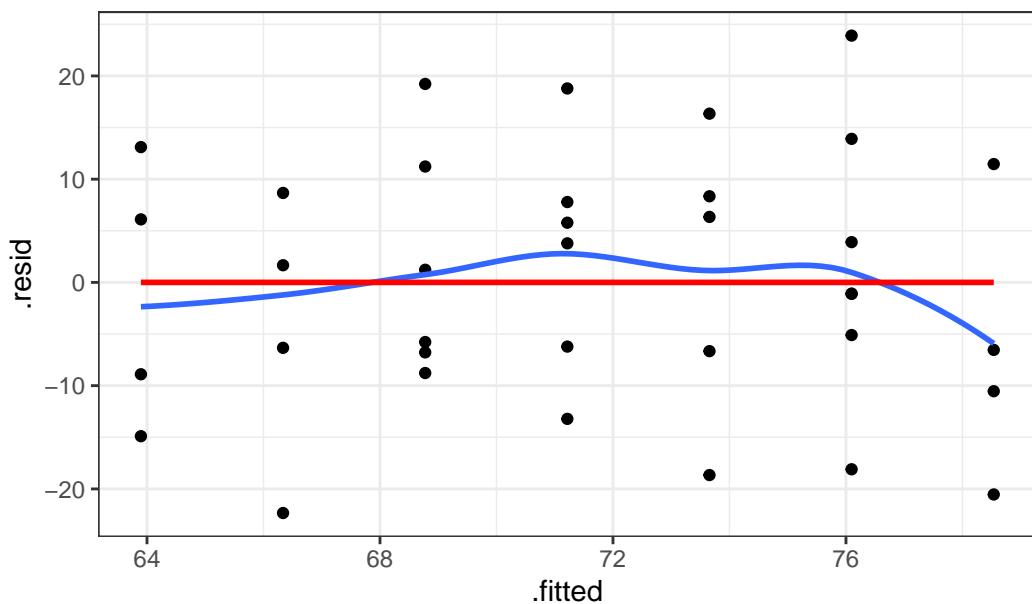
- **logLik** is the log-likelihood value for the model, and is most commonly used for a model (like the ordinary least squares model fit by `lm` that is fit using the method of maximum likelihood). Thus, the log-likelihood value will be maximized in this fit.
- **AIC** is the Akaike Information Criterion for the model. When comparing models fitted by maximum likelihood to the same outcome variable (using the same transformation, for example), the smaller the AIC, the better the fit.
- **BIC** is the Bayes Information Criterion for the model. When comparing models fitted by maximum likelihood to the same outcome variable (using the same transformation, for example), the smaller the BIC, the better the fit. BIC often prefers models with fewer coefficients to estimate than does AIC.
 - AIC and BIC can be estimated using several different approaches in R, but we'll need to use the same one across multiple models if we're comparing the results, because the concepts are only defined up to a constant.
- **deviance** is the fitted model's deviance, a measure of lack of fit. It is a generalization of the residual sum of squares seen in the ANOVA table, and takes the same value in the case of a simple linear regression model fit with `lm` as we have here. For some generalized linear models, we'll use this for hypothesis testing, just as the ANOVA table does in the linear model case.

15.12 Plotting Residuals vs. Fitted Values

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

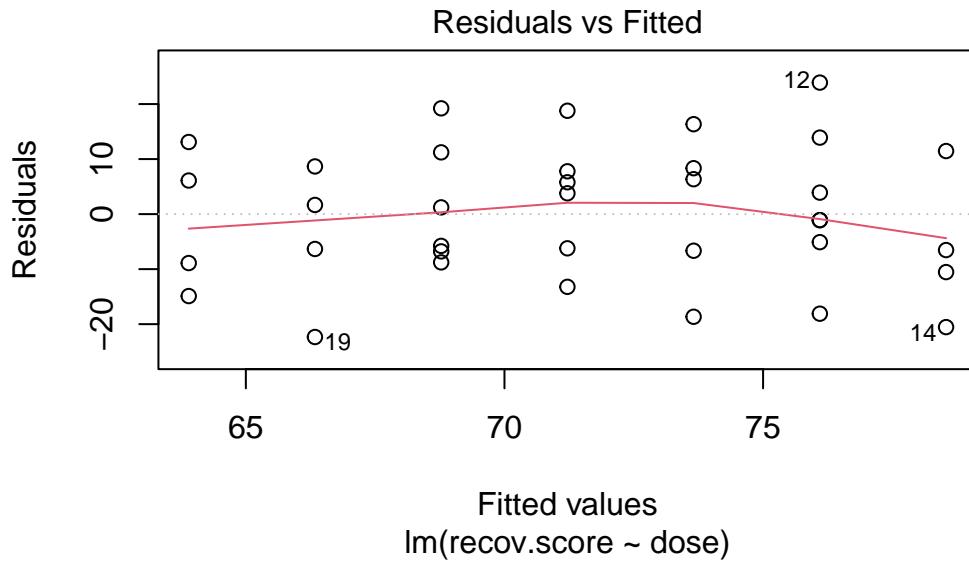
```
augment(m1) %>%  
  ggplot(., aes(x = .fitted, y = .resid)) +  
  geom_point() +  
  geom_smooth(method = "loess", formula = y ~ x, se = F) +  
  geom_smooth(method = "lm", formula = y ~ x, se = F, col = "red") +  
  labs(title = "Residuals vs. Fitted values for Model m1")
```

Residuals vs. Fitted values for Model m1



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

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



We hope in this plot to see a generally random scatter of points, perhaps looking like a “fuzzy football”. Since we only have seven possible `dose` values, we obtain only seven distinct predicted values, which explains the seven vertical lines in the plot. Here, the smooth red line indicates a gentle curve, but no evidence of a strong curve, or any other regular pattern in this residual plot.

16 The WCGS

16.1 Setup: Packages Used Here

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

library(knitr)
library(janitor)
library(broom)
library(patchwork)
library(tidyverse)

theme_set(theme_bw())
```

We will also use the `geom_density_ridges` function from the `ggridges` package, and the `favstats` function from the `mosaic` package, and the `ggpairs` function from the `GGally` package.

16.2 The Western Collaborative Group Study (wcgs) data set

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

The Western Collaborative Group Study (WCGS) was designed to test the hypothesis that the so-called Type A behavior pattern (TABP) - “characterized particularly by excessive drive, aggressiveness, and ambition, frequently in association with a relatively greater preoccupation with competitive activity, vocational deadlines, and similar pressures” - is a cause of coronary heart disease (CHD). Two additional goals, developed later in the study, were (1) to investigate the comparability of formulas developed in WCGS and in the Framingham Study (FS) for prediction of CHD risk, and (2) to determine how addition of TABP to an existing

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

multivariate prediction formula affects ability to select subjects for intervention programs.

The study enrolled over 3,000 men ages 39-59 who were employed in San Francisco or Los Angeles, during 1960 and 1961.

```
wcgs <- read_csv("data/wcgs.csv") |>
  mutate(across(where(is.character), as_factor))

wcgs

# A tibble: 3,154 x 22
   id    age agec   height weight lnwght wghtcat   bmi    sbp lnsbp    dbp    chol
   <dbl> <dbl> <fct>  <dbl>   <dbl>   <dbl> <fct>   <dbl> <dbl> <dbl> <dbl> <dbl>
 1 2343    50 46-50     67    200    5.30 170-200  31.3   132   4.88    90   249
 2 3656    51 51-55     73    192    5.26 170-200  25.3   120   4.79    74   194
 3 3526    59 56-60     70    200    5.30 170-200  28.7   158   5.06    94   258
 4 22057   51 51-55     69    150    5.01 140-170  22.1   126   4.84    80   173
 5 12927   44 41-45     71    160    5.08 140-170  22.3   126   4.84    80   214
 6 16029   47 46-50     64    158    5.06 140-170  27.1   116   4.75    76   206
 7 3894    40 35-40     70    162    5.09 140-170  23.2   122   4.80    78   190
 8 11389   41 41-45     70    160    5.08 140-170  23.0   130   4.87    84   212
 9 12681   50 46-50     71    195    5.27 170-200  27.2   112   4.72    70   130
10 10005   43 41-45     68    187    5.23 170-200  28.4   120   4.79    80   233
# ... with 3,144 more rows, and 10 more variables: behpat <fct>, dibpat <fct>,
#   smoke <fct>, ncigs <dbl>, arcus <dbl>, chd69 <fct>, typchd69 <dbl>,
#   time169 <dbl>, t1 <dbl>, uni <dbl>
# i Use `print(n = ...)` to see more rows, and `colnames()` to see all variable names
```

Here, we have 3154 rows (subjects) and 22 columns (variables). After importing the data and creating a tibble with `read_csv`, I used `mutate(across(where(is.character), as_factor))` to convert all variables containing character data into factors.

16.2.1 Structure of `wcgs`

We can specify the (sometimes terrible) variable names, through the `names` function, or we can add other elements of the structure, so that we can identify elements of particular interest.

