

Statistics for Data Science Notes

Andrew Sage - Stat 255: Lawrence University

2021-09-12

Contents

1 Exploratory Data Analysis	5
1.1 Exploring Diamond Prices	5
1.2 Tidy Data	8
1.3 Models with a Quantitative Explanatory Variable	23
1.4 Multiple Regression Model	29
1.5 Least-Squares Estimation	35
1.6 ANalysis Of VAriance	38
1.7 Interaction Models with Categorical Variables	46
1.8 Interaction Models with Categorical and Quantitative Variables	57
1.9 Interaction Models with Quantitative Variables	66
2 Interval Estimation via Simulation	75
2.1 Sampling Variability and Margin of Error	75
2.2 Bootstrap Sampling for Election Example	85
2.3 Bootstrapping Distribution of Sample Mean	92
2.4 Bootstrapping Distribution for Statistics other than Mean	107
2.5 Bootstrap Distribution for Difference in Sample Means	113
2.6 Bootstrap Distribution for Simple Linear Regression Coefficients	118
2.7 Bootstrap Distribution for Multiple Regression Coefficients	123
3 Simulation-Based Hypothesis Tests	131
3.1 Introduction to Hypothesis Testing via Simulation	131
3.2 Simulation-Based Hypothesis Test for Regression Coefficients	142
3.3 Simulation-Based Hypothesis Test for F-Statistics	150
3.4 “Theory-Based” Tests and Intervals in R	166
3.5 Responsible Use of Hypothesis Testing and p-values	178
4 The Normal Error Linear Regression Model	187
4.1 Standard Error Formulas and Theory-Based Intervals	187
4.2 The Normal Error Regression Model	196
4.3 Tests and Intervals in Normal Error Regression Model	209
4.4 F-Distribution	212
4.5 Intervals for Predicted Values	217
4.6 Regression Model Assumptions	228

4.7	Transformations on Response Variable	242
4.8	The Regression Effect	248
5	Logistic Regression	253
5.1	Logistic Regression	253
5.2	Interpretations in a Logistic Regression Model	258
5.3	Multiple Logistic Regression	262
6	Building Models for Interpretation	267
6.1	Introduction to Model Building	267
6.2	Transformations of Explanatory Variables	279
6.3	Modeling SAT scores: Simpson's Paradox and Quadratic Terms .	283
6.4	Polynomial Regression	292
6.5	Adjusted R ² , AIC, and BIC	302
7	Building Models for Prediction	307
7.1	Training and Test Error	307
7.2	Variance-Bias Tradeoff	319
7.3	Cross-Validation	321
7.4	Variable Engineering	323
7.5	Cross-Validation with caret	333
8	Advanced Regression and Nonparametric Approaches	341
8.1	Penalized Regression: Ridge and Lasso	341
8.2	Decision Trees	351
8.3	Regression Splines	360

Chapter 1

Exploratory Data Analysis

1.1 Exploring Diamond Prices

We consider a dataset with prices (in \$ US) and other information on 53,940 round cut diamonds. The first 6 rows are shown below.

```
library(tidyverse)
data(diamonds)
head(diamonds)

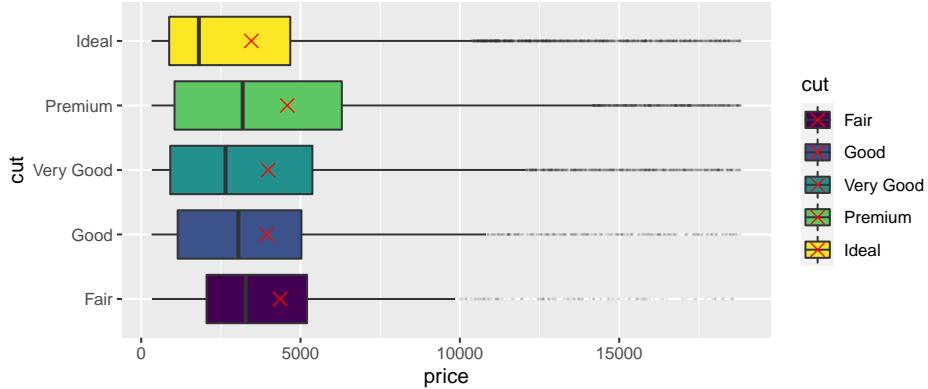
##   carat      cut color clarity depth table price     x     y     z
## 1 0.23    Ideal    E    SI2  61.5    55   326 3.95 3.98 2.43
## 2 0.21  Premium    E    SI1  59.8    61   326 3.89 3.84 2.31
## 3 0.23      Good    E    VS1  56.9    65   327 4.05 4.07 2.31
## 4 0.29  Premium    I    VS2  62.4    58   334 4.20 4.23 2.63
## 5 0.31      Good    J    SI2  63.3    58   335 4.34 4.35 2.75
## 6 0.24  Very Good    J   VVS2  62.8    57   336 3.94 3.96 2.48
```

The dataset includes both:

- **categorical** (or factor) variables cut, color, clarity, and
- **quantitative** (or numeric) variables, depth, table, price, x, y, z

1.1.1 Boxplot of Diamond Prices

```
ggplot(data=diamonds, aes(x=price, y=cut, fill=cut)) +
  geom_boxplot(outlier.size=0.01, outlier.alpha = 0.1) +
  stat_summary(fun=mean, geom="point", shape=4, color="red", size=3)
```

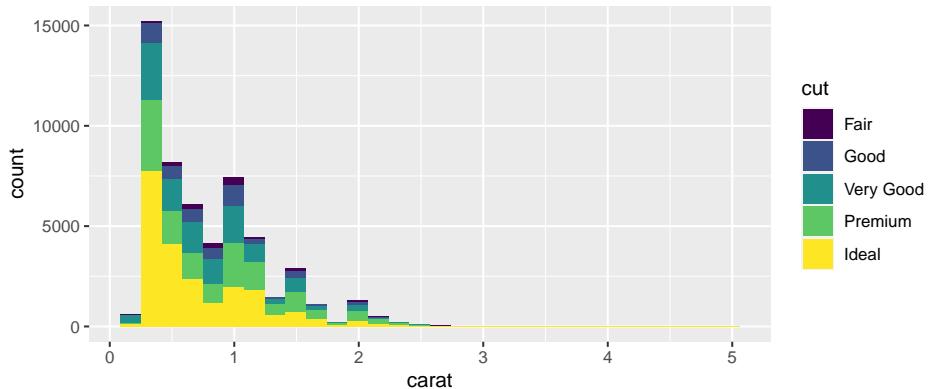


What do we notice about the relationship between price and cut? Is this surprising?

1.1.2 Histogram of carat size and quality of cut

Next, we examine a histogram, displaying price, cut, and carat size.

```
ggplot(data=diamonds, aes(x=carat, fill=cut)) + geom_histogram()
```



How does the information in this plot help explain the surprising result we saw in the boxplot?

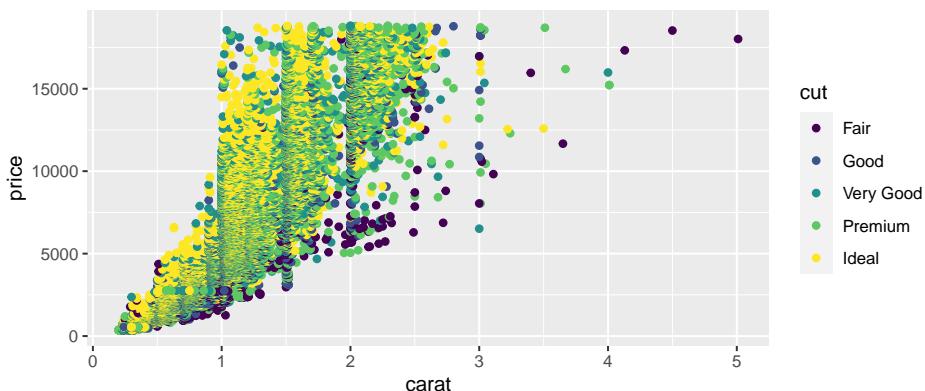
1.1.3 Table 1: Average carat size and price by quality of cut

```
## # A tibble: 5 x 4
##   cut      N Avg_carat Avg_price
##   <ord>    <int>     <dbl>      <dbl>
## 1 Fair     1610     1.05      4359.
## 2 Good     4906     0.849     3929.
## 3 Very Good 12082    0.806     3982.
## 4 Premium   13791    0.892     4584.
## 5 Ideal    21551    0.703     3458.
```

1.1.4 Scatterplot of carat size and quality of cut

Next, we use a scatterplot to visualize cut, price, and carat size.

```
ggplot(data=diamonds, aes(x=carat, y=price, color=cut)) + geom_point()
```



What should we conclude about the relationship between price and quality of cut? Are better cuts generally more expensive? less expensive? about the same? Does the relationship between price and cut seem to depend on carat size?

1.1.5 Terminology

The diamonds dataset is an example of two statistical concepts:

Simpson's Paradox refers to a situation when an apparent relationship between two variables changes or reverses when additional variable(s) are considered.

Example: diamonds with higher quality of cut appear less expensive than lower quality cuts, until we account for carat size

An **interaction** between two variables X and Y occurs when the relationship between X and a third variable Z depends on Y.

Example: the relationship between cut and price depends on carat size, so there is an interaction between cut and carat size.

1.2 Tidy Data

1.2.1 Representations of Data

Data can be displayed in many different tabular forms. We'll discuss one useful form, called **tidy** data.

Learning Outcomes:

1. Define tidy data.
2. Recognize when data are tidy form.

Consider the following representations of the same dataset, which displays the number of tuberculosis cases in different countries, relative to population. This example comes from R for Data Science by Wickham and Grolemund

1.2.2 Representation 1

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

1.2.3 Representation 2

country	year	type	count
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

1.2.4 Representation 3

country	year	rate
Afghanistan	1999	745/19987071
Afghanistan	2000	2666/20595360
Brazil	1999	37737/172006362
Brazil	2000	80488/174504898
China	1999	212258/1272915272
China	2000	213766/1280428583

1.2.5 Representation 4

Table A:

country	1999	2000
Afghanistan	745	2666
Brazil	37737	80488
China	212258	213766

Table B:

```
kable(table4b)
```

country	1999	2000
Afghanistan	19987071	20595360
Brazil	172006362	174504898
China	1272915272	1280428583

1.2.6 Variables and Observations

In this example, we have observed data on various countries at different points in time. The record for a single country, in a given year is called an **observation**.

For each observation, we record the country, year, number of cases, and population. These are called **variables**.

1.2.7 Tidy Data

A dataset is said to be **tidy** when it satisfies the following conditions:

1. Each variable has its own column.
2. Each observation has its own row.
3. Each value must has own cell.

In fact, any two of these imply the third.

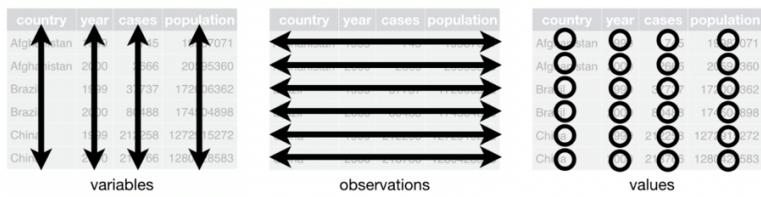


Image from <https://r4ds.had.co.nz/tidy-data.html>

Image from R for Data Science by Wickham and Grolemund

1.2.8 Representation 1 Tidy

Representation 1 is in tidy form.

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

1.2.9 Representation 2 not Tidy

Representation 2 is not in tidy form.

- Observations are spread over multiple rows.
- The variables `cases` and `population` do not have their own column.
- The column `type` contains variable names, not values.

country	year	type	count
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

1.2.10 Representation 3 not Tidy

Representation 3 is not in tidy form.

The variables `cases` and `population` do not have their own columns, but are combined in a single column called `rate`.

country	year	rate
Afghanistan	1999	745/19987071
Afghanistan	2000	2666/20595360
Brazil	1999	37737/172006362
Brazil	2000	80488/174504898
China	1999	212258/1272915272
China	2000	213766/1280428583

1.2.11 Representation 4 is not Tidy

Representation 4 is not in tidy form.

- The variable `year` is spread across multiple columns.
- The variables `cases` and `population` are spread over multiple tables.

country	1999	2000
Afghanistan	745	2666
Brazil	37737	80488
China	212258	213766

country	1999	2000
Afghanistan	19987071	20595360
Brazil	172006362	174504898
China	1272915272	1280428583

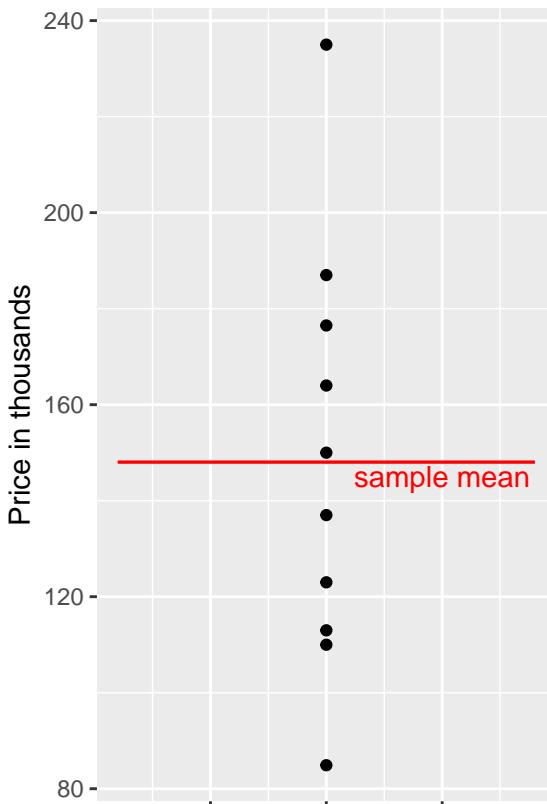
1.2.12 Why Use Tidy Data

- Data are often easiest to work with when they are in tidy form
- The `tidyverse()` R package is useful for creating graphs, and calculating summary statistics when data are in tidy form.
- Sometimes there is good reason for data to not be in tidy form. This is ok, but it makes it harder to work with.
- In this class, we will focus on data that are already in tidy form. However, if you come across data on your own, you should check that it is tidy before attempting to use the techniques we'll see in this class.
- In **CMSC/STAT 205: Data-Scientific Programming**, we study how to convert data into tidy form if it is not already. More information can be found in R For Data Science by Wickham and Grolemund.

1.2.13 Predicting Price of New House

Shown below are the prices of 10 houses sold in Ames, IA between 2006 and 2010.

```
## [1] 123.00 187.00 176.50 113.00 163.99 110.00 84.90 150.00 235.00 137.00
```



- Suppose we want to predict the price of a new house, that was sold in this time period. (note that new means a house not among the original 10, rather than a newly-built house.)
- Using the information available the best we can do is to use the sample mean for our prediction.

```
mean(Houses$SalePrice)
```

```
## [1] 148.039
```

1.2.14 A Very Simple Model

- If we have the prices of n houses, y_1, y_2, \dots, y_n , and want to predict the price of a new house, we use the model:

$$\widehat{\text{Price}} = \bar{y}, \text{ where } \bar{y} = \frac{\sum_{i=1}^n y_i}{n}$$

- The symbol $\widehat{\text{Price}}$, represents the predicted, or expected, price.

1.2.15 Simple Model in R

```
M0 <- lm(data=Houses, SalePrice~1)
summary(M0)

##
## Call:
## lm(formula = SalePrice ~ 1, data = Houses)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -63.139 -32.539  -4.539  25.334  86.961
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 148.04     13.97    10.6 0.0000022 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 44.17 on 9 degrees of freedom
```

1.2.16 Quantifying Total Variability in Prices

- Although the mean price represents our best prediction, we certainly shouldn't expect the price of the new house to exactly match the mean of the original 10. There is considerable variability between house prices.
- We can get a sense of how much variability we should expect in our prediction by looking at how much the values in our dataset differ from the predicted (mean) price.

- $\sum_{i=1}^n (y_i - \bar{y}) = 0$, so this is not a helpful measure.

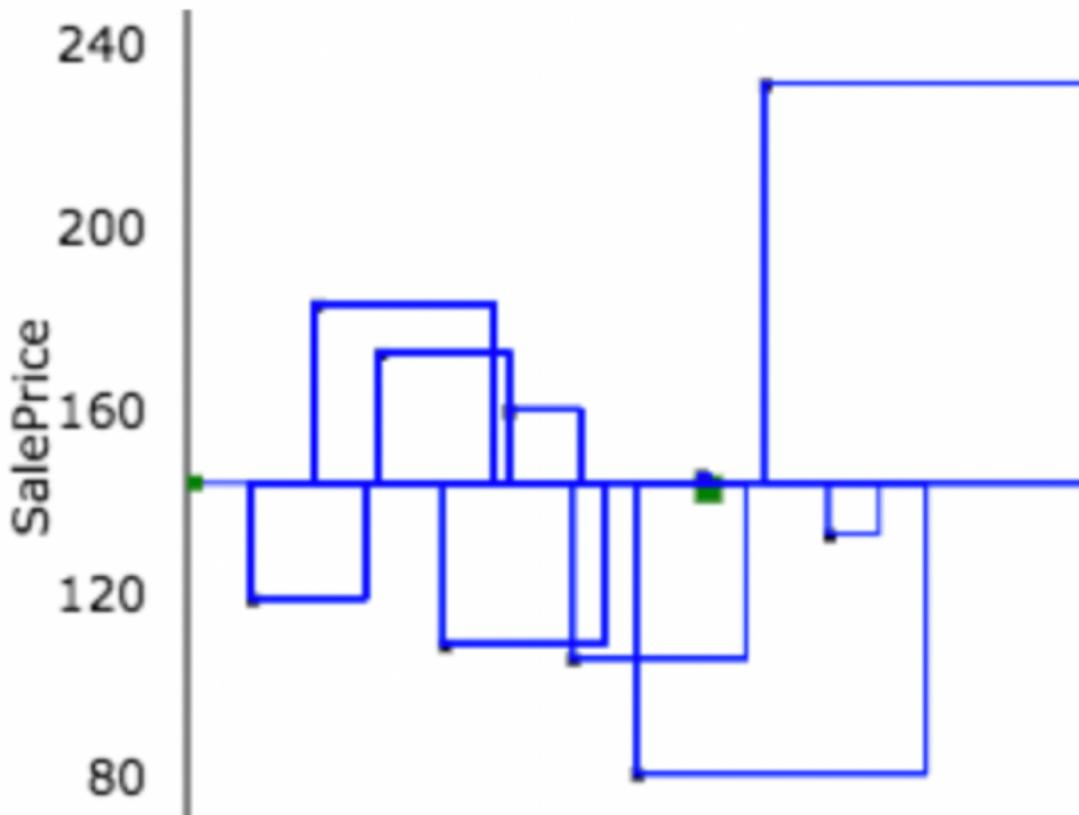


1.2.17 Total Sum of Squares

We'll measure total variability in price, using :

$$\sum_{i=1}^n (y_i - \bar{y})^2$$

We'll call this the total sum of squares, and abbreviate it SST.



Created at <http://www.rosmarchance.com/applets/RegShuff.html>

1.2.18 Total Variation Calculations in Simple Model

- Calculate $\sum_{i=1}^n (y_i - \bar{y})$ in R.

```
sum(Houses$SalePrice - mean(Houses$SalePrice))
```

```
## [1] 0.0000000000001421085
```

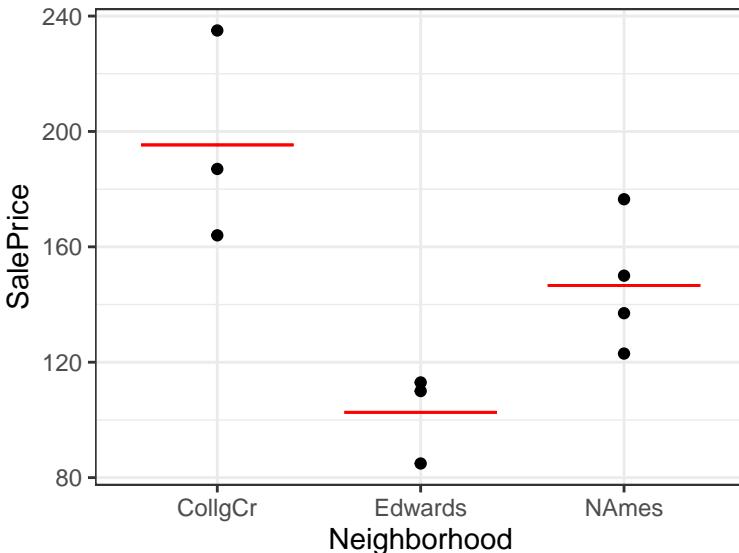
- Calculate $SST = \sum_{i=1}^n (y_i - \bar{y})^2$ in R.

```
sum((Houses$SalePrice - mean(Houses$SalePrice))^2)
```

```
## [1] 17558.52
```

1.2.19 Adding Information about Neighborhood

- Now, suppose we know the neighborhood of each house. We can use this information to improve our predictions.
- The prediction for a new house is given by the average price in that neighborhood.



Neighborhood	AveragePrice
CollgCr	195.3300
Edwards	102.6333
NAmes	146.6250

1.2.20 Explanatory and Response Variables

- The variable we are trying to predict (price) is called the **response variable** (denoted Y).
- The variable(s) we use to help us make the prediction (neighborhood) is(are) called **explanatory variables** (denoted X). These are also referred to as **predictor variables** or **covariates**.

1.2.21 Model by Neighborhood

```
M_Nbhd <- lm(data=Houses, SalePrice ~ Neighborhood)
summary(M_Nbhd)
```

```

## 
## Call:
## lm(formula = SalePrice ~ Neighborhood, data = Houses)
## 
## Residuals:
##    Min      1Q  Median      3Q     Max 
## -31.340 -15.706 -2.477  9.617 39.670 
## 
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)             195.33     14.89   13.118 0.00000349 ***  
## NeighborhoodEdwards     -92.70     21.06   -4.402  0.00315 **   
## NeighborhoodNAmes       -48.71     19.70   -2.473  0.04267 *    
## ---                        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 25.79 on 7 degrees of freedom
## Multiple R-squared:  0.7348, Adjusted R-squared:  0.6591 
## F-statistic: 9.699 on 2 and 7 DF,  p-value: 0.009603

```

1.2.22 R Output for Model with Categorical Variables

```

##                               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)             195.33000 14.89037 13.117871 0.000003490079  
## NeighborhoodEdwards     -92.69667 21.05817 -4.401934 0.003149456985  
## NeighborhoodNAmes       -48.70500 19.69811 -2.472572 0.042671888138  

```

- For categorical explanatory variables, R treats the category that comes first alphabetically (in this case CCreek), as a baseline. The intercept gives the prediction for this category.
 - We would expect a house in College Creek to cost 195.33 thousand dollars.
- Each of the other rows in the coefficients table represent the difference between the expected response (price) for that category (neighborhood), compared to the baseline.
 - We would expect a house in Edwards to cost 92.70 thousand less than a house in College Creek. (hence costing 102.63 thousand)
 - We would expect a house in North Ames to cost 48.71 thousand less than a house in College Creek. (hence costing 146.62 thousand)

1.2.23 Model Notation for Houses by Neighborhood

In the model can be expressed in the form:

$$\widehat{\text{Price}} = b_0 + b_1 \times I_{\text{Edwards}} + b_2 \times I_{\text{NAmes}}$$

$$\widehat{\text{Price}} = 195.33 + -92.7 \times I_{\text{Edwards}} + -48.71 \times I_{\text{NAmes}}, \text{ where}$$

I represents an indicator variables, taking on values 0 or 1.

- Example:

$$I_{\text{Edwards}} = \begin{cases} 1 & \text{if house is in Edwards Neighborhood} \\ 0 & \text{otherwise} \end{cases}$$

Predicted Prices:

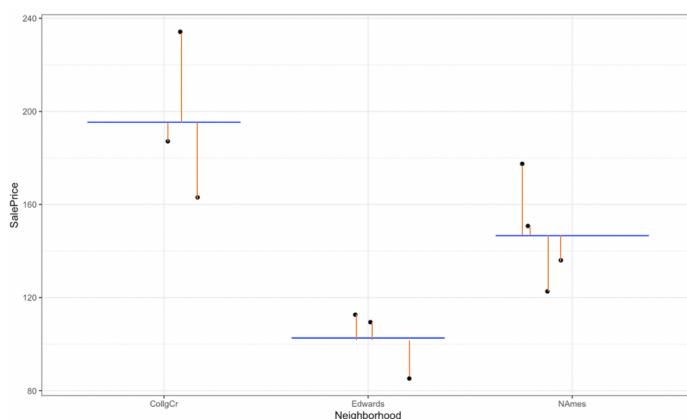
College Creek: $\widehat{\text{Price}} = 195.33 + -92.7 \times 0 + -48.71 \times 0 = 195.33$ thousand.

Edwards: $\widehat{\text{Price}} = 195.33 + -92.7 \times 1 + -48.71 \times 0 = 102.63$ thousand.

North Ames: $\widehat{\text{Price}} = 195.33 + -92.7 \times 0 + -48.71 \times 1 = 146.62$ thousand.

1.2.24 Residuals for Neighborhood Model

- The difference between the true and predicted values ($y_i - \hat{y}_i$) is called the *i*th residual.



1.2.25 Residuals for Neighborhood Model (cont.)

	SalePrice	Predicted	Residual	ResidualSq
## 1	123.00	146.6250	-23.625000	558.14063
## 2	187.00	195.3300	-8.330000	69.38890
## 3	176.50	146.6250	29.875000	892.51563
## 4	113.00	102.6333	10.366667	107.46778
## 5	163.99	195.3300	-31.340000	982.19560
## 6	110.00	102.6333	7.366667	54.26778

```

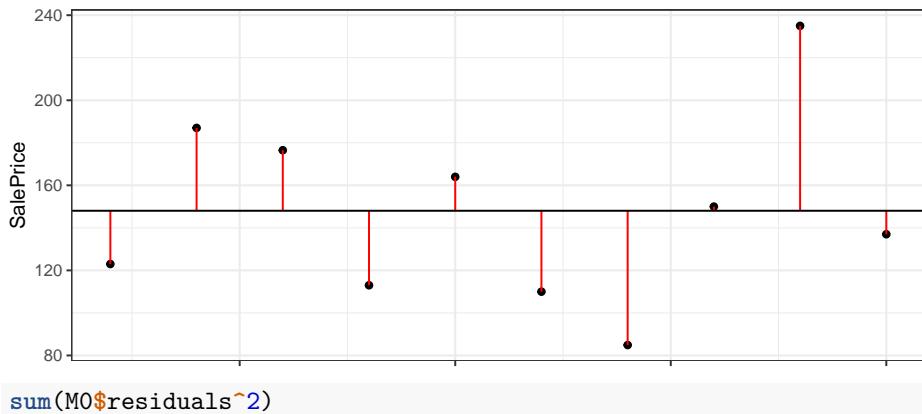
## 7      84.90 102.6333 -17.733333 314.47111
## 8     150.00 146.6250   3.375000 11.39063
## 9     235.00 195.3300  39.670000 1573.70890
## 10    137.00 146.6250  -9.625000  92.64062
sum(M_Nbhd$residuals^2)

## [1] 4656.188

```

1.2.26 Variability Explained by Model with Neighborhood

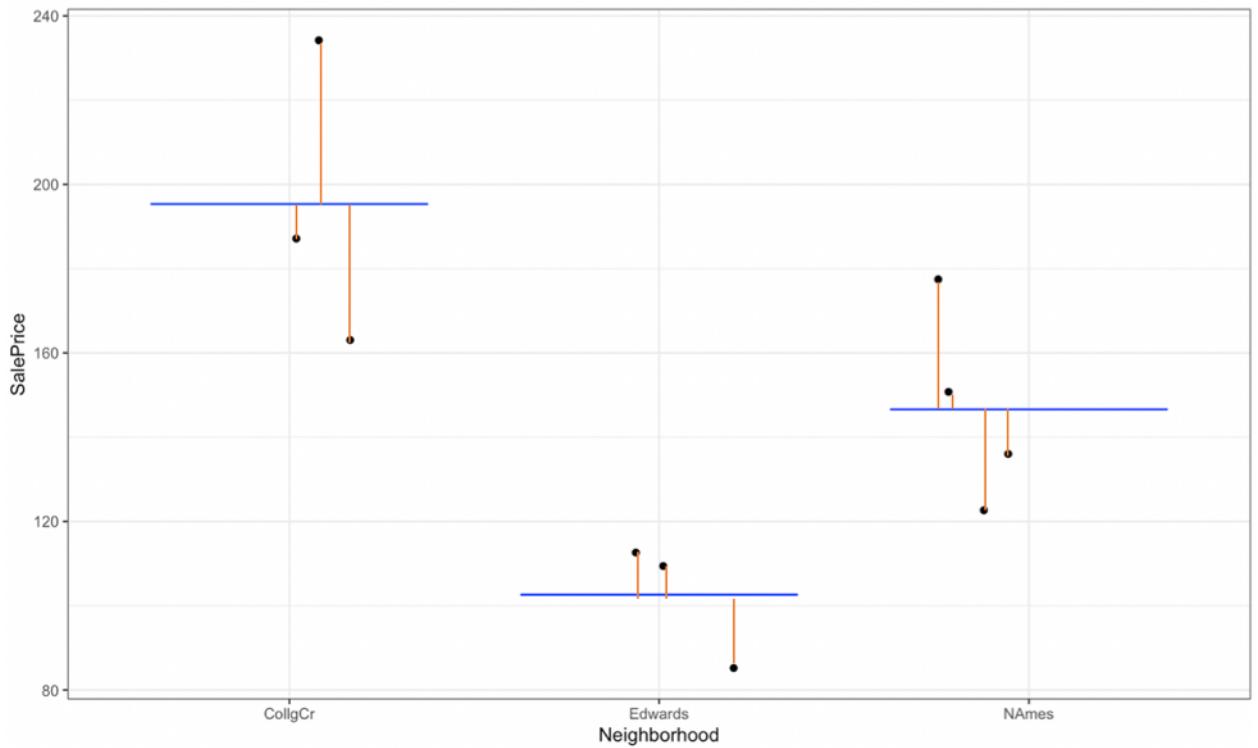
Intercept-Only Model:



```
sum(M0$residuals^2)
```

```
## [1] 17558.52
```

Model Using Neighborhood



```
sum(M_Nbhd$residuals^2)
```

```
## [1] 4656.188
```

1.2.27 Quantifying Variability Explained

- the total variability in house prices is the sum of the squared differences between price and average price.

$$\text{Total Variability in Price} = \text{SST} = \sum_{i=1}^n (y_i - \bar{y})^2$$

- the variability remaining unexplained even after accounting for neighborhood is given by the sum of squared residuals. We abbreviate this SSR, for sum of squared residuals.

$$\text{SSR} = \text{Variability Remaining} = \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

- the variability explained by the model, abbreviated SSM, is given by

$$SSM = SST - SSR$$

It can be shown that $SSM = \sum_{i=1}^n (\hat{y}_i - \bar{y})^2$. These abbreviations here vary across texts. Be careful!

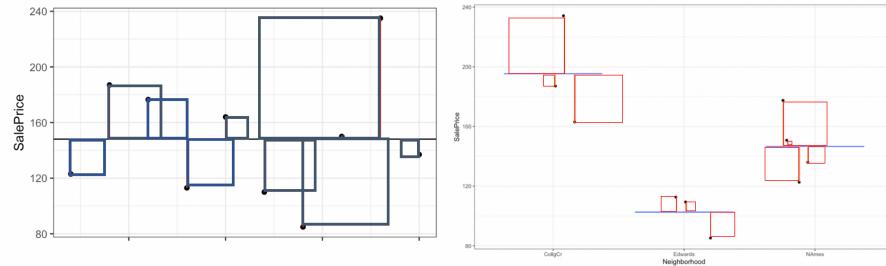
1.2.28 Coefficient of Determination

- The **coefficient of determination (abbreviated R^2)** is defined as

$$R^2 = \frac{\text{Variability Explained by Model}}{\text{Total Variability}} = \frac{SSM}{SST} = \frac{\sum_{i=1}^n (\hat{y}_i - \bar{y})^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

- R^2 can be interpreted as the proportion of total variability in the response variable that is explained by the model using the given explanatory variable(s).

1.2.29 R^2 Visually



- Blue Area = Total Variability (SST)
- Red Area = Variability Remaining Unexplained by Model (SSR)
- Blue Area - Red Area = Variability Explained by Model (SSM)
- $R^2 = \frac{\text{Area of Blue Squares} - \text{Area of Red Squares}}{\text{Area of Blue Squares}} = \frac{SSM}{SST} = \frac{SSM}{SST}$

1.2.30 Variation Explained by Neighborhood Model

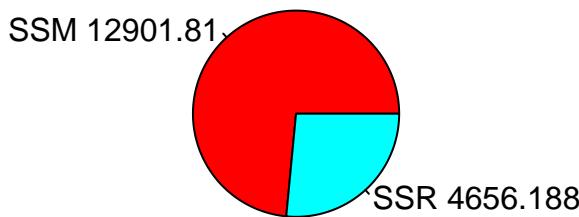
- Total variability in house prices $SST = 17,558.52$

- Variability remaining unexplained after model accounting for neighborhood: SSR = 4,656.188
- Variability explained by model: SSM=SST-SSR=17,558.52 - 4,656.188 = 12,901.81

$$R^2 = \frac{12,901.81}{17,558.52} \approx 0.7348$$

- 73.5% of the variation in house price is explained by the model using neighborhood as an explanatory variable.
- This matches the value of “Multiple R-squared” in the 2nd last line of the R model summary.

Total Variability in House Prices (SST)

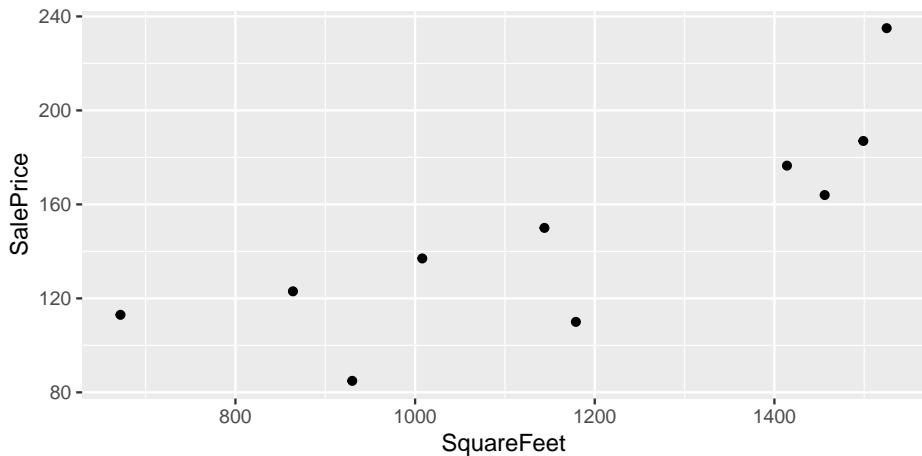


1.3 Models with a Quantitative Explanatory Variable

1.3.1 Model using SquareFeet in House

Now, suppose we do not know the neighborhood, but do know the size of the house in square feet.

```
ggplot(data=Houses, aes(x=SquareFeet, y=SalePrice)) + geom_point()
```



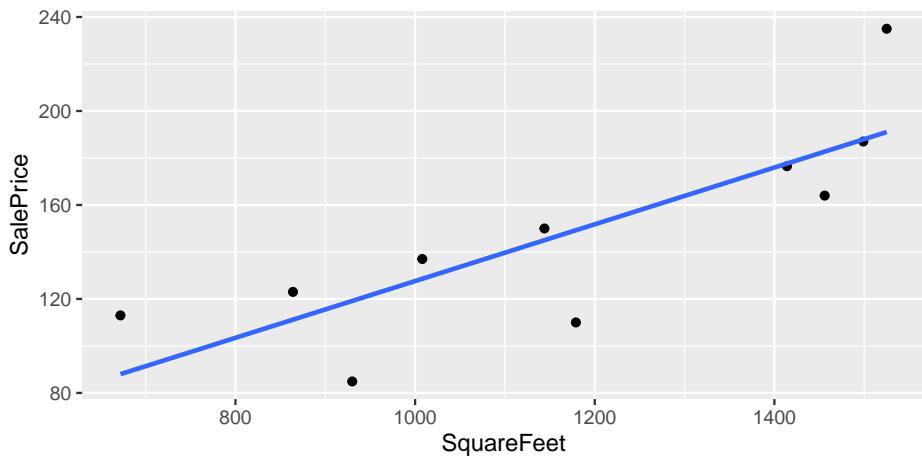
Suppose we want to predict the price of houses with the following number of square feet:

- 990 square feet
- 1235 square feet
- 1476 square feet

1.3.2 Model on SquareFeet in House

Since square feet is a quantitative variable, we can make predictions by fitting a line to the data.

```
ggplot(data=Houses, aes(x=SquareFeet, y=SalePrice)) + geom_point() +
  stat_smooth(method="lm", se=FALSE)
```



- For a house with 990 square feet, predicted price is about \$125 thousand.
- For a house with 1235 square feet, predicted price is about \$155 thousand.
- For a house with 1476 square feet, predicted price is about \$185 thousand.

1.3.3 Model using Square Feet

```
M_SqFt <- lm(data=Houses, SalePrice~SquareFeet)
summary(M_SqFt)

##
## Call:
## lm(formula = SalePrice ~ SquareFeet, data = Houses)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -39.235 -14.309   2.052  10.966  43.971
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.82000  36.35455  0.188  0.85586
## SquareFeet    0.12079   0.03022   3.997  0.00397 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 27.06 on 8 degrees of freedom
## Multiple R-squared:  0.6663, Adjusted R-squared:  0.6246
## F-statistic: 15.97 on 1 and 8 DF,  p-value: 0.003967
```

1.3.4 Model for SquareFeet and Interpretations

In the model using both square feet and neighborhood, the regression equation is

$$\widehat{\text{Price}} = b_0 + b_1 \times \text{SquareFeet}$$

$$\widehat{\text{Price}} = 6.82 + 0.121 \times \text{SquareFeet}$$

- $\widehat{\text{Price}}$ represents the expected, or predicted, price.
- The slope, b_1 represents the expected change in price (in thousands) per one-unit increase in square feet.
 - The price of a house is expected to increase by 121 dollars for each additional square foot.

- The intercept, b_0 represents the expected price of a house with 0 square feet.
 - In this situation, this is not a meaningful interpretation.

1.3.5 Calculating Predicted Prices

$$\widehat{\text{Price}} = 6.82 + 0.121 \times \text{SquareFeet}$$

- Predicted price for a house with 990 square feet:

$$\widehat{\text{Price}} = 6.82 + 0.121 \times 990 = 126.4 \text{ thousand dollars.}$$

- Predicted price for a house with 1235 square feet:

$$\widehat{\text{Price}} = 6.82 + 0.121 \times 1235 = 156.0 \text{ thousand dollars.}$$

- Predicted price for a house with 1476 square feet:

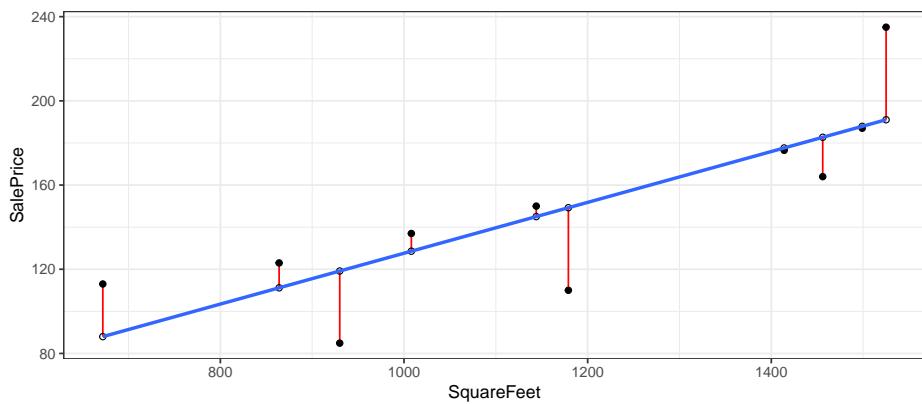
$$\widehat{\text{Price}} = 6.82 + 0.121 \times 1476 = 185.1 \text{ thousand dollars.}$$

1.3.6 Residuals for SquareFeet Model

The difference between the actual and predicted price is called the **residual**.

```
M_SqFt$residuals
```

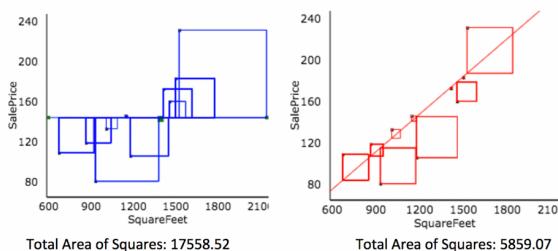
```
##           1          2          3          4          5          6
##  11.8149181 -0.8885824 -1.1211847 25.0071576 -18.7044870 -39.2348498
##           7          8          9         10
## -34.2574142   4.9929021  43.9708019   8.4207385
```



1.3.7 Residuals for SquareFeet Model (cont.)

```
##   SalePrice Predicted   Residual  ResidualSq
## 1    123.00  111.18508  11.8149181 139.5922893
## 2    187.00  187.88858  -0.8885824  0.7895786
## 3    176.50  177.62118  -1.1211847  1.2570550
## 4    113.00  87.99284  25.0071576  625.3579304
## 5    163.99  182.69449  -18.7044870 349.8578356
## 6    110.00  149.23485  -39.2348498 1539.3734427
## 7     84.90  119.15741  -34.2574142 1173.5704309
## 8    150.00  145.00710   4.9929021  24.9290718
## 9    235.00  191.02920  43.9708019 1933.4314183
## 10   137.00  128.57926   8.4207385  70.9088361
sum(M_SqFt$residuals^2)
## [1] 5859.068
```

1.3.8 Variation Explained by SquareFeet Model



Created at <http://www.rossmanchance.com/applets/RegShuffle.htm>.

- Blue Area = Total Variability (SST)
- Red Area = Variability Remaining Unexplained by Model (SSR)
- Blue Area - Red Area = Variability Explained by Model (SSM)
- $R^2 = \frac{\text{Area of Blue Squares} - \text{Area of Red Squares}}{\text{Area of Blue Squares}} = \frac{\text{SST} - \text{SSR}}{\text{SST}} = \frac{\text{SSM}}{\text{SST}}$

1.3.9 Variation Explained by SquareFeet Model

- Total variability in house prices SST = 17,558.52
- Variability remaining unexplained after accounting for square feet is SSR = 5,859.07

- Variation explained by model accounting for square feet is

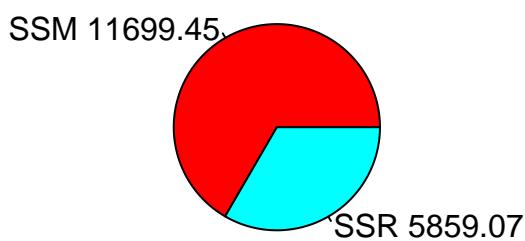
$$SSM = 17,558.52 - 5,859.07 = 11,699.45$$

- Proportion of variation explained by model accounting for square feet is

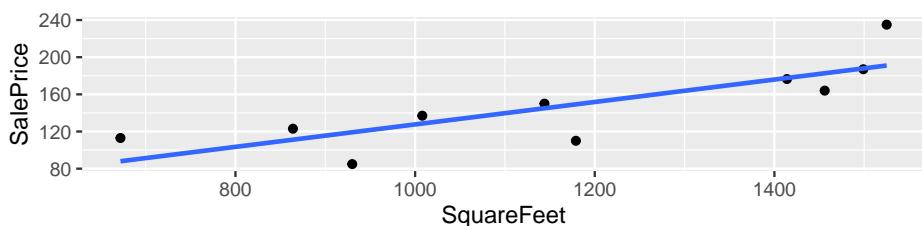
$$R^2 = \frac{11,699.45}{17,558.52} \approx 0.6663$$

- 66.6% of the variation in house price is explained by the model using square feet as an explanatory variable.

Total Variability in house Prices (SST)



1.3.10 Linear Correlation Coefficient



- For linear models with a single quantitative variable, the **linear correlation coefficient** $r = \sqrt{R^2}$, or $r = -\sqrt{R^2}$ (with sign matching the sign on the slope of the line), provides information about the strength and direction of the linear relationship between the variables.
- $-1 \leq r \leq 1$, and r close to ± 1 provides evidence of strong linear relationship, while r close to 0 suggests linear relationship is weak.
- r is only relevant for models with a single quantitative explanatory variable and a quantitative response variable, while R^2 is relevant for any linear model with a quantitative response variable.

```
cor(Houses$SalePrice, Houses$SquareFeet)
```

```
## [1] 0.8162794
```

1.4 Multiple Regression Model

1.4.1 Multiple Regression Model

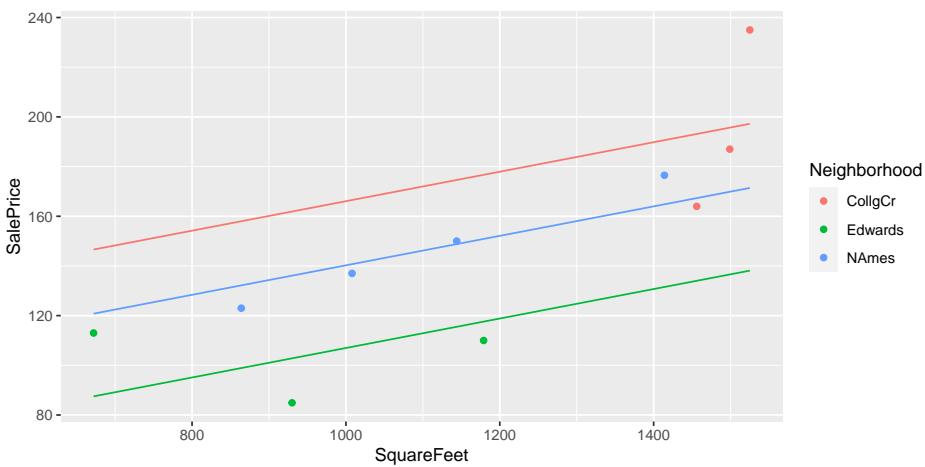
Suppose we have information on both the neighborhood and square feet in the houses. We can account for both of these together using a **multiple regression model**, i.e. a model with more than one explanatory variable.

Houses

```
## # A tibble: 10 x 3
##   Neighborhood SquareFeet SalePrice
##   <fct>          <int>     <dbl>
## 1 NAmes           864      123
## 2 CollgCr         1499     187
## 3 NAmes           1414     176.
## 4 Edwards          672      113
## 5 CollgCr         1456     164.
## 6 Edwards          1179     110
## 7 Edwards          930      84.9
## 8 NAmes           1144     150
## 9 CollgCr         1525     235
## 10 NAmes          1008     137
```

1.4.2 2-Variable Model with Constant Slope

We'll assume the rate of increase wrt. square feet (i.e. slope) is the same in each neighborhood, but that some neighborhoods are more expensive than others.



1.4.3 House Price 2-Variable Model Summary

```
M_Nbhd_SqFt <- lm(data=Houses, SalePrice~SquareFeet+Neighborhood)
summary(M_Nbhd_SqFt)

##
## Call:
## lm(formula = SalePrice ~ SquareFeet + Neighborhood, data = Houses)
##
## Residuals:
##    Min      1Q  Median      3Q     Max 
## -29.125 -9.050 -5.653  9.069 37.791 
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 106.72593   68.92188   1.549   0.172    
## SquareFeet     0.05933    0.04517   1.314   0.237    
## NeighborhoodEdwards -59.09436   32.49761  -1.818   0.119    
## NeighborhoodNAmes  -25.81232   25.59807  -1.008   0.352    
## 
## Residual standard error: 24.55 on 6 degrees of freedom
## Multiple R-squared:  0.7941, Adjusted R-squared:  0.6911 
## F-statistic: 7.711 on 3 and 6 DF,  p-value: 0.01757
```

1.4.4 MR Model for SquareFeet and Neighborhood

In the model using both square feet and neighborhood, the regression equation is

$$\widehat{\text{Price}} = b_0 + b_1 \times \text{SquareFeet} + b_2 \times I_{\text{Edwards}} + b_3 \times I_{\text{NAmes}}$$

$$\widehat{\text{Price}} = 106.73 + 0.06 \times \text{SquareFeet} + -59.09 \times I_{\text{Edwards}} + -25.81 \times I_{\text{NAmes}}$$

- The intercept b_0 represents the expected price of a house in College Creek with 0 square feet.
 - the intercept has no meaningful interpretation in this context
- b_1 represents the expected change in price (in thousands) per one-unit increase in square feet, assuming neighborhood is the same.
 - on average, we expect the price of a house to increase by \$0.05933 thousand (i.e. \$59.33) for each additional square foot, assuming the houses are in the same neighborhood.
- b_2 and b_3 represent the expected difference in price between a house in the Edwards (or North Ames) neighborhood, compared to the College Creek neighborhood, assuming square footage is the same.

- We expect a house in the Edwards neighborhood to cost \$59.094 less than a house in the College Creek Neighborhood, assuming the houses are the same size.
- We expect a house in the North Ames Neighborhood to cost \$25.812 less than a house in the College Creek Neighborhood, assuming the houses are the same size.

1.4.5 Predicting Price in MR Model

$$\widehat{\text{Price}} = 106.73 + 0.06 \times \text{SquareFeet} + -59.09 \times I_{\text{Edwards}} + -25.81 \times I_{\text{NAmes}}$$

- 848 square foot house in College Creek

$$\widehat{\text{Price}} = 106.73 + 0.06 \times 848 + -59.09 \times 0 + -25.81 \times 0 = 157.0378 \text{ thousand}$$

- 1200 square foot house in North Ames

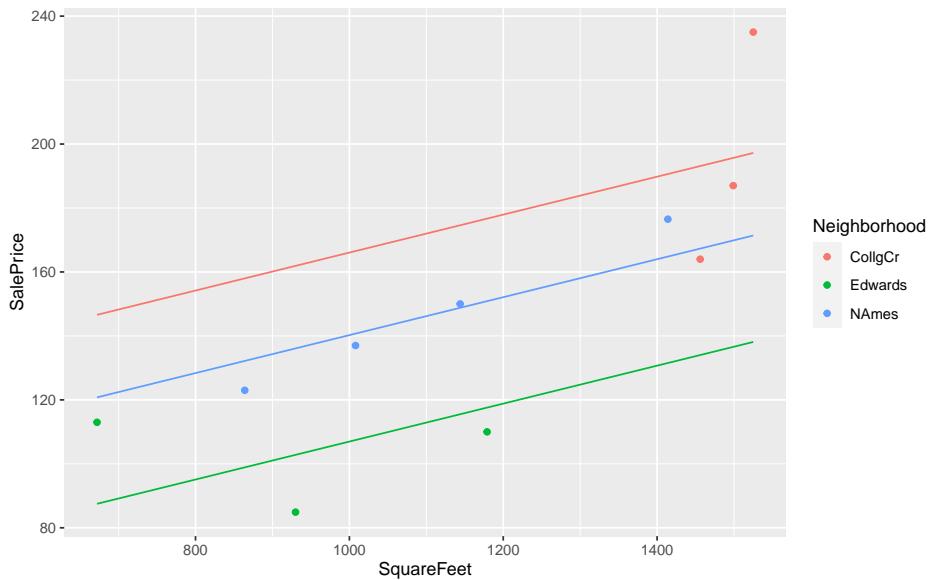
$$\widehat{\text{Price}} = 106.73 + 0.06 \times 1200 + -59.09 \times 0 + -25.81 \times 1 = 152.1096 \text{ thousand}$$

- 2314 square foot house in Edwards

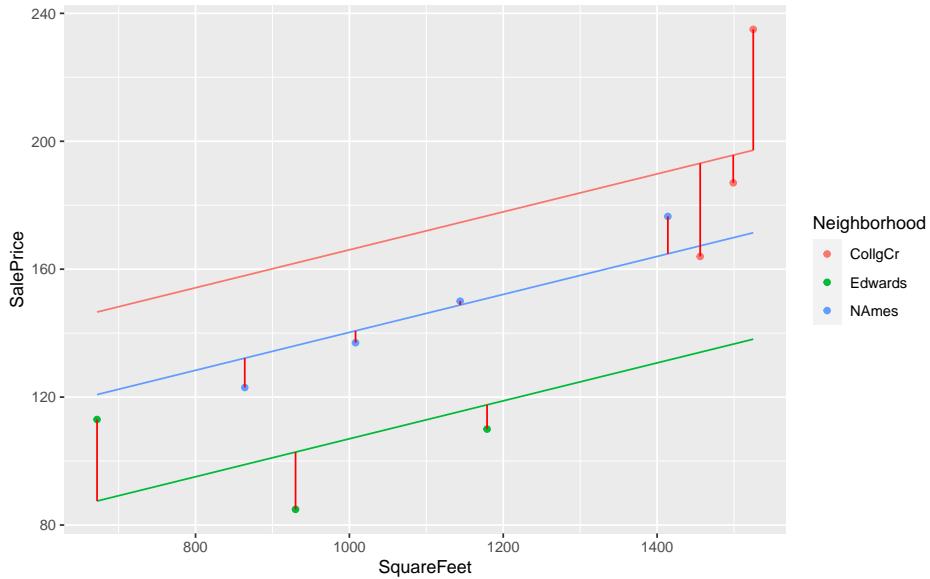
$$\widehat{\text{Price}} = 106.73 + 0.06 \times \text{SquareFeet} + -59.09 \times 1 + -25.81 \times 0 = 184.9212 \text{ thousand}$$

1.4.6 Risk of Extrapolation

Note that 2314 square feet is well outside the range of our observed data. We should treat this prediction with caution, since we don't know whether the trend we see in our data will continue.



1.4.7 Residuals for 2-Variable Model



1.4.8 Residuals for 2-Variable Model (cont.)

```
##      SalePrice Predicted   Residual ResidualSq
## 1     123.00  132.1774 -9.177395  84.224570
```

```

## 2      187.00 195.6662 -8.666221  75.103383
## 3      176.50 164.8106 11.689410 136.642314
## 4      113.00  87.5034 25.496603 650.076744
## 5      163.99 193.1149 -29.124898 848.259699
## 6      110.00 117.5853 -7.585270 57.536321
## 7       84.90 102.8113 -17.911333 320.815835
## 8      150.00 148.7907  1.209343  1.462509
## 9      235.00 197.2089 37.791119 1428.168680
## 10     137.00 140.7214 -3.721358 13.848508

sum(M_Nbhd_SqFt$residuals^2)

## [1] 3616.139

```

1.4.9 Variation Explained by 2-Variable Model

- Total Variation in house prices: $SST=17,558.52$
- Variation remaining unexplained after accounting for square feet is $SSR=3,616.139$
- Variation explained by model accounting for square feet is

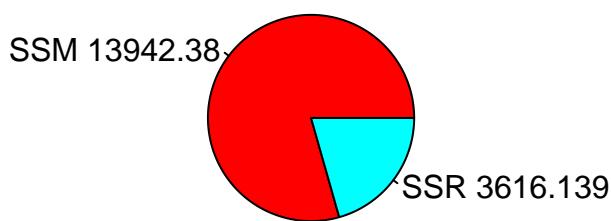
$$SSM = SST - SSR = 17,558.52 - 3,616.139 = 13,942.38$$

- Proportion of variation in house prices explained by model is:

$$R^2 = \frac{13,942.38}{17,558.52} \approx 0.794$$

- 79.4% of the variation in house price is explained by the model using square feet as an explanatory variable.

Total Variability in House Prices (SST)



1.4.10 Model Comparison Summary

Model	Variables	Unexplained Variability	Variability Explained	R^2
0	None	17558.52489	0	0
1	Nbhd	4656.1875667	12902.3373233	0.734819
2	Sq. Ft.	5859.0678887	11699.4570013	0.6663121
3	Nbhd, Sq. Ft.	3616.1385638	13942.3863262	0.7940523

Comments on R^2 :

- R^2 will never decrease when a new variable is added to a model.
- This does not mean that adding more variables to a model always improves its ability to make predictions on new data.
- R^2 measures how well a model fits the data on which it was built.
- It is possible for a model with high R^2 to “overfit” the data it was built from, and thus perform poorly on new data. We will discuss this idea extensively later in the course.

1.5 Least-Squares Estimation

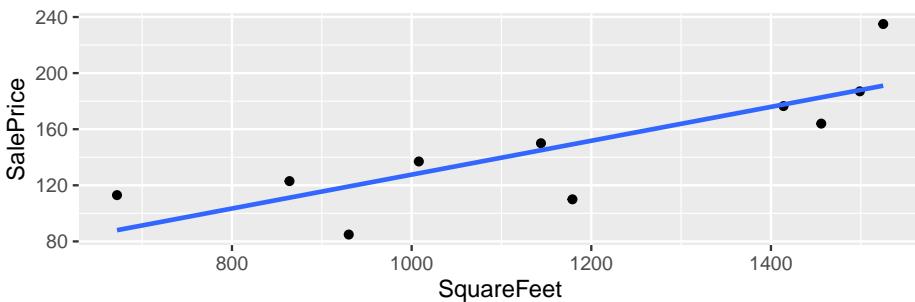
1.5.1 Estimation of Model Coefficients

Learning Outcome:

1. Explain how obtain model coefficients b_0, b_1, b_p .

1.5.2 Least-Squares Estimation

```
ggplot(data=Houses, aes(x=SquareFeet, y=SalePrice)) + geom_point() +
  stat_smooth(method="lm", se=FALSE)
```



- The line $\text{Price} = 6.82 + 0.12 \times \text{Square Feet}$ is considered the “line of best fit” in the sense that it minimizes the sum of the squared residuals.
- This Rossman-Chance applet provides an illustration of the line of best fit.

1.5.3 Least Squares in General Form

Consider a dataset of n observations and p explanatory variables,

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} & y_1 \\ x_{21} & x_{22} & \cdots & x_{2p} & y_2 \\ \vdots & \vdots & & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} & y_n \end{bmatrix}$$

where y_i represents the response variable for case i and x_{ip} represent the value or category indicator of explanatory variable p for case i .

The line of best fit: $\hat{y}_i = b_0 + b_1 x_{i1} + b_2 x_{i2} + \dots + b_p x_{ip}$ is, where b_0, b_1, \dots, b_p are chosen to minimize:

$$\sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n (y_i - (b_0 + b_1 x_{i1} + b_2 x_{i2} + \dots + b_p x_{ip}))^2$$

1.5.4 Least-Squares Estimation in Simple Linear Regression

- Consider a **simple linear regression(SLR)** model, which is one with a single quantitative explanatory variable.
- $\hat{y}_i = b_0 + b_1 x_i$
- we need to minimize:

$$\sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n (y_i - (b_0 + b_1 x_i))^2$$

1.5.5 Least-Squares Estimation in Simple Linear Regression (cont.)

Using calculus, it can be shown that this quantity is minimized when

$$\begin{aligned} \bullet \quad b_1 &= \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2} = \frac{\sum_{i=1}^n x_i y_i - \frac{\sum_{i=1}^n x_i \sum_{i=1}^n y_i}{n}}{\left(\sum_{i=1}^n x_i^2 - \frac{\left(\sum_{i=1}^n x_i \right)^2}{n} \right)} \\ \bullet \quad b_0 &= \bar{y} - b_1 \bar{x} \text{ (where } \bar{y} = \frac{\sum_{i=1}^n y_i}{n}, \text{ and } \bar{x} = \frac{\sum_{i=1}^n x_i}{n}). \end{aligned}$$

1.5.6 LS Estimation for One Categorical Variable

- Consider a model with a single categorical variable (such as neighborhood), with $G+1$ categories, numbered $g = 0, 2, \dots, G$
- Then $\hat{y}_i = b_0 + b_1 x_{i1} + \dots + b_G x_{iG}$.
- we need to minimize

$$\sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n (y_i - (b_0 + b_1 x_{i1} + \dots + b_G x_{iG}))^2.$$

- It can be shown that this is achieved when
 - $b_0 = \bar{y}_0$ (i.e. the average response in the “baseline group”), and
 - $b_j = \bar{y}_j - \bar{y}_0$

1.5.7 LS Estimation More Generally

- For multiple regression models, the logic is the same. We need to choose b_0, b_1, \dots, b_p in order to minimize

$$\sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n (y_i - (b_0 + b_1 x_{i1} + b_2 x_{i2} + \dots + b_p x_{ip}))^2$$

- The mathematics, however are more complicated and require inverting a matrix. This goes beyond the scope of this class, so we will let R do the estimation and use the results.
- More on least squares estimation in multiple regression can be found here.

1.6 ANalysis Of VAriance

1.6.1 ANalysis Of VAriance

Learning Outcomes

1. Use ANOVA F-statistics to compare models and sub-models.
2. Use F-statistics to measure variability between groups, compared to variability within groups.

1.6.2 Comparing Submodels

Model	Variables	Unexplained Variability	Variability Explained	R^2
0	None	17558.52489	0	0
1	Nbhd.	4656.1875667	12902.3373233	0.734819
2	Sq. Ft	5859.0678887	11699.4570013	0.6663121
3	Nbhd, Sq. Ft.	3616.1385638	13942.3863262	0.7940523

- Notice that Model 1 is a submodel of Model 3, since all variables used in Model 1 are also used in Model 3.
- Model 2 is also a submodel of Model 3.
- Model 0 is a submodel of Models 1, 2, and 3.
- Models 1 and 2 are not submodels of each other, since Model 1 contains a variable used in Model 2 and Model 2 contains a variable not used in Model 1.

1.6.3 Comparison of Sub-Models

When one model is a submodel of another, we can compare the amount of variability explained by the models, using a technique known as **ANalysis Of VAriance (ANOVA)**.

Reduced Model: $\hat{y}_i = b_0 + b_1x_{i1} + b_2x_{i2} + \dots + b_qx_{iq}$

Full Model: $\hat{y}_i = b_0 + b_1x_{i1} + b_2x_{i2} + \dots + b_qx_{iq} + b_{q+1}x_{iq+1} + \dots + b_px_{ip}$

p = ## variables in Full Model

q = ## variables in Reduced Model

n = number of observations

We calculate a statistic called F:

$$\begin{aligned}
 F &= \frac{\frac{\text{Unexplained Variability in Reduced Model} - \text{Unexplained Variability in Full Model}}{p-q}}{\frac{\text{Unexplained Variability in Full Model}}{n-(p+1)}} \\
 &= \frac{\frac{\text{SSR}_{\text{Reduced}} - \text{SSR}_{\text{Full}}}{p-q}}{\frac{\text{SSR}_{\text{Full}}}{n-(p+1)}}
 \end{aligned}$$

1.6.4 Comments on F-Statistic

- The F-statistic measures the amount of variability explained by adding additional variable(s) to the model, relative to the total amount of unexplained variability.
- Large values of F indicate that adding the additional explanatory variables is helpful in explaining variability in the response variable
- Small values of F indicate that adding new explanatory variables does not make much of a difference in explaining variability in the response variable
- What counts as “large” is depends on n, p , and q . We will revisit this later in the course.

1.6.5 ANOVA F-Statistic

Let's Calculate an ANOVA F-Statistic to compare Models 2 and 3.

Reduced Model: $\widehat{\text{Price}} = b_0 + b_1 \times \text{SquareFeet}$

Full Model: $\widehat{\text{Price}} = b_0 + b_1 \times \text{SquareFeet} + b_2 \times \text{I}_{\text{Edwards}} + b_3 \times \text{I}_{\text{NAmes}}$

$$\begin{aligned}
 F &= \frac{\frac{\text{SSR}_{\text{Reduced}} - \text{SSR}_{\text{Full}}}{p-q}}{\frac{\text{SSR}_{\text{Full}}}{n-(p+1)}} \\
 &= \frac{\frac{5,859.07 - 3,616.14}{3-1}}{\frac{3,616.13}{10-(3+1)}}
 \end{aligned}$$

```
SSR2 <- sum(M_SqFt$residuals^2); SSR3 <- sum(M_Nbhd_SqFt$residuals^2);
((SSR2-SSR3)/(3-1))/((SSR3)/(10-(3+1)))
```

```
## [1] 1.860766
```

1.6.6 ANOVA F-Statistic for M2 vs M3 in R

```
anova(M_SqFt, M_Nbhd_SqFt)

## Analysis of Variance Table
##
## Model 1: SalePrice ~ SquareFeet
## Model 2: SalePrice ~ SquareFeet + Neighborhood
##   Res.Df   RSS Df Sum of Sq    F Pr(>F)
## 1      8 5859.1
## 2      6 3616.1  2    2242.9 1.8608 0.2351
```

Notice the F-statistic has the same value.

Later, we will examine what this tells us about adding Neighborhood to a model already containing square feet as an explanatory variable.

1.6.7 ANOVA F-Statistic for M1 vs M0

Now, let's compare Models 0 and 1.

Reduced Model: $\widehat{\text{Price}}_i = b_0$

Full Model: $\widehat{\text{Price}}_i = b_0 + b_1 I_{\text{Edwards}} + b_2 I_{\text{NAmes}}$

$$F = \frac{\frac{\text{SSR}_{\text{Reduced}} - \text{SSR}_{\text{Full}}}{p-q}}{\frac{\text{SSR}_{\text{Full}}}{n-(p+1)}} \\ = \frac{\frac{17558.52 - 4656.19}{2-0}}{\frac{4656.19}{10-(2+1)}}$$

```
SSR0 <- sum(M0$residuals^2); SSR1 <- sum(M_Nbhd$residuals^2);
((SSR0-SSR1)/(2-0))/((SSR1)/(10-(2+1)))
```

```
## [1] 9.698531
```

1.6.8 ANOVA F-Statistic for M0 vs M1 in R

```
anova(M0, M_Nbhd)
```

```
## Analysis of Variance Table
##
## Model 1: SalePrice ~ 1
## Model 2: SalePrice ~ Neighborhood
##   Res.Df   RSS Df Sum of Sq    F Pr(>F)
```

```

## 1      9 17558.5
## 2      7 4656.2  2     12902 9.6985 0.009603 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

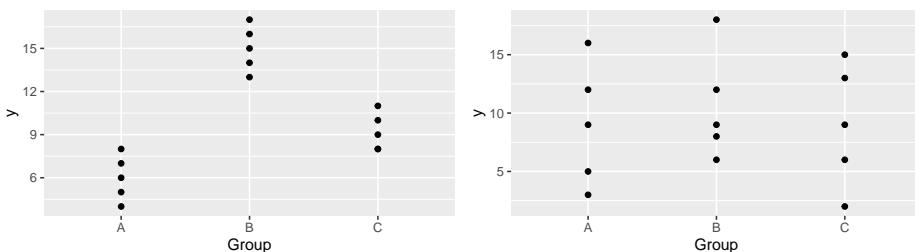
```

1.6.9 ANOVA F-Statistic for Categorical Variables

- The difference between M1 and M0 is that M1 considers the house's neighborhood, while M0 does not.
- If neighborhood is helpful in modeling house price, then we would expect to see a high F-statistic.
- Another way to think about this is that if the amount of variability in house prices between different neighborhoods is large, relative to the amount of variability within neighborhoods, then the F-statistic should be large.
- In fact, an alternative (an mathematically equivalent) way to calculate the F-statistic is to calculate the ratio of variability between different neighborhoods, relative to the amount of variability within neighborhoods.

1.6.10 F-Statistic for Categorical Variables Illustration

An F-statistic compares the amount of variability between groups to the amount of variability within groups.



	Scenario 1	Scenario 2
variation between groups	High	Low
variation within groups	Low	High
F Statistic	Large	Small
Result	Evidence of Group Differences	No evidence of differences

1.6.11 Alternative F-Statistic Formula

For a categorical variable with g groups,

- let $\bar{y}_{1\cdot}, \dots, \bar{y}_{g\cdot}$ represent the mean response for each group.
- let n_1, \dots, n_g represent the sample size for each group
- Then $\frac{\sum_{i=1}^g \sum_{j=1}^{n_i} n_i (\bar{y}_{i\cdot} - \bar{y}_{\cdot\cdot})^2}{g-1}$ gives a measure of how much the group means differ, and
- $\frac{\sum_{i=1}^g \sum_{j=1}^{n_i} (y_{ij} - \bar{y}_{i\cdot})^2}{n-g}$ gives a measure of how much individual observations differ within groups

1.6.12 Alternative F-Statistic Formula (cont.)

- An alternative formula for this F-statistic is:

$$F = \frac{\text{Variability between Neighborhoods}}{\text{Variability within Neighborhoods}} = \frac{\frac{\sum_{i=1}^g \sum_{j=1}^{n_i} n_i (\bar{y}_{i\cdot} - \bar{y}_{\cdot\cdot})^2}{g-1}}{\frac{\sum_{i=1}^g \sum_{j=1}^{n_i} (y_{ij} - \bar{y}_{i\cdot})^2}{n-g}}$$

- It can be shown that this statistic is equivalent to the one we saw previously.

1.6.13 Calculating F-Statistic for Categorical Variables

We have seen previously that:

- $\bar{y}_{\cdot\cdot} = 148.039$ (overall average price), and $n = 10$
- $\bar{y}_{1\cdot} = 195.330$ (average price in College Creek), and $n_1 = 3$
- $\bar{y}_{2\cdot} = 102.633$ (average price in Edwards), and $n_2 = 4$
- $\bar{y}_{3\cdot} = 146.625$ (average price in North Ames), and $n_3 = 3$

Then,

$$\bullet \frac{\sum_{i=1}^g \sum_{j=1}^{n_i} (y_{ij} - \bar{y}_{..})^2}{\frac{12902}{2}}, \text{ and}$$

$$\bullet \frac{\sum_{i=1}^g \sum_{j=1}^{n_i} (y_{ij} - \bar{y}_{i.})^2}{\frac{4656}{7}} = \frac{(123.00 - 146.625)^2 + (187.00 - 195.33)^2 + \dots + (137.00 - 146.625)^2}{10-3} =$$

1.6.14 Calculating F-Statistic for Categorical Variables

$$F = \frac{\frac{\sum_{i=1}^g \sum_{j=1}^{n_i} n_i (y_{ij} - \bar{y}_{..})^2}{g-1}}{\frac{\sum_{i=1}^g \sum_{j=1}^{n_i} (y_{ij} - \bar{y}_{i.})^2}{n-g}} = \frac{\frac{(195.330 - 148.039)^2 + (102.633 - 148.039)^2 + (146.625 - 148.039)^2}{3-1}}{\frac{(123.00 - 146.625)^2 + (187.00 - 195.33)^2 + \dots + (137.00 - 146.625)^2}{10-3}} = \frac{\frac{12902}{2}}{\frac{4656}{7}}$$

- Note that the quantity in the third line is equivalent to the sum of the squared residuals using M2. Thus, we can calculate F using:

```
((3*(195.330-148.039)^2+3*(102.633-148.039)^2+4*(146.625-148.039)^2)/(3-1))/(sum(M_Nbhd$residuals))
```

```
## [1] 9.6986
```

1.6.15 Alternative Calculation in R

This interpretation of the F-statistic can be seen using the AOV command in R.

```
AOV_Nbhd <- aov(data=Houses, SalePrice~Neighborhood)
summary(AOV_Nbhd)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## Neighborhood  2   12902    6451   9.699 0.0096 **
## Residuals     7    4656     665
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- The *Neighborhood* line represents the variability between neighborhoods
- The *Residuals* line represents the variability within neighborhoods

- The first two columns give the quantities we use in our formula. The third column, representing the ratio of the first two columns is called a **mean square**.

1.6.16 F-Statistic in R Output

The last line in the summary output includes the F-statistic for the specified model, compared to a reduced model that includes only the intercept.

Reduced Model: $\hat{Y} = b_0$

Full Model: $\hat{Y} = b_0 + b_1 X_{i1} + \dots + b_p X_{ip}$

This statistic addresses the question “*Do any of the explanatory variables help explain variability in Y?*”.

When there is only one explanatory variable in the model, this statistic can be used to test whether there is evidence that this statistic is associated with Y .

1.6.17 F-Statistic in R Output M1

The F-statistic compares a full model that includes neighborhood to a reduced model that predicts each price using the overall average.

```
summary(M_Nbhd)
```

```
## 
## Call:
## lm(formula = SalePrice ~ Neighborhood, data = Houses)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -31.340  -15.706   -2.477   9.617  39.670 
## 
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)             195.33     14.89  13.118 0.00000349 ***
## NeighborhoodEdwards    -92.70     21.06  -4.402   0.00315 **  
## NeighborhoodNAmes      -48.71     19.70  -2.473   0.04267 *   
## ---                     
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 25.79 on 7 degrees of freedom
## Multiple R-squared:  0.7348, Adjusted R-squared:  0.6591 
## F-statistic: 9.699 on 2 and 7 DF,  p-value: 0.009603
```

1.6.18 F-Statistic in R Output M2

The F-statistic compares a full model that includes square feet to a reduced model that predicts each price using the overall average.

```
summary(M_SqFt)

##
## Call:
## lm(formula = SalePrice ~ SquareFeet, data = Houses)
##
## Residuals:
##     Min      1Q  Median      3Q      Max
## -39.235 -14.309   2.052  10.966  43.971
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.82000   36.35455   0.188  0.85586
## SquareFeet    0.12079    0.03022   3.997  0.00397 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 27.06 on 8 degrees of freedom
## Multiple R-squared:  0.6663, Adjusted R-squared:  0.6246
## F-statistic: 15.97 on 1 and 8 DF,  p-value: 0.003967
```

1.6.19 F-Statistic in R Output M3

The F-statistic compares a full model that includes square feet and neighborhood to a reduced model that predicts each price using only the overall average.

```
summary(M_Nbhd_SqFt)

##
## Call:
## lm(formula = SalePrice ~ SquareFeet + Neighborhood, data = Houses)
##
## Residuals:
##     Min      1Q  Median      3Q      Max
## -29.125 -9.050  -5.653   9.069  37.791
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept) 106.72593   68.92188   1.549   0.172
## SquareFeet    0.05933    0.04517   1.314   0.237
## NeighborhoodEdwards -59.09436   32.49761  -1.818   0.119
```

```
## NeighborhoodNAmes -25.81232 25.59807 -1.008 0.352
##
## Residual standard error: 24.55 on 6 degrees of freedom
## Multiple R-squared: 0.7941, Adjusted R-squared: 0.6911
## F-statistic: 7.711 on 3 and 6 DF, p-value: 0.01757
```

1.6.20 When to Use F-Statistics for Model Comparison

- We have used F-statistics to compare models 1 and 3, and models 0 and 2.
- We could also calculate F-statistics comparing models 2 and 3, models 0 and 1, and models 0 and 3.
- We cannot use an F-statistic to compare models 1 and 2, since neither is a submodel of the other.
- When comparing a model to the “intercept-only” model, we can use the model summary output. When comparing other to other submodels, use the `aov()` or `anova()` commands.

1.6.21 Interaction Models with Categorical Variables

Learning Outcomes

1. Fit models involving interactions.
2. Make interpret regression coefficients for models with interactions.
3. Compare and contrast the assumptions behind models with and without interaction terms.
- We will build a model to predict the weight of a bear, using other characteristics including season, sex, and age.

1.7 Interaction Models with Categorical Variables

1.7.1 Bear Data

Data are available in the `Bolstad` R package. We examine the structure of the dataset.

```
library(Bolstad)
data(bears)
glimpse(bears)

## #> #> Rows: 143
## #> #> Columns: 12
## #> #> $ ID      <int> 39, 41, 41, 41, 41, 43, 43, 45, 45, 45, 48, 69, 83, 83, 83, 83, 91-
## #> #> $ Age     <int> 19, 19, 20, 23, 29, 19, 20, 55, 67, 81, NA, 115, 117, 124, 140-
## #> #> $ Month   <int> 7, 7, 8, 11, 5, 7, 8, 7, 9, 10, 7, 9, 4, 8, 8, 4, 9, 7, 4, ~
## #> #> $ Sex      <int> 1, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, ~
## #> #> $ Head.L   <dbl> 10.0, 11.0, 12.0, 12.5, 12.0, 11.0, 12.0, 16.5, 16.5, 15.5, 16-
## #> #> $ Head.W   <dbl> 5.0, 6.5, 6.0, 5.0, 6.0, 5.5, 5.5, 9.0, 9.0, 8.0, 8.0, 10.0, 7-
## #> #> $ Neck.G   <dbl> 15.0, 20.0, 17.0, 20.5, 18.0, 16.0, 17.0, 28.0, 27.0, 31.0, 32-
## #> #> $ Length    <dbl> 45.0, 47.5, 57.0, 59.5, 62.0, 53.0, 56.0, 67.5, 78.0, 72.0, 77-
## #> #> $ Chest.G   <dbl> 23.0, 24.0, 27.0, 38.0, 31.0, 26.0, 30.5, 45.0, 49.0, 54.0, 52-
## #> #> $ Weight    <int> 65, 70, 74, 142, 121, 80, 108, 344, 371, 416, 432, 348, 476, 4-
## #> #> $ Obs.No    <int> 1, 1, 2, 3, 4, 1, 2, 1, 1, 1, 2, 3, 4, 1, 1, 2, 1, 1, 2, ~
## #> #> $ Name      <fct> Allen, Berta, Berta, Berta, Clyde, Doc, Doc, Qui-
```

1.7.2 Bears Data Cleaning

Notice that we have multiple observations on the same bears. The procedures we have learned so far require observations to be independent of each other. Thus, we'll keep only the first observation on each bear.

```
Bears_Subset <- bears %>% filter(Obs.No == 1)
```

The variables Month and Sex are coded as integers, but it really makes more sense to think of these as categorical variables. Thus, we will convert them to factors.

```
Bears_Subset$Month <- as.factor(Bears_Subset$Month)
Bears_Subset$Sex <- as.factor(Bears_Subset$Sex)
```

1.7.3 Examining the Subset

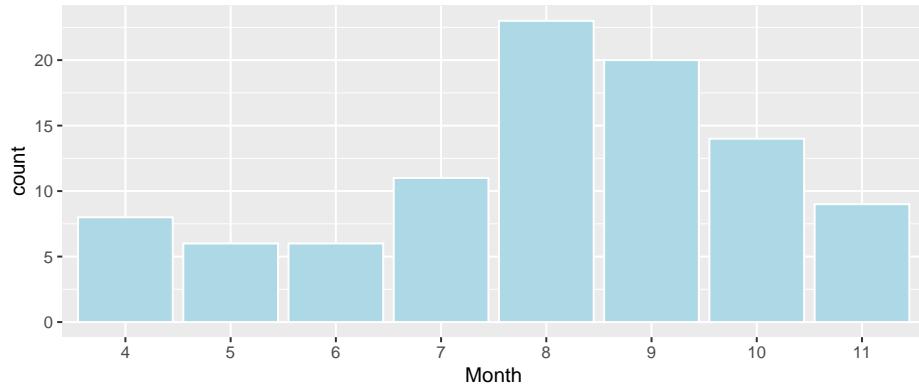
```
glimpse(Bears_Subset)

## #> #> Rows: 97
## #> #> Columns: 12
## #> #> $ ID      <int> 39, 41, 43, 45, 48, 69, 83, 91, 179, 241, 253, 274, 280, 394, ~
## #> #> $ Age     <int> 19, 19, 19, 55, 81, NA, 115, 104, 100, 56, 51, 57, 53, NA, 68, ~
## #> #> $ Month   <fct> 7, 7, 7, 7, 9, 10, 7, 8, 4, 7, 4, 9, 5, 6, 8, 8, 8, 8, 8, 9-
## #> #> $ Sex      <fct> 1, 2, 1, 1, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 1, 2, 1, 1, ~
```

```
## $ Head.L <dbl> 10.0, 11.0, 11.0, 16.5, 15.5, 16.0, 17.0, 15.5, 13.0, 15.0, 13-
## $ Head.W <dbl> 5.0, 6.5, 5.5, 9.0, 8.0, 8.0, 10.0, 6.5, 7.0, 7.5, 8.0, 7.0, 6-
## $ Neck.G <dbl> 15.0, 20.0, 16.0, 28.0, 31.0, 32.0, 31.5, 22.0, 21.0, 26.5, 27-
## $ Length <dbl> 45.0, 47.5, 53.0, 67.5, 72.0, 77.0, 72.0, 62.0, 70.0, 73.5, 68-
## $ Chest.G <dbl> 23, 24, 26, 45, 54, 52, 49, 35, 41, 41, 49, 38, 31, 32, 44, 19-
## $ Weight <int> 65, 70, 80, 344, 416, 432, 348, 166, 220, 262, 360, 204, 144, ~
## $ Obs.No <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ Name    <fct> Allen, Berta, Clyde, Doc, Quincy, Kooch, Charlie, Geraldine, F~
```

1.7.4 Examining the Month Variable

```
ggplot(data=Bears_Subset, aes(x=Month)) + geom_bar(color="white", fill="lightblue")
```



Notice that there are no measurements between December and March, the bears' hibernation season.

1.7.5 Season Variable

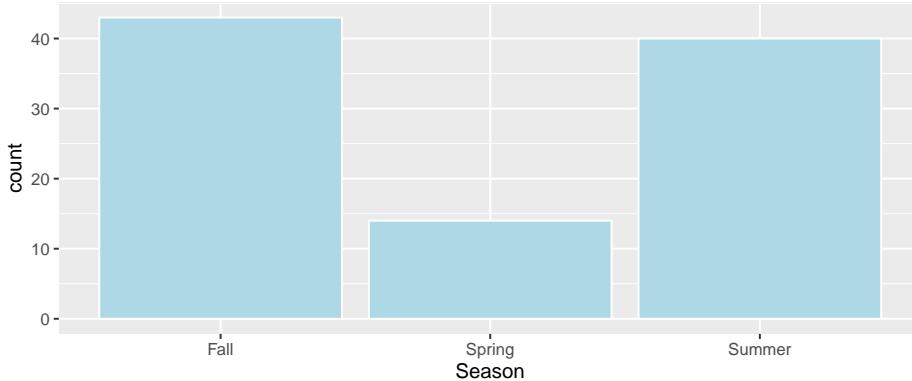
When a variable has many categories, it becomes harder to compare the categories. Sometimes, it is helpful to create a new variable with fewer variables by combining categories.

Let's combine the months of April and May into a category called "Spring", June, July, and August into "Summer", and "September", "October", and "November", into "Fall".