```
str(wcgs)
```

```

tibble [3,154 x 22] (S3: tbl_df/tbl/data.frame)
$ id      : num [1:3154] 2343 3656 3526 22057 12927 ...
$ age     : num [1:3154] 50 51 59 51 44 47 40 41 50 43 ...
$ agec    : Factor w/ 5 levels "46-50","51-55",...: 1 2 3 2 4 1 5 4 1 4 ...
$ height  : num [1:3154] 67 73 70 69 71 64 70 70 71 68 ...
$ weight  : num [1:3154] 200 192 200 150 160 158 162 160 195 187 ...
$ lnwght  : num [1:3154] 5.3 5.26 5.3 5.01 5.08 ...
$ wghtcat: Factor w/ 4 levels "170-200","140-170",...: 1 1 1 2 2 2 2 1 1 ...
$ bmi     : num [1:3154] 31.3 25.3 28.7 22.1 22.3 ...
$ sbp     : num [1:3154] 132 120 158 126 126 116 122 130 112 120 ...
$ lnsbp   : num [1:3154] 4.88 4.79 5.06 4.84 4.84 ...
$ dbp     : num [1:3154] 90 74 94 80 80 76 78 84 70 80 ...
$ chol    : num [1:3154] 249 194 258 173 214 206 190 212 130 233 ...
$ behpat  : Factor w/ 4 levels "A1","A2","B3",...: 1 1 1 1 1 1 1 1 1 1 ...
$ dibpat  : Factor w/ 2 levels "Type A","Type B": 1 1 1 1 1 1 1 1 1 1 ...
$ smoke   : Factor w/ 2 levels "Yes","No": 1 1 2 2 2 1 2 1 2 1 ...
$ ncigs   : num [1:3154] 25 25 0 0 0 80 0 25 0 25 ...
$ arcus   : num [1:3154] 1 0 1 1 0 0 0 0 1 0 ...
$ chd69   : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
$ typchd69: num [1:3154] 0 0 0 0 0 0 0 0 0 0 ...
$ time169 : num [1:3154] 1367 2991 2960 3069 3081 ...
$ t1      : num [1:3154] -1.63 -4.06 0.64 1.12 2.43 ...
$ uni     : num [1:3154] 0.486 0.186 0.728 0.624 0.379 ...

```

16.2.2 Codebook for wcgs

This table was lovingly hand-crafted, and involved a lot of typing. We'll look for better ways in 432.

| Name | Stored As | Type | Details (units, levels, etc.) |
|---------|------------|--------------|---|
| id | integer | (nominal) | ID #, nominal and uninteresting |
| age | integer | quantitative | age, in years - no decimal places |
| agec | factor (5) | (ordinal) | age: 35-40, 41-45, 46-50, 51-55, 56-60 |
| height | integer | quantitative | height, in inches |
| weight | integer | quantitative | weight, in pounds |
| lnwght | number | quantitative | natural logarithm of weight |
| wghtcat | factor (4) | (ordinal) | wt: < 140, 140-170, 170-200, > 200 |
| bmi | number | quantitative | body-mass index: $703 * \text{weight in lb} / (\text{height in in})^2$ |
| sbp | integer | quantitative | systolic blood pressure, in mm Hg |
| lnsbp | number | quantitative | natural logarithm of sbp |
| dbp | integer | quantitative | diastolic blood pressure, mm Hg |

| Name | Stored As | Type | Details (units, levels, etc.) |
|----------|------------|--------------|---|
| chol | integer | quantitative | total cholesterol, mg/dL |
| behpat | factor (4) | (nominal) | behavioral pattern: A1, A2, B3 or B4 |
| dibpat | factor (2) | (binary) | behavioral pattern: A or B |
| smoke | factor (2) | (binary) | cigarette smoker: Yes or No |
| ncigs | integer | quantitative | number of cigarettes smoked per day |
| arcus | integer | (nominal) | arcus senilis present (1) or absent (0) |
| chd69 | factor (2) | (binary) | CHD event: Yes or No |
| typchd69 | integer | (4 levels) | event: 0 = no CHD, 1 = MI or SD, 2 = silent MI, 3 = angina |
| time169 | integer | quantitative | follow-up time in days |
| t1 | number | quantitative | heavy-tailed (random draws) |
| uni | number | quantitative | light-tailed (random draws) |

16.2.3 Quick Summary

```
summary(wcgs)
```

| id | age | agec | height | weight |
|----------------|---------------|---------------|---------------|---------------|
| Min. : 2001 | Min. :39.00 | 46-50: 750 | Min. :60.00 | Min. : 78 |
| 1st Qu.: 3741 | 1st Qu.:42.00 | 51-55: 528 | 1st Qu.:68.00 | 1st Qu.:155 |
| Median :11406 | Median :45.00 | 56-60: 242 | Median :70.00 | Median :170 |
| Mean : 10478 | Mean :46.28 | 41-45:1091 | Mean :69.78 | Mean :170 |
| 3rd Qu.:13115 | 3rd Qu.:50.00 | 35-40: 543 | 3rd Qu.:72.00 | 3rd Qu.:182 |
| Max. :22101 | Max. :59.00 | | Max. :78.00 | Max. :320 |
| | | | | |
| lnwght | wghtcat | bmi | sbp | lnsbp |
| Min. :4.357 | 170-200:1171 | Min. :11.19 | Min. : 98.0 | Min. :4.585 |
| 1st Qu.:5.043 | 140-170:1538 | 1st Qu.:22.96 | 1st Qu.:120.0 | 1st Qu.:4.787 |
| Median :5.136 | > 200 : 213 | Median :24.39 | Median :126.0 | Median :4.836 |
| Mean : 5.128 | < 140 : 232 | Mean :24.52 | Mean :128.6 | Mean :4.850 |
| 3rd Qu.:5.204 | | 3rd Qu.:25.84 | 3rd Qu.:136.0 | 3rd Qu.:4.913 |
| Max. :5.768 | | Max. :38.95 | Max. :230.0 | Max. :5.438 |
| | | | | |
| dbp | chol | behpat | dibpat | smoke |
| Min. : 58.00 | Min. :103.0 | A1: 264 | Type A:1589 | Yes:1502 |
| 1st Qu.: 76.00 | 1st Qu.:197.2 | A2:1325 | Type B:1565 | No :1652 |
| Median : 80.00 | Median :223.0 | B3:1216 | | |
| Mean : 82.02 | Mean :226.4 | B4: 349 | | |
| 3rd Qu.: 86.00 | 3rd Qu.:253.0 | | | |

```

Max.    :150.00   Max.    :645.0
NA's     :12

ncigs      arcus      chd69      typchd69      time169
Min.    : 0.0    Min.    :0.0000    No :2897    Min.    :0.0000    Min.    : 18
1st Qu.: 0.0    1st Qu.:0.0000    Yes: 257   1st Qu.:0.0000    1st Qu.:2842
Median   : 0.0    Median   :0.0000          Median :0.0000    Median   :2942
Mean     :11.6    Mean     :0.2985          Mean    :0.1363    Mean     :2684
3rd Qu.:20.0    3rd Qu.:1.0000          3rd Qu.:0.0000    3rd Qu.:3037
Max.     :99.0    Max.     :1.0000          Max.    :3.0000    Max.    :3430
NA's     :2

t1          uni
Min.    :-47.43147  Min.    :0.0007097
1st Qu.: -1.00337  1st Qu.:0.2573755
Median   : 0.00748  Median   :0.5157779
Mean     : -0.03336 Mean     :0.5052159
3rd Qu.:  0.97575  3rd Qu.:0.7559902
Max.     : 47.01623 Max.     :0.9994496
NA's     :39

```

For a more detailed description, we might consider `Hmisc::describe`, `psych::describe`, `mosaic::favstats`, etc.

16.3 Are the SBPs Normally Distributed?

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

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

```

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

bin_w <- 5 # specify binwidth

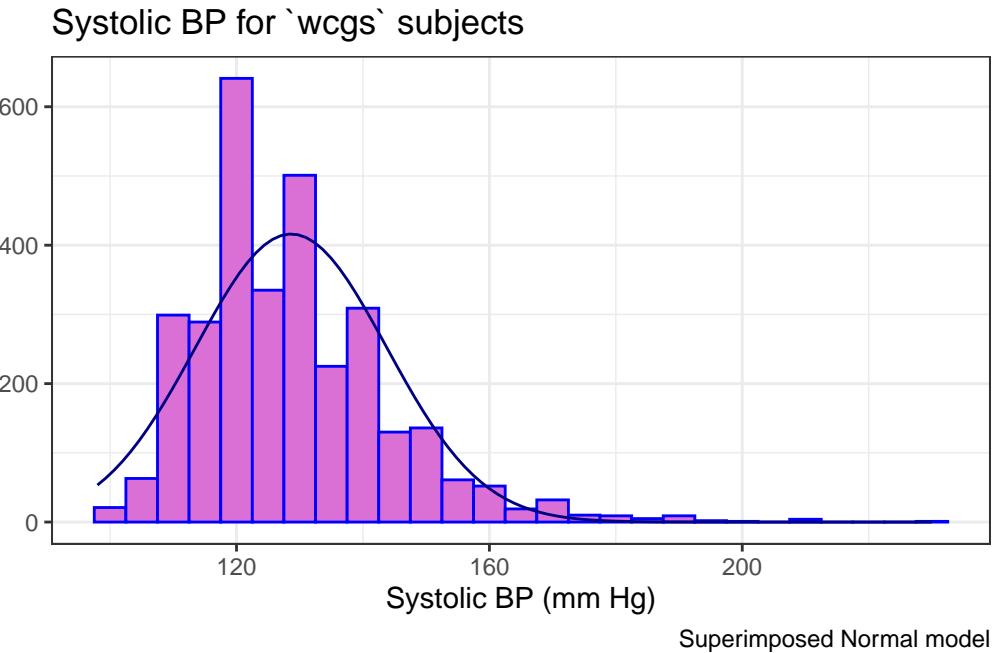
ggplot(wcgs, aes(x = sbp)) +
  geom_histogram(binwidth = bin_w,
                 fill = "orchid",
                 col = "blue") +

```

```

stat_function(
  fun = function(x) dnorm(x, mean = res$mean,
                           sd = res$sd) *
    res$n * bin_w,
  col = "navy") +
  labs(title = "Systolic BP for `wcgs` subjects",
       x = "Systolic BP (mm Hg)", y = "",
       caption = "Superimposed Normal model")

```



Since the data contain both `sbp` and `lnsbp` (its natural logarithm), let's compare them. Note that in preparing the graph, we'll need to change the location for the text annotation.

```

res <- mosaic::favstats(~ lnsbp, data = wcgs)
bin_w <- 0.05 # specify binwidth

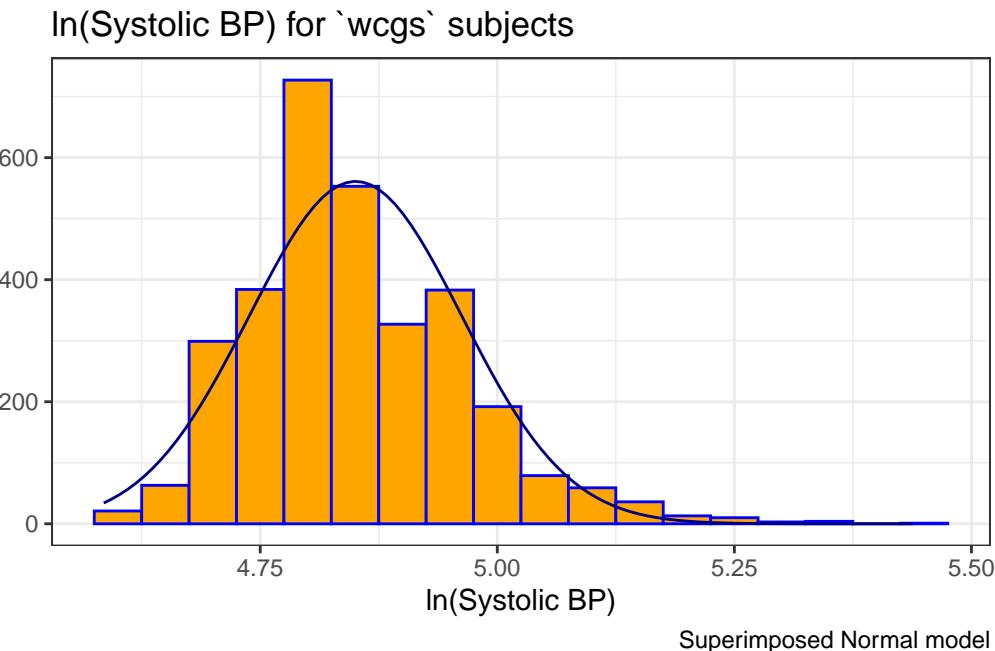
ggplot(wcgs, aes(x = lnsbp)) +
  geom_histogram(binwidth = bin_w,
                 fill = "orange",
                 col = "blue") +
  stat_function(
    fun = function(x) dnorm(x, mean = res$mean,

```

```

sd = res$sd) *
res$n * bin_w,
col = "navy") +
labs(title = "ln(Systolic BP) for `wcgs` subjects",
x = "ln(Systolic BP)", y = "",
caption = "Superimposed Normal model")

```



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

```

p1 <- ggplot(wcgs, aes(sample = sbp)) +
  geom_qq(color = "orchid") +
  geom_qq_line(color = "red") +
  labs(y = "Ordered SBP", title = "sbp in wcgs")

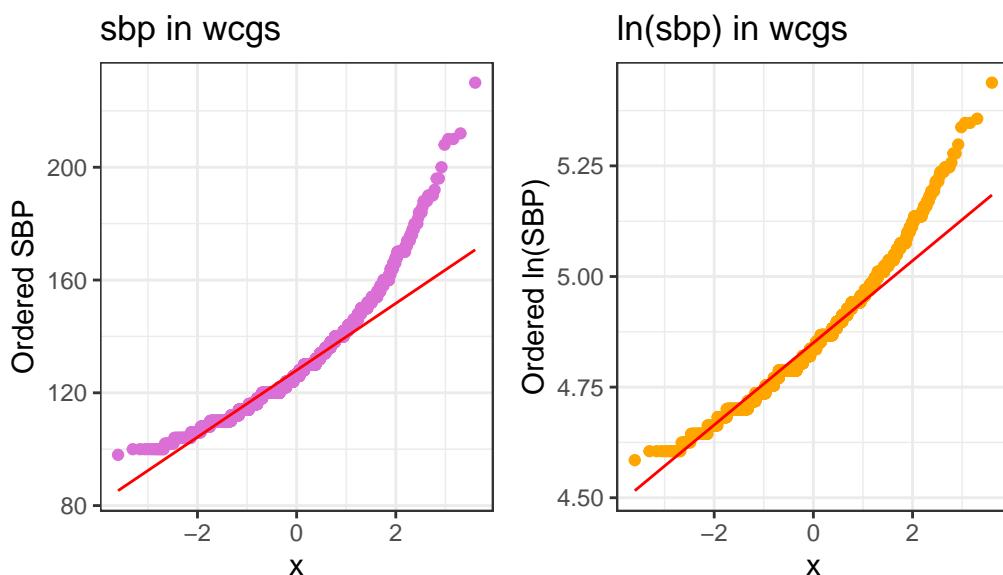
p2 <- ggplot(wcgs, aes(sample = lnsbp)) +
  geom_qq(color = "orange") +
  geom_qq_line(color = "red") +
  labs(y = "Ordered ln(SBP)", title = "ln(sbp) in wcgs")

## next step requires library(patchwork)

```

```
p1 + p2 +
  plot_annotation(title = "Normal Q-Q plots of SBP and ln(SBP) in wcgs")
```

Normal Q–Q plots of SBP and $\ln(\text{SBP})$ in `wcgs`



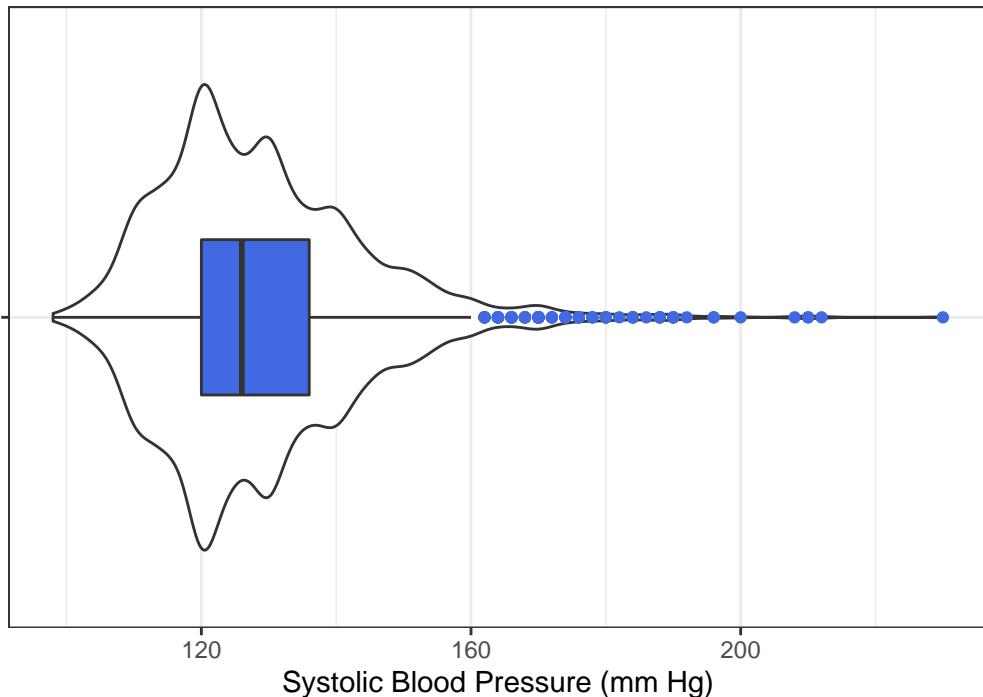
There's at best a small improvement from `sbp` to `lnsbp` in terms of approximation by a Normal distribution.

16.4 Identifying and Describing SBP outliers

It looks like there's an outlier (or a series of them) in the SBP data.