```
Bears_Subset <- Bears_Subset %>% mutate(Season = ifelse(Month %in% 4:5, "Spring",
ifelse(Month %in% 6:8, "Summer", "Fall")))
Bears_Subset$Season <- as.factor(Bears_Subset$Season)
```

1.7.6 Visualizing Season Variable

```
ggplot(data=Bears_Subset, aes(x=Season)) + geom_bar(color="white", fill="lightblue")
```



1.7.7 Bears Data Summary

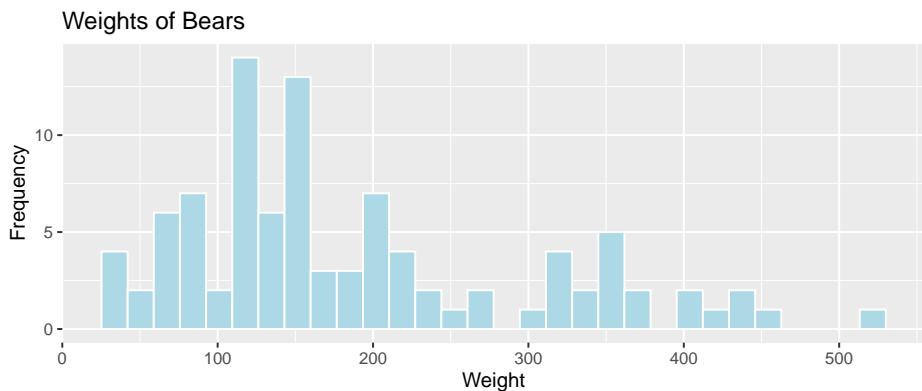
```
summary(Bears_Subset)
```

```
##      ID          Age        Month       Sex     Head.L
##  Min. : 39.0   Min. : 8.00    8 :23  1:62  Min. : 9.00
##  1st Qu.:525.0  1st Qu.: 17.00   9 :20  2:35  1st Qu.:12.00
##  Median :579.0   Median : 34.00  10:14          Median :13.00
##  Mean   :537.6   Mean   : 42.64   7 :11          Mean   :13.29
##  3rd Qu.:640.0   3rd Qu.: 57.25  11: 9          3rd Qu.:14.50
##  Max.   :911.0   Max.   :177.00   4 : 8          Max.   :18.50
##           NA's   :41  (Other):12
##      Head.W        Neck.G       Length      Chest.G
##  Min. : 4.000   Min. :10.00   Min. :36.00   Min. :19.00
##  1st Qu.: 5.000  1st Qu.:17.50  1st Qu.:54.50  1st Qu.:30.00
##  Median : 6.000  Median :20.00  Median :61.00  Median :34.00
##  Mean   : 6.364  Mean   :21.03  Mean   :60.41  Mean   :35.93
##  3rd Qu.: 7.000  3rd Qu.:24.00  3rd Qu.:67.00  3rd Qu.:42.00
##  Max.   :10.000  Max.   :32.00  Max.   :83.00  Max.   :55.00
##
##      Weight      Obs.No      Name     Season
##  Min. : 26.0   Min. :1  Ian   : 2  Fall  :43
##  1st Qu.:114.0  1st Qu.:1  Abe   : 1  Spring:14
##  Median :154.0   Median :1  Addy  : 1  Summer:40
##  Mean   :187.2   Mean   :1  Albert : 1
##  3rd Qu.:236.0  3rd Qu.:1  Allen  : 1
##  Max.   :514.0   Max.   :1  (Other):89
```

```
##          NA's : 2
```

1.7.8 Histogram of Bear Weights

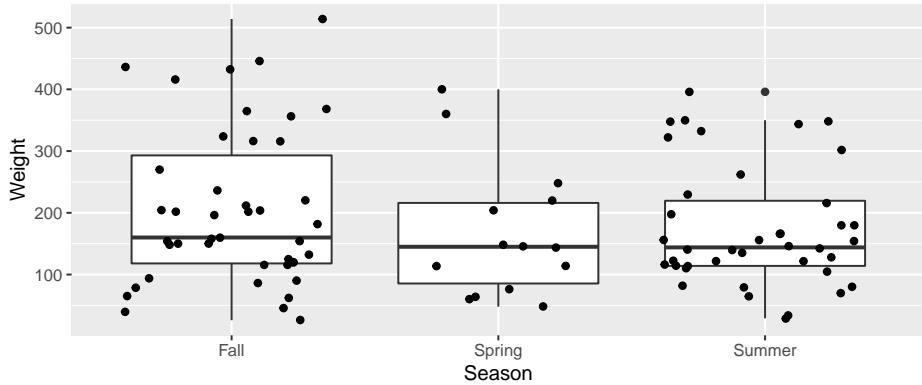
```
ggplot(data=Bears_Subset, aes(x=Weight)) +
  geom_histogram(color="white", fill="lightblue") +
  xlab("Weight") + ylab("Frequency") + ggtitle("Weights of Bears")
```



We see that bears most commonly weigh between 100 and 200 lbs, and the distribution of weights is right-skewed.

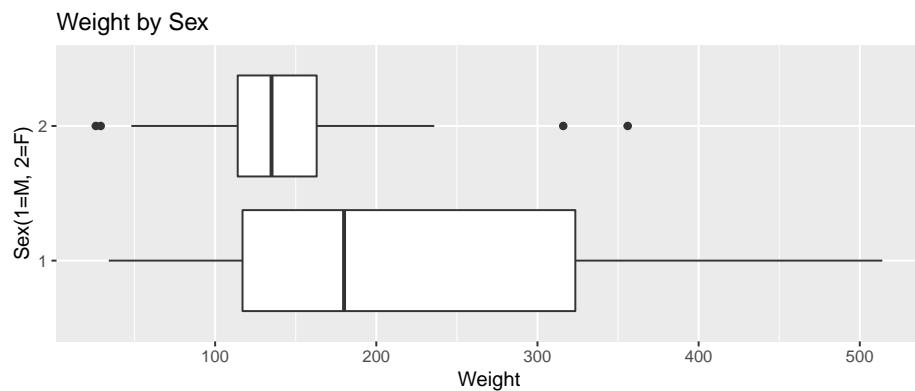
1.7.9 Weight by Season

```
ggplot(data=Bears_Subset, aes(x=Season, y=Weight)) + geom_boxplot() + geom_jitter()
```



1.7.10 Boxplot of Weight and Sex

```
ggplot(data=Bears_Subset, aes(y=Weight, x=Sex)) +
  geom_boxplot() +
  xlab("Sex(1=M, 2=F)") + ylab("Weight") + ggtitle("Weight by Sex") + coord_flip()
```



We see that male bears (Category 1) weigh more than female bears on average, and that there is more variability in the weights of male bears than female bears.

1.7.11 Bears: Season and Sex Model without Interaction

$$\widehat{\text{Weight}} = b_0 + b_1 \times I_{\text{Spring}} + b_2 \times I_{\text{Summer}} + b_3 \times I_{\text{Female}}$$

Season	Male	Female
Fall	b_0	$b_0 + b_3$
Spring	$b_0 + b_1$	$b_0 + b_1 + b_3$
Summer	$b_0 + b_2$	$b_0 + b_2 + b_3$

Model Assumptions:

- Allows weights to differ by sex and season.
- Assumes difference between sexes is the same in each season and difference between seasons is the same for each sex.

1.7.12 Bears Season and Sex Model Output

Let's model weights of male and female bears individually by season.

```
Bears_M_Season_Sex <- lm(data=Bears_Subset, Weight~Season + Sex)
summary(Bears_M_Season_Sex)

##
## Call:
## lm(formula = Weight ~ Season + Sex, data = Bears_Subset)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -186.73  -76.56  -17.32   63.44  287.27
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)
## (Intercept) 226.73     18.14 12.500 < 0.0000000000000002 ***
## SeasonSpring -25.54     33.56 -0.761    0.44856
## SeasonSummer -28.17     23.79 -1.184    0.23939
## Sex2         -67.24     23.06 -2.916    0.00445 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 108.3 on 93 degrees of freedom
## Multiple R-squared:  0.1024, Adjusted R-squared:  0.07343
## F-statistic: 3.536 on 3 and 93 DF,  p-value: 0.01774
```

1.7.13 Predictions using Season and Sex

$$\widehat{\text{Weight}} = 226.73 - 25.54 \times I_{\text{Spring}} - 28.17 \times I_{\text{Summer}} - 67.24 \times I_{\text{Female}}$$

	Male	Female
Fall	226.73 -	226.73 -
	25.54(0) -	25.54(0) -
	28.17(0) -	28.17(0) -
	67.24(0)=226.73	24(1)=159.59
Spring	226.73 -	226.73 -
	25.54(1) -	25.54(1) -
	28.17(0) -	28.17(0) -
	67.24(0)=201.07	24(1)=133.95
Summer	226.73 -	226.73 -
	25.54(0) -	25.54(0) -
	28.17(1) -	28.17(1) -
	67.24(0)=198.55	24(1)=131.32

1.7.14 Weight using Season and Sex Model Interpretations

$$\widehat{\text{Weight}} = 226.73 - 25.54 \times I_{\text{Spring}} - 28.17 \times I_{\text{Summer}} - 67.24 \times I_{\text{Female}}$$

Note that fall and male are treated as the “baseline” levels in this model.

- On average, male bears are expected to weigh 226.73 lbs in the fall.
- On average, bears of the same sex are expected to weigh 25.54 lbs less in the spring than in the fall.
- On average, bears of the same sex are expected to weigh 28.17 lbs less in the summer than in the fall.
- On average, female bears are expected to weigh 67.24 lbs than male bears, in the same season.

The model with season and sex explains about 10% of the variation in bear weights.

1.7.15 Season and Sex with Interaction

$$\widehat{\text{Weight}} = b_0 + b_1 \times I_{\text{Spring}} + b_2 \times I_{\text{Summer}} + b_3 \times I_{\text{Female}} + b_4 \times I_{\text{Spring}} I_{\text{Female}} + b_5 \times I_{\text{Summer}} I_{\text{Female}}$$

	Male	Female
Fall	b_0	$b_0 + b_3$
Spring	$b_0 + b_1$	$b_0 + b_1 + b_3 + b_4$
Summer	$b_0 + b_2$	$b_0 + b_2 + b_3 + b_5$

Model Assumptions:

- Allows for differences between seasons and sexes.
- Allows for differences between sexes to vary between seasons and difference between seasons to vary between sexes.

b_4 and b_5 are called **interaction effects**.

1.7.16 Bears Season and Sex Interaction Model

To fit an interaction model in R, use * instead of +

```
Bears_M_Season_Sex_Int <- lm(data=Bears_Subset, Weight~Season * Sex)
summary(Bears_M_Season_Sex_Int)
```

```

## Call:
## lm(formula = Weight ~ Season * Sex, data = Bears_Subset)
##
## Residuals:
##   Min     1Q Median     3Q    Max
## -181.14 -73.14 -13.07  58.81 292.86
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 221.14     20.28 10.905 <0.0000000000000002 ***
## SeasonSpring -14.00     45.99 -0.304    0.762
## SeasonSummer -17.95     29.49 -0.608    0.544
## Sex2         -50.07     35.54 -1.409    0.162
## SeasonSpring:Sex2 -29.08     68.34 -0.425    0.672
## SeasonSummer:Sex2 -30.41     50.73 -0.599    0.550
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 109.2 on 91 degrees of freedom
## Multiple R-squared:  0.1064, Adjusted R-squared:  0.0573
## F-statistic: 2.167 on 5 and 91 DF,  p-value: 0.06458

```

1.7.17 Predictions for Season and Sex Interaction Model

$$\widehat{\text{Weight}} = b_0 + b_1 \times I_{\text{Spring}} + b_2 \times I_{\text{Summer}} + b_3 \times I_{\text{Female}} + b_4 \times I_{\text{Spring}}I_{\text{Female}} + b_5 \times I_{\text{Summer}}I_{\text{Female}}$$

$$\begin{aligned} \widehat{\text{Weight}} = & 221.14 - 14.00 \times I_{\text{Spring}} - 17.95 \times I_{\text{Summer}} - 50.07 \times I_{\text{Female}} \\ & - 29.08 \times I_{\text{Spring}}I_{\text{Female}} - 30.41 \times I_{\text{Summer}}I_{\text{Female}} \end{aligned}$$

	Male	Female
Fall	221.14 -14.00(0) -17.95(0) -50.07(0) - 29.08(0)(0)	221.14 -14.00(0) -17.95(0) -50.07(1) - 29.08(0)(1)
	-	-
	30.41(0)(0)= =171.07	230.41(0)(1)

	Male	Female
Spring	221.14 -14.00(1) -17.95(0) -50.07(0) - 29.08(1)(0) - 30.41(0)(0) =207.14	211.375 + 8.01221.14 -14.00(1) -17.95(0) -50.07(1) - 29.08(1)(1) -
Summer	221.14 -14.00(0) -17.95(1) -50.07(0) - 29.08(0)(0) - 30.41(1)(0)	221.14 -14.00(0) -17.95(1) -50.07(1) - 29.08(0)(1) - 30.41(1)(1)
	=203.19	=128.00 =122.71

1.7.18 Season and Sex Interaction Model Interpretations

$$\widehat{\text{Weight}} = b_0 + b_1 \times I_{\text{Spring}} + b_2 \times I_{\text{Summer}} + b_3 \times I_{\text{Female}} + b_4 \times I_{\text{Spring}} I_{\text{Female}} + b_5 \times I_{\text{Summer}} I_{\text{Female}}$$

	Male	Female
Fall	b_0	$b_0 + b_3$
Spring	$b_0 + b_1$	$b_0 + b_1 + b_3 + b_4$
Summer	$b_0 + b_2$	$b_0 + b_2 + b_3 + b_5$

- b_0 represents expected weight of male bear in fall
- b_1 represents difference between expected male bear weight in spring, compared to fall
- b_2 represents difference between expected male bear weight in summer, compared to fall
- b_3 represents difference between expected female bear weight, compared to male bear weight in fall
- b_4 represents difference in expected weights between the sexes in the spring, compared to the difference in the fall
- b_5 represents difference in expected weights between the sexes in the sum-

mer, compared to the difference in the fall

1.7.19 Interpretations for Season and Sex Interaction Model

$$\widehat{\text{Weight}} = 221.14 - 14.00 \times I_{\text{Spring}} - 17.95 \times I_{\text{Summer}} - 50.07 \times I_{\text{Female}} \\ - 29.08 \times I_{\text{Spring}} I_{\text{Female}} - 30.41 \times I_{\text{Summer}} I_{\text{Female}}$$

- On average, male bears are expected to weigh 221.14 lbs in the fall.
- On average male bears are expected to weigh 14 lbs less in the spring than in the fall.
- On average, male bears are expected to weigh 17.95 lbs less in the summer than in the fall.
- On average, female bears are expected to weigh 50.07 lbs less than male bears in the fall.
- On average, the female bears are expected to weigh 29.08 lbs. less relative to male bears in the spring, compared to the expected difference the fall. Thus, female bears are expected to weigh $50.07 + 29.08 = 79.17$ lbs less than male bears in the spring.
- On average, female bears are expected to weigh 30.41 lbs less, relative to male bears in the summer, compared to the expected difference in the fall. Thus, female bears are expected to weigh $50.07 + 30.41 = 80.48$ lbs less than male bears in the summer.

The interaction model explains about 10.6% of the variation in bear weights.

1.7.20 Predicting New Observations in R

We can calculate predictions directly in R by putting the new data in a data.frame and calling the `predict()` function.

```
Season <- c("Fall", "Fall", "Spring", "Spring", "Summer", "Summer")
Sex <- factor(c(1,2,1,2,1,2))
NewBears <- data.frame(Season, Sex)

predict(Bears_M_Season_Sex, newdata=NewBears)

##      1       2       3       4       5       6
## 226.7290 159.4900 201.1909 133.9520 198.5586 131.3197
```

```
predict(Bears_M_Season_Sex_Int, newdata=NewBears)

##      1      2      3      4      5      6
## 221.1379 171.0714 207.1429 128.0000 203.1923 122.7143
```

1.8 Interaction Models with Categorical and Quantitative Variables

1.8.1 Summary of Bears Dataset

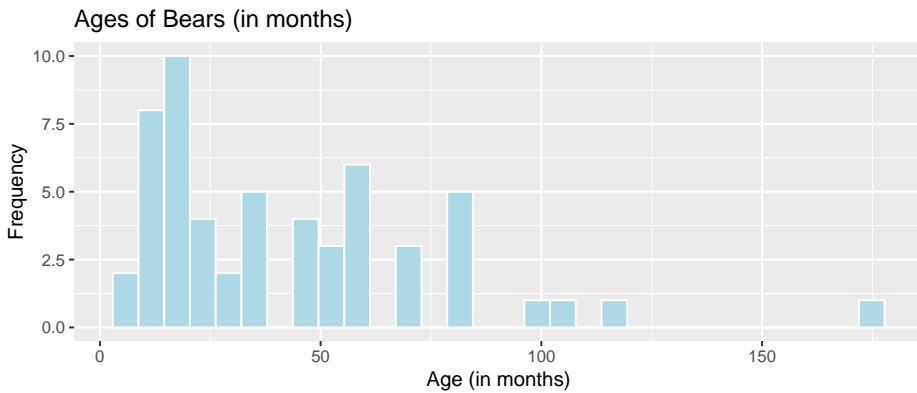
```
summary(Bears_Subset)

##      ID          Age         Month       Sex        Head.L
##  Min.   : 39.0   Min.   : 8.00   8   :23   1:62   Min.   : 9.00
##  1st Qu.:525.0  1st Qu.: 17.00  9   :20   2:35   1st Qu.:12.00
##  Median :579.0  Median : 34.00  10  :14           Median :13.00
##  Mean   :537.6  Mean   : 42.64  7   :11           Mean   :13.29
##  3rd Qu.:640.0  3rd Qu.: 57.25  11  : 9           3rd Qu.:14.50
##  Max.   :911.0  Max.   :177.00  4   : 8           Max.   :18.50
##             NA's   :41    (Other):12
##      Head.W        Neck.G        Length       Chest.G
##  Min.   : 4.000   Min.   :10.00   Min.   :36.00   Min.   :19.00
##  1st Qu.: 5.000   1st Qu.:17.50   1st Qu.:54.50   1st Qu.:30.00
##  Median : 6.000   Median :20.00   Median :61.00   Median :34.00
##  Mean   : 6.364   Mean   :21.03   Mean   :60.41   Mean   :35.93
##  3rd Qu.: 7.000   3rd Qu.:24.00   3rd Qu.:67.00   3rd Qu.:42.00
##  Max.   :10.000   Max.   :32.00   Max.   :83.00   Max.   :55.00
##
##      Weight        Obs.No       Name       Season
##  Min.   : 26.0   Min.   :1   Ian   : 2   Fall  :43
##  1st Qu.:114.0  1st Qu.:1   Abe   : 1   Spring:14
##  Median :154.0  Median :1   Addy  : 1   Summer:40
##  Mean   :187.2  Mean   :1   Albert : 1
##  3rd Qu.:236.0  3rd Qu.:1   Allen  : 1
##  Max.   :514.0  Max.   :1   (Other):89
##             NA's   : 2
```

1.8.2 Histogram of Bear Ages

```
ggplot(data=Bears_Subset, aes(x=Age)) +
  geom_histogram(color="white", fill="lightblue") +
```

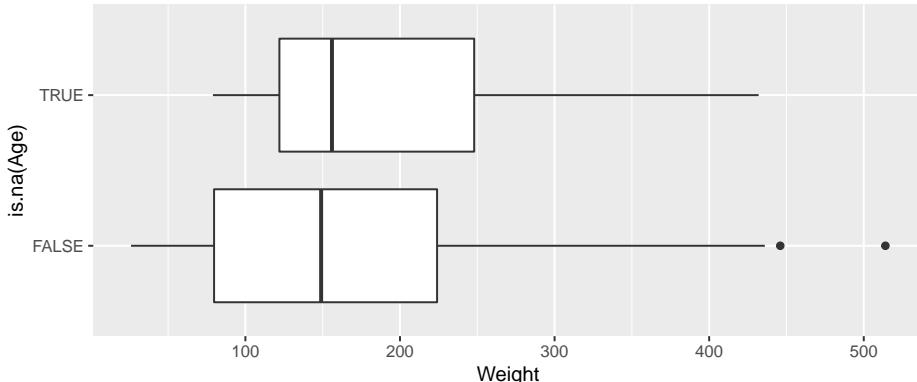
```
xlab("Age (in months)") + ylab("Frequency") + ggtitle("Ages of Bears (in months)")
```



1.8.3 Bears with Missing Ages

Recall that 41 bears had missing ages. They will be ignored if we use age in our model. To see how this might impact predicted weights, let's look at how weights compare for bears with and without missing ages.

```
ggplot(data=Bears_Subset, aes(x=is.na(Age), y=Weight)) + geom_boxplot() + coord_flip()
```

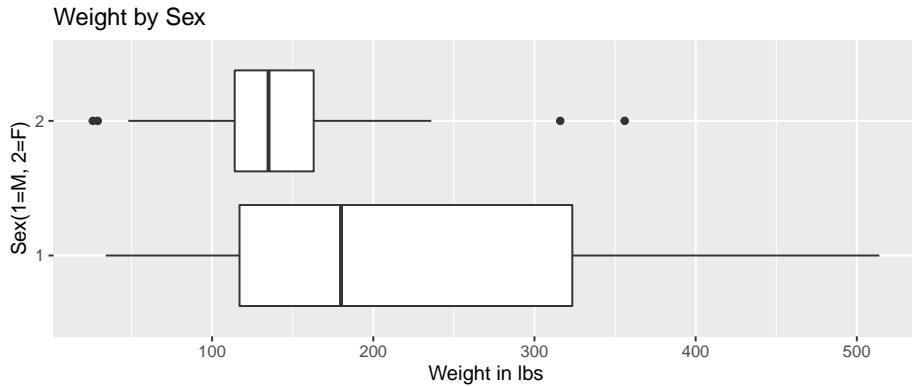


Bears with missing ages do not seem to be systematically different than those whose ages are recorded, with respect to weight, so the missing ages should not cause too much concern with our model results.

1.8.4 Boxplot of Weight and Sex

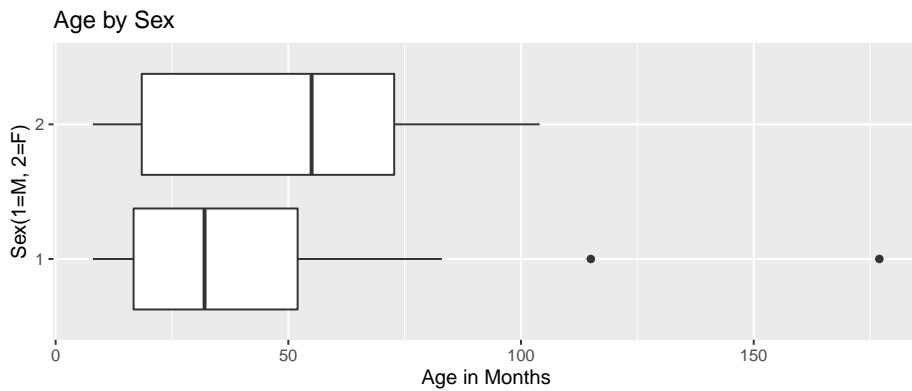
```
ggplot(data=Bears_Subset, aes(y=Weight, x=Sex)) +
  geom_boxplot()
```

```
xlab("Sex(1=M, 2=F)") + ylab("Weight in lbs") + ggtitle("Weight by Sex") + coord_flip()
```



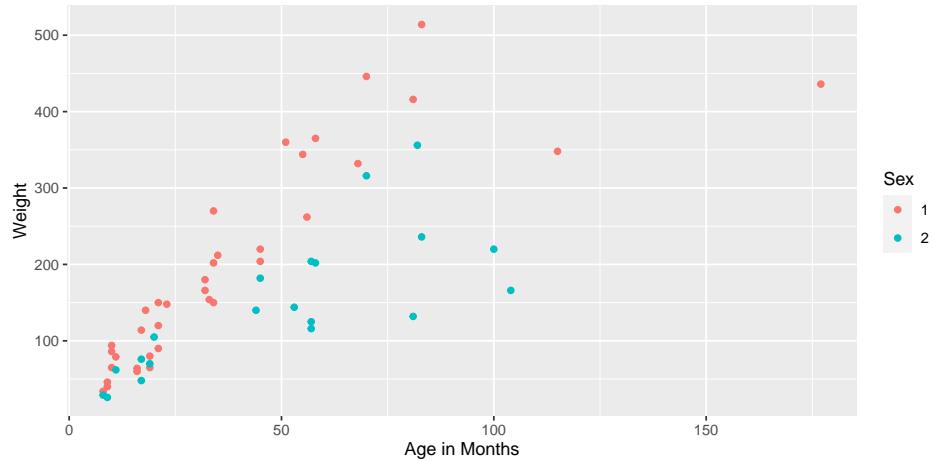
1.8.5 Boxplot of Age and Sex

```
ggplot(data=Bears_Subset, aes(y=Age, x=Sex)) +
  geom_boxplot() +
  xlab("Sex(1=M, 2=F)") + ylab("Age in Months") + ggtitle("Age by Sex") + coord_flip()
```



The median age for female bears is older than for male bears. There are 2 male bears that are much older than any others.

1.8.6 Scatterplot of Age and Weight



We see that there is a positive, roughly linear, relationship between age and weight.

We should note that this linear trend is not likely to continue outside the range of our observed ages.

1.8.7 Bears: Age and Sex Model

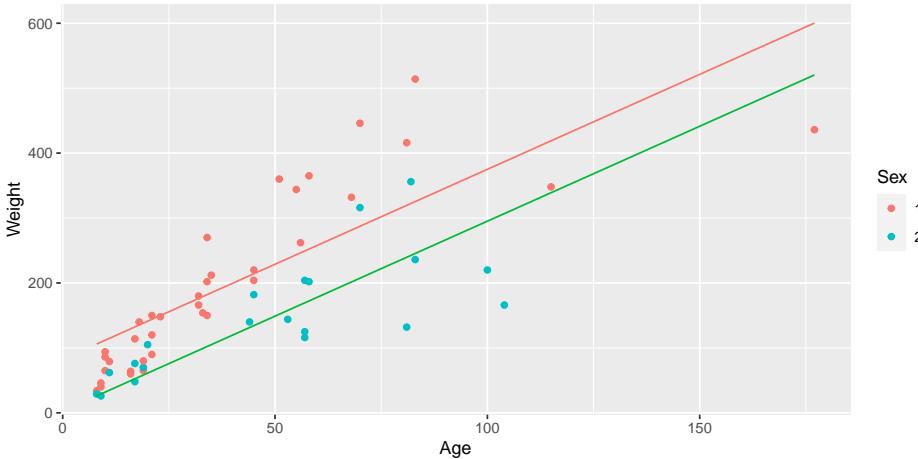
$$\widehat{\text{Weight}} = b_0 + b_1 \times \text{Age} + b_2 \times I_{Female}$$

Sex	Pred. Weight
M	$b_0 + b_1 \times \text{Age}$
F	$(b_0 + b_2) + b_1 \times \text{Age}$

Model Assumptions:

- Assumes weight increases linearly with age
- Allows for differences in expected weight for male and female bears of same age
- Assumes male and female bears gain weight at the same rate as they age

1.8.8 Constant Slope Model for Bears



1.8.9 Bears Age and Sex Model Output

```
Bears_M_Age_Sex <- lm(data=Bears_Subset, Weight ~ Age + Sex)
summary(Bears_M_Age_Sex)

##
## Call:
## lm(formula = Weight ~ Age + Sex, data = Bears_Subset)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -164.194  -48.483   -3.723   27.766  188.684 
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)    
## (Intercept) 82.6049    16.4019   5.036 0.0000058437067215 ***
## Age          2.9242     0.2914  10.035 0.0000000000000744 ***
## Sex2        -79.8967    20.1416  -3.967     0.00022 ***  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 71.33 on 53 degrees of freedom
## (41 observations deleted due to missingness)
## Multiple R-squared:  0.6679, Adjusted R-squared:  0.6554 
## F-statistic: 53.29 on 2 and 53 DF,  p-value: 0.0000000000002061
```

1.8.10 Predicted Bear Weights from Age and Sex Model

$$\widehat{\text{Weight}} = 82.60 + 2.92 \times \text{Age} - 79.90 \times I_{Female}$$

Suppose Sally and Yogi are 25 month old bears, Sally is a female, Yogi a male.

Sally's Predicted Weight:

$$\widehat{\text{Weight}} = 82.60 + 2.920 \times 25 - 78.90 \times 1 \approx 75.8 \text{ lbs.}$$

Yogi's Predicted Weight:

$$\widehat{\text{Weight}} = 82.60 + 2.920 \times 25 - 78.90 \times 0 \approx 155.7 \text{ lbs.}$$

1.8.11 Age and Sex Model for Male and Female Bears

The interaction model allows for different intercepts, but assumes that the average monthly weight increase is the same for male and female bears.

$$\widehat{\text{Weight}} = 82.60 + 2.92 \times \text{Age} - 79.90 \times I_{Female}$$

Male Bears:

$$\widehat{\text{Weight}} = 82.60 + 2.92 \times \text{Age}$$

Female Bears:

$$\widehat{\text{Weight}} = (82.60 - 79.90) + (2.92) \times \text{Age} = 2.7 + 2.92 \times \text{Age}$$

1.8.12 Bears Predictions with predict

```
Sex <- factor(c(1, 2))
Age <- c(25, 25)
NewBears <- data.frame(Age, Sex)

predict(Bears_M_Age_Sex, newdata=NewBears)

##           1           2
## 155.71061  75.81394
```

1.8.13 Age and Sex Model Interpretations

- For bears of the same sex, weight is expected to increase by 2.92 lbs. each month.
- On average, a female bear is expected to weigh 79.90 lbs less than a male bear of the same age.

- The intercept $b_0 = 82.60$ represents the expected weight of a male bear at birth. We should treat this interpretation with caution, since all bears in the dataset were at least 8 months old.

Approximately 67% of the variation in bear weights is explained by the model using age and sex as explanatory variables.

1.8.14 Bears Age and Sex Model with Interaction

$$\widehat{\text{Weight}} = b_0 + b_1 \times \text{Age} + b_2 \times I_{Female} + b_3 \times \text{Age} \times I_{Female}$$

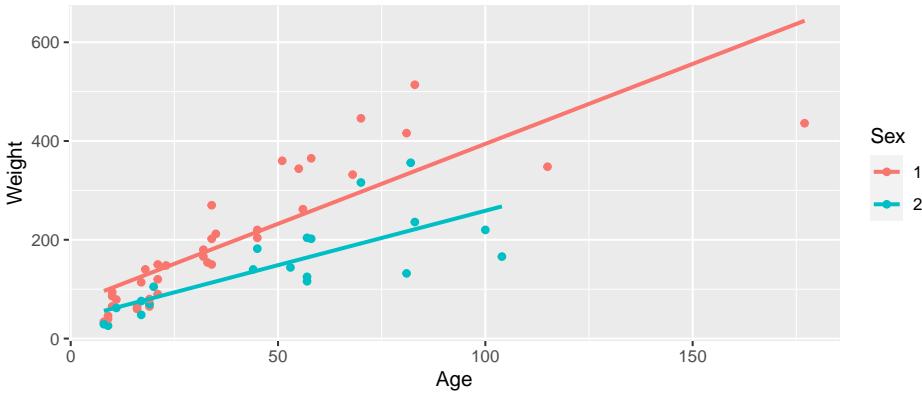
Sex	Pred. Weight
M	$b_0 + b_1 \times \text{Age}$
F	$(b_0 + b_2) + (b_1 + b_3) \times \text{Age}$

Model Assumptions:

- Assumes weight increases linearly with age
- Allows for differences in expected weight for male and female bears of same age
- Allows male and female bears to gain weight at different rates as they age

1.8.15 Bears Age and Sex Interaction Model

```
ggplot(data=Bears_Subset, aes(x=Age, y=Weight, color=Sex)) +
  geom_point() + stat_smooth(method="lm", se=FALSE)
```



1.8.16 Summary for Age-Sex Interaction Model

```
Bears_M_Age_Sex_Int <- lm(data=Bears_Subset, Weight~ Age*Sex)
summary(Bears_M_Age_Sex_Int)

##
## Call:
## lm(formula = Weight ~ Age * Sex, data = Bears_Subset)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -207.583  -38.854   -9.574   23.905  174.802
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)    
## (Intercept) 70.4322    17.7260   3.973 0.000219 *** 
## Age         3.2381    0.3435   9.428 0.000000000000765 ***
## Sex2        -31.9574   35.0314  -0.912 0.365848    
## Age:Sex2    -1.0350    0.6237  -1.659 0.103037    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 70.18 on 52 degrees of freedom
## (41 observations deleted due to missingness)
## Multiple R-squared:  0.6846, Adjusted R-squared:  0.6664 
## F-statistic: 37.62 on 3 and 52 DF,  p-value: 0.0000000000004552
```

1.8.17 Predictions from Interaction Model for Bears

$$\widehat{\text{Weight}} = 70.43 + 3.24 \times \text{Age} - 31.96 \times I_{Female} - 1.04 \times \text{Age} \times I_{Female}$$

Suppose Sally and Yogi are 25 month old bears, Sally is a female, Yogi a male.

Sally's Predicted Weight:

$$\widehat{\text{Weight}} = 70.43 + 3.24 \times 25 - 31.96 \times 1 - 1.04 \times 25 \times 1 \approx 93.55 \text{ lbs.}$$

Yogi's Predicted Weight:

$$\widehat{\text{Weight}} = 70.43 + 3.24 \times 25 - 31.96 \times 0 - 1.04 \times 25 \times 0 \approx 151.38 \text{ lbs.}$$

1.8.18 Interaction Model for Male and Female Bears

The interaction model allow for different intercepts and slopes between male and female bears.

$$\widehat{\text{Weight}} = 70.43 + 3.24 \times \text{Age} - 31.96 \times I_{Female} - 1.04 \times \text{Age} \times I_{Female}$$

Male Bears:

$$\widehat{\text{Weight}} = 70.43 + 3.24 \times \text{Age}$$

Female Bears:

$$\widehat{\text{Weight}} = (70.43 - 31.96) + (3.24 - 1.04) \times \text{Age} = 38.46 - 2.20 \times \text{Age}$$

1.8.19 Bears Interaction Model Predictions with predict

```
Sex <- factor(c(1, 2))
Age <- c(25, 25)
NewBears <- data.frame(Age, Sex)

predict(Bears_M_Age_Sex_Int, newdata=NewBears)

##           1           2
## 151.38566  93.55306
```

1.8.20 Interpretations for Age and Sex Model with Interaction

$$\widehat{\text{Weight}} = b_0 + b_1 \times \text{Age} + b_2 \times I_{Female} + b_3 \times \text{Age} \times I_{Female}$$

Sex	Pred. Weight
M	$b_0 + b_1 \times \text{Age}$
F	$(b_0 + b_2) + (b_1 + b_3) \times \text{Age}$

Interpretations:

b_0 : expected weight of a male bear at birth (caution:extrapolation)

b_1 : expected weight gain per month for male bears

b_2 : expected difference in weight between female and male bears at birth (caution:extrapolation)

b_3 : expected difference in monthly weight gain for female bears, compared to male bears

1.8.21 Bears Weight Model Considerations

- R^2 increased from 0.67 to 0.68 when the interaction term is added. This is a relatively small increase, so we might question whether the interaction term is needed.

- The constant slope model allows us to combine information across sexes to estimate the expected slope. The interaction model treats the two sexes completely separately, thus has less information to use for each estimate.
- Which model is preferable is not clear. In addition to the data, we should consider other relevant information. Do experts who study bears believe it is reasonable to assume that male and female bears grow at the same rate per month?
- While the models yield drastically different predictions for very young bears, the differences are not as big for bear 8 months or older. Regardless of model we use, we should be careful about making predictions for bears that are younger or older than those that we have data on.

1.8.22 Bears Weight Model Considerations (continued)

Both of these models contain assumptions that are probably unrealistic

- Both models assume that bears of the same sex gain weight linearly with age.
- A more realistic model might assume that bears gain weight more quickly when they are younger, and that the rate of growth slows once they reach adulthood.
- Are there variables not included in the model that might be predictive of a bear's weight?

Of course there is no statistical model that perfectly describes expected weight gain of bears. The question is whether we can find a model that provides an approximation that is reasonable enough to draw conclusions from.

As statistician George Box famously said,

"All models are wrong, but some are useful."

1.9 Interaction Models with Quantitative Variables

1.9.1 2015 Cars Dataset

We consider data from the Kelly Blue Book, pertaining to new cars, released in 2015. We'll investigate the relationship between price, length, and time it takes to accelerate from 0 to 60 mph.

```
data(Cars2015)
glimpse(Cars2015)
```

```

## Rows: 110
## Columns: 20
## $ Make      <fct> Chevrolet, Hyundai, Kia, Mitsubishi, Nissan, Dodge, Chevrole-
## $ Model     <fct> Spark, Accent, Rio, Mirage, Versa Note, Dart, Cruze LS, 500L-
## $ Type      <fct> Hatchback, Hatchback, Sedan, Hatchback, Sedan, Se-
## $ LowPrice   <dbl> 12.270, 14.745, 13.990, 12.995, 14.180, 16.495, 16.170, 19.3-
## $ HighPrice  <dbl> 25.560, 17.495, 18.290, 15.395, 17.960, 23.795, 25.660, 24.6-
## $ Drive      <fct> FWD, FWD, FWD, FWD, FWD, FWD, FWD, FWD, FWD, AWD, ~
## $ CityMPG   <int> 30, 28, 28, 37, 31, 23, 24, 24, 28, 30, 27, 27, 25, 27, 30, ~
## $ HwyMPG    <int> 39, 37, 36, 44, 40, 35, 36, 33, 38, 35, 33, 36, 36, 37, 39, ~
## $ FuelCap   <dbl> 9.0, 11.4, 11.3, 9.2, 10.9, 14.2, 15.6, 13.1, 12.4, 11.1, 11-
## $ Length     <int> 145, 172, 172, 149, 164, 184, 181, 167, 179, 154, 156, 180, ~
## $ Width      <int> 63, 67, 68, 66, 67, 72, 71, 70, 72, 67, 68, 69, 70, 68, 69, ~
## $ Wheelbase  <int> 94, 101, 101, 97, 102, 106, 106, 103, 104, 99, 98, 104, 104, ~
## $ Height     <int> 61, 57, 57, 59, 61, 58, 58, 66, 58, 59, 58, 58, 57, 58, 59, ~
## $ UTurn      <int> 34, 37, 37, 32, 37, 38, 38, 37, 39, 34, 35, 38, 37, 36, 37, ~
## $ Weight     <int> 2345, 2550, 2575, 2085, 2470, 3260, 3140, 3330, 2990, 2385, ~
## $ Acc030    <dbl> 4.4, 3.7, 3.5, 4.4, 4.0, 3.4, 3.7, 3.9, 3.4, 3.9, 3.9, 3.7, ~
## $ Acc060    <dbl> 12.8, 10.3, 9.5, 12.1, 10.9, 9.3, 9.8, 9.5, 9.2, 10.8, 11.1, ~
## $ QtrMile   <dbl> 19.4, 17.8, 17.3, 19.0, 18.2, 17.2, 17.6, 17.4, 17.1, 18.3, ~
## $ PageNum   <int> 123, 148, 163, 188, 196, 128, 119, 131, 136, 216, 179, 205, ~
## $ Size       <fct> Small, Small, Small, Small, Small, Small, Small, Small, Smal-

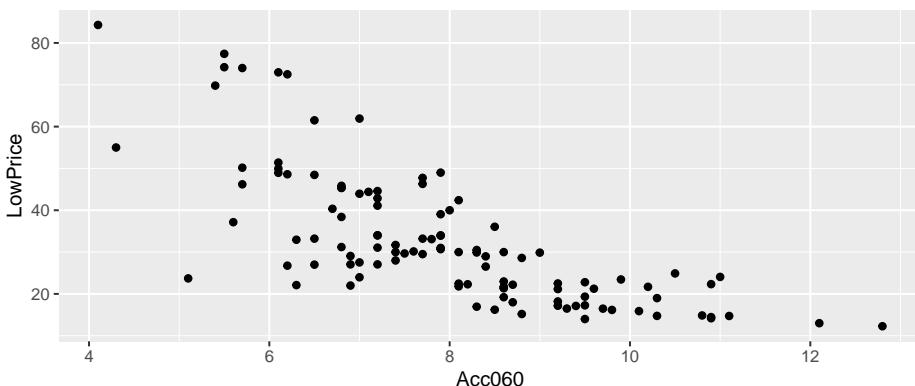
```

1.9.2 Car Price and Acceleration Time

`LowPrice` represents the price of a standard (non-luxury) model of a car. `Acc060` represents time it takes to accelerate from 0 to 60 mph.

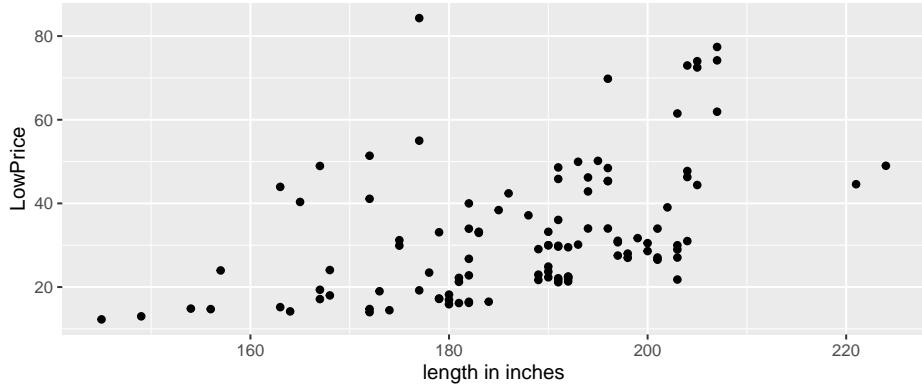
```
data(Cars2015)
```

```
CarsA060 <- ggplot(data=Cars2015, aes(x=Acc060, y=LowPrice)) + geom_point()
CarsA060
```



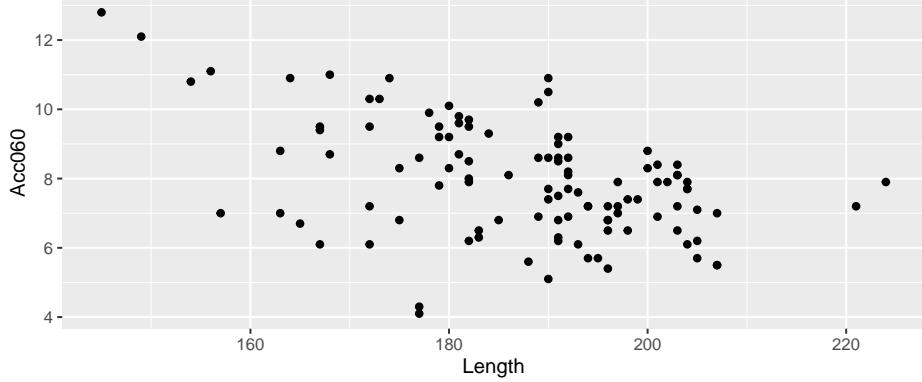
1.9.3 Length and Sale Price

```
ggplot(data=Cars2015, aes(x=Length, y=LowPrice)) + geom_point() + xlab("length in inches")
```



1.9.4 Relationship between Acc060 and Length

```
ggplot(data=Cars2015, aes(x=Length, y=Acc060)) + geom_point()
```



Lack of correlation among explanatory variables is a good thing. If explanatory variables are highly correlated, then using them together is unlikely to perform much better than using one or the other. In fact, it can cause problems, as we'll see later.

1.9.5 Modeling Price using Acc060

$$\widehat{Price} = b_0 + b_1 \times \text{Acc. Time}$$

- Model assumes expected price is a linear function of acceleration time.

Parameter Interpretations:

b_0 represents intercept of regression line, i.e. expected price of a car that can accelerate from 0 to 60 mph in no time. This is not a meaningful interpretation in context.

b_1 represents slope of regression line, i.e. expected change in price for each additional second it takes to accelerate from 0 to 60 mph.

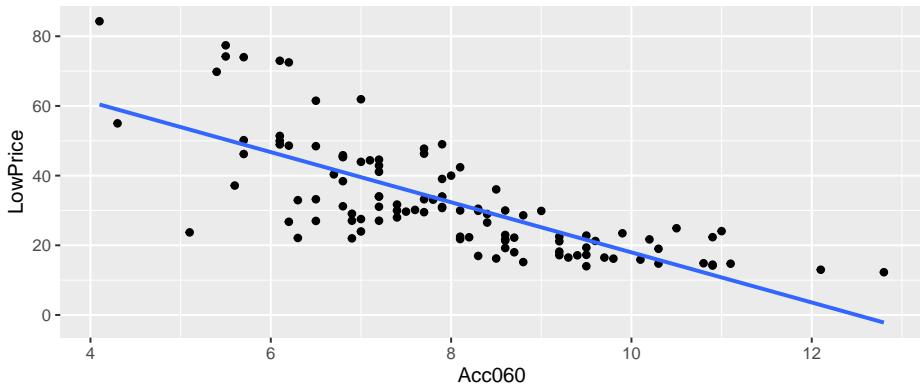
1.9.6 Modeling for Car Price and Acceleration

```
Cars_M_A060 <- lm(data=Cars2015, LowPrice~Acc060)
summary(Cars_M_A060)

##
## Call:
## lm(formula = LowPrice ~ Acc060, data = Cars2015)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -29.512   -6.544   -1.265    4.759   27.195
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)
## (Intercept)  89.9036    5.0523  17.79 <0.0000000000000002 ***
## Acc060       -7.1933    0.6234 -11.54 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.71 on 108 degrees of freedom
## Multiple R-squared:  0.5521, Adjusted R-squared:  0.548
## F-statistic: 133.1 on 1 and 108 DF,  p-value: < 0.0000000000000002
```

1.9.7 Price and A060 Regression Line

```
CarsA060 + stat_smooth(method="lm", se=FALSE)
```



1.9.8 Acc060 Model Interpretations

$$\widehat{Price} = b_0 + b_1 \times \text{Acc. Time}$$

$$\widehat{Price} = 89.90 - 7.193 \times \text{Acc. Time}$$

- Intercept b_0 might be interpreted as the price of a car that can accelerate from 0 to 60 in no time, but this is not a meaningful interpretation since there are no such cars.
- $b_1 = -7.193$ tells us that on average, the price of a car is expected to decrease by 7.19 thousand dollars for each additional second it takes to accelerate from 0 to 60 mph.
- $R^2 = 0.5521$ tells us that 55% of the variation in price is explained by the linear model using acceleration time as the explanatory variable.

1.9.9 Modeling Price using Acc060 and Length

$$\widehat{Price} = b_0 + b_1 \times \text{Acc. Time} + b_2 \times \text{Length}$$

Model Assumptions:

- Assumes expected price is a linear function of acceleration time and length.
- Assumes that expected price increases at same rate with respect to acc. time, for cars of all lengths.
- Assumes that expected price increases at same rate with respect to length, for cars of all acceleration times.

1.9.10 Model Using Length and Acceleration Time

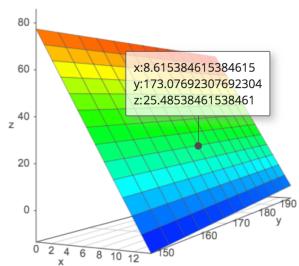
```
Cars_M_A060_L <- lm(data=Cars2015, LowPrice ~ Acc060 + Length)
summary(Cars_M_A060_L)

##
## Call:
## lm(formula = LowPrice ~ Acc060 + Length, data = Cars2015)
##
## Residuals:
##    Min      1Q  Median      3Q     Max 
## -27.969  -6.625  -1.418   4.934  28.394 
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)    
## (Intercept) 51.83936   18.00737   2.879     0.00482 **  
## Acc060      -6.48340    0.69248  -9.363 0.000000000000000141 ***
## Length       0.17316    0.07874   2.199     0.03003 *   
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
##
## Residual standard error: 10.52 on 107 degrees of freedom
## Multiple R-squared:  0.5715, Adjusted R-squared:  0.5635 
## F-statistic: 71.36 on 2 and 107 DF,  p-value: < 0.00000000000000022
```

1.9.11 Regression Plane

$$\widehat{Price} = b_0 + b_1 \times \text{Acc. Time} + b_2 \times \text{Length}$$

$$\widehat{Price} = 51.83936 + -6.48340 \times \text{Acc. Time} + 0.17316 \times \text{Length}$$



We can further explore the regression plane a academo.org.

1.9.12 Acc060 and Length Model Interpretations

$$\widehat{Price} = b_0 + b_1 \times \text{Acc. Time} + b_2 \times \text{Length}$$

$$\widehat{Price} = 51.83936 + -6.48340 \times \text{Acc. Time} + 0.17316 \times \text{Length}$$

- Intercept b_0 might be interpreted as the price of a car that can accelerate from 0 to 60 in no time and has length 0, but this is not a meaningful interpretation since there are no such cars.
- $b_1 = -6.4834$ tells us that on average, the price of a car is expected to decrease by 6.48 thousand dollars for each additional second it takes to accelerate from 0 to 60 mph., assuming length is held constant.
- $b_2 = 0.17316$ tells us that on average, the price of a car is expected to increase by 0.173 thousand dollars (or 173) for each additional inch in length, assuming the time it takes to accelerate from 0 to 60 mph. is held constant.
- $R^2 = 0.5715$ tells us that 57% of the variation in price is explained by the linear model using acceleration time and length as the explanatory variables.

1.9.13 Modeling Price using Acc060 and Length with Interaction

$$\widehat{Price} = b_0 + b_1 \times \text{Acc. Time} + b_2 \times \text{Length} + b_3 \times \text{Acc. Time} \times \text{Length}$$

Model Assumptions:

- Assumes expected price is a linear function of acceleration time and length.
- Assumes that rate of increase in expected price increases with respect to acc. time differs depending on length of the car.
- Assumes that rate of increase in expected price increases with respect to length differs depending on acceleration time of the car.

1.9.14 Predictions using Price, Acc060, Length Model with Interaction

$$\widehat{Price} = b_0 + b_1 \times \text{Acc. Time} + b_2 \times \text{Length} + b_3 \times \text{Acc. Time} \times \text{Length}$$

Predicted Price for 150 inch car:

$$\begin{aligned}\widehat{Price} &= b_0 + b_1 \times \text{Acc. Time} + b_2 \times 150 + b_3 \times \text{Acc. Time} \times 150 \\ &= (b_0 + 150b_2) + (b_1 + 150b_3) \times \text{Acc. Time}\end{aligned}$$

Predicted Price for 200 inch car:

$$\widehat{Price} = b_0 + b_1 \times \text{Acc. Time} + b_2 \times 200 + b_3 \times \text{Acc. Time} \times 200 \\ = (b_0 + 200b_2) + (b_1 + 200b_3) \times \text{Acc. Time}$$

1.9.15 Interaction Model for Quantitative Variables

We specify an interaction model in R using the `*` between the explanatory variables.

```
Cars_M_A060_L_Int <- lm(data=Cars2015, LowPrice ~ Acc060 * Length)
summary(Cars_M_A060_L_Int)
```

```
## 
## Call:
## lm(formula = LowPrice ~ Acc060 * Length, data = Cars2015)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -31.0782  -5.2974  -0.6007  4.2479  31.3622 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -160.25438   54.11260  -2.961  0.00378 ** 
## Acc060       19.00490    6.21545   3.058  0.00282 ** 
## Length        1.34893    0.29447   4.581 0.0000127 *** 
## Acc060:Length -0.14260    0.03459  -4.123 0.0000745 *** 
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 9.814 on 106 degrees of freedom
## Multiple R-squared:  0.6307, Adjusted R-squared:  0.6203 
## F-statistic: 60.35 on 3 and 106 DF,  p-value: < 0.0000000000000022
```

1.9.16 Cars Interaction Model Interpretations

$$\widehat{Price} = -160.25438 + 19.00490 \times \text{Acc. Time} + 1.34893 \times \text{Length} - 0.14260 \times \text{Acc. Time} \times \text{Length}$$

For a car that is 150 inches long:

$$\widehat{Price} = -160.25438 + 19.00490 \times \text{Acc. Time} + 1.34893 \times 150 - 0.14260 \times \text{Acc. Time} \times 150 \\ = 42.085 - 2.385 \times \text{Acc. Time}$$

For a 150 inch car, price is expected to decrease by 2.385 thousand dollars for each additional second it takes to accelerate from 0 to 60 mph.

$$\widehat{\text{Price}} = -160.25438 + 19.00490 \times \text{Acc. Time} + 1.34893 \times 200 - 0.14260 \times \text{Acc. Time} \times 200 \\ = 109.532 - 9.515 \times \text{Acc. Time}$$

For a 200 inch car, price is expected to decrease by 9.515 thousand dollars for each additional second it takes to accelerate from 0 to 60 mph.

When using an interaction model, we can only interpret coefficients with respect to one variable for a given value of the other variable.

$R^2 = 0.6307$ tells us that 63% of variation in car price is explained by the model using length and acceleration time with interaction.

1.9.17 Predicting Car Price Using predict

Predict the price of a 200 inch car that can accelerate from 0 to 60 mph in 8 seconds.

```
predict(Cars_M_A060_L_Int, newdata=data.frame(Acc060=8, Length=200))
```

```
##           1
## 33.40344
```

Predict the price of a 150 inch car that can accelerate from 0 to 60 mph in 10 seconds.

```
predict(Cars_M_A060_L_Int, newdata=data.frame(Acc060=10, Length=150))
```

```
##           1
## 18.22735
```

Chapter 2

Interval Estimation via Simulation

2.1 Sampling Variability and Margin of Error

2.1.1 Sampling Variability

- Next, we turn our attention to the topic of quantifying uncertainty associated with a sample or experiment performed using random assignment.
- Our goal is to generalize results to a larger population.
- If our sample was collected in a way that is not representative of a larger population of interest, then we should be cautious about generalizing our results.
- If we already have data about the entire population we are interested in, then these results are not relevant.

2.1.2 Election Polling Example

A Washington Post poll, of 702 registered voters in the state of Wisconsin, taken from September 8-13, 2020 showed that:

- 50% intend to vote for Joe Biden
- 46% intend to vote for Donald Trump
- 4% weren't sure or intend to vote for other candidates

Is this strong enough information to say that Joe Biden is ahead among all voters in Wisconsin? Is it possible that Biden is not really ahead, and the poll just happened to sample more of his supporters but random chance?

2.1.3 Investigation by Simulation

Key Questions:

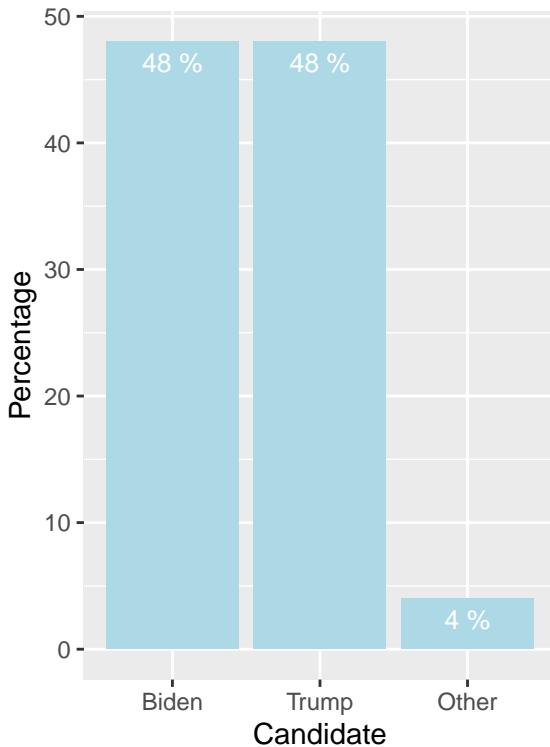
By how much could proportions in a random sample of 702 voters plausibly differ from proportions for all Wisconsin voters?

We'll investigate this using the following steps:

1. Assume we know the percentage of voters in the state supporting each candidate.
2. Simulate taking random samples of 702 voters. Record how far the poll result differs from those of the population.

2.1.4 Larger Population

- As of August 1, 2020, there were 3,420,587 registered voters in the state of Wisconsin.
- We'll assume (hypothetically) that for the upcoming election:
 - 1,641,882 (48%) plan to vote for the Biden
 - 1,641,882 (48%) plan to vote for the Trump
 - 136,823 (4%) plan to vote for a third-party or independent candidate



2.1.5 A sample of 702 voters

```
Vote <- factor(c(rep("Biden", 1641882), rep("Trump", 1641882), rep("Other", 136823)))
Population <- as.data.frame(Vote)

Sample <- sample_n(Population, 702, replace=FALSE)
```

Population Summary:

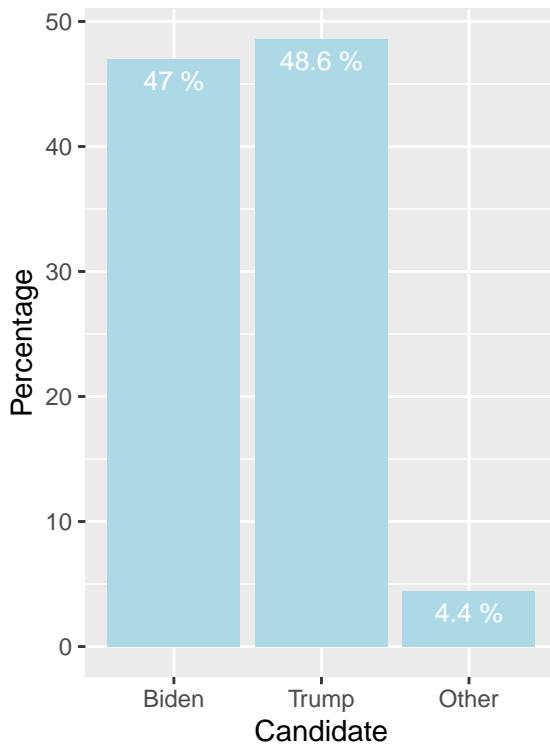
```
summary(Population$Vote)
```

```
##   Biden   Other   Trump
## 1641882 136823 1641882
```

Results in random sample of 702 voters:

```
summary(Sample$Vote)
```

```
## Biden Other Trump
## 330    31    341
```



2.1.6 A Second Sample of 702 Voters

Let's simulate taking another random sample of 702 voters.

```
Sample <- sample_n(Population, 702, replace=FALSE)
```

Population Summary:

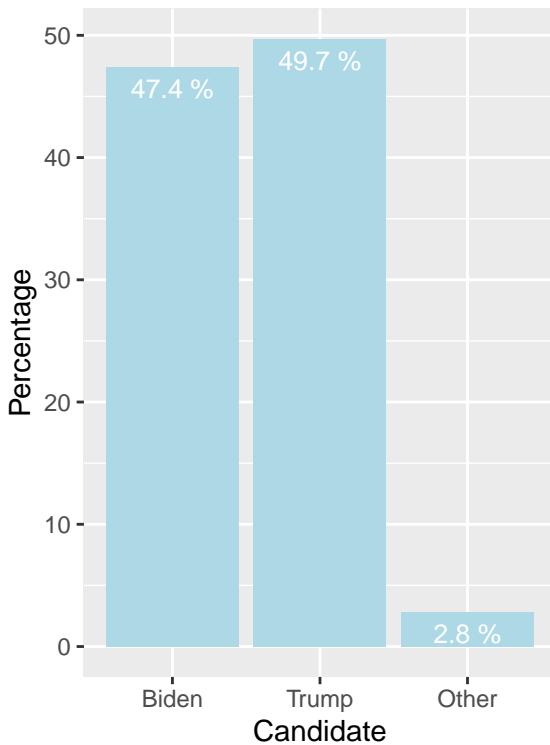
```
summary(Population$Vote)
```

```
## Biden Other Trump
## 1641882 136823 1641882
```

Results in random sample of 702 voters:

```
summary(Sample$Vote)
```

```
## Biden Other Trump
## 333    20   349
```



2.1.7 A Third Sample of 702 Voters

Let's simulate taking another third random sample of 702 voters.

```
Sample <- sample_n(Population, 702, replace=FALSE)
```

Population Summary:

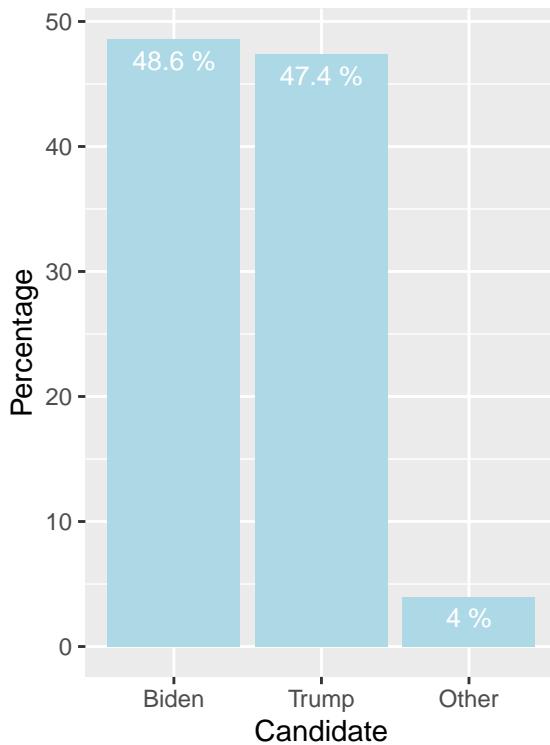
```
summary(Population$Vote)
```

```
## Biden Other Trump
## 1641882 136823 1641882
```

Results in random sample of 702 voters:

```
summary(Sample$Vote)
```

```
## Biden Other Trump
## 341     28    333
```



2.1.8 A Fourth Sample of 702 Voters

Let's simulate taking another fourth random sample of 702 voters.

```
Sample <- sample_n(Population, 702, replace=FALSE)
```

Population Summary:

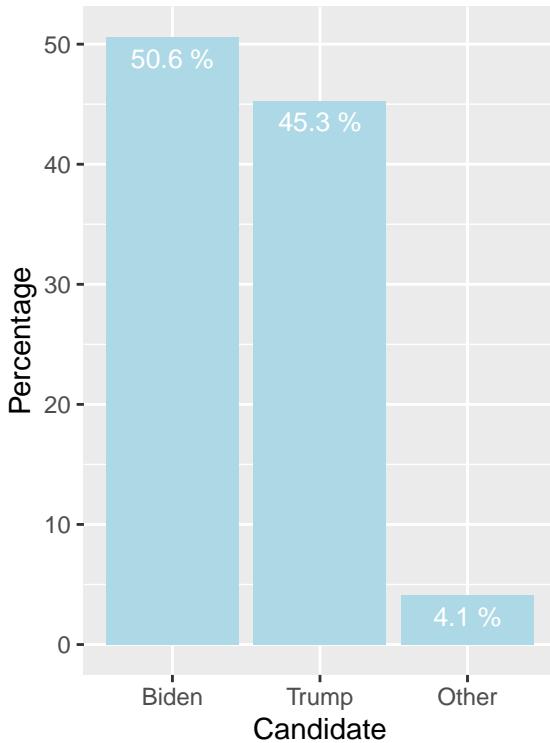
```
summary(Population$Vote)
```

```
## Biden Other Trump
## 1641882 136823 1641882
```

Results in random sample of 702 voters:

```
summary(Sample$Vote)
```

```
## Biden Other Trump
## 355 29 318
```



2.1.9 A Fifth Sample of 702 Voters

Let's simulate taking another fourth random sample of 702 voters.

```
Sample <- sample_n(Population, 702, replace=FALSE)
```

Population Summary:

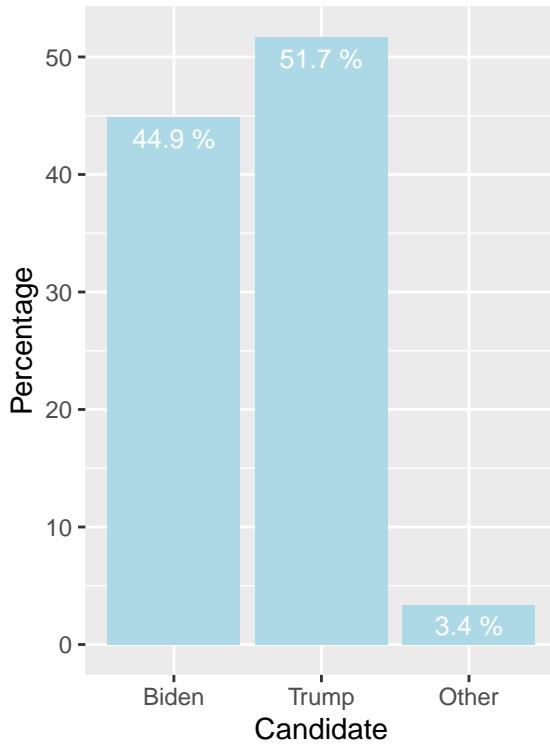
```
summary(Population$Vote)
```

```
## Biden Other Trump
## 1641882 136823 1641882
```

Results in random sample of 702 voters:

```
summary(Sample$Vote)
```

```
## Biden Other Trump
## 315     24    363
```



2.1.10 Simulating Many Polls

Let's simulate 10,000 different polls of 702 voters and look at how much variability we see in the difference in proportions supporting each candidate.

```
set.seed(09132020)

Biden <- rep(NA, 10000) #blank vector to store proportion supporting Biden
Trump <- rep(NA, 10000) #blank vector to store proportion supporting Trump
Other <- rep(NA, 10000) #blank vector to store proportion supporting Other

for (i in 1:10000){
  Sample <- sample_n(Population, 702, replace=FALSE) #sample 1000 voters
  Biden[i] <- sum(Sample == "Biden")/702 ## record number of Biden votes
  Trump[i] <- sum(Sample == "Trump")/702 ## record number of Trump votes
  Other[i] <- sum(Sample == "Other")/702 ## record number of Other votes
}

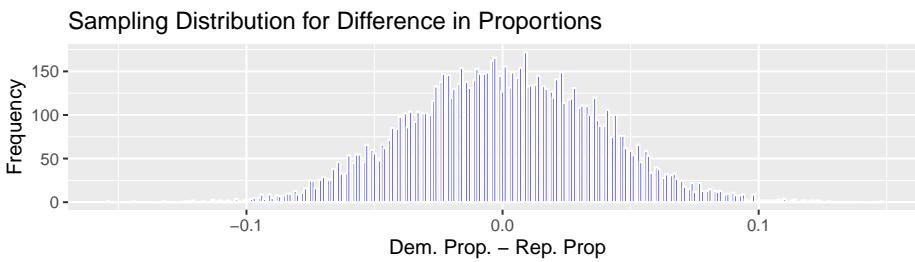
Sim <- 1:10000
PollingSim <- data.frame(Sim, Biden, Trump, Other)
PollingSim$PropDiff <- PollingSim$Biden-PollingSim$Trump
```

2.1.11 Histogram of Difference in Sample Proportions

The histogram shows the difference in proportion favoring each party's candidate in the 10,000 samples.

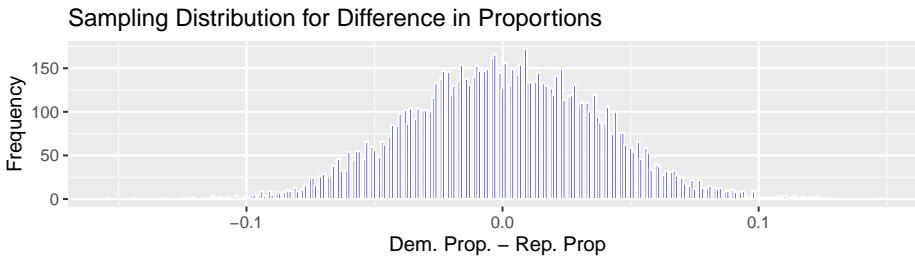
Recall that the samples were taken under the assumption that there is no difference in support between the Republican or Democrat in the population.

```
Vote_Samp_Dist <- ggplot(data=PollingSim, aes(x=PropDiff)) + geom_histogram(fill="blue", color="black", binwidth=0.005) + xlab(" Dem. Prop. - Rep. Prop") + ylab("Frequency") + ggtitle("Sampling Distribution for Difference in Proportions")
Vote_Samp_Dist
```



This distribution is called the **sampling distribution** for the difference in proportions.

2.1.12 Sampling Dist. for Diff. in Proportions



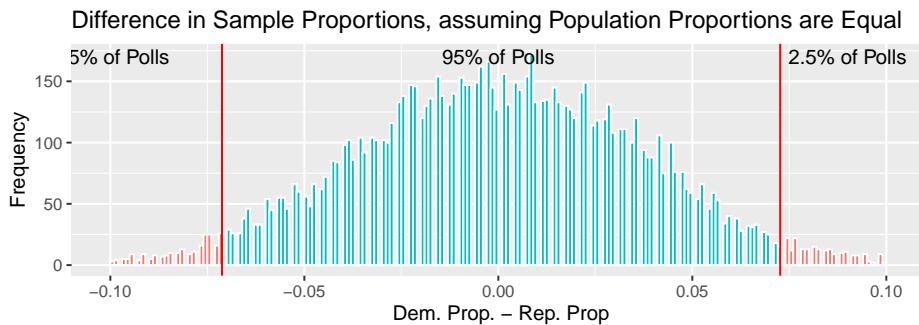
```
mean(PollingSim$PropDiff)
```

```
## [1] 0.0003595442
SE <- sd(PollingSim$PropDiff)
SE
```

```
## [1] 0.03689199
```

The sampling distribution for the difference in candidate support is approximately centered at 0, as we would expect. The standard deviation in difference between candidates is about 3.7 percentage points.

2.1.13 Difference in Sample Proportions (cont.)



```
c(quantile(PollingSim$PropDiff, .025), quantile(PollingSim$PropDiff, .975))

##           2.5%      97.5%
## -0.07122507  0.07264957
```

- Approximately 95% of the polls were within ± 0.07 of the actual difference, which in this case was assumed to be 0.

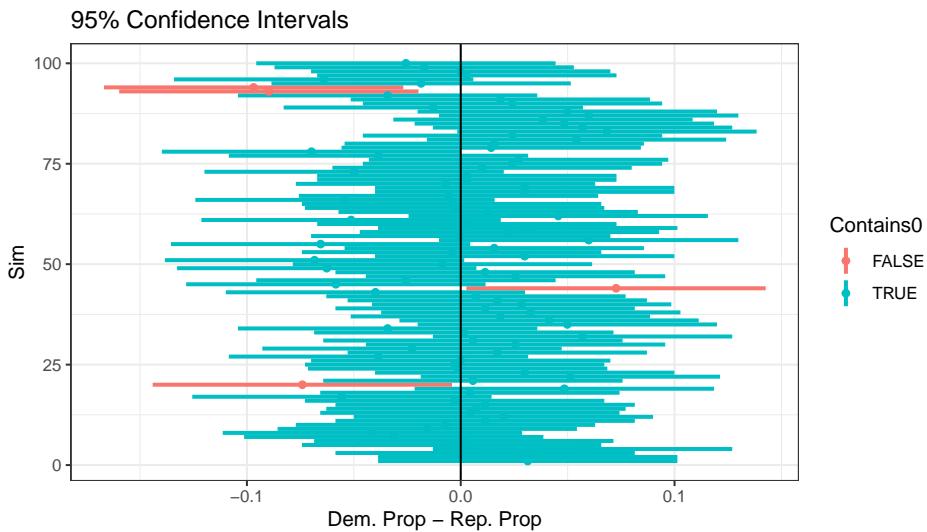
2.1.14 Conclusions from Election Polling Simulation

Since Biden is ahead by 4 percentage points in the Washington Post survey of 702 voters, it is plausible that he could really be ahead by as many as about 11 percentage points, or behind by as many as about 3.

- The ± 0.07 is called a **margin of error**, and the resulting range of plausible values is called a **confidence interval** for the difference in proportions.
- A confidence interval gives a reasonable range for a value pertaining to the entire population, using information from a sample.

2.1.15 What does “95% Confidence” Mean?

95% confidence means that if we were to repeat our procedure on many, random samples, then in the long run, 95% of them would contain the true difference in proportions supporting the candidates.



Here, we see results for 100 random samples. Notice that when we do $\text{Prop. Diff} \pm 0.07$, approximately 95% lead to a confidence interval containing 0 (the assumed true difference).

2.2 Bootstrap Sampling for Election Example

2.2.1 A More Realistic Scenario

- The prior simulation is based on the unrealistic assumption that we know the proportion of voters in the entire population favoring each candidate.
- If we really know this, there would be no need to take polls.
- In reality, we will only have the result of a single sample, and from that, we will need to determine how much our result might vary from the overall population parameter.

2.2.2 Example: Election Polling Survey

Recall the Washington Post Poll of 702 Wisconsin voters that showed 50% supporting Biden, and 46% supporting Trump.

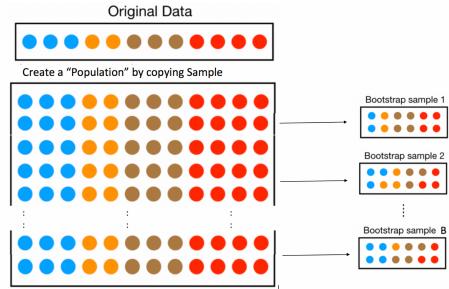
Although the poll does not release exact counts, we can infer that 351 voters picked Biden, 323 picked Trump, and 28 picked someone else.

Without assuming anything about the larger population, how can we account for sampling variability to obtain a reasonable range for the true difference in proportion of voters supporting the candidates?

2.2.3 Illustration of the Bootstrap “Idea”

Bootstrap sampling is a popular modern approach for measuring sampling variability.

The idea is to mimic sampling from a population, by copying the sample many times, and then sampling from these copies.



We then calculate the statistic of interest (i.e. difference in proportions) in each bootstrap sample to get a sense of the amount of variability between samples.

2.2.4 The Bootstrap “Idea”

1. Create many (theoretically infinitely many) copies of the original sample.
2. Take a sample of the same size as our original sample. This is known as a **bootstrap sample**. Since we have created many copies of the original sample some cases will come up multiple times and others not at all in our bootstrap sample.
3. Calculate the difference in proportions supporting each candidate in the bootstrap sample.
4. Repeat steps 2 and 3 many (say 10,000) times, keeping track of the differences in proportions supporting each candidate.
5. Look at the distribution of differences calculated from the bootstrap samples. The variability in this distribution can be used to approximate the variability in the sampling distribution for the difference in proportions.

2.2.5 Bootstrap via Resampling Illustration

Copying the original sample infinitely many times, and then sampling from these copies is equivalent to simply sampling from the original sample, using replacement. This is how bootstrap sampling is done in practice.

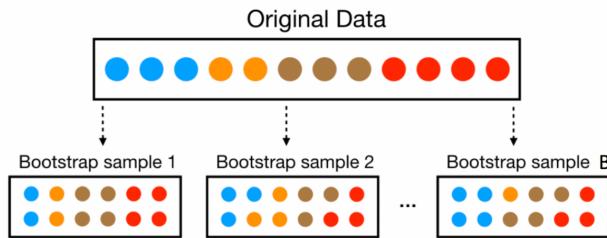


Image from <https://bradleyboehmke.github.io/HOML/process.html>

2.2.6 Bootstrap Sampling with Replacement

In practice, the bootstrap is performed by resampling from the original data, with replacement.

Procedure:

1. Take a sample of the same size as our original sample, by sampling cases from the original sample, with replacement. Since we are sampling with replacement, some cases will come up multiple times and others not at all in our bootstrap sample.
2. Calculate the difference in proportions supporting each candidate in the bootstrap sample.
3. Repeat steps 2 and 3 many (say 10,000) times, keeping track of the differences in proportions supporting each candidate.
4. Look at the distribution of differences calculated from the bootstrap samples. The variability in this distribution can be used to approximate the variability in the sampling distribution for the difference in proportions.

2.2.7 Code for Bootstrapping for Difference In Proportions

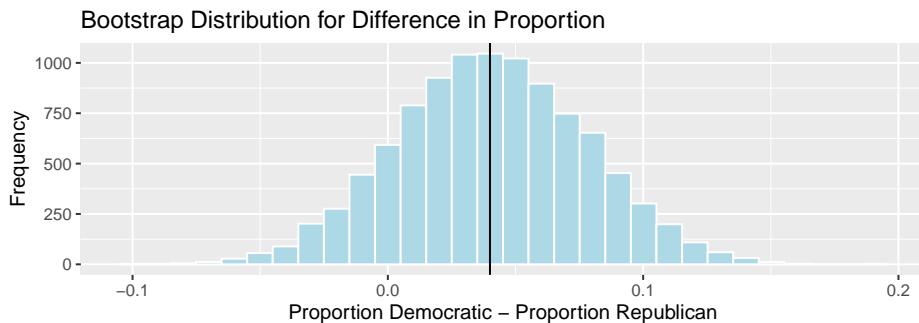
```

Vote <- c(rep("Biden", 351), rep("Trump", 323), rep("Other", 28))
Sample <- data.frame(Vote)

Difference <- rep(NA, 10000)
for (i in 1:10000){
  BootstrapSample <- sample_n(Sample, 702, replace=TRUE)
  Difference[i] <- (sum(BootstrapSample$Vote=="Biden") - sum(BootstrapSample$Vote=="Trump"))/702
}
Vote_BootstrapResults <- data.frame(Difference)

```

2.2.8 Bootstrap Dist. for Diff. in Proportion



- Note that the bootstrap distribution is centered at approximately 0.04, which was the observed difference we saw in our sample.
- Individual sample differences seem to deviate by up to about ± 0.07 .

Bootstrap standard error:

```
sd(Vote_BootstrapResults$Difference)
```

```
## [1] 0.03700877
```

2.2.9 Bootstrap Percentile Confidence Interval

An approximate 95% percentile confidence interval for the difference in support is found by finding the 0.025 and 0.975 quantiles of the bootstrap distribution, since 95% of all bootstrap samples produce statistics between these values.

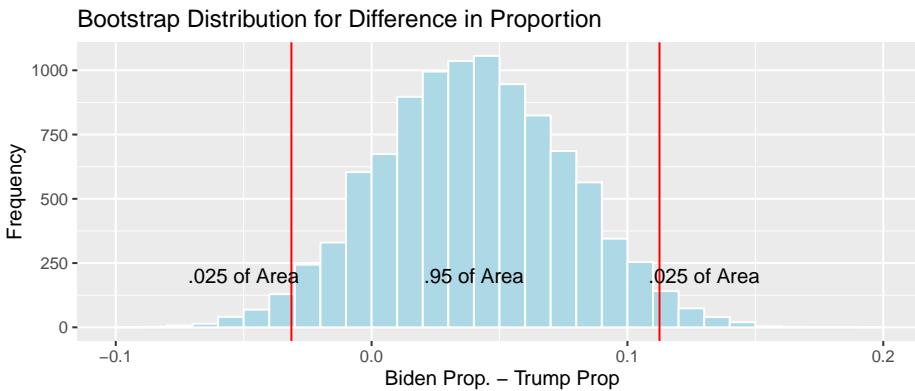
```
q.025 <- quantile(Vote_BootstrapResults$Difference, 0.025)
q.975 <- quantile(Vote_BootstrapResults$Difference, 0.975)

c(q.025, q.975)
```

```
##           2.5%      97.5%
## -0.03133903  0.11253561
```

We can be 95% confident that Biden is between 11.3 percentage points ahead, and 3.1 percentage points behind.

2.2.10 95% Bootstrap Percentile Confidence Interval Visually



We can be 95% confident that Biden is between 11.3 percentage points ahead, and 3.1 percentage points behind.

2.2.11 Bootstrap Standard Error

The **bootstrap standard error** is the standard deviation of the bootstrap distribution. It measures the amount of variability in a statistic (in this case difference in proportions) between samples of the given size.