```
ggplot(wcgs, aes(x = "", y = sbp)) +
  geom_violin() +
  geom_boxplot(width = 0.3, fill = "royalblue",
               outlier.color = "royalblue") +
  labs(title = "Boxplot with Violin of SBP in `wcgs` data",
       y = "Systolic Blood Pressure (mm Hg)",
       x = "") +
  coord_flip()
```

Boxplot with Violin of SBP in `wcgs` data



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

| | n | missing | distinct | Info | Mean | Gmd | .05 | .10 |
|-----------|------|---------|----------|-------|-------|-------|-----|-----|
| wcgs\$sbp | 3154 | 0 | 62 | 0.996 | 128.6 | 16.25 | 110 | 112 |
| | .25 | .50 | .75 | .90 | .95 | | | |
| | 120 | 126 | 136 | 148 | 156 | | | |

lowest : 98 100 102 104 106, highest: 200 208 210 212 230

The maximum value here is 230, and is clearly the most extreme value in the data set. One way to gauge this is to describe that observation's **Z score**, the number of standard deviations away from the mean that the observation falls. Here, the maximum value, 230 is 6.71 standard deviations above the mean, and thus has a Z score of 6.7.

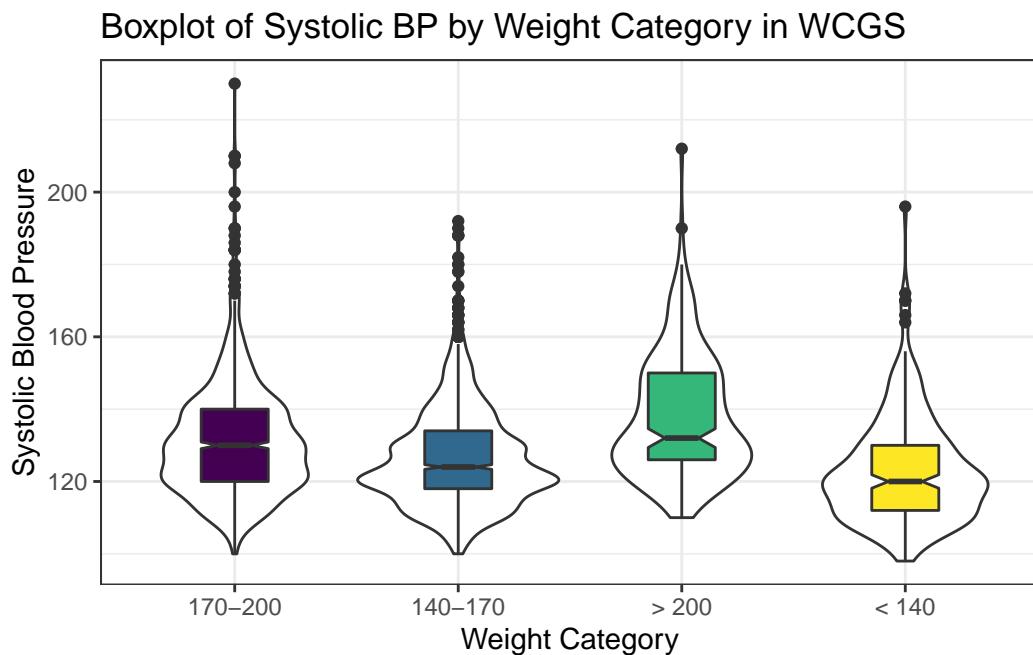
A negative Z score would indicate a point below the mean, while a positive Z score indicates, as we've seen, a point above the mean. The minimum systolic blood pressure, 98 is 2.03 standard deviations *below* the mean, so it has a Z score of -2.

Recall that the Empirical Rule suggests that if a variable follows a Normal distribution, it would have approximately 95% of its observations falling inside a Z score of (-2, 2), and 99.74% falling inside a Z score range of (-3, 3). Do the systolic blood pressures appear Normally distributed?

16.5 Does Weight Category Relate to SBP?

The data are collected into four groups based on the subject's weight (in pounds).

```
ggplot(wcgs, aes(x = wghtcat, y = sbp)) +  
  geom_violin() +  
  geom_boxplot(aes(fill = wghtcat), width = 0.3, notch = TRUE) +  
  scale_fill_viridis_d() +  
  guides(fill = "none") +  
  labs(title = "Boxplot of Systolic BP by Weight Category in WCGS",  
       x = "Weight Category", y = "Systolic Blood Pressure")
```



16.6 Re-Leveling a Factor

Well, that's not so good. We really want those weight categories (the *levels*) to be ordered more sensibly.

```
wcgs |> tabyl(wghtcat)
```

```
wghtcat      n      percent
170-200 1171 0.37127457
140-170 1538 0.48763475
> 200   213 0.06753329
< 140   232 0.07355739
```

Like all *factor* variables in R, the categories are specified as levels. We want to change the order of the levels in a new version of this factor variable so they make sense. There are multiple ways to do this, but I prefer the `fct_relevel` function from the `forcats` package (part of the tidyverse.) Which order is more appropriate?

I'll add a new variable to the `wcgs` data called `weight_f` that relevels the `wghtcat` data.

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

wcgs |> tabyl(weight_f)

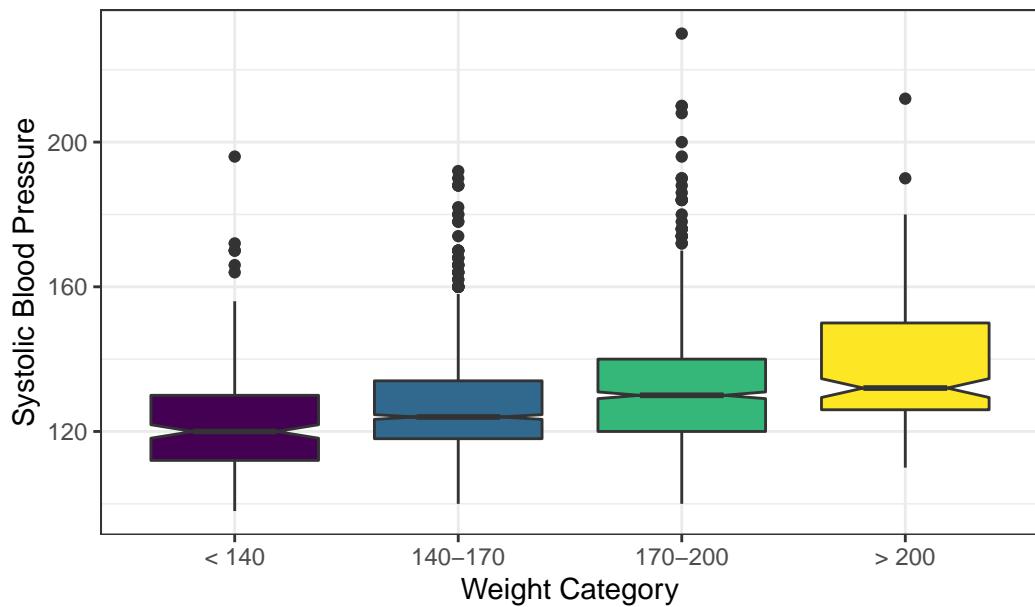
weight_f      n      percent
< 140   232 0.07355739
140-170 1538 0.48763475
170-200 1171 0.37127457
> 200   213 0.06753329
```

For more on the `forcats` package, check out Wickham and Grolemund (2022), especially the Section on Factors.

16.6.1 SBP by Weight Category

```
ggplot(wcgs, aes(x = weight_f, y = sbp, fill = weight_f)) +
  geom_boxplot(notch = TRUE) +
  scale_fill_viridis_d() +
  guides(fill = "none") +
  labs(title = "Systolic Blood Pressure by Reordered Weight Category in WCGS",
       x = "Weight Category", y = "Systolic Blood Pressure")
```

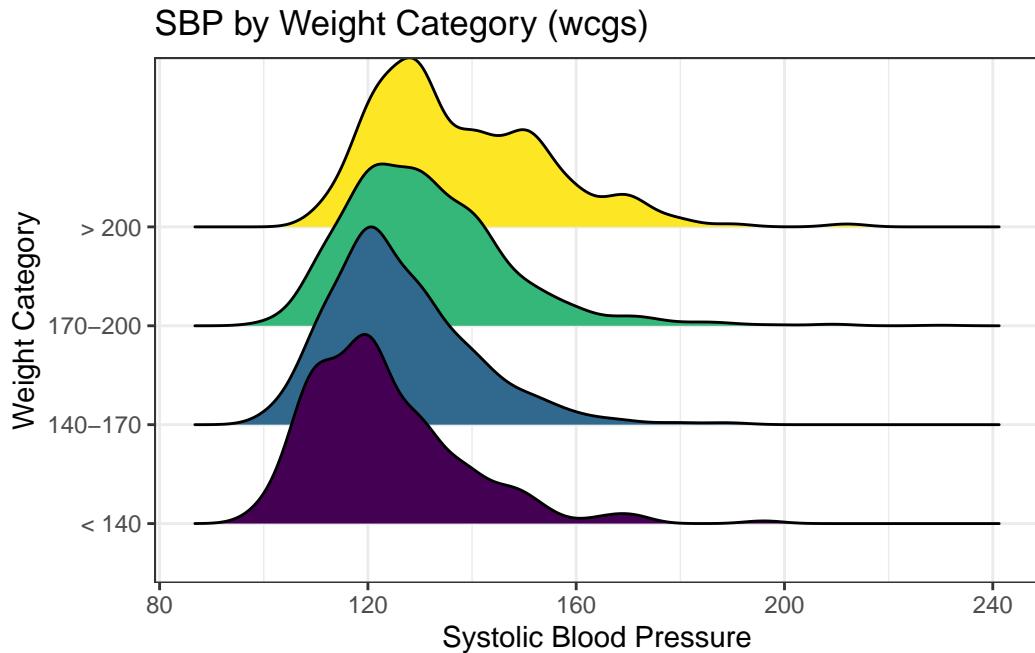
Systolic Blood Pressure by Reordered Weight Category in WC



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

```
ggplot(wcgs, aes(x = sbp, y = weight_f, fill = weight_f, height = ..density..)) +  
  ggridges::geom_density_ridges(scale = 2) +  
  scale_fill_viridis_d() +  
  guides(fill = "none") +  
  labs(title = "SBP by Weight Category (wcgs)",  
       x = "Systolic Blood Pressure",  
       y = "Weight Category")
```

Picking joint bandwidth of 3.74



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

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

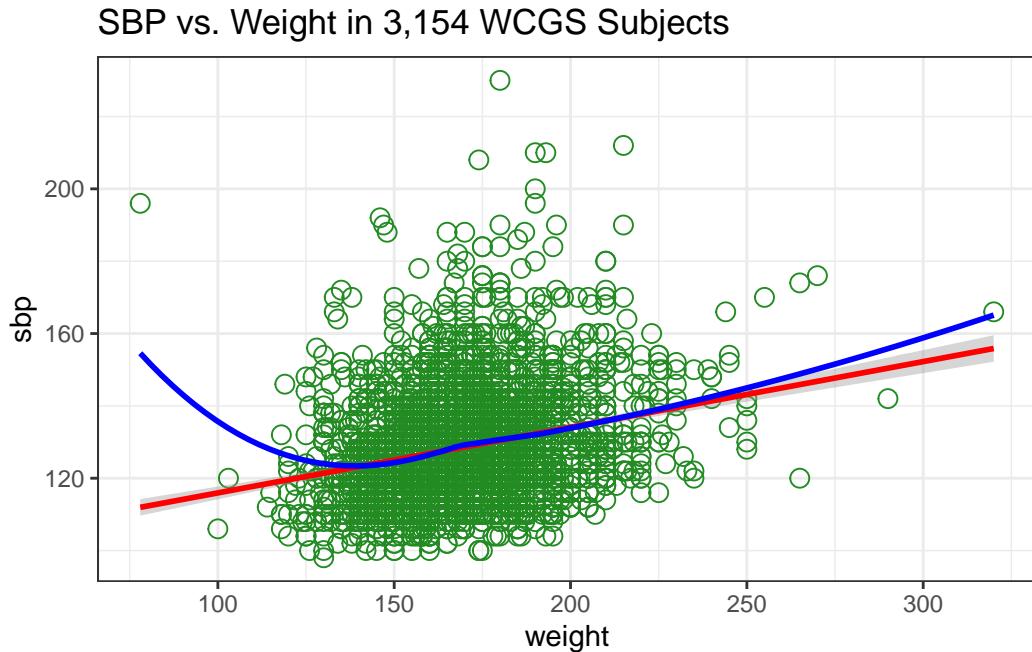
| | weight_f | min | Q1 | median | Q3 | max | mean | sd | n | missing |
|---|----------|-----|-----|--------|-----|-----|----------|----------|------|---------|
| 1 | < 140 | 98 | 112 | 120 | 130 | 196 | 123.1379 | 14.73394 | 232 | 0 |
| 2 | 140-170 | 100 | 118 | 124 | 134 | 192 | 126.2939 | 13.65294 | 1538 | 0 |
| 3 | 170-200 | 100 | 120 | 130 | 140 | 230 | 131.1136 | 15.57024 | 1171 | 0 |
| 4 | > 200 | 110 | 126 | 132 | 150 | 212 | 137.8685 | 16.75522 | 213 | 0 |

16.7 Are Weight and SBP Linked?

Let's build a scatter plot of SBP (Outcome) by Weight (Predictor), rather than breaking down into categories.

```
ggplot(wcgs, aes(x = weight, y = sbp)) +
  geom_point(size=3, shape=1, color="forestgreen") + ## default size = 2
  stat_smooth(method=lm, color="red") + ## add se=FALSE to hide conf. interval
  stat_smooth(method=loess, se=FALSE, color="blue") +
```

```
ggttitle("SBP vs. Weight in 3,154 WCGS Subjects")
```



- The mass of the data is hidden from us - showing 3154 points in one plot can produce little more than a blur where there are lots of points on top of each other.
- Here the least squares regression line (in red), and loess scatterplot smoother, (in blue) can help.

The relationship between systolic blood pressure and weight appears to be very close to linear, but of course there is considerable scatter around that generally linear relationship. It turns out that the Pearson correlation of these two variables is 0.253.

16.8 SBP and Weight by Arcus Senilis groups?

An issue of interest to us will be to assess whether the SBP-Weight relationship we see above is similar among subjects who have arcus senilis and those who do not.

Arcus senilis is an old age syndrome where there is a white, grey, or blue opaque ring in the corneal margin (peripheral corneal opacity), or white ring in front of the periphery of the iris. It is present at birth but then fades; however, it is quite commonly present in the elderly. It can also appear earlier in life as a result of hypercholesterolemia.

Wikipedia article on Arcus Senilis, retrieved 2017-08-15

Let's start with a quick look at the `arcus` data.

```
wcgs |> tabyl(arcus)
```

| arcus | n | percent | valid_percent |
|-------|------|--------------|---------------|
| 0 | 2211 | 0.7010145847 | 0.7014594 |
| 1 | 941 | 0.2983512999 | 0.2985406 |
| NA | 2 | 0.0006341154 | NA |

We have 2 missing values, so we probably want to do something about that before plotting the data, and we may also want to create a factor variable with more meaningful labels than 1 (which means yes, arcus senilis is present) and 0 (which means no, it isn't.)

```
wcgs <- wcgs |>  
  mutate(arcus_f = fct_recode(factor(arcus),  
    "Arcus senilis" = "1",  
    "No arcus senilis" = "0"),  
    arcus_f = fct_relevel(arcus_f, "Arcus senilis"))  
  
wcgs |> tabyl(arcus_f, arcus)
```

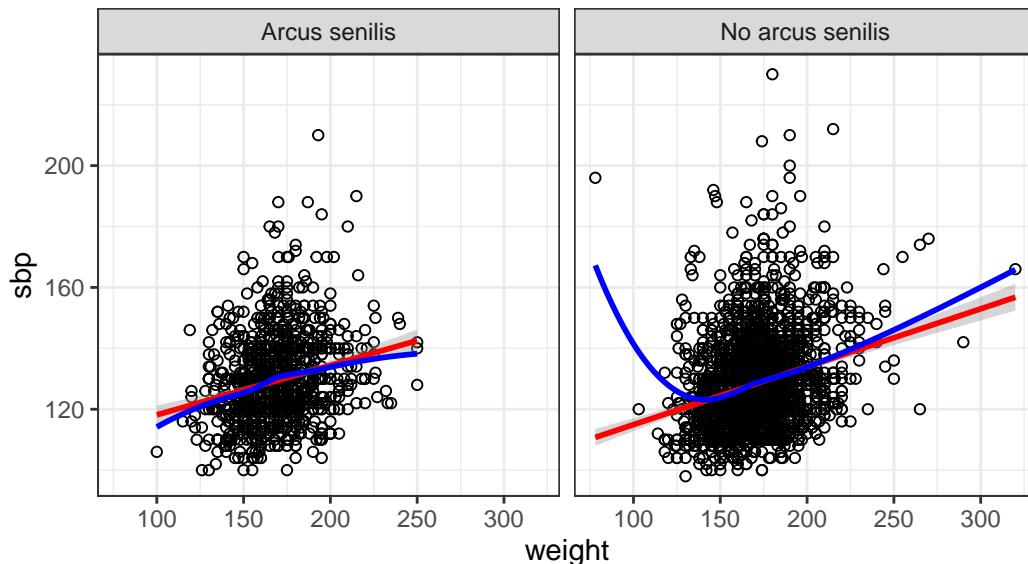
| arcus_f | 0 | 1 | NA_ |
|------------------|------|-----|-----|
| Arcus senilis | 0 | 941 | 0 |
| No arcus senilis | 2211 | 0 | 0 |
| <NA> | 0 | 0 | 2 |

Let's build a version of the `wcgs` data that eliminates all missing data in the variables of immediate interest, and then plot the SBP-weight relationship in groups of patients with and without arcus senilis.

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

```
facet_wrap(~ arcus_f)
```

SBP vs. Weight by Arcus Senilis status



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

```
model.noarcus <-  
  lm(sbp ~ weight, data = filter(wcgs, arcus == 0))  
  
tidy(model.noarcus) |> kable(digits = 2)
```

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

```
glance(model.noarcus) |> select(r.squared:p.value, AIC) |> kable(digits = 3)
```

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

```
summary(model.noarcus)
```

Call:

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

Residuals:

| Min | 1Q | Median | 3Q | Max |
|---------|---------|--------|-------|--------|
| -29.011 | -10.251 | -2.447 | 7.553 | 99.848 |

Coefficients:

| | Estimate | Std. Error | t value | Pr(> t) |
|-------------|----------|------------|---------|------------|
| (Intercept) | 95.9219 | 2.5552 | 37.54 | <2e-16 *** |
| weight | 0.1902 | 0.0149 | 12.77 | <2e-16 *** |

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 14.8 on 2209 degrees of freedom

Multiple R-squared: 0.0687, Adjusted R-squared: 0.06828

F-statistic: 163 on 1 and 2209 DF, p-value: < 2.2e-16

The linear model for the 2211 patients without Arcus Senilis has R-squared = 6.87%.

- The regression equation is 95.92 - 0.19 weight, for those patients without Arcus Senilis.