```
sd(Vote_BootstrapResults$Difference)
```

```
## [1] 0.03700877
```

If large number of polls of 702 voters were taken, the standard deviation in the distribution of the difference in support between Biden and Trump is estimated to be 0.0370088

2.2.12 Standard Error Confidence Intervals

- Standard error is the standard deviation of the distribution of a statistic, across many samples of the given size.
- When the sampling distribution of a statistic is symmetric and bell-shaped, then an approximately 95% of all samples will lead to a statistic that is within 2 standard errors of the desired population quantity.
- An approximate 95% bootstrap standard error formula is given by:

$$\text{Statistic} \pm 2 \times \text{Standard Error}$$

- it is only appropriate to use the bootstrap standard error confidence interval method when a sampling distribution is symmetric and bell-shaped

2.2.13 Bootstrap Standard Error Confidence Interval

```
SE <- sd(Vote_BootstrapResults$Difference)
c(0.04 - 2*SE, 0.04 + 2*SE)

## [1] -0.03401753 0.11401753
```

We can be 95% confident that Biden is between 11.4 percentage points ahead, and 3.4 percentage points behind.

The bootstrap percentile confidence interval and bootstrap standard error confidence intervals largely agree, which we expect to happen when the bootstrap distribution is symmetric and bell-shaped.

2.2.14 Comparison of Bootstrap Confidence Intervals

- A 95% bootstrap percentile confidence interval is determined by finding the 0.025 and 0.975 quantiles of the bootstrap distribution (i.e. by taking the middle 95% of the distribution).
- A 95% standard error confidence interval is determined using

$$\text{Statistic} \pm 2 \times \text{Standard Error}$$

When the bootstrap distribution is symmetric and bell-shaped, these intervals will be approximately the same.

2.2.15 90% Bootstrap Percentile Confidence Interval

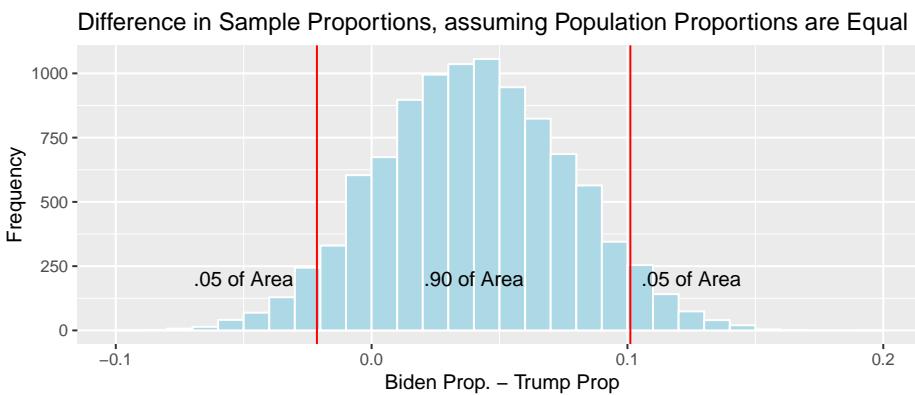
```
q.05 <- quantile(Vote_BootstrapResults$Difference, 0.05)
q.95 <- quantile(Vote_BootstrapResults$Difference, 0.95)

c(q.05, q.95)
```

```
##           5%          95%
## -0.02136752 0.10113960
```

We can be 90% confident that Biden is between 10 percentage points ahead, and 2 percentage points behind.

2.2.16 90% Bootstrap Confidence Interval Visually



We can be 90% confident that Biden is between 10 percentage points ahead, and 2 percentage points behind.

2.2.17 Comments on Election Polling

- In reality, it is very hard to obtain a “random” sample of voters. Voters can be hard to reach, and many people do not respond to polls. Often it is people with the strongest opinions who do. Furthermore, some demographic groups are harder to contact than others, and these groups are likely to be underrepresented in surveys.
- In order to account for sampling challenges, pollsters tend to weight responses so that their proportions match those of registered voters. For example, if fewer college students respond to the survey, then those that do are weighted more heavily to make up for it.
- Weighting helps offset underrepresentation, but creates additional challenges. It is not always clear what weights to use. This is especially tricky for election polling, since the demographic breakdown of voters who turn out differs from election to election.
- Because of these challenges, polling involves a “practical” margin of error, in addition to the “mathematical” margin of error.
- Political analysts such as those at [fivethirtyeight.com] (<https://fivethirtyeight.com/>), average results from many different polls to effectively obtain larger samples. This reduces the “mathematical” margin of error, but the “practical” margin of error remains.
- Historically, election polling in the United States has been fairly accurate (usually within ± 5 percentage points of the polling average) in major

national or statewide elections. Polling has proven more challenging and sometimes resulted in larger errors in the UK and other countries.

2.3 Bootstrapping Distribution of Sample Mean

2.3.1 General Bootstrapping Procedure

The bootstrap procedure can be applied to quantify uncertainty associated with a wide range of statistics (for example, sample proportions, means, medians, standard deviations, regression coefficients, F-statistics, etc.)

Given a statistic that was calculated from a sample...

Procedure:

1. Take a sample of the same size as the original sample, by sampling cases from the original sample, with replacement.
2. Calculate the statistic of interest in the bootstrap sample.
3. Repeat steps 2 and 3 many (say 10,000) times, keeping track of the statistic calculated in each bootstrap sample.
4. Look at the distribution of statistics calculated from the bootstrap samples. The variability in this distribution can be used to approximate the variability in the sampling distribution for the statistic of interest.

2.3.2 Bootstrap Illustration

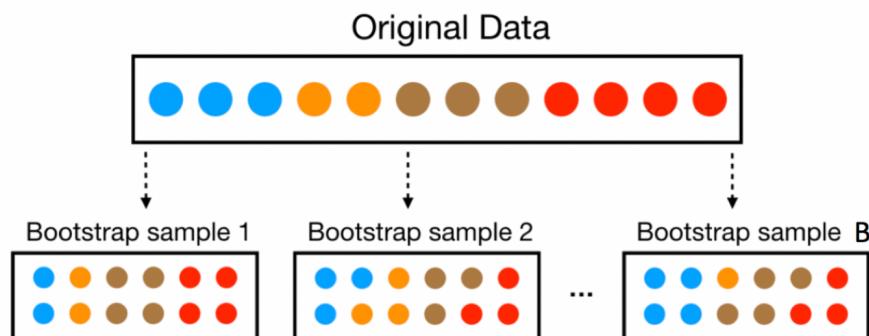


Image from <https://bradleyboehmke.github.io/HOML/process.html>

2.3.3 Mercury Levels in Florida Lakes

A 2004 study by Lange, T., Royals, H. and Connor, L. examined Mercury accumulation in large-mouth bass, taken from a sample of 53 Florida Lakes. If Mercury accumulation exceeds 0.5 ppm, then there are environmental concerns. In fact, the legal safety limit in Canada is 0.5 ppm, although it is 1 ppm in the United States.



Figure 2.1: <https://www.maine.gov/ifw/fish-wildlife/fisheries/species-information/largemouth-bass.html>

2.3.4 Florida Lakes Dataset

```
data("FloridaLakes")
glimpse(FloridaLakes)

## # Rows: 53
## # Columns: 12
## $ ID                  <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1-
## $ Lake                <chr> "Alligator", "Annie", "Apopka", "Blue Cypress", "Bri-
## $ Alkalinity          <dbl> 5.9, 3.5, 116.0, 39.4, 2.5, 19.6, 5.2, 71.4, 26.4, 4-
## $ pH                  <dbl> 6.1, 5.1, 9.1, 6.9, 4.6, 7.3, 5.4, 8.1, 5.8, 6.4, 5.-
## $ Calcium              <dbl> 3.0, 1.9, 44.1, 16.4, 2.9, 4.5, 2.8, 55.2, 9.2, 4.6, ~
## $ Chlorophyll         <dbl> 0.7, 3.2, 128.3, 3.5, 1.8, 44.1, 3.4, 33.7, 1.6, 22.-
## $ AvgMercury           <dbl> 1.23, 1.33, 0.04, 0.44, 1.20, 0.27, 0.48, 0.19, 0.83-
## $ NumSamples            <int> 5, 7, 6, 12, 12, 14, 10, 12, 24, 12, 12, 12, 7, 43, ~
## $ MinMercury            <dbl> 0.85, 0.92, 0.04, 0.13, 0.69, 0.04, 0.30, 0.08, 0.26-
## $ MaxMercury            <dbl> 1.43, 1.90, 0.06, 0.84, 1.50, 0.48, 0.72, 0.38, 1.40-
## $ ThreeYrStdMercury    <dbl> 1.53, 1.33, 0.04, 0.44, 1.33, 0.25, 0.45, 0.16, 0.72-
## $ AgeData               <int> 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1-
```

2.3.5 Mercury Level in Sample of 53 Florida Lakes

```
##             Lake AvgMercury
## Alligator      1.23
## Annie          1.33
```

```

##          Apopka      0.04
##      Blue Cypress    0.44
##          Brick       1.20
##          Bryant      0.27
##          Cherry       0.48
##          Crescent     0.19
##      Deer Point      0.83
##          Dias         0.81
##          Dorr         0.71
##          Down         0.50
##          Eaton         0.49
##  East Tohopekaliga   1.16
##          Farm-13      0.05
##          George        0.15
##          Griffin       0.19
##          Harney        0.77
##          Hart         1.08
##          Hatchineha    0.98

##          Lake AvgMercury
##      Iamonia      0.63
##      Istokpoga    0.56
##      Jackson      0.41
##      Josephine     0.73
##      Kingsley      0.34
##      Kissimmee     0.59
##      Lochloosa     0.34
##      Louisa        0.84
##      Miccasukee    0.50
##      Minneola      0.34
##      Monroe        0.28
##      Newmans       0.34
##      Ocean Pond    0.87
##  Ocheese Pond     0.56
##      Okeechobee    0.17
##      Orange        0.18
##      Panasoffkee   0.19
##      Parker        0.04
##      Placid        0.49
##      Puzzle        1.10

##          Lake AvgMercury
##      Rodman       0.16
##      Rousseau     0.10
##      Sampson      0.48
##      Shipp         0.21
##      Talquin      0.86

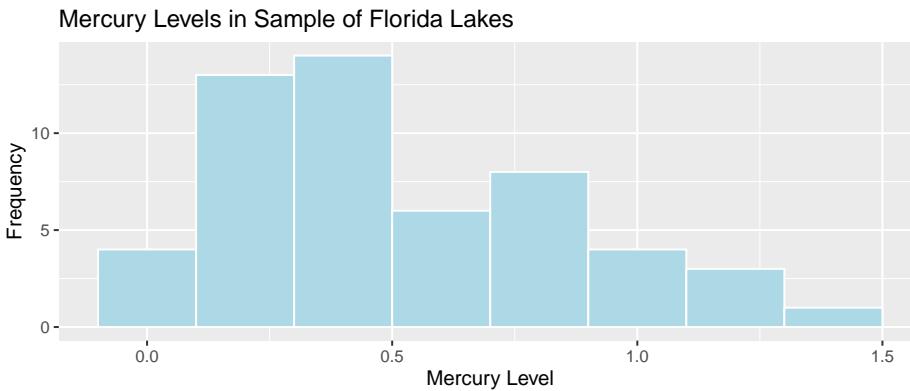
```

```

##      Tarpon      0.52
##  Tohopekaliga  0.65
##    Trafford    0.27
##      Trout     0.94
##  Tsala Apopka  0.40
##      Weir      0.43
##    Wildcat    0.25
##      Yale      0.27

```

2.3.6 Distribution of Mercury Levels



```

## # A tibble: 1 x 5
##   MeanHg MedianHg StDevHG PropOver1     N
##     <dbl>     <dbl>     <dbl>     <dbl> <int>
## 1  0.527     0.48     0.341     0.113     53

```

2.3.7 Bootstrapping for Mean Mercury Level in All Lakes

In our sample of 53 lakes, the mean mercury level is 0.527 ppm. However, this was just a random sample of lakes. We are interested in a confidence interval for the mean mercury level for all Florida Lakes.

Bootstrapping Procedure

1. Take a bootstrap sample of size 53, by sampling lakes with replacement.
2. Calculate the mean mercury concentration in the bootstrap sample.
3. Repeat steps 2 and 3 10,000 times, keeping track of the mean mercury concentrations in each bootstrap sample.
4. Look at the distribution of mean concentrations from the bootstrap samples. The variability in this distribution can be used to approximate the variability in the sampling distribution for the sample mean.

2.3.8 Original Sample

```

##          Lake AvgMercury
##      Alligator      1.23
##      Annie        1.33
##      Apopka       0.04
##      Blue Cypress   0.44
##      Brick         1.20
##      Bryant        0.27
##      Cherry        0.48
##      Crescent      0.19
##      Deer Point    0.83
##      Dias           0.81
##      Dorr           0.71
##      Down           0.50
##      Eaton          0.49
##  East Tohopekaliga 1.16
##      Farm-13       0.05
##      George        0.15
##      Griffin       0.19
##      Harney        0.77
##      Hart          1.08
##      Hatchineha    0.98

##          Lake AvgMercury
##      Iamonia       0.63
##      Istokpoga     0.56
##      Jackson       0.41
##      Josephine     0.73
##      Kingsley      0.34
##      Kissimmee     0.59
##      Lochloosa      0.34
##      Louisa        0.84
##      Miccasukee    0.50
##      Minneola       0.34
##      Monroe         0.28
##      Newmans        0.34
##      Ocean Pond     0.87
##  Ocheese Pond     0.56
##      Okeechobee     0.17
##      Orange         0.18
##      Panasoffkee    0.19
##      Parker         0.04
##      Placid         0.49
##      Puzzle         1.10

```

```

##          Lake AvgMercury
##    Rodman      0.16
##  Rousseau      0.10
##   Sampson      0.48
##     Shipp      0.21
##   Talquin      0.86
##    Tarpon      0.52
## Tohopekaliga   0.65
##   Trafford     0.27
##     Trout      0.94
## Tsala Apopka   0.40
##      Weir      0.43
##   Wildcat      0.25
##      Yale      0.27

mean(FloridaLakes$AvgMercury)

```

```
## [1] 0.5271698
```

2.3.9 Five Bootstrap Samples

The `sample_n()` function samples the specified number rows from a data frame, with or without replacement.

```

BootstrapSample1 <- sample_n(FloridaLakes, 53, replace=TRUE) %>% arrange(Lake)

BootstrapSample2 <- sample_n(FloridaLakes, 53, replace=TRUE) %>% arrange(Lake)

BootstrapSample3 <- sample_n(FloridaLakes, 53, replace=TRUE) %>% arrange(Lake)

BootstrapSample4 <- sample_n(FloridaLakes, 53, replace=TRUE) %>% arrange(Lake)

BootstrapSample5 <- sample_n(FloridaLakes, 53, replace=TRUE) %>% arrange(Lake)

```

2.3.10 Bootstrap Sample 1

```

##          Lake AvgMercury
## Alligator      1.23
##    Annie      1.33
##   Apopka      0.04
##   Bryant      0.27
##   Bryant      0.27
##   Cherry      0.48
##   Cherry      0.48
## Deer Point    0.83
## Deer Point    0.83

```

```

##          Dias      0.81
##          Dias      0.81
##    Farm-13     0.05
##    Farm-13     0.05
##    Farm-13     0.05
##    Griffin     0.19
##      Hart     1.08
##  Hatchineha   0.98
##  Hatchineha   0.98
##    Iamonia    0.63
##  Istokpoga    0.56

##      Lake AvgMercury
## Josephine    0.73
## Josephine    0.73
##  Kingsley    0.34
##  Kingsley    0.34
## Kissimmee    0.59
## Kissimmee    0.59
##  Lochloosa    0.34
## Miccasukee   0.50
##  Newmans     0.34
##  Newmans     0.34
##  Newmans     0.34
## Okeechobee   0.17
## Okeechobee   0.17
##      Parker    0.04
##      Parker    0.04
##    Placid     0.49
##    Rodman     0.16
##  Rousseau    0.10
##    Sampson    0.48
##    Sampson    0.48

##      Lake AvgMercury
##    Sampson    0.48
##    Talquin    0.86
##    Talquin    0.86
##    Talquin    0.86
##      Tarpon    0.52
##      Trout     0.94
##      Trout     0.94
##      Trout     0.94
## Tsala Apopka 0.40
##    Wildcat    0.25
##    Wildcat    0.25
##    Wildcat    0.25

```

```
##          Yale      0.27
mean(BootstrapSample1$AvgMercury)

## [1] 0.5109434
```

2.3.11 Bootstrap Sample 2

```
##          Lake AvgMercury
##          Annie     1.33
##          Apopka    0.04
##          Brick     1.20
##          Cherry    0.48
##          Cherry    0.48
##          Cherry    0.48
##          Crescent   0.19
##          Deer Point 0.83
##          Dorr       0.71
##          Down      0.50
##          East Tohopekaliga 1.16
##          East Tohopekaliga 1.16
##          Eaton      0.49
##          Farm-13    0.05
##          George     0.15
##          Griffin    0.19
##          Griffin    0.19
##          Iamonia    0.63
##          Iamonia    0.63
##          Istokpoga   0.56

##          Lake AvgMercury
##          Josephine  0.73
##          Kissimmee   0.59
##          Kissimmee   0.59
##          Louisa     0.84
##          Miccasulkee 0.50
##          Minneola    0.34
##          Ocean Pond  0.87
##          Okeechobee   0.17
##          Okeechobee   0.17
##          Orange      0.18
##          Orange      0.18
##          Orange      0.18
##          Orange      0.18
##          Panasoffkee 0.19
##          Panasoffkee 0.19
```

```

##   Panasoffkee      0.19
##   Puzzle          1.10
##   Puzzle          1.10
##   Puzzle          1.10
##   Rodman          0.16

##       Lake AvgMercury
##   Rodman          0.16
##   Rodman          0.16
##   Shipp           0.21
##   Talquin         0.86
##   Talquin         0.86
##   Talquin         0.86
##   Tarpon           0.52
##   Tohopekaliga    0.65
##   Trafford         0.27
##   Trout            0.94
##   Weir             0.43
##   Weir             0.43
##   Yale             0.27

mean(BootstrapSample2$AvgMercury)

```

```
## [1] 0.5211321
```

2.3.12 Bootstrap Sample 3

```

##       Lake AvgMercury
##   Annie        1.33
##   Annie        1.33
##   Annie        1.33
##   Annie        1.33
##   Apopka       0.04
##   Apopka       0.04
##   Blue Cypress 0.44
##   Blue Cypress 0.44
##   Blue Cypress 0.44
##   Blue Cypress 0.44
##   Cherry       0.48
##   Down         0.50
##   Eaton        0.49
##   Eaton        0.49
##   George       0.15
##   Hart         1.08
##   Hart         1.08
##   Hatchineha  0.98

```

```
##      Iamonia      0.63
##      Jackson      0.41

##      Lake AvgMercury
##      Kissimmee    0.59
##      Louisa       0.84
##      Miccasukee   0.50
##      Minneola     0.34
##      Newmans      0.34
##      Ocean Pond   0.87
##      Ocean Pond   0.87
##      Ocheese Pond  0.56
##      Ocheese Pond  0.56
##      Orange        0.18
##      Orange        0.18
##      Panasoffkee   0.19
##      Rodman        0.16
##      Rodman        0.16
##      Rousseau      0.10
##      Sampson       0.48
##      Sampson       0.48
##      Shipp          0.21
##      Talquin       0.86
##      Tarpon         0.52

##      Lake AvgMercury
##      Trafford     0.27
##      Trafford     0.27
##      Trafford     0.27
##      Trout         0.94
##      Tsala Apopka  0.40
##      Weir          0.43
##      Wildcat       0.25
##      Wildcat       0.25
##      Wildcat       0.25
##      Yale           0.27
##      Yale           0.27
##      Yale           0.27
##      Yale           0.27

mean(BootstrapSample3$AvgMercury)
```

```
## [1] 0.5066038
```

2.3.13 Bootstrap Sample 4

```

##          Lake AvgMercury
##  Alligator      1.23
##  Alligator      1.23
##    Annie        1.33
##    Annie        1.33
##   Apopka        0.04
##   Bryant        0.27
##   Cherry        0.48
##  Crescent       0.19
## Deer Point     0.83
##     Down        0.50
## Farm-13         0.05
##   George        0.15
##  Griffin        0.19
##   Harney        0.77
##     Hart        1.08
## Hatchineha     0.98
## Hatchineha     0.98
## Hatchineha     0.98
## Istokpoga      0.56
##   Jackson       0.41

##          Lake AvgMercury
##  Kingsley       0.34
## Miccasukee     0.50
## Miccasukee     0.50
##  Minneola       0.34
##  Minneola       0.34
##  Minneola       0.34
##    Monroe        0.28
##  Newmans        0.34
## Ocean Pond     0.87
## Ocean Pond     0.87
## Okeechobee     0.17
##    Orange        0.18
##    Orange        0.18
##    Orange        0.18
## Panasoffkee    0.19
##    Placid        0.49
##    Placid        0.49
##    Placid        0.49
##   Puzzle         1.10
##  Rousseau       0.10

```

```

##          Lake AvgMercury
##    Rousseau      0.10
##    Sampson       0.48
##    Shipp         0.21
##    Talquin       0.86
##    Tarpon        0.52
##    Tarpon        0.52
##    Tohopekaliga  0.65
##    Trafford      0.27
##    Trafford      0.27
##    Weir          0.43
##    Wildcat       0.25
##    Yale          0.27
##    Yale          0.27

mean(BootstrapSample4$AvgMercury)

```

```
## [1] 0.5088679
```

2.3.14 Bootstrap Sample 5

```

##          Lake AvgMercury
##    Alligator     1.23
##    Alligator     1.23
##    Annie         1.33
##    Apopka        0.04
##    Brick          1.20
##    Bryant        0.27
##    Cherry        0.48
##    Down          0.50
##    Down          0.50
##    East Tohopekaliga 1.16
##    Eaton          0.49
##    Farm-13        0.05
##    Farm-13        0.05
##    Farm-13        0.05
##    Harney         0.77
##    Harney         0.77
##    Hart           1.08
##    Hatchineha    0.98
##    Lochloosa      0.34
##    Lochloosa      0.34

##          Lake AvgMercury
##    Louisa        0.84
##    Miccasukee    0.50

```

```

##   Miccasukee      0.50
##   Minneola        0.34
##   Monroe          0.28
##   Monroe          0.28
##   Monroe          0.28
##   Monroe          0.28
##   Newmans         0.34
##   Newmans         0.34
##   Newmans         0.34
##   Okeechobee      0.17
##   Orange          0.18
##   Orange          0.18
##   Placid          0.49
##   Placid          0.49
##   Puzzle           1.10
##   Puzzle           1.10
##   Rodman          0.16
##   Rousseau         0.10

##             Lake AvgMercury
##   Shipp        0.21
##   Shipp        0.21
##   Talquin      0.86
##   Talquin      0.86
##   Tarpon        0.52
##   Trafford      0.27
##   Trafford      0.27
##   Trafford      0.27
##   Trafford      0.27
##   Tsala Apopka 0.40
##   Weir          0.43
##   Weir          0.43
##   Wildcat       0.25

mean(BootstrapSample5$AvgMercury)

## [1] 0.4981132

```

2.3.15 Code for Florida Lakes Bootstrap for Sample Mean

```

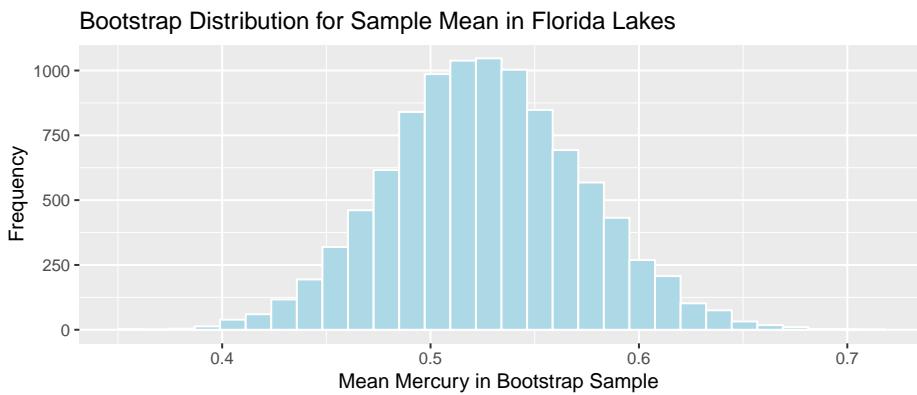
MeanHg <- rep(NA, 10000)
for (i in 1:10000){
  BootstrapSample <- sample_n(FloridaLakes, 53, replace=TRUE)
  MeanHg[i] <- mean(BootstrapSample$AvgMercury)
}

```

```
Lakes_Bootstrap_Results_Mean <- data.frame(MeanHg)
```

2.3.16 Florida Lakes Bootstrap Distribution for Mean

```
Lakes_Bootstrap_Mean <- ggplot(data=Lakes_Bootstrap_Results_Mean, aes(x=MeanHg)) +
  geom_histogram(color="white", fill="lightblue") +
  xlab("Mean Mercury in Bootstrap Sample ") + ylab("Frequency") +
  ggtitle("Bootstrap Distribution for Sample Mean in Florida Lakes") +
  theme(legend.position = "none")
Lakes_Bootstrap_Mean
```

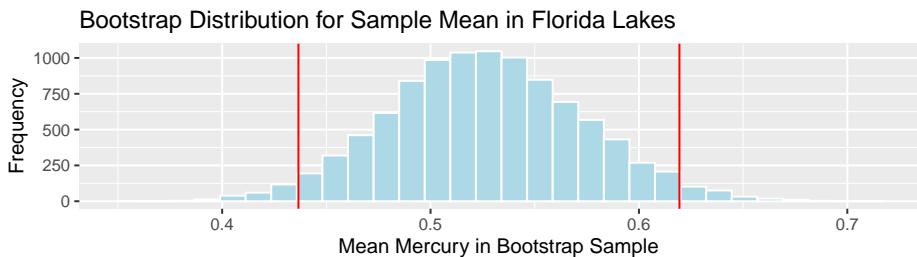


2.3.17 Bootstrap Percentile Interval for Mean Hg Level

```
q.025 <- quantile(Lakes_Bootstrap_Results_Mean$MeanHg, 0.025)
q.975 <- quantile(Lakes_Bootstrap_Results_Mean$MeanHg, 0.975)
c(q.025, q.975)

##      2.5%    97.5%
## 0.4366038 0.6194340

Lakes_Bootstrap_Mean + geom_vline(xintercept=c(q.025,q.975), color="red")
```



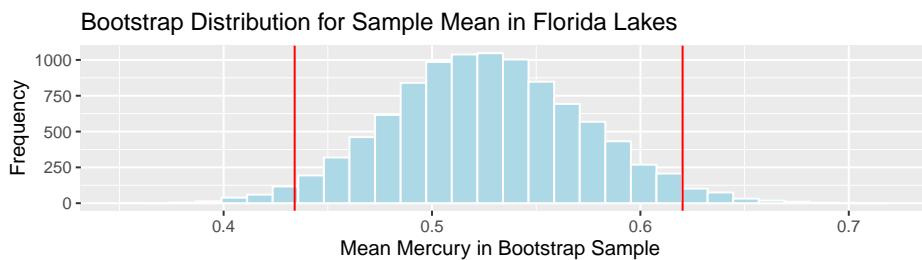
We are 95% confident that the mean mercury level in all Florida Lakes is between 0.44 and 0.62 ppm.

2.3.18 Bootstrap SE Interval for Mean Hg Level

```
SampMean <- mean(FloridaLakes$AvgMercury)
SE <- sd(Lakes_Bootstrap_Results_Mean$MeanHg)
c(SampMean-2*SE, SampMean+2*SE)
```

```
## [1] 0.4340912 0.6202485
```

```
Lakes_Bootstrap_Mean + geom_vline(xintercept=c(c(SampMean-2*SE, SampMean+2*SE)), color=
```



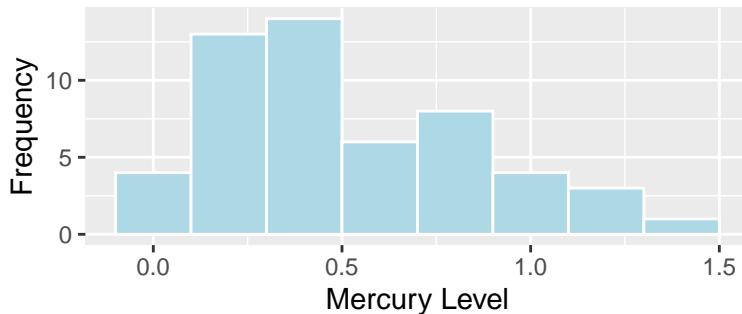
We are 95% confident that the mean mercury level in all Florida Lakes is between 0.43 and 0.62 ppm.

Again, the percentile and standard error intervals closely agree.

2.3.19 Standard Deviation and Standard Error

- Standard error is the standard deviation of the distribution of a statistic, across many samples of the given size. This is different than the sample standard deviation, which pertains to the amount of variability between individuals in the sample

Mercury Levels in Sample of Florida Lakes



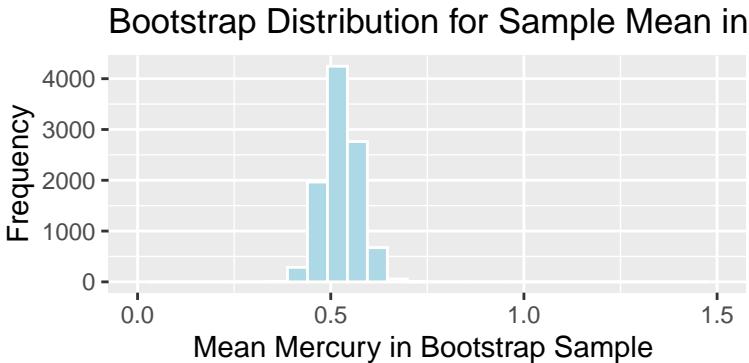
2.4. BOOTSTRAPPING DISTRIBUTION FOR STATISTICS OTHER THAN MEAN107

Standard Deviation:

```
sd(FloridaLakes$AvgMercury)
```

```
## [1] 0.3410356
```

The standard deviation in mercury levels between individual lakes is 0.341 ppm.



Standard Error for Mean:

```
SE <- sd(Lakes_Bootstrap_Results_Mean$MeanHg); SE
```

```
## [1] 0.04653932
```

The standard deviation in the distribution for mean mercury levels between different samples of 53 lakes is approximately 0.0465393 ppm.

2.4 Bootstrapping Distribution for Statistics other than Mean

2.4.1 Other Quantities for Florida Lakes

We might be interested in quantities other than mean mercury level for the lakes.

For example:

- standard deviation in mercury level

```
sd(FloridaLakes$AvgMercury)
```

```
## [1] 0.3410356
```

- percentage of lakes with mercury level exceeding 1 ppm

```
mean(FloridaLakes$AvgMercury>1)
```

```
## [1] 0.1132075
```

- median mercury level

```
median(FloridaLakes$AvgMercury)
```

```
## [1] 0.48
```

2.4.2 Bootstrapping for Other Quantities

We can also use bootstrapping to obtain confidence intervals for the median and standard deviation in mercury levels in Florida lakes.

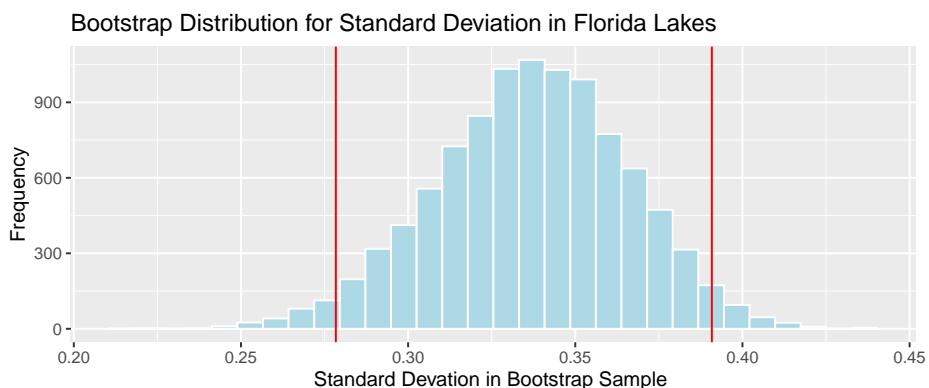
```
StDevHg <- rep(NA, 10000)
PropOver1 <- rep(NA, 10000)
MedianHg <- rep(NA, 10000)

for (i in 1:10000){
  BootstrapSample <- sample_n(FloridaLakes, 53, replace=TRUE)
  StDevHg[i] <- sd(BootstrapSample$AvgMercury)
  PropOver1[i] <- mean(BootstrapSample$AvgMercury>1)
  MedianHg[i] <- median(BootstrapSample$AvgMercury)
}
Lakes_Bootstrap_Results_Other <- data.frame(MedianHg, PropOver1, StDevHg)
```

2.4.3 Lakes Bootstrap Percentile CI for St. Dev.

```
q.025 <- quantile(Lakes_Bootstrap_Results_Other$StDevHg, 0.025)
q.975 <- quantile(Lakes_Bootstrap_Results_Other$StDevHg, 0.975)
c(q.025, q.975)
```

```
##      2.5%    97.5%
## 0.2783774 0.3908507
```



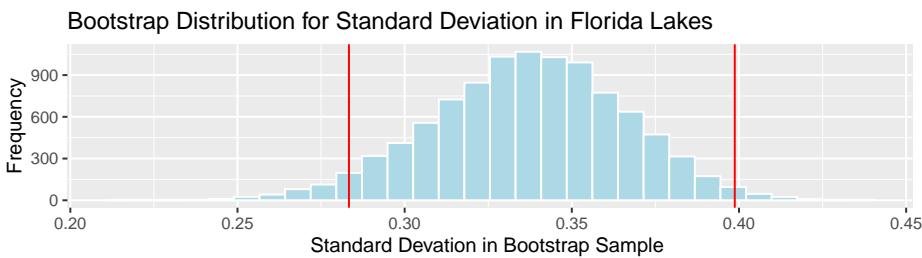
2.4. BOOTSTRAPPING DISTRIBUTION FOR STATISTICS OTHER THAN MEAN109

We are 95% confident that the standard deviation in mercury level for all Florida Lakes is between 0.28 and 0.39 ppm.

2.4.4 Bootstrap SE Interval for St.Dev. in Hg Level

```
SampSD <- sd(FloridaLakes$AvgMercury)
SE <- sd(Lakes_Bootstrap_Results_Other$StDevHg)
c(SampSD-2*SE, SampSD+2*SE)

## [1] 0.2833343 0.3987369
Lakes_Bootstrap_SD + geom_vline(xintercept=c(c(SampSD-2*SE, SampSD+2*SE)), color="red")
```



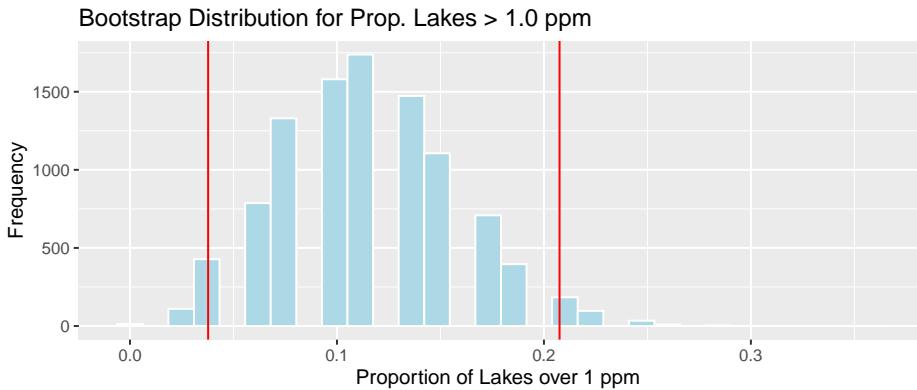
We are 95% confident that the standard deviation in mercury level for all Florida Lakes is between 0.28 and 0.39 ppm.

Again, the percentile and standard error intervals closely agree.

2.4.5 Bootstrap Percentile CI for Prop. > 1 ppm

```
q.025 <- quantile(Lakes_Bootstrap_Results_Other$PropOver1, 0.025)
q.975 <- quantile(Lakes_Bootstrap_Results_Other$PropOver1, 0.975)
c(q.025, q.975)

##      2.5%      97.5%
## 0.03773585 0.20754717
```

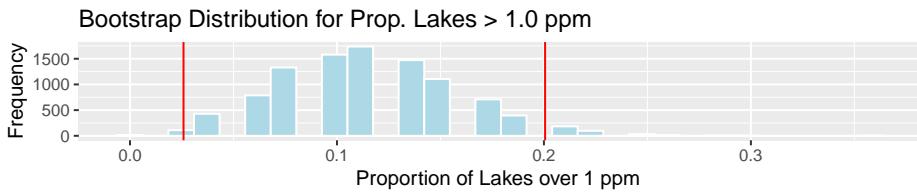


We are 95% confident that the proportion of all Florida lakes with mercury level over 1 ppm is between 0.04 and 0.21.

2.4.6 Bootstrap SE Interval for Prop > 1 ppm

```
PropOver1 <- mean(FloridaLakes$AvgMercury>1)
SE <- sd(Lakes_Bootstrap_Results_Other$PropOver1)
c(PropOver1-2*SE, PropOver1+2*SE)
```

```
## [1] 0.02585519 0.20055991
Lakes_Bootstrap_PropOver1 + geom_vline(xintercept=c(PropOver1-2*SE, PropOver1+2*SE))
```

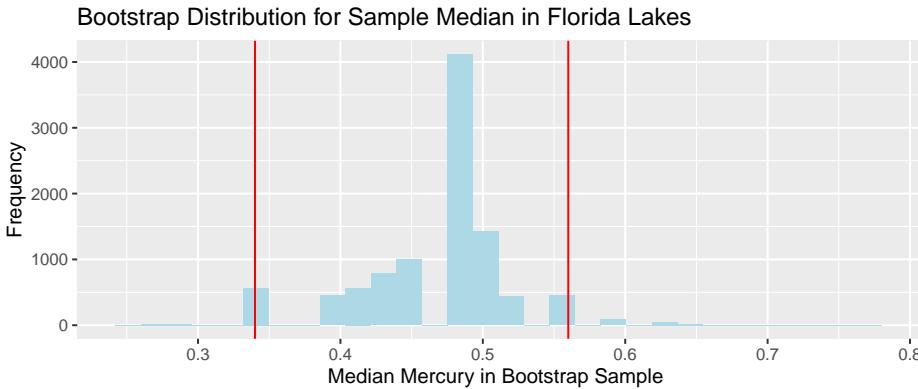


We are 95% confident that the standard deviation in mercury level in all Florida Lakes is between 0.03 and 0.2 ppm.

There are small differences between the bootstrap percentile interval and bootstrap standard error intervals. This is due to slight right-skewness in the bootstrap situations. When this happens, the bootstrap percentile interval is usually more reliable.

2.4. BOOTSTRAPPING DISTRIBUTION FOR STATISTICS OTHER THAN MEAN111

2.4.7 Bootstrap Percentile CI for Median



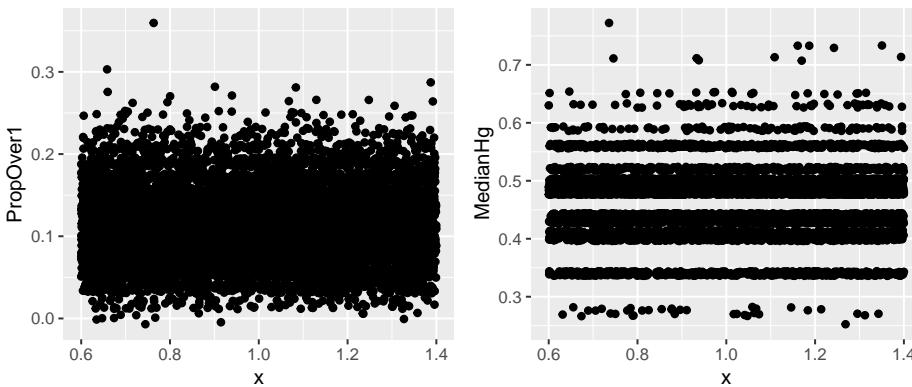
- We should not draw conclusions from this bootstrap distribution. The bootstrap is unreliable when we see the same values coming up repeatedly in clusters, with large gaps in between.
- This can be an issue for statistics that are a single value from the dataset (for example median)

2.4.8 When Gaps are/ aren't OK

- Sometimes, `ggplot()` shows gaps in a histogram, due mainly to binwidth. If the points seems to follow a fairly smooth trend (such as for $\text{prop} > 1$), then bootstrapping is ok. If there are large clusters and gaps (such as for median), bootstrapping is inadvisable.

Jitter plots can help us look for clusters and gaps.

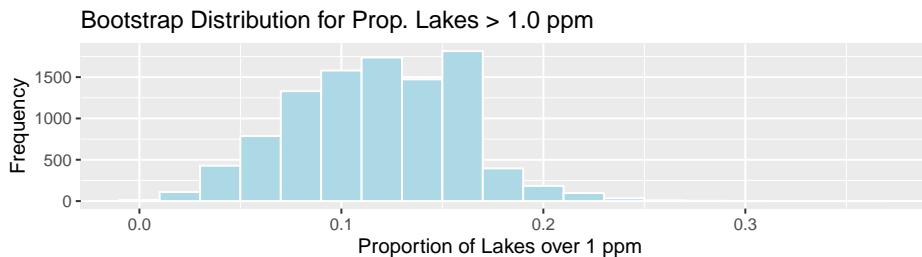
```
V1 <- ggplot(data=Lakes_Bootstrap_Results_Other, aes(y=PropOver1, x=1)) + geom_jitter()
V2 <- ggplot(data=Lakes_Bootstrap_Results_Other, aes(y=MedianHg, x=1)) + geom_jitter()
grid.arrange(V1, V2, ncol=2)
```



2.4.9 Changing Binwidth in Histogram

- Sometimes, default settings in `geom_histogram()` lead to less than optimal graphs. (For example, oddly-placed gaps that do not accurately represent the shape of the data)
- When a histogram shows undesired gaps, that are not really indicative of large gaps in the data, we can sometimes get rid of them by adjusting the binwidth.
- Before you do this, explore the data, such as through jitter plots. Do not change binwidth to intentionally manipulate or hide undesirable information. Your goal should be to find a plot that accurately displays the shape/trend in the data.

```
ggplot(data=Lakes_Bootstrap_Results_Other, aes(x=PropOver1)) +
  geom_histogram(color="white", fill="lightblue", binwidth=0.02) +
  xlab("Proportion of Lakes over 1 ppm") + ylab("Frequency") +
  ggtitle("Bootstrap Distribution for Prop. Lakes > 1.0 ppm")
```



2.4.10 What Bootstrapping Does and Doesn't Do

- The purpose of bootstrapping is to quantify the amount of uncertainty associated with a statistic that was calculated from a sample.
- A common misconception is that bootstrapping somehow increases the size of a sample by creating copies (or sampling with replacement). **This is wrong!!!!** Bootstrap samples are obtained by sampling from the original data, so they contain no new information and do not increase sample size. They simply help us understand how much our sample result could reasonably differ from that of the full population.

2.5 Bootstrap Distribution for Difference in Sample Means

2.5.1 Mercury Levels in Northern vs Southern Florida Lakes

We previously estimated the mean mercury level in Florida Lakes. We might be interested in whether mercury levels are higher or lower, on average, in Northern Florida compared to Southern Florida.

We'll divide the state along route 50, which runs East-West, passing through Northern Orlando.



Figure 2.2: from Google Maps

2.5.2 Comparing Northern and Southern Lakes

We add a variable indicating whether each lake lies in the northern or southern part of the state.

```
library(Lock5Data)
data(FloridaLakes)
#Location relative to rt. 50
FloridaLakes$Location <- as.factor(c("S", "S", "N", "S", "S", "N", "N", "N", "N", "N", "S", "N", "S", "N"))
head(FloridaLakes %>% select(Lake, AvgMercury, Location))

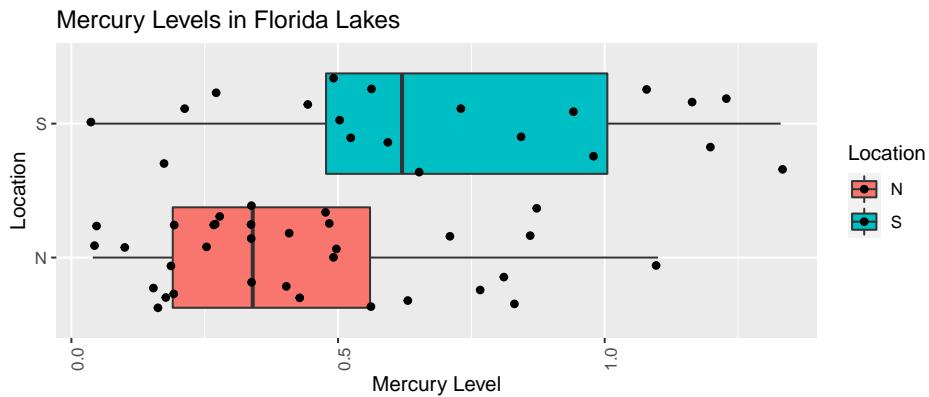
## # A tibble: 6 x 3
##   Lake           AvgMercury Location
##   <chr>          <dbl> <fct>
## 1 Alligator      1.23  S
## 2 Annie          1.33  S
## 3 Apopka         0.04  N
## 4 Blue Cypress   0.44  S
```

```
## 5 Brick          1.2  S
## 6 Bryant         0.27 N
```

2.5.3 Comparing Lakes in North and South Florida

```
LakesBP <- ggplot(data=FloridaLakes, aes(x=Location, y=AvgMercury, fill=Location)) +
  geom_boxplot() + geom_jitter() + ggtitle("Mercury Levels in Florida Lakes") +
  xlab("Location") + ylab("Mercury Level") + theme(axis.text.x = element_text(angle = 90))

LakesBP
```



2.5.4 Comparing Lakes in North and South Florida (cont.)

```
LakesTable <- FloridaLakes %>% group_by(Location) %>% summarize(MeanHg=mean(AvgMercury),
  StDevHg=sd(AvgMercury),
  N=n())

LakesTable

## # A tibble: 2 x 4
##   Location MeanHg  StDevHg     N
##   <fct>     <dbl>    <dbl> <int>
## 1 N          0.425   0.270    33
## 2 S          0.696   0.384    20
```

2.5.5 Model for Northern and Southern Lakes

$$\widehat{Hg} = b_0 + b_1 I_{\text{South}}$$

- b_0 represents the mean mercury level for lakes in North Florida, and

- b_1 represents the mean difference in mercury level for lakes in South Florida, compared to North Florida

2.5.6 Model for Lakes R Output

```
Lakes_M <- lm(data=FloridaLakes, AvgMercury ~ Location)
summary(Lakes_M)

##
## Call:
## lm(formula = AvgMercury ~ Location, data = FloridaLakes)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.65650 -0.23455 -0.08455  0.24350  0.67545
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)
## (Intercept) 0.42455   0.05519   7.692 0.000000000441 ***
## LocationS   0.27195   0.08985   3.027     0.00387 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3171 on 51 degrees of freedom
## Multiple R-squared:  0.1523, Adjusted R-squared:  0.1357
## F-statistic: 9.162 on 1 and 51 DF,  p-value: 0.003868
```

2.5.7 Interpreting Lakes Regression Output

$$\widehat{Hg} = 0.4245455 + 0.2719545I_{\text{South}}$$

- $b_1 = 0.27915 = 0.6965 - 0.4245$ is equal to the difference in mean mercury levels between Northern and Southern lakes. (We've already seen that for categorical variables, the least-squares estimate is the mean, so this makes sense.)
- We can use b_1 to assess the size of the difference in mean mercury concentration levels.
- Since the lakes we observed are only a sample of all lakes, we cannot assume the difference in mercury concentrations is exactly 0.4245 for all Northern vs Southern Florida lakes. We can use bootstrapping to find a reasonable range for this difference.

2.5.8 Bootstrapping for Northern vs Southern Lakes

Bootstrapping Procedure

1. Take bootstrap samples of **33 northern Lakes, and 20 southern Lakes**, by sampling with replacement.
2. Fit a model and record regression coefficient b_1 .
3. Repeat steps 2 and 3 10,000 times, keeping track of the regression coefficient estimates in each bootstrap sample.
4. Look at the distribution of regression coefficients in the bootstrap samples. The variability in this distribution can be used to approximate the variability in the sampling distributions for the b_1 .

2.5.9 Code for Bootstrapping for N vs S Lakes

```

b1 <- rep(NA, 10000)  #vector to store b1 values
for (i in 1:10000){
  NLakes <- sample_n(FloridaLakes %>% filter(Location=="N"), 33, replace=TRUE)  ## sample
  SLakes <- sample_n(FloridaLakes %>% filter(Location=="S"), 20, replace=TRUE)  ## sample
  BootstrapSample <- rbind(NLakes, SLakes)    ## combine Northern and Southern Lakes
  M <- lm(data=BootstrapSample, AvgMercury ~ Location) ## fit linear model
  b1[i] <- coef(M)[2] ## record b1
}
NS_Lakes_Bootstrap_Results <- data.frame(b1)  #save results as dataframe

```

2.5.10 Lakes: Bootstrap Percentile CI for Avg. Diff.

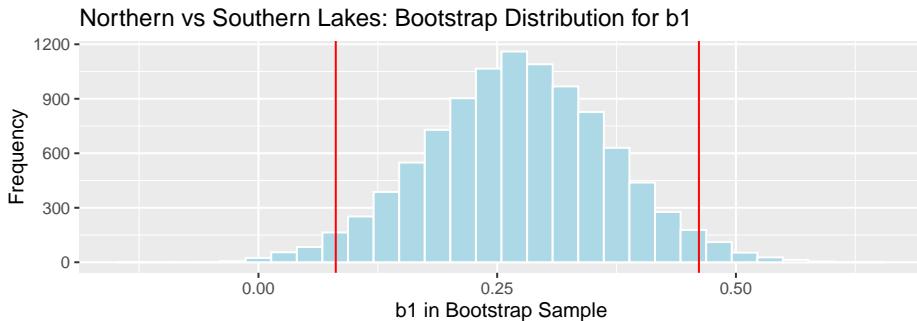
```

q.025 <- quantile(NS_Lakes_Bootstrap_Results$b1, 0.025)
q.975 <- quantile(NS_Lakes_Bootstrap_Results$b1, 0.975)
c(q.025, q.975)

##          2.5%      97.5%
## 0.08095682 0.46122992

```

2.5. BOOTSTRAP DISTRIBUTION FOR DIFFERENCE IN SAMPLE MEANS 117



We are 95% confident the average mercury level in Southern Lakes is between 0.08 and 0.46 ppm higher than in Northern Florida.

2.5.11 Lakes: Bootstrap SE CI for Avg. Difference

```
SE <- sd(NS_Lakes_Bootstrap_Results$b1)  
SE
```

```
## [1] 0.09585589
```

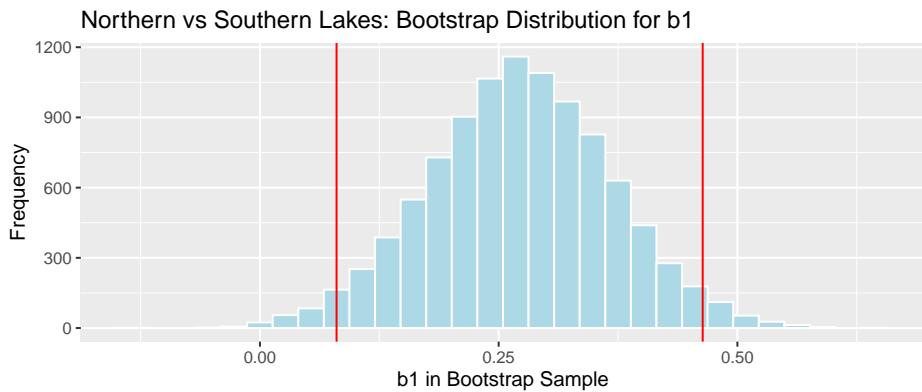
Note that this is fairly close to the standard error for b_1 reported in the R model output, but does not match exactly. R is calculating standard errors using a different approach that we'll discuss later.

```
MeanDiff <- coef(Lakes_M)[2]  
c(MeanDiff-2*SE, MeanDiff+2*SE)
```

```
## LocationS  LocationS  
## 0.08024277 0.46366633
```

2.5.12 Lakes: Bootstrap SE CI for Avg. Difference (cont.)

```
NS_Lakes_Bootstrap_Plot_b1 + geom_vline(xintercept=c(MeanDiff-2*SE, MeanDiff+2*SE), color="red")
```



We are 95% confident that the standard deviation in mercury level is all Florida Lakes is between 0.08 and 0.46 ppm.

2.6 Bootstrap Distribution for Simple Linear Regression Coefficients

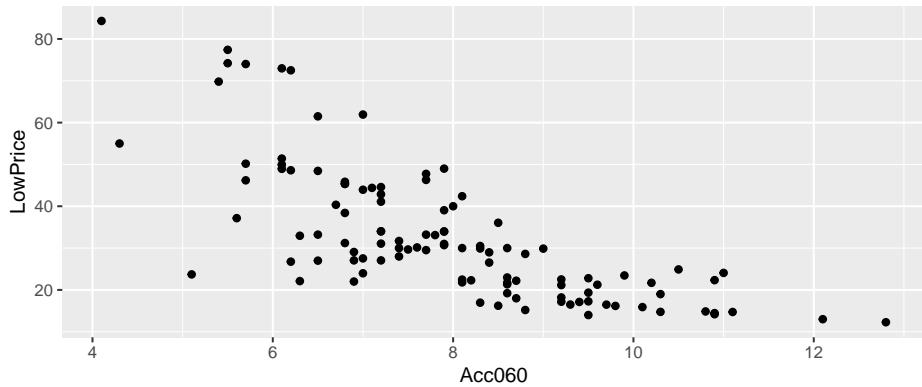
2.6.1 Bootstrapping for Cars Regression Coefficients

Recall the regression line estimating the relationship between a car's price and acceleration time.

This line was calculated using a sample of 110 cars, released in 2015.

$$\widehat{Price} = 89.90 - 7.19 \times \text{Acc. Time}$$

CarsA060



2.6.2 Cars Acc060 Model Output

```
summary(Cars_M_A060)

##
## Call:
## lm(formula = LowPrice ~ Acc060, data = Cars2015)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -29.512 -6.544 -1.265  4.759 27.195
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)
## (Intercept) 89.9036   5.0523  17.79 <0.0000000000000002 ***
## Acc060      -7.1933   0.6234 -11.54 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.71 on 108 degrees of freedom
## Multiple R-squared:  0.5521, Adjusted R-squared:  0.548
## F-statistic: 133.1 on 1 and 108 DF,  p-value: < 0.0000000000000002
```

2.6.3 Bootstrapping Regression Slope for All Cars

Since b_0 and b_1 were calculated from a sample of 110 new 2015 cars, we do not expect them to be exactly the same as what we would have obtained if we had data on all 2015 new cars. We'll use bootstrapping to get a sense for how much variability is associated with the slope and intercept of the regression line.

Bootstrapping Procedure

1. Take a bootstrap sample of size 110, by sampling cars with replacement.
2. Fit a linear regression model to the bootstrap sample with price as the response variable, and Acc060 as the explanatory variable. Record b_0 and b_1 .
3. Repeat steps 2 and 3 10,000 times, keeping track of the values on b_0 and b_1 in each bootstrap sample.
4. Look at the distribution of b_0 and b_1 from the bootstrap samples. The variability in these distributions can be used to approximate the variability in the sampling distribution for the intercept and slope.

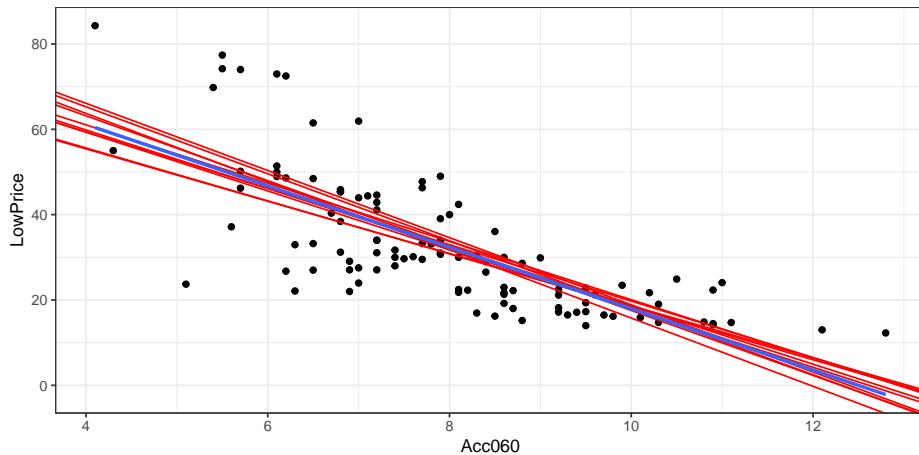
2.6.4 Code for Cars Regression Bootstrap

```
b0 <- rep(NA, 10000)
b1 <- rep(NA, 10000)

for (i in 1:10000){
  BootstrapSample <- sample_n(Cars2015, 110, replace=TRUE)
  Model_Bootstrap <- lm(data=BootstrapSample, LowPrice~Acc060)
  b0[i] <- Model_Bootstrap$coeff[1]
  b1[i] <- Model_Bootstrap$coeff[2]
}
Cars_Bootstrap_Results_Acc060 <- data.frame(b0, b1)
```

2.6.5 First 10 Bootstrap Regression Lines

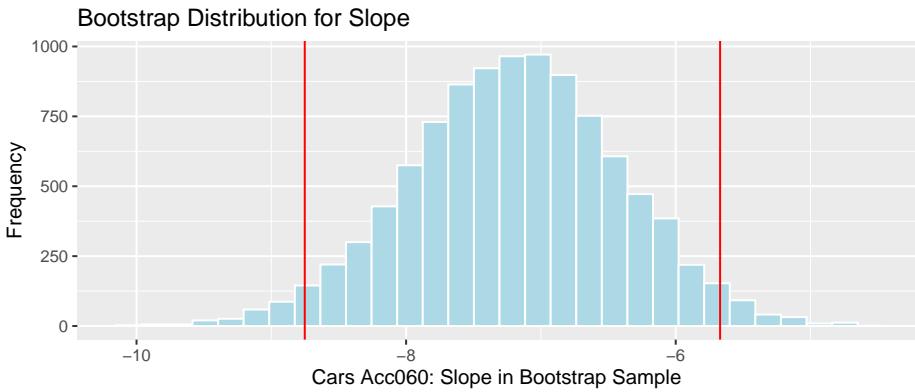
```
ggplot(data=Cars2015, aes(x=Acc060,y=LowPrice))+ geom_point()+
  theme_bw() +
  geom_abline(data=Cars_Bootstrap_Results_Acc060[1:10, ], aes(slope=b1,intercept=b0), color="red", show.legend = FALSE)
  stat_smooth( method="lm", se=FALSE)
```



2.6.6 Bootstrap Percentile CI for Slope of Cars Reg. Line

```
q.025 <- quantile(Cars_Bootstrap_Results_Acc060$b1, 0.025)
q.975 <- quantile(Cars_Bootstrap_Results_Acc060$b1, 0.975)
c(q.025, q.975)

##          2.5%      97.5%
## -8.751976 -5.670654
```



We are 95% confident that the average price of a new 2015 car decreases between -8.75 and -5.67 thousand dollars for each additional second it takes to accelerate from 0 to 60 mph.

2.6.7 Cars: Bootstrap SE CI Slope

```
SE <- sd(Cars_Bootstrap_Results_Acc060$b1)
SE
```

```
## [1] 0.7802172
```

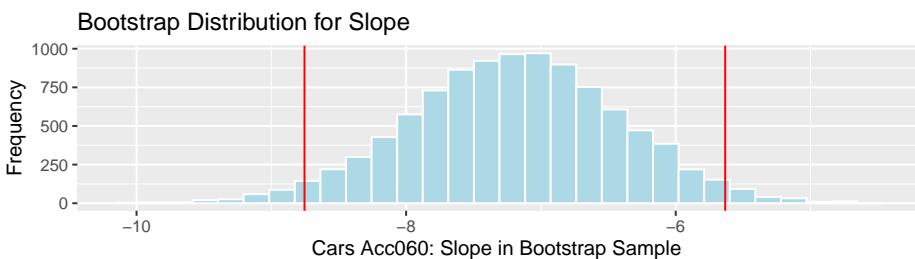
The bootstrap standard error is higher than the standard error in Acc060 line of the regression output.

```
Slope <- coef(Cars_M_A060)[2]
c(Slope-2*SE, Slope+2*SE)
```

```
##     Acc060      Acc060
## -8.753774 -5.632905
```

2.6.8 Cars: Bootstrap SE CI Slope (cont.)

```
Cars_Acc060_B_Slope_Plot + geom_vline(xintercept=c(Slope-2*SE, Slope+2*SE), color="red")
```

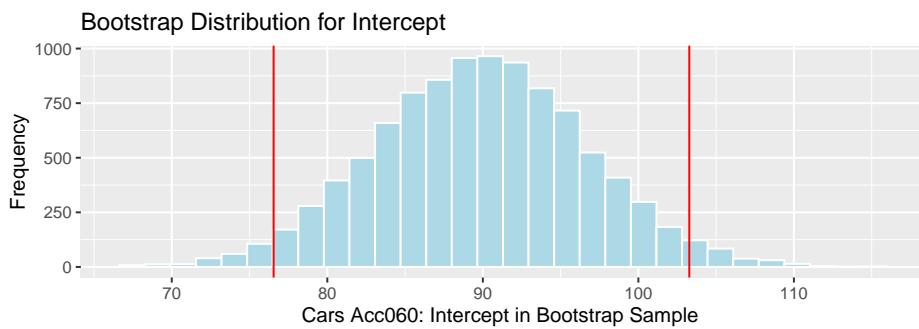


We are 95% confident that the standard deviation in mercury level is all Florida Lakes is between -8.75 and -5.63 ppm.

2.6.9 Bootstrap Percentile CI for Int. of Cars Reg. Line

```
q.025 <- quantile(Cars_Bootstrap_Results_Acc060$b0, 0.025)
q.975 <- quantile(Cars_Bootstrap_Results_Acc060$b0, 0.975)
c(q.025, q.975)

##      2.5%    97.5%
## 76.54801 103.27401
```



We are 95% confident that the intercept for the regression line relating price to acceleration time is between 76.55 and 103.27 thousand dollars. This intercept does not have a meaningful interpretation in context.

2.6.10 Cars: Bootstrap SE CI Intercept

```
SE <- sd(Cars_Bootstrap_Results_Acc060$b0)
SE
```

```
## [1] 6.796352
```

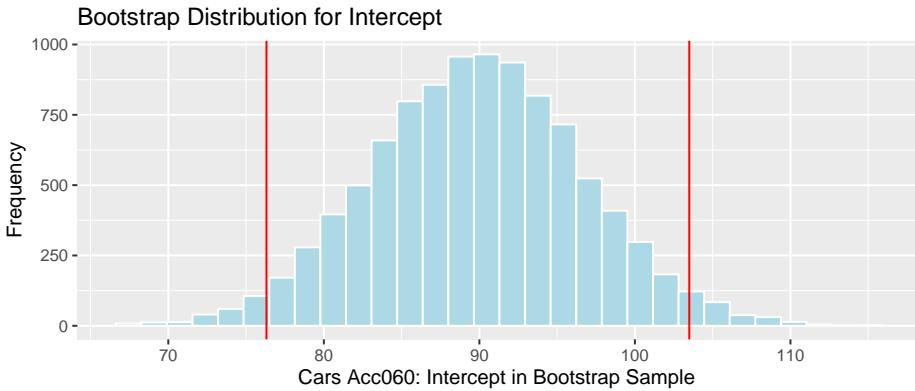
The bootstrap standard error is higher than the standard error for the intercept in the regression output.

```
Int <- coef(Cars_M_A060)[1]
c(Int-2*SE, Int+2*SE)
```

```
## (Intercept) (Intercept)
##    76.31088   103.49628
```

2.6.11 Cars: Bootstrap SE CI Intercept (cont.)

```
Cars_Acc060_B_Intercept_Plot + geom_vline(xintercept=c(Int-2*SE, Int+2*SE), color="red")
```



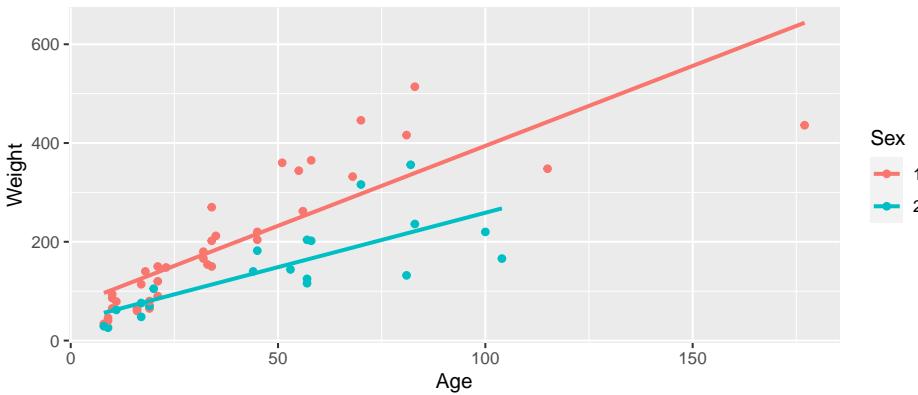
We are 95% confident that the standard deviation in mercury level is all Florida Lakes is between 76.31 and 103.5 ppm.

2.7 Bootstrap Distribution for Multiple Regression Coefficients

2.7.1 Model for Predicting Bear Weights

We previously used a linear regression model to predict the weights of wild bears, using a sample of 97 bears. Recall the model and its interpretations.

```
ggplot(data=Bears_Subset, aes(x=Age, y=Weight, color=Sex)) +
  geom_point() + stat_smooth(method="lm", se=FALSE)
```



2.7.2 Model for Predicting Bear Weights (cont.)

```
Bear_M_Age_Sex_Int <- lm(data=Bears_Subset, Weight~ Age*Sex)
summary(Bear_M_Age_Sex_Int)

##
## Call:
## lm(formula = Weight ~ Age * Sex, data = Bears_Subset)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -207.583  -38.854   -9.574   23.905  174.802
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)    
## (Intercept) 70.4322   17.7260   3.973 0.000219 ***  
## Age         3.2381    0.3435   9.428 0.000000000000765 ***
## Sex2        -31.9574   35.0314  -0.912 0.365848    
## Age:Sex2    -1.0350    0.6237  -1.659 0.103037    
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 70.18 on 52 degrees of freedom
## (41 observations deleted due to missingness)
## Multiple R-squared:  0.6846, Adjusted R-squared:  0.6664 
## F-statistic: 37.62 on 3 and 52 DF,  p-value: 0.0000000000004552
```

2.7.3 Model Interpretations in Bears Interaction Model

$$\widehat{\text{Weight}} = 70.43 + 3.24 \times \text{Age} - 31.95 \times I_{Female} - 1.04 \times \text{Age} \times I_{Female}$$

This model was fit using a sample of 97 wild bears. If we were to take a different sample, and fit a regression model, we would obtain different values for b_0 , b_1 , b_2 , and b_3 , as well as relevant quantities $b_0 - b_2$ and $b_1 - b_3$, due to sampling variability.

Questions of interest: Find a reasonable range for the following quantities:

1. the mean monthly weight gain among all male bears
2. the mean monthly weight gain among all female bears
3. the mean weight among all 24 month old male bears
4. the mean weight among all 24 month old female bears

Just as we did for sample means and proportions, we can answer these questions via bootstrapping.

2.7.4 Quantities of Interest

First, we need to find expressions for the quantities we are interested in, in terms of the model coefficients.

$$\widehat{\text{Weight}} = b_0 + b_1 \times \text{Age} - b_2 \times I_{Female} + b_3 \times \text{Age} \times I_{Female}$$

Expected Weight for Female Bears:

$$\begin{aligned}\widehat{\text{Weight}} &= b_0 + b_1 \times \text{Age} + b_2 + b_3 \times \text{Age} \\ &= (b_0 + b_2) + (b_1 + b_3) \times \text{Age}\end{aligned}$$

Expected Weight for Male Bears:

$$\widehat{\text{Weight}} = b_0 + b_1 \times \text{Age}$$

Questions of interest: Find a reasonable range for the following quantities:

1. the mean weight gain per month among all male bears (b_1)
2. the mean weight gain per month among all female bears ($b_1 + b_3$)
3. the mean weight among all 24 month old male bears ($b_0 + 24b_1$)
4. the mean weight among all 24 month old female bears ($b_0 + b_2 + 24(b_1 + b_3)$)

2.7.5 Bootstrapping for Bears Regression Coefficients

Bootstrapping Procedure

1. Take a bootstrap sample of size 97, by sampling bears with replacement.
2. Fit a model and record regression coefficients b_0, b_1, b_2, b_3 .
3. Repeat steps 2 and 3 10,000 times, keeping track of the regression coefficient estimates in each bootstrap sample.
4. Look at the distribution of regression coefficients in the bootstrap samples. The variability in this distribution can be used to approximate the variability in the sampling distributions for the regression coefficients.

2.7.6 Bootstrap Code for Bears Regression Coefficients

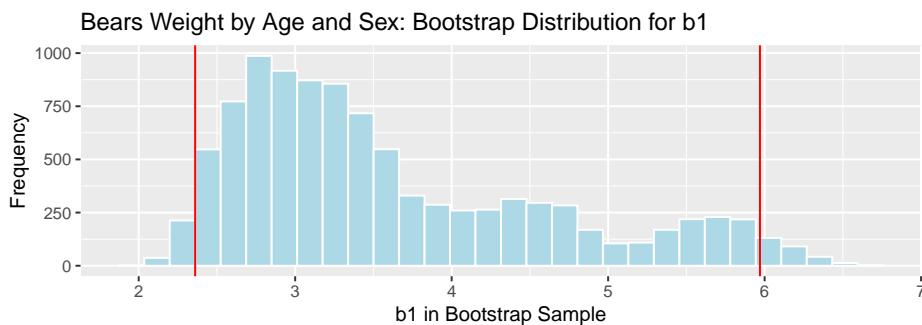
```
b0 <- rep(NA, 10000)
b1 <- rep(NA, 10000)
b2 <- rep(NA, 10000)
b3 <- rep(NA, 10000)
for (i in 1:10000){
  BootstrapSample <- sample_n(Bears_Subset, 97, replace=TRUE) #take bootstrap sample
  M <- lm(data=BootstrapSample, Weight ~ Age*Sex) ## fit linear model
  b0[i] <- coef(M)[1] ## record b0
  b1[i] <- coef(M)[2] ## record b1
  b2[i] <- coef(M)[3] ## record b2
  b3[i] <- coef(M)[4] ## record b3
}
Bears_Bootstrap_Results <- data.frame(b0, b1, b2, b3)
```

2.7.7 Bootstrap Percentile CI for b_1 in Bears Model

The average weight gain per month for male bears is represented by b_1 .

```
q.025 <- quantile(Bears_Bootstrap_Results$b1, 0.025)
q.975 <- quantile(Bears_Bootstrap_Results$b1, 0.975)
c(q.025, q.975)
```

```
##      2.5%    97.5%
## 2.360746 5.970272
```



We are 95% confident that male bears gain between 2.36 and 5.97 pounds per month, on average.

2.7.8 Bootstrap SE CI for b_1 in Bears Model

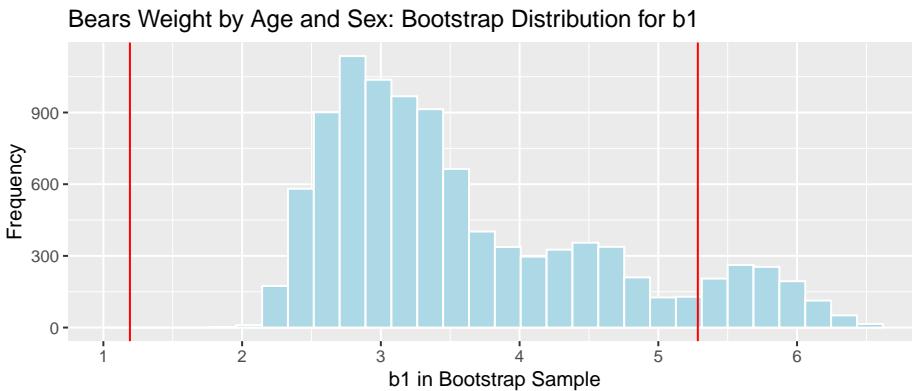
```
b1 <- coef(Bear_M_Age_Sex_Int)[2]
SE <- sd(Bears_Bootstrap_Results$b1)
SE

## [1] 1.023381
c(b1-2*SE, b1+2*SE)

##      Age      Age
## 1.191375 5.284900
```

2.7.9 Bootstrap SE CI for b_1 in Bears Model

```
Bears_plot_b1 + geom_vline(xintercept=c(b1-2*SE, b1+2*SE), color="red")
```



There is a considerable discrepancy between the percentile and standard error confidence intervals. Since the bootstrap distribution for b_1 is not symmetric, only the percentile interval is reliable.

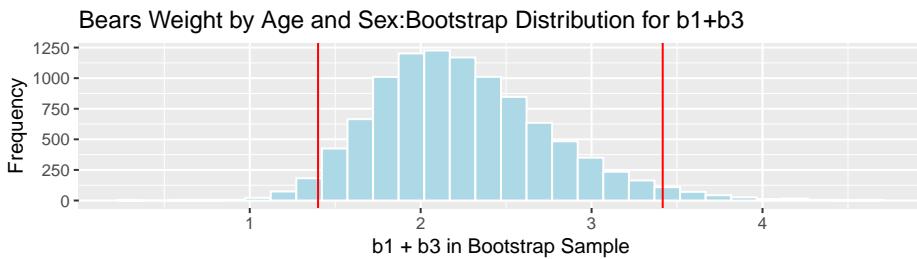
2.7.10 Bootstrap Percentile CI for $b_1 + b_3$

The average weight gain per month for female bears is represented by $b_1 + b_3$.

```
q.025 <- quantile(Bears_Bootstrap_Results$b1 + Bears_Bootstrap_Results$b3, 0.025)
q.975 <- quantile(Bears_Bootstrap_Results$b1 + Bears_Bootstrap_Results$b3, 0.975)

c(q.025, q.975)

##      2.5%      97.5%
## 1.399662 3.416039
```



We are 95% confident that female bears gain between 1.3996618 and 3.416039 pounds per month, on average.

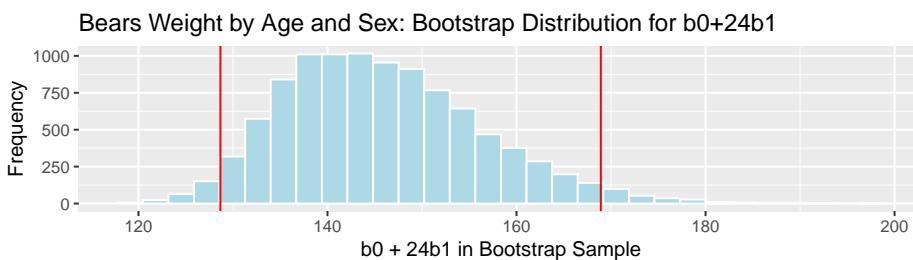
2.7.11 Bootstrap Percentile CI for $b_0 + 24b_1$

The mean weight of all 24 month old male bears is represented by $b_0 + 24b_1$.

```
q.025 <- quantile(Bears_Bootstrap_Results$b0 + 24*Bears_Bootstrap_Results$b1, 0.025)
q.975 <- quantile(Bears_Bootstrap_Results$b0 + 24*Bears_Bootstrap_Results$b1, 0.975)

c(q.025, q.975)

##      2.5%    97.5%
## 128.6692 168.9187
```



We are 95% confident that the mean weight of all 24 month old bears is between 128.6692134 and 168.9186551 pounds.

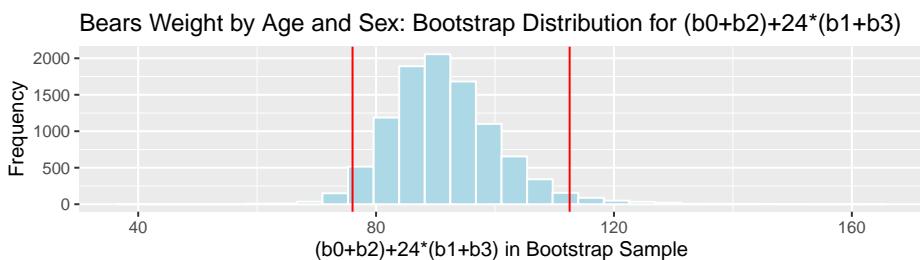
2.7.12 Bootstrap Percentile CI for $(b_0 + b_2) + 24(b_1 + b_3)$

The mean weight of all 24 month old female bears is represented by $(b_0 + b_2) + 24(b_1 + b_3)$.

```
q.025 <- quantile(Bears_Bootstrap_Results$b0 + Bears_Bootstrap_Results$b2 +
                    24*(Bears_Bootstrap_Results$b1 + Bears_Bootstrap_Results$b3), 0.025)
q.975 <- quantile(Bears_Bootstrap_Results$b0 + Bears_Bootstrap_Results$b2 +
                    24*(Bears_Bootstrap_Results$b1 + Bears_Bootstrap_Results$b3), 0.975)
```

```
c(q.025, q.975)
##      2.5%    97.5%
##  76.04432 112.54605
```

2.7.13 Bootstrap Percentile CI for $(b_0 + b_2) + 24(b_1 + b_3)$



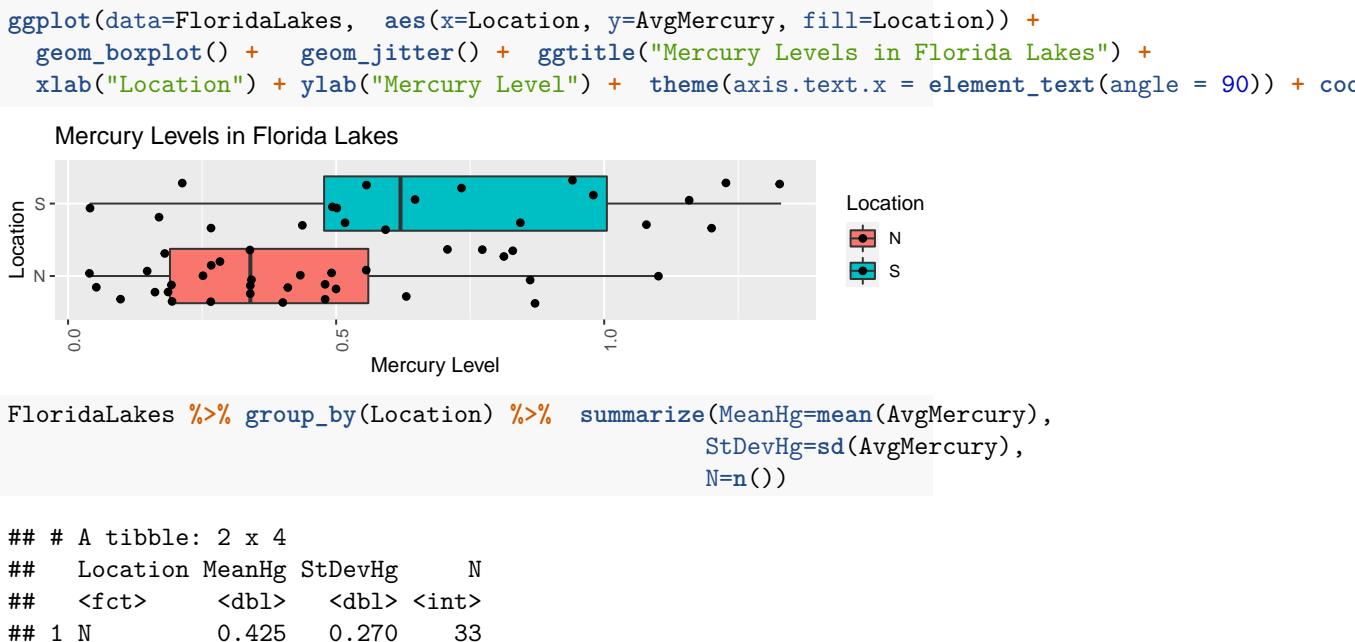
We are 95% confident that the mean weight of all 24 month old female bears is between 76.0443239 and 112.5460455 pounds.

Chapter 3

Simulation-Based Hypothesis Tests

3.1 Introduction to Hypothesis Testing via Simulation

3.1.1 Difference between North and South?