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

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

tidy(model.witharcus) |> kable(digits = 2)
```

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

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

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

```
summary(model.witharcus)
```

Call:

```
lm(formula = sbp ~ weight, data = filter(wcgs, arcus == 1))
```

Residuals:

| Min | 1Q | Median | 3Q | Max |
|---------|--------|--------|-------|--------|
| -30.335 | -9.636 | -1.961 | 7.973 | 76.738 |

Coefficients:

| | Estimate | Std. Error | t value | Pr(> t) |
|----------------|-----------|------------|---------|--------------|
| (Intercept) | 101.87847 | 3.75572 | 27.126 | < 2e-16 *** |
| weight | 0.16261 | 0.02201 | 7.388 | 3.29e-13 *** |
| --- | | | | |
| Signif. codes: | 0 | '***' | 0.001 | '**' |
| | 0.01 | '*' | 0.05 | '. ' |
| | 0.1 | ' ' | 1 | |

Residual standard error: 14.19 on 939 degrees of freedom

Multiple R-squared: 0.05494, Adjusted R-squared: 0.05393

F-statistic: 54.58 on 1 and 939 DF, p-value: 3.29e-13

The linear model for the 941 patients with Arcus Senilis has R-squared = 5.49%.

- The regression equation is 101.88 - 0.163 weight, for those patients with Arcus Senilis.

16.11 Including Arcus Status in the model

```
model3 <- lm(sbp ~ weight * arcus, data = filter(wcgs, !is.na(arcus)))  
tidy(model3) |> kable(digits = 2)
```

| term | estimate | std.error | statistic | p.value |
|--------------|----------|-----------|-----------|---------|
| (Intercept) | 95.92 | 2.52 | 38.00 | 0.00 |
| weight | 0.19 | 0.01 | 12.92 | 0.00 |
| arcus | 5.96 | 4.62 | 1.29 | 0.20 |
| weight:arcus | -0.03 | 0.03 | -1.02 | 0.31 |

```
glance(model3) |> select(r.squared:p.value, AIC) |> kable(digits = 3)
```

| r.squared | adj.r.squared | sigma | statistic | p.value | AIC |
|-----------|---------------|-------|-----------|---------|----------|
| 0.066 | 0.065 | 14.62 | 74.094 | 0 | 25860.96 |

```
summary(model3)
```

Call:

```
lm(formula = sbp ~ weight * arcus, data = filter(wcgs, !is.na(arcus)))
```

Residuals:

| Min | 1Q | Median | 3Q | Max |
|---------|---------|--------|-------|--------|
| -30.335 | -10.152 | -2.349 | 7.669 | 99.848 |

Coefficients:

| | Estimate | Std. Error | t value | Pr(> t) |
|--------------|----------|------------|---------|------------|
| (Intercept) | 95.92190 | 2.52440 | 37.998 | <2e-16 *** |
| weight | 0.19017 | 0.01472 | 12.921 | <2e-16 *** |
| arcus | 5.95657 | 4.61972 | 1.289 | 0.197 |
| weight:arcus | -0.02756 | 0.02703 | -1.019 | 0.308 |

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 14.62 on 3148 degrees of freedom

```
Multiple R-squared:  0.06595,   Adjusted R-squared:  0.06506
F-statistic: 74.09 on 3 and 3148 DF,  p-value: < 2.2e-16
```

The actual regression equation in this setting includes both weight, and an indicator variable (1 = yes, 0 = no) for arcus senilis status, and the product term combining weight and that 1/0 indicator. In 432, we'll spend substantial time and energy discussing these product terms, but we'll not do much of that in 431.

- Note the use of the product term `weight*arcus` in the setup of the model to allow both the slope of weight and the intercept term in the model to change depending on arcus senilis status.
 - For a patient who has arcus, the regression equation is $SBP = 95.92 + 0.19 \text{ weight} + 5.96 (1) - 0.028 \text{ weight (1)} = 101.88 + 0.162 \text{ weight}$.
 - For a patient without arcus senilis, the regression equation is $SBP = 95.92 + 0.19 \text{ weight} + 5.96 (0) - 0.028 \text{ weight (0)} = 95.92 + 0.19 \text{ weight}$.

The linear model including the interaction of weight and arcus to predict sbp for the 3152 patients with known Arcus Senilis status has $R^2 = 6.6\%$. Again, we'll discuss interaction more substantially in 432.

16.12 Predictions from these Linear Models

What is our predicted SBP for a subject weighing 175 pounds?

How does that change if our subject weighs 200 pounds?

Recall that

- *Without* Arcus Senilis, linear model for $SBP = 95.9 + 0.19 \times \text{weight}$
- *With* Arcus Senilis, linear model for $SBP = 101.9 + 0.16 \times \text{weight}$

So the predictions for a 175 pound subject are:

- $95.9 + 0.19 \times 175 = 129 \text{ mm Hg}$ without Arcus Senilis, and
- $101.9 + 0.16 \times 175 = 130 \text{ mm Hg}$ with Arcus Senilis.

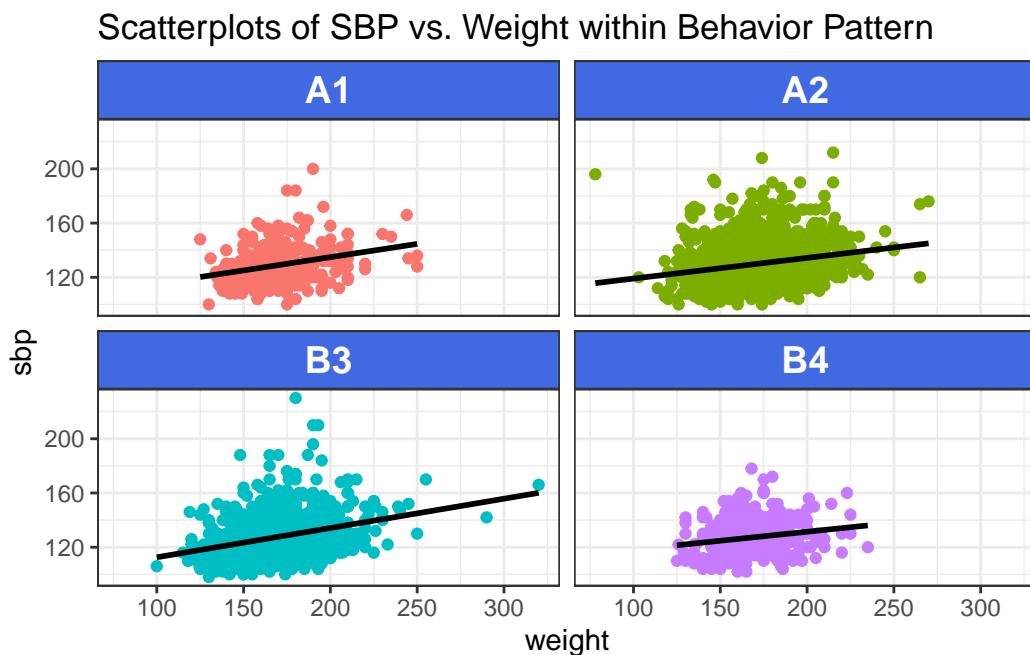
And thus, the predictions for a 200 pound subject are:

- $95.9 + 0.19 \times 200 = 134 \text{ mm Hg}$ without Arcus Senilis, and
- $101.9 + 0.16 \times 200 = 134.4 \text{ mm Hg}$ with Arcus Senilis.

16.13 Scatterplots with Facets Across a Categorical Variable

We can use facets in `ggplot2` to show scatterplots across the levels of a categorical variable, like `behpatt`.

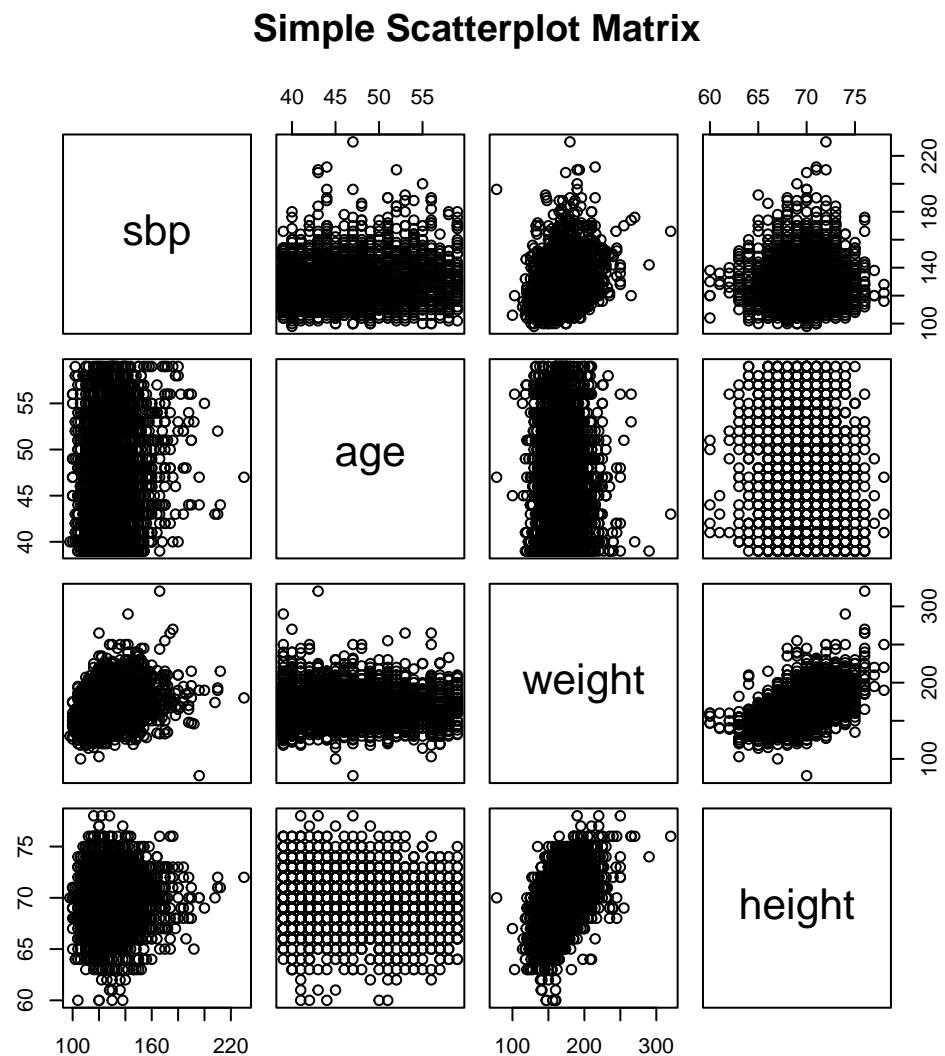
```
ggplot(wcgs, aes(x = weight, y = sbp, col = behpat)) +  
  geom_point() +  
  facet_wrap(~ behpat) +  
  geom_smooth(method = "lm", se = FALSE,  
              formula = y ~ x, col = "black") +  
  guides(color = "none") +  
  theme(strip.text = element_text(face="bold", size=rel(1.25), color="white"),  
        strip.background = element_rect(fill="royalblue")) +  
  labs(title = "Scatterplots of SBP vs. Weight within Behavior Pattern")
```



16.14 Scatterplot and Correlation Matrices

A **scatterplot matrix** can be very helpful in understanding relationships between multiple variables simultaneously. There are several ways to build such a thing, including the `pairs` function...

```
pairs (~ sbp + age + weight + height, data=wcgs, main="Simple Scatterplot Matrix")
```



16.14.1 Displaying a Correlation Matrix

```
wcgs |>  
  dplyr::select(sbp, age, weight, height) |>  
  cor() |>  
  kable(digits = 3)
```

| | sbp | age | weight | height |
|--------|-------|--------|--------|--------|
| sbp | 1.000 | 0.166 | 0.253 | 0.018 |
| age | 0.166 | 1.000 | -0.034 | -0.095 |
| weight | 0.253 | -0.034 | 1.000 | 0.533 |
| height | 0.018 | -0.095 | 0.533 | 1.000 |

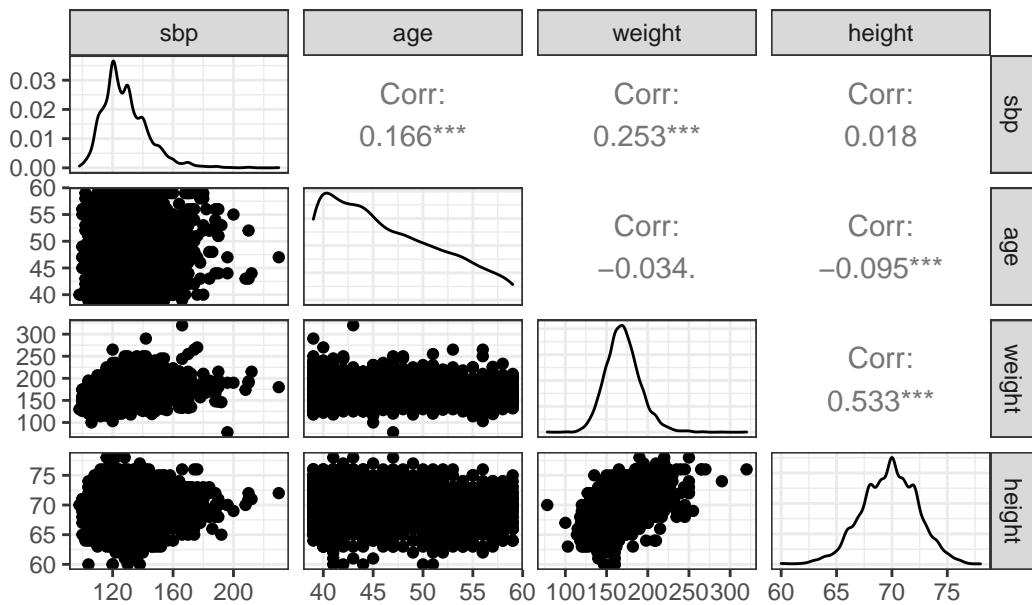
16.14.2 Using the GGally package

The `ggplot2` system doesn't have a built-in scatterplot system. There are some nice add-ins in the world, though. One option I sort of like is in the `GGally` package, which can produce both correlation matrices and scatterplot matrices.

The `ggpairs` function provides a density plot on each diagonal, Pearson correlations on the upper right and scatterplots on the lower left of the matrix.

```
GGally::ggpairs(wcgs |> select(sbp, age, weight, height),  
                 title = "Scatterplot Matrix via ggpairs")
```

Scatterplot Matrix via ggpairs



Part II

Part B. Comparing Summaries

17 Part B placeholder

The Course Notes for Part B will appear soon.

Part III

Part C. Building Models

18 Part C placeholder

The Course Notes for Part C will appear soon.

A Getting Data Into R

Using data from an R package

To use data from an R package, for instance, the `bechdel` data from the `fivethirtyeight` package, you can simply load the relevant package with `library` and then the data frame will be available

```
library(fivethirtyeight)
library(tidyverse)

bechdel

# A tibble: 1,794 x 15
  year imdb     title test  clean~1 binary budget domgr~2 intgr~3 code  budge~4
  <int> <chr>   <chr> <ord>   <chr>   <int>   <dbl> <dbl> <chr>   <int>
1  2013 tt1711~ 21 &~ nota~ notalk FAIL    1.3 e7  2.57e7  4.22e7 2013~  1.3 e7
2  2012 tt1343~ Dred~ ok-d~ ok      PASS    4.5 e7  1.34e7  4.09e7 2012~  4.57e7
3  2013 tt2024~ 12 Y~ nota~ notalk FAIL    2   e7  5.31e7  1.59e8 2013~  2   e7
4  2013 tt1272~ 2 Gu~ nota~ notalk FAIL    6.1 e7  7.56e7  1.32e8 2013~  6.1 e7
5  2013 tt0453~ 42 men   men     FAIL    4   e7  9.50e7  9.50e7 2013~  4   e7
6  2013 tt1335~ 47 R~ men   men     FAIL    2.25e8 3.84e7  1.46e8 2013~  2.25e8
7  2013 tt1606~ A Go~ nota~ notalk FAIL    9.2 e7  6.73e7  3.04e8 2013~  9.2 e7
8  2013 tt2194~ Abou~ ok-d~ ok      PASS    1.2 e7  1.53e7  8.73e7 2013~  1.2 e7
9  2013 tt1814~ Admi~ ok   ok      PASS    1.3 e7  1.80e7  1.80e7 2013~  1.3 e7
10 2013 tt1815~ Afte~ nota~ notalk FAIL    1.3 e8  6.05e7  2.44e8 2013~  1.3 e8
# ... with 1,784 more rows, 4 more variables: domgross_2013 <dbl>,
#   intgross_2013 <dbl>, period_code <int>, decade_code <int>, and abbreviated
#   variable names 1: clean_test, 2: domgross, 3: intgross, 4: budget_2013
# i Use `print(n = ...)` to see more rows, and `colnames()` to see all variable names
```

Using `read_rds` to read in an R data set

We have provided the `nnyfs.Rds` data file on the course data page.

Suppose you have downloaded this data file into a directory on your computer called `data` which is a sub-directory of the directory where you plan to do your work, perhaps called `431-nnyfs`.

Open RStudio and create a new project into the `431-nnyfs` directory on your computer. You should see a `data` subdirectory in the Files window in RStudio after the project is created.

Now, read in the `nnyfs.Rds` file to a new tibble in R called `nnyfs_new` with the following command:

```
nnyfs_new <- read_rds("data/nnyfs.Rds")
```

Here are the results...