```
## 2 S      0.696  0.384   20
```

3.1.2 Model for Lakes Example

$$\widehat{Hg} = b_0 + b_1 I_{\text{South}}$$

- b_0 represents the mean mercury level for lakes in North Florida, and
- b_1 represents the mean difference in mercury level for lakes in South Florida, compared to North Florida

3.1.3 Model for Lakes R Output

```
Lakes_M <- lm(data=FloridaLakes, AvgMercury ~ Location)
summary(Lakes_M)

##
## Call:
## lm(formula = AvgMercury ~ Location, data = FloridaLakes)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -0.65650 -0.23455 -0.08455  0.24350  0.67545 
##
## Coefficients:
##             Estimate Std. Error t value    Pr(>|t|)    
## (Intercept) 0.42455   0.05519   7.692 0.000000000441 ***
## LocationS   0.27195   0.08985   3.027   0.00387 **  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3171 on 51 degrees of freedom
## Multiple R-squared:  0.1523, Adjusted R-squared:  0.1357 
## F-statistic: 9.162 on 1 and 51 DF,  p-value: 0.003868
```

3.1.4 Interpreting Lakes Regression Output

$$\widehat{Hg} = 0.4245455 + 0.2719545 I_{\text{South}}$$

- $b_1 = 0.27915 = 0.6965 - 0.4245$ is equal to the difference in mean mercury levels between Northern and Southern lakes. (We've already seen that for categorical variables, the least-squares estimate is the mean, so this makes sense.)

3.1. INTRODUCTION TO HYPOTHESIS TESTING VIA SIMULATION 133

- We can use b_1 to assess the size of the difference in mean mercury concentration levels.

3.1.5 Differences between Northern and Southern Lakes?

- Do these results provide evidence that among all Florida lakes, the mean mercury level is higher in the South than in the North?
- Is it possible that there is really no difference in mean mercury level, and we just happened, by random chance, to select lakes in the south that had higher mercury levels than most?

Key Question:

- How likely is it that we would have observed a difference in means (i.e. a value of b_1) as extreme as $0.6965 - 0.4245 = 0.27195$ ppm, merely by chance, if there is really no relationship between location and mercury level?

3.1.6 Investigation by Simulation

We can answer the key question using simulation.

We'll simulate situations where there is no relationship between location and mercury level, and see how often we observe a value of b_1 as extreme as 0.27195.

Procedure:

1. Randomly shuffle the locations of the lakes, so that any relationship between location and mercury level is due only to chance.
2. Calculate the difference in mean mercury levels (i.e. value of b_1) in “Northern” and “Southern” lakes, using the shuffled data.
3. Repeat steps 1 and 2 many (say 10,000) times, recording the difference in means (i.e. value of b_1) each time.
4. Analyze the distribution of mean differences, simulated under the assumption that there is no relationship between location and mercury level. Look whether the actual difference we observed is consistent with the simulation results.

3.1.7 First Lakes Shuffle Simulation

```
ShuffledLakes <- FloridaLakes    ## create copy of dataset
ShuffledLakes$Location <- ShuffledLakes$Location[sample(1:nrow(ShuffledLakes))]
```

Lake	Location	AvgMercury	Shuffled Location
Alligator	S	1.23	N
Annie	S	1.33	N
Apopka	N	0.04	N
Blue Cypress	S	0.44	N
Brick	S	1.20	N
Bryant	N	0.27	S

3.1.8 First Shuffle Model Results

Recall this model was fit under an assumption of no relationship between location and average mercury.

```
M_Lakes_Shuffle <- lm(data=ShuffledLakes, AvgMercury~Location)
summary(M_Lakes_Shuffle)

##
## Call:
## lm(formula = AvgMercury ~ Location, data = ShuffledLakes)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -0.4891 -0.2591 -0.0440  0.2460  0.8009 
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)    
## (Intercept) 0.529091  0.059944  8.826 0.00000000000761 ***
## LocationS   -0.005091  0.097582 -0.052          0.959    
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3444 on 51 degrees of freedom
## Multiple R-squared:  5.336e-05, Adjusted R-squared:  -0.01955 
## F-statistic: 0.002722 on 1 and 51 DF,  p-value: 0.9586
```

3.1.9 Second Lakes Shuffle Simulation

```
ShuffledLakes <- FloridaLakes    ## create copy of dataset
ShuffledLakes$Location <- ShuffledLakes$Location[sample(1:nrow(ShuffledLakes))]
kable(head(Shuffle1df))
```

Lake	Location	AvgMercury	Shuffled Location
Alligator	S	1.23	N
Annie	S	1.33	N
Apopka	N	0.04	N
Blue Cypress	S	0.44	N
Brick	S	1.20	N
Bryant	N	0.27	S

3.1.10 Second Shuffle Model Results

Recall this model was fit under an assumption of no relationship between location and average mercury.

```
M_Lakes_Shuffle <- lm(data=ShuffledLakes, AvgMercury~Location)
summary(M_Lakes_Shuffle)

##
## Call:
## lm(formula = AvgMercury ~ Location, data = ShuffledLakes)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -0.51485 -0.27485 -0.05485  0.27515  0.77515 
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)    
## (Intercept) 0.55485   0.05961   9.308 0.0000000000141 ***
## LocationS   -0.07335   0.09704  -0.756      0.453    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3425 on 51 degrees of freedom
## Multiple R-squared:  0.01108,    Adjusted R-squared:  -0.008313 
## F-statistic: 0.5713 on 1 and 51 DF,  p-value: 0.4532
```

3.1.11 Third Lakes Shuffle Simulation

```
ShuffledLakes <- FloridaLakes    ## create copy of dataset
ShuffledLakes$Location <- ShuffledLakes$Location[sample(1:nrow(ShuffledLakes))]
kable(head(Shuffle1df))
```

Lake	Location	AvgMercury	Shuffled Location
Alligator	S	1.23	N
Annie	S	1.33	N
Apopka	N	0.04	N
Blue Cypress	S	0.44	N
Brick	S	1.20	N
Bryant	N	0.27	N

3.1.12 Third Shuffle Model Results

Recall this model was fit under an assumption of no relationship between location and average mercury.

```
M_Lakes_Shuffle <- lm(data=ShuffledLakes, AvgMercury~Location)
summary(M_Lakes_Shuffle)

##
## Call:
## lm(formula = AvgMercury ~ Location, data = ShuffledLakes)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -0.56200 -0.26200 -0.08182  0.22818  0.74818 
##
## Coefficients:
##             Estimate Std. Error t value    Pr(>|t|)    
## (Intercept)  0.48182   0.05905   8.16 0.000000000818 ***
## LocationS    0.12018   0.09612   1.25      0.217    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3392 on 51 degrees of freedom
## Multiple R-squared:  0.02974,    Adjusted R-squared:  0.01072 
## F-statistic: 1.563 on 1 and 51 DF,  p-value: 0.2169
```

3.1.13 Fourth Lakes Shuffle Simulation

```
ShuffledLakes <- FloridaLakes    ## create copy of dataset
ShuffledLakes$Location <- ShuffledLakes$Location[sample(1:nrow(ShuffledLakes))]
kable(head(Shuffle1df))
```

Lake	Location	AvgMercury	Shuffled Location
Alligator	S	1.23	N
Annie	S	1.33	S
Apopka	N	0.04	S
Blue Cypress	S	0.44	N
Brick	S	1.20	N
Bryant	N	0.27	N

3.1.14 Fourth Shuffle Model Results

Recall this model was fit under an assumption of no relationship between location and average mercury.

```
M_Lakes_Shuffle <- lm(data=ShuffledLakes, AvgMercury~Location)
summary(M_Lakes_Shuffle)

##
## Call:
## lm(formula = AvgMercury ~ Location, data = ShuffledLakes)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -0.5261 -0.2730 -0.0630  0.2470  0.7639 
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)    
## (Intercept) 0.56606   0.05929  9.548 0.0000000000061 ***
## LocationS  -0.10306   0.09651 -1.068      0.291    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3406 on 51 degrees of freedom
## Multiple R-squared:  0.02187,    Adjusted R-squared:  0.002691 
## F-statistic:  1.14 on 1 and 51 DF,  p-value: 0.2906
```

3.1.15 Fifth Lakes Shuffle Simulation

```
ShuffledLakes <- FloridaLakes    ## create copy of dataset
ShuffledLakes$Location <- ShuffledLakes$Location[sample(1:nrow(ShuffledLakes))]
kable(head(Shuffle1df))
```

Lake	Location	AvgMercury	Shuffled Location
Alligator	S	1.23	N
Annie	S	1.33	N
Apopka	N	0.04	N
Blue Cypress	S	0.44	S
Brick	S	1.20	S
Bryant	N	0.27	N

3.1.16 Fifth Shuffle Model Results

Recall this model was fit under an assumption of no relationship between location and average mercury.

```
M_Lakes_Shuffle <- lm(data=ShuffledLakes, AvgMercury~Location)
summary(M_Lakes_Shuffle)

##
## Call:
## lm(formula = AvgMercury ~ Location, data = ShuffledLakes)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -0.50455 -0.24850 -0.05455  0.22545  0.78545 
##
## Coefficients:
##             Estimate Std. Error t value    Pr(>|t|)    
## (Intercept)  0.54455   0.05981   9.104 0.0000000000286 ***
## LocationS   -0.04605   0.09737  -0.473    0.638    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3436 on 51 degrees of freedom
## Multiple R-squared:  0.004366, Adjusted R-squared:  -0.01516 
## F-statistic: 0.2236 on 1 and 51 DF,  p-value: 0.6383
```

3.1.17 Code for Simulation Investigation

```
b1 <- Lakes_M$coef[2] ## record value of b1 from actual data

## perform simulation
b1Sim <- rep(NA, 10000)          ## vector to hold results
ShuffledLakes <- FloridaLakes    ## create copy of dataset
for (i in 1:10000){
  #randomly shuffle locations
```

```

ShuffledLakes$Location <- ShuffledLakes$Location[sample(1:nrow(ShuffledLakes))]
ShuffledLakes_M<- lm(data=ShuffledLakes, AvgMercury ~ Location) #fit model to shuffled data
b1Sim[i] <- ShuffledLakes_M$coef[2] ## record b1 from shuffled model
}
NSLakes_SimulationResults <- data.frame(b1Sim) #save results in dataframe

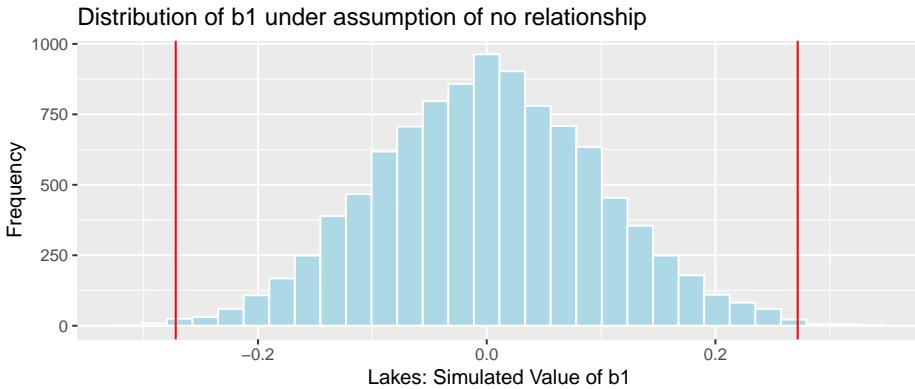
```

3.1.18 Simulation Results

```

NSLakes_SimulationResultsPlot <- ggplot(data=NSLakes_SimulationResults, aes(x=b1Sim)) +
  geom_histogram(fill="lightblue", color="white") +
  geom_vline(xintercept=c(b1, -1*b1), color="red") +
  xlab("Lakes: Simulated Value of b1") + ylab("Frequency") +
  ggtitle("Distribution of b1 under assumption of no relationship")
NSLakes_SimulationResultsPlot

```



It appears unlikely that we would observe a value of b_1 as extreme as 0.27195 ppm by chance, if there is really no relationship between location and mercury level.

3.1.19 Conclusions

Number of simulations resulting in simulation value of b_1 more extreme than 0.27195.

```
sum(abs(b1Sim) > abs(b1))
```

```
## [1] 31
```

Proportion of simulations resulting in simulation value of b_1 more extreme than 0.27195.

```
mean(abs(b1Sim) > abs(b1))
```

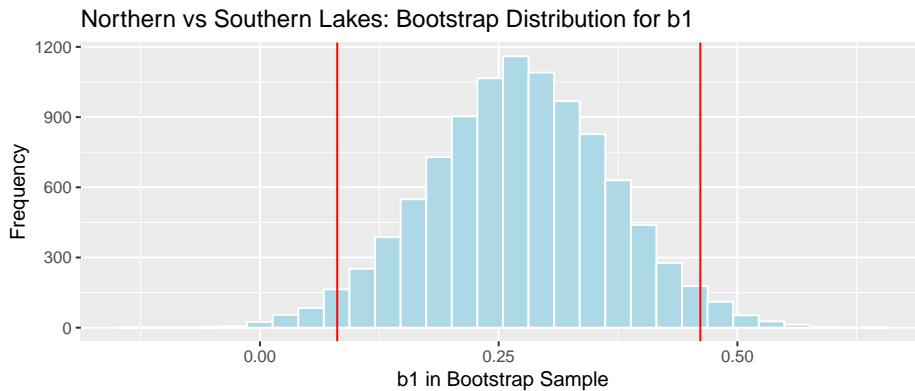
```
## [1] 0.0031
```

The probability of observing a value of b_1 as extreme as 0.27195 by chance, when there is no relationship between location and mercury level is very low.

There is strong evidence of a relationship between location and mercury level. In this case, there is strong evidence that mercury level is higher in Southern Lakes than northern Lakes.

3.1.20 Recall Lakes Bootstrap for Difference

```
##      2.5%      97.5%
## 0.08095682 0.46122992
NS_Lakes_Bootstrap_Plot_b1 + geom_vline(xintercept = c(q.025, q.975), color="red")
```



We are 95% confident the average mercury level in Southern Lakes is between 0.08 and 0.46 ppm higher than in Northern Florida.

The fact that the interval does not contain 0 is consistent with the hypothesis test.

3.1.21 Hypothesis Testing Terminology

We can think of the simulation as a test of the following hypotheses:

Hypothesis 1: There is no relationship between location and mercury level. (Thus the difference of 0.27 we observed in our data occurred just by chance).

Hypothesis 2: The difference we observed did not occur by chance (suggesting a relationship between location and mercury level).

The “no relationship,” or “chance alone” hypothesis is called the **null hypothesis**. The other hypothesis is called the **alternative hypothesis**.

3.1.22 Hypothesis Testing Terminology (Continued)

We used b_1 to measure difference in mean mercury levels between the locations in our observed data.

We found that the probability of observing a difference as extreme as 0.27 when Hypothesis 1 is true is very low (approximately 0.0017)

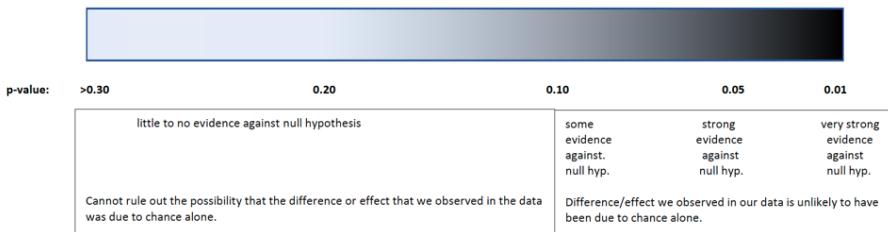
- The statistic used to measure the difference or relationship we are interested in is called a **test statistic**.
 - In this case, the test statistic is (b_1)
- The **p-value** is the probability of observing a test statistic as extreme or more extreme than we did when the null hypothesis is true.
 - A low p-value provides evidence against the null hypothesis.
 - A high p-value means that the data could have plausibly been obtained when the null hypothesis is true, and thus the null hypothesis cannot be ruled out.
 - A high p-value does not mean that the null hypothesis is true or probably true. A p-value can only tell us the strength of evidence against the null hypothesis, and should never be interpreted as support for the null hypothesis.

3.1.23 How Low Should the p-value Be

```
knitr::include_graphics("pvals.png")
```

Definition: A p-value represents the probability of observing a result as extreme or more extreme than we did, assuming the null hypothesis (and any other model assumptions we make) are true.

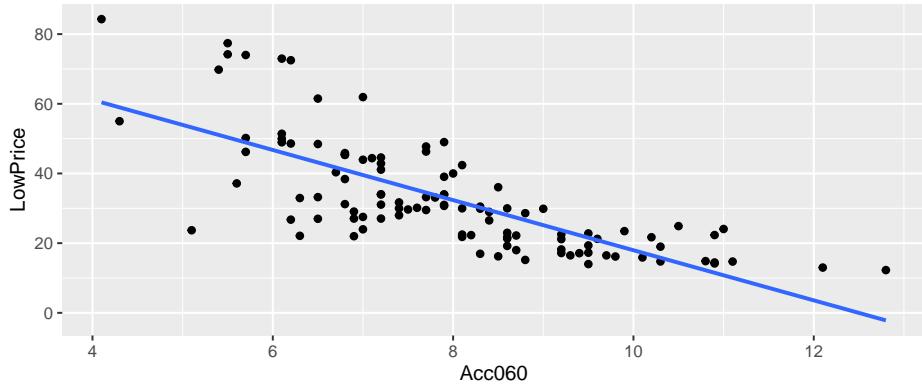
Darker color indicates stronger evidence against null hypothesis:



3.2 Simulation-Based Hypothesis Test for Regression Coefficients

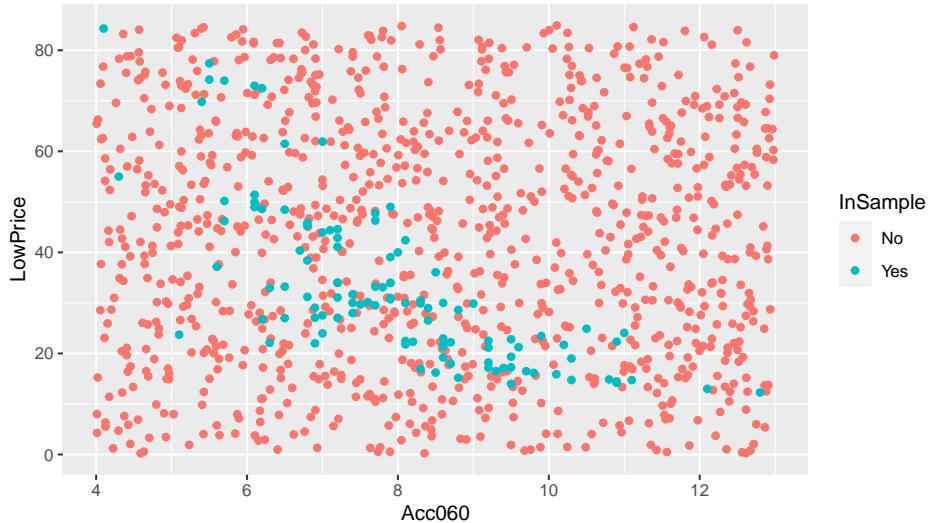
3.2.1 Is Car Price Associated with Acceleration Time?

```
ggplot(data=Cars2015, aes(x=Acc060, y=LowPrice)) + geom_point() +
  stat_smooth(method="lm", se=FALSE)
```



$$\widehat{Price} = b_0 + b_1 \times \text{Acc. Time} = 89.9036 - 7.1933 \times \text{Acc. Time}$$

3.2.2 What if there really was no Relationship?



Is it possible that there is really no relationship between price and acceleration time, and we just happened to choose a sample that led to a slope of -7.1933,

by chance?

3.2.3 Acc060 Key Question and Hypotheses

If there is really no relationship between price and acceleration time, then we would expect a slope (i.e value of b_1) equal to 0.

Key Question:

- How likely is it that we would have observed a slope (i.e. a value of b_1) as extreme as -7.1933 merely by chance, if there is really no relationship between price and acceleration time?

Null Hypothesis: There is no relationship between price and acceleration time, and the slope we observed occurred merely by chance.

Alternative Hypothesis: The slope we observed is due to more than chance, suggesting there is a relationship between price and acceleration time.

3.2.4 Simulation-Based Hypothesis Test for Cars

Procedure:

1. Randomly shuffle the acceleration times, so that any relationship between acceleration time and price is due only to chance.
2. Fit a regression line to the shuffled data and record the slope of the regression line.
3. Repeat steps 1 and 2 many (say 10,000) times, recording the slope (i.e. value of b_1) each time.
4. Analyze the distribution of slopes, simulated under the assumption that there is no relationship between price and acceleration time. Look whether the actual slope we observed is consistent with the simulation results.

3.2.5 First Cars Shuffle Simulation

```
ShuffledCars <- Cars2015    ## create copy of dataset
ShuffledCars$Acc060 <- ShuffledCars$Acc060[sample(1:nrow(ShuffledCars))]
```

Make	Model	LowPrice	Acc060	ShuffledAcc060
Chevrolet	Spark	12.270	12.8	8.6
Hyundai	Accent	14.745	10.3	6.5
Kia	Rio	13.990	9.5	7.2
Mitsubishi	Mirage	12.995	12.1	7.4
Nissan	Versa Note	14.180	10.9	5.4
Dodge	Dart	16.495	9.3	8.4

3.2.6 First Cars Shuffle Model Results

Recall this model was fit under an assumption of no relationship between price and acceleration time.

```
M_Cars_Shuffle <- lm(data=ShuffledCars, LowPrice~Acc060)
summary(M_Cars_Shuffle)

##
## Call:
## lm(formula = LowPrice ~ Acc060, data = ShuffledCars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -21.145 -11.306 - 3.949   9.605  50.885 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 25.5358    7.5156   3.398 0.000952 ***
## Acc060      0.9162    0.9273   0.988 0.325362    
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.93 on 108 degrees of freedom
## Multiple R-squared:  0.008957,  Adjusted R-squared:  -0.0002189 
## F-statistic: 0.9761 on 1 and 108 DF,  p-value: 0.3254
```

3.2.7 Second Cars Shuffle Simulation

```
ShuffledCars <- Cars2015    ## create copy of dataset
ShuffledCars$Acc060 <- ShuffledCars$Acc060[sample(1:nrow(ShuffledCars))]
```

Make	Model	LowPrice	Acc060	ShuffledAcc060
Chevrolet	Spark	12.270	12.8	7.9
Hyundai	Accent	14.745	10.3	7.9
Kia	Rio	13.990	9.5	7.4
Mitsubishi	Mirage	12.995	12.1	6.1
Nissan	Versa Note	14.180	10.9	7.2
Dodge	Dart	16.495	9.3	5.5

3.2.8 Second Cars Shuffle Model Results

Recall this model was fit under an assumption of no relationship between price and acceleration time.

```
M_Cars_Shuffle <- lm(data=ShuffledCars, LowPrice~Acc060)
summary(M_Cars_Shuffle)

##
## Call:
## lm(formula = LowPrice ~ Acc060, data = ShuffledCars)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -20.550 -11.017 - 3.430  9.464  51.609
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 35.364     7.545   4.687 0.00000814 ***
## Acc060      -0.322     0.931  -0.346     0.73
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.99 on 108 degrees of freedom
## Multiple R-squared:  0.001106,   Adjusted R-squared:  -0.008143
## F-statistic: 0.1196 on 1 and 108 DF,  p-value: 0.7301
```

3.2.9 Third Cars Shuffle Simulation

```
ShuffledCars <- Cars2015    ## create copy of dataset
ShuffledCars$Acc060 <- ShuffledCars$Acc060[sample(1:nrow(ShuffledCars))]
```

Make	Model	LowPrice	Acc060	ShuffledAcc060
Chevrolet	Spark	12.270	12.8	6.9
Hyundai	Accent	14.745	10.3	7.9
Kia	Rio	13.990	9.5	10.8
Mitsubishi	Mirage	12.995	12.1	6.1
Nissan	Versa Note	14.180	10.9	8.4
Dodge	Dart	16.495	9.3	7.4

3.2.10 Third Cars Shuffle Model Results

Recall this model was fit under an assumption of no relationship between price and acceleration time.

```
M_Cars_Shuffle <- lm(data=ShuffledCars, LowPrice~Acc060)
summary(M_Cars_Shuffle)
```

```
##
## Call:
## lm(formula = LowPrice ~ Acc060, data = ShuffledCars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -21.798 -10.745 - 3.390   9.009  49.507
##
## Coefficients:
##             Estimate Std. Error t value    Pr(>|t|)    
## (Intercept) 41.3835    7.5024   5.516 0.000000239 ***
## Acc060     -1.0804    0.9257  -1.167    0.246    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.9 on 108 degrees of freedom
## Multiple R-squared:  0.01246,    Adjusted R-squared:  0.003311 
## F-statistic: 1.362 on 1 and 108 DF,  p-value: 0.2457
```

3.2.11 Fourth Cars Shuffle Simulation

```
ShuffledCars <- Cars2015    ## create copy of dataset
ShuffledCars$Acc060 <- ShuffledCars$Acc060[sample(1:nrow(ShuffledCars))]
```

Make	Model	LowPrice	Acc060	ShuffledAcc060
Chevrolet	Spark	12.270	12.8	8.6
Hyundai	Accent	14.745	10.3	11.0
Kia	Rio	13.990	9.5	6.9
Mitsubishi	Mirage	12.995	12.1	10.3
Nissan	Versa Note	14.180	10.9	9.8
Dodge	Dart	16.495	9.3	6.5

3.2.12 Fourth Cars Shuffle Model Results

Recall this model was fit under an assumption of no relationship between price and acceleration time.

```
M_Cars_Shuffle <- lm(data=ShuffledCars, LowPrice~Acc060)
summary(M_Cars_Shuffle)
```

```
##
## Call:
## lm(formula = LowPrice ~ Acc060, data = ShuffledCars)
##
## Residuals:
##      Min      1Q Median      3Q     Max
## -24.382 -11.808 - 3.246   9.664  48.778
##
## Coefficients:
##             Estimate Std. Error t value    Pr(>|t|)
## (Intercept) 45.9647    7.4380   6.180 0.0000000116 ***
## Acc060      -1.6576    0.9178  -1.806    0.0737 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.76 on 108 degrees of freedom
## Multiple R-squared:  0.02932,    Adjusted R-squared:  0.02033
## F-statistic: 3.262 on 1 and 108 DF,  p-value: 0.07369
```

3.2.13 Fifth Cars Shuffle Simulation

```
ShuffledCars <- Cars2015    ## create copy of dataset
ShuffledCars$Acc060 <- ShuffledCars$Acc060[sample(1:nrow(ShuffledCars))]
```

Make	Model	LowPrice	Acc060	ShuffledAcc060
Chevrolet	Spark	12.270	12.8	8.6
Hyundai	Accent	14.745	10.3	7.7
Kia	Rio	13.990	9.5	7.7
Mitsubishi	Mirage	12.995	12.1	6.8
Nissan	Versa Note	14.180	10.9	7.9
Dodge	Dart	16.495	9.3	10.9

3.2.14 Fifth Cars Shuffle Model Results

Recall this model was fit under an assumption of no relationship between price and acceleration time.

```
M_Cars_Shuffle <- lm(data=ShuffledCars, LowPrice~Acc060)
summary(M_Cars_Shuffle)

##
## Call:
## lm(formula = LowPrice ~ Acc060, data = ShuffledCars)
##
## Residuals:
##    Min      1Q  Median      3Q     Max 
## -20.469 -11.188 - 3.441   8.841  50.951 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 37.3890    7.5361   4.961 0.00000262 ***
## Acc060      -0.5771    0.9299  -0.621    0.536    
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.97 on 108 degrees of freedom
## Multiple R-squared:  0.003554, Adjusted R-squared:  -0.005672 
## F-statistic: 0.3852 on 1 and 108 DF,  p-value: 0.5361
```

3.2.15 Code for Acc060 Simulation Investigation

```
b1 <- Cars_M_A060$coef[2] ## record value of b1 from actual data

## perform simulation
b1Sim <- rep(NA, 10000)          ## vector to hold results
ShuffledCars <- Cars2015        ## create copy of dataset
for (i in 1:10000){
  #randomly shuffle acceleration times
```

```

ShuffledCars$Acc060 <- ShuffledCars$Acc060[sample(1:nrow(ShuffledCars))]
ShuffledCars_M<- lm(data=ShuffledCars, LowPrice ~ Acc060) #fit model to shuffled data
b1Sim[i] <- ShuffledCars_M$coef[2] ## record b1 from shuffled model
}
Cars_A060SimulationResults <- data.frame(b1Sim) #save results in dataframe

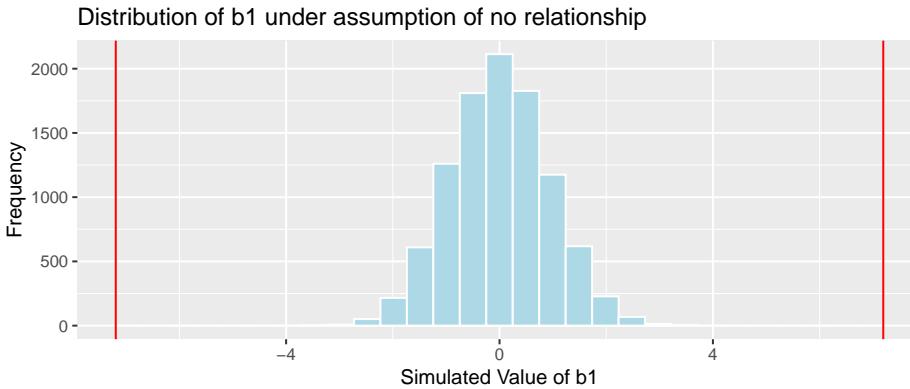
```

3.2.16 Acc060 Simulation Results

```

b1 <- Cars_M_A060$coef[2] ## record value of b1 from actual data
Cars_A060SimulationResultsPlot <- ggplot(data=Cars_A060SimulationResults, aes(x=b1Sim)) +
  geom_histogram(fill="lightblue", color="white") +
  geom_vline(xintercept=c(b1, -1*b1), color="red") +
  xlab("Simulated Value of b1") + ylab("Frequency") +
  ggtitle("Distribution of b1 under assumption of no relationship")
Cars_A060SimulationResultsPlot

```



It is extremely unlikely that we would observe a value of b_1 as extreme as -7.1933 by chance, if there is really no relationship between price and acceleration time.

3.2.17 Acc060 P-value and Conclusion

Proportion of simulations resulting in simulation value of b_2 more extreme than -7.1933.

```

mean(abs(b1Sim) > abs(b1))
## [1] 0

```

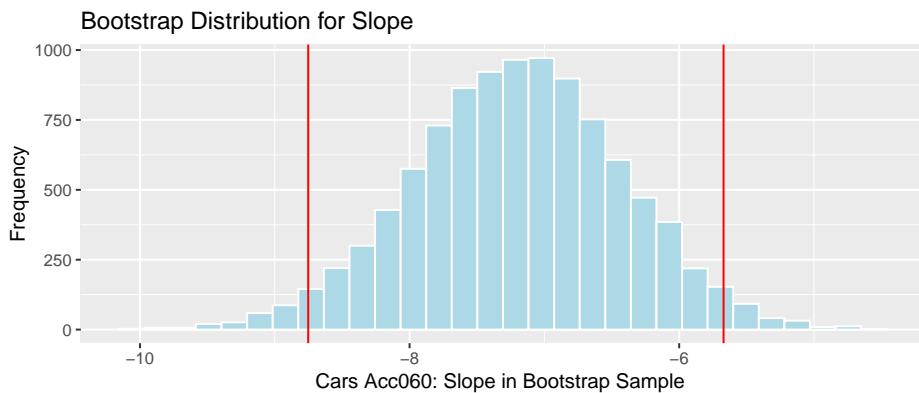
The probability of observing a value of b_1 as extreme as -7.1933 by chance, when there is no relationship between location and mercury level practically zero.

There is very strong evidence of a relationship between price and acceleration time.

3.2.18 Cars Bootstrap

```
q.025 <- quantile(Cars_Bootstrap_Results_Acc060$b1, .025)
q.975 <- quantile(Cars_Bootstrap_Results_Acc060$b1, .975)

Cars_Acc060_B_Slope_Plot + geom_vline(xintercept=c(q.025, q.975), col="red")
```

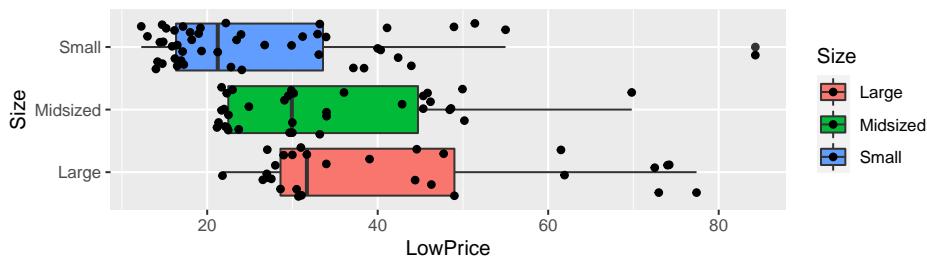


We are 95% confident that the average price of a new 2015 car decreases between -8.75 and -5.67 thousand dollars for each additional second it takes to accelerate from 0 to 60 mph.

3.3 Simulation-Based Hypothesis Test for F-Statistics

3.3.1 Relationship Price and Car Size

```
ggplot(data=Cars2015, aes(x=Size, y=LowPrice, fill=Size)) +
  geom_boxplot() + geom_jitter() + coord_flip()
```



```
Cars2015 %>% group_by(Size) %>% summarize(MeanPrice = mean(LowPrice),
                                              StDevPrice=sd(LowPrice),
                                              N=n())
## # A tibble: 3 x 4
##   Size     MeanPrice StDevPrice     N
##   <fct>      <dbl>      <dbl> <int>
## 1 Large       42.3       17.9    29
## 2 Midsized    33.2       12.0    34
## 3 Small       26.7       14.4    47
```

3.3.2 Cars Questions of Interest

1. Do the data provide evidence of a relationship between price and size of vehicle?
2. Is there evidence of a difference in average price between...
 - a) large and midsized cars?
 - b) large and small cars?
 - c) small and midsized cars?

3.3.3 Cars Price and Size Model

```
Cars_M_Size = lm(data=Cars2015, LowPrice~Size)
summary(Cars_M_Size)

##
## Call:
## lm(formula = LowPrice ~ Size, data = Cars2015)
##
## Residuals:
##     Min      1Q      Median      3Q      Max 
## -20.516 -11.190   -4.005    9.064   57.648 
## 
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)    
## (Intercept)  42.311     2.737  15.460 < 0.000000000000002 *** 
## SizeMidsized -9.098     3.725  -2.442          0.0162 *    
## SizeSmall    -15.659    3.480   -4.499          0.0000174 *** 
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##  
## Residual standard error: 14.74 on 107 degrees of freedom  
## Multiple R-squared:  0.1593, Adjusted R-squared:  0.1436  
## F-statistic: 10.14 on 2 and 107 DF,  p-value: 0.00009271
```

3.3.4 Relationship between Size and Price

1. Do the data provide evidence of a relationship between price and size of vehicle?

$$\widehat{\text{Price}} = b_0 + b_1 \times I_{\text{Midsized}} + b_2 \times I_{\text{Large}}$$

- b_0 represents expected price of large cars.
- b_1 represents expected difference in price between large and midsized cars.
- b_2 represents expected difference in price between large and small cars.

Unfortunately, none of these measure whether there is an overall relationship between price and size.

Recall that the ANOVA F-statistic combines information from all of the groups. Thus, it can be used to assess the strength of evidence of differences.

3.3.5 F-Statistic for Car Size and Price

```
Cars_A_Size <- aov(data=Cars2015, LowPrice~Size)
summary(Cars_A_Size)

##           Df Sum Sq Mean Sq F value    Pr(>F)
## Size        2   4405   2202.7   10.14 0.0000927 ***
## Residuals  107  23242    217.2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.3.6 Key Question in Car Size Investigation

Null Hypothesis: There is no relationship between price and car size.

Alternative Hypothesis: There is a relationship between price and car size.

Key Question: How likely is it that we would have obtained an F-statistic as extreme as 10.14 by chance, if there is really no relationship between price and size?

3.3.7 Simulation-Based Test for F-Statistic

We'll simulate situations where there is no relationship between size and price, and see how often we observe an F-statistic as extreme as 10.14.

Procedure:

1. Randomly shuffle the sizes of the vehicles, so that any relationship between size and price is due only to chance.
2. Fit a model, using the shuffled data, with price as the response variable, and size as the explanatory variable. Record the F-statistic.
3. Repeat steps 1 and 2 many (say 10,000) times, recording the F-statistic each time.
4. Analyze the distribution of F-statistics, simulated under the assumption that there is no relationship between size and price. Look whether the actual F-statistic we observed is consistent with the simulation results.

3.3.8 First Cars Shuffle Simulation

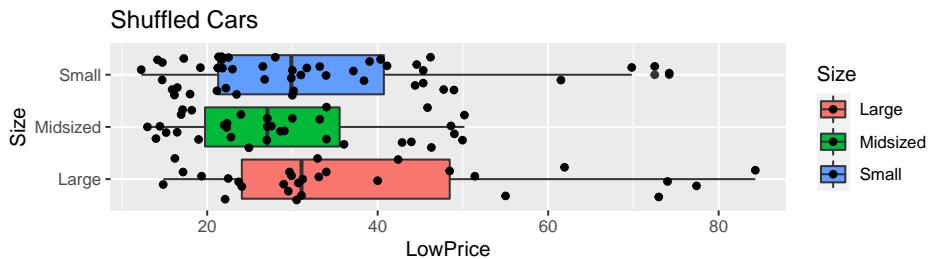
```
ShuffledCars <- Cars2015    ## create copy of dataset
ShuffledCars$Size <- ShuffledCars$Size[sample(1:nrow(ShuffledCars))]
```

Make	Model	LowPrice	Size	ShuffledSize
Chevrolet	Spark	12.270	Small	Small
Hyundai	Accent	14.745	Small	Small
Kia	Rio	13.990	Small	Midsized
Mitsubishi	Mirage	12.995	Small	Midsized
Nissan	Versa Note	14.180	Small	Small
Dodge	Dart	16.495	Small	Small

3.3.9 First Cars Shuffle Model Results

Recall this model was fit under an assumption of no relationship between price and size.

```
ggplot(data=ShuffledCars, aes(x=Size, y=LowPrice, fill=Size)) +
  geom_boxplot() + geom_jitter() + coord_flip() + ggtitle("Shuffled Cars")
```



```
Cars_A_Size_Shuffle <- aov(data=ShuffledCars, LowPrice~Size)
summary(Cars_A_Size_Shuffle)
```

```
##           Df Sum Sq Mean Sq F value Pr(>F)
## Size        2   1328   663.8   2.699 0.0719 .
## Residuals  107  26320   246.0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.3.10 Second Cars Shuffle Simulation

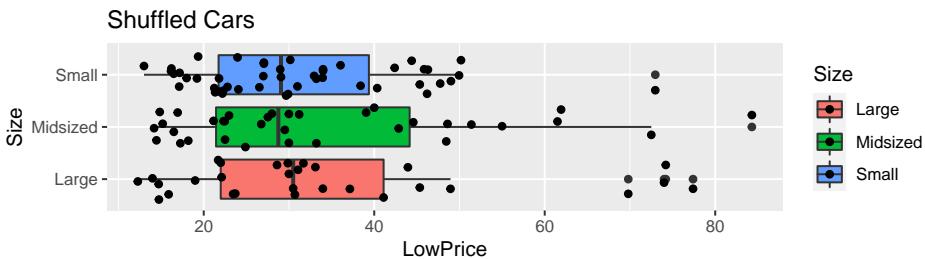
```
ShuffledCars <- Cars2015      ## create copy of dataset
ShuffledCars$Size <- ShuffledCars$Size[sample(1:nrow(ShuffledCars))]
```

Make	Model	LowPrice	Size	ShuffledSize
Chevrolet	Spark	12.270	Small	Large
Hyundai	Accent	14.745	Small	Large
Kia	Rio	13.990	Small	Large
Mitsubishi	Mirage	12.995	Small	Small
Nissan	Versa Note	14.180	Small	Midsized
Dodge	Dart	16.495	Small	Midsized

3.3.11 Second Cars Shuffle Model Results

Recall this model was fit under an assumption of no relationship between price and size.

```
ggplot(data=ShuffledCars, aes(x=Size, y=LowPrice, fill=Size)) +
  geom_boxplot() + geom_jitter() + coord_flip() + ggtitle("Shuffled Cars")
```



```
Cars_A_Size_Shuffle <- aov(data=ShuffledCars, LowPrice~Size)
summary(Cars_A_Size_Shuffle)
```

```
##           Df Sum Sq Mean Sq F value Pr(>F)
## Size        2   231   115.6   0.451  0.638
## Residuals 107 27417   256.2
```

3.3.12 Third Cars Shuffle Simulation

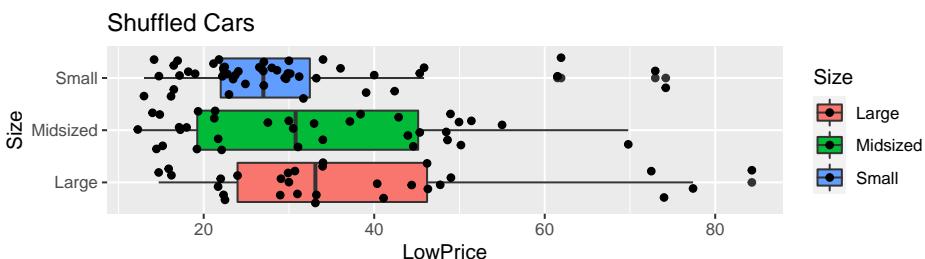
```
ShuffledCars <- Cars2015    ## create copy of dataset
ShuffledCars$Size <- ShuffledCars$Size[sample(1:nrow(ShuffledCars))]
```

Make	Model	LowPrice	Size	ShuffledSize
Chevrolet	Spark	12.270	Small	Midsized
Hyundai	Accent	14.745	Small	Small
Kia	Rio	13.990	Small	Midsized
Mitsubishi	Mirage	12.995	Small	Small
Nissan	Versa Note	14.180	Small	Small
Dodge	Dart	16.495	Small	Small

3.3.13 Third Cars Shuffle Model Results

Recall this model was fit under an assumption of no relationship between price and size.

```
ggplot(data=ShuffledCars, aes(x=Size, y=LowPrice, fill=Size)) +
  geom_boxplot() + geom_jitter() + coord_flip() + ggtitle("Shuffled Cars")
```



```
Cars_A_Size_Shuffle <- aov(data=ShuffledCars, LowPrice~Size)
summary(Cars_A_Size_Shuffle)

##           Df Sum Sq Mean Sq F value Pr(>F)
## Size          2   1134   566.8   2.287  0.106
## Residuals    107  26514   247.8
```

3.3.14 Fourth Cars Shuffle Simulation

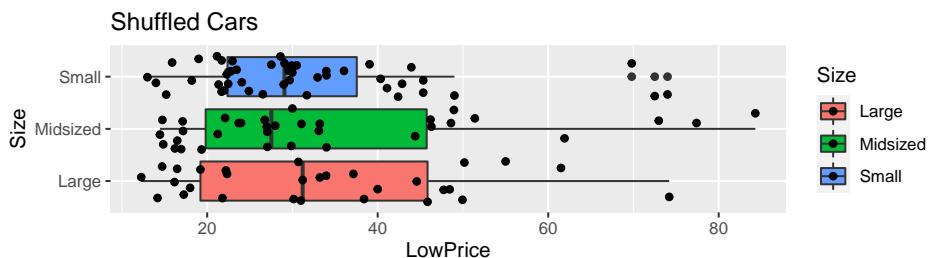
```
ShuffledCars <- Cars2015      ## create copy of dataset
ShuffledCars$Size <- ShuffledCars$Size[sample(1:nrow(ShuffledCars))]
```

Make	Model	LowPrice	Size	ShuffledSize
Chevrolet	Spark	12.270	Small	Large
Hyundai	Accent	14.745	Small	Midsized
Kia	Rio	13.990	Small	Small
Mitsubishi	Mirage	12.995	Small	Small
Nissan	Versa Note	14.180	Small	Large
Dodge	Dart	16.495	Small	Midsized

3.3.15 Fourth Cars Shuffle Model Results

Recall this model was fit under an assumption of no relationship between price and size.

```
ggplot(data=ShuffledCars, aes(x=Size, y=LowPrice, fill=Size)) +
  geom_boxplot() + geom_jitter() + coord_flip() + ggtitle("Shuffled Cars")
```



```
Cars_A_Size_Shuffle <- aov(data=ShuffledCars, LowPrice~Size)
summary(Cars_A_Size_Shuffle)
```

```
##           Df Sum Sq Mean Sq F value Pr(>F)
## Size          2   128   64.21   0.25   0.78
## Residuals    107  27519  257.19
```

3.3.16 Fifth Cars Shuffle Simulation

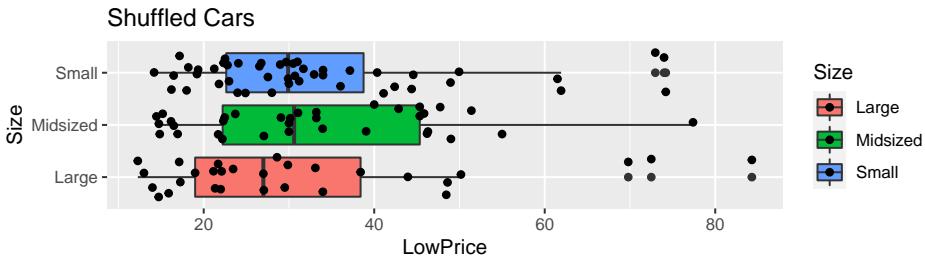
```
ShuffledCars <- Cars2015    ## create copy of dataset
ShuffledCars$Size <- ShuffledCars$Size[sample(1:nrow(ShuffledCars))]
```

Make	Model	LowPrice	Size	ShuffledSize
Chevrolet	Spark	12.270	Small	Large
Hyundai	Accent	14.745	Small	Midsized
Kia	Rio	13.990	Small	Large
Mitsubishi	Mirage	12.995	Small	Large
Nissan	Versa Note	14.180	Small	Small
Dodge	Dart	16.495	Small	Midsized

3.3.17 Fifth Cars Shuffle Model Results

Recall this model was fit under an assumption of no relationship between price and size.

```
ggplot(data=ShuffledCars, aes(x=Size, y=LowPrice, fill=Size)) +
  geom_boxplot() + geom_jitter() + coord_flip() + ggtitle("Shuffled Cars")
```



```
Cars_A_Size_Shuffle <- aov(data=ShuffledCars, LowPrice~Size)
summary(Cars_A_Size_Shuffle)
```

```
##           Df Sum Sq Mean Sq F value Pr(>F)
## Size        2    46   22.86   0.089  0.915
## Residuals 107 27602  257.96
```

3.3.18 Car Size F-Statistic Simulation

```
Fstat <- summary(Cars_M_Size)$fstatistic[1] ## record value of F-statistic from actual data

## perform simulation
FSim <- rep(NA, 10000)          ## vector to hold results
ShuffledCars <- Cars2015       ## create copy of dataset
for (i in 1:10000){
```

```

#randomly shuffle acceleration times
ShuffledCars$Size <- ShuffledCars$Size[sample(1:nrow(ShuffledCars))]
ShuffledCars_M<- lm(data=ShuffledCars, LowPrice ~ Size) #fit model to shuffled data
FSim[i] <- summary(ShuffledCars_M)$fstatistic[1] ## record F from shuffled model
}
CarSize_SimulationResults <- data.frame(FSim) #save results in dataframe

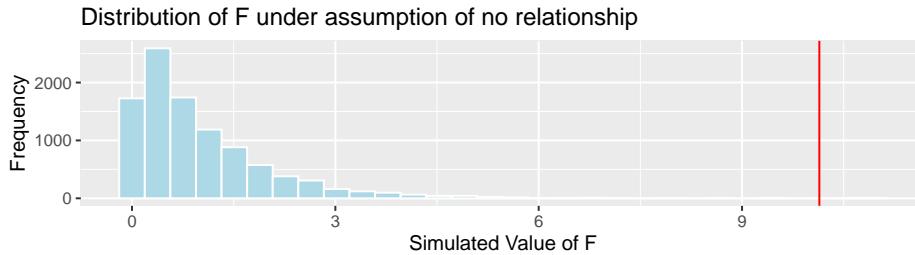
```

3.3.19 F-statistic for Size Simulation Results

```

CarSize_SimulationResults_Plot <- ggplot(data=CarSize_SimulationResults, aes(x=FSim))
  geom_histogram(fill="lightblue", color="white") + geom_vline(xintercept=c(Fstat), c
  xlab("Simulated Value of F") + ylab("Frequency") + ggtitle("Distribution of F under
CarSize_SimulationResults_Plot

```



p-value:

```
mean(FSim > Fstat)
```

```
## [1] 0.0002
```

The data provide strong evidence of a relationship between price and size.

3.3.20 Differences Between Different Sizes

Now that we have evidence that car price is related to size, we might want to know which sizes differ from each other.

Is there evidence of a difference in average price between...

- a) large and midsized cars?
- b) large and small cars?
- c) small and midsized cars?

3.3.21 Regression Coefficients for Tests Between Sizes

$$\widehat{\text{Price}} = b_0 + b_1 \times I_{\text{Midsized}} + b_2 \times I_{\text{Large}}$$

- b_0 represents expected price of large cars.
- b_1 represents expected difference in price between large and midsized cars.
- b_2 represents expected difference in price between large and small cars.

Thus, we can answer each question by looking at the appropriate regression coefficient.

- a) large and midsized cars? (b_1)
- b) large and small cars? (b_2)
- c) small and midsized cars? ($b_1 - b_2$)

3.3.22 Simulation for Differences between Types of Cars

We'll simulate situations where there is no relationship between size and price, and see how often we observe results for b_1 , b_2 , and $b_1 - b_2$ as extreme as we did in the actual data.

Procedure:

1. Randomly shuffle the sizes of the vehicles, so that any relationship between size and price is due only to chance.
2. Fit a model, using the shuffled data, with price as the response variable, and size as the explanatory variable. Record the values of b_1 , b_2 , and $b_1 - b_2$.
3. Repeat steps 1 and 2 many (say 10,000) times, recording the values of b_1 , b_2 , and $b_1 - b_2$ each time.
4. Analyze the distribution of b_1 , b_2 , $b_1 - b_2$, simulated under the assumption that there is no relationship between size and price. Look whether the actual values we observed are consistent with the simulation results.

3.3.23 Code for Simulation-Based Test of Prices by Size

```

b1 <- Cars_M_Size$coefficients[2]  #record b1 from actual data
b2 <- Cars_M_Size$coefficients[3]  #record b2 from actual data

## perform simulation
b1Sim <- rep(NA, 10000)          ## vector to hold results
b2Sim <- rep(NA, 10000)          ## vector to hold results
ShuffledCars <- Cars2015      ## create copy of dataset
for (i in 1:10000){
  #randomly shuffle acceleration times
  ShuffledCars$Size <- ShuffledCars$Size[sample(1:nrow(ShuffledCars))]
}

```

```

ShuffledCars_M<- lm(data=ShuffledCars, LowPrice ~ Size)    #fit model to shuffled data
b1Sim[i] <- ShuffledCars_M$coefficients[2]    ## record b1 from shuffled model
b2Sim[i] <- ShuffledCars_M$coefficients[3]    ## record b2 from shuffled model
}
Cars_Size2_SimulationResults <- data.frame(b1Sim, b2Sim)    #save results in dataframe

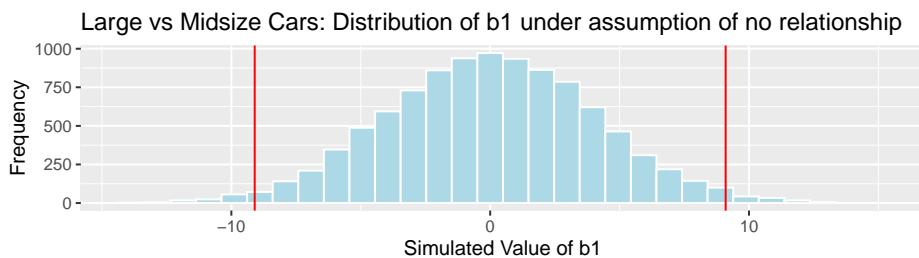
```

3.3.24 Car Size Simulation-Based Results for b_1

```

Cars_Size2_SimulationResultsPlot_b1 <- ggplot(data=Cars_Size2_SimulationResults, aes(x=
  geom_histogram(fill="lightblue", color="white") +
  geom_vline(xintercept=c(b1, -1*b1), color="red") +
  xlab("Simulated Value of b1") + ylab("Frequency") +
  ggtitle("Large vs Midsize Cars: Distribution of b1 under assumption of no relationship")
Cars_Size2_SimulationResultsPlot_b1

```



p-value:

```
mean(abs(b1Sim)>abs(b1))
```

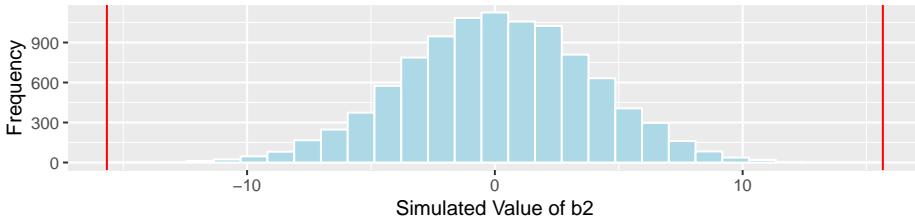
```
## [1] 0.0249
```

3.3.25 Car Size Simulation-Based Results for b_2

```

Cars_Size2_SimulationResultsPlot_b2 <- ggplot(data=Cars_Size2_SimulationResults, aes(x=
  geom_histogram(fill="lightblue", color="white") +
  geom_vline(xintercept=c(b2, -1*b2), color="red") +
  xlab("Simulated Value of b2") + ylab("Frequency") +
  ggtitle("Large vs Small Cars: Distribution of b2 under assumption of no relationship")
Cars_Size2_SimulationResultsPlot_b2

```

Large vs Small Cars: Distribution of b_2 under assumption of no relationship

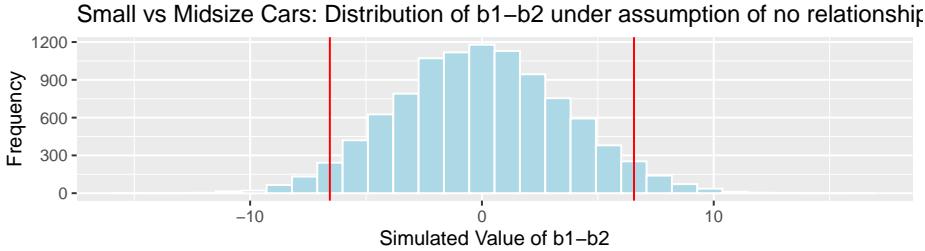
p-value:

```
mean(abs(b2Sim)>abs(b2))
```

```
## [1] 0
```

3.3.26 Car Size Simulation-Based Results for $b_1 - b_2$

```
Cars_Size2_SimulationResultsPlot_b1_b2 <- ggplot(data=Cars_Size2_SimulationResults, aes(x=b1Sim-b2Sim))
  geom_histogram(fill="lightblue", color="white") + geom_vline(xintercept=c(b1-b2, -1*(b1-b2)), col="red")
  xlab("Simulated Value of  $b_1 - b_2$ ") + ylab("Frequency") +
  ggtitle("Small vs Midsize Cars: Distribution of  $b_1 - b_2$  under assumption of no relationship")
Cars_Size2_SimulationResultsPlot_b1_b2
```



p-value:

```
mean(abs(b1Sim-b2Sim)>abs(b1-b2))
```

```
## [1] 0.0711
```

3.3.27 Conclusions and Bonferroni Correction

- We might normally conclude that there is evidence of differences in group means in the p-value is less than 0.05. However, since we are performing multiple tests simultaneously, there is an increased chance that at least one of them will yield a small p-value just by chance. Thus, we should be more strict in deciding what constitutes evidence against the null hypothesis.

- A common rule (known as the **Bonferroni correction**) is to divide the values usually used as criteria for evidence by the number of tests. In this example:
 - Here, we would say there is some evidence of differences between groups if the p-value is less than $0.10/3=0.0333$.
 - We would say there is strong evidence of differences if the p-value is less than $0.05/3=0.0167$

Comparison	Coefficient	p-value	Evidence of Difference
large vs midsize	b_1	0.0249	Some evidence
large vs small	b_2	0	Strong evidence
small vs midsize	$b_1 - b_2$	0.0711	No evidence

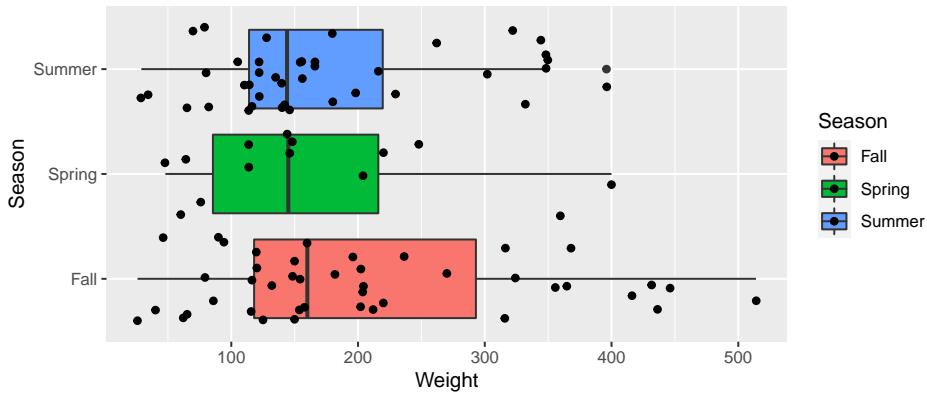
3.3.28 Summary of Tests Between Multiple Groups

When testing for differences between more than two groups:

1. Perform an overall test, using the F-statistic. A large F-statistic and small p-value tell us there is evidence of differences between at least some of the groups.
2. If the F-tests yields evidence of differences, perform tests on individual model coefficients to determine which groups differ. Use a more strict cutoff criteria, such as the Bonferroni correction.

3.3.29 Bear Weights by Season

```
ggplot(data=Bears_Subset, aes(y=Weight, x=Season, fill=Season)) +
  geom_boxplot() + geom_jitter() + coord_flip()
```



3.3.30 Bear Weights by Season Model

```
Bears_M_Season <- lm(data=Bears_Subset, Weight~Season)
summary(Bears_M_Season)

##
## Call:
## lm(formula = Weight ~ Season, data = Bears_Subset)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -178.84  -79.84  -29.02   54.98  309.16 
## 
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)    
## (Intercept) 204.84     17.16  11.939 <0.0000000000000002 ***
## SeasonSpring -37.27     34.62  -1.076     0.284    
## SeasonSummer -29.81     24.71  -1.206     0.231    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 112.5 on 94 degrees of freedom
## Multiple R-squared:  0.02034,    Adjusted R-squared:  -0.0005074 
## F-statistic: 0.9757 on 2 and 94 DF,  p-value: 0.3807
```

3.3.31 F-Statistic for Bear Weights by Season

```
Bears_A_Season <- aov(data=Bears_Subset, Weight~Season)
summary(Bears_A_Season)
```

```

##           Df  Sum Sq Mean Sq F value Pr(>F)
## Season       2   24699   12350   0.976  0.381
## Residuals   94 1189818   12658

```

3.3.32 Hypotheses for Bears Seasons F-Test

Null Hypothesis: Mean weight of bears is the same in each season.

Alternative Hypothesis: Mean weight of bears differs between seasons.

Key Question: What is the probability of observing an F-statistic as extreme as 0.976 if there is really no relationship between weight and season?

3.3.33 Simulation-Based Test for Bears F-Statistic

We'll simulate situations where there is no relationship between size and price, and see how often we observe an F-statistic as extreme as 0.976.

Procedure:

1. Randomly shuffle the seasons, so that any relationship between weight and season is due only to chance.
2. Fit a model, using the shuffled data, with weight as the response variable, and season as the explanatory variable. Record the F-statistic.
3. Repeat steps 1 and 2 many (say 10,000) times, recording the F-statistic each time.
4. Analyze the distribution of F-statistics, simulated under the assumption that there is no relationship between season and weight. Look whether the actual F-statistic we observed is consistent with the simulation results.

3.3.34 Bears F-Statistic Simulation

```

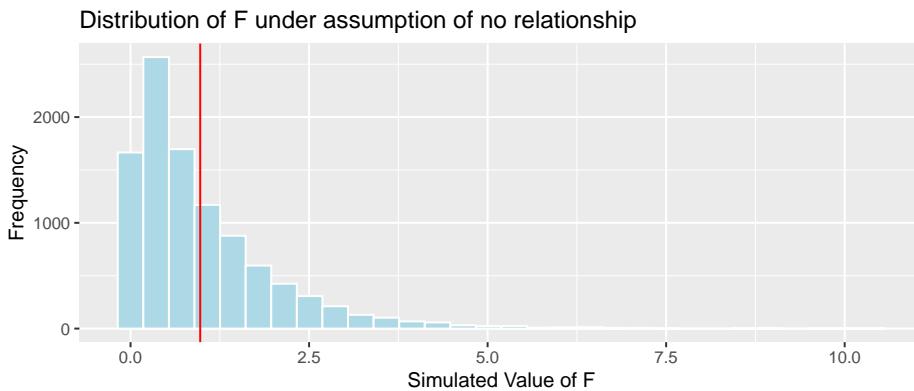
Fstat <- summary(Bears_M_Season)$fstatistic[1] ## record value of F-statistic from actual data

## perform simulation
FSim <- rep(NA, 10000)          ## vector to hold results
ShuffledBears <- Bears_Subset    ## create copy of dataset
for (i in 1:10000){
  #randomly shuffle acceleration times
  ShuffledBears$Season <- ShuffledBears$Season[sample(1:nrow(ShuffledBears))]
  ShuffledBears_M<- lm(data=ShuffledBears, Weight ~ Season)  #fit model to shuffled data
  FSim[i] <- summary(ShuffledBears_M)$fstatistic[1] ## record F from shuffled model
}

```

```
Bears_Seasons_SimulationResults <- data.frame(FSim) #save results in dataframe
```

3.3.35 F-statistic for Bears Season Simulation



```
mean(FSim > Fstat)
```

```
## [1] 0.3777
```

It is not at all unusual to observe an F-statistic as extreme or more extreme than we did if there is really no relationship between weight and season.

There is no evidence that average bear weights differ between seasons.

3.3.36 Don't Accept Null Hypothesis

- In the previous example, we concluded that there is no evidence that average bear weights differ between seasons.
 - This is different than saying that bear weights are the same in each season. Why would it be inappropriate to say this?

3.3.37 Comparison of Weights by Season

Season	MeanWeight	StDevWeight	N
Fall	204.8372	125.71414	43
Spring	167.5714	108.74155	14
Summer	175.0250	97.70796	40

3.3.38 Don't Accept Null Hypothesis (Cont.)

The data do show differences in average weight between seasons. It's just that we can't rule out the possibility that these differences are due to chance alone.

- A hypothesis test can only tell us the strength of evidence against the null hypothesis. The absence of evidence against the null hypothesis should not be interpreted as evidence for the null hypothesis.
- We should never say that the data support/prove/confirm the null hypothesis.
- We can only say that the data do not provide evidence against the null hypothesis.

3.4 “Theory-Based” Tests and Intervals in R

3.4.1 R’s Theory-Based Tests and Intervals

In the previous sections, we've used simulation to obtain confidence intervals and perform hypothesis tests.

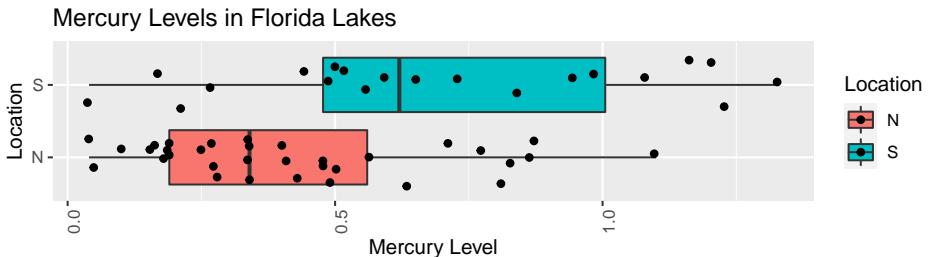
In fact, R provides intervals and tests directly, without performing simulations. These are calculated using theory-based approximation formulas that often (but not always) lead to the same conclusions as the simulation-based methods.

In this section, we'll look at how to obtain these tests and intervals in R.

In Chapter 5, we'll explore more of the reasoning behind these approaches.

3.4.2 Difference between North and South?

```
ggplot(data=FloridaLakes, aes(x=Location, y=AvgMercury, fill=Location)) +
  geom_boxplot() + geom_jitter() + ggtitle("Mercury Levels in Florida Lakes") +
  xlab("Location") + ylab("Mercury Level") + theme(axis.text.x = element_text(angle =
```



```
FloridaLakes %>% group_by(Location) %>% summarize(MeanHg=mean(AvgMercury),
                                                       StDevHg=sd(AvgMercury),
                                                       N=n())
```

```
## # A tibble: 2 x 4
##   Location MeanHg  StDevHg     N
##   <fct>    <dbl>    <dbl> <int>
## 1 N        0.425    0.270    33
## 2 S        0.696    0.384    20
```

3.4.3 Parameters and Statistics

- A **statistic** is a quantity calculated from sample data (such as \bar{x}, b_0, b_1, \dots , etc.)
- A **parameter** is a quantity pertaining to the entire population. Parameters are assumed to be fixed, but unknown.
- It is common to use Greek letters to represent parameters. For example, we'll use β_0, β_1, \dots to represent the parameters corresponding to b_0, b_1, \dots
 - Example: In our sample of 53 Florida lakes, the average difference in mercury levels in northern and southern lakes was $b_1 = 0.27195$. We can use this make conclusions about β_1 , the average difference in mercury levels between all northern and southern lakes.
- Confidence intervals and hypothesis tests lead to statements about parameters, based on information provided by statistics.

3.4.4 Hypotheses in Florida Lakes Example

Null Hypothesis: There is no difference in mean mercury level between lakes in Northern Florida and Southern Florida. This is equivalent to saying $\beta_1 = 0$.

Alternative Hypothesis: The mean mercury level differs between lakes in Northern and Southern Florida. This is equivalent to saying $\beta_1 \neq 0$.

3.4.5 Model for Lakes R Output

```
Lakes_M <- lm(data=FloridaLakes, AvgMercury ~ Location)
summary(Lakes_M)

##
## Call:
## lm(formula = AvgMercury ~ Location, data = FloridaLakes)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.65650 -0.23455 -0.08455  0.24350  0.67545
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)
## (Intercept) 0.42455   0.05519  7.692 0.000000000441 ***
## LocationS   0.27195   0.08985  3.027 0.00387 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3171 on 51 degrees of freedom
## Multiple R-squared:  0.1523, Adjusted R-squared:  0.1357
## F-statistic: 9.162 on 1 and 51 DF,  p-value: 0.003868
```

The quantity $\text{Pr}(>|t|)$ in the coefficients table contains p-values pertaining to the test of the null hypothesis that that parameter is 0.

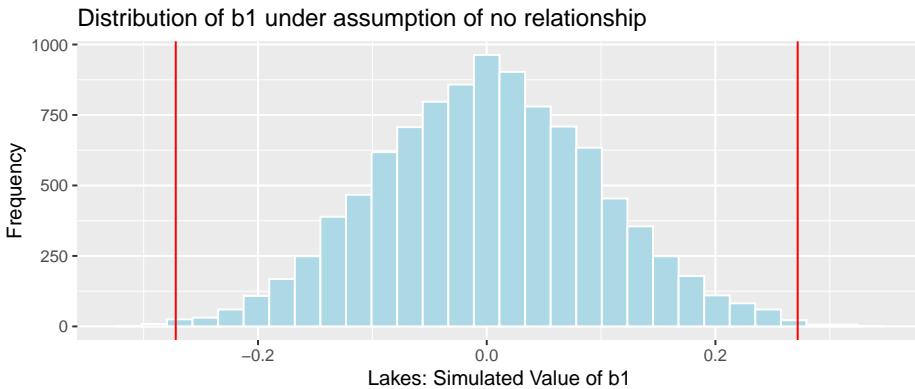
3.4.6 Lakes Model Model p-value Interpretations

```
summary(Lakes_M)$coefficients

##             Estimate Std. Error t value     Pr(>|t|)
## (Intercept) 0.4245455 0.05519303 7.692012 0.0000000004414948
## LocationS   0.2719545 0.08984774 3.026838 0.0038681284084829
```

- The p-value of 0.00387 on the line “LocationS” provides strong evidence against the null hypothesis $\beta_1 = 0$. Thus, there is evidence of a difference in mercury levels between northern and southern lakes. Recall, we obtained a p-value of 0.004 using simulation)
- The p-value on ≈ 0 on the line (Intercept) tells us there is strong evidence against the null hypothesis $\beta_0 = 0$. That is, there is strong evidence that the mean mercury level in northern lakes is not 0. Of course, we already knew that already.

3.4.7 Recall Lakes Simulation Results



Simulation-based p-value:

```
## [1] 0.0031
```

3.4.8 Lakes Model Model p-value Interpretations

```
summary(Lakes_M)$coefficients

##                   Estimate Std. Error   t value      Pr(>|t|)    
## (Intercept) 0.4245455 0.05519303 7.692012 0.0000000004414948
## LocationS   0.2719545 0.08984774 3.026838 0.0038681284084829
```

- The p-value on ≈ 0 on the line (Intercept) tells us there is strong evidence against the null hypothesis $\beta_0 = 0$. That is, there is strong evidence that the mean mercury level in northern lakes is not 0. Of course, we already knew that already.

3.4.9 Confidence Intervals in R

The `confint()` command returns “theory-based” confidence intervals for model parameters β_0 and β_1 .

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

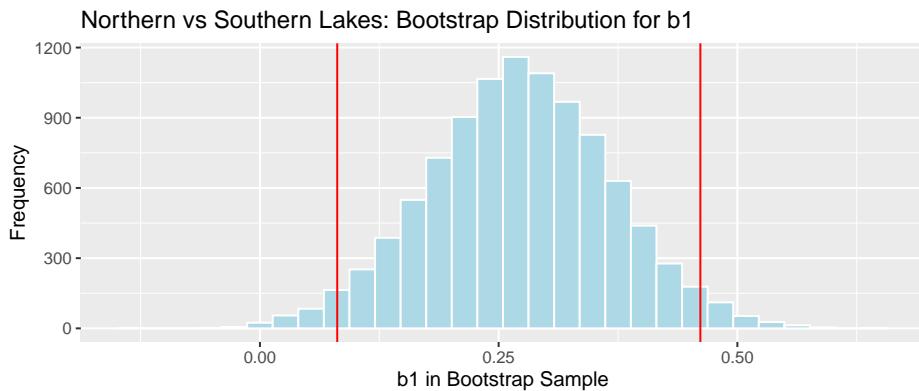
```
##             2.5 %    97.5 %
## (Intercept) 0.31374083 0.5353501
## LocationS   0.09157768 0.4523314
```

We are 95% confident that the mean mercury level in northern lakes is between 0.31 and 0.54 ppm.

We are 95% confident that the mean mercury level in southern lakes is between 0.09 and 0.45 ppm higher than in northern lakes.

3.4.10 Recall Lakes Bootstrap CI for Difference

```
##      2.5%      97.5%
## 0.08095682 0.46122992
NS_Lakes_Bootstrap_Plot_b1 + geom_vline(xintercept = c(q.025, q.975), color="red")
```



We are 95% confident the average mercury level in Southern Lakes is between 0.08 and 0.46 ppm higher than in Northern Florida.

3.4.11 Interpretations of p-values in Cars Example

```
summary(Cars_M_A060)$coefficients
```

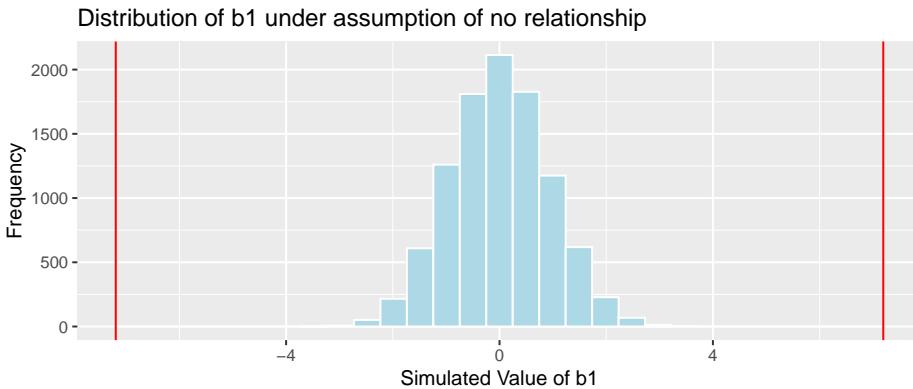
	Estimate	Std. Error	t value	Pr(> t)
## (Intercept)	89.903579	5.0523246	17.79450	0.00000000000000000000000000000000688871
## Acc060	-7.193339	0.6234005	-11.53887	0.000000000000000000000000000000014854277120867462701

The p-value of ≈ 0 on the Acc060 line provides strong evidence against the null hypothesis that $\beta_1 = 0$. (Recall, β_1 represents the slope of the regression line pertaining to all cars.) Thus, there is strong evidence of a relationship between car price and acceleration time.

The p-value of ≈ 0 on the (Intercept) line provides strong evidence against the null hypothesis that $\beta_0 = 0$. This is not meaningful, since the intercept has no meaningful interpretation.

3.4.12 Comparison to Simulation-Based P-value

```
Cars_A060SimulationResultsPlot
```



Simulation-Based p-value:

```
mean(abs(Cars_A060SimulationResults$b1Sim) > abs(coef(Cars_M_A060)[2]))  
## [1] 0
```

3.4.13 Confidence Interval in Cars Example

```
confint(Cars_M_A060)
```

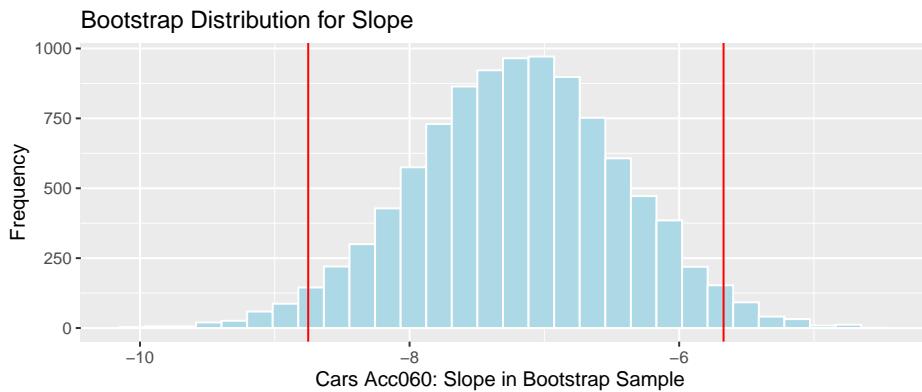
```
##              2.5 %    97.5 %
## (Intercept) 79.888995 99.918163
## Acc060      -8.429027 -5.957651
```

We are 95% confident that the mean price of a car decreases between 5.96 and 8.43 thousand dollars for each additional second it takes to accelerate from 0 to 60 mph.

3.4.14 Comparison to Bootstrap CI

```
q.025 <- quantile(Cars_Bootstrap_Results_Acc060$b1, .025)
q.975 <- quantile(Cars_Bootstrap_Results_Acc060$b1, .975)

Cars_Acc060_B_Slope_Plot + geom_vline(xintercept=c(q.025, q.975), col="red")
```



We are 95% confident that the average price of a new 2015 car decreases between -8.75 and -5.67 thousand dollars for each additional second it takes to accelerate from 0 to 60 mph.

3.4.15 Price Differences between Car Sizes

Recall this model was fit under an assumption of no relationship between price and size.



3.4.16 Theory-Based F-Test for Cars

Null Hypothesis: There are no differences in average price between small, medium, and large cars.

Alternative Hypothesis: There are differences in average price between the different car sizes.

```
summary(Cars_A_Size)
```

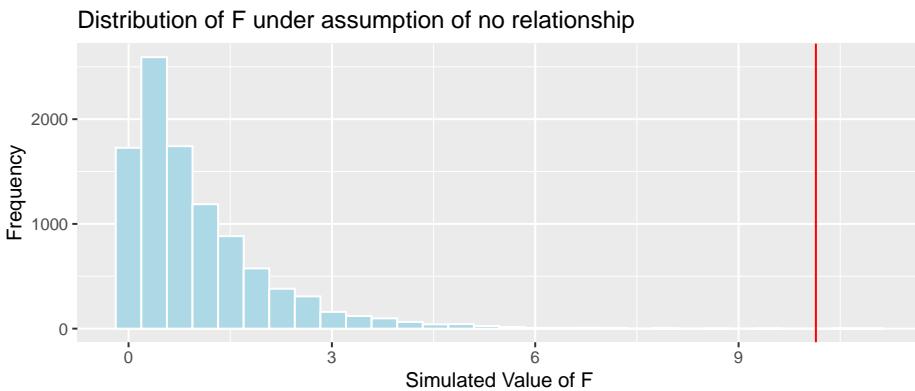
	Df	Sum Sq	Mean Sq	F value	Pr(>F)
## Size	2	4405	2202.7	10.14	0.0000927 ***

```
## Residuals    107  23242   217.2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The p-value of 0.0000927 provides strong evidence against the null hypothesis. There is evidence of differences in average price between types of cars.

3.4.17 Comparison to Simulation-Based F-Test

CarSize_SimulationResults_Plot



Simulation-based p-value:

```
## [1] 0.0002
```

3.4.18 Comparing Sizes

Having established that some sizes differ from one-another, we can obtain p-values for tests comparing individual sizes using the `pairwise.t.test()` command.

```
pairwise.t.test(Cars2015$LowPrice, Cars2015$Size, p.adj="none")
```

```
##
##  Pairwise comparisons using t tests with pooled SD
##
## data:  Cars2015$LowPrice and Cars2015$Size
##
##          Large      Midsized
## Midsized 0.016      -
## Small     0.000017 0.051
##
## P value adjustment method: none
```

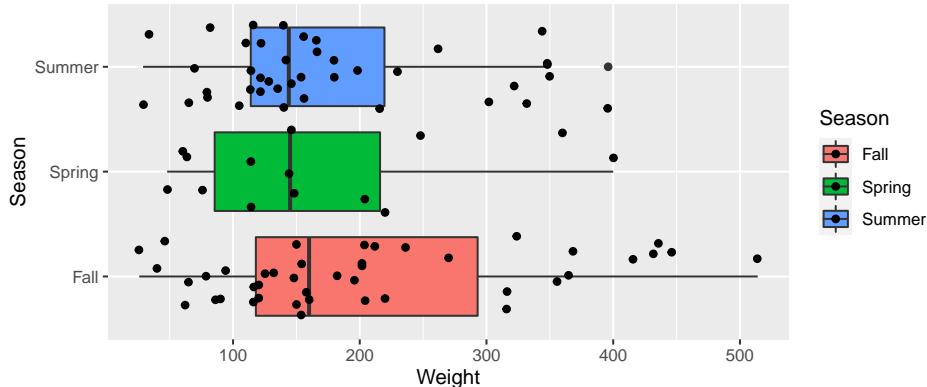
We should account for performing multiple tests via the Bonferroni correction.

3.4.19 Comparision to Simulation-Based Pairwise Tests

Comparison	Coefficient	p-value	Evidence of Difference
large vs midsize	b_1	0.0249	Some evidence
large vs small	b_2	0	Strong evidence
small vs midsize	$b_1 - b_2$	0.0711	No evidence

3.4.20 Bear Weights by Season

```
ggplot(data=Bears_Subset, aes(y=Weight, x=Season, fill=Season)) +
  geom_boxplot() + geom_jitter() + coord_flip()
```



3.4.21 Bears Seasons F-Test

Null Hypothesis: Mean weight of bears is the same in each season.

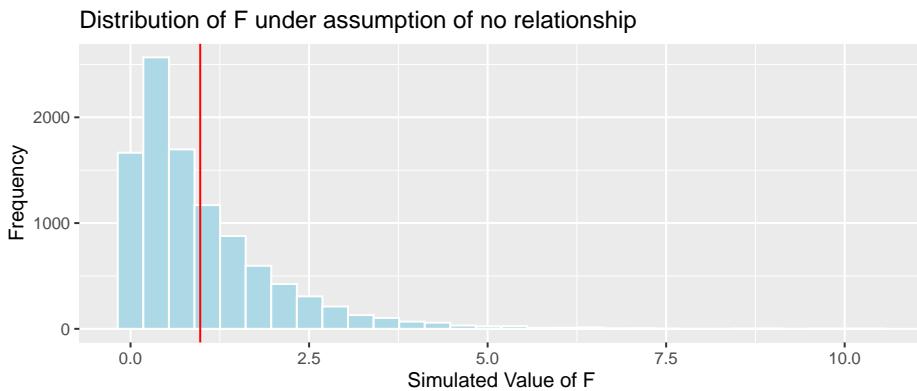
Alternative Hypothesis: Mean weight of bears differs between seasons.

```
Bears_A_Season <- aov(data=Bears_Subset, Weight~Season)
summary(Bears_A_Season)
```

```
##           Df Sum Sq Mean Sq F value Pr(>F)
## Season       2   24699   12350    0.976  0.381
## Residuals  94 1189818   12658
```

The large p-value means there is no evidence of differences in average weight between seasons.

3.4.22 Comparison to Simulation-Based Test



```
## [1] 0.3777
```

3.4.23 Bears Age and Weight Interaction Model

```
summary(Bear_M_Age_Sex_Int)

##
## Call:
## lm(formula = Weight ~ Age * Sex, data = Bears_Subset)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -207.583  -38.854   -9.574   23.905  174.802
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)
## (Intercept) 70.4322   17.7260   3.973 0.000219 ***
## Age          3.2381    0.3435   9.428 0.000000000000765 ***
## Sex2        -31.9574   35.0314  -0.912 0.365848
## Age:Sex2     -1.0350    0.6237  -1.659 0.103037
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 70.18 on 52 degrees of freedom
## (41 observations deleted due to missingness)
## Multiple R-squared:  0.6846, Adjusted R-squared:  0.6664
## F-statistic: 37.62 on 3 and 52 DF,  p-value: 0.0000000000004552
```

3.4.24 Model and Interpretations

$$\widehat{\text{Weight}} = b_0 + b_1 \times \text{Age} + b_2 \times \text{I}_{\text{Female}} + b_3 \times \text{Age} \times \text{I}_{\text{Female}}$$

Sex	Pred. Weight
M	$b_0 + b_1 \times \text{Age}$
F	$(b_0 + b_2) + (b_1 + b_3) \times \text{Age}$

Interpretations:

b_0 : expected weight of a male bear at birth (caution:extrapolation)

b_1 : expected weight gain per month for male bears

b_2 : expected difference in weight between female and male bears at birth (caution:extrapolation)

b_3 : expected difference in monthly weight gain for female bears, compared to male bears

3.4.25 Bears Hypothesis Tests from R Outpu

```
summary(Bears_M_Age_Sex_Int)$coefficients
```

```
##             Estimate Std. Error     t value    Pr(>|t|) 
## (Intercept) 70.432231 17.7259878 3.9733882 0.0002192800056185228
## Age         3.238137  0.3434770  9.4275225 0.00000000000007646023
## Sex2       -31.957367 35.0314208 -0.9122487 0.3658478717994073648
## Age:Sex2    -1.035010  0.6236905 -1.6594923 0.1030365993379950412
```

- The p-value of 0.0002 provides evidence against the null hypothesis $\beta_0 = 0$. There is strong evidence that male bears weigh more than 0 lbs at birth. (duh!)
- The p-value of ≈ 0 provides strong evidence that $\beta_1 \neq 0$. There is strong evidence that male bears gain weight as they age.
- The p-value of ≈ 0.365 indicates it is plausible that $\beta_2 = 0$. It is plausible that there is no difference in mean weight of male bears, compared to female bears in the fall.
- The p-value of 0.10 provides weak evidence against the null hypothesis $\beta_3 = 0$. There is at least a little evidence of interaction between age and sex, but the evidence is weak, and should be considered in context of other information.

3.4.26 Confidence Intervals in Bears Int. Model

```
confint(Bears_M_Age_Sex_Int)
```

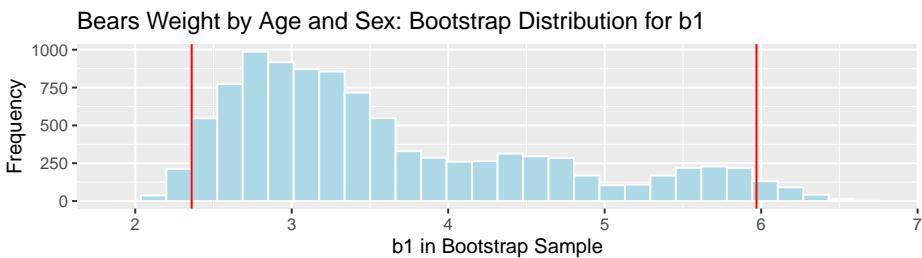
```
##             2.5 %      97.5 %
## (Intercept) 34.862434 106.002027
## Age          2.548900  3.927374
## Sex2        -102.253056 38.338322
## Age:Sex2     -2.286536  0.216517
```

- We are 95% confident that male bears weigh between 35 and 106 lbs at birth, on average. This, however, is an extrapolation, and should not be given much attention.
- We are 95% confident that $2.55 \leq \beta_1 \leq 3.93$. That is, on average male bears gain between 2.5 and 3.9 lbs per month.
- We are 95% confident that $-102.25 \leq \beta_2 \leq 28.34$. That is, female bears weight between 102 lbs less, and 38 lbs more than male bears on average, at birth. Again, this is an extrapolation.
- The interaction effect is harder to interpret, but the fact that the interval contains 0, indicates that it is plausible that there is no interaction at all, and that male bears and female bears could plausibly gain weight at the same rate.

3.4.27 Comparison to Bootstrap CI for β_1

The average weight gain per month for male bears is represented by b_1 .

```
##      2.5%      97.5%
## 2.360746 5.970272
```



We are 95% confident that male bears gain between 2.36 and 5.97 pounds per month, on average.

The simulation-based and theory-based methods disagree! The theory-based methods only work when the sampling distribution is symmetric and bell-shaped.

3.5 Responsible Use of Hypothesis Testing and p-values

3.5.1 Statistical Significance vs Practical Importance

- “(S)cientists have embraced and even avidly pursued meaningless differences solely because they are statistically significant, and have ignored important effects because they failed to pass the screen of statistical significance...It is a safe bet that people have suffered or died because scientists (and editors, regulators, journalists and others) have used significance tests to interpret results, and have consequently failed to identify the most beneficial courses of action.” -ASA statement on p-values, 2016

3.5.2 What a p-value tells us

Performing responsible statistical inference requires understanding what p-values do and do not tell us, and how they should and should not be interpreted.

- A low p-value tells us that the data we observed are inconsistent with our null hypothesis or some assumption we make in our model.
- A large p-value tells us that the data we observed could have plausibly been obtained under our supposed model and null hypothesis.
- A p-value never provides evidence supporting the null hypothesis, it only tells us the strength of evidence against it.
- A p-value is impacted by
 - the size of the difference between group, or change per unit increase (effect size)
 - the amount of variability in the data
 - the sample size
- Sometimes, a p-value tells us more about sample size, than relationship we’re actually interested in.
- A p-value does not tell us the “size” of a difference or effect, or whether it is practically meaningful.

3.5.3 Flights from New York to Chicago

A traveler lives in New York and wants to fly to Chicago. They consider flying out of two New York airports:

- Newark (EWR)
 - LaGuardia (LGA)

We have data on the times of flights from both airports to Chicago's O'Hare airport from 2013 (more than 14,000 flights).

Assuming these flights represent a random sample of all flights from these airports to Chicago, consider how the traveler might use this information to decide which airport to fly out of.

3.5.4 Flight Data

3.5.5 Summarizing Flight Data

```

flights$origin <- as.factor(flights$origin)
flights$dest <- as.factor(flights$dest)
summary(flights %>% select(origin, dest, air_time))

##   origin          dest      air_time
##   EWR:120835    ORD     : 17283   Min.   : 20.0
##   JFK:111279    ATL     : 17215   1st Qu.: 82.0
##   LGA:104662    LAX     : 16174   Median  :129.0
##             BOS     : 15508   Mean    :150.7
##             MCO     : 14082   3rd Qu.:192.0
##             CLT     : 14064   Max.    :695.0
##             (Other):242450  NA's    :9430

```

3.5.6 Cleaning the Flights Data

We'll create a dataset containing only flights from Newark and Laguardia to O'Hare, and only the variables we're interested in.

```

Flights_NY_CHI <- flights %>%
  filter(origin %in% c("EWR", "LGA") & dest == "ORD") %>%
  select(origin, dest, air_time)
summary(Flights_NY_CHI)

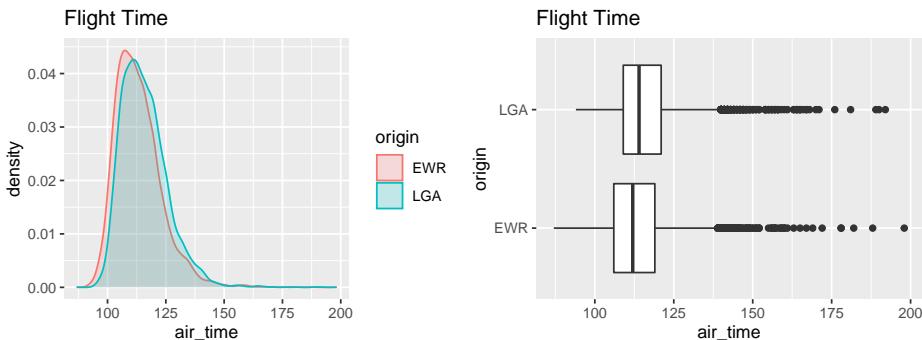
```

```

##   origin          dest      air_time
##   EWR:6100    ORD     :14957   Min.   : 87.0
##   JFK:  0     ABQ     :     0   1st Qu.:108.0
##   LGA:8857    ACK     :     0   Median  :113.0
##             ALB     :     0   Mean    :114.8
##             ANC     :     0   3rd Qu.:120.0
##             ATL     :     0   Max.    :198.0
##             (Other):     0   NA's    :622

```

3.5.7 Visualizing New York to Chicago Flights



origin	Mean_Airtime	SD	n
EWR	113.2603	9.987122	5828
LGA	115.7998	9.865270	8507

3.5.8 Model for Airlines Data

```
M_Flights <- lm(data=Flights_NY_CHI, air_time~origin)
summary(M_Flights)

##
## Call:
## lm(formula = air_time ~ origin, data = Flights_NY_CHI)
##
## Residuals:
##    Min     1Q Median     3Q    Max 
## -26.26  -7.26  -1.26   5.20  84.74 
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)    
## (Intercept) 113.2603    0.1299  872.06 <0.0000000000000002 *** 
## originLGA     2.5395    0.1686   15.06 <0.0000000000000002 *** 
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
##
## Residual standard error: 9.915 on 14333 degrees of freedom
##   (622 observations deleted due to missingness)
## Multiple R-squared:  0.01558,    Adjusted R-squared:  0.01551 
## F-statistic: 226.9 on 1 and 14333 DF,  p-value: < 0.000000000000022
```

3.5.9 Confidence Interval for Flights

```
confint(M_Flights)
```

```
##              2.5 %    97.5 %
## (Intercept) 113.00572 113.514871
## originLGA     2.20905   2.869984
```

- Flights from LGA are estimated to take 2.5 minutes longer than flights from EWR on average.
- The very low p-value provides strong evidence of a difference in mean flight time.
- We are 95% confident that flights from LGA to ORD take between 2.2 and 2.9 minutes longer, on average, than flights from EWR to ORD.

3.5.10 Flights Conclusions?

Would you base your decision of which airport to fly out of on this information?

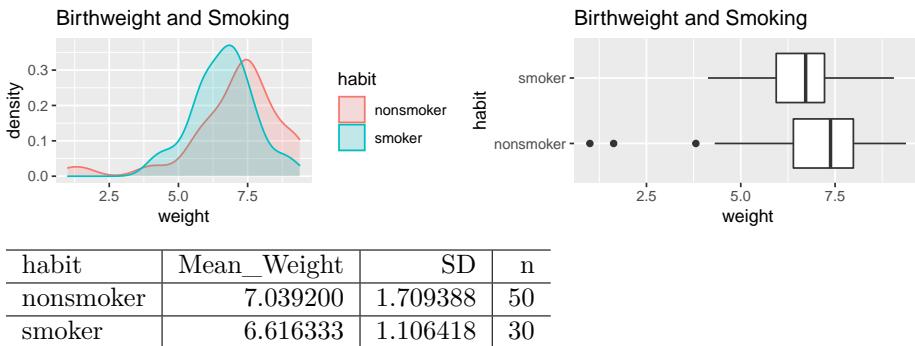
Although we have a low p-value, indicating a discernable difference, the size of this difference (2-3 minutes in airtime) is very small. A traveler would most likely have other, more important considerations when deciding which airport to fly from.

The low p-value is due to the very large sample size, rather than the size of the difference.

Note: there is also some question about whether it is appropriate to use a hypothesis test or confidence interval here at all. We have data on all flights in 2013, so one could argue that we have the entire population already. Perhaps, we could view this as a sample and generalize to flights in other years, though conditions change, so it is not clear that these flights from 2013 would be representative of flights in other years.

3.5.11 Smoking and Birthweight Example

We consider data on the relationship between a pregnant mother's smoking and the birthweight of the baby. Data come from a sample of 80 babies born in North Carolina in 2004. Thirty of the mothers were smokers, and fifty were nonsmokers.



3.5.12 Model for Birthweight

```
M_Birthwt <- lm(data=NCBirths, weight~habit)
summary(M_Birthwt)

##
## Call:
## lm(formula = weight ~ habit, data = NCBirths)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -6.0392 -0.6763  0.2372  0.8280  2.4437
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)
## (Intercept) 7.0392     0.2140  32.89 <0.0000000000000002 ***
## habitsmoker -0.4229     0.3495  -1.21      0.23
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.514 on 78 degrees of freedom
## Multiple R-squared:  0.01842,    Adjusted R-squared:  0.005834
## F-statistic: 1.464 on 1 and 78 DF,  p-value: 0.23
```

3.5.13 Conclusions from Birthweight Data

```
confint(M_Birthwt)

##
##              2.5 %    97.5 %
## (Intercept) 6.613070 7.4653303
## habitsmoker -1.118735 0.2730012
```

- The average birthweight of babies whose mothers are smokers is estimated to be about 0.42 lbs less than the average birthweight for babies whose mothers are nonsmokers.
- The large p-value of 0.23, tells us that there is not enough evidence to say that a mother's smoking is associated with lower birthweights. It is plausible that this difference could have occurred by chance.
- We are 95% confident that the average birthweight of babies whose mothers are smokers is between 1.12 lbs less and 0.27 lbs more than the average birthweight for babies whose mothers are nonsmokers.

3.5.14 Context of Birthweight Data

Many studies have shown that a mother's smoking puts a baby at risk of low birthweight. Do our results contradict this research?

Should we conclude that smoking has no impact on birthweights? (i.e. accept the null hypothesis?)

3.5.15 Impact of Small Sample Size

Notice that we observed a difference of about 0.4 lbs. in mean birthweight, which is a considerable difference.

The large p-value is mostly due to the relatively small sample size. Even though we observed a mean difference of 0.4 lbs, the sample is too small to allow us to say conclusively that smoking is associated with lower birthweights.

This is very different from concluding that smoking does not impact birthweight.

This is an example of why we should never "accept the null hypothesis" or say that our data "support the null hypothesis."

3.5.16 Larger Dataset

In fact, this sample of 80 babies is part of a larger dataset, consisting of 1,000 babies born in NC in 2004. When we consider the full dataset, notice that the difference between the groups is similar, but the p-value is much smaller, providing stronger evidence of a relationship between a mother's smoking and lower birthweight.

```
M_Birthwt_Full <- lm(data=ncbirths, weight~habit)
summary(M_Birthwt_Full)
```

```

## 
## Call:
## lm(formula = weight ~ habit, data = ncbirths)
## 
## Residuals:
##   Min     1Q Median     3Q    Max 
## -6.1443 -0.7043  0.1657  0.9157  4.6057 
## 
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)    
## (Intercept) 7.14427   0.05086 140.472 <0.0000000000000002 *** 
## habitsmoker -0.31554   0.14321  -2.203          0.0278 *   
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 1.503 on 997 degrees of freedom 
##   (1 observation deleted due to missingness) 
## Multiple R-squared:  0.004846,  Adjusted R-squared:  0.003848 
## F-statistic: 4.855 on 1 and 997 DF,  p-value: 0.02779

```

3.5.17 Cautions and Advice

p-values are only (a small) part of a statistical analysis.

- For small samples, real differences might not be statistically significant.
-Don't accept null hypothesis. Gather more information.
- For large, even very small differences will be statistically significant.
-Look at confidence interval. Is difference practically important?
- When many hypotheses are tested at once (such as many food items)
some will produce a significant result just by chance.
-Use a multiple testing correction, such as Bonferroni
- Interpret p-values on a “sliding scale”
– 0.049 is practically the same as 0.051
- Is sample representative of larger population?
- Were treatments randomly assigned (for experiments)?
- Are there other variables to consider?

Chapter 4

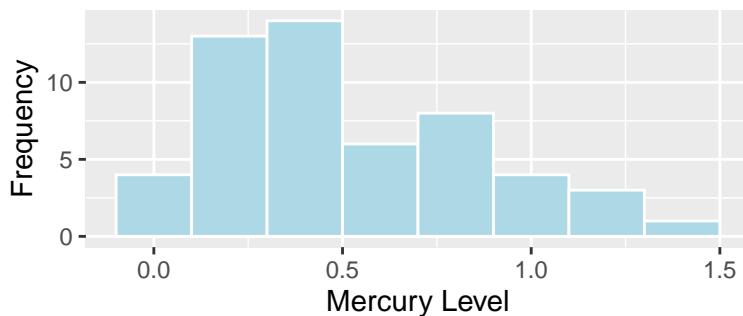
The Normal Error Linear Regression Model

4.1 Standard Error Formulas and Theory-Based Intervals

4.1.1 Standard Error for the Mean

- Standard error is the standard deviation of the distribution of a statistic, across many samples of the given size. This is different than the sample standard deviation, which pertains to the amount of variability between individuals in the sample.

Mercury Levels in Sample of Florida Lakes

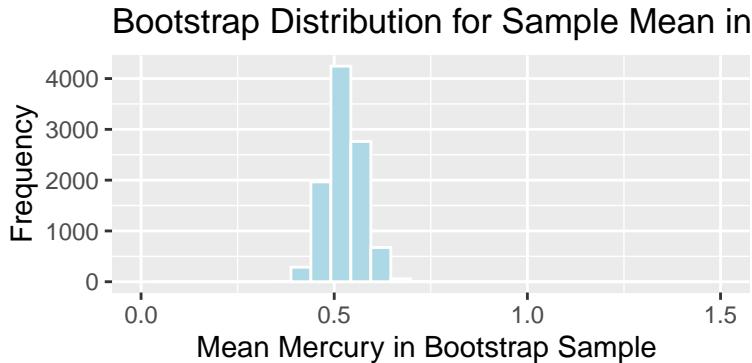


Standard Deviation:

```
sd(FloridaLakes$AvgMercury)
```

```
## [1] 0.3410356
```

The standard deviation in mercury levels between individual lakes is 0.341 ppm.



Standard Error for Mean:

```
SE <- sd(Lakes_Bootstrap_Results_Mean$MeanHg); SE
```

```
## [1] 0.04653932
```

The standard deviation in the distribution for mean mercury levels between different samples of 53 lakes is approximately 0.0465393 ppm.

4.1.2 Standard Error for the Mean (continued)

When the statistic is the sample mean \bar{x} , the standard error is given by:

$$SE(\bar{x}) = \frac{s}{\sqrt{n}},$$

where $s = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{(n-1)}}$ represents the sample standard deviation and n represents the sample size.

```
sd(FloridaLakes$AvgMercury)/sqrt(53)
```

```
## [1] 0.04684485
```

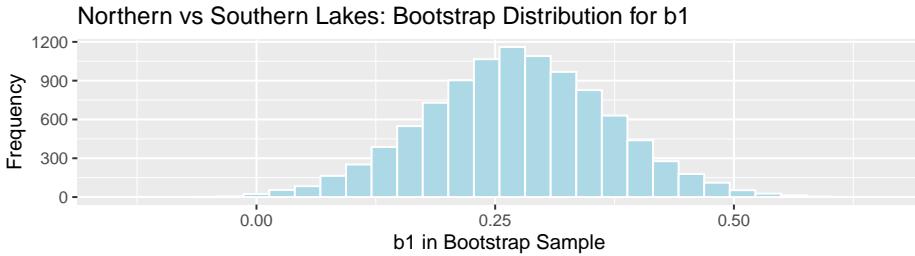
This is an approximation of the standard deviation in the distribution of sample means for samples of size 53.

4.1.3 Standard Error in R Output for Lakes

```
summary(lm(data=FloridaLakes, AvgMercury~1))

##
## Call:
## lm(formula = AvgMercury ~ 1, data = FloridaLakes)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.48717 -0.25717 -0.04717  0.24283  0.80283
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)
## (Intercept) 0.52717    0.04684   11.25 0.0000000000000151 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.341 on 52 degrees of freedom
```

4.1.4 Standard Error for Difference in Means



Standard Error for Mean:

```
SE <- sd(NS_Lakes_Bootstrap_Results$b1); SE
## [1] 0.09585589
```

4.1.5 Standard Error for Difference in Means (cont.)

When the statistic is the sample mean $\bar{x}_1 - \bar{x}_2$, the standard error is given by:

$$SE(\bar{x}) = s \sqrt{\frac{1}{n_1} + \frac{1}{n_2}},$$

where $s = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{(n-2)}}$ represents the sample standard deviation and n represents the sample size.

- The standard error estimate $s\sqrt{\frac{1}{n_1+n_2}}$ is called a “pooled” estimate since it combines information from all groups to estimate σ . When there is reason to believe standard deviation differs between groups, an “unpooled” standard error estimate of $\sqrt{\frac{s_1^2}{n_0} + \frac{s_2^2}{n_2}}$, where s_g represents the standard deviation for group g

4.1.6 Standard Error in R Output for Lakes Difference

```
summary(Lakes_M)

##
## Call:
## lm(formula = AvgMercury ~ Location, data = FloridaLakes)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.65650 -0.23455 -0.08455  0.24350  0.67545
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)
## (Intercept) 0.42455   0.05519   7.692 0.000000000441 ***
## LocationS   0.27195   0.08985   3.027   0.00387 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3171 on 51 degrees of freedom
## Multiple R-squared:  0.1523, Adjusted R-squared:  0.1357
## F-statistic: 9.162 on 1 and 51 DF,  p-value: 0.003868
```

4.1.7 Standard Error for Slope and Intercept in SLR

Standard Error for Slope of Regression Line:

$$SE(b_1) = \sqrt{\frac{s^2}{\sum(x_i - \bar{x})^2}}$$

Standard Error for Intercept of Regression Line:

$$SE(b_0) = s \sqrt{\frac{1}{n} + \frac{\bar{x}^2}{\sum(x_i - \bar{x})^2}}$$

where $s = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{(n-2)}}$ represents the sample standard deviation and n represents the sample size.

These standard errors pertain to the variability in slope and intercept of regression line between samples of 110 cars.

4.1.8 Standard Error for Slope and Intercept in SLR (cont.)

```
s <- summary(Cars_M_A060)$sigma
xbar <- mean(Cars2015$Acc060)
SSx <- sum((Cars2015$Acc060-xbar)^2)
```

SE for Slope:

```
sqrt(s^2/SSx)

## [1] 0.6234005
s*sqrt(1/110 + xbar^2/SSx)

## [1] 5.052325
```

4.1.9 R Output for Cars Slope

```
summary(Cars_M_A060)

##
## Call:
## lm(formula = LowPrice ~ Acc060, data = Cars2015)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -29.512  -6.544  -1.265   4.759  27.195
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)
## (Intercept)  89.9036    5.0523   17.79 <0.0000000000000002 ***
```

```

## Acc060      -7.1933      0.6234  -11.54 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.71 on 108 degrees of freedom
## Multiple R-squared:  0.5521, Adjusted R-squared:  0.548
## F-statistic: 133.1 on 1 and 108 DF,  p-value: < 0.0000000000000022

```

4.1.10 SE(b_j) in MLR

When there is more than one explanatory variable, estimates of regression coefficients and their standard errors become more complicated, and involves inversion of a “design matrix.”

This link provides additional information on the topic. Understanding it will likely require experience with linear algebra (i.e MATH 250).

Estimation in MLR goes beyond the scope of this class. For MLR in this class, you may use the estimates and standard errors reported in the R output, without being expected to calculate them yourself.

4.1.11 R Output for SE's in MLR

```

summary(Bear_M_Age_Sex_Int)

##
## Call:
## lm(formula = Weight ~ Age * Sex, data = Bears_Subset)
##
## Residuals:
##    Min     1Q   Median     3Q    Max
## -207.583 -38.854  -9.574   23.905 174.802
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)
## (Intercept) 70.4322   17.7260   3.973 0.000219 ***
## Age         3.2381    0.3435   9.428 0.00000000000765 ***
## Sex2        -31.9574   35.0314  -0.912 0.365848
## Age:Sex2    -1.0350    0.6237  -1.659 0.103037
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 70.18 on 52 degrees of freedom
##   (41 observations deleted due to missingness)
## Multiple R-squared:  0.6846, Adjusted R-squared:  0.6664

```

```
## F-statistic: 37.62 on 3 and 52 DF, p-value: 0.0000000000004552
```

4.1.12 Theory-Based Confidence Intervals

If the sampling distribution for a statistic is symmetric and bell-shaped, we can obtain an approximate 95% confidence interval using the formula:

$$\text{Statistic} \pm 2 \times \text{Standard Error},$$

where the standard error is calculated by formula, rather than via bootstrap simulations.

4.1.13 Comparison of CI Methods

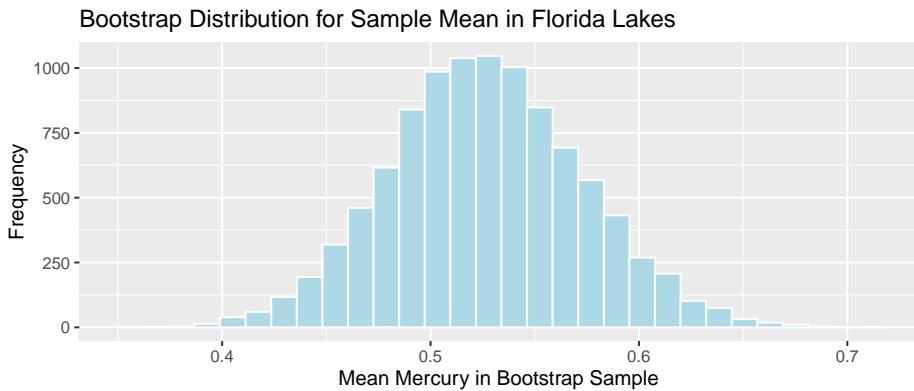
We've now seen 3 different ways to obtain confidence intervals based on statistics, calculated from data.

The table below tells us what must be true of the sampling distribution for a statistic in order to use each technique.

Technique	No Gaps	Bell-Shaped	Known Formula for SE
Bootstrap Percentile	x		
Bootstrap Standard Error	x	x	
Theory-Based	x	x	x

4.1.14 When to use Each CI Method Ex.1

Mean mercury level for all Florida lakes:



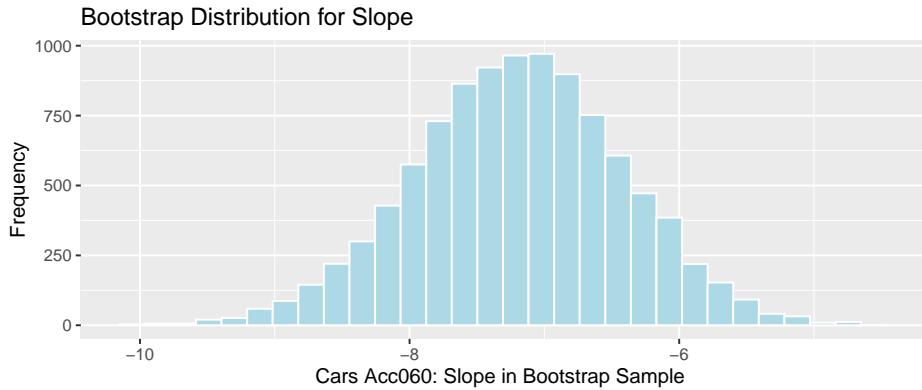
It is appropriate to use any of the 3 CI methods since

- sampling distribution is symmetric and bell-shaped with no gaps
- there is a known formula to calculate standard error for a sample mean

The methods should all produce similar results.

4.1.15 When to use Each CI Method Ex.2

Slope of regression line in cars example



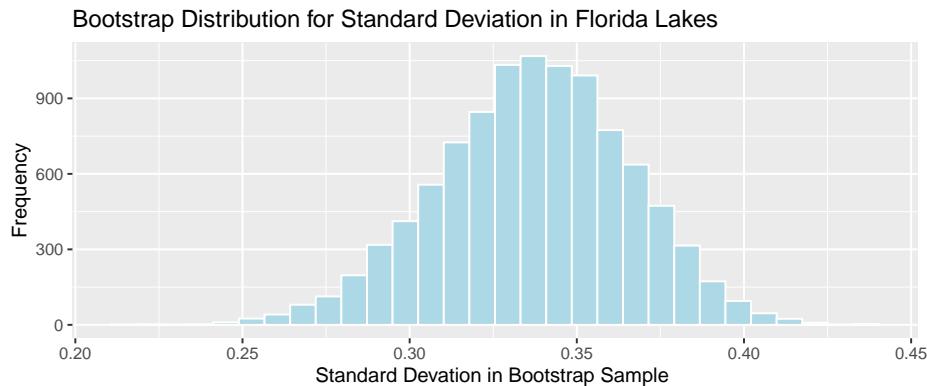
It is appropriate to use any of the 3 CI methods since

- sampling distribution is symmetric and bell-shaped with no gaps
- there is a known formula to calculate standard error for a slope of regression line

The methods should all produce similar results.

4.1.16 When to use Each CI Method Ex.3

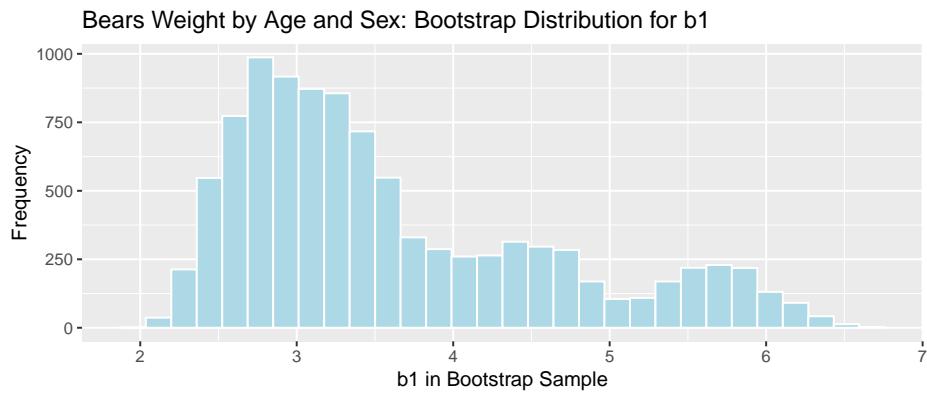
Standard deviation of mercury level in Florida Lakes



- It is appropriate to use the bootstrap percentile, and bootstrap standard error CI's since the sampling distribution is symmetric and bell-shaped.
- We cannot use the theory-based interval because we do not have a formula to calculate the standard error, associated with an estimate of σ .

4.1.17 When to use Each CI Method Ex.4

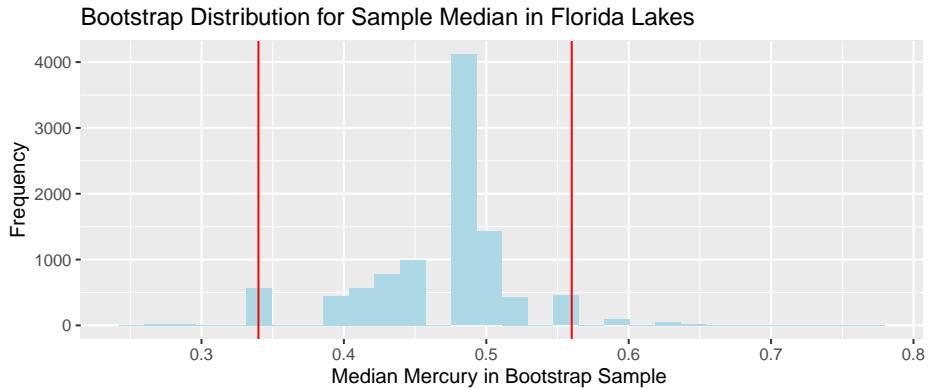
Regression slope in bears age model



- It is appropriate to use the bootstrap percentile CI, since the sampling distribution has no gaps.
- Since the distribution is not symmetric, it would be inappropriate to use the bootstrap standard error, or theory-based confidence interval (Although R does calculate a SE, using it to produce a CI would be unreliable).

4.1.18 When to use Each CI Method Ex.5

Median mercury level in Florida Lakes

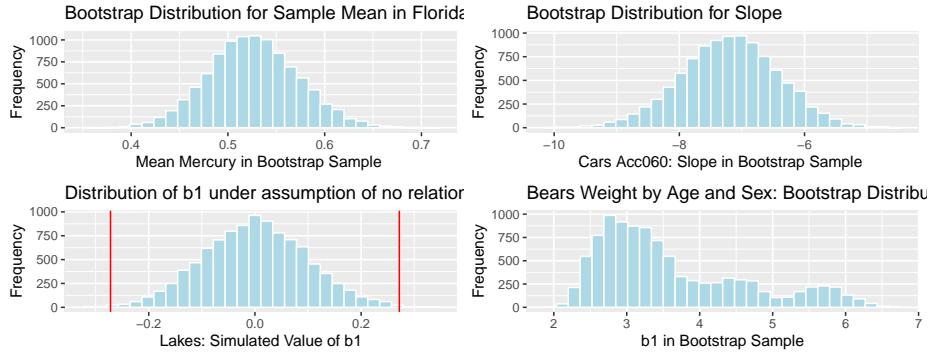


- Since the distribution has gaps, and is not symmetric, none of these procedures are appropriate.
- In some cases (usually with larger sample size), a bootstrap distribution for the median will not have these gaps. In these situations, the percentile bootstrap interval would be appropriate. If the distribution is bell-shaped, the standard error method would also be appropriate. We would not be able to use the theory-based approach, because there is no formula for the standard error associated with a sample median.

4.2 The Normal Error Regression Model

4.2.1 Symmetric, Bell-Shaped Distributions

Notice that when we used simulation to approximate the sampling distributions of statistics, many (but not all) of these turned out to be symmetric and bell-shaped.



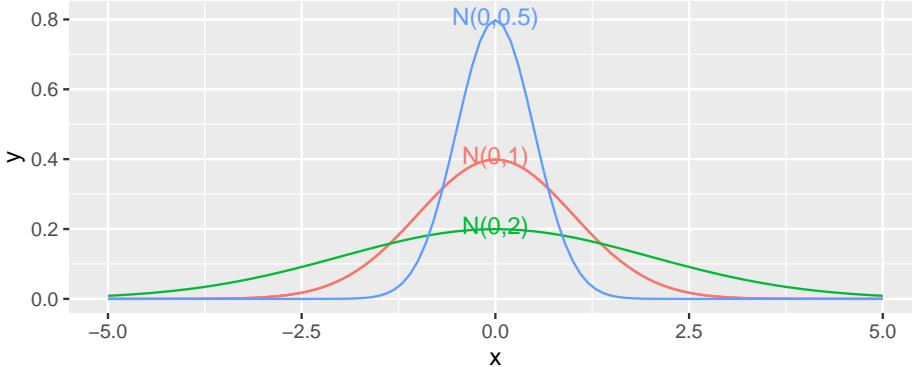
4.2.2 Normal Distribution

Symmetric and bell-shaped sampling distributions can be approximated with the normal distribution.

A normal distribution is defined by two parameters:

- μ representing the center of the distribution
- σ representing the standard deviation

This distribution is denoted $\mathcal{N}(0, \sigma)$.



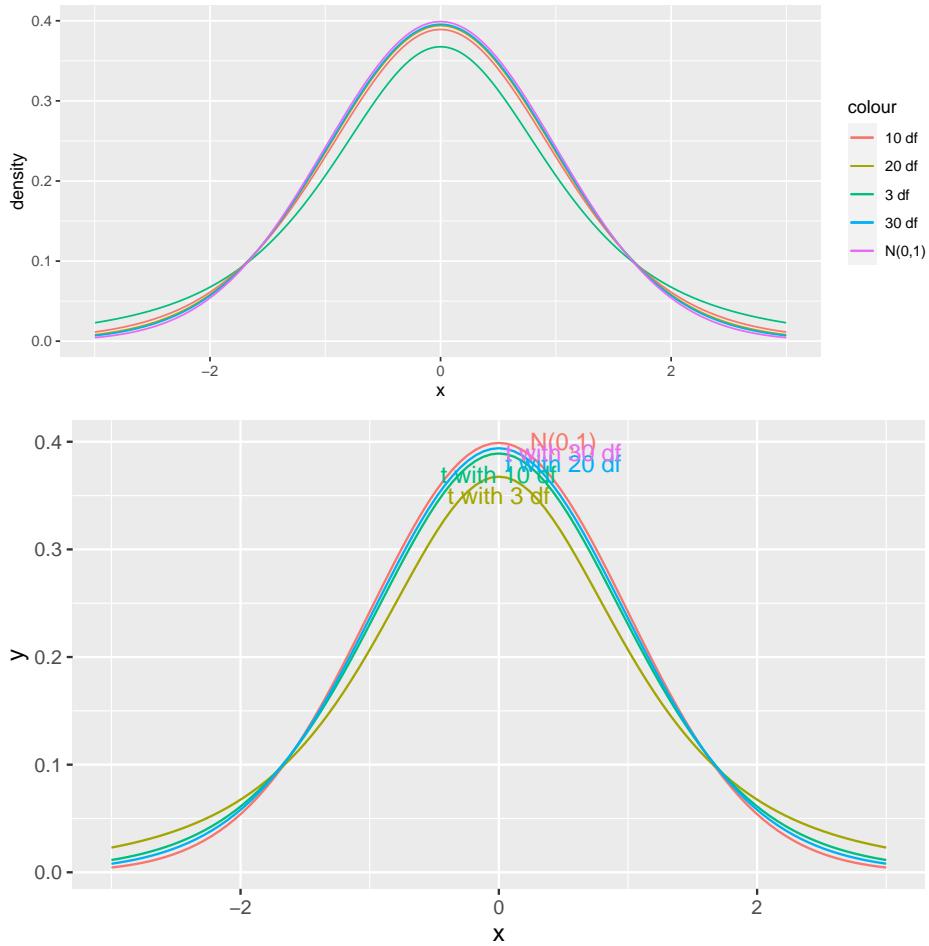
Note that for standard deviation σ , σ^2 is called the variance. Some books denote the normal distribution as $\mathcal{N}(0, \sigma^2)$, instead of $\mathcal{N}(0, \sigma)$.

4.2.3 t-distribution

A t-distribution is a symmetric, bell-shaped curve, with thicker tails (hence more variability), than a $\mathcal{N}(0, 1)$ distribution.

```
gf_dist("t", df=3, color = "#3399FF", kind = "density") %>%
  gf_dist("t", df=10, color = "#FF9933", kind = "density") %>%
```

```
gf_dist("t", df=20, color = ~ "20 df", kind = "density") %>%
gf_dist("t", df=30, color = ~ "30 df", kind = "density") %>%
gf_dist("norm", color = ~ "N(0,1)", kind = "density") + xlim(c(-3,3))
```



4.2.4 Normal Error Regression Model

In regression, it is common to assume the following:

1. The expected value of the response variable is a function of one or more explanatory variables. Often, this function is assumed to be linear.
2. Individual observations vary from their expected value according to a normal distribution.

The first part is often referred to as *signal* and the second as *noise*.

This can be written mathematically as:

$$Y_i = \beta_0 + \beta_1 X_{i1} + \dots + \beta_p X_{ip} + \epsilon_i, \text{ with } \epsilon_i \sim \mathcal{N}(0, \sigma).$$

4.2.5 Example: Ice Cream Dispensor

Suppose an ice cream machine is manufactured to dispense 2 oz. of ice cream per second, on average, but that the actual amount dispensed each time it is used varies due to factors such as

- force applied to dispenser
- temperature
- build-up of ice cream
- other factors

Suppose that the actual amount dispensed varies from its expectation according to a $\mathcal{N}(0, 0.5)$ distribution.

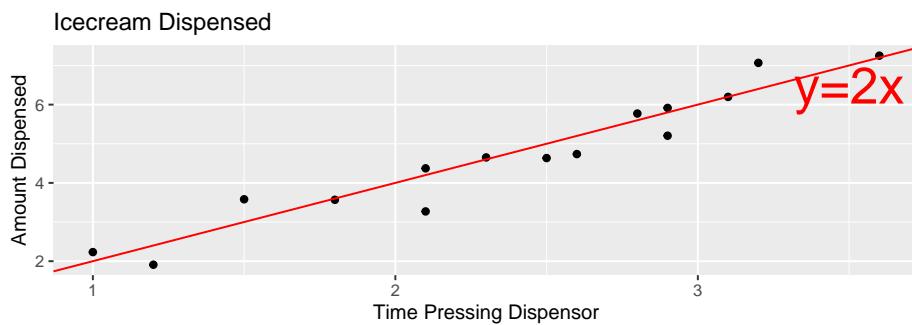




Image from: <https://www.partstown.com/about-us/spaceman-ice-cream-machine-troubleshooting>

4.2.6 Model for Ice Cream Dispensor

True Model:

$$Y_i = 2X_i + \epsilon_i, \text{ where } \epsilon_i \sim \mathcal{N}(0, 0.5)$$

where

- y_i represents amount dispensed for person i ,
- x_i represents time person i holds dispensor
- ϵ_i is the error term, representing variation due to unobservable factors

4.2.7 Simulating Data from a Model

Suppose that 15 people use the icecream machine, taking the following amounts of time in seconds:

```
set.seed(10082020)
time <- c(1, 1.2, 1.5, 1.8, 2.1, 2.1, 2.3, 2.5, 2.6, 2.8, 2.9, 2.9, 3.1, 3.2, 3.6)
```

The code below simulates the amount of icecream dispensed for each person.

```

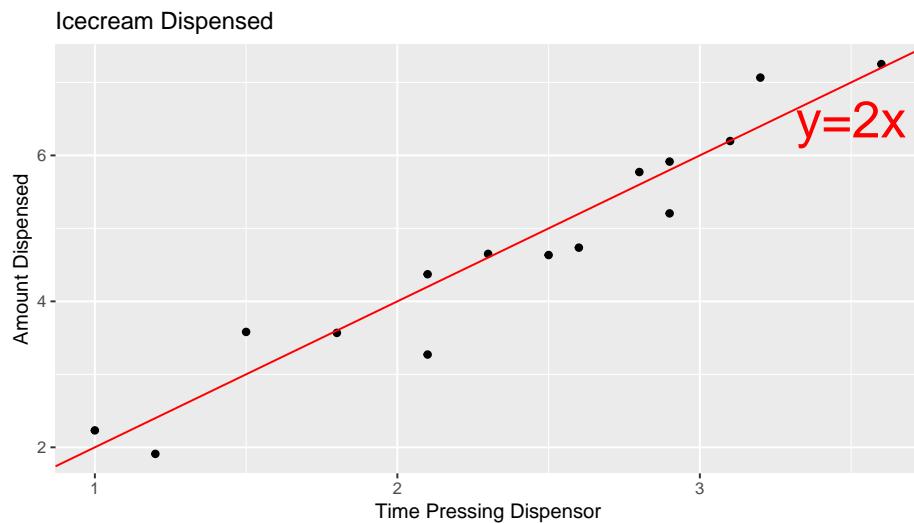
amount <- 2*time + rnorm(15, 0, 0.5)
Icecream1 <- data.frame(time, amount)

kable(t(round(Icecream1, 2)))

```

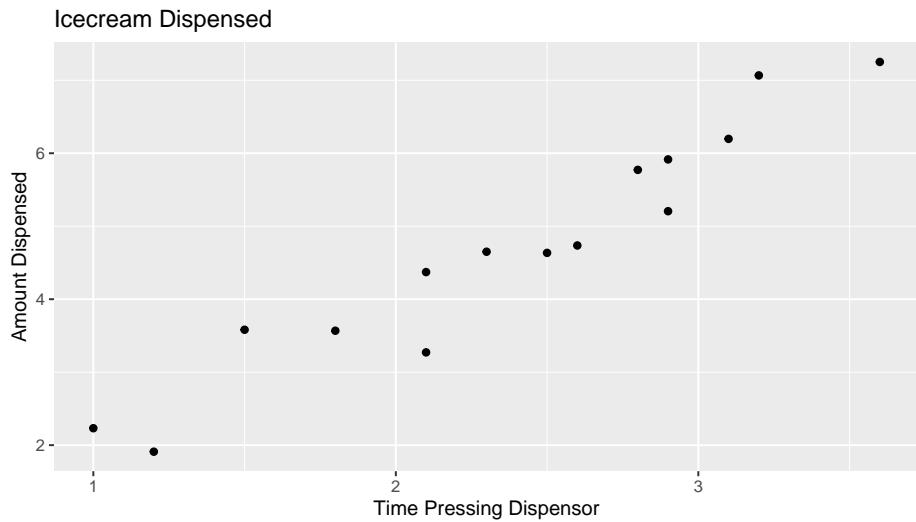
time	1.00	1.20	1.50	1.80	2.10	2.10	2.30	2.50	2.60	2.80	2.90	2.90	3.1	3.20	3.60
amount	2.23	1.91	3.58	3.57	4.37	3.27	4.65	4.63	4.74	5.77	5.21	5.92	6.2	7.07	7.25

4.2.8 Scatterplot for Ice Cream Sample Data

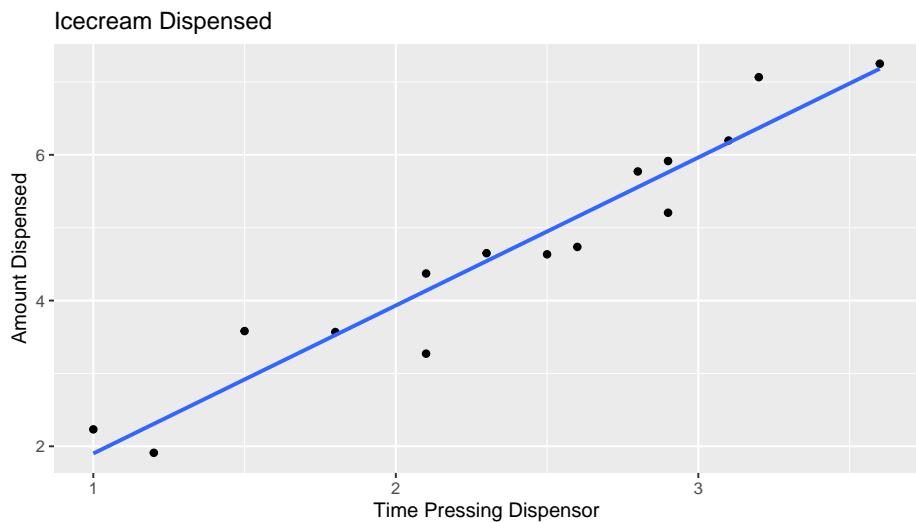


The red line represents the expectation function $E(Y_i) = 2X_i$.

4.2.9 What We Actually Get to See



4.2.10 What We Actually Get to See (cont.)



The blue line represents the least squares regression line.

4.2.11 Estimation in Ice Cream Example

Assume we know that the model has the form $Y_i = \beta_0 + \beta_1 X + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \sigma)$, but we do not know the values of β_0 , β_1 , or σ .

- b_0 and b_1 are calculated by minimizing SSR, and serve as estimates of β_0 and β_1
- are calculated by minimizing SSR, and serve as estimates of β_0 and β_1
- $s = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{(n-(p+1))}} = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{(n-2)}} = \sqrt{\frac{\text{SSR}}{n-2}}$ is an estimate of σ .

4.2.12 Ice Cream Model R Output

```
IC_Model <- lm(data=Icecream1, lm(amount~time))
summary(IC_Model)

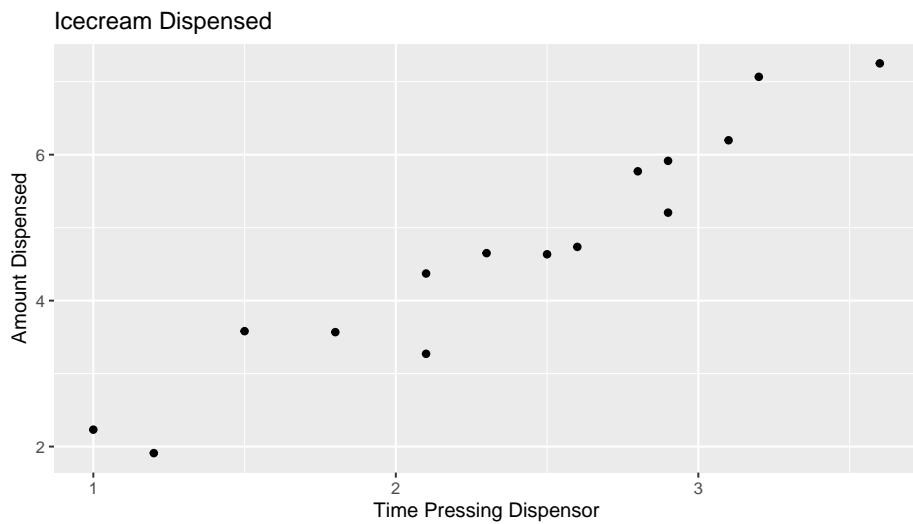
##
## Call:
## lm(formula = lm(amount ~ time), data = Icecream1)
##
## Residuals:
##    Min      1Q  Median      3Q     Max
## -0.8645 -0.3553  0.0685  0.2252  0.6963
##
## Coefficients:
##             Estimate Std. Error t value    Pr(>|t|)
## (Intercept) -0.1299     0.3968  -0.327     0.749
## time         2.0312     0.1598  12.714 0.0000000104 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4527 on 13 degrees of freedom
## Multiple R-squared:  0.9256, Adjusted R-squared:  0.9198
## F-statistic: 161.6 on 1 and 13 DF,  p-value: 0.00000001042
```

4.2.13 Ice Cream Model: Estimated Regression Equation

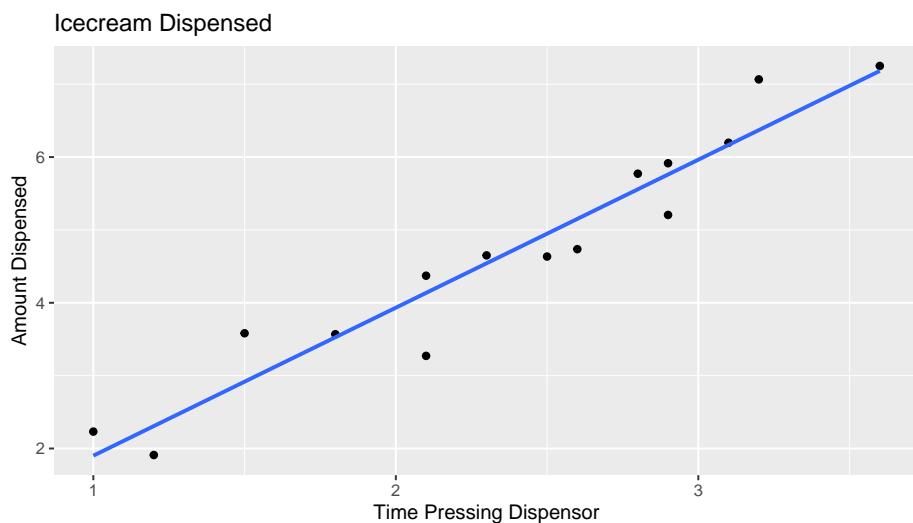
- The estimated regression equation is $E(Y_i) = -0.1299087 + 2.0312489 \times X_i$
- The estimate $b_0 = -0.1299087$ is close to the true value of $\beta_0 = 0$
- The estimate $b_1 = 2.0312489$ is close to the true value of $\beta_1 = 2$.
- The estimate $s = 0.4527185$ is close to the true value of $\sigma = 0.5$.

Remember that in a real situation, we would not know the values of β_0 and β_1 , or σ and would have only estimates b_0 and b_1 , and s .

4.2.14 What We Actually Get to See

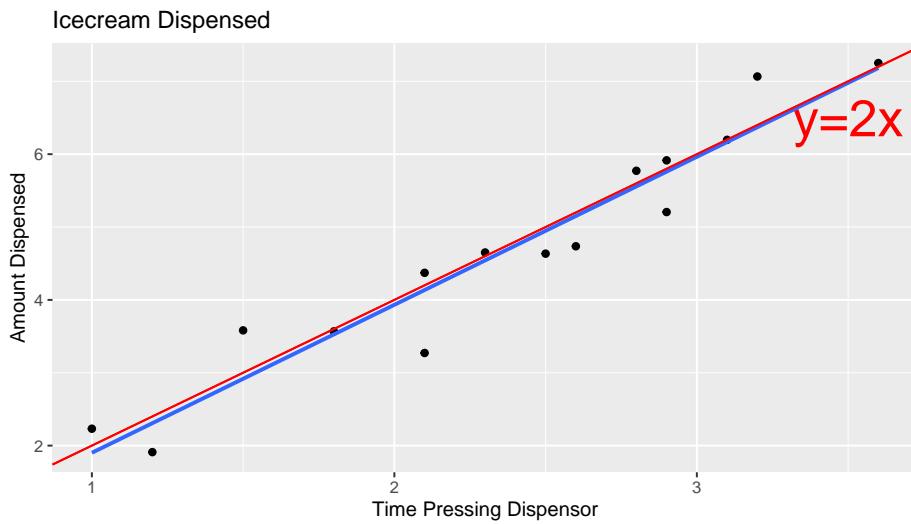


4.2.15 What We Actually Get to See (cont.)



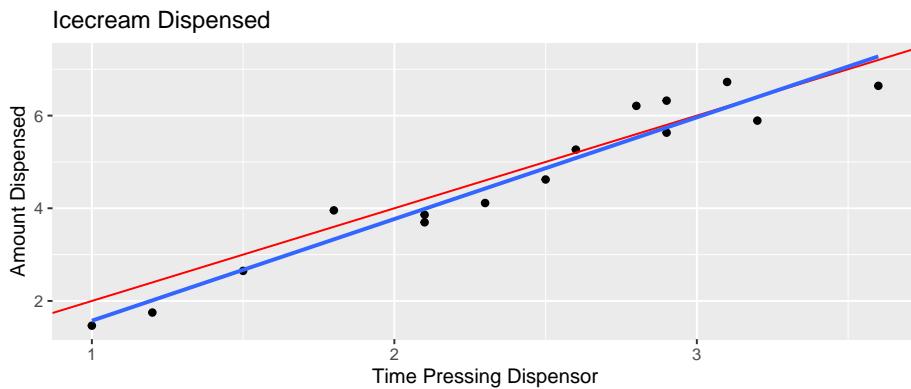
The blue line represents the least squares regression line.

4.2.16 True and Estimated Regression Line



The red line represents the true expectation function $E(\text{Amount}) = 0 + 2 \times \text{Time}$.
The blue line represents the estimated regression line $\widehat{\text{Amount}} = -0.1299087 + 2.0312489 \times \text{Time}$

4.2.17 Second Sample

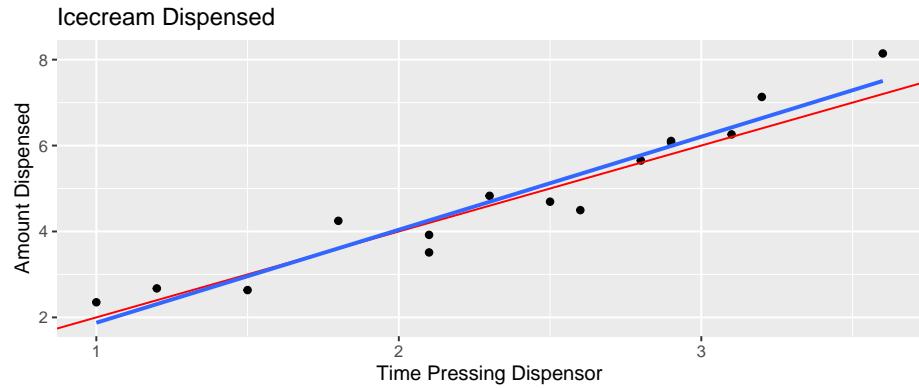


The red line represents the expectation function $E(Y_i) = 2X_i$.

Estimated Regression Equation: $\widehat{\text{Amount}} = -0.6196117 + 2.1936879 \times \text{Time}$

Estimate of σ : 0.4423186

4.2.18 Third Sample

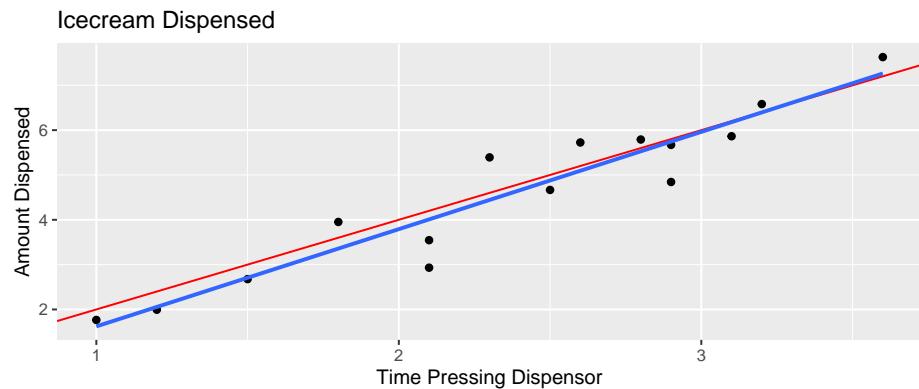


The red line represents the expectation function $E(Y_i) = 2X_i$.

Estimated Regression Equation: $\widehat{\text{Amount}} = -0.2893779 + 2.164916 \times \text{Time}$

Estimate of σ : 0.4939971

4.2.19 Fourth Sample

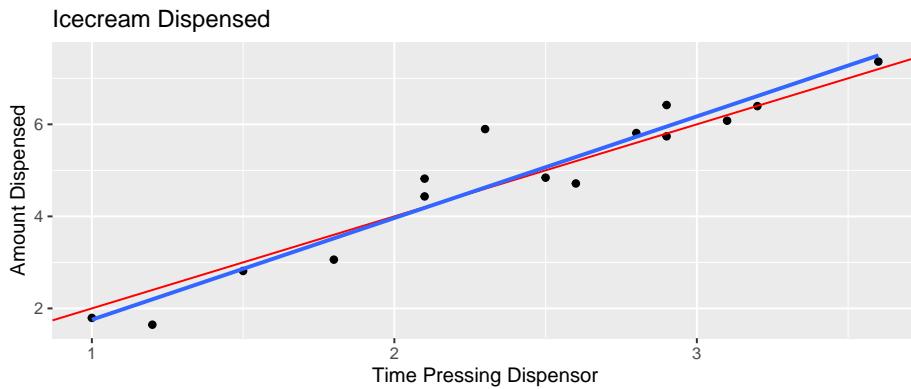


The red line represents the expectation function $E(Y_i) = 2X_i$.

Estimated Regression Equation: $\widehat{\text{Amount}} = -0.5478614 + 2.1699256 \times \text{Time}$

Estimate of σ : 0.5718905

4.2.20 Fifth Sample



The red line represents the expectation function $E(Y_i) = 2X_i$.

Estimated Regression Equation: $\widehat{\text{Amount}} = -0.4527901 + 2.2084662 \times \text{Time}$

Estimate of σ : 0.512594

4.2.21 Generate Many Samples from IC Example

Let's suppose that many more customers come along in groups of 15, and push the dispenser for the same amounts of time as the group in our original sample. We'll calculate b_0 , b_1 , and s for each sample.

This gives us the sampling distributions for each of these three statistics.

We'll also store the standard errors for b_0 and b_1 , which we'll use later.

```

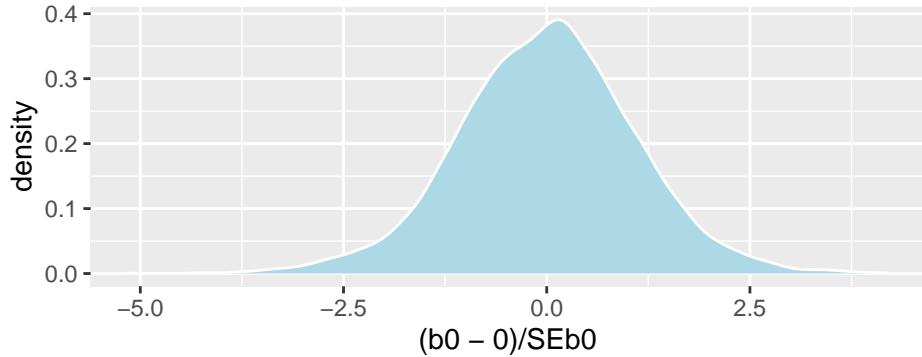
b0 <- rep(NA, 10000)
b1 <- rep(NA, 10000)
SEb0 <- rep(NA, 10000)
SEb1 <- rep(NA, 10000)

for (i in 1:10000){
  e <- rnorm(15, 0, 0.5)
  y <- 2*time+e
  df <- data.frame(time, y)
  M <- lm(data=df, y~time)
  b0[i] <- M$coef[1]
  b1[i] <- M$coef[2]
  SEb0[i] <- summary(M)$coefficients[1,2]
  SEb1[i] <- summary(M)$coefficients[2,2]
}
IC_SampDist <- data.frame(b0, b1, SEb0, SEb1)

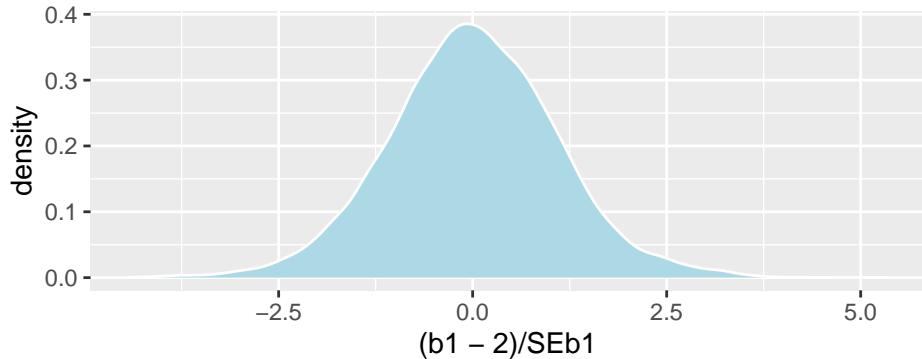
```

4.2.22 Distribution of $\frac{b_0 - \beta_0}{SE(b_0)}$ and $\frac{b_1 - \beta_1}{SE(b_1)}$

```
ggplot(data=IC_SampDist, aes(x=(b0-0)/SEb0)) +
  geom_density(fill="lightblue", color="white") + stat_function(fun=dt)
```



```
ggplot(data=IC_SampDist, aes(x=(b1-2)/SEb1)) +
  geom_density(fill="lightblue", color="white")
```



4.2.23 Distribution of $\frac{b_j - \beta_j}{SE(b_j)}$

Important Fact: If $Y_i = \beta_0 + \beta_1 X_{i1} + \dots + \beta_p X_{ip} + \epsilon_i$, with $\epsilon_i \sim \mathcal{N}(0, \sigma)$, then

$$t = \frac{b_j - \beta_j}{SE(b_j)}$$

follows a t-distribution with $n - (p + 1)$ degrees of freedom.

4.3 Tests and Intervals in Normal Error Regression Model

4.3.1 Confidence Interval for β_j

Important Fact: If $Y_i = \beta_0 + \beta_1 X_{i1} + \dots + \beta_p X_{ip} + \epsilon_i$, with $\epsilon_i \sim \mathcal{N}(0, \sigma)$, then

$$t = \frac{b_j - \beta_j}{\text{SE}(b_j)}$$

follows a t-distribution with $n - (p + 1)$ degrees of freedom.

A 95% confidence interval for β_j is given by

$$b_j \pm t^* (\text{SE}(b_j)),$$

where t^* is chosen to achieve the desired confidence level.

- For a 95% confidence interval, use $t^* = 2$.

4.3.2 Hypothesis test for $\beta_j = \gamma$

- If $Y_i = \beta_0 + \beta_1 X_{i1} + \dots + \beta_p X_{ip} + \epsilon_i$, with $\epsilon_i \sim \mathcal{N}(0, \sigma)$, then a test statistic for the null hypothesis: $\beta_j = \gamma$ is given by:

$$t = \frac{b_j - \gamma}{\text{SE}(b_j)},$$

and calculate a p-value using a t-distribution with $n - (p + 1)$ df.

4.3.3 Theory-Based CI for β_0 in Ice Cream Example

95% confidence interval for β_0 :

$$b_0 \pm t^* \text{SE}(b_0) = -0.4527901 \pm 2 \times 0.4492685$$

```
b0 <- summary(IC_Model)$coefficients[1,1]
SEb0 <- summary(IC_Model)$coefficients[1,2]
c(b0 - 2*SEb0, b0 + 2*SEb0)

## [1] -1.3513270  0.4457469
```

We can be 95% confident that β_0 is between -1.35 and 0.45.

4.3.4 Theory-Based CI for β_1 in Ice Cream Example

95% confidence interval for β_1 :

$$\hat{\beta}_1 \pm 2\text{SE}(\hat{\beta}_1) = 2.2084662 \pm 2 \times 0.180898$$

```
b1 <- summary(IC_Model)$coefficients[2,1]
SEb1 <- summary(IC_Model)$coefficients[2,2]
c(b1 - 2*SEb1, b1 + 2*SEb1)

## [1] 1.846670 2.570262
```

We can be 95% confident that β_1 is between 1.85 and 2.57, meaning that we expect that the average amount of ice cream dispensed per second held is between 1.85 and 2.57 oz.

4.3.5 Confidence Interval in R

We can calculate t-distribution based confidence intervals automatically, using the `confint` command.

```
confint(IC_Model, level=0.95)

##               2.5 %    97.5 %
## (Intercept) -1.423376  0.5177955
## time         1.817660  2.5992725
```

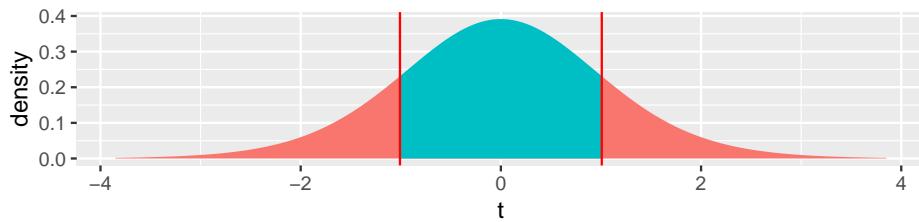
4.3.6 IC Example: Hypothesis Test for $\beta_0 = 0$

Null Hypothesis: $\beta_0 = 0$

Test Statistic: $t = \frac{\hat{\beta}_0 - 0}{\text{SE}(\hat{\beta}_0)} = \frac{-0.4527901 - 0}{0.4492685} = -1.0078385$

p-value:

```
b0 <- summary(IC_Model)$coeff[1,1]; SEb0 <- summary(IC_Model)$coeff[1,2]
ts <- (b0-0)/SEb0
```



```
2*pt(-abs(ts), df=13)
```

```
## [1] 0.3319236
```

Since the p-value is large, it is plausible that $\beta_0 = 0$

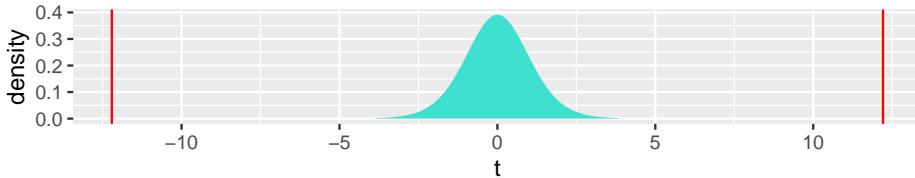
4.3.7 IC Example: Hypothesis Test for $\beta_1 = 0$

Null Hypothesis: $\beta_1 = 0$

Test Statistic: $t = \frac{b_1 - 0}{SE(b_1)} = \frac{2.2084662 - 0}{0.180898} = 12.2083504$

p-value:

```
b1 <- summary(IC_Model)$coeff[2,1]; SEb1 <- summary(IC_Model)$coeff[2,2]
ts <- (b1-0)/SEb1
```



```
2*pt(-abs(ts), df=13)
```

```
## [1] 0.00000001699457
```

Since the p-value is low, there is strong evidence against $\beta_1 = 0$. Thus, there is evidence of a relationship between time pressed and amount dispensed.

4.3.8 t-statistics and p-values in R

Note that our results match the coefficients table in the R model output.

```
summary(IC_Model)
```

```
##
## Call:
## lm(formula = lm(amount ~ time), data = Icecream2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.5741 -0.2707 -0.1347  0.1644  1.2702
##
## Coefficients:
##             Estimate Std. Error t value    Pr(>|t|)
## (Intercept) -0.4528     0.4493 -1.008     0.332
```

```

## time           2.2085      0.1809   12.208 0.000000017 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5126 on 13 degrees of freedom
## Multiple R-squared:  0.9198, Adjusted R-squared:  0.9136
## F-statistic:  149 on 1 and 13 DF,  p-value: 0.00000001699

```

4.3.9 Coefficients Table in R Output

For $\hat{Y} = b_0 + b_1 X_{i1} + b_2 X_{i2} + \dots + b_p X_{ip}$,

- **Estimate** gives the least-squares estimates b_0, b_1, \dots, b_p
- **Standard Error** gives estimates of the standard deviation in the sampling distribution for estimate. (i.e. how much uncertainty is there about the estimate?)
- **t value** is the estimate divided by its standard error.
- **Pr(>|t|)** is a p-value for the hypothesis test of whether quantity represented b_j could plausibly be 0.

This p-value is an approximation of the kind of p-value we have obtained through simulation. It is reliable only when certain assumptions are reasonable.

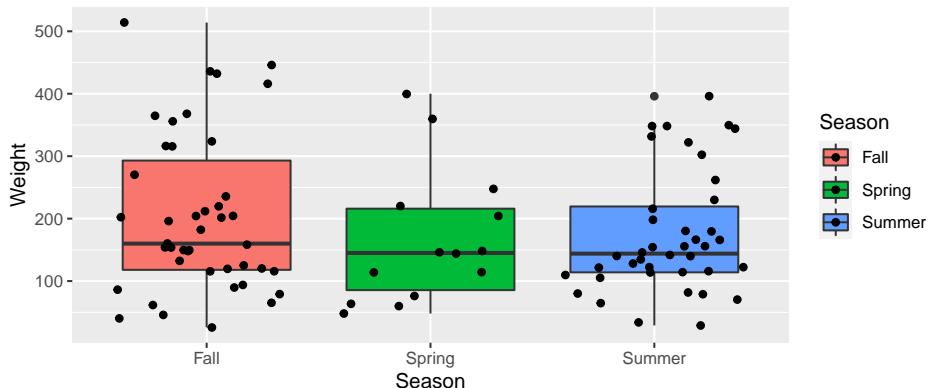
4.4 F-Distribution

4.4.1 Recall Bear Weights by Season

```

ggplot(data=Bears_Subset, aes(y=Weight, x=Season, fill=Season)) +
  geom_boxplot() + geom_jitter()

```



4.4.2 Model for Bear Weights by Season Model

```
summary(Bears_M_Season)

##
## Call:
## lm(formula = Weight ~ Season, data = Bears_Subset)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -178.84 -79.84 -29.02  54.98 309.16
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept) 204.84     17.16 11.939 <0.0000000000000002 ***
## SeasonSpring -37.27     34.62 -1.076          0.284
## SeasonSummer -29.81     24.71 -1.206          0.231
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 112.5 on 94 degrees of freedom
## Multiple R-squared:  0.02034,   Adjusted R-squared:  -0.0005074
## F-statistic: 0.9757 on 2 and 94 DF,  p-value: 0.3807
```

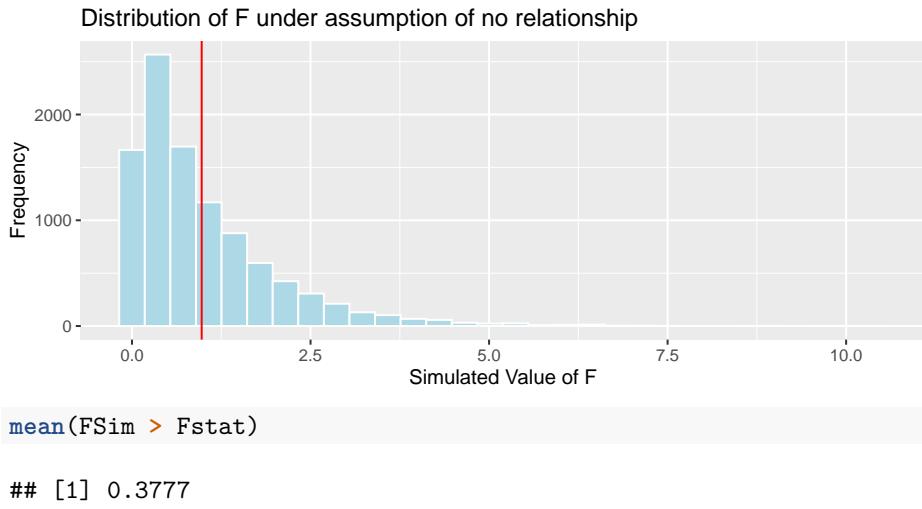
4.4.3 F-Statistic for Bear Weights by Season

```
Bears_A_Season <- aov(data=Bears_Subset, Weight~Season)
summary(Bears_A_Season)
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Season	2	24699	12350	0.976	0.381
Residuals	94	1189818	12658		

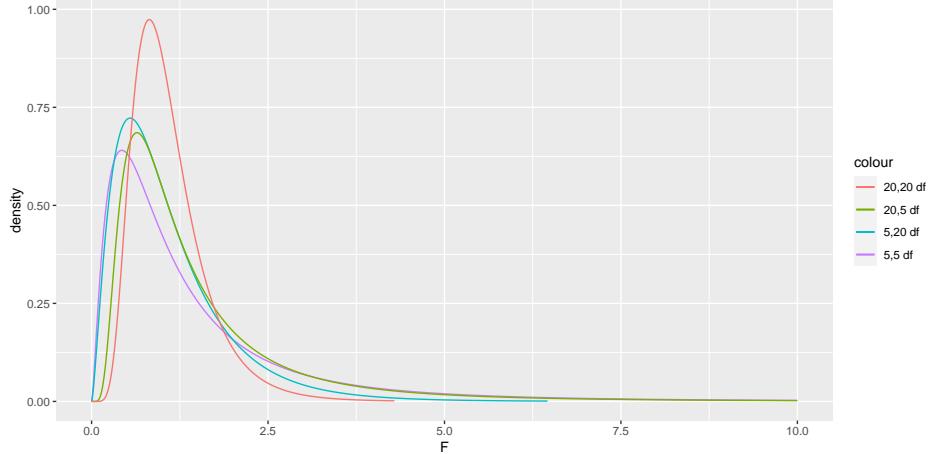
4.4.4 Simulated F-Stats for Bear Weights by Season

```
Fstat <- summary(Bears_M_Season)$fstatistic[1]
Bears_Seasons_SimulationResultsPlot
```



4.4.5 F-Distribution

An [F distribution] is a right-skewed distribution. It is defined by two parameters, ν_1, ν_2 , called numerator and denominator degrees of freedom.



4.4.6 Distribution of F-Statistic

If $Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \dots + \beta_q X_{iq} + \epsilon_i$, with $\epsilon_i \sim \mathcal{N}(0, \sigma)$,

and $Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \dots + \beta_q X_{iq} + \beta_{q+1} X_{iq+1} \dots + \beta_p X_{ip} + \epsilon_i$, is another proposed model, then

$$F = \frac{\frac{\text{Unexplained Variability in Reduced Model} - \text{Unexplained Variability in Full Model}}{p-q}}{\frac{\text{Unexplained Variability in Full Model}}{n-(p+1)}}$$

follows an F-distribution with $(p-q)$ and $(n-(p+1))$ degrees of freedom.

4.4.7 F-Test for a Single Categorical Variable

We have seen that for a categorical variable with g groups, the proposed models reduce to

$$Y_i = \beta_0 + \epsilon_i, \text{ with } \epsilon_i \sim \mathcal{N}(0, \sigma),$$

$$\text{and } Y_i = \beta_0 + \beta_1 I_{\text{Group2 } i} + \dots + \beta_{g-1} I_{\text{Group } g} + \epsilon_i,$$

and the F-statistic is equivalent to

$$F = \frac{\text{Variability between Groups}}{\text{Variability within Groups}} = \frac{\frac{\sum_{i=1}^g \sum_{j=1}^{n_i} n_i (y_{i\cdot} - \bar{y}_{..})^2}{g-1}}{\frac{\sum_{i=1}^g \sum_{j=1}^{n_i} (y_{ij} - \bar{y}_{i\cdot})^2}{n-g}}$$

and this statistic follows an F-distribution with $(g-1)$ and $(n-g)$ degrees of freedom.

4.4.8 F-Test for Bear Weights by Season

```
Bears_A_Season <- aov(data=Bears_Subset, Weight~Season)
summary(Bears_A_Season)

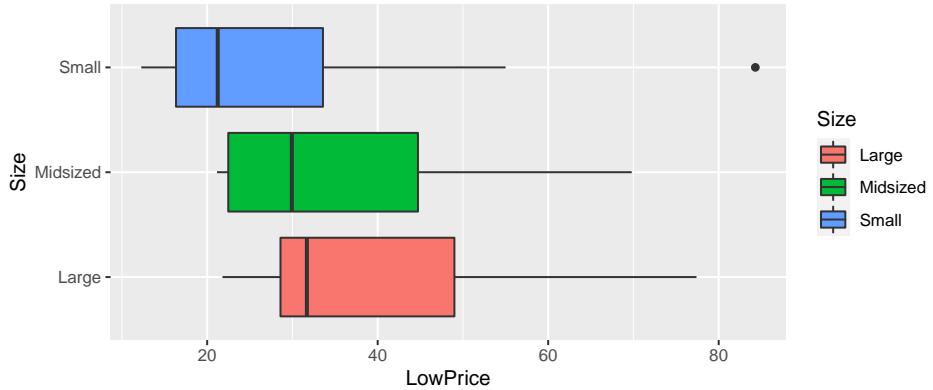
##          Df  Sum Sq Mean Sq F value Pr(>F)
## Season       2    24699   12350   0.976  0.381
## Residuals   94   1189818   12658
```

The p-value we obtained is very similar to the one we obtained using the simulation-based test.

In this case, even though we had concerns about normality, they did not have much impact on the p-value from the F-distribution. The F-test is fairly robust to minor departures from normality.

4.4.9 F-Test for Car Size and Price

```
ggplot(data=Cars2015, aes(x=Size, y=LowPrice)) + geom_boxplot(aes(fill=Size)) + coord_polar(theta="y")
```



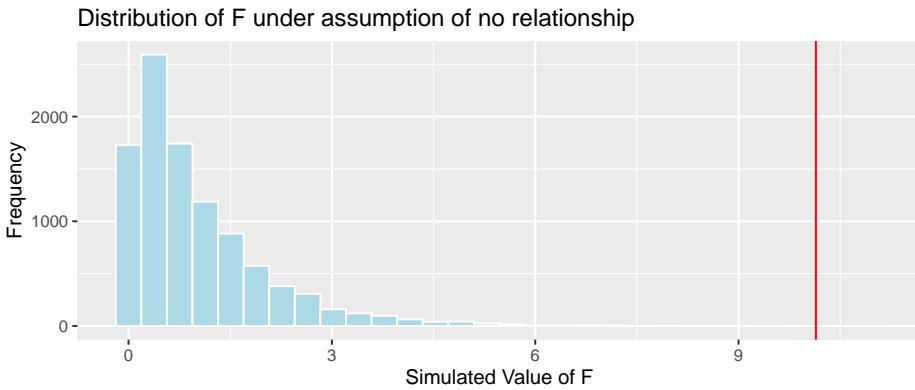
4.4.10 F-Statistic for Car Size and Price

```
Cars_A_Size <- aov(data=Cars2015, LowPrice~Size)
summary(Cars_A_Size)
```

```
##              Df Sum Sq Mean Sq F value    Pr(>F)
## Size          2   4405   2202.7   10.14 0.0000927 ***
## Residuals    107  23242    217.2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.4.11 Recall Simulation-Based F-test

```
Fstat <- summary(Cars_M_Size)$fstatistic[1]
CarSize_SimulationResults_Plot
```



```
mean(CarSize_SimulationResults$FSim > Fstat)
```

```
## [1] 0.0002
```

The data provide strong evidence of a relationship between price and size.