```
nnyfs_new
```

```
# A tibble: 1,518 x 45
  SEQN sex    age_ch~1 race_~2 educ_~3 langu~4 sampl~5 incom~6 age_a~7 educ_~8
  <dbl> <fct>   <dbl> <fct>   <dbl> <fct>   <dbl> <dbl> <dbl> <dbl> <dbl>
  1 71917 Female     15 3_Blac~      9 English  28299.   0.21    46 2_9-11~
  2 71918 Female     8 3_Blac~      2 English  15127.    5       46 3_High~
  3 71919 Female     14 2_Whit~     8 English  29977.    5       42 5_Coll~
  4 71920 Female     15 2_Whit~     8 English  80652.   0.87    53 3_High~
  5 71921 Male       3 2_Whit~     NA English  55592.   4.34    31 3_High~
  6 71922 Male       12 1_Hisp~     6 English  27365.    5       42 4_Some~
  7 71923 Male       12 2_Whit~     5 English  86673.    5       39 2_9-11~
  8 71924 Female     8 4_Othe~     2 English  39549.   2.74    31 3_High~
  9 71925 Male       7 1_Hisp~     0 English  42333.   0.46    45 2_9-11~
 10 71926 Male      8 3_Blac~     2 English  15307.   1.57    56 3_High~
# ... with 1,508 more rows, 35 more variables: respondent <fct>,
#   salt_used <fct>, energy <dbl>, protein <dbl>, sugar <dbl>, fat <dbl>,
#   diet_yesterday <fct>, water <dbl>, plank_time <dbl>, height <dbl>,
#   weight <dbl>, bmi <dbl>, bmi_cat <fct>, arm_length <dbl>, waist <dbl>,
#   arm_circ <dbl>, calf_circ <dbl>, calf_skinfold <dbl>,
#   triceps_skinfold <dbl>, subscapular_skinfold <dbl>, active_days <dbl>,
#   tv_hours <dbl>, computer_hours <dbl>, physical_last_week <fct>, ...
# i Use `print(n = ...)` to see more rows, and `colnames()` to see all variable names
```

Using `read_csv` to read in a comma-separated version of a data file

We have provided the `nnyfs.csv` data file on the course data page.

Suppose you have downloaded this data file into a directory on your computer called `data` which is a sub-directory of the directory where you plan to do your work, perhaps called `431-nnyfs`.

Open RStudio and create a new project into the `431-nnyfs` directory on your computer. You should see a `data` subdirectory in the Files window in RStudio after the project is created.

Now, read in the `nnyfs.csv` file to a new tibble in R called `nnyfs_new2` with the following command:

```
nnyfs_new2 <- read_csv("data/nnyfs.csv")  
  
Rows: 1518 Columns: 45  
-- Column specification -----  
Delimiter: ","  
chr (18): sex, race_eth, language, educ_adult, respondent, salt_used, diet_y...  
dbl (27): SEQN, age_child, educ_child, sampling_wt, income_pov, age_adult, e...  
  
i Use `spec()` to retrieve the full column specification for this data.  
i Specify the column types or set `show_col_types = FALSE` to quiet this message.  
  
nnyfs_new2  
  
# A tibble: 1,518 x 45  
  SEQN sex    age_ch~1 race_~2 educ_~3 langu~4 sampl~5 incom~6 age_a~7 educ_~8  
  <dbl> <chr>   <dbl> <chr>    <dbl> <chr>    <dbl> <dbl>    <dbl> <chr>  
1 71917 Female     15 3_Blac~      9 English  28299.   0.21    46 2_9-11~  
2 71918 Female     8 3_Blac~      2 English  15127.    5       46 3_High~  
3 71919 Female     14 2_Whit~     8 English  29977.    5       42 5_Coll~  
4 71920 Female     15 2_Whit~     8 English  80652.   0.87    53 3_High~  
5 71921 Male       3 2_Whit~     NA English  55592.   4.34    31 3_High~  
6 71922 Male       12 1_Hisp~     6 English  27365.    5       42 4_Some~  
7 71923 Male       12 2_Whit~     5 English  86673.    5       39 2_9-11~  
8 71924 Female     8 4_Othe~     2 English  39549.   2.74    31 3_High~  
9 71925 Male       7 1_Hisp~     0 English  42333.   0.46    45 2_9-11~  
10 71926 Male      8 3_Blac~     2 English  15307.   1.57    56 3_High~  
# ... with 1,508 more rows, 35 more variables: 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_s Skinfold <dbl>,
```

```
# triceps_skinfold <dbl>, subscapular_skinfold <dbl>, active_days <dbl>,
# tv_hours <dbl>, computer_hours <dbl>, physical_last_week <chr>, ...
# i Use `print(n = ...)` to see more rows, and `colnames()` to see all variable names
```

If you also want to convert the `character` variables to `factors`, as you will often want to do before analyzing the results, you should instead use:

```
nnyfs_new3 <- read_csv("data/nnyfs.csv") %>%
  mutate(across(where(is.character), as_factor))
```

```
Rows: 1518 Columns: 45
-- Column specification -----
Delimiter: ","
chr (18): sex, race_eth, language, educ_adult, respondent, salt_used, diet_y...
dbl (27): SEQN, age_child, educ_child, sampling_wt, income_pov, age_adult, e...
i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
nnyfs_new3
```

```
# A tibble: 1,518 x 45
  SEQN sex    age_ch~1 race_~2 educ_~3 langu~4 sampl~5 incom~6 age_a~7 educ_~8
  <dbl> <fct>   <dbl> <fct>   <dbl> <fct>   <dbl> <dbl>   <dbl> <fct>
  1 71917 Female     15 3_Blac~      9 English  28299.   0.21    46 2_9-11~
  2 71918 Female     8 3_Blac~      2 English  15127.    5       46 3_High~
  3 71919 Female     14 2_Whit~     8 English  29977.    5       42 5_Coll~
  4 71920 Female     15 2_Whit~     8 English  80652.   0.87    53 3_High~
  5 71921 Male       3 2_Whit~     NA English  55592.   4.34    31 3_High~
  6 71922 Male       12 1_Hisp~     6 English  27365.    5       42 4_Some~
  7 71923 Male       12 2_Whit~     5 English  86673.    5       39 2_9-11~
  8 71924 Female     8 4_Othe~     2 English  39549.   2.74    31 3_High~
  9 71925 Male       7 1_Hisp~     0 English  42333.   0.46    45 2_9-11~
 10 71926 Male      8 3_Blac~     2 English  15307.   1.57    56 3_High~
# ... with 1,508 more rows, 35 more variables: respondent <fct>,
# salt_used <fct>, energy <dbl>, protein <dbl>, sugar <dbl>, fat <dbl>,
# diet_yesterday <fct>, water <dbl>, plank_time <dbl>, height <dbl>,
# weight <dbl>, bmi <dbl>, bmi_cat <fct>, arm_length <dbl>, waist <dbl>,
# arm_circ <dbl>, calf_circ <dbl>, calf_skinfold <dbl>,
# triceps_skinfold <dbl>, subscapular_skinfold <dbl>, active_days <dbl>,
```

```
#   tv_hours <dbl>, computer_hours <dbl>, physical_last_week <fct>, ...
# i Use `print(n = ...)` to see more rows, and `colnames()` to see all variable names
```

Note that, for example, `sex` and `race_eth` are now listed as factor (`fctr`) variables. One place where this distinction between `character` and `factor` variables matters is when you summarize the data.

```
summary(nnyfs_new2$race_eth)
```

| Length | Class | Mode |
|--------|-----------|-----------|
| 1518 | character | character |

```
summary(nnyfs_new3$race_eth)
```

| | | |
|------------------------|----------------------|------------|
| 3_Black Non-Hispanic | 2_White Non-Hispanic | 1_Hispanic |
| 338 | 610 | 450 |
| 4_Other Race/Ethnicity | | |
| 120 | | |

Converting Character Variables into Factors

The command you want to create `newdata` from `olddata` is:

```
newdata <- olddata %>%
  mutate(across(where(is.character), as_factor))
```

For more on factors, visit <https://r4ds.had.co.nz/factors.html>

Converting Data Frames to Tibbles

Use `as_tibble()` or simply `tibble()` to assign the attributes of a tibble to a data frame. Note that `read_rds` and `read_csv` automatically create tibbles.

For more on tibbles, visit <https://r4ds.had.co.nz/tibbles.html>.

For more advice

Consider visiting the software tutorials page under the R and Data heading on our main web site.

B References

- Bock, David E., Paul F. Velleman, and Richard D. De Veaux. 2004. *Stats: Modelling the World*. Boston MA: Pearson Addison-Wesley.
- Gelman, Andrew, and Jennifer Hill. 2007. *Data Analysis Using Regression and Multilevel-Hierarchical Models*. New York: Cambridge University Press. <http://www.stat.columbia.edu/~gelman/arm/>.
- Gelman, Andrew, and Deborah Nolan. 2017. *Teaching Statistics: A Bag of Tricks*. Second Edition. Oxford, UK: Oxford University Press.
- Ismay, Chester, and Albert Y. Kim. 2022. *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.
- Ramsey, Fred L., and Daniel W. Schafer. 2002. *The Statistical Sleuth: A Course in Methods of Data Analysis*. Second. Pacific Grove, CA: Duxbury.
- Vittinghoff, Eric, David V. Glidden, Stephen C. Shiboski, and Charles E. McCulloch. 2012. *Regression Methods in Biostatistics: Linear, Logistic, Survival, and Repeated Measures Models*. Second. Springer-Verlag, Inc. <http://www.biostat.ucsf.edu/vgsm/>.
- Wainer, Howard. 1997. *Visual Revelations: Graphical Tales of Fate and Deception from Napoleon Bonaparte to Ross Perot*. New York: Springer-Verlag.
- . 2005. *Graphic Discovery: A Trout in the Milk and Other Visual Adventures*. Princeton, NJ: Princeton University Press.
- . 2013. *Medical Illuminations: Using Evidence, Visualization and Statistical Thinking to Improve Healthcare*. New York: Oxford University Press.
- Wickham, Hadley, and Garrett Grolemund. 2022. *R for Data Science*. Second. O'Reilly. <https://r4ds.hadley.nz/>.
- Yamada, SB, and EG Boulding. 1998. "Claw Morphology, Prey Size Selection and Foraging Efficiency in Generalist and Specialist Shell-Breaking Crabs." *Journal of Experimental Marine Biology and Ecology* 220: 191–211. http://www.science.oregonstate.edu/~yamadas/SylviaCV/BehrensYamada_Boulding1998.pdf.