The results of the simulation based F-test and theory-based approximation are consistent with one-another.

4.5 Intervals for Predicted Values

4.5.1 Estimation and Prediction

Recall the icecream dispenser that is known to dispense icecream at a rate of 2 oz. per second on average, with individual amounts varying according to a normal distribution with mean 0 and standard deviation 0.5

Consider the following two questions:

1. On average, how much icecream will be dispensed for people who press the dispenser for 1.5 seconds?
2. If a single person presses the dispenser for 1.5 seconds, how much icecream will be dispensed?

The first question is one of estimation. The second pertains to prediction.

4.5.2 Uncertainty in Estimation and Prediction

In estimation and prediction, we must think about two sources of variability.

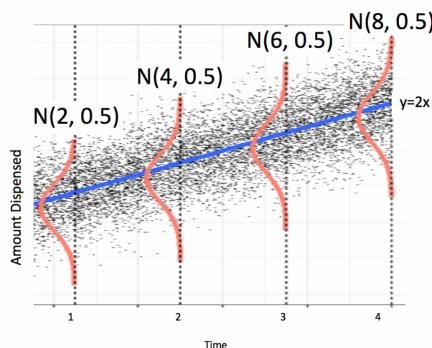
1. We are using data to estimate β_0 and β_1 , which introduces sampling variability.

2. Even if we did know β_0 and β_1 , there is variability in individual observations, which follows a $\mathcal{N}(0, \sigma)$ distribution.

In an estimation problem, we only need to think about (1). When predicting the value of a single new observation, we need to think about both (1) and (2).

Thus, intervals for predictions of individual observations carry more uncertainty and are wider than confidence intervals for $E(Y|X)$.

4.5.3 Uncertainty Between Individuals

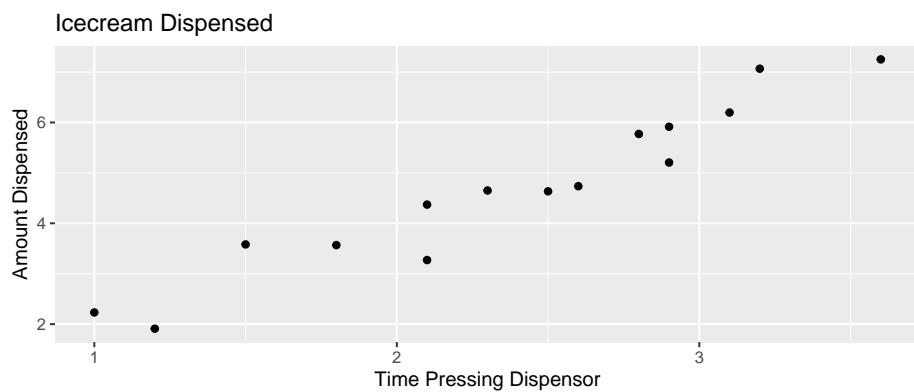


Modified from image at <https://bookdown.org/robak/bookdown-bysjh/ch-poissonreg.html>

4.5.4 Estimation in IC Example

```
kable(t(round(Icecream1, 2)))
```

time	1.00	1.20	1.50	1.80	2.10	2.10	2.30	2.50	2.60	2.80	2.90	2.90	3.1	3.20
amount	2.23	1.91	3.58	3.57	4.37	3.27	4.65	4.63	4.74	5.77	5.21	5.92	6.2	7.07



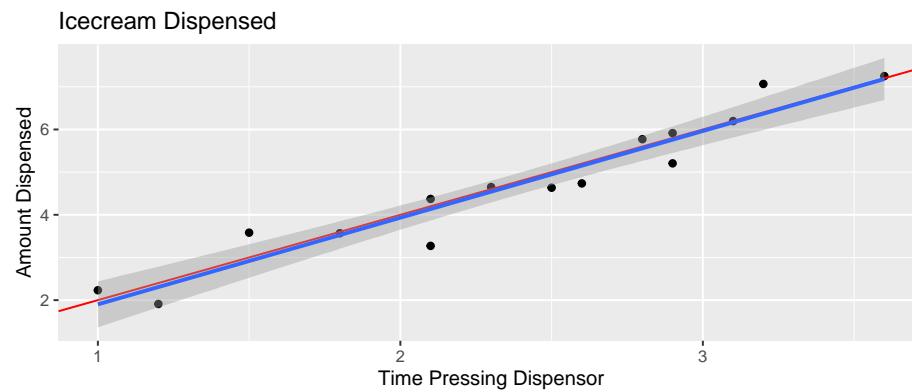
In the estimation setting, we are trying to determine the location of the regression line for the entire population.

Uncertainty comes from the fact that we only have data from a sample.

4.5.5 Estimation in IC Example

```
kable(t(round(Icecream1, 2)))
```

time	1.00	1.20	1.50	1.80	2.10	2.10	2.30	2.50	2.60	2.80	2.90	2.90	3.1	3.20	3.60
amount	2.23	1.91	3.58	3.57	4.37	3.27	4.65	4.63	4.74	5.77	5.21	5.92	6.2	7.07	7.25



4.5.6 Recall Ice Cream Model Output

```
summary(IC_Model)
```

```
##
## Call:
## lm(formula = amount ~ time), data = Icecream2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.5741 -0.2707 -0.1347  0.1644  1.2702
##
## Coefficients:
##             Estimate Std. Error t value    Pr(>|t|)
## (Intercept) -0.4528     0.4493 -1.008    0.332
## time         2.2085     0.1809 12.208 0.000000017 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.5126 on 13 degrees of freedom
## Multiple R-squared:  0.9198, Adjusted R-squared:  0.9136
## F-statistic:   149 on 1 and 13 DF,  p-value: 0.00000001699
```

4.5.7 Estimation in SLR

The first question:

“On average, how much icecream will be dispensed for people who press the dispensor for 1.5 seconds?”

Is a question of estimation. It is of the form, for a given X , on average what do we expect to be true of Y .

In the icecream question, we can answer this exactly, since we know β_0 and β_1 .

In a real situation, we don’t know these and have to estimate them from the data, which introduces uncertainty.

The standard error for an expected response $E(Y|X)$ is

$$SE(\hat{Y}|X = x^*) = s \sqrt{\frac{1}{n} + \frac{(x^* - \bar{x})^2}{\sum_{i=1}^n (x_i - \bar{x})^2}}$$

A 95% confidence interval for $E(Y|X = x^*)$ is given by

$$b_0 + b_1 x^* \pm t^* SE(\hat{Y}|X = x^*)$$

4.5.8 CI Calculation for Expected Response

Confidence interval for $E(Y|(X = 1.5))$:

$$\begin{aligned} & b_0 + b_1 x^* \pm t^* SE(\hat{Y}|X = x^*) \\ &= b_0 + b_1 x^* \pm 2s \sqrt{\frac{1}{n} + \frac{(x^* - \bar{x})^2}{\sum_{i=1}^n (x_i - \bar{x})^2}} \\ &= -0.4527901 + 2.2084662 \pm 20.512594 \sqrt{\frac{1}{15} + \frac{(1.5 - 2.3733)^2}{8.02933}} \end{aligned}$$

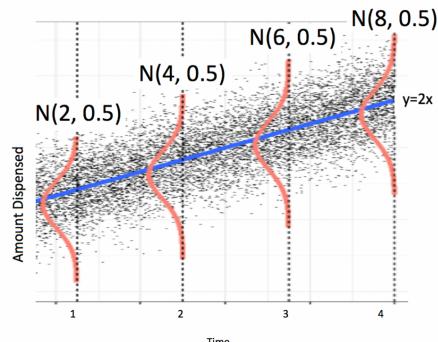
4.5.9 Confidence Interval for Expected Response in R

```
predict(IC_Model, newdata=data.frame(time=1.5), interval = "confidence", level=0.95)

##          fit      lwr      upr
## 1 2.859909 2.414664 3.305154
```

We are 95% confident that the mean amount dispensed when held for 1.5 seconds is between 2.41 and 3.31 oz.

4.5.10 Recall Uncertainty Between Individuals



Modified from image at <https://bookdown.org/robback/bookdown-bysh/ch-poissonreg.html>

4.5.11 Prediction Variance in SLR

Fact: For two independent random quantities, the variance of the sum is the sum of the variances.

$$\text{Var}(\mathbb{E}(Y|X = x^*)) = \sigma^2 \left(\frac{1}{n} + \frac{(x^* - \bar{x})^2}{\sum_{i=1}^n (x_i - \bar{x})^2} \right)$$

$$\text{Var}(Y|X) = \text{Var}(\epsilon_i) = \sigma^2$$

Thus the variance associated with predicted value $Y^*|(X = x^*)$ is

$$\sigma^2 \left(\frac{1}{n} + \frac{(x^* - \bar{x})^2}{\sum_{i=1}^n (x_i - \bar{x})^2} \right) + \sigma^2$$

4.5.12 Prediction Standard Error in SLR

Variance associated with predicted value $Y^*|(X = x^*)$:

$$\sigma^2 \left(\frac{1}{n} + \frac{(x^* - \bar{x})^2}{\sum_{i=1}^n (x_i - \bar{x})^2} \right) + \sigma^2$$

Thus the standard error for the predicted value is

$$s \sqrt{\left(\frac{1}{n} + \frac{(x^* - \bar{x})^2}{\sum_{i=1}^n (x_i - \bar{x})^2} \right) + 1}$$

4.5.13 Prediction interval in SLR

A prediction interval for $Y^*|(X = x^*)$ is given by

$$\beta_0 + \beta_1 x^* \pm t^* s \sqrt{\left(\frac{1}{n} + \frac{(x^* - \bar{x})^2}{\sum_{i=1}^n (x_i - \bar{x})^2} \right) + 1}$$

4.5.14 General Prediction Interval

In general, a prediction interval is

$$\beta_0 + \beta_1 x_1^* + \dots + \beta_n x_p^* \pm t^* \sqrt{\text{SE}(\mathbb{E}(Y|X = x^*)) + s^2}$$

4.5.15 Confidence Interval in R

```
predict(IC_Model, newdata=data.frame(time=1.5), interval = "confidence", level=0.95)

##          fit      lwr      upr
## 1 2.859909 2.414664 3.305154
```

We are 95% confident that the mean amount of ice cream dispensed when the dispenser is held for 1.5 seconds is between 2.41 and 3.31 oz.

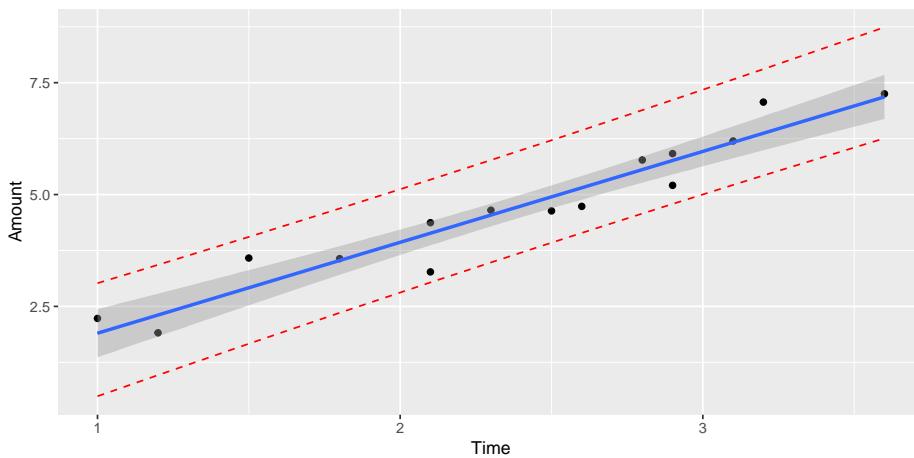
4.5.16 Prediction Interval in R

```
predict(IC_Model, newdata=data.frame(time=1.5), interval = "prediction", level=0.95)

##          fit      lwr      upr
## 1 2.859909 1.66636 4.053459
```

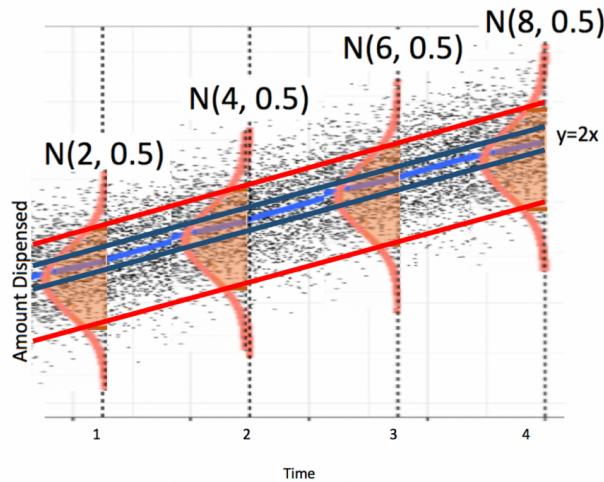
We are 95% confident that in individual who holds the dispenser for 1.5 seconds will get between 1.67 and 4.05 oz of ice cream.

4.5.17 Confidence and Prediction Intervals



The prediction interval (in red) is wider than the confidence interval (in blue), since it must account for variability between individuals, in addition to sampling variability.

4.5.18 Confidence and Prediction Bands



Modified from image at <https://bookdown.org/robact/bookdown-bysh/ch-poissonreg.html>

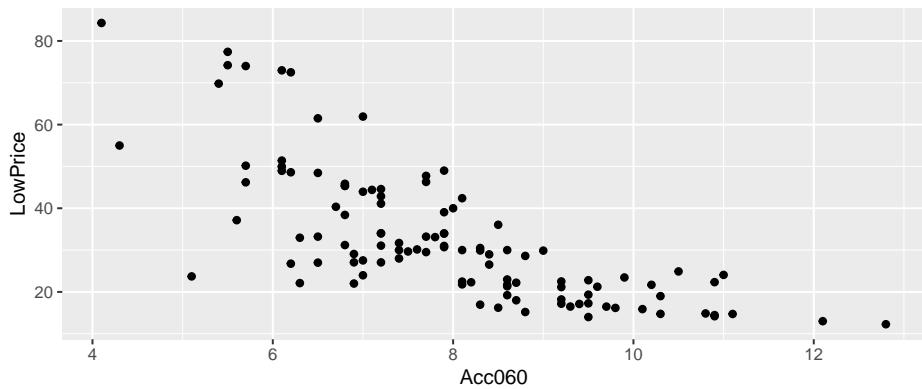
4.5.19 Model for Car Price and Acceleration Time

Recall the regression line estimating the relationship between a car's price and acceleration time.

This line was calculated using a sample of 110 cars, released in 2015.

$$\text{Price}_i = \beta_0 + \beta_1 \times \text{Acc. Time}_i + \epsilon_i, \text{ where } \epsilon_i \sim \mathcal{N}(0, \sigma).$$

CarsA060



4.5.20 Acc060 Model Output

```
summary(Cars_M_A060)

##
## Call:
## lm(formula = LowPrice ~ Acc060, data = Cars2015)
##
## Residuals:
##   Min     1Q   Median     3Q    Max
## -29.512  -6.544  -1.265   4.759  27.195
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept) 89.9036    5.0523  17.79 <0.0000000000000002 ***
## Acc060      -7.1933    0.6234 -11.54 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.71 on 108 degrees of freedom
## Multiple R-squared:  0.5521, Adjusted R-squared:  0.548
## F-statistic: 133.1 on 1 and 108 DF,  p-value: < 0.0000000000000022
```

4.5.21 Questions of Interest for Car Acceleration

1. What is a reasonable range for the average price of all new 2015 cars that can accelerate from 0 to 60 mph in 7 seconds?
2. If a car I am looking to buy can accelerate from 0 to 60 mph in 7 seconds, what price range should I expect?

4.5.22 Confidence Interval for Average Price

What is a reasonable range for the average price of all new 2015 cars that can accelerate from 0 to 60 mph in 7 seconds?

```
predict(Cars_M_A060, newdata=data.frame(Acc060=7), interval="confidence", level=0.95)

##       fit      lwr      upr
## 1 39.5502 37.21856 41.88184
```

We are 95% confident that the average price of new 2015 cars that accelerate from 0 to 60 mph in 7 seconds is between 37.2 and 41.9 thousand dollars.

Note: this is a confidence interval for $\beta_0 - 7\beta_1$.

4.5.23 Prediction Interval for Price of a Single Car

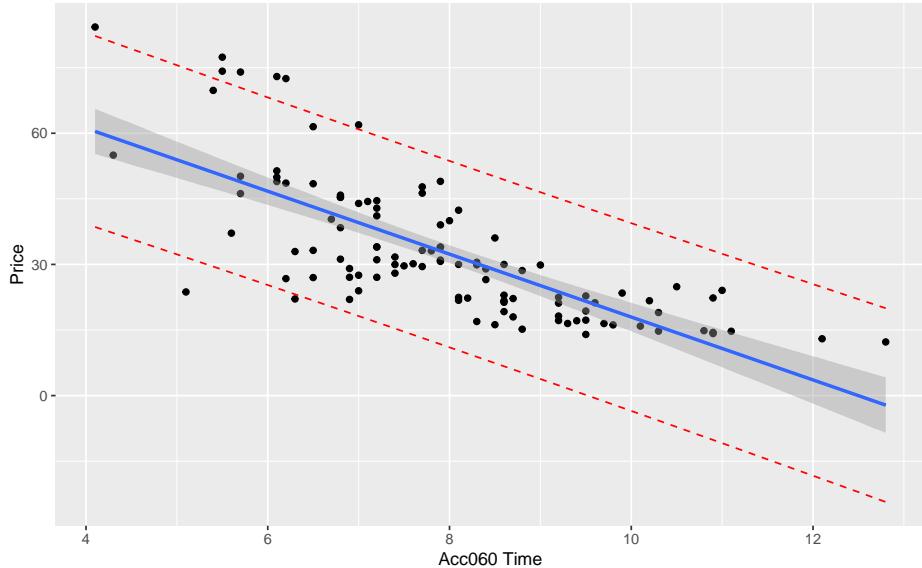
If a car I am looking to buy can accelerate from 0 to 60 mph in 7 seconds, what price range should I expect?

```
predict(Cars_M_A060, newdata=data.frame(Acc060=7), interval="prediction", level=0.95)

##      fit      lwr      upr
## 1 39.5502 18.19826 60.90215
```

We are 95% confident that a single new 2015 car that accelerates from 0 to 60 mph in 7 seconds will cost between 18.2 and 60.9 thousand dollars.

4.5.24 Visualization of Intervals for Car Prices

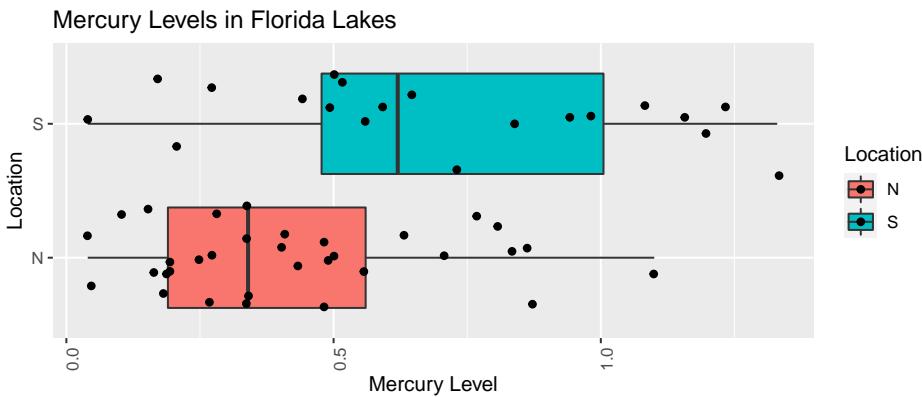


4.5.25 Intervals for Mercury Florida Lakes

Recall our sample of 53 Florida Lakes, 33 in the north, and 20 in the south.

$$\text{Mercury}_i = \beta_0 + \beta_1 \times \text{I}_{\text{South}_i} + \epsilon_i, \text{ where } \epsilon_i \sim \mathcal{N}(0, \sigma).$$

LakesBP



4.5.26 Lakes Model Conclusions

```
summary(Lakes_M)

##
## Call:
## lm(formula = AvgMercury ~ Location, data = FloridaLakes)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -0.65650 -0.23455 -0.08455  0.24350  0.67545 
## 
## Coefficients:
##             Estimate Std. Error t value    Pr(>|t|)    
## (Intercept) 0.42455   0.05519   7.692 0.000000000441 ***
## LocationS   0.27195   0.08985   3.027   0.00387 **  
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.3171 on 51 degrees of freedom
## Multiple R-squared:  0.1523, Adjusted R-squared:  0.1357 
## F-statistic: 9.162 on 1 and 51 DF,  p-value: 0.003868
```

4.5.27 Lakes Questions of Interest

1. Calculate an interval that we are 95% confident contains the mean mercury concentration for all lakes in Northern Florida. Do the same for Southern Florida.
2. Calculate an interval that we are 95% confident contains the mean mercury concentration for an individual lake in Northern Florida. Do the same for

a lake in Southern Florida.

4.5.28 Confidence Interval for Florida Lakes

```
predict(Lakes_M, newdata=data.frame(Location=c("N", "S")), interval="confidence", level=0.95)

##          fit      lwr      upr
## 1 0.4245455 0.3137408 0.5353501
## 2 0.6965000 0.5541689 0.8388311
```

We are 95% confident that the mean mercury level in North Florida is between 0.31 and 0.54 ppm.

We are 95% confident that the mean mercury level in South Florida is between 0.55 and 0.84 ppm.

Note: these are confidence intervals for β_0 , and $\beta_0 + \beta_1$, respectively.

4.5.29 Prediction Interval for Florida Lakes

```
predict(Lakes_M, newdata=data.frame(Location=c("N", "S")), interval="prediction", level=0.95)

##          fit      lwr      upr
## 1 0.4245455 -0.22155101 1.070642
## 2 0.6965000  0.04425685 1.348743
```

We are 95% confident that an individual lake in North Florida will have mercury level between 0 and 1.07 ppm.

We are 95% confident that the mean mercury level in South Florida is between 0.04 and 1.35 ppm.

Note that the normality assumption, which allows for negative mercury levels leads to a somewhat nonsensical result.

4.6 Regression Model Assumptions

4.6.1 What We're Assuming

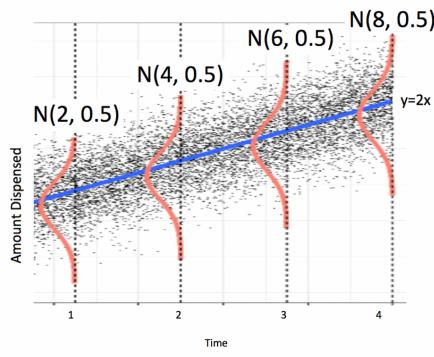
Let's think carefully about what we are assuming in order to use the hypothesis tests and confidence intervals provided in R.

The statement $Y_i = \beta_0 + \beta_1 X_{i1} + \dots + \beta_p X_{ip} + \epsilon_i$, with $\epsilon_i \sim \mathcal{N}(0, \sigma)$ implies the following:

1. Linearity: the expected value of Y is a linear function of X_1, X_2, \dots, X_p .

2. Normality: Given the values of X_1, X_2, \dots, X_p , Y follows a normal distribution.
3. Constant Variance: Regardless of the values of X_1, X_2, \dots, X_p , the variance (or standard deviation) in the normal distribution for Y is the same.
4. Independence: each observation is independent of the rest.

4.6.2 Illustration of Model Assumptions



Modified from image at <https://bookdown.org/robak/bookdown-bysh/ch-poissonreg.html>

4.6.3 Comments on Model Assumptions

We know that these held true in the ice cream example, because we generated the data in a way that was consistent with these. In practice, we will have only the data, without knowing the exact mechanism that produced it. We should only rely on the t-distribution based p-values and confidence intervals in the R output if these appear to be reasonable assumptions.

4.6.4 Situations that Violate Model Assumptions

Let's generate some data that violate the model assumptions. Then we'll look at how to detect these violations.

```
set.seed(10102020)
time <- runif(50, 1,3)
amount <- 2*time + rnorm(50, 0, 0.5) ## no violation
amount_lin_viol <- 2*time^2 + rnorm(50, 0, 0.5) ## linearity violation
amount_norm_viol <- 2*time + rexp(50, 1)-1
amount_cvar_viol <- 2*time + rnorm(50,0,time^2)
Violations <- data.frame(amount, amount_lin_viol, amount_norm_viol, amount_cvar_viol)
no_viol_Model <- lm(data=Violations, amount ~ time)
```

```
lin_viol_Model <- lm(data=Violations, amount_lin_viol~time)
norm_viol_Model <- lm(data=Violations, amount_norm_viol~time)
cvar_viol_Model <- lm(data=Violations, amount_cvar_viol~time)
```

4.6.5 Checking Model Assumptions

The following plots are useful when assessing the appropriateness of the normal error regression model.

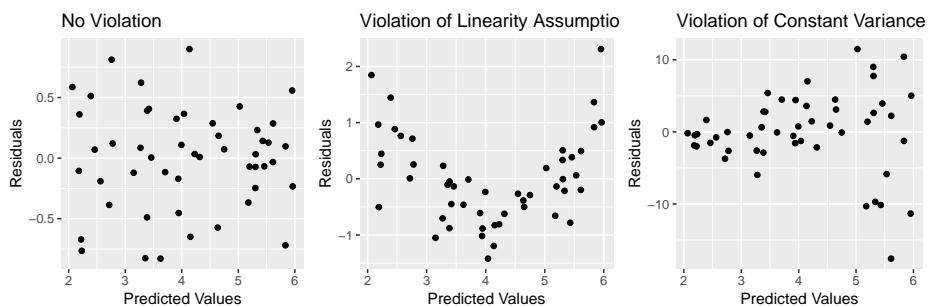
1. Scatterplot of residuals against predicted values
2. Histogram of standardized residuals
 - heavy skewness indicates a problem with normality assumption
3. Normal quantile plot
 - severe departures from diagonal line indicate problem with normality assumption

4.6.6 Residual vs Predicted Plots

These plots are useful for detecting issues with the linearity or constant variance assumption.

- curvature indicates a problem with linearity assumption
- “funnel” or “megaphone” shape indicates problem with constant variance assumption

```
P1 <- ggplot(data=Violations, aes(y=no_viol_Model$residuals, x=no_viol_Model$fitted.values))
P2 <- ggplot(data=Violations, aes(y=lin_viol_Model$residuals, x=no_viol_Model$fitted.values))
P3 <- ggplot(data=Violations, aes(y=cvar_viol_Model$residuals, x=no_viol_Model$fitted.values))
grid.arrange(P1, P2, P3, ncol=3)
```



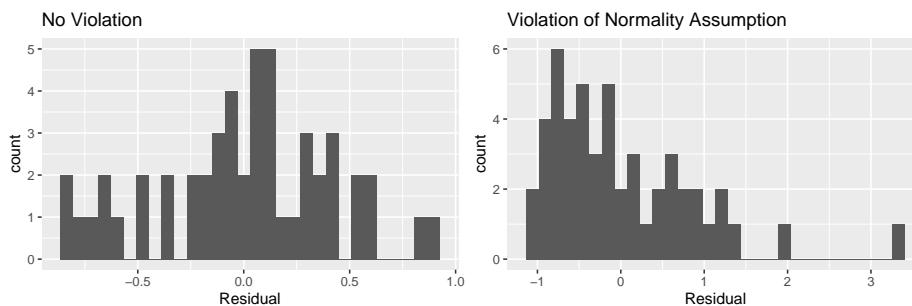
If there is only one explanatory variable, plotting the residuals against that variable reveals the same information.

4.6.7 Histogram of Residuals

Useful for assessing normality assumption.

- Severe skewness indicates violation of normality assumption

```
P1 <- ggplot(data=Violations, aes(x=no_viol_Model$residuals)) + geom_histogram() + ggtitle("No Violation")
P2 <- ggplot(data=Violations, aes(x=norm_viol_Model$residuals)) + geom_histogram() + ggtitle("Violation of Normality Assumption")
grid.arrange(P1, P2, ncol=2)
```



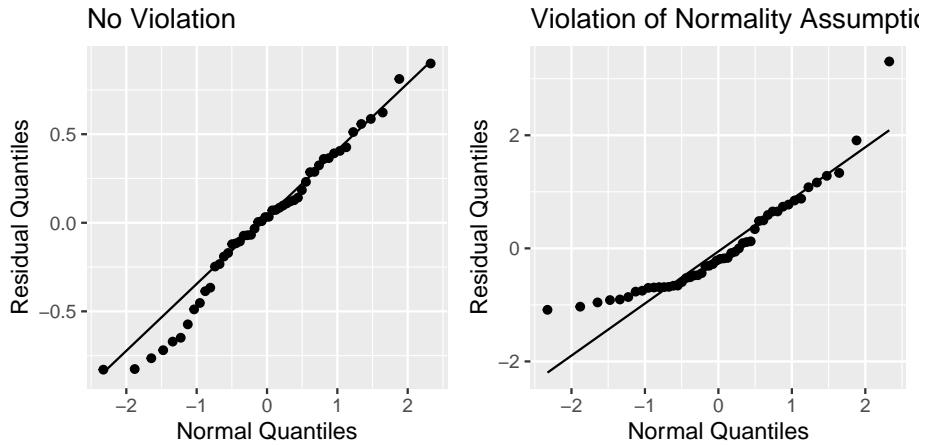
4.6.8 Normal Quantile-Quantile (QQ) Plot

Sometimes histograms can be inconclusive, especially when sample size is smaller.

A Normal quantile-quantile plot displays quantiles of the residuals against the expected quantiles of a normal distribution.

- Severe departures from diagonal line indicate a problem with normality assumption.

```
P1 <- ggplot(data=Violations, aes(sample = no_viol_Model$residuals)) + stat_qq() + stat_qq_line()
P2 <- ggplot(data=Violations, aes(sample = norm_viol_Model$residuals)) + stat_qq() + stat_qq_line()
grid.arrange(P1, P2, ncol=2)
```



4.6.9 Checking Model Assumptions - Independence

Independence is often difficult to assess through plots of data, but it is important to think about whether there were factors in the data collection that would cause some observations to be more highly correlated than others.

For example:

1. People in the study who are related.
2. Some plants grown in the same greenhouse and others in different greenhouses.
3. Some observations taken in same time period and others at different times.

All of these require more complicated models that account for correlation using spatial and time structure.

4.6.10 Summary of Checks for Model Assumptions

Model assumption	How to detect violation
Linearity	Curvature in residual plot
Constant Variance	Funnel shape in residual plot

Model assumption	How to detect violation
Normality	Skewness is histogram of residuals or departure from diag. line in QQ plot
Independence	No graphical check, carefully examine data collection

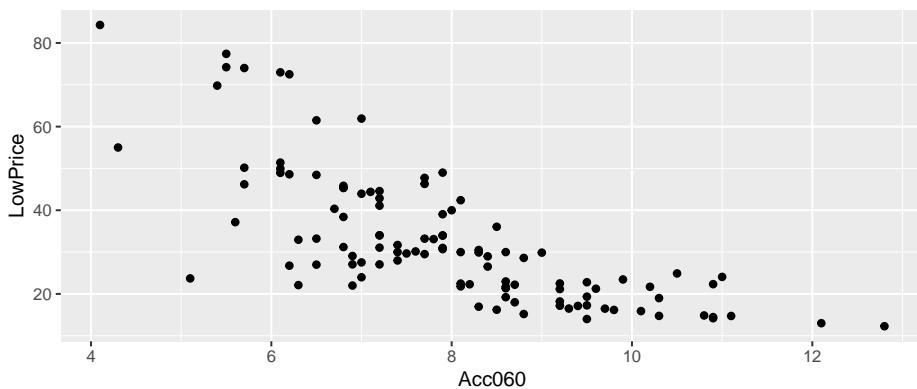
4.6.11 Model for Car Price and Acceleration Time

Recall the regression line estimating the relationship between a car's price and acceleration time.

This line was calculated using a sample of 110 cars, released in 2015.

$\text{Price}_i = \beta_0 + \beta_1 \times \text{Acc. Time}_i + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \sigma)$.

CarsA060



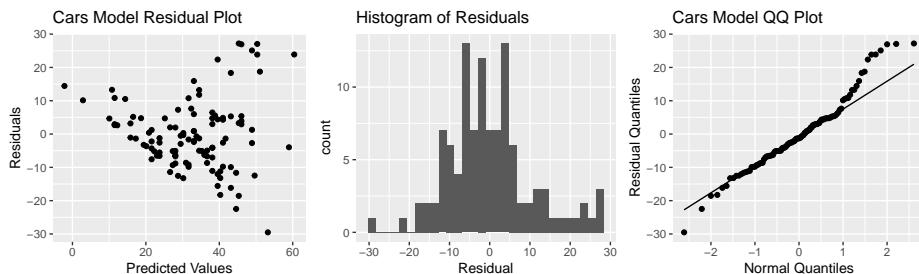
4.6.12 What We're Assuming in Cars A060 Model

1. Linearity: the expected price of a car is a linear function of its acceleration time.
2. Normality: for any given acceleration time, the prices of actual cars follow a normal distribution. For example the distribution of prices for all cars that accelerate from 0 to 60 in 8 seconds is normal, and so is the distribution of prices of cars that accelerate from 0 to 60 in 10 seconds (though these normal distributions have different means.)
3. Constant Variance: the normal distribution for prices is the same for all acceleration times.
4. Independence: no two cars are any more alike than any others.

We should only use the p-values and confidence intervals provided by R, which depend on the t-distribution approximation, if we believe these assumptions are reasonable.

4.6.13 Checking Model Assumptions for A060 Model

```
P1 <- ggplot(data=Cars2015, aes(y=Cars_M_A060$residuals, x=Cars_M_A060$fitted.values))
P2 <- ggplot(data=Cars2015, aes(x=Cars_M_A060$residuals)) + geom_histogram() + ggtitle("Histogram of Residuals")
P3 <- ggplot(data=Cars2015, aes(sample = Cars_M_A060$residuals)) + stat_qq() + stat_qq_line()
grid.arrange(P1, P2, P3, ncol=3)
```



There is a funnel-shape in the residual plot, indicating a concern about the constant variance assumption. There appears to be more variability in prices for more expensive cars than for cheaper cars. There is also some concern about the normality assumption, as the histogram and QQ plot indicate right-skew in the residuals.

4.6.14 Normal Model Car Acceleration Time

```
summary(Cars_M_A060)

##
## Call:
## lm(formula = LowPrice ~ Acc060, data = Cars2015)
##
## Residuals:
##    Min     1Q   Median     3Q    Max
## -29.512  -6.544  -1.265   4.759  27.195
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept) 89.9036    5.0523  17.79 <0.0000000000000002 ***
## Acc060      -7.1933    0.6234 -11.54 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.71 on 108 degrees of freedom
## Multiple R-squared:  0.5521, Adjusted R-squared:  0.548
## F-statistic: 133.1 on 1 and 108 DF,  p-value: < 0.0000000000000022
```

The large t-statistic and small p-value provide strong evidence that $\beta_1 \neq 0$. This means there is strong evidence of a relationship between price and acceleration time.

4.6.15 Confidence Interval for β_1 in Cars Example

```
confint(Cars_M_A060, level=0.95)
```

```
##              2.5 %    97.5 %
## (Intercept) 79.888995 99.918163
## Acc060      -8.429027 -5.957651
```

We are 95% confident that the average price of a new 2015 car decreases between 8.43 and 5.96 thousand dollars for each additional second it takes to accelerate from 0 to 60 mph.

Bootstrap Confidence Interval for β_1 :

```
##      2.5%    97.5%
## -8.751976 -5.670654
```

The bootstrap confidence interval is slightly wider than the one based on the t-approximation. This difference can be attributed to the questions about the constant variance and normality assumptions.

4.6.16 Confidence Intervals for Expected Price Given Acc060

```
predict(Cars_M_A060, newdata=data.frame(Acc060=7), interval="confidence")

##          fit      lwr      upr
## 1 39.5502 37.21856 41.88184

predict(Cars_M_A060, newdata=data.frame(Acc060=10), interval="confidence")

##          fit      lwr      upr
## 1 17.97018 14.71565 21.22472
```

We are 95% confident that the mean price for all cars that can accelerate from 0 to 60 mph in 7 seconds is between 37.2 and 41.9 thousand dollars.

We are 95% confident that the mean price for all cars that can accelerate from 0 to 60 mph in 10 seconds is between 14.7 and 22.2 thousand dollars.

4.6.17 Prediction Intervals for Expected Price Given Acc060

```
predict(Cars_M_A060, newdata=data.frame(Acc060=7), interval="prediction")

##          fit      lwr      upr
## 1 39.5502 18.19826 60.90215

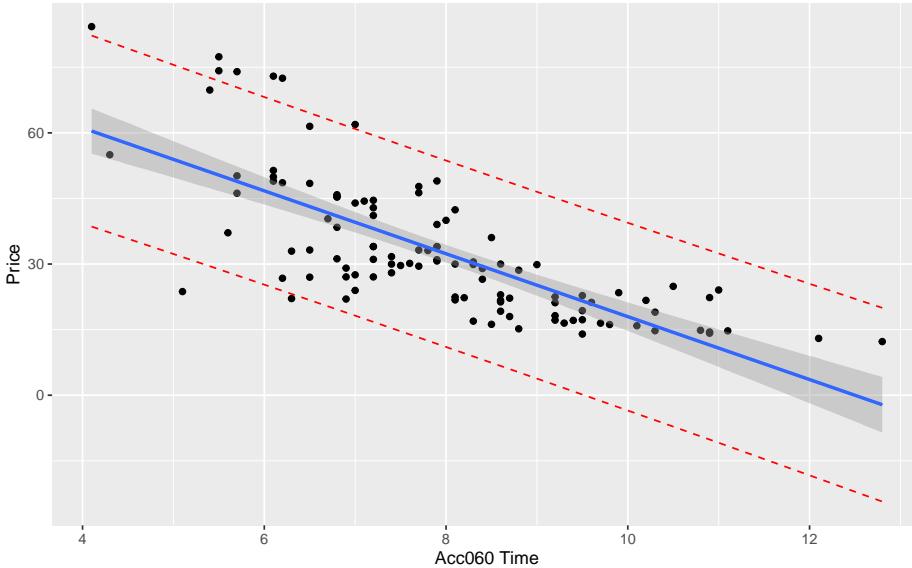
predict(Cars_M_A060, newdata=data.frame(Acc060=10), interval="prediction")

##          fit      lwr      upr
## 1 17.97018 -3.502148 39.44252
```

We are 95% confident that a single car that can accelerate from 0 to 60 mph in 7 seconds will cost between 18.2 thousand and 60.9 thousand dollars.

We are 95% confident that a single car that can accelerate from 0 to 60 mph in 10 seconds will cost between 0 thousand and 39.4 thousand dollars.

4.6.18 Confidence and Prediction Interval Illustration



4.6.19 Concerns about Intervals and Model Assumptions

- The confidence and prediction interval for the more expensive car ($\text{Acc060}=7$) is wider than for the less expensive one ($\text{Acc060}=10$). This seems inconsistent with the data, which showed more variability about prices for more expensive cars than less expensive ones.
 - The intervals are computed using same value for s , which is a result of the constant variance assumption. Our residual plot showed us this assumption might not be valid in this situation.
- The confidence and prediction intervals are symmetric about the expected price, even though the distribution of residuals was right-skewed. - This is the result of the normality assumption, which our histogram and QQ-plot showed might not be valid here.

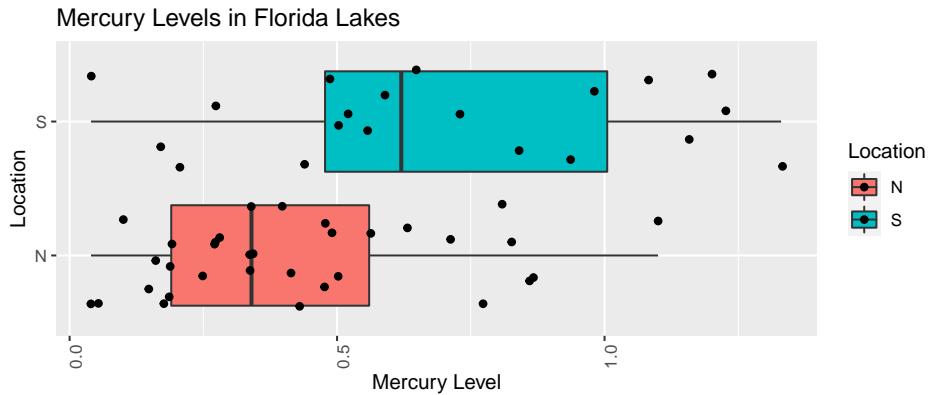
Since we had concerns about the model assumptions, the intervals might not be reliable. We saw that the confidence interval for β_1 differed somewhat, but not terribly, from the one produced via Bootstrap. It is harder to tell the degree to which the confidence and prediction intervals for price for a given acceleration time might be off, but we should treat these with caution.

4.6.20 Model for Mercury Florida Lakes

Recall our sample of 53 Florida Lakes, 33 in the north, and 20 in the south.

$\text{Mercury}_i = \beta_0 + \beta_1 \times I_{\text{South}_i} + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \sigma)$.

LakesBP



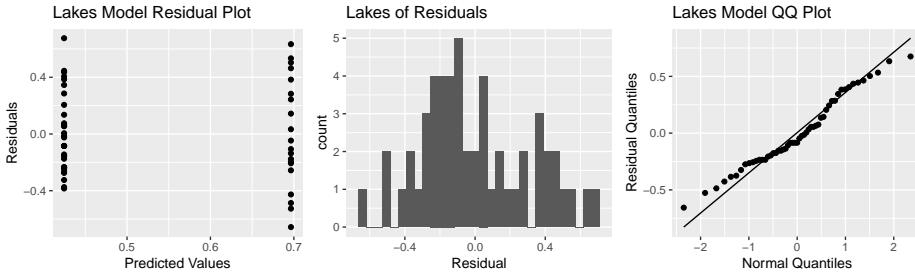
4.6.21 What We're Assuming in Lakes Model

1. Linearity: there is an expected mercury concentration for lakes in North Florida, and another for lakes in South Florida.
2. Normality: mercury concentrations of individual lakes in the north are normally distributed, and so are mercury concentrations in the south. These normal distributions might have different means.
3. Constant Variance: the normal distribution for mercury concentrations in North Florida has the same standard deviation as the normal distribution for mercury concentrations in South Florida
4. Independence: no two lakes are any more alike than any others. We might have concerns about this, due to some lakes being geographically closer to each other than others.

We should only use the p-values and confidence intervals provided by R, which depend on the t-distribution approximation, if we believe these assumptions are reasonable.

4.6.22 Checking Model Assumptions for Lakes Model

```
P1 <- ggplot(data=FloridaLakes, aes(y=Lakes_M$residuals, x=Lakes_M$fitted.values)) + g
P2 <- ggplot(data=FloridaLakes, aes(x=Lakes_M$residuals)) + geom_histogram() + ggtitle(
P3 <- ggplot(data=FloridaLakes, aes(sample = Lakes_M$residuals)) + stat_qq() + stat_qq(
grid.arrange(P1, P2, P3, ncol=3)
```



Notice that we see two lines of predicted values and residuals. This makes sense since all lakes in North Florida will have the same predicted value, as will all lakes in Southern Florida.

There appears to be a little more variability in residuals for Southern Florida (on the right), than Northern Florida, causing some concern about the constant variance assumption.

Overall, though, the assumptions seem mostly reasonable.

4.6.23 Lakes Model Output

```
summary(Lakes_M)

##
## Call:
## lm(formula = AvgMercury ~ Location, data = FloridaLakes)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -0.65650 -0.23455 -0.08455  0.24350  0.67545 
## 
## Coefficients:
##             Estimate Std. Error t value    Pr(>|t|)    
## (Intercept) 0.42455   0.05519   7.692 0.000000000441 ***
## LocationS   0.27195   0.08985   3.027     0.00387 **  
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.3171 on 51 degrees of freedom
## Multiple R-squared:  0.1523, Adjusted R-squared:  0.1357 
## F-statistic: 9.162 on 1 and 51 DF,  p-value: 0.003868
```

4.6.24 Lakes Model Interpretations

$\text{Mercury}_i = \beta_0 + \beta_1 \times I_{\text{South}_i} + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \sigma)$.

β_0 represents the mean mercury concentration for lakes in North Florida. The large t-statistic and small p-value on the intercept line tell us there is strong evidence that the mean mercury level among all lakes in Northern Florida is not 0. This is neither surprising, nor informative.

β_1 represents the average difference in mercury concentrations between lakes in South and North Florida. The large t-statistic and small p-value tell us there is strong evidence of a difference in mean mercury concentrations in South Florida, compared to North Florida.

4.6.25 t-based Confidence Intervals for Lakes

```
confint(Lakes_M, level=0.95)

##           2.5 %    97.5 %
## (Intercept) 0.31374083 0.5353501
## LocationS   0.09157768 0.4523314
```

We can be 95% confident that the mean mercury concentration for lakes in North Florida is between 0.314 and 0.535 ppm.

We can be 95% confident that the mean mercury concentration for lakes in South Florida is between 0.09 and 0.45 ppm higher than for lakes in North Florida.

95% Bootstrap CI for β_1

```
##      2.5%    97.5%
## 0.08095682 0.46122992
```

In this situation, the bootstrap interval and the interval obtained using the t-approximation are almost identical. The model assumptions appeared reasonable, this is not surprising.

4.6.26 Impact of Model Assumption Violations

Model assumption	Impact
Linearity	predictions and intervals unreliable

Model assumption	Impact
Constant	predictions still reliable;
Variance	some intervals will be too wide and others too narrow.
Normality	predictions still reliable; intervals will be symmetric when they shouldn't be
Independence	predictions unreliable and intervals unreliable

4.6.27 General Comments on Model Assumptions

- We shouldn't think about model assumptions being satisfied as a yes/no question.
- In reality assumptions are never perfectly satisfied, so it's a question of how severe violations must be in order to impact results. This is context dependent.
- A model might be reasonable for certain purposes (i.e. confidence interval for β_1) but not for others (i.e. prediction of response value for new observation).
- When model assumptions are a concern, consider either a more flexible technique (such as a nonparametric method or statistical machine learning algorithm), or perform a transformation of the response or explanatory variables before fitting the model

- Remember that all statistical techniques are approximations

4.7 Transformations on Response Variable

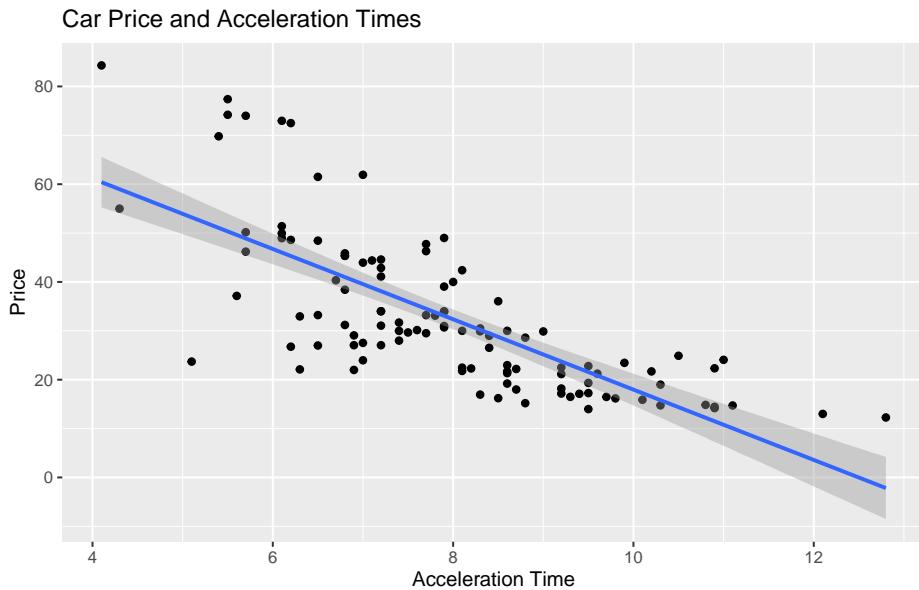
4.7.1 Transformations

When residual plots yield model inadequacy, we might try to correct these by applying a transformation to the response variable.

When working with “heavy-tailed”, or right-skewed data, it is often helpful to work with the logarithm of the response variable.

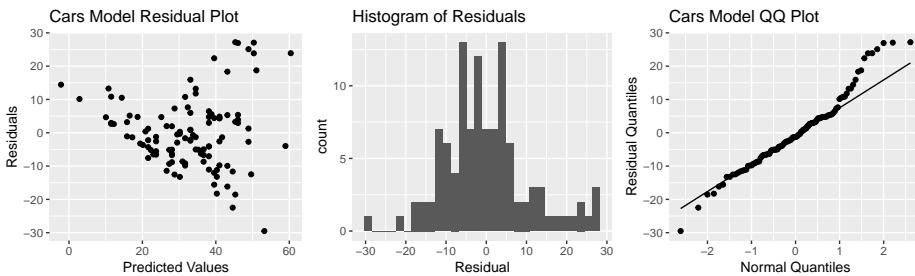
Note: In R, `log()` denotes the natural (base e) logarithm, often denoted `ln()`. We can actually use any logarithm, but the natural logarithm is commonly used.

4.7.2 Recall Model Assumptions for A060 Model



4.7.3 Recall Model Assumptions for A060 Model

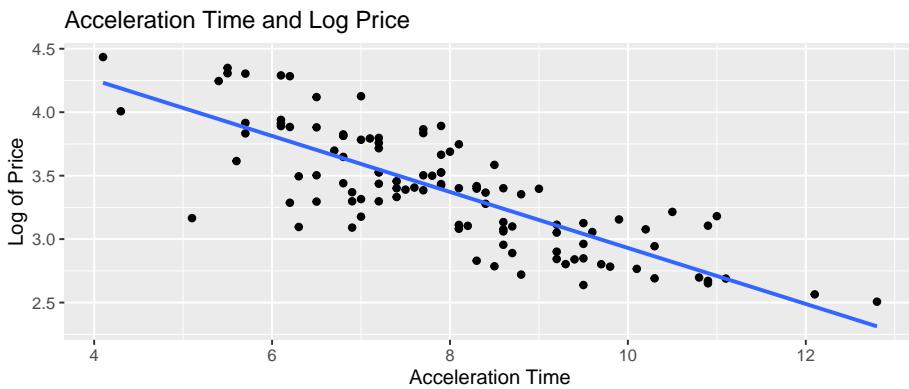
```
P1 <- ggplot(data=Cars2015, aes(y=Cars_M_A060$residuals, x=Cars_M_A060$fitted.values))
P2 <- ggplot(data=Cars2015, aes(x=Cars_M_A060$residuals)) + geom_histogram() + ggtitle("Histogram of Residuals")
P3 <- ggplot(data=Cars2015, aes(sample = Cars_M_A060$residuals)) + stat_qq() + stat_qq_line()
grid.arrange(P1, P2, P3, ncol=3)
```



There is a funnel-shape in the residual plot, indicating a concern about the constant variance assumption. There appears to be more variability in prices for more expensive cars than for cheaper cars. There is also some concern about the normality assumption, as the histogram and QQ plot indicate right-skew in the residuals.

4.7.4 Plot of LogPrice and Acc060

```
ggplot(data=Cars2015, aes(x=Acc060, y=log(LowPrice))) + geom_point() +
  xlab("Acceleration Time") + ylab("Log of Price") +
  ggtitle("Acceleration Time and Log Price") + stat_smooth(method="lm", se=FALSE)
```



4.7.5 Model for Log Transform

```
Cars_M_Log <- lm(data=Cars2015, log(LowPrice)~Acc060)
summary(Cars_M_Log)
```

```
##
## Call:
## lm(formula = log(LowPrice) ~ Acc060, data = Cars2015)
##
```

```

## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.84587 -0.19396  0.00908  0.18615  0.53350
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)
## (Intercept) 5.13682   0.13021  39.45 <0.000000000000002 ***
## Acc060     -0.22064   0.01607 -13.73 <0.000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.276 on 108 degrees of freedom
## Multiple R-squared:  0.6359, Adjusted R-squared:  0.6325
## F-statistic: 188.6 on 1 and 108 DF,  p-value: < 0.0000000000000022

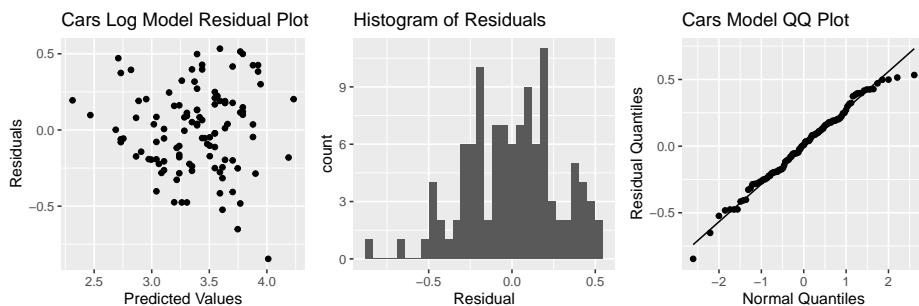
```

4.7.6 LogPrice Model: What We're Assuming

1. Linearity: the log of expected price of a car is a linear function of its acceleration time.
2. Normality: for any given acceleration time, the log of prices of actual cars follow a normal distribution.
3. Constant Variance: the normal distribution for log of price is the same for all acceleration times.
4. Independence: no two cars are any more alike than any others.

We should only use the p-values and confidence intervals provided by R, which depend on the t-distribution approximation, if we believe these assumptions are reasonable.

4.7.7 Model Assumption Check for Transformed Model



There is still some concern about constant variance, though perhaps not as much. The normality assumption appears more reasonable.

4.7.8 Model for Log of Car Price

$$\widehat{\text{Log Price}} = b_0 + b_1 \times \text{Acc060}$$

Thus

$$\widehat{\text{Price}} = e^{b_0 + b_1 \times \text{Acc060}}$$

$$e^{b_0} e^{b_1 \times \text{Acc060}}$$

4.7.9 Predictions using Transformed Model

Prediction Equation:

$$\widehat{\text{Price}} = e^{5.13582} e^{-0.22064 \times \text{Acc060}}$$

Predicted price for car that takes 7 seconds to accelerate:

$$\widehat{\text{Price}} = e^{5.13582} e^{-0.22064 \times 7} = 36.3$$

Predicted price for car that takes 10 seconds to accelerate:

$$\widehat{\text{Price}} = e^{5.13582} e^{-0.22064 \times 10} = 18.7$$

4.7.10 Predictions Using Log Model (Cont.)

Predictions are for $\log(\text{Price})$, so we need to exponentiate.

```
predict(Cars_M_Log, newdata=data.frame(Acc060=c(7)))
```

```
##      1
## 3.592343
exp(predict(Cars_M_Log, newdata=data.frame(Acc060=c(7))))
```

```
##      1
## 36.31908
```

A car that accelerates from 0 to 60 mph in 7 seconds is expected to cost 36.3 thousand dollars.

4.7.11 Model Interpretations for Transformed Model

- e^{b_0} is theoretically the expected price of a car that can accelerate from 0 to 60 mph in no time, but this is not a meaningful interpretation.
- For each additional second it takes a car to accelerate, price is expected to multiply by a factor of e^{b_1} .
- For each additional second in acceleration time, price is expected to multiply by a factor of $e^{-0.22} = 0.80$. Thus, each 1-second increase in acceleration time is estimated to be associated with a 20% drop in price, on average.

4.7.12 Confidence Intervals for Price using Log Model

```
confint(Cars_M_Log)
```

```
##               2.5 %      97.5 %
## (Intercept) 4.8787105 5.3949208
## Acc060     -0.2524862 -0.1887916
```

- We are 95% confident that the price of a car changes, on average, by multiplicative factor between $e^{-0.252} = 0.7773$ and $e^{-0.189} = 0.828$ for each additional second in acceleration time. That is, we believe the price decreases between 17% and 23% on average for each additional second in acceleration time.

4.7.13 Log Model: Confidence Interval for Expected Response

```
predict(Cars_M_Log, newdata=data.frame(Acc060=c(7)), interval="confidence")

##       fit      lwr      upr
## 1 3.592343 3.53225 3.652436

exp(predict(Cars_M_Log, newdata=data.frame(Acc060=c(7)), interval="confidence"))

##       fit      lwr      upr
## 1 36.31908 34.20083 38.56852
```

We are 95% confident that the mean price among all cars that accelerate from 0 to 60 mph in 7 seconds is between $e^{3.53225} = 34.2$ and $e^{3.652436} = 38.6$ thousand dollars.

4.7.14 Log Model: Prediction Interval for Expected Response

```
predict(Cars_M_Log, newdata=data.frame(Acc060=c(7)), interval="prediction")

##      fit      lwr      upr
## 1 3.592343 3.042041 4.142645

exp(predict(Cars_M_Log, newdata=data.frame(Acc060=c(7)), interval="prediction"))

##      fit      lwr      upr
## 1 36.31908 20.94796 62.96917
```

We are 95% confident that the expected price for a car that accelerates from 0 to 60 mph in 7 seconds is between $e^{3.04} = 20.9$ and $e^{4.14} = 63.9$ thousand dollars.

4.7.15 Confidence Interval Comparison

95% Confidence interval for average price of cars that take 7 seconds to accelerate:

Original Model:

```
predict(Cars_M_A060, newdata=data.frame(Acc060=7), interval="confidence", level=0.95)

##      fit      lwr      upr
## 1 39.5502 37.21856 41.88184
```

Transformed Model:

```
exp(predict(Cars_M_Log, newdata=data.frame(Acc060=c(7)), interval="confidence", level=0.95))

##      fit      lwr      upr
## 1 36.31908 34.20083 38.56852
```

4.7.16 Prediction Interval Comparison

95% Prediction interval for price of an individual car that takes 7 seconds to accelerate:

Original Model:

```
predict(Cars_M_A060, newdata=data.frame(Acc060=7), interval="prediction", level=0.95)

##      fit      lwr      upr
## 1 39.5502 18.19826 60.90215
```

Transformed Model:

```
exp(predict(Cars_M_Log, newdata=data.frame(Acc060=c(7)), interval="prediction", level=
```

```
##          fit      lwr      upr
## 1 36.31908 20.94796 62.96917
```

Notice that the transformed interval is not symmetric and allows for a longer “tail” on the right than the left.

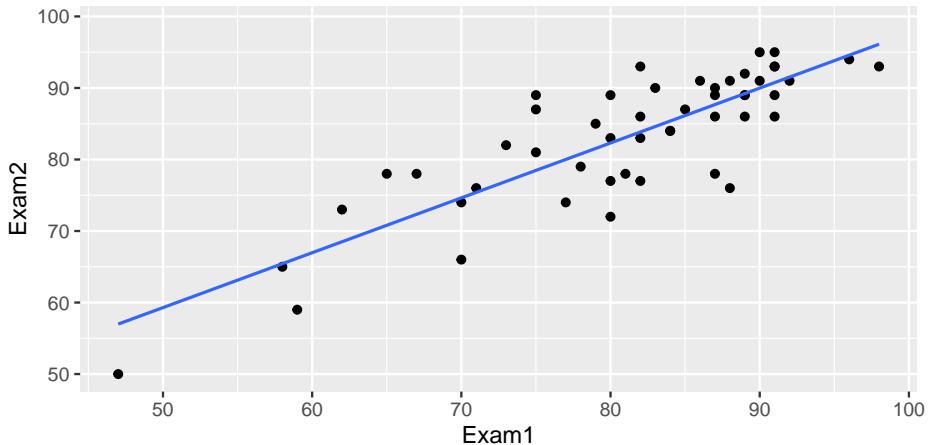
4.7.17 Comments on Transformations

- We could have used another transformation, such as $\sqrt{\text{Price}}$
- The log transform leads to a nice interpretation involving percent change. Other transformations might yield better predictions, but are often hard to interpret.
- There is often a tradeoff between model complexity and interpretability. We'll talk more about this.
- We did an example of a transformation in a model with a single explanatory variable.
- If the explanatory variable is categorical:
 - e^{b_0} represents the expected response in the baseline category
 - e^{b_j} represents the number of times larger the expected response in category j is, compared to the baseline category.
- When working with multiple regression models, it is still important to mention holding other variables constant when interpreting parameters associated with one of the variables.

4.8 The Regression Effect

4.8.1 The Regression Effect

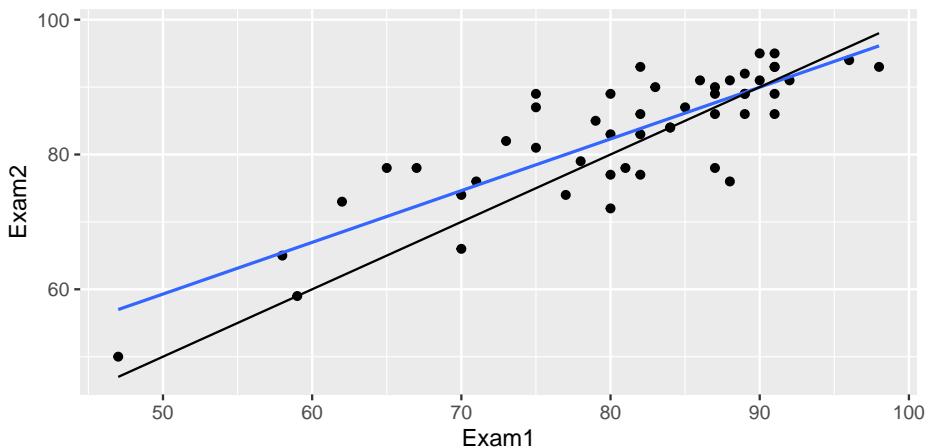
Exam 1 vs Exam 2 scores for intro stat students at another college



What is the relationship between scores on the two exams?

4.8.2 The Regression Effect

Exam 1 vs Exam 2 scores for intro stat students at another college



How many of the 6 students who scored below 70 on Exam 1 improved their scores on Exam 2?

How many of the 7 students who scored above 90 improved on Exam 2?

4.8.3 The Regression Effect

A low score on an exam is often the result of both poor preparation and bad luck.

250CHAPTER 4. THE NORMAL ERROR LINEAR REGRESSION MODEL

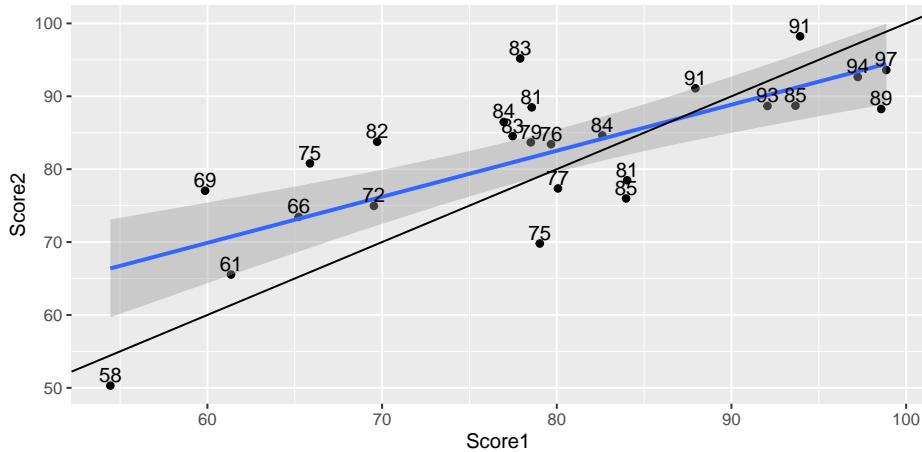
A high score often results from both good preparation and good luck.

While changes in study habits and preparation likely explain some improvement in low scores, we would also expect the lowest performers to improve simply because of better luck.

Likewise, some of the highest performers may simply not be as lucky on exam 2, so a small dropoff should not be interpreted as weaker understanding of the exam material.

4.8.4 Simulating Regression Effect

```
set.seed(110322018)
Understanding <- rnorm(25, 80, 10)
Score1 <- Understanding + rnorm(25, 0, 5)
Score2 <- Understanding + rnorm(25, 0, 5)
Understanding <- round(Understanding,0)
TestSim <- data.frame(Understanding, Score1, Score2)
ggplot(data=TestSim, aes(y=Score2, x=Score1)) + geom_point() + stat_smooth(method="lm")
  geom_abline(slope=1) + geom_text(aes(label=Understanding), vjust = 0, nudge_y = 0.5)
```



This phenomenon is called the **regression effect**.

4.8.5 Test Scores Simulation - Highest Scores

```
kable(head(TestSim%>%arrange(desc(Score1))))
```

Understanding	Score1	Score2
97	98.86412	93.60285
89	98.57157	88.25851
94	97.23330	92.65175
91	93.92857	98.23312
85	93.66503	88.70963
93	92.06243	88.67015

These students' success on test 1 is due to a strong understanding and good luck. We would expect the understanding to carry over to test 2 (provided the student continues to study in a similar way), but not necessarily the luck.

4.8.6 Test Scores Simulation - Lowest Scores

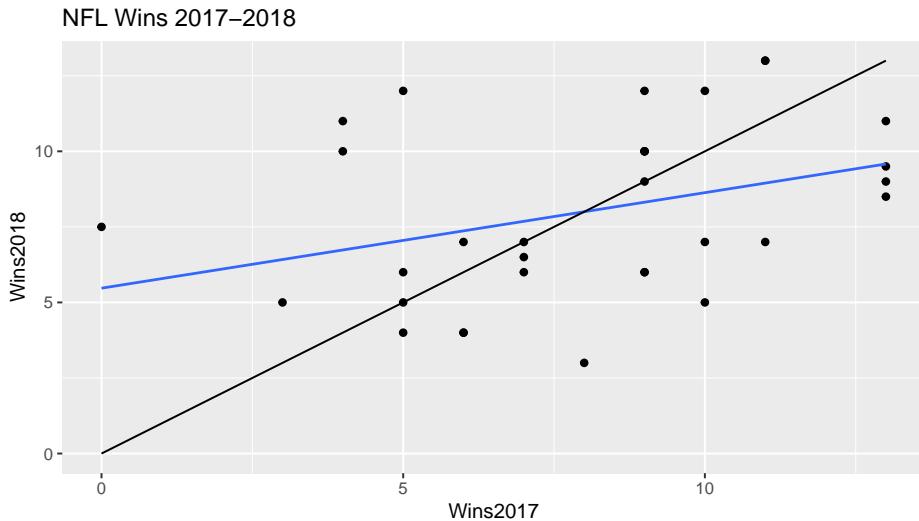
```
kable(head(TestSim %>% arrange(Score1)))
```

Understanding	Score1	Score2
58	54.44354	50.30597
69	59.86641	77.04696
61	61.35228	65.54305
66	65.22433	73.45304
75	65.87041	80.79416
72	69.53082	74.96092

These students' lack of success on test 1 is due to a low understanding and poor luck. We would expect the understanding to carry over to test 2 (unless the student improves their preparation), but not necessarily the luck.

4.8.7 Another Example

Wins by NFL teams in 2017 and 2018



4.8.8 Other Examples of Regression Effect

A 1973 article by Kahneman, D. and Tversky, A., “On the Psychology of Prediction,” Pysch. Rev. 80:237-251 describes an instance of the regression effect in the training of Israeli air force pilots.

Trainees were praised after performing well and criticized after performing badly. The flight instructors observed that “high praise for good execution of complex maneuvers typically results in a decrement of performance on the next try.”

Kahneman and Tversky write that :

“We normally reinforce others when their behavior is good and punish them when their behavior is bad. By regression alone, therefore, they [the trainees] are most likely to improve after being punished and most likely to deteriorate after being rewarded. Consequently, we are exposed to a lifetime schedule in which we are most often rewarded for punishing others, and punished for rewarding.”

Chapter 5

Logistic Regression

5.1 Logistic Regression

5.1.1 Modeling Binary Response

- So far, we have modeled only quantitative response variables.
- The normal error regression model makes the assumption that the response variable is normally distributed, given the value(s) of the explanatory variables.
- Now, we'll look at how to model a categorical response variable. We'll consider only situations where the response is binary (i.e. has 2 categories)

5.1.2 Credit Card Dataset

We'll consider a dataset pertaining to 10,000 credit cards. The goal is to predict whether or not the user will default on the payment, using information on the credit card balance, user's annual income, and whether or not the user is a student. Data come from Introduction to Statistical Learning by James, Witten, Hastie, Tibshirani.

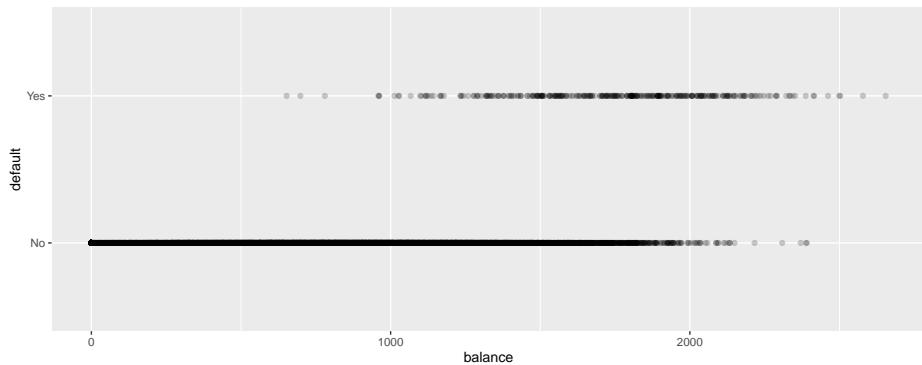
```
library(ISLR)
data(Default)
summary(Default)

##   default    student      balance          income
##   No :9667   No :7056   Min.   : 0.0   Min.   : 772
##   Yes: 333   Yes:2944   1st Qu.:481.7   1st Qu.:21340
##                           Median :823.6   Median :34553
```

```
##          Mean    : 835.4      Mean    :33517
## 3rd Qu.:1166.3      3rd Qu.:43808
## Max.   :2654.3      Max.   :73554
```

5.1.3 Default and Balance

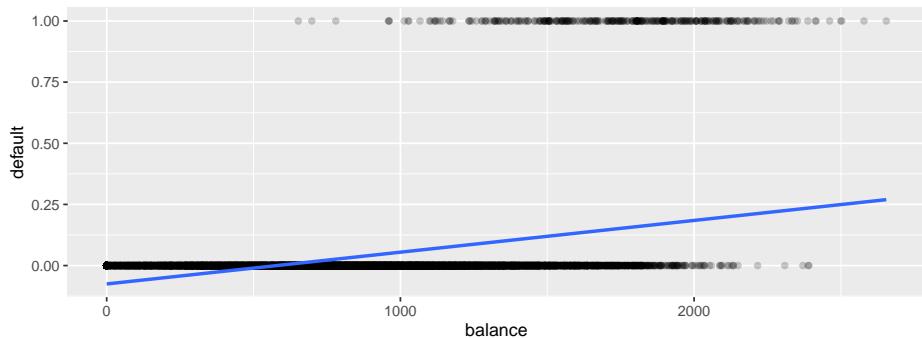
```
ggplot(data=Default, aes(y=default, x=balance)) + geom_point(alpha=0.2)
```



5.1.4 Linear Regression Model for Education Level

```
#convert default from yes/no to 0/1
Default$default <- as.numeric(Default$default=="Yes")
```

```
ggplot(data=Default, aes(y=default, x= balance)) + geom_point(alpha=0.2) + stat_smooth()
```



There are a lot of problems with this model!

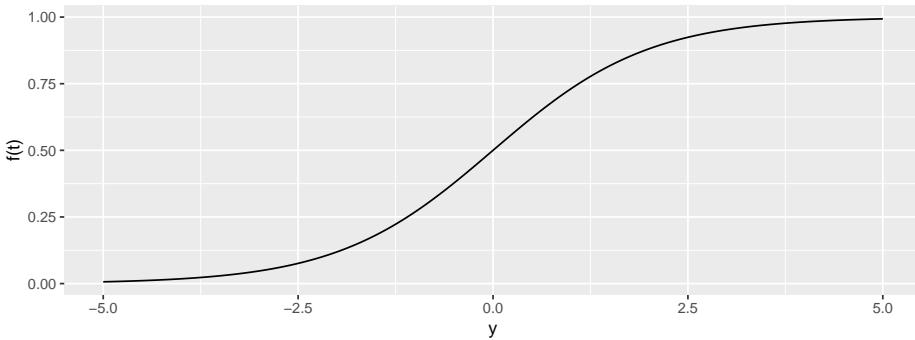
5.1.5 Transforming into interval (0,1)

- When Y takes on only values 0 or 1, $E(Y)$ can be interpreted as the probability that $Y = 1$ (denoted $P(Y = 1)$)
- The simple linear regression model assumes that $E(Y_i) = \beta_0 + \beta_1 x_i$. More generally, the normal error regression model assumes $E(Y_i) = \beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}$
- Assuming $E(Y_i)$ is a linear function of the x 's allows Y_1 to take on values outside (0,1).
- In order to obtain sensible probability estimates, we need to transform the values of $\beta_0 + \beta_1 x_i$ into the interval (0,1).
- To accomplish this, we'll use the function.

$$f(t) = \frac{e^t}{1 + e^t}$$

5.1.6 Transforming to (0,1)

The function $f(t) = \frac{e^t}{1+e^t}$ takes any real number t and returns a number in the interval (0,1).



5.1.7 The Logit Function

- If $p = \frac{e^t}{1+e^t}$, then $t = \log\left(\frac{p}{1-p}\right)$.
- The function $f(p) = \log\left(\frac{p}{1-p}\right)$ for $p \in (0, 1)$ is called the **logit function**.
- We can interpret p as $P(Y = 1)$, and $\frac{p}{1-p}$ is called the **odds** of $Y = 1$.
- Thus, $\text{logit}(p) = \log\left(\frac{p}{1-p}\right)$ returns the log of the odds.

- The function $f(t) = \frac{e^t}{1+e^t}$ for $t \in (-\infty, \infty)$ is called the inverse logit function.

5.1.8 The Logistic Regression Model

- Starting with our linear model $E(Y_i) = \beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}$, we need to transform $\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}$ into $(0,1)$.
- Let $\pi_i = \frac{e^{\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}}}{1 + e^{\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}}}$
- Then $0 \leq \pi_i \leq 1$, and π_i represents an estimate of $P(Y_i = 1)$.
- The **logistic regression model** assumes that:

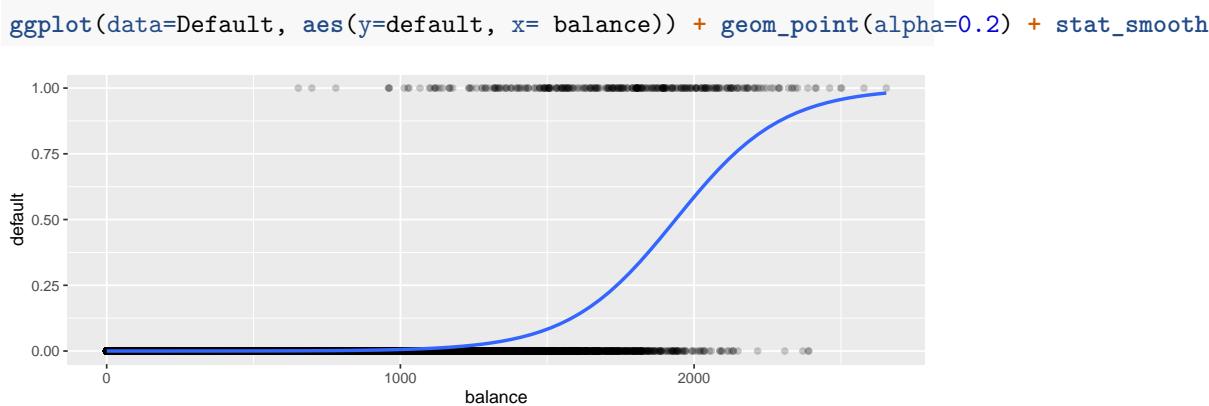
– $Y_i \in \{0, 1\}$

$$- P(Y_i = 1) = \pi_i = \frac{e^{\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}}}{1 + e^{\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}}}$$

$$\text{i.e. } \beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip} = \log\left(\frac{\pi_i}{1 - \pi_i}\right).$$

- Instead of assuming that the expected response is a linear function of the explanatory variables, we are assuming that it is a function of a linear function of the explanatory variables.
- This is an example of a *generalized linear model*, a topic that will be explored in detail in STAT 455.

5.1.9 Logistic Regression Model for Balance



5.1.10 Fitting the Logistic Regression Model in R

```
CCDefault_M <- glm(data=Default, default ~ balance, family = binomial(link = "logit"))
summary(CCDefault_M)

##
## Call:
## glm(formula = default ~ balance, family = binomial(link = "logit"),
##      data = Default)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.2697 -0.1465 -0.0589 -0.0221  3.7589
##
## Coefficients:
##             Estimate Std. Error z value     Pr(>|z|)
## (Intercept) -10.6513306  0.3611574 -29.49 <0.0000000000000002 ***
## balance      0.0054989  0.0002204   24.95 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 2920.6 on 9999 degrees of freedom
## Residual deviance: 1596.5 on 9998 degrees of freedom
## AIC: 1600.5
##
## Number of Fisher Scoring iterations: 8
```

5.1.11 The Logistic Regression Equation

The regression equation is:

$$P(\text{Default}) = \hat{\pi}_i = \frac{e^{-10.65+0.0055 \times \text{balance}}}{1 + e^{-10.65+0.0055 \times \text{balance}}}$$

- For a \$1,000 balance, the estimated default probability is $\frac{e^{-10.65+0.0055(1000)}}{1+e^{-10.65+0.0055(1000)}} \approx 0.006$
- For a \$1,500 balance, the estimated default probability is $\frac{e^{-10.65+0.0055(1500)}}{1+e^{-10.65+0.0055(1500)}} \approx 0.08$
- For a \$2,000 balance, the estimated default probability is $\frac{e^{-10.65+0.0055(2000)}}{1+e^{-10.65+0.0055(2000)}} \approx 0.59$

5.1.12 Predict in R

```
predict(CCDefault_M, newdata=data.frame((balance=1000)), type="response")

##           1
## 0.005752145

predict(CCDefault_M, newdata=data.frame((balance=1500)), type="response")

##           1
## 0.08294762

predict(CCDefault_M, newdata=data.frame((balance=2000)), type="response")

##           1
## 0.5857694
```

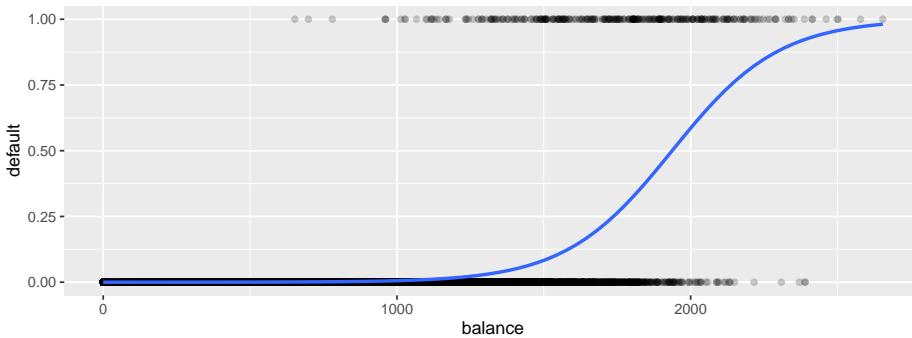
5.1.13 Where to the b's come from?

- Recall that for a quantitative response variable, the values of b_1, b_2, \dots, b_p are chosen in a way that minimizes $\sum_{i=1}^n (y_i - (\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}))^2$.
- Least squares does not work well in this generalized setting. Instead, the b's are calculated using a more advanced technique, known as **maximum likelihood estimation**.

5.2 Interpretations in a Logistic Regression Model

5.2.1 Recall Logistic Regression Curve for Credit Card Data

```
ggplot(data=Default, aes(y=default, x= balance)) + geom_point(alpha=0.2) +
  stat_smooth(method="glm", se=FALSE, method.args = list(family=binomial))
```



5.2.2 Recall Credit Card Model Output

```
M <- glm(data=Default, default ~ balance, family = binomial(link = "logit"))
summary(M)

##
## Call:
## glm(formula = default ~ balance, family = binomial(link = "logit"),
##      data = Default)
##
## Deviance Residuals:
##    Min      1Q   Median      3Q     Max
## -2.2697 -0.1465 -0.0589 -0.0221  3.7589
##
## Coefficients:
##             Estimate Std. Error z value     Pr(>|z|)
## (Intercept) -10.6513306  0.3611574 -29.49 <0.0000000000000002 ***
## balance      0.0054989  0.0002204   24.95 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1596.5  on 9998  degrees of freedom
## AIC: 1600.5
##
## Number of Fisher Scoring iterations: 8
```

5.2.3 Balance Logistic Model Equation

The regression equation is:

$$P(\text{Default}) = \hat{\pi}_i = \frac{e^{-10.65+0.0055 \times \text{balance}}}{1 + e^{-10.65+0.0055 \times \text{balance}}}$$

- For a \$1,000 balance, the estimated default probability is $\hat{\pi}_i = \frac{e^{10.65+0.0055(1000)}}{1+e^{10.65+0.0055(1000)}} \approx 0.005752145$
- For a \$1,500 balance, the estimated default probability is $\hat{\pi}_i = \frac{e^{10.65+0.0055(1500)}}{1+e^{10.65+0.0055(1500)}} \approx 0.08294762$
- For a \$2,000 balance, the estimated default probability is $\hat{\pi}_i = \frac{e^{10.65+0.0055(2000)}}{1+e^{10.65+0.0055(2000)}} \approx 0.5857694$

5.2.4 Odds

The odds of default are given by $\frac{\pi_i}{1-\pi_i}$.

Examples:

- The estimated odds of default for a \$1,000 balance are $\frac{0.005752145}{1-0.005752145} \approx 1 : 173$.
- The estimated odds of default for a \$1,500 balance are $\frac{0.08294762}{1-0.08294762} \approx 1 : 11$.
- The estimated odds of default for a \$2,000 balance are $\frac{0.5857694}{1-0.5857694} \approx 1.414 : 1$.

5.2.5 Odds Ratio

The quantity $\frac{\frac{\pi_i}{1-\pi_i}}{\frac{\pi_j}{1-\pi_j}}$ represents the odds ratio of a default for user i , compared to user j . This quantity is called the **odds ratio**.

Example:

The default odds ratio for a \$1,000 payment, compared to a \$2,000 payment is

The odds ratio is $\frac{\frac{1}{173}}{\frac{1}{1.414}} \approx 1 : 244$.

The odds of a default are about 244 times larger for a \$2,000 payment than a \$1,000 payment.

5.2.6 Interpretation of β_1

Consider the odds ratio for a case j with explanatory variable $x + 1$, compared to case i with explanatory variable x .

That is $\log\left(\frac{\pi_j}{1-\pi_i}\right) = \beta_0 + \beta_1 x$, and $\log\left(\frac{\pi_j}{1-\pi_j}\right) = \beta_0 + \beta_1(x+1)$.

$$\log\left(\frac{\frac{\pi_j}{1-\pi_j}}{\frac{\pi_i}{1-\pi_i}}\right) = \log\left(\frac{\pi_j}{1-\pi_j}\right) - \log\left(\frac{\pi_i}{1-\pi_i}\right) = \beta_0 + \beta_1(x+1) - (\beta_0 + \beta_1(x)) = \beta_1.$$

Thus a 1-unit increase in x is associated with a multiplicative change in the log odds by a factor of β_1 .

A 1-unit increase in x is associated with a multiplicative change in the odds by a factor of e^{β_1} .

5.2.7 Interpretation in Credit Card Example

$$b_1 = 0.0055$$

The odds of default are estimated to multiply by $e^{0.0055} \approx 1.0055$ for each 1-dollar increase in balance on the credit card.

That is, the odds of default are estimated to increase by 0.55% for each additional dollar on the card balance.

An increase of d dollars in credit card balance is estimated to be associated with an increase odds of default by a factor of $e^{0.0055d}$.

The odds of default for a balance of \$2,000 are estimated to be $e^{0.0055 \times 1000} \approx 244$ times as great as the odds of default for a \$1,000 balance.

5.2.8 Hypothesis Tests in Logistic Regression

- The p-value on the “balance” line of the regression output is associated with the null hypothesis $\beta_1 = 0$, that is that there is no relationship between balance and the odds of defaulting on the payment.
- The fact that the p-value is so small tells us that there is strong evidence of a relationship between balance and odds of default.

5.2.9 Confidence Intervals for β_1

```
confint(M, level = 0.95)

##              2.5 %      97.5 %
## (Intercept) -11.383288936 -9.966565064
## balance      0.005078926  0.005943365
```

We are 95% confident that a 1 dollar increase in credit card balance is associate with an increased odds of default by a factor between $e^{0.00508} \approx 1.0051$ and $e^{0.00594} \approx 1.0060$.

This is a profile-likelihood interval, which you can read more about here.

5.3 Multiple Logistic Regression

5.3.1 Logistic Regression Models with Multiple Explanatory Variables

We can also perform logistic regression in situations where there are multiple explanatory variables.

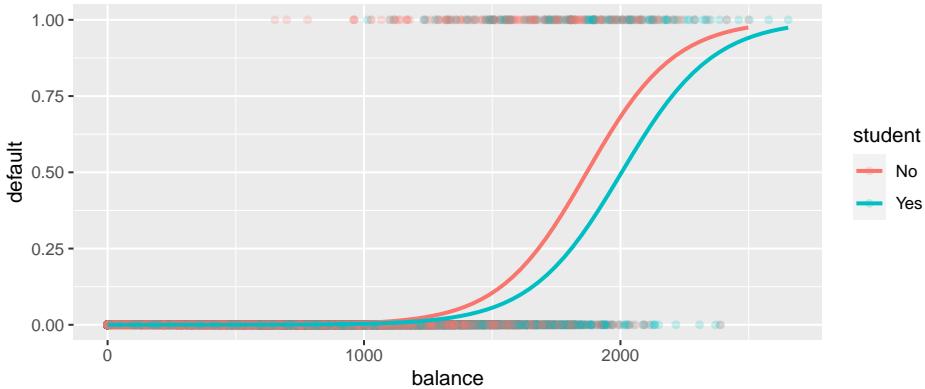
5.3.2 Logistic Model with Multiple Predictors

```
CCDefault_M2 <- glm(data=Default, default ~ balance + student, family = binomial(link = "logit"))
summary(CCDefault_M2)

##
## Call:
## glm(formula = default ~ balance + student, family = binomial(link = "logit"),
##      data = Default)
##
## Deviance Residuals:
##      Min        1Q     Median        3Q       Max
## -2.4578  -0.1422  -0.0559  -0.0203   3.7435
##
## Coefficients:
##             Estimate Std. Error z value    Pr(>|z|)
## (Intercept) -10.7494959  0.3691914 -29.116 < 0.0000000000000002 ***
## balance      0.0057381  0.0002318  24.750 < 0.0000000000000002 ***
## studentYes   -0.7148776  0.1475190  -4.846      0.00000126 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1571.7  on 9997  degrees of freedom
## AIC: 1577.7
##
## Number of Fisher Scoring iterations: 8
```

5.3.3 Multiple Logistic Model Illustration

```
ggplot(data=Default, aes(y=default, x= balance, color=student)) + geom_point(alpha=0.2) + stat_sm
```



5.3.4 Multiple Logistic Regression Interpretation

The regression equation is:

$$P(\text{Default}) = \hat{\pi}_i = \frac{e^{-10.75+0.00575 \times \text{balance} - 0.7149 \times I_{\text{student}}}}{1 + e^{-10.75+0.00575 \times \text{balance} - 0.7149 \times I_{\text{student}}}}$$

- The odds of default are estimated to multiply by $e^{0.005738} \approx 1.00575$ for each 1 dollar increase in balance whether the user is a student or nonstudent. Thus, the estimated odds of default increase by about 0.05%.
- We might be more interested in what would happen with a larger increase, say \$100. The odds of default are estimated to multiply by $e^{0.005738 \times 100} \approx 1.775$ for each \$100 increase in balance for students as well as nonstudents. Thus, the estimated odds of default increase by about 77.5%.

The odds of default for students are estimated to be $e^{-0.7149} \approx 0.49$ as high for students as non-students, assuming balance amount is held constant.

5.3.5 Hypothesis Tests in Multiple Logistic Regression Model

- There is strong evidence of a relationship between balance and odds of default, provided we are comparing students to students, or nonstudents to nonstudents.
- There is evidence that students are less likely to default than nonstudents, provided the balance on the card is the same.

5.3.6 Multiple Logistic Regression Model with Interaction

```
CCDefault_M_Int <- glm(data=Default, default ~ balance * student, family = binomial(link = "logit"))
summary(CCDefault_M_Int)

##
## Call:
## glm(formula = default ~ balance * student, family = binomial(link = "logit"),
##      data = Default)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.4839  -0.1415  -0.0553  -0.0202   3.7628
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -10.8746818  0.4639679 -23.438 <0.0000000000000002 ***
## balance                  0.0058188  0.0002937  19.812 <0.0000000000000002 ***
## studentYes            -0.3512310  0.8037333  -0.437     0.662
## balance:studentYes   -0.0002196  0.0004781  -0.459     0.646
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1571.5  on 9996  degrees of freedom
## AIC: 1579.5
##
## Number of Fisher Scoring iterations: 8
```

5.3.7 Interpretations for Logistic Model with Interaction

- The regression equation is:

$$P(\text{Default}) = \hat{\pi}_i = \frac{e^{-10.87+0.0058 \times \text{balance} - 0.35 \times \text{I}_{\text{student}} - 0.0002 \times \text{balance} \times \text{I}_{\text{student}}}}{1 + e^{-10.87+0.0058 \times \text{balance} - 0.35 \times \text{I}_{\text{student}} - 0.0002 \times \text{balance} \times \text{I}_{\text{student}}}}$$

- Since estimate of the interaction effect is so small and the p-value on this estimate is large, it is plausible that there is no interaction at all. Thus, the simpler non-interaction model is preferable.

5.3.8 Logistic Regression Key Points

- Y is a binary response variable.
- π_i is a function of explanatory variables x_{i1}, \dots, x_{ip} .
- $E(Y_i) = \pi_i = \frac{e^{\beta_0 + \beta_1 x_i + \dots + \beta_p x_{ip}}}{1 + e^{\beta_0 + \beta_1 x_i + \dots + \beta_p x_{ip}}}$
- $\beta_0 + \beta_1 x_i + \dots + \beta_p x_{ip} = \log\left(\frac{\pi_i}{1 - \pi_i}\right)$
- For quantitative x_j , when all other explanatory variables are held constant, the odds of “success” multiply by a factor of e_j^β for each 1 unit increase in x_j
- For categorical x_j , when all other explanatory variables are held constant, the odds of “success” are e_j^β times higher for category j than for the “baseline category.”
- For models with interaction, we can only interpret β_j when the values of all other explanatory variables are given (since the effect of x_j depends on the other variables.)

Chapter 6

Building Models for Interpretation

6.1 Introduction to Model Building

6.1.1 Overview of Model Building

- So far, we've dealt with models with 2 or fewer variables. Often, we want to use more complex models.
- We'll need to decide how many variables to include in the model. This is not an obvious decision, and will be different, depending on the purpose of the model.
- We'll also need to make other decisions, such as whether or not to use interaction terms, or transformations.

6.1.2 Interpretation vs Prediction

While there are many interests in model building, we can often characterize these into two major types:

Building Models for Interpretation

- Intent is to help us understand relationships between variables. For example “what is expected to happen to Y when X increases by this much?”

- While these models can be used to make predictions, this is not the primary or sole intent.
- Aim for simple model that gives a realistic approximation of real situation.
 - limited number of explanatory variables (usually ≤ 10)
 - include interactions and use transformations only when necessary
 - avoid including highly-correlated explanatory variables
 - account for lurking variables (think Simpson's Paradox)

6.1.3 Interpretation vs Prediction (cont.)

While there are many interests in model building, we can often characterize these into two major types:

Building Models for Prediction

- Goal is to make the most accurate predictions possible.
- Not concerned with understanding relationships between variables. Not worried model being too complicated to interpret, as long as it yields good predictions.
- Aim for a model that best captures the signal in the data, without being thrown off by noise.
 - Large number of predictors is ok
 - Don't make model so complicated that it overfits the data.

6.1.4 Model for Price of 2015 Cars

What factors contribute to the price of a car?

Build a model for the price of a new 2015 car, in order to help us answer this question.

6.1.5 Glimpse of Cars Dataset

```
data(Cars2015)
glimpse(Cars2015)
```

```
## Rows: 110
```

```
## Columns: 20
## $ Make      <fct> Chevrolet, Hyundai, Kia, Mitsubishi, Nissan, Dodge, Chevrole-
## $ Model     <fct> Spark, Accent, Rio, Mirage, Versa Note, Dart, Cruze LS, 500L-
## $ Type      <fct> Hatchback, Hatchback, Sedan, Hatchback, Hatchback, Sedan, Se-
## $ LowPrice   <dbl> 12.270, 14.745, 13.990, 12.995, 14.180, 16.495, 16.170, 19.3-
## $ HighPrice  <dbl> 25.560, 17.495, 18.290, 15.395, 17.960, 23.795, 25.660, 24.6-
## $ Drive     <fct> FWD, AWD, ~
## $ CityMPG   <int> 30, 28, 28, 37, 31, 23, 24, 24, 28, 30, 27, 27, 25, 27, 30, ~
## $ HwyMPG    <int> 39, 37, 36, 44, 40, 35, 36, 33, 38, 35, 33, 36, 36, 37, 39, ~
## $ FuelCap   <dbl> 9.0, 11.4, 11.3, 9.2, 10.9, 14.2, 15.6, 13.1, 12.4, 11.1, 11-
## $ Length    <int> 145, 172, 172, 149, 164, 184, 181, 167, 179, 154, 156, 180, ~
## $ Width     <int> 63, 67, 68, 66, 67, 72, 71, 70, 72, 67, 68, 69, 70, 68, 69, ~
## $ Wheelbase <int> 94, 101, 101, 97, 102, 106, 106, 103, 104, 99, 98, 104, 104, ~
## $ Height    <int> 61, 57, 57, 59, 61, 58, 58, 66, 58, 59, 58, 58, 57, 58, 59, ~
## $ UTurn     <int> 34, 37, 37, 32, 37, 38, 38, 37, 39, 34, 35, 38, 37, 36, 37, ~
## $ Weight    <int> 2345, 2550, 2575, 2085, 2470, 3260, 3140, 3330, 2990, 2385, ~
## $ Acc030    <dbl> 4.4, 3.7, 3.5, 4.4, 4.0, 3.4, 3.7, 3.9, 3.4, 3.9, 3.9, 3.7, ~
## $ Acc060    <dbl> 12.8, 10.3, 9.5, 12.1, 10.9, 9.3, 9.8, 9.5, 9.2, 10.8, 11.1, ~
## $ QtrMile   <dbl> 19.4, 17.8, 17.3, 19.0, 18.2, 17.2, 17.6, 17.4, 17.1, 18.3, ~
## $ PageNum   <int> 123, 148, 163, 188, 196, 128, 119, 131, 136, 216, 179, 205, ~
## $ Size      <fct> Small, Small, Small, Small, Small, Small, Small, Smal-
```

6.1.6 Categorical Variables

```
Cars_Cat <- select_if(Cars2015, is.factor)
summary(Cars_Cat)
```

	Make	Model	Type	Drive	Size
## Chevrolet	8	CTS : 2	7Pass : 15	AWD:25	Large : 29
## Ford	7	2 Touring : 1	Hatchback:11	FWD:63	Midsized:34
## Hyundai	7	200 : 1	Sedan : 46	RWD:22	Small : 47
## Toyota	7	3 i Touring: 1	Sporty : 11		
## Audi	6	3 Series GT: 1	SUV : 18		
## Nissan	6	300 : 1	Wagon : 9		
## (Other)	69	(Other) : 103			

6.1.7 Correlation Matrix for Quantitative Variables

```
Cars_Num <- select_if(Cars2015, is.numeric)
C <- cor(Cars_Num, use = "pairwise.complete.obs")
round(C, 2)
```

```
##          LowPrice HighPrice CityMPG HwyMPG FuelCap Length Width Wheelbase
```

```

## LowPrice    1.00    0.91   -0.65  -0.59    0.57    0.47   0.48    0.46
## HighPrice   0.91    1.00   -0.56  -0.49    0.47    0.39   0.37    0.39
## CityMPG     -0.65   -0.56    1.00   0.93   -0.77  -0.72  -0.78   -0.69
## HwyMPG      -0.59   -0.49    0.93   1.00   -0.75  -0.64  -0.75   -0.64
## FuelCap     0.57    0.47   -0.77  -0.75    1.00    0.82   0.85    0.79
## Length      0.47    0.39   -0.72  -0.64    0.82    1.00   0.81    0.92
## Width       0.48    0.37   -0.78  -0.75    0.85    0.81   1.00    0.76
## Wheelbase   0.46    0.39   -0.69  -0.64    0.79    0.92   0.76    1.00
## Height      0.02   -0.10   -0.39  -0.54    0.58    0.46   0.62    0.49
## UTurn       0.40    0.31   -0.73  -0.68    0.76    0.84   0.77    0.81
## Weight      0.55    0.43   -0.83  -0.84    0.91    0.82   0.91    0.81
## Acc030     -0.76   -0.74    0.64   0.51   -0.47  -0.38  -0.41   -0.31
## Acc060     -0.74   -0.72    0.68   0.52   -0.49  -0.47  -0.46   -0.38
## QtrMile    -0.76   -0.76    0.65   0.49   -0.45  -0.42  -0.41   -0.35
## PageNum    -0.23   -0.20    0.28   0.15   -0.15  -0.23  -0.20   -0.24
##             Height UTurn Weight Acc030 Acc060 QtrMile PageNum
## LowPrice    0.02  0.40   0.55  -0.76  -0.74  -0.76  -0.23
## HighPrice   -0.10 0.31   0.43  -0.74  -0.72  -0.76  -0.20
## CityMPG     -0.39 -0.73  -0.83   0.64   0.68   0.65   0.28
## HwyMPG      -0.54 -0.68  -0.84   0.51   0.52   0.49   0.15
## FuelCap     0.58  0.76   0.91  -0.47  -0.49  -0.45  -0.15
## Length      0.46  0.84   0.82  -0.38  -0.47  -0.42  -0.23
## Width       0.62  0.77   0.91  -0.41  -0.46  -0.41  -0.20
## Wheelbase   0.49  0.81   0.81  -0.31  -0.38  -0.35  -0.24
## Height      1.00  0.55   0.71   0.21   0.21   0.25   0.06
## UTurn       0.55  1.00   0.80  -0.36  -0.41  -0.37  -0.22
## Weight      0.71  0.80   1.00  -0.41  -0.43  -0.39  -0.20
## Acc030     0.21 -0.36  -0.41   1.00   0.95   0.95   0.25
## Acc060     0.21 -0.41  -0.43   0.95   1.00   0.99   0.26
## QtrMile    0.25 -0.37  -0.39   0.95   0.99   1.00   0.26
## PageNum    0.06 -0.22  -0.20   0.25   0.26   0.26   1.00

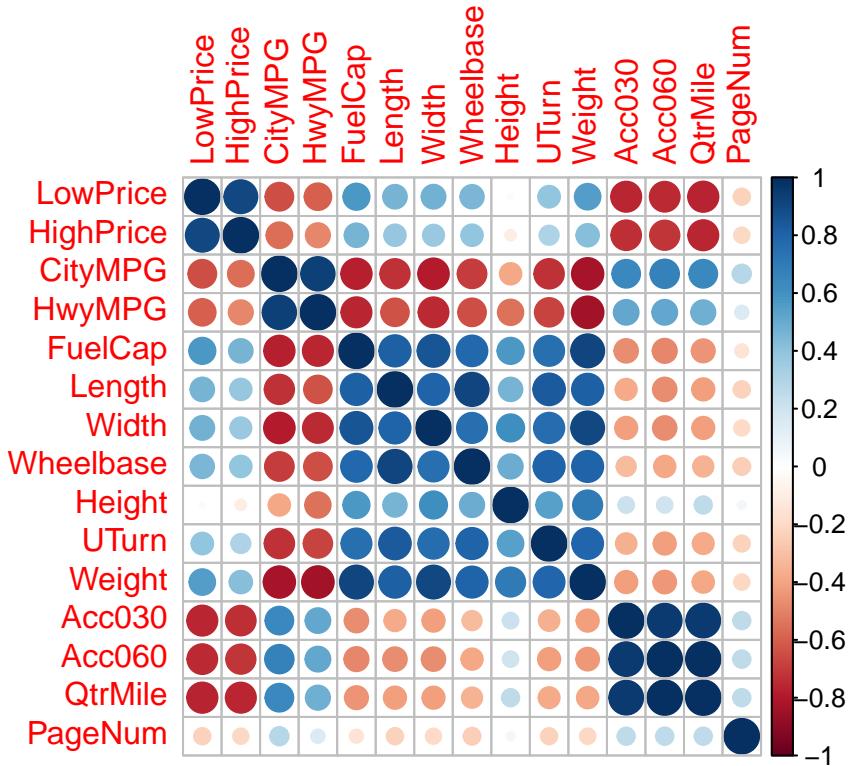
```

6.1.8 Correlation Plot for Quantitative Variables

```

library(corrplot)
C <- corrplot(C)

```



6.1.9 Multicollinearity in Modeling

What happens when we include two highly-correlated explanatory variables in the same model?

```
cor(Cars2015$Acc060, Cars2015$QtrMile)

## [1] 0.9916625
```

6.1.10 Model Using Acceleration Time

```
Cars_M1 <- lm(data=Cars2015, log(LowPrice) ~ Acc060)
summary(Cars_M1)

##
## Call:
## lm(formula = log(LowPrice) ~ Acc060, data = Cars2015)
##
## Residuals:
```

```

##      Min       1Q   Median       3Q      Max
## -0.84587 -0.19396  0.00908  0.18615  0.53350
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)
## (Intercept) 5.13682   0.13021  39.45 <0.0000000000000002 ***
## Acc060     -0.22064   0.01607 -13.73 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.276 on 108 degrees of freedom
## Multiple R-squared:  0.6359, Adjusted R-squared:  0.6325
## F-statistic: 188.6 on 1 and 108 DF,  p-value: < 0.0000000000000022

```

6.1.11 Confidence Interval for Acc. Slope

```

exp(confint(Cars_M1))

##                  2.5 %    97.5 %
## (Intercept) 131.4610408 220.284693
## Acc060      0.7768669  0.827959

```

We are 95% confident that a 1-second increase in acceleration time is associated with an average price decrease between 17% and 22.5%.

6.1.12 Model Using Quarter Mile Time

```

Cars_M2 <- lm(data=Cars2015, log(LowPrice) ~ QtrMile)
summary(Cars_M2)

##
## Call:
## lm(formula = log(LowPrice) ~ QtrMile, data = Cars2015)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.91465 -0.19501  0.02039  0.17538  0.60073
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)
## (Intercept) 7.8559     0.3248  24.19 <0.0000000000000002 ***
## QtrMile     -0.2776     0.0201 -13.81 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

```
##
## Residual standard error: 0.275 on 108 degrees of freedom
## Multiple R-squared:  0.6385, Adjusted R-squared:  0.6351
## F-statistic: 190.7 on 1 and 108 DF,  p-value: < 0.00000000000000022
```

6.1.13 Confidence Interval for Quarter Mile Time

```
exp(confint(Cars_M2))

##           2.5 %      97.5 %
## (Intercept) 1355.8297704 4913.077313
## QtrMile     0.7279941   0.788385
```

We are 95% confident that a 1-second increase in quarter mile time is associated with a price decrease between 21% and 27%, on average.

6.1.14 Model Using Quarter Mile Time and Acc. Time

```
Cars_M3 <- lm(data=Cars2015, log(LowPrice) ~ QtrMile + Acc060)
summary(Cars_M3)

##
## Call:
## lm(formula = log(LowPrice) ~ QtrMile + Acc060, data = Cars2015)
##
## Residuals:
##       Min     1Q   Median     3Q    Max
## -0.89124 -0.20030  0.01001  0.17576  0.57462
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 6.83974   1.54354   4.431 0.0000227 ***
## QtrMile     -0.17316   0.15640  -1.107   0.271    
## Acc060      -0.08389   0.12455  -0.673   0.502    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2757 on 107 degrees of freedom
## Multiple R-squared:  0.64, Adjusted R-squared:  0.6332
## F-statistic: 95.1 on 2 and 107 DF,  p-value: < 0.00000000000000022
```

6.1.15 Intervals for Model with High Correlation

```
exp(confint(Cars_M3))

##               2.5 %      97.5 %
## (Intercept) 43.8095999 19922.799158
## QtrMile     0.6168071   1.146686
## Acc060      0.7183525   1.177065
```

It does not make sense to talk about holding QtrMile constant as Acc060 increases, or vice-versa. Trying to do so leads to nonsensical answers.

We are 95% confident that a 1-second increase in quarter mile time is associated with an average price change between a 38% decrease and 15% increase, assuming acceleration time is held constant.

We are 95% confident that a 1-second increase in acceleration time is associated with an average price change between a 28% decrease and 18% increase, assuming quarter mile time is held constant.

6.1.16 Problems with Multicollinearity in Modeling

Because these variables are so highly correlated, it the model cannot separate the effect of one from the other, and thus is uncertain about both. Notice the very large standard errors associated with both regression coefficients, which lead to very wide confidence intervals.

In fact, if two variables are perfectly correlated, it will be impossible to fit them both in a model, and you will get an error message.

6.1.17 Impact on Prediction

Suppose we want to predict the price of a car that can accelerate from 0 to 60 mph in 9.5 seconds, and completes a quarter mile in 17.3 seconds.

```
exp(predict(Cars_M1, newdata = data.frame(Acc060=9.5, QtrMile=17.3)))

##           1
## 20.92084

exp(predict(Cars_M2, newdata = data.frame(Acc060=9.5, QtrMile=17.3)))

##           1
## 21.18223

exp(predict(Cars_M3, newdata = data.frame(Acc060=9.5, QtrMile=17.3)))
```

```
##      1
## 21.05489
```

The predicted values are similar. Multicollinearity does not hurt predictions, only interpretations.

6.1.18 Recall Correlation Matrix for Quantitative Variables

Let's use quartermile time as an explanatory variable. Should we add others?

Correlations with Quarter mile Time

```
Cars_Num <- select_if(Cars2015, is.numeric)
C <- cor(Cars_Num)
round(C[,14],2)

##  LowPrice HighPrice   CityMPG   HwyMPG FuelCap Length   Width Wheelbase
##    -0.76     -0.76     0.65     0.49   -0.45   -0.42   -0.41   -0.35
##  Height     UTurn    Weight Acc030  Acc060  QtrMile  PageNum
##    0.25     -0.37    -0.39     0.95     0.99     1.00     0.26
```

6.1.19 Adding Weight to Model

```
Cars_M4 <- lm(data=Cars2015, log(LowPrice) ~ QtrMile + Weight)
summary(Cars_M4)

##
## Call:
## lm(formula = log(LowPrice) ~ QtrMile + Weight, data = Cars2015)
##
## Residuals:
##       Min     1Q   Median     3Q    Max 
## -0.79365 -0.13931 -0.01368  0.15773  0.42234 
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)    
## (Intercept) 6.21326823  0.33491778 18.552 < 0.0000000000000002 ***
## QtrMile     -0.22482146  0.01748563 -12.858 < 0.0000000000000002 ***
## Weight      0.00020606  0.00002641   7.803   0.0000000000043 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2206 on 107 degrees of freedom
## Multiple R-squared:  0.7696, Adjusted R-squared:  0.7653
```

```
## F-statistic: 178.7 on 2 and 107 DF, p-value: < 0.00000000000000022
R2 went up from 0.64 to 0.76!
```

6.1.20 Do we need an Interaction Term?

```
Cars_M5 <- lm(data=Cars2015, log(LowPrice) ~ QtrMile * Weight)
summary(Cars_M5)

##
## Call:
## lm(formula = log(LowPrice) ~ QtrMile * Weight, data = Cars2015)
##
## Residuals:
##     Min      1Q  Median      3Q      Max
## -0.82013 -0.12076 -0.01464  0.14717  0.41928
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)        4.1114189  1.3270870   3.098  0.00249 ***
## QtrMile           -0.0963226  0.0804413  -1.197  0.23381
## Weight            0.0008110  0.0003707   2.188  0.03089 *
## QtrMile:Weight  -0.0000373  0.0000228  -1.636  0.10482
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2188 on 106 degrees of freedom
## Multiple R-squared:  0.7752, Adjusted R-squared:  0.7689
## F-statistic: 121.9 on 3 and 106 DF, p-value: < 0.00000000000000022
```

p-value on interaction is not that small. R^2 didn't go up much. Let's not use it.

6.1.21 Add HWY MPG?

```
Cars_M6 <- lm(data=Cars2015, log(LowPrice) ~ QtrMile + Weight + HwyMPG)
summary(Cars_M6)

##
## Call:
## lm(formula = log(LowPrice) ~ QtrMile + Weight + HwyMPG, data = Cars2015)
##
## Residuals:
##     Min      1Q  Median      3Q      Max
## -0.82013 -0.12076 -0.01464  0.14717  0.41928
```

```

## -0.82308 -0.14513 -0.01922  0.16732  0.41390
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)
## (Intercept) 6.54954436  0.42196132 15.522 < 0.0000000000000002 ***
## QtrMile     -0.21699008  0.01843615 -11.770 < 0.0000000000000002 ***
## Weight      0.00015922  0.00004456   3.573      0.000532 ***
## HwyMPG     -0.00961141  0.00737658  -1.303      0.195410
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2198 on 106 degrees of freedom
## Multiple R-squared:  0.7732, Adjusted R-squared:  0.7668
## F-statistic: 120.5 on 3 and 106 DF,  p-value: < 0.00000000000000022

```

HwyMPG doesn't make change R^2 much, and has a high correlation with weight. Let's not include it.

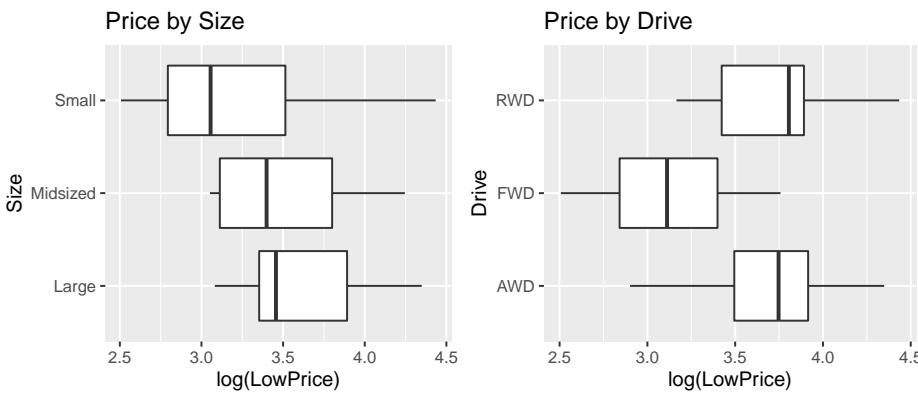
6.1.22 Categorical Variables to Consider

Relationship between Price, Size, and Drive

```

P1 <- ggplot(data=Cars2015, aes(x=log(LowPrice), y=Size)) + geom_boxplot() + ggtitle("Price by Size")
P2 <- ggplot(data=Cars2015, aes(x=log(LowPrice), y=Drive)) + geom_boxplot() + ggtitle("Price by Drive")
grid.arrange(P1, P2, ncol=2)

```



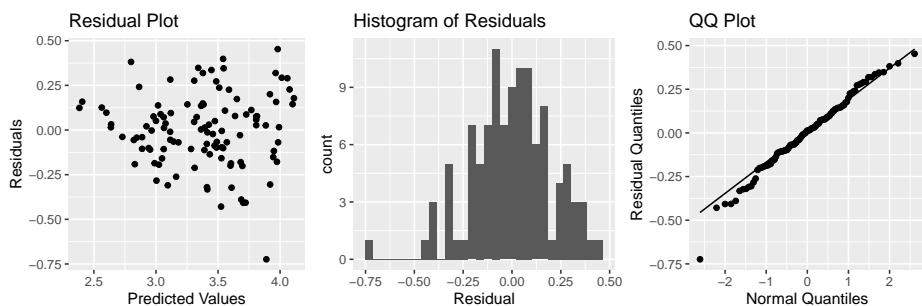
Information about size is already included, through the weight variable. Let's add drive type to the model.

6.1.23 Model with QtrMile, Weight, and Drive

```
Cars_M7 <- lm(data=Cars2015, log(LowPrice) ~ QtrMile + Weight + Drive)
summary(Cars_M7)
```

```
##
## Call:
## lm(formula = log(LowPrice) ~ QtrMile + Weight + Drive, data = Cars2015)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -0.72386 -0.10882  0.01269  0.13306  0.45304
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)    
## (Intercept) 5.81406789  0.33961789 17.119 < 0.0000000000000002 *** 
## QtrMile     -0.19007439  0.01959554 -9.700 0.0000000000000289 *** 
## Weight      0.00020496  0.00002583  7.936 0.000000000002420675 ***
## DriveFWD   -0.22403222  0.05704513 -3.927 0.000154 ***    
## DriveRWD   -0.13884399  0.06227709 -2.229 0.027913 *      
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2077 on 105 degrees of freedom
## Multiple R-squared:  0.7995, Adjusted R-squared:  0.7919 
## F-statistic: 104.7 on 4 and 105 DF,  p-value: < 0.0000000000000022
```

6.1.24 Check of Model Assumptions



There is slight concern about constant variance, but otherwise, the model assumptions look good.

6.1.25 Coefficients and Exponentiation

```
Cars_M7$coefficients

## (Intercept)      QtrMile       Weight     DriveFWD     DriveRWD
## 5.8140678915 -0.1900743859  0.0002049586 -0.2240322171 -0.1388439916

exp(Cars_M7$coefficients)

## (Intercept)      QtrMile       Weight     DriveFWD     DriveRWD
## 334.9790161    0.8268976   1.0002050   0.7992894   0.8703638
```

6.1.26 Interpretation of Coefficients

```
exp(Cars_M7$coefficients)

## (Intercept)      QtrMile       Weight     DriveFWD     DriveRWD
## 334.9790161    0.8268976   1.0002050   0.7992894   0.8703638
```

The price of a car is expected to decrease by 17% for each additional second it takes to drive a quartermile, assuming weight, and drive type are held constant.

The price of a car is expected to increase by 0.02% for each additional pound, assuming quarter mile time, and drive type are held constant. Thus, a 100 lb increase is associated with an expected 2% increase in price, assuming quarter mile time, and drive type are held constant.

FWD cars are expected to cost 20% less than AWD cars, assuming quarter mile time and weight are held constant.

RWD cars are expected to cost 13% less than AWD cars, assuming quarter mile time and weight are held constant.

6.2 Transformations of Explanatory Variables

6.2.1 Residual vs Explanatory Variable Plots

We've previously plotted residuals against predicted values, in order to assess the linearity and constant variance assumptions.

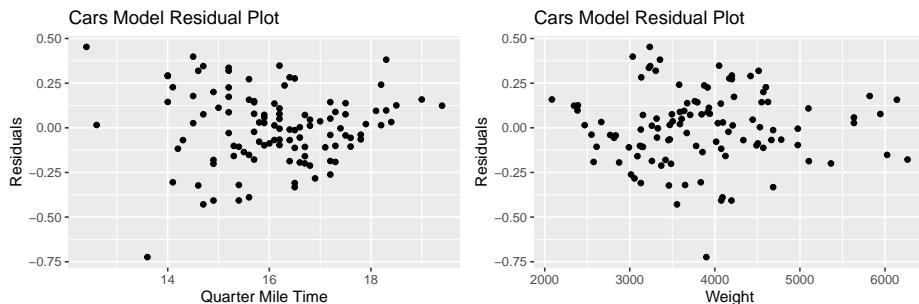
It can also be useful to plot residuals against the quantitative explanatory variables in a model. If we notice curvature in these plots, it tells us that we should consider using a nonlinear function of the explanatory variable in our model.

6.2.2 Cars Model

```
Cars_M7 <- lm(data=Cars2015, log(LowPrice) ~ QtrMile + Weight + Drive)
summary(Cars_M7)

##
## Call:
## lm(formula = log(LowPrice) ~ QtrMile + Weight + Drive, data = Cars2015)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -0.72386 -0.10882  0.01269  0.13306  0.45304
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)    
## (Intercept) 5.81406789  0.33961789 17.119 < 0.0000000000000002 *** 
## QtrMile     -0.19007439  0.01959554 -9.700 0.0000000000000289 *** 
## Weight      0.00020496  0.00002583  7.936 0.000000000002420675 *** 
## DriveFWD   -0.22403222  0.05704513 -3.927 0.000154 ***    
## DriveRWD   -0.13884399  0.06227709 -2.229 0.027913 *      
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
##
## Residual standard error: 0.2077 on 105 degrees of freedom
## Multiple R-squared:  0.7995, Adjusted R-squared:  0.7919 
## F-statistic: 104.7 on 4 and 105 DF,  p-value: < 0.0000000000000022
```

6.2.3 Residual vs Explanatory Variable Plots for Cars Model



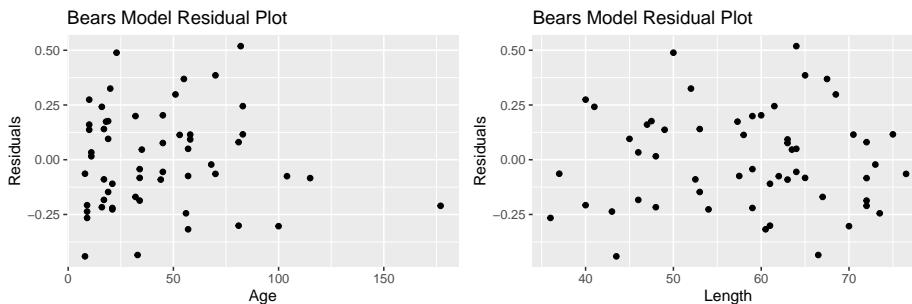
There is no reason to be concerned about these plots.

6.2.4 Model for Weights of Bears

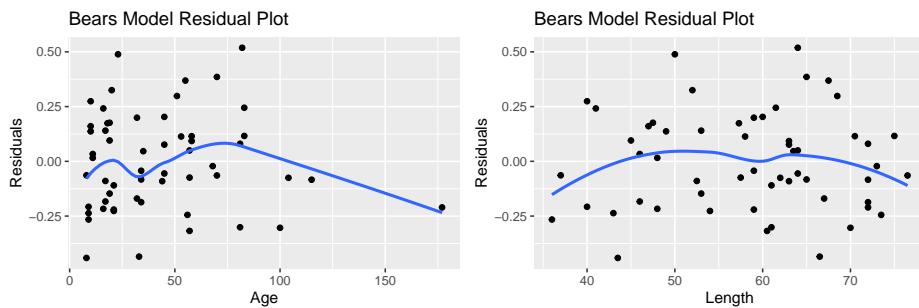
```
Bears_M1_MLR <- lm(data=Bears_Subset, log(Weight)~Age + Sex + Length + Season)
summary(Bears_M1_MLR)
```

```
##
## Call:
## lm(formula = log(Weight) ~ Age + Sex + Length + Season, data = Bears_Subset)
##
## Residuals:
##      Min        1Q    Median        3Q       Max
## -0.44119 -0.18423 -0.03242  0.14523  0.51870
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)
## (Intercept) 1.613971  0.232306  6.948 0.00000000725 ***
## Age          0.003517  0.001459  2.412 0.0196 *
## Sex2         -0.147521  0.070228 -2.101 0.0407 *
## Length        0.056271  0.004508 12.483 < 0.0000000000000002 ***
## SeasonSpring -0.060092  0.093670 -0.642 0.5241
## SeasonSummer -0.133893  0.072951 -1.835 0.0724 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.234 on 50 degrees of freedom
##   (41 observations deleted due to missingness)
## Multiple R-squared:  0.91, Adjusted R-squared:  0.901
## F-statistic: 101.1 on 5 and 50 DF,  p-value: < 0.0000000000000022
```

6.2.5 Bears Model: Residuals by Explanatory Variable Plots



6.2.6 Bears Model: Residuals by Explanatory Variable Plots



The fact that a few ages are much greater than the rest can cause these to have undue influence on the model. We also see some sign of curvature in the residual vs length plot.

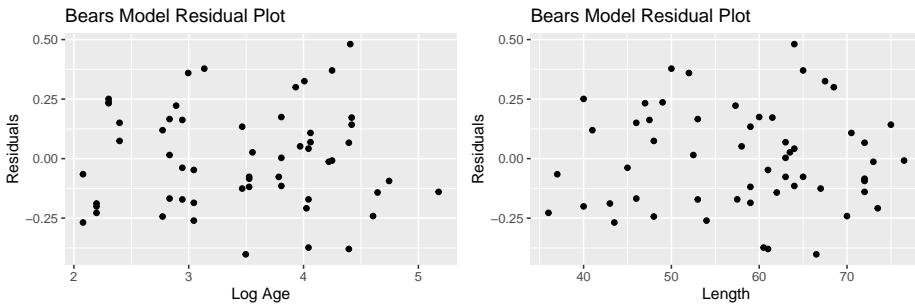
6.2.7 Model for Weights of Bears

```
Bears_M2_MLR <- lm(data=Bears_Subset, log(Weight) ~ log(Age) + Sex + Length + Season)
summary(Bears_M2_MLR)

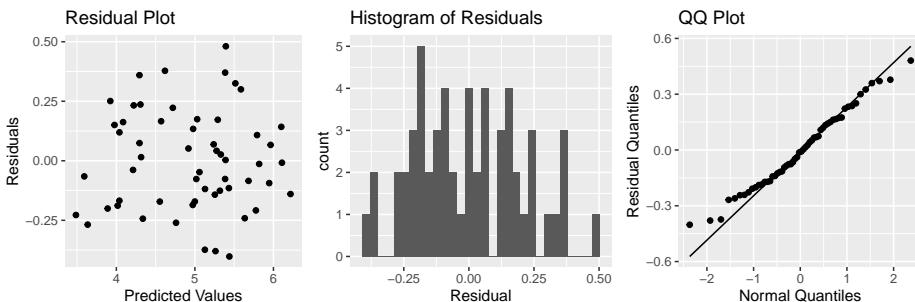
##
## Call:
## lm(formula = log(Weight) ~ log(Age) + Sex + Length + Season,
##     data = Bears_Subset)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.40196 -0.16848 -0.01066  0.15341  0.48085
##
## Coefficients:
##             Estimate Std. Error t value    Pr(>|t|)
## (Intercept) 1.474059  0.180134  8.183 0.000000000868 ***
## log(Age)     0.323951  0.087084  3.720  0.000505 ***
## Sex2        -0.233222  0.072787 -3.204  0.002360 **
## Length       0.042589  0.006499  6.553 0.0000000300327 ***
## SeasonSpring -0.078946  0.087634 -0.901  0.371982
## SeasonSummer -0.131581  0.068210 -1.929  0.059409 .
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2188 on 50 degrees of freedom
## (41 observations deleted due to missingness)
```

```
## Multiple R-squared:  0.9213, Adjusted R-squared:  0.9134
## F-statistic: 117.1 on 5 and 50 DF,  p-value: < 0.00000000000000022
```

6.2.8 Bears Model: Residuals by Explanatory Variable Plots



6.2.9 Bears Model: Residual Plots



6.3 Modeling SAT scores: Simpson's Paradox and Quadratic Terms

6.3.1 SAT Scores Dataset

We'll now look at a dataset containing education data on all 50 states. Among the variables are average SAT score, average teacher salary, and fraction of students who took the SAT.

```
glimpse(SAT)
```

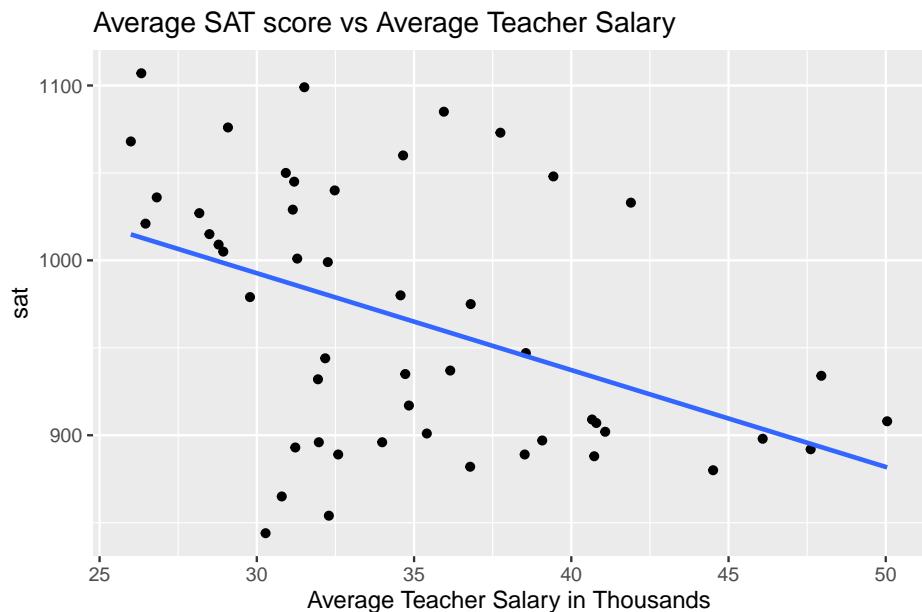
```
## Rows: 50
## Columns: 8
```

```
## $ state  <fct> Alabama, Alaska, Arizona, Arkansas, California, Colorado, Conne-
## $ expend <dbl> 4.405, 8.963, 4.778, 4.459, 4.992, 5.443, 8.817, 7.030, 5.718, ~
## $ ratio  <dbl> 17.2, 17.6, 19.3, 17.1, 24.0, 18.4, 14.4, 16.6, 19.1, 16.3, 17.~
## $ salary  <dbl> 31.144, 47.951, 32.175, 28.934, 41.078, 34.571, 50.045, 39.076, ~
## $ frac   <int> 8, 47, 27, 6, 45, 29, 81, 68, 48, 65, 57, 15, 13, 58, 5, 9, 11, ~
## $ verbal <int> 491, 445, 448, 482, 417, 462, 431, 429, 420, 406, 407, 468, 488~
## $ math   <int> 538, 489, 496, 523, 485, 518, 477, 468, 469, 448, 482, 511, 560~
## $ sat    <int> 1029, 934, 944, 1005, 902, 980, 908, 897, 889, 854, 889, 979, 1~
```

6.3.2 Teacher Salaries and SAT Scores

The plot displays average SAT score against average teacher salary for all 50 US states.

```
ggplot(data=SAT, aes(y=sat, x=salary)) + geom_point() +
  stat_smooth(method="lm", se=FALSE) +
  ggtitle("Average SAT score vs Average Teacher Salary") +
  xlab("Average Teacher Salary in Thousands")
```



What conclusion do you draw from the plot?

Are these results surprising?

6.3.3 Simple Linear Regression Model

```
SAT_M1 <- lm(data=SAT, sat~salary)
summary(SAT_M1)

##
## Call:
## lm(formula = sat ~ salary, data = SAT)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -147.125 -45.354   4.073  42.193 125.279 
## 
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)    
## (Intercept) 1158.859    57.659  20.098 < 0.000000000000002 ***
## salary       -5.540     1.632   -3.394     0.00139 **  
## ---      
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 67.89 on 48 degrees of freedom
## Multiple R-squared:  0.1935, Adjusted R-squared:  0.1767 
## F-statistic: 11.52 on 1 and 48 DF,  p-value: 0.001391
```

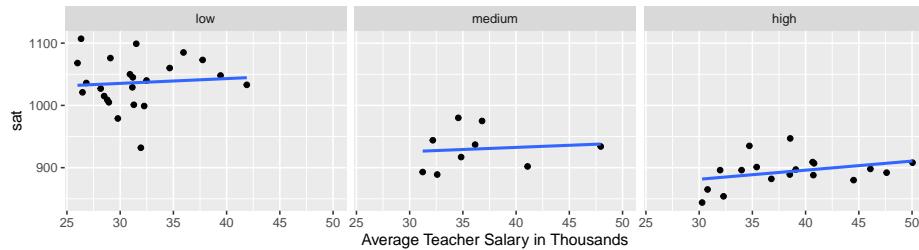
6.3.4 A Closer Look

Let's break the data down by the percentage of students who take the SAT.

Low = 0%-22%
 Medium = 22-49%
 High = 49-81%

```
SAT <- mutate(SAT, fracgrp = cut(frac,
  breaks=c(0, 22, 49, 81),
  labels=c("low", "medium", "high")))
```

6.3.5 A Closer Look



Now what conclusions do you draw from the plots?

6.3.6 Multiple Regression Model

```
SAT_M2 <- lm(data=SAT, sat~salary+frac)
summary(SAT_M2)

##
## Call:
## lm(formula = sat ~ salary + frac, data = SAT)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -78.313 -26.731   3.168  18.951  75.590 
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)    
## (Intercept) 987.9005    31.8775  30.991 < 0.0000000000000002 ***
## salary       2.1804     1.0291   2.119     0.0394 *  
## frac        -2.7787     0.2285 -12.163   0.0000000000000004 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
##
## Residual standard error: 33.69 on 47 degrees of freedom
## Multiple R-squared:  0.8056, Adjusted R-squared:  0.7973 
## F-statistic: 97.36 on 2 and 47 DF,  p-value: < 0.0000000000000022
```

6.3.7 Add Other Variables?

```
SAT_Num <- select_if(SAT, is.numeric)
C <- cor(SAT_Num, use = "pairwise.complete.obs")
round(C, 2)
```

```

##      expend ratio salary  frac verbal  math   sat
## expend    1.00 -0.37    0.87  0.59 -0.41 -0.35 -0.38
## ratio     -0.37  1.00    0.00 -0.21   0.06  0.10  0.08
## salary     0.87  0.00    1.00  0.62 -0.48 -0.40 -0.44
## frac       0.59 -0.21    0.62  1.00 -0.89 -0.87 -0.89
## verbal    -0.41  0.06   -0.48 -0.89   1.00  0.97  0.99
## math      -0.35  0.10   -0.40 -0.87   0.97  1.00  0.99
## sat       -0.38  0.08   -0.44 -0.89   0.99  0.99  1.00

```

6.3.8 Add Student to Teacher Ratio

```

SAT_M3 <- lm(data=SAT, sat~salary+frac+ratio)
summary(SAT_M3)

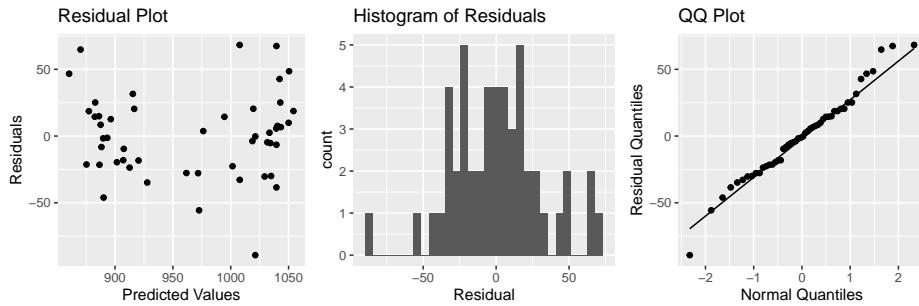
```

```

##
## Call:
## lm(formula = sat ~ salary + frac + ratio, data = SAT)
##
## Residuals:
##      Min      1Q      Median      3Q      Max
## -89.244 -21.485  -0.798  17.685  68.262
##
## Coefficients:
##             Estimate Std. Error t value     Pr(>|t|)
## (Intercept) 1057.8982   44.3287  23.865 <0.000000000000002 ***
## salary       2.5525    1.0045   2.541      0.0145 *
## frac        -2.9134    0.2282 -12.764 <0.000000000000002 ***
## ratio       -4.6394    2.1215  -2.187      0.0339 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 32.41 on 46 degrees of freedom
## Multiple R-squared:  0.8239, Adjusted R-squared:  0.8124
## F-statistic: 71.72 on 3 and 46 DF,  p-value: < 0.0000000000000022

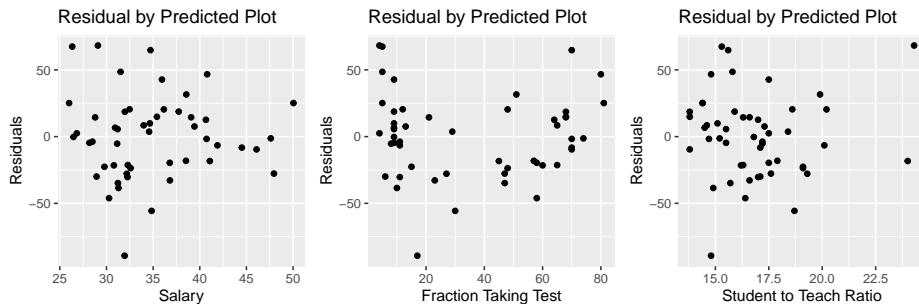
```

6.3.9 Residual Plots for SAT 3-variable Model



There is some sign of a quadratic trend in the residual plot, creating concern about the linearity assumption.

6.3.10 Plots of Residuals Against Predictors



There is also a quadratic trend in the plot involving the fraction variable.

6.3.11 Model Using Frac^2

```
SAT_M4 <- lm(data=SAT, sat~salary+frac+I(frac^2)+ratio)
summary(SAT_M4)

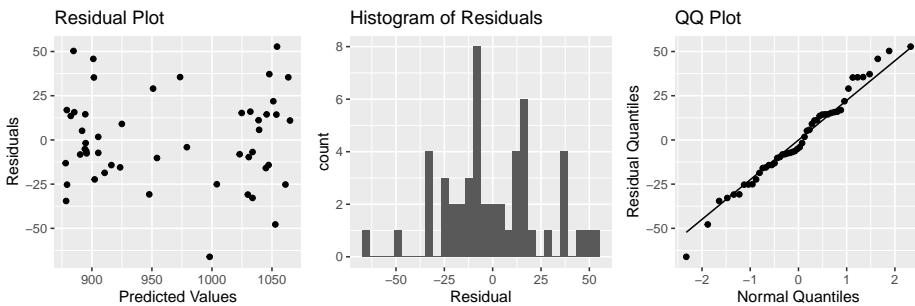
##
## Call:
## lm(formula = sat ~ salary + frac + I(frac^2) + ratio, data = SAT)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -66.09  -15.20   -4.64  15.06  52.77 
##
## Coefficients:
```

```

##             Estimate Std. Error t value      Pr(>|t|)
## (Intercept) 1039.21242   36.28206  28.643 < 0.0000000000000002 ***
## salary       1.80708    0.83150   2.173      0.0351 *
## frac        -6.64001   0.77668  -8.549     0.0000000000555 ***
## I(frac^2)    0.05065    0.01025   4.942     0.0000111676728 ***
## ratio        -0.04058   1.96174  -0.021      0.9836
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.38 on 45 degrees of freedom
## Multiple R-squared:  0.8858, Adjusted R-squared:  0.8757
## F-statistic: 87.28 on 4 and 45 DF,  p-value: < 0.0000000000000022

```

6.3.12 Residual Plots for Quadratic SAT Model



6.3.13 Model with Linear Term on Frac

```

summary(SAT_M3)

##
## Call:
## lm(formula = sat ~ salary + frac + ratio, data = SAT)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -89.244 -21.485 -0.798  17.685  68.262 
## 
## Coefficients:
##             Estimate Std. Error t value      Pr(>|t|)
## (Intercept) 1057.8982   44.3287  23.865 <0.0000000000000002 ***
## salary       2.5525    1.0045   2.541      0.0145 *
## frac        -2.9134   0.2282 -12.764 <0.0000000000000002 ***
## ratio        -4.6394   2.1215  -2.187      0.0339 *

```

```

## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 32.41 on 46 degrees of freedom
## Multiple R-squared: 0.8239, Adjusted R-squared: 0.8124
## F-statistic: 71.72 on 3 and 46 DF, p-value: < 0.00000000000000022

```

6.3.14 Interpretations for Model with Linear Terms

On average, a \$1,000 dollar increase in average teacher salary is associated with a 2.5 point increase in average SAT score assuming fraction of students taking the SAT, and student to teacher ratio are held constant.

On average, a 1% increase in percentage of students taking the SAT is associated with a 2.9 point decrease in average SAT score assuming average teacher salary, and student to teacher ratio are held constant.

On average, a 1 student per teacher increase in student to teacher ratio is associated with a 4.6 point from in average SAT score, assuming average teacher salary, and percentage of students taking the SAT are held constant.

6.3.15 Model with Quadratic Term on Frac

```

summary(SAT_M4)

##
## Call:
## lm(formula = sat ~ salary + frac + I(frac^2) + ratio, data = SAT)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -66.09 -15.20  -4.64  15.06  52.77
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept) 1039.21242   36.28206  28.643 < 0.000000000000002 ***
## salary       1.80708    0.83150   2.173      0.0351 *
## frac        -6.64001   0.77668  -8.549      0.000000000555 ***
## I(frac^2)    0.05065    0.01025   4.942      0.0000111676728 ***
## ratio        -0.04058   1.96174  -0.021      0.9836
##
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 26.38 on 45 degrees of freedom
## Multiple R-squared: 0.8858, Adjusted R-squared: 0.8757

```

```
## F-statistic: 87.28 on 4 and 45 DF, p-value: < 0.00000000000000022
```

6.3.16 Interpretations for Model with Linear Terms

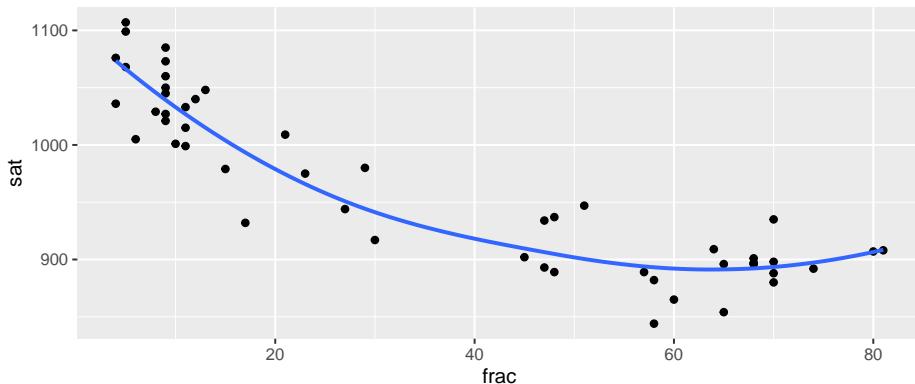
On average, a \$1,000 dollar increase in average teacher salary is associated with a 1.8 point increase in average SAT score assuming fraction of students taking the SAT, and student to teacher ratio are held constant.

On average, a 1 student per teacher increase in student to teacher ratio is associated with a 0.05 point from in average SAT score, assuming average teacher salary, and percentage of students taking the SAT are held constant.

We cannot give a clear interpretation of the fraction variable, since it occurs in both linear and quadratic terms. In fact, the vertex of the parabola given by $y = -6.64x + 0.05x^2$ occurs at $x = \frac{6.64}{2(0.05)} \approx 66$. So the model estimates that SAT score decreases in a quadratic fashion with respect to fraction taking the test, until that fraction reaches 66 percent of student, then is expected to increase.

6.3.17 Plot of SAT and Frac

```
ggplot(data=SAT, aes(x=frac, y=sat)) + geom_point() + stat_smooth(se=FALSE)
```



We do see some possible quadratic trend, but we should be really careful about extrapolation.

6.3.18 Summary of SAT Model

- Modeling SAT scores based on teacher salary alone led to misleading results, due to Simpson's Paradox. This is corrected by adding percentage of students taking the test to the model.

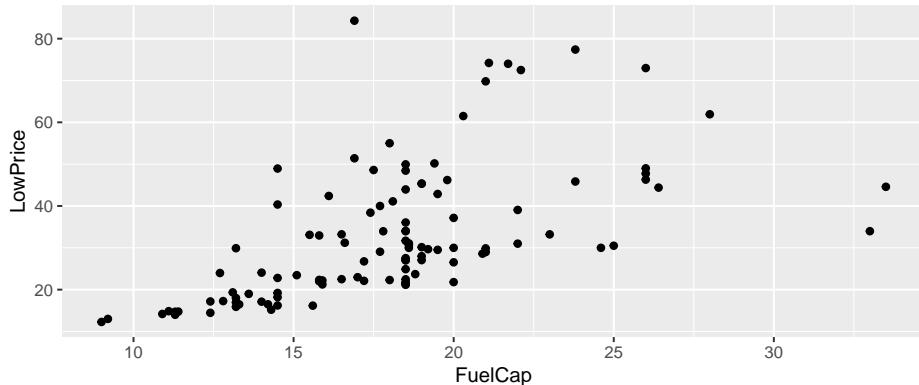
- Including a quadratic term on the proportion taking the test improves the model fit, and validity of model assumptions, but also makes the model harder to interpret. We need to use judgement when deciding whether or not to include quadratic or higher power terms.
- There is no clear reason to expect an interaction between these variables, so we did not include an interaction effect in the model.

6.4 Polynomial Regression

6.4.1 Car Price and Fuel Capacity

The plot shows the relationship between price (in thousands) and fuel capacity (in gallons) for a sample of 2015 new cars.

```
ggplot(data=Cars2015, aes(y=LowPrice, x=FuelCap)) + geom_point()
```



6.4.2 Polynomial Regression for Car Fuel Cap.

To this point, we have mostly considered models with the explanatory variables entered as linear functions. There is no reason we need to restrict ourselves in this manner. Sometimes, using powers of explanatory variables will allow us to better model our response.

Suppose Y is the price of a 2015 new car, and X is its fuel capacity in gallons.

Let's consider the following models.

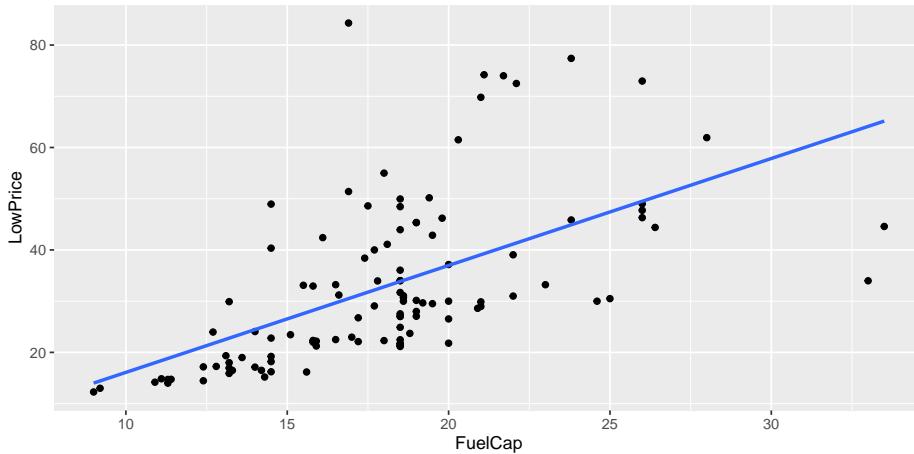
$$\begin{aligned} Y &= \beta_0 + \beta_1 X + \epsilon_i \\ Y &= \beta_0 + \beta_1 X + \beta_2 X^2 + \epsilon_i \\ Y &= \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon_i \\ Y &= \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 X^4 + \epsilon_i \end{aligned}$$

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 X^4 + \beta_5 X^5 + \epsilon_i,$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma)$ in each model.

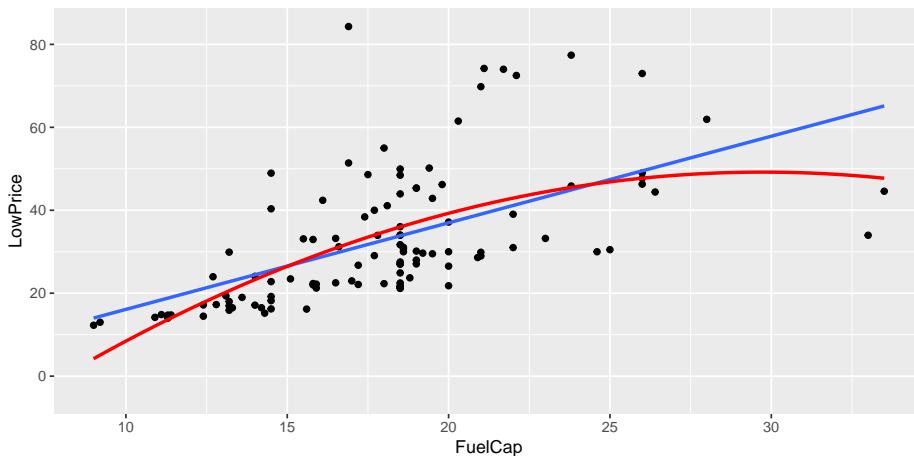
6.4.3 Price by Fuel Cap: Linear Model

```
ggplot(data=Cars2015, aes(y=LowPrice, x=FuelCap)) + geom_point() + stat_smooth(method="lm", se=FALSE)
```



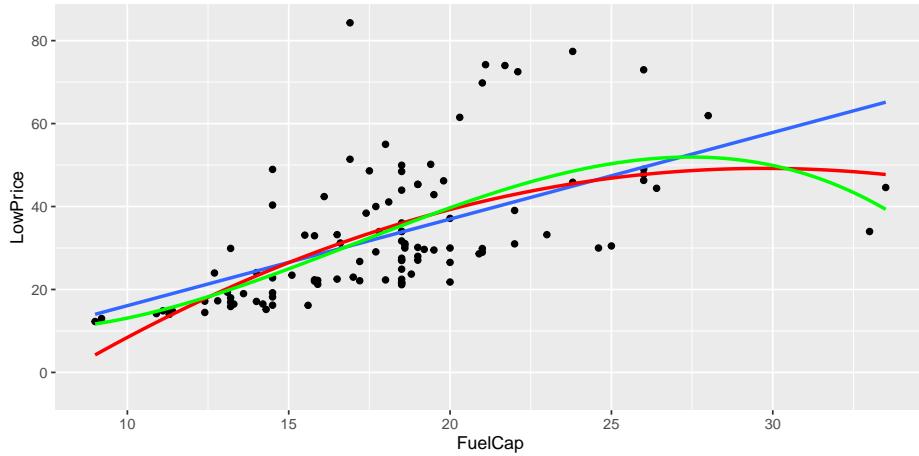
6.4.4 Price by Fuel Cap: Quadratic Model

```
ggplot(data=Cars2015, aes(y=LowPrice, x=FuelCap)) + geom_point() + stat_smooth(method="lm", se=FALSE) + stat_smooth(method="lm", se=TRUE, fill=NA, formula=y ~ poly(x, 2, raw=TRUE), colour="red")
```



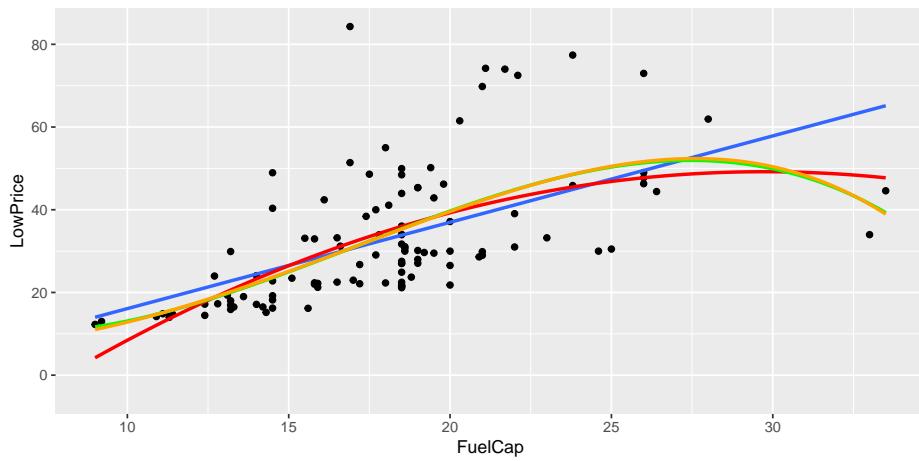
6.4.5 Price by Fuel Cap: Cubic Model

```
ggplot(data=Cars2015, aes(y=LowPrice, x=FuelCap)) + geom_point() + stat_smooth(method="lm", se=TRUE, fill=NA, formula=y ~ poly(x, 2, raw=TRUE), colour="red") + stat_smooth(method="lm", se=TRUE, fill=NA, formula=y ~ poly(x, 3, raw=TRUE), colour="green")
```



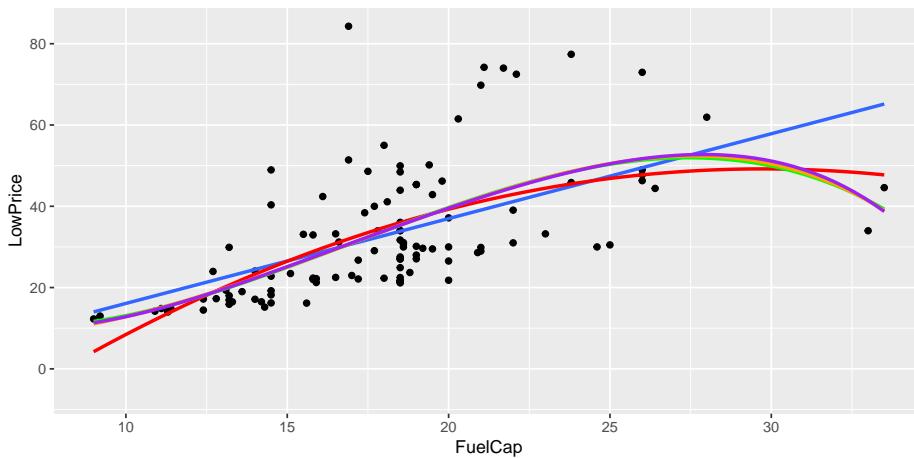
6.4.6 Price by Fuel Cap: Fourth-Degree Model

```
ggplot(data=Cars2015, aes(y=LowPrice, x=FuelCap)) + geom_point() + stat_smooth(method="lm", se=TRUE, fill=NA, formula=y ~ poly(x, 3, raw=TRUE), colour="green") + stat_smooth(method="lm", se=TRUE, fill=NA, formula=y ~ poly(x, 4, raw=TRUE), colour="orange")
```



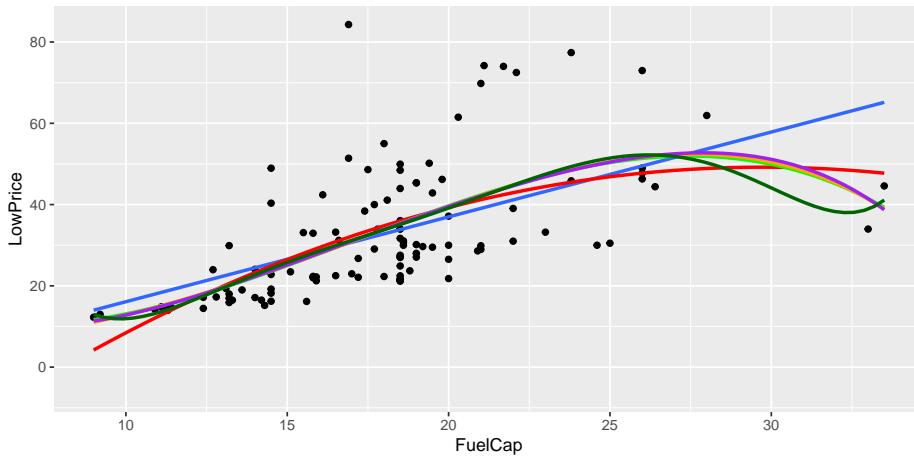
6.4.7 Price by Fuel Cap: Fifth-Degree Model

```
ggplot(data=Cars2015, aes(y=LowPrice, x=FuelCap)) + geom_point() + stat_smooth(method="lm", se=FALSE, formula=y ~ poly(x, 3, raw=TRUE), colour="green") + stat_smooth(method="lm", se=FALSE, formula=y ~ poly(x, 4, raw=TRUE), colour="orange") + stat_smooth(method="lm", se=FALSE, formula=y ~ poly(x, 5, raw=TRUE), colour="purple")
```



6.4.8 Price by Fuel Cap: Sixth-Degree Model

```
ggplot(data=Cars2015, aes(y=LowPrice, x=FuelCap)) + geom_point() + stat_smooth(method="lm", se=FALSE, formula=y ~ poly(x, 6, raw=TRUE), colour="green") + stat_smooth(method="lm", se=FALSE, formula=y ~ poly(x, 7, raw=TRUE), colour="orange") + stat_smooth(method="lm", se=FALSE, formula=y ~ poly(x, 8, raw=TRUE), colour="purple")
```



6.4.9 Fitting Polynonimal Models in R

```
CarLength_M1 <- lm(data=Cars2015, LowPrice~FuelCap)
CarLength_M2 <- lm(data=Cars2015, LowPrice~FuelCap + I(FuelCap^2))
CarLength_M3 <- lm(data=Cars2015, LowPrice~FuelCap + I(FuelCap^2) + I(FuelCap^3))
CarLength_M4 <- lm(data=Cars2015, LowPrice~FuelCap + I(FuelCap^2) + I(FuelCap^3) + I(FuelCap^4))
CarLength_M5 <- lm(data=Cars2015, LowPrice~FuelCap + I(FuelCap^2) + I(FuelCap^3) + I(FuelCap^4) + I(FuelCap^5))
CarLength_M6 <- lm(data=Cars2015, LowPrice~FuelCap + I(FuelCap^2) + I(FuelCap^3) + I(FuelCap^4) + I(FuelCap^5) + I(FuelCap^6))
```

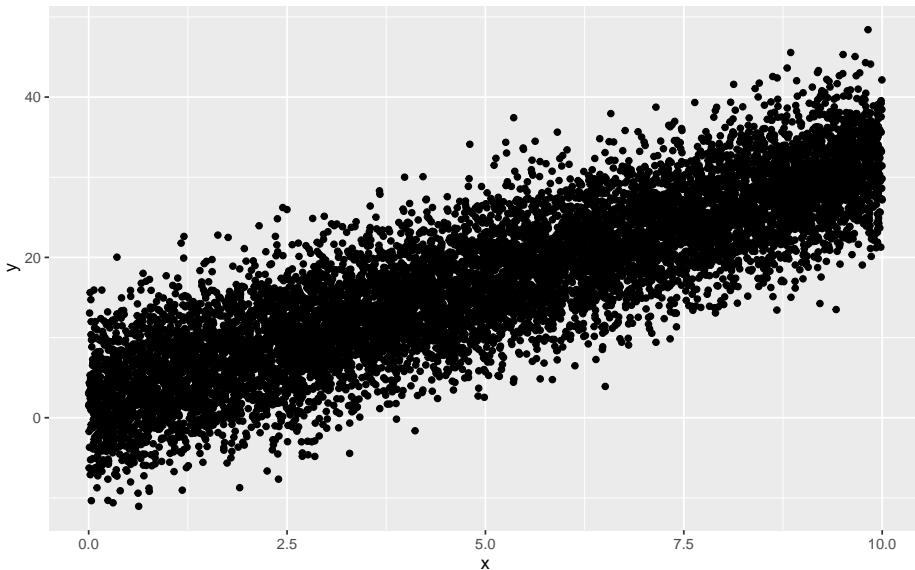
6.4.10 R^2 for each Model

Degree	R2
1	0.3287984
2	0.3744551
3	0.3908866
4	0.3909811
5	0.3910171
6	0.3926863

6.4.11 Model Complexity: Simulation Example

```
set.seed(02252019)
x <- runif(10000, 0, 10)
y <- 2 + 3*x + rnorm(10000,0, 5)
df <- data.frame(x,y)
```

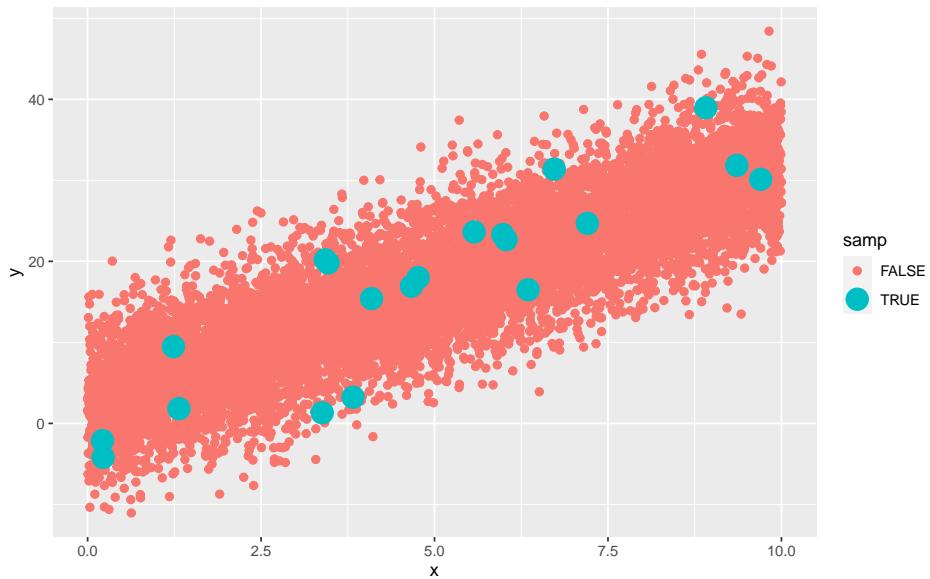
6.4.12 Plot of Simulated Data



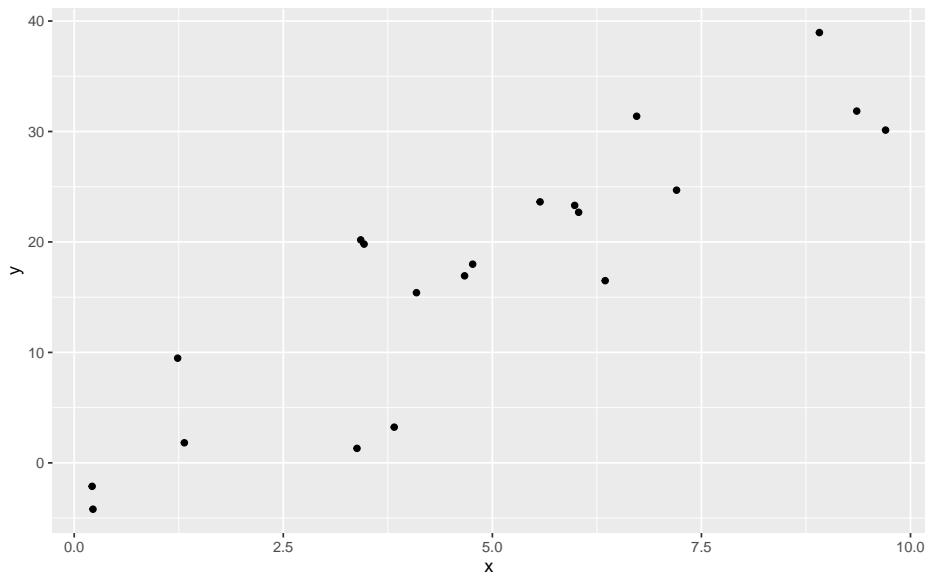
6.4.13 Selecting a Random Sample of 100

```
samp <- sample(1:nrow(df), 20)
Sampdf <- df[samp, ]
df$samp <- rownames(df) %in% samp
df <- df %>% arrange(samp)
```

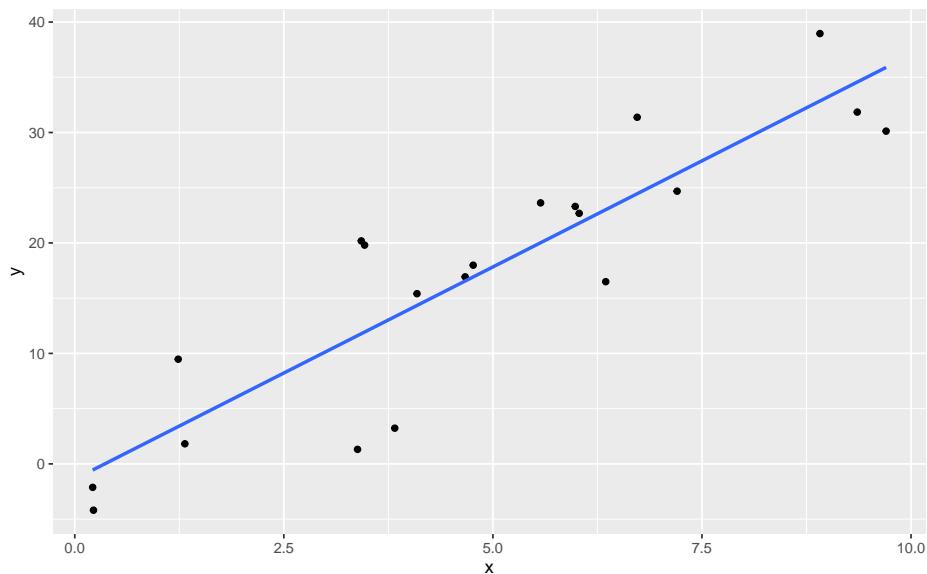
6.4.14 Plot Including Sample



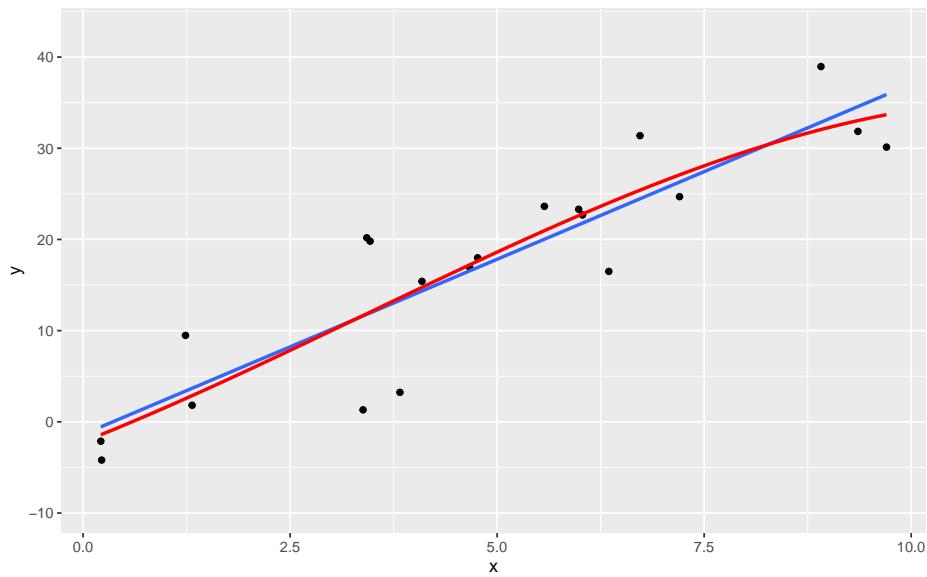
6.4.15 Plot of Sample Data



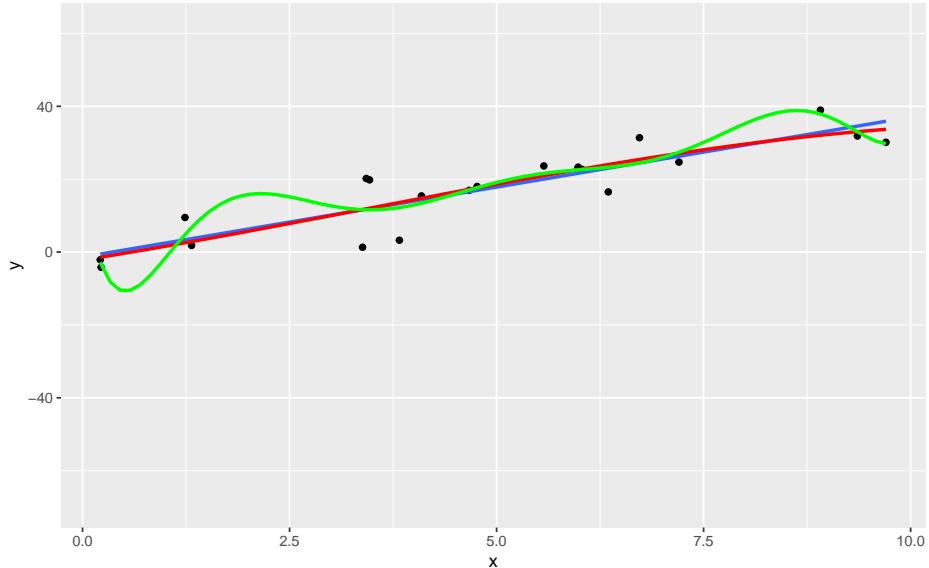
6.4.16 Linear Model to Sample Data



6.4.17 Cubic Model



6.4.18 Eighth-Degree Model



6.4.19 R^2 Values for Polynomial Models

```
Sim_M1 <- lm(data=Sampdf, y~x); summary(Sim_M1)$r.squared
```

```
## [1] 0.7915161
```

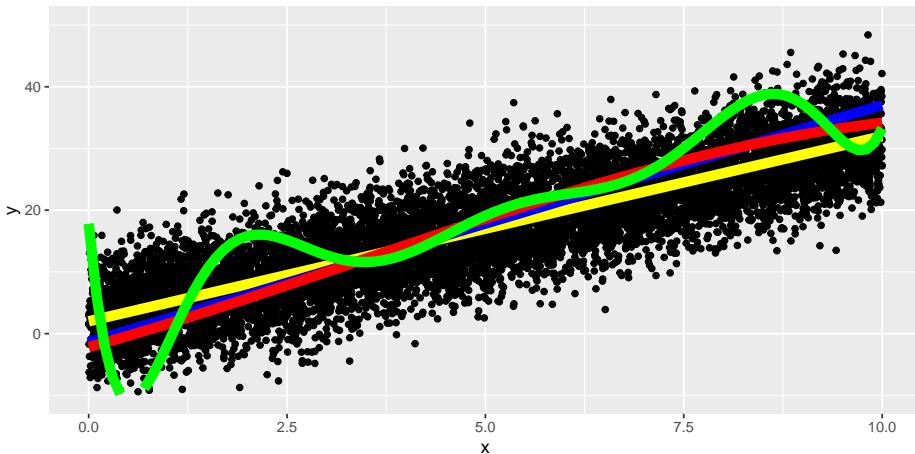
```
Sim_M2 <- lm(data=Sampdf, y~x+I(x^2)+I(x^3)); summary(Sim_M2)$r.squared
```

```
## [1] 0.7976537
```

```
Sim_M3 <- lm(data=Sampdf, y~x+I(x^2)+I(x^3)+I(x^4)+I(x^5)+I(x^6)+I(x^7)+I(x^8)); summary(Sim_M3)$r.squared
```

```
## [1] 0.826652
```

6.4.20 Predicting Data from Larger Population



Although the higher-degree model fits the sample best, it is clear that the linear model is the correct one, and would perform better on future data.

The higher-degree models overfits the sample data, capturing sampling variability in addition to true signal.

6.4.21 Model Complexity and Fit

- Adding terms to a model can never decrease R^2 .
- This is because adding new terms makes that model more flexible. At worse, the coefficients for the new term would be zero, and the model fit would be unchanged.
- This does NOT mean that adding new terms to the model will improve its ability to predict future data, not used to train the model.
- Adding extra terms can cause the model to “overfit” the data on which it was trained, and thus perform worse on new data than a simpler model would.
- This applies to adding additional variables to a model, in addition to adding higher power terms in a polynomial model.

6.5 Adjusted R^2 , AIC, and BIC

6.5.1 Variable Selection

When additional variables are added to a model, RSS never increases, hence R^2 never decreases.

Other diagnostics have been introduced to decrease when a term is added to a model and does little to help explain variability.

These include:

- Adjusted R^2
- Akaike Information Criterion (AIC)
- Bayesian Information Criterion (BIC)

6.5.2 Adjusted R^2

$$R^2 = 1 - \frac{\text{SSR}}{\text{SST}} = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}.$$

Adjusted R^2 :

$$R_a^2 = 1 - \frac{\text{SSR}/(n - (p + 1))}{\text{SST}/(n - 1)} = 1 - \left(\frac{n - 1}{n - (p + 1)} \right) (1 - R^2)$$

Large values of R_a^2 are desirable.

Adding variables that result in little to no change in RSS results in a decrease in adjusted R^2 .

6.5.3 Akaike Information Criterion (AIC)

$$\text{AIC} = n \log \left(\frac{\text{SSR}}{n} \right) + 2(p + 1)$$

Low values of AIC are desirable.

As p increases, RSS decreases, but $2p$ increases. The question is whether RSS decreases enough to offset the increase in $2(p + 1)$.

6.5.4 Bayesian Information Criterion (BIC)

$$\text{BIC} = n \log \left(\frac{\text{RSS}}{n} \right) + \log(n)(p + 1)$$

Low values of BIC are desirable.

BIC penalizes additional terms more harshly than AIC, especially when n is large. Thus BIC will lead to a “smaller” model.

6.5.5 Comparison of Adjusted R^2 , AIC, BIC

Degree	R2	AdjR2	AIC	BIC
1	0.3287984	0.3225836	882.2615	890.3629
2	0.3744551	0.3627627	876.5123	887.3143
3	0.3908866	0.3736476	875.5843	889.0867
4	0.3909811	0.3677804	877.5672	893.7701
5	0.3910171	0.3617390	879.5607	898.4641
6	0.3926863	0.3573088	881.2588	902.8627

Adjusted R^2 and AIC favor the 3rd degree model. BIC favors the 2nd degree model.

6.5.6 Output for Quadratic Model

```
summary(CarLength_M2)

##
## Call:
## lm(formula = LowPrice ~ FuelCap + I(FuelCap^2), data = Cars2015)
##
## Residuals:
##      Min      1Q      Median      3Q      Max 
## -17.477  -8.710   -3.047    5.139   52.348 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -43.29149   14.71368  -2.942  0.00400 ** 
## FuelCap       6.21903    1.50430   4.134 0.000071 *** 
## I(FuelCap^2) -0.10454    0.03741  -2.795  0.00616 ** 
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
##
## Residual standard error: 12.71 on 107 degrees of freedom
```

```
## Multiple R-squared:  0.3745, Adjusted R-squared:  0.3628
## F-statistic: 32.03 on 2 and 107 DF,  p-value: 0.00000000001258
```

6.5.7 Output for Cubic Model

```
summary(CarLength_M3)

##
## Call:
## lm(formula = LowPrice ~ FuelCap + I(FuelCap^2) + I(FuelCap^3),
##      data = Cars2015)
##
## Residuals:
##    Min      1Q  Median      3Q     Max
## -19.835 -7.787 -2.869  5.539 53.688
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 25.695569  43.326305  0.593  0.5544
## FuelCap     -5.072924   6.842216 -0.741  0.4601
## I(FuelCap^2)  0.473979   0.344124  1.377  0.1713
## I(FuelCap^3) -0.009266   0.005480 -1.691  0.0938 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.6 on 106 degrees of freedom
## Multiple R-squared:  0.3909, Adjusted R-squared:  0.3736
## F-statistic: 22.67 on 3 and 106 DF,  p-value: 0.00000000002036
```

6.5.8 Conclusions for Car Price Polynomial Example

The adjusted R², and AIC measures favor the cubic model:

$$\widehat{\text{Price}} = 25.70 - 5.07 \times \text{Fuel Cap.} + 0.47 \times \text{Fuel Cap.}^2 - 0.01 \times \text{Fuel Cap.}^3$$

The BIC measure favors the quadratic model:

$$\widehat{\text{Price}} = -43.29 + 6.21 \times \text{Fuel Cap.} - 0.10 \times \text{Fuel Cap.}^2$$

6.5.9 Interpreting Coefficients in Polynomial Regression

It is hard to attach meaning to the coefficients in polynomial regression. We cannot talk about holding Fuel Cap.² constant while increasing Fuel Cap.

We can get a general sense of the shape of the regression curve, from the signs of the coefficients, but that's about it.

This illustrates a trade-off between model complexity and interpretation. More complex models might fit the data better, but are typically harder to interpret.

6.5.10 Modeling for the Purpose of Interpretation

- Model driven by research question
- Include variables of interest
- Include potential confounders (like in SAT example)
- Aim for high R^2 but not highest
- Use F-tests, AIC, BIC, Adjusted R^2 to balance goodness of fit with model complexity
- Avoid messy transformations where possible
- Aim for model complex enough to capture nature of data, but simple enough to give clear interpretations

Chapter 7

Building Models for Prediction

7.1 Training and Test Error

7.1.1 Modeling for the Purpose of Prediction

- When our purpose is purely prediction, we don't need to worry about keeping the model simple enough to interpret.
- Goal is to fit data well enough to make good predictions on new data without modeling random noise (overfitting)
- A model that is too simple suffers from high bias
- A model that is too complex suffers from high variance and is prone to overfitting
- The right balance is different for every dataset
- Measuring error on data used to fit the model (training data) does not accurately predict how well model will be able to predict new data (test data)

7.1.2 Model for Predicting Car Price

Consider the following five models for predicting the price of a new car. The models are fit, using the sample of 110 cars released in 2015, that we've seen

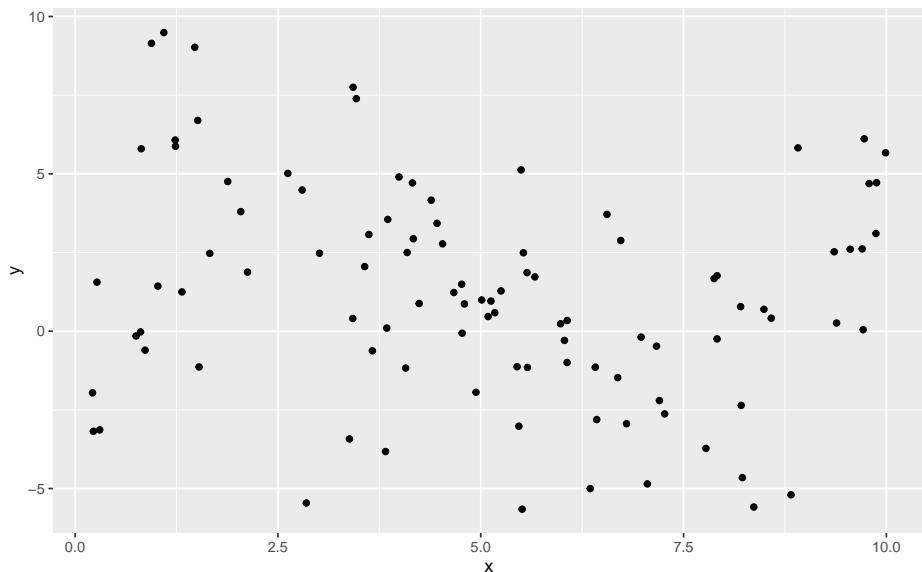
previously.

```
M1 <- lm(data=Cars2015, LowPrice~Acc060)
M2 <- lm(data=Cars2015, LowPrice~Acc060 + I(Acc060^2))
M3 <- lm(data=Cars2015, LowPrice~Acc060 + I(Acc060^2) + Size)
M4 <- lm(data=Cars2015, LowPrice~Acc060 + I(Acc060^2) + Size + PageNum)
M5 <- lm(data=Cars2015, LowPrice~Acc060 * I(Acc060^2) * Size)
M6 <- lm(data=Cars2015, LowPrice~Acc060 * I(Acc060^2) * Size * PageNum)
```

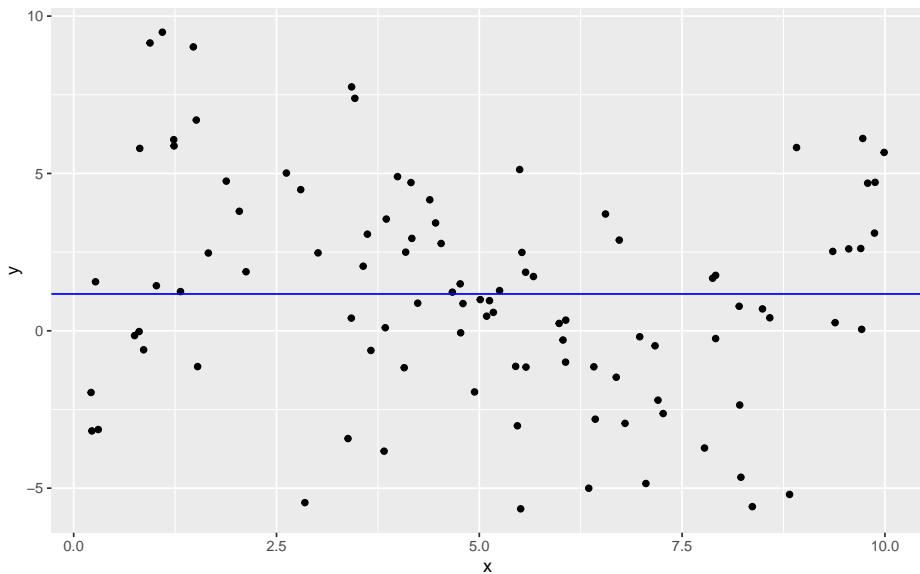
Which would perform best if we were making predictions on a different set of cars, released in 2015?

7.1.3 Prediction Simulation Example

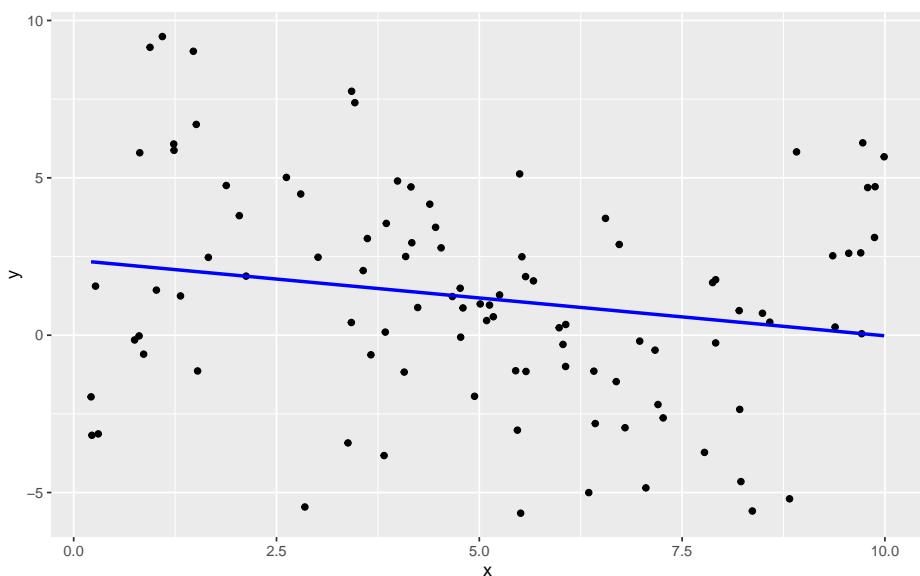
Suppose we have a set of 100 observations of a single explanatory variable x , and response variable y . A scatterplot of the data is shown below.



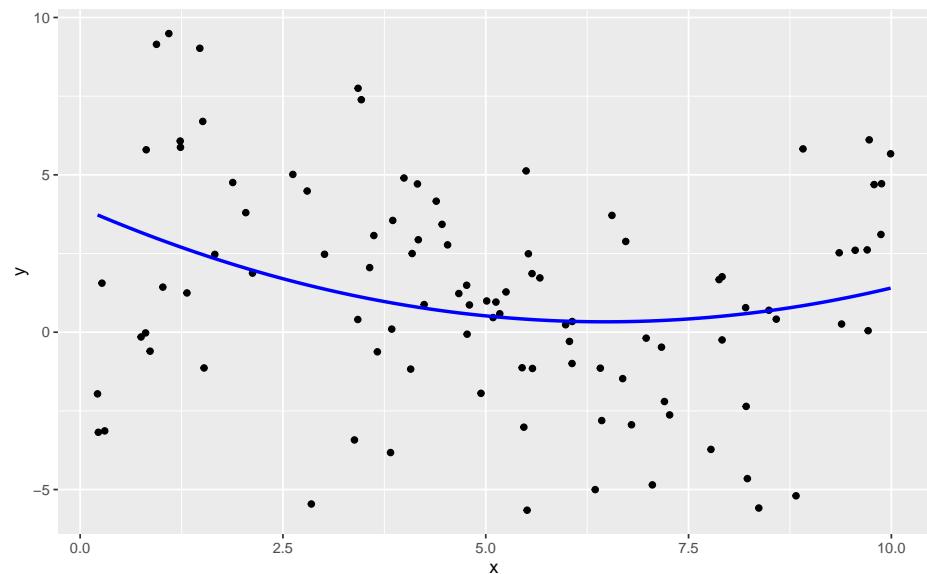
7.1.4 Constant Model to Sample Data



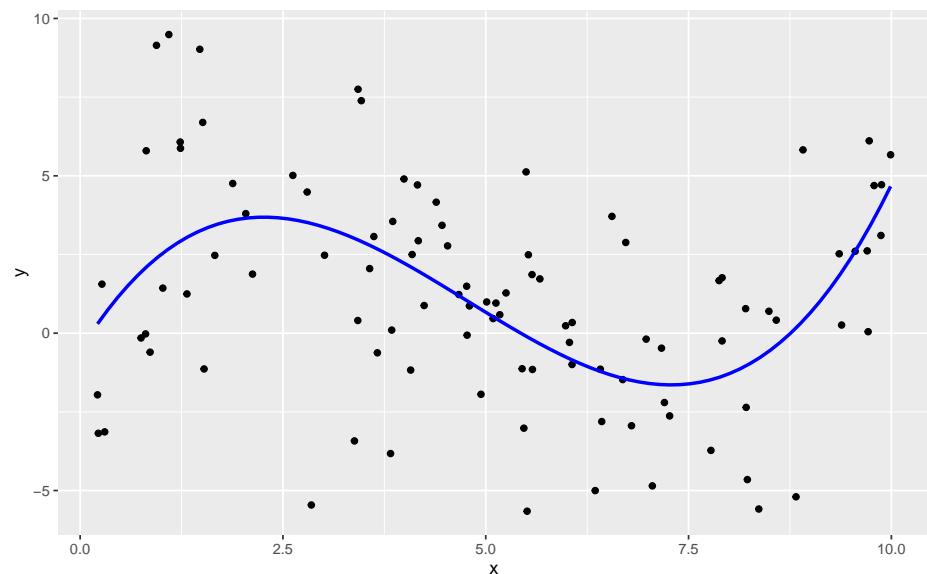
7.1.5 Linear Model to Sample Data



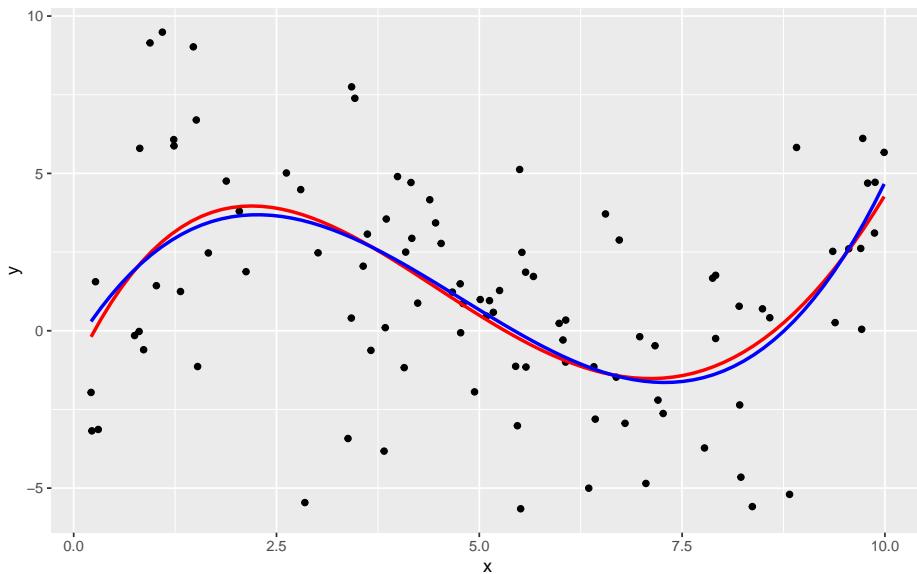
7.1.6 Quadratic Model



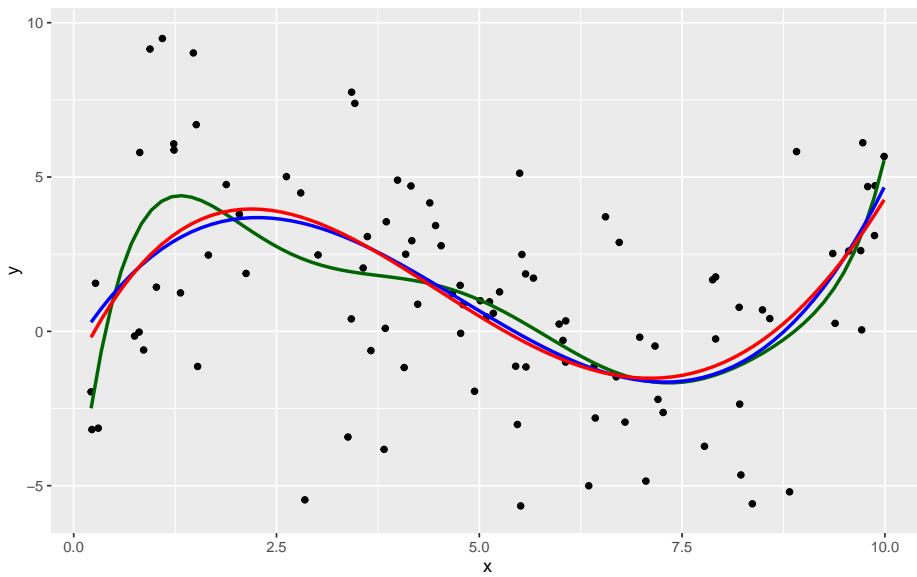
7.1.7 Cubic Model



7.1.8 Quartic Model

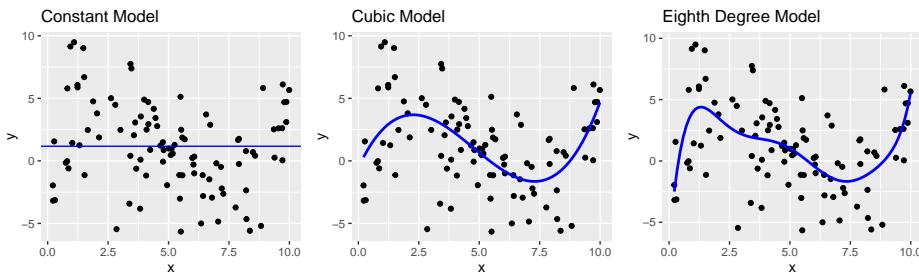


7.1.9 Degree 8 Model



7.1.10 Model Complexity

- The complexity of the model increases as we add higher-order terms. This makes the model more flexible. The curve is allowed to have more twists and bends.
- For higher-order, more complex models, individual points have more influence on the shape of the curve.



7.1.11 New Data for Prediction

Now, suppose we have a new dataset of 100, x-values, and want to predict y . The first 5 rows of the new dataset are shown

x	Prediction
3.196237	?
1.475586	?
5.278882	?
5.529299	?
7.626731	?

7.1.12 Fit Polynomial Models

```
Sim_M0 <- lm(data=Sampdf, y~1)
Sim_M1 <- lm(data=Sampdf, y~x)
Sim_M2 <- lm(data=Sampdf, y~x+I(x^2))
Sim_M3 <- lm(data=Sampdf, y~x+I(x^2)+I(x^3))
Sim_M4 <- lm(data=Sampdf, y~x+I(x^2)+I(x^3)+I(x^4))
Sim_M5 <- lm(data=Sampdf, y~x+I(x^2)+I(x^3)+I(x^4)+I(x^5))
Sim_M6 <- lm(data=Sampdf, y~x+I(x^2)+I(x^3)+I(x^4)+I(x^5)+I(x^6))
Sim_M7 <- lm(data=Sampdf, y~x+I(x^2)+I(x^3)+I(x^4)+I(x^5)+I(x^6)+I(x^7))
Sim_M8 <- lm(data=Sampdf, y~x+I(x^2)+I(x^3)+I(x^4)+I(x^5)+I(x^6)+I(x^7)+I(x^8))
```

7.1.13 Predictions for New Data

```
Newdf$Deg0Pred <- predict(Sim_M0, newdata=Newdf)
Newdf$Deg1Pred <- predict(Sim_M1, newdata=Newdf)
Newdf$Deg2Pred <- predict(Sim_M2, newdata=Newdf)
Newdf$Deg3Pred <- predict(Sim_M3, newdata=Newdf)
Newdf$Deg4Pred <- predict(Sim_M4, newdata=Newdf)
Newdf$Deg5Pred <- predict(Sim_M5, newdata=Newdf)
Newdf$Deg6Pred <- predict(Sim_M6, newdata=Newdf)
Newdf$Deg7Pred <- predict(Sim_M7, newdata=Newdf)
Newdf$Deg8Pred <- predict(Sim_M8, newdata=Newdf)
```

7.1.14 Predicted Values

```
kable(Newdf %>% dplyr::select(-c(y, samp)) %>% round(2) %>% head(5))
```

	x	Deg0Pred	Deg1Pred	Deg2Pred	Deg3Pred	Deg4Pred	Deg5Pred	Deg6Pred	Deg7Pred
108	5.53	1.17	1.05	0.41	-0.14	-0.30	0.10	0.40	0.25
4371	2.05	1.17	1.89	2.02	3.66	3.95	4.26	3.82	3.37
4839	3.16	1.17	1.63	1.29	3.24	3.35	2.78	2.24	2.15
6907	2.06	1.17	1.89	2.02	3.66	3.95	4.25	3.80	3.36
7334	2.92	1.17	1.68	1.42	3.43	3.60	3.14	2.51	2.31

7.1.15 Predicted Values and True y

In fact, since these data were simulated, we know the true value of y .

```
kable(Newdf %>% dplyr::select(-c(samp)) %>% round(2) %>% head(5))
```

	x	y	Deg0Pred	Deg1Pred	Deg2Pred	Deg3Pred	Deg4Pred	Deg5Pred	Deg6Pred	Deg7Pred
108	5.53	5.49	1.17	1.05	0.41	-0.14	-0.30	0.10	0.40	0.25
4371	2.05	3.92	1.17	1.89	2.02	3.66	3.95	4.26	3.82	3.37
4839	3.16	1.46	1.17	1.63	1.29	3.24	3.35	2.78	2.24	2.15
6907	2.06	6.79	1.17	1.89	2.02	3.66	3.95	4.25	3.80	3.36
7334	2.92	-1.03	1.17	1.68	1.42	3.43	3.60	3.14	2.51	2.31

7.1.16 Evaluating Predictions - RMSE

For quantitative response variables, we can evaluate the predictions by calculating the average of the squared differences between the true and predicted values. Often, we look at the square root of this quantity. This is called the Root Mean Square Prediction Error (RMSPE).

$$\text{RMSPE} = \sqrt{\sum_{i=1}^{n'} \frac{(y_i - \hat{y}_i)^2}{n'}},$$

where n' represents the number of new cases being predicted.

7.1.17 Calculate Prediction Error

```
RMSPE0 <- sqrt(mean((Newdf$y-Newdf$Deg0Pred)^2))
RMSPE1 <- sqrt(mean((Newdf$y-Newdf$Deg1Pred)^2))
RMSPE2 <- sqrt(mean((Newdf$y-Newdf$Deg2Pred)^2))
RMSPE3 <- sqrt(mean((Newdf$y-Newdf$Deg3Pred)^2))
RMSPE4 <- sqrt(mean((Newdf$y-Newdf$Deg4Pred)^2))
RMSPE5 <- sqrt(mean((Newdf$y-Newdf$Deg5Pred)^2))
RMSPE6 <- sqrt(mean((Newdf$y-Newdf$Deg6Pred)^2))
RMSPE7 <- sqrt(mean((Newdf$y-Newdf$Deg7Pred)^2))
RMSPE8 <- sqrt(mean((Newdf$y-Newdf$Deg8Pred)^2))
```

7.1.18 Prediction RMSPE by Model

Degree	RMSPE
0	4.051309
1	3.849624
2	3.726767
3	3.256592
4	3.283513
5	3.341336
6	3.346908
7	3.370821
8	3.350198

7.1.19 Training and Test Data

- The new data on which we make predictions is called **test data**.
- The data used to fit the model is called **training data**.
- In the training data, we know the values of the explanatory and response variables. In the test data, we know only the values of the explanatory variables and want to predict the values of the response variable.
- For quantitative response variables, if we find out the values of y , we can evaluate predictions, using RMSPE. This is referred to as **test error**.

- We can also calculate root mean square error on the training data from the known y-values. This is called **training error**.

7.1.20 Training Data Error

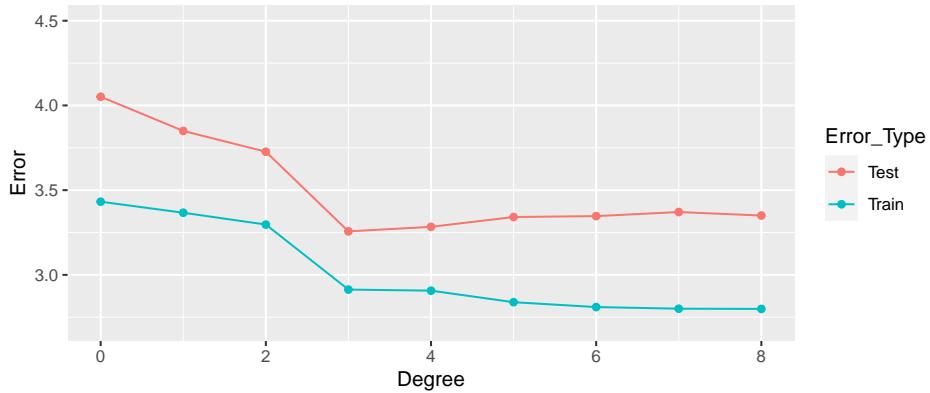
```
RMSE0 <- sqrt(mean(Sim_M0$residuals^2))
RMSE1 <- sqrt(mean(Sim_M1$residuals^2))
RMSE2 <- sqrt(mean(Sim_M2$residuals^2))
RMSE3 <- sqrt(mean(Sim_M3$residuals^2))
RMSE4 <- sqrt(mean(Sim_M4$residuals^2))
RMSE5 <- sqrt(mean(Sim_M5$residuals^2))
RMSE6 <- sqrt(mean(Sim_M6$residuals^2))
RMSE7 <- sqrt(mean(Sim_M7$residuals^2))
RMSE8 <- sqrt(mean(Sim_M8$residuals^2))
```

7.1.21 Test and Training Error

```
Degree <- 0:8
Test <- c(RMSPE0, RMSPE1, RMSPE2, RMSPE3, RMSPE4, RMSPE5, RMSPE6, RMSPE7, RMSPE8)
Train <- c(RMSE0, RMSE1, RMSE2, RMSE3, RMSE4, RMSE5, RMSE6, RMSE7, RMSE8)
RMSPEdf <- data.frame(Degree, Test, Train)
RMSPEdf
```

	Degree	Test	Train
## 1	0	4.051309	3.431842
## 2	1	3.849624	3.366650
## 3	2	3.726767	3.296821
## 4	3	3.256592	2.913233
## 5	4	3.283513	2.906845
## 6	5	3.341336	2.838738
## 7	6	3.346908	2.809928
## 8	7	3.370821	2.800280
## 9	8	3.350198	2.799066

7.1.22 Graph of RMSPE



- Training error decreases as model becomes more complex
- Testing error is lowest for the 3rd degree model, then starts to increase again

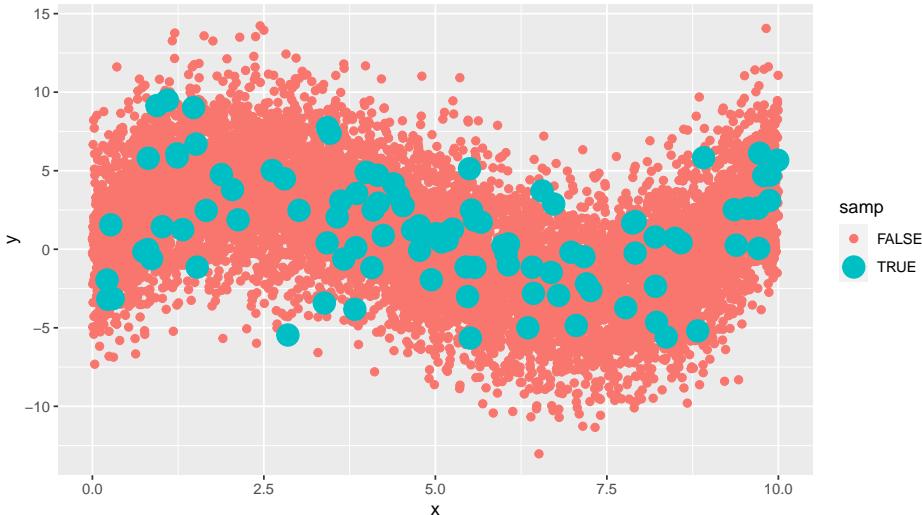
7.1.23 Third Degree Model Summary

```
summary(Sim_M3)

##
## Call:
## lm(formula = y ~ x + I(x^2) + I(x^3), data = Sampdf)
##
## Residuals:
##    Min     1Q   Median     3Q    Max 
## -8.9451 -1.7976  0.1685  1.3988  6.8064 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -0.54165   1.26803  -0.427  0.670221  
## x            4.16638   1.09405   3.808  0.000247 *** 
## I(x^2)      -1.20601   0.25186  -4.788 0.00000610 *** 
## I(x^3)       0.08419   0.01622   5.191 0.00000117 *** 
## ---        
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## Residual standard error: 2.973 on 96 degrees of freedom
## Multiple R-squared:  0.2794, Adjusted R-squared:  0.2569 
## F-statistic: 12.41 on 3 and 96 DF,  p-value: 0.0000006309
```

7.1.24 True Data Model

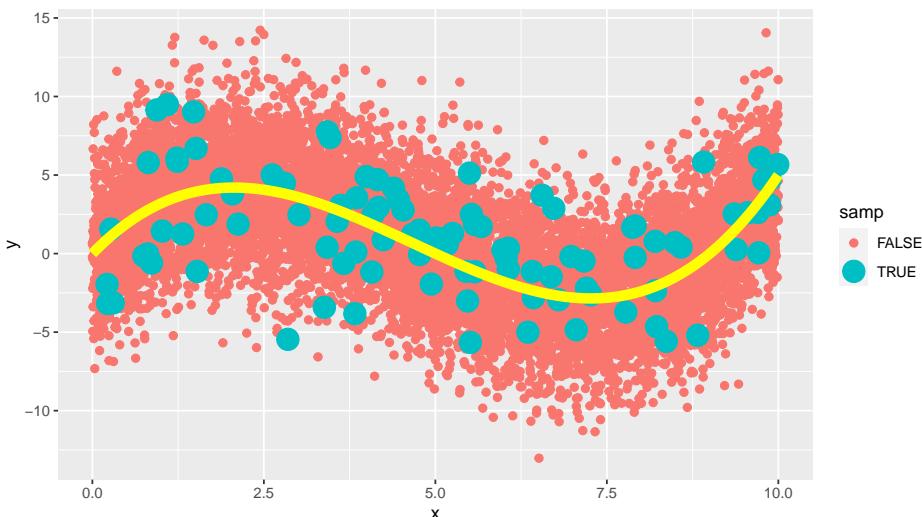
In fact, the data were generated from the model $y_i = 4.5x - 1.4x^2 + 0.1x^3 + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, 3)$



7.1.25 True Data Model

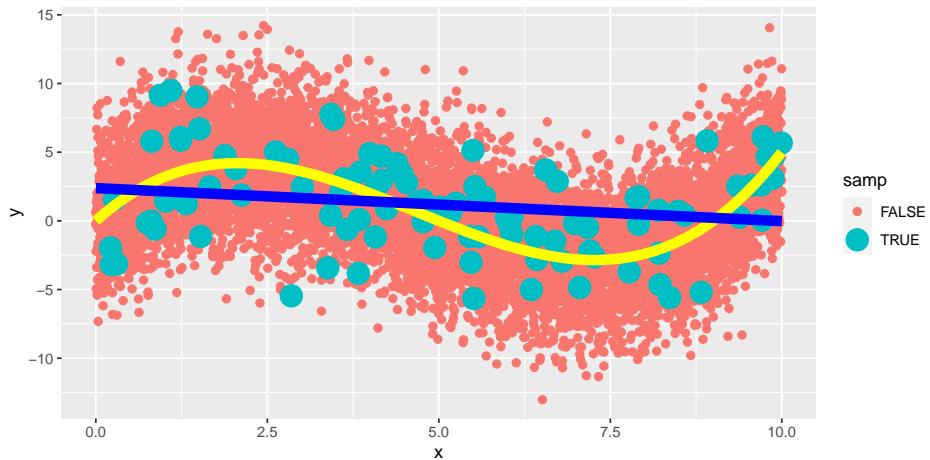
In fact, the data were generated from the model $y_i = 4.5x - 1.4x^2 + 0.1x^3 + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, 3)$

The yellow curve represents the true expected response curve.



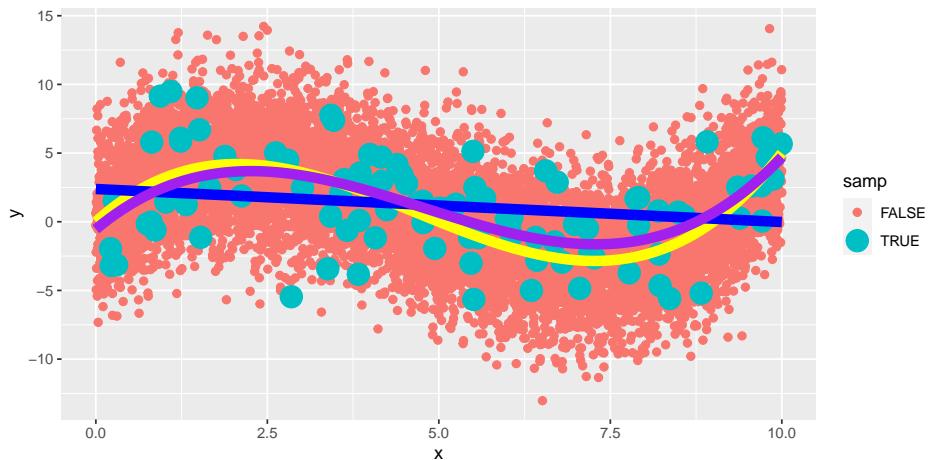
7.1.26 Linear Model on Larger Population

The yellow curve represents the true expected response curve.



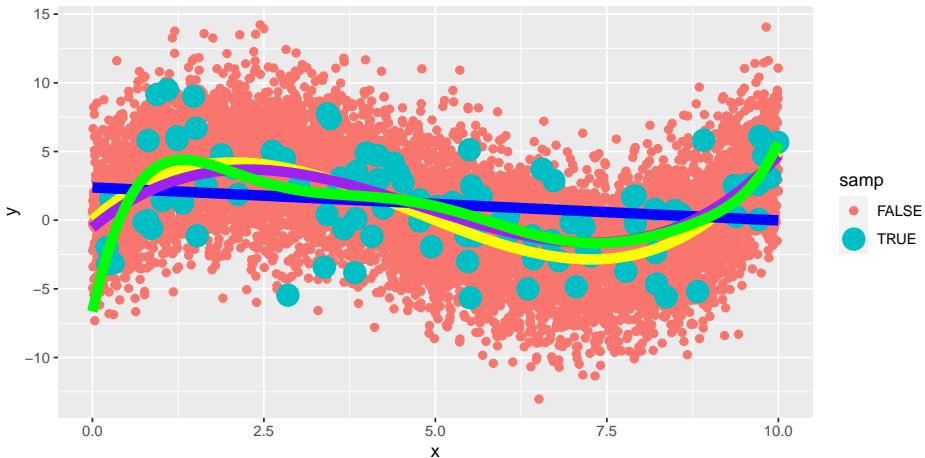
7.1.27 Cubic Model on Larger Population

The yellow curve represents the true expected response curve.



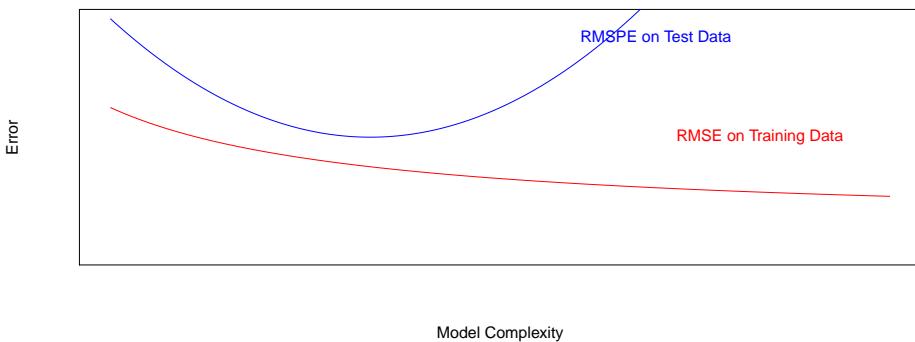
7.1.28 Eighth-Degree Model on Larger Population

The yellow curve represents the true expected response curve.



The 8th degree model performs worse than the cubic. The extra terms cause the model to be “too flexible,” and it starts to model random fluctuations (noise) in the training data, that do not capture the true trend for the population. This is called **overfitting**.

7.1.29 Model Complexity, Training Error, and Test Error



7.2 Variance-Bias Tradeoff

7.2.1 What Contributors to Prediction Error?

Suppose $Y_i = f(x_i) + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \sigma)$.

Let \hat{f} represent the function of our explanatory variable(s) x^* used to predict the value of response variable y^* . Thus $\hat{y}^* = \hat{f}(x^*)$.

There are three factors that contribute to the expected value of $(y^* - \hat{y})^2 = (y^* - \hat{f}(x^*))^2$.

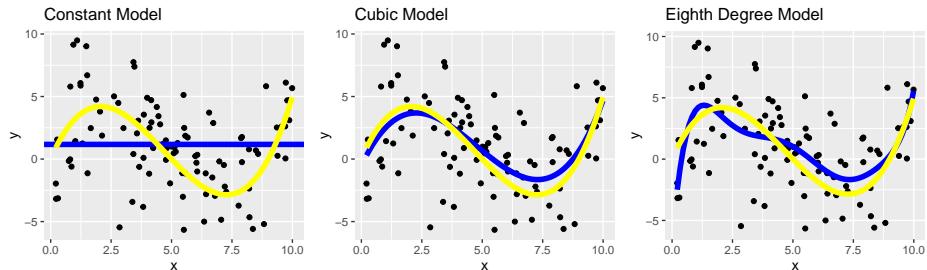
1. **Bias associated with fitting model:** Model bias pertains to the difference between the true response function value $f(x^*)$, and the average value of $\hat{f}(x^*)$ that would be obtained in the long run over many samples.
- for example, if the true response function f is cubic, then using a constant, linear, or quadratic model would result in biased predictions for most values of x^* .
2. **Variance associated with fitting model:** Individual observations in the training data are subject to random sampling variability. The more flexible a model is, the more weight is put on each individual observation increasing the variance associated with the model.
3. **Variability associated with prediction:** Even if we knew the true value $f(x^*)$, which represents the expected value of y^* given $x = x^*$, the actual value of y^* will vary due to random noise (i.e. the $\epsilon_i \sim \mathcal{N}(0, \sigma)$ term).

7.2.2 Variance and Bias

The third source of variability cannot be controlled or eliminated. The first two, however are things we can control. If we could figure out how to minimize bias while also minimizing variance associated with a prediction, that would be great! But...

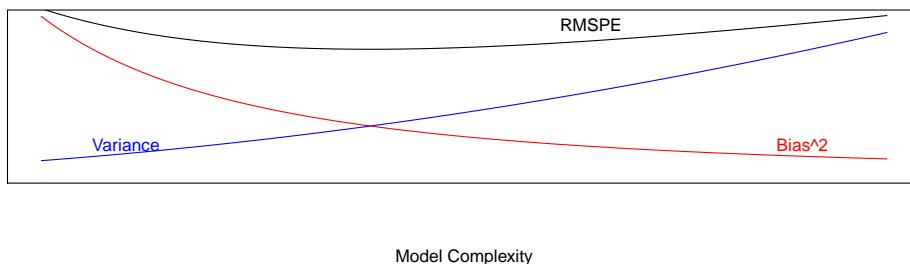
The constant model suffers from high bias. Since it does not include a linear, quadratic, or cubic term, it cannot accurately approximate the true regression function.

The Eighth degree model suffers from high variance. Although it could, in theory, approximate the true regression function correctly, it is too flexible, and is thrown off because of the influence of individual points with high degrees of variability.



7.2.3 Variance-Bias Tradeoff

As model complexity (flexibility) increases, bias decreases. Variance, however, increases.



In fact, it can be shown that:

$$\text{Expected RMSPE} = \text{Variance} + \text{Bias}^2$$

Our goal is to find the “sweetspot” where expected RMSPE is minimized.

7.3 Cross-Validation

7.3.1 Cross-Validation (CV)

We've seen that training error is not an accurate approximation of test error. Instead, we'll approximate test error, by setting aside a set of the training data, and using it as if it were a test set. This process is called **cross-validation**, and the set we put aside is called the **validation set**.

1. Partition data into disjoint sets (folds). Approximately 5 folds recommended.
2. Build a model using 4 of the 5 folds.
3. Use model to predict responses for remaining fold.
4. Calculate root mean square error $RMSPE = \sqrt{\frac{\sum((\hat{y}_i - y_i)^2)}{n'}}$.
5. Repeat for each of 5 folds.
6. Average RMSPE values across folds.

If computational resources permit, it is often beneficial to perform CV multiple times, using different sets of folds.

7.3.2 Cross-Validation Illustration

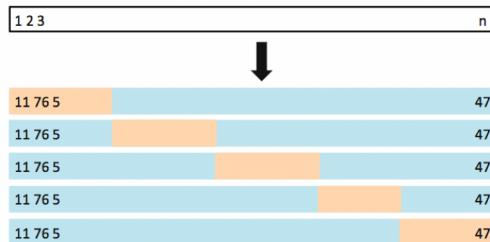


Image from Introduction to Statistical Learning by James, Witten, Hastie, Tibshirani

7.3.3 Models for Price of New Cars

Consider the following five models for predicting the price of a new car:

```
M1 <- lm(data=Cars2015, LowPrice~Acc060)
M2 <- lm(data=Cars2015, LowPrice~Acc060 + I(Acc060^2))
M3 <- lm(data=Cars2015, LowPrice~Acc060 + I(Acc060^2) + Size)
M4 <- lm(data=Cars2015, LowPrice~Acc060 + I(Acc060^2) + Size + PageNum)
M5 <- lm(data=Cars2015, LowPrice~Acc060 * I(Acc060^2) * Size)
M6 <- lm(data=Cars2015, LowPrice~Acc060 * I(Acc060^2) * Size * PageNum)
```

7.3.4 Cross Validation in Cars Dataset

To Use cross validation on the cars dataset, we'll fit each model to 4 folds (88 cars), while withholding the other 1 fold (22 cars).

We then make predictions on the last fold, and see which model yields the lowest MSPE.

t (Fold)

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## fold1    76   82   73   14   30   15  109    5  102   92   25   24    6
## fold2    65   80  107   62    8    4   91   75    2   41   99   81   37
## fold3    68  101  108   31   23   59  105   42   35   21    1   84  106
## fold4    50  104   13   52   29   98   66   54   49  110   38   48   63
## fold5    55   22   17    3   10   28   39   51   20   47   69   83   88
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22]
## fold1    36   16   57   44   72   53   86   94   34
## fold2    74   56   27   90   19   40   61   78   71
## fold3    60   43   77   64   67  100    7   58   87
## fold4    12  103   97   85   95   96   32   45   89
## fold5     9   70   46   11   33   79   26   93   18
```

7.3.5 CV Results

Rows show RMSPE, columns show model.

```
kable(RMSPE)
```

11.329817	11.460672	9.989957	10.130743	10.808705	10.66698
11.904377	12.406512	10.998581	10.993984	13.447907	17.27614
10.044277	9.399102	8.482429	9.167676	7.906023	13.84186
9.868489	9.081304	10.658656	10.661173	9.435986	12.09450
10.630225	9.538196	9.178224	9.681109	8.266503	18.80454

Mean RMSPE

```
apply(RMSPE, 2, mean)
```

```
## [1] 10.755437 10.377157 9.861570 10.126937 9.973025 14.536803
```

7.3.6 CV Conclusions

- Model 3 achieves the best performance using cross-validation.
- This is the model that includes a quadratic function of acceleration time and size.
- Including page number, or interactions harms the performance of the model.
- We should use model 3 if we want to make predictions on new data.

7.4 Variable Engineering

7.4.1 AirBnB Dataset

We consider a dataset with information on AirBnB's in the United States. Data come from <https://www.kaggle.com/rudymizrahi/airbnb-listings-in-major-us-cities-deloitte-ml>.

Our goal is to predict the prices of AirBnB's usinf information from 21 potential explanatory variables.

```
Train <- read.csv("Train.csv")
Test <- read.csv("Test.csv")

dim(Train)

## [1] 1000    22
```

```
dim(Test)
## [1] 1000 22
```

7.4.2 Gimpse of Training Data

```
glimpse(Train)
## Rows: 1,000
## Columns: 22
## $ id <int> 20746223, 16766700, 20652600, 16570684, 4435766~
## $ property_type <fct> House, Apartment, Apartment, House, Apartment, ~
## $ room_type <fct> Private room, Private room, Private room, Priva-
## $ accommodates <int> 2, 2, 2, 2, 5, 4, 5, 2, 2, 2, 4, 5, 1, 2, ~
## $ bathrooms <dbl> 1.0, 1.0, 1.0, 1.0, 1.0, 1.5, 2.0, 1.0, 1.~
## $ bed_type <fct> Real Bed, Real Bed, Real Bed, Real Bed, Real Be-
## $ cancellation_policy <fct> moderate, flexible, strict, flexible, moderate, ~
## $ cleaning_fee <lgl> TRUE, TRUE, TRUE, TRUE, FALSE, TRUE, FALSE, TRU-
## $ city <fct> SF, NYC, NYC, NYC, NYC, LA, Chicago, NYC, NYC, ~
## $ host_has_profile_pic <fct> t, ~
## $ host_identity_verified <fct> t, f, t, f, t, t, f, t, t, t, t, t, ~
## $ host_response_rate <fct> , 100%, 100%, 88%, , 91%, 91%, 100%, , 10~
## $ instant_bookable <fct> f, f, t, f, f, f, f, f, t, f, f, f, f, t, ~
## $ last_review <fct> 7/5/17, 8/8/17, 9/26/17, , 7/21/16, 4/16/17, 5/~
## $ latitude <dbl> 37.78375, 40.80097, 40.71063, 40.58060, 40.8208~
## $ longitude <dbl> -122.45705, -73.94890, -73.99526, -73.94464, -7~
## $ number_of_reviews <int> 4, 8, 90, 0, 3, 64, 8, 53, 0, 12, 21, 4, 0, 2, ~
## $ review_scores_rating <int> 100, 97, 92, NA, 73, 100, 88, 97, NA, 92, 98, 9~
## $ zipcode <int> 94118, 10026, 10002, 11235, 10031, 90026, 60614~
## $ bedrooms <int> 1, 1, 1, 1, 2, 1, 3, 1, 1, 1, 1, 2, 1, 1, 1, ~
## $ beds <int> 1, 1, 1, 1, 2, 3, 4, 2, 1, 1, 1, 2, 4, 2, 1, ~
## $ price <int> 150, 89, 112, 100, 39, 178, 200, 150, 90, 90, 8~
```

7.4.3 Gimpse of Test Data

```
glimpse(Test)
## Rows: 1,000
## Columns: 22
## $ id <int> 15854065, 18569489, 18071076, 14355535, 1751375~
## $ property_type <fct> Apartment, Apartment, Apartment, Apartment, Hou-
## $ room_type <fct> Entire home/apt, Entire home/apt, Private room, ~
## $ accommodates <int> 2, 4, 15, 6, 2, 3, 2, 7, 1, 4, 10, 2, 3, 2, 3, ~
```

```

## $ bathrooms      <dbl> 1.0, 1.0, 2.0, 1.0, 1.5, 1.0, 4.0, 1.0, 1.-
## $ bed_type       <fct> Real Bed, Real Bed, Real Bed, Real Be-
## $ cancellation_policy <fct> flexible, moderate, strict, strict, strict, mod-
## $ cleaning_fee    <lgl> FALSE, TRUE, TRUE, TRUE, TRUE, FALSE, FA-
## $ city            <fct> NYC, LA, NYC, SF, DC, LA, NYC, LA, LA, ~
## $ host_has_profile_pic <fct> t, ~
## $ host_identity_verified <fct> f, t, f, t, t, t, t, t, t, t, f, t, f, f, ~
## $ host_response_rate <fct> , 100%, 67%, 100%, 100%, 100%, 100%, 100%, , ~
## $ instant_bookable   <fct> f, f, f, f, t, f, f, f, t, f, f, f, f, f, ~
## $ last_review        <fct> 6/3/16, 4/30/17, , , 5/6/17, 4/30/17, , , 12/20-
## $ latitude           <dbl> 40.79256, 34.02429, 40.75390, 37.73539, 38.8713-
## $ longitude          <dbl> -73.97596, -118.41038, -73.99416, -122.42337, --
## $ number_of_reviews   <int> 5, 22, 0, 0, 105, 56, 0, 0, 20, 5, 0, 221, 1, 2-
## $ review_scores_rating <int> 80, 96, NA, NA, 90, 98, NA, NA, 94, 100, NA, 97-
## $ zipcode            <int> 10025, 90034, 10018, 94110, 20024, 90814, 10065-
## $ bedrooms           <int> 0, 1, 1, 2, 1, 1, 3, 1, 1, 1, 1, 1, 1, 1, 1, ~
## $ beds               <int> 1, 2, 2, 4, 1, 2, 1, 4, 1, 2, 6, 1, 1, 1, 2, 1, ~
## $ price              <lgl> NA, ~

```

7.4.4 Distribution of Prices

```
ggplot(data=Train, aes(x=price)) + geom_histogram(fill="blue", color="white") +
  ylab("frequency") + xlab("Price") + ggtitle("Distribution of Prices")
```

7.4.5 Prices by City

```
ggplot(data=Train, aes(y=price, x=city)) + geom_violin(aes(fill=city)) + geom_boxplot(width=0.2)
```

7.4.6 Price by City

```
T1 <- Train %>% group_by(city) %>% summarize(Mean_Price = mean(price),
                                                SD_Price = sd(price),
                                                N = n())
kable(T1, caption="Average House Price by City")
```

7.4.7 Price by Bedrooms

```
ggplot(data=Train, aes(x=bedrooms, y=price)) + geom_point() + stat_smooth(method="lm")
```

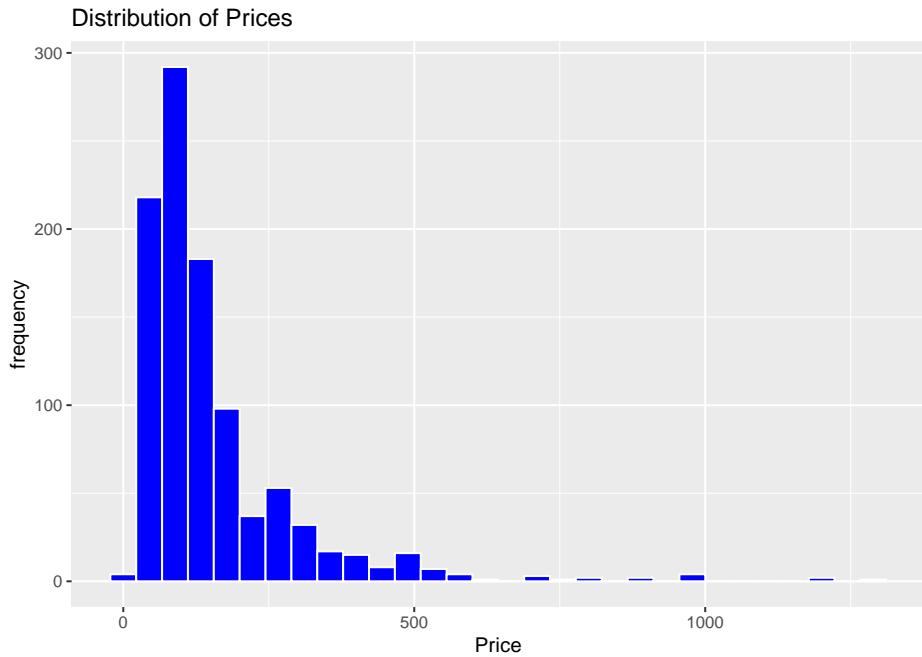


Figure 7.1: Prices

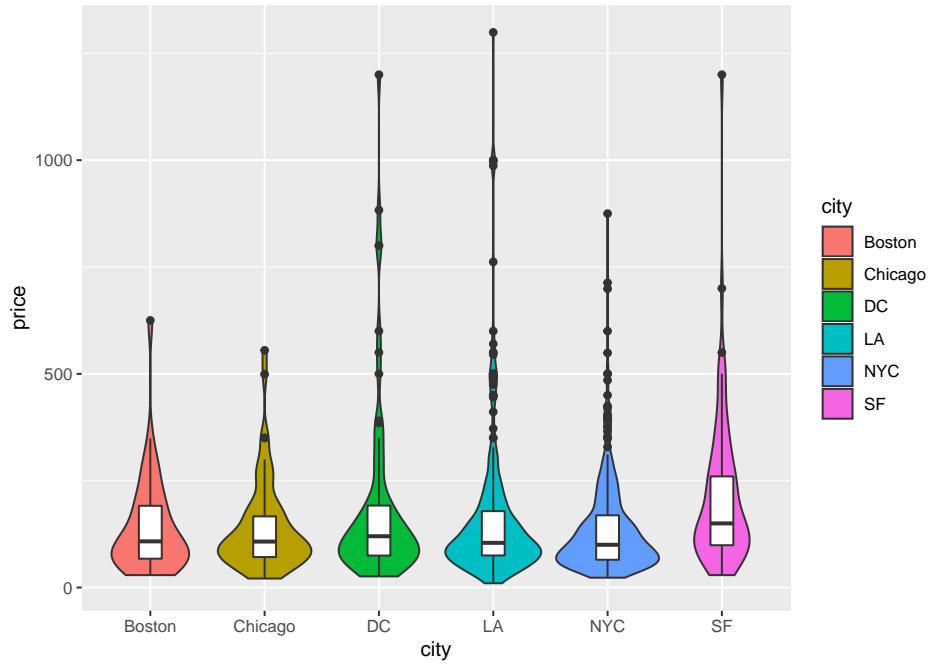


Figure 7.2: Price Distribution by City

Table 7.1: Average House Price by City

city	Mean_Price	SD_Price	N
Boston	141.5263	114.5935	38
Chicago	140.4677	105.8382	62
DC	188.8421	212.6684	76
LA	161.2550	167.0404	302
NYC	137.3101	108.5510	445
SF	215.1299	178.8882	77

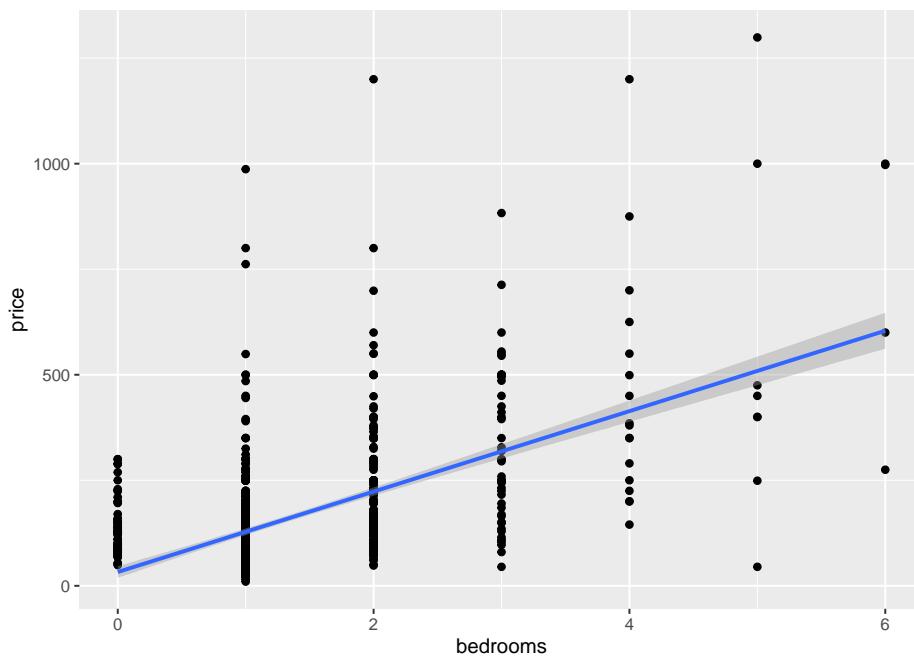


Figure 7.3: Price by Bedrooms

7.4.8 Correlation Plot

```
library(corrplot)
Train_num <- select_if(Train, is.numeric)
C <- cor(Train_num, use="pairwise.complete.obs")
corrplot(C)
```

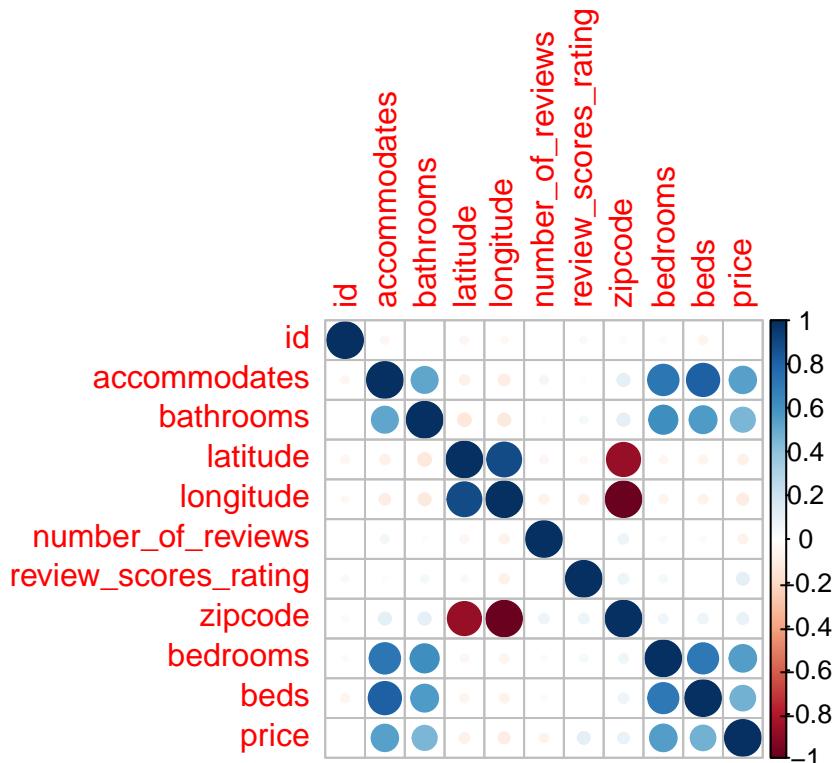


Figure 7.4: Correlation Plot for Quantitative Variables

7.4.9 Price by Room Type

```
T2 <- Train %>% group_by(room_type) %>% summarize(Mean_Price = mean(price),
                                                       SD_Price = sd(price),
                                                       N = n())
kable(T2, caption="Average Price by Room Type")
```

Table 7.2: Average Price by Room Type

room_type	Mean_Price	SD_Price	N
Entire home/apt	217.97154	167.52817	527
Private room	85.11833	49.56877	431
Shared room	77.35714	147.62111	42

Table 7.3: Average House Price by Cancellation Policy and Cleaning Fee

cancellation_policy	cleaning_fee	Mean_Price	SD_Price	N
flexible	FALSE	146.6939	164.42209	147
flexible	TRUE	130.7576	96.96386	165
moderate	FALSE	144.0784	185.30697	51
moderate	TRUE	151.8479	125.28516	217
strict	FALSE	135.4531	132.56995	64
strict	TRUE	175.2861	160.93712	353
super_strict_30	FALSE	279.0000	NA	1
super_strict_30	TRUE	272.0000	108.89444	2

7.4.10 Price by Cancellation Policy

```
T3 <- Train %>% group_by(cancellation_policy, cleaning_fee) %>% summarize(Mean_Price = mean(price),
  SD_Price = sd(price),
  N = n())
kable(T3, caption="Average House Price by Cancellation Policy and Cleaning Fee")
```

7.4.11 Feature Engineering

Now, we'll create new variables, or modify existing variables. Include description of each variable you change and create, and relevant table or graph.

We convert the host response rate variable to numeric.

```
Train$host_response_rate <- str_remove(Train$host_response_rate, "%")
Test$host_response_rate <- str_remove(Test$host_response_rate, "%")
Train$host_response_rate <- as.numeric(as.character(Train$host_response_rate))
Test$host_response_rate <- as.numeric(as.character(Test$host_response_rate))
```

7.4.12 Price by Response Rate

```
ggplot(data=Train, aes(x=host_response_rate, y=price)) + geom_point() +
  stat_smooth(method="lm") + xlab("Host Response Rate") + ggtitle("Price by Host Response Rate")
```

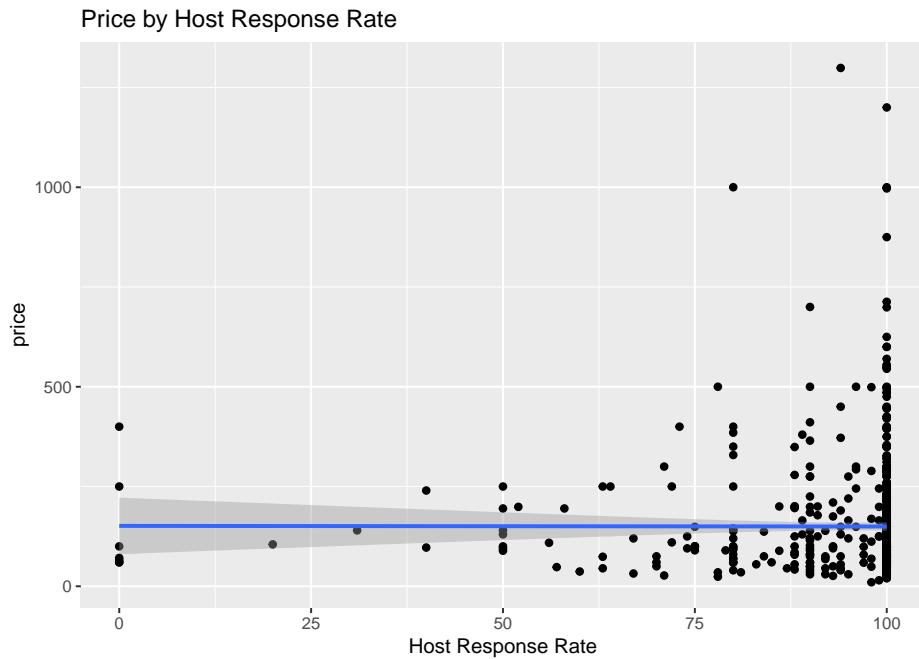


Figure 7.5: Price by Host Response Rate

7.4.13 More Variable Engineering

Our goal is to create new variables that might help up better predict price, than the ones originally contained in the dataset.

1. group together infrequent property types into a category called “other”. We keep the four most frequent categories: apartment, house, townhouse, and condominium.
2. create a yes/no variable for whether or not there was a review in 2017.
3. create a yes/no variable for whether or not there was an online review.
4. modify the `host_has_profile_pic` and `host_identity_verified` variables to group missing values and false's together.
5. create a variable to tell whether the last review was made on a weekend.

Table 7.4: Average Price by Recent Review

has_review	Mean_Price	SD_Price	N
FALSE	175.7309	182.5895	223
TRUE	148.8005	132.5472	777

7.4.14 Code for Variable Engineering

```
Train <- Train %>% mutate(property_type = fct_lump(property_type, n=4),
                           has_review = !is.na(review_scores_rating),
                           host_has_profile_pic = host_has_profile_pic=="t",
                           host_identity_verified = host_identity_verified == "t",
                           weekend = weekdays(as.Date(last_review)) %in% c("Saturday"))
Test <- Test %>% mutate(property_type = fct_lump(property_type, n=4),
                           has_review = !is.na(review_scores_rating),
                           host_has_profile_pic = host_has_profile_pic=="t",
                           host_identity_verified = host_identity_verified == "t",
                           weekend = weekdays(as.Date(last_review)) %in% c("Saturday"))
```

7.4.15 Removing Unneeded Variable

Since we've used the `last_review` variable to create a variable, we'll get rid of the original one.

```
#Train <- Train %>% select(-c(last_review))
#Test <- Test %>% select(-c(last_review))
```

7.4.16 Price by Property Type

```
ggplot(data=Train, aes(y=price, x=property_type)) + geom_violin( aes(fill=property_type)) +
  geom_boxplot(width=0.2)
```

7.4.17 Price by Review

```
T4 <- Train %>% group_by(has_review) %>% summarize(Mean_Price = mean(price),
                                                       SD_Price = sd(price),
                                                       N = n())
kable(T4, caption="Average Price by Recent Review")
```

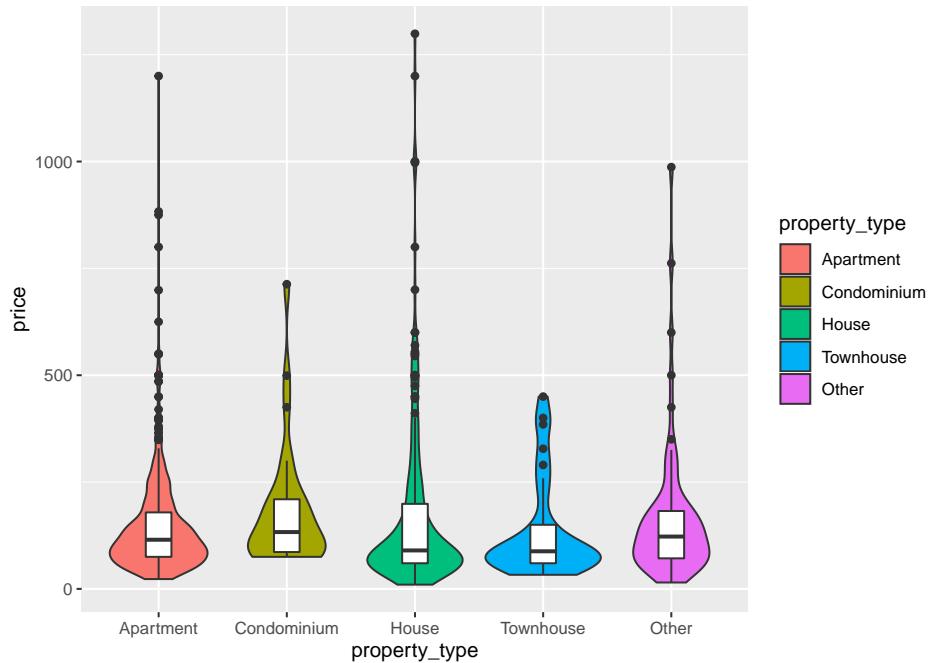


Figure 7.6: Price Distribution by Property Type

Table 7.5: Average Price by Day of Review

weekend	Mean_Price	SD_Price	N
FALSE	157.0287	146.7895	907
TRUE	133.1290	131.4779	93

7.4.18 Price by Weekend Review

```
T5 <- Train %>% group_by(weekend) %>% summarize(Mean_Price = mean(price),
                                                    SD_Price = sd(price),
                                                    N = n())
kable(T5, caption="Average Price by Day of Review")
```

7.5 Cross-Validation with `caret`

7.5.1 `caret` Package

We'll perform cross-validation, using the R package `caret`. It stands for Classification And REgression Training.

```
library(caret)
```

We'll perform 5 repeats of 5-fold cross validation.

```
control <- trainControl(method="repeatedcv", number=5, repeats=5 )
```

7.5.2 Impute Missing Values

We cannot fit a model when there are missing values in the data. We'll fill missing values by replacing them with the median value for that variable.

```
preProcValues <- preprocess(Train %>% select(-c(price)), method = c("medianImpute"))
Train <- predict(preProcValues, Train)
Test <- predict(preProcValues, Test)
```

7.5.3 Models to Compare

We consider 8 models:

1. simple linear regression model using only bedrooms as explanatory variable.
2. multiple regression model with the four quantitative explanatory variables most highly correlated with price.
3. same variables as in (2), with interactions included.
4. multiple regression model with bedrooms, and two categorical variables: room type, and city.
5. same model as in (4), with interactions included.
6. multiple regression model with combination of categorical and quantitative variables mentioned so far.
7. model including almost all variables, and leaving out only those that we wouldn't expect to have much relationship with price (latitude, longitude, id, zipcode, weekday, weekend)

8. model including all variables in the dataset

7.5.4 Models in R

```

set.seed(11082020)
model1 <- train(data=Train, price ~ bedrooms, method="lm", trControl=control)
set.seed(11082020)
model2 <- train(data=Train, price ~ bedrooms + accommodates + bathrooms + beds, method="lm",
set.seed(11082020)
model3 <- train(data=Train, price ~ bedrooms * accommodates * bathrooms * beds, method="lm",
set.seed(11082020)
model4 <- train(data=Train, price ~ bedrooms + city + room_type , method="lm", trControl=control)
set.seed(11082020)
model5 <- train(data=Train, price ~ bedrooms * city * room_type , method="lm", trControl=control)
set.seed(11082020)
model6 <- train(data=Train, price ~ bedrooms + accommodates + bathrooms + beds + city +
cancellation_policy + cleaning_fee , method="lm", trControl=control)
set.seed(11082020)
#exclude latitude, longitude, id, zipcode, weekday, weekend
model7 <- train(data=Train, price ~ property_type + bed_type + host_has_profile_pic +
accommodates + bathrooms + beds + city +room_type + cancellation_policy +
cleaning_fee + host_identity_verified + host_response_rate + instant +
number_of_reviews + review_scores_rating+ has_review, method="lm",
set.seed(11082020)
model8 <- train(data=Train, price ~ ., method="lm", trControl=control)

```

7.5.5 Record RMSPE for Each Model

```

r1 <- model1$results$RMSE
r2 <- model2$results$RMSE
r3 <- model3$results$RMSE
r4 <- model4$results$RMSE
r5 <- model5$results$RMSE
r6 <- model6$results$RMSE
r7 <- model7$results$RMSE
r8 <- model8$results$RMSE
Model <- 1:8
RMSPE <- c(r1, r2, r3, r4, r5, r6, r7, r8)
Res <- data.frame(Model, RMSPE)

```

7.5.6 MSPE's by Model

Cross-Validation Results

```
kable(Res)
```

Model	RMSPE
1	121.1541
2	117.1564
3	118.8492
4	112.2782
5	112.8955
6	109.9661
7	111.6570
8	139.5865

7.5.7 Models for log(price)

We also consider predicting log(price), using the same 8 models.

```
set.seed(11082020)
model1 <- train(data=Train, log(price) ~ bedrooms, method="lm", trControl=control)
set.seed(11082020)
model2 <- train(data=Train, log(price) ~ bedrooms + accommodates + bathrooms + beds,
                 method="lm", trControl=control)
set.seed(11082020)
model3 <- train(data=Train, log(price) ~ bedrooms * accommodates * bathrooms * beds,
                 method="lm", trControl=control)
set.seed(11082020)
model4 <- train(data=Train, log(price) ~ bedrooms + city + room_type ,
                 method="lm", trControl=control)
set.seed(11082020)
model5 <- train(data=Train, log(price) ~ bedrooms * city * room_type ,
                 method="lm", trControl=control)
set.seed(11082020)
model6 <- train(data=Train, log(price) ~ bedrooms + accommodates + bathrooms + beds +
                 city +room_type + cancellation_policy + cleaning_fee , method="lm", trControl=
set.seed(11082020)
#exclude latitude, longitude, id, zipcode, weekday, weekend
model7 <- train(data=Train, log(price) ~ property_type + bed_type + host_has_profile_pic +
                 bedrooms + accommodates + bathrooms + beds + city +room_type +
                 cancellation_policy + cleaning_fee + host_identity_verified +
                 host_response_rate + instant_bookable + number_of_reviews +
                 review_scores_rating + has_review, method="lm", trControl=control)
set.seed(11082020)
model8 <- train(data=Train, log(price) ~ ., method="lm", trControl=control)
```

7.5.8 Record RMSPE for log Models

```
r1 <- model1$results$RMSE
r2 <- model2$results$RMSE
r3 <- model3$results$RMSE
r4 <- model4$results$RMSE
r5 <- model5$results$RMSE
r6 <- model6$results$RMSE
r7 <- model7$results$RMSE
r8 <- model8$results$RMSE
Model <- 1:8
RMSPE <- c(r1, r2, r3, r4, r5, r6, r7, r8)
Res <- data.frame(Model, RMSPE)
```

7.5.9 RMSPE for Log Models

Cross Validation Results for Log Model

```
kable(Res)
```

Model	RMSPE
1	0.6361369
2	0.5961026
3	0.5863683
4	0.5088273
5	0.5211533
6	0.4996018
7	0.5080958
8	0.7377079

7.5.10 CV Results

We see that model 6 was best at predicting price as well as predicting $\log(\text{price})$. We can't compare these directly using the R output from `caret` because RMSPE is computed on different scales.

Instead, we'll convert the predictions for $\log(\text{price})$ back to price, and calculate RMSPE ourselves for these two models. We partition the data into a training set, containing 80% of the data, and a test set, containing the remaining 20%, and repeat this procedure 10 times. This is not true cross-validation, since we aren't dividing into distinct folds, and withholding each fold once, but it has the same effect of evaluating the model on data not used to train it.

7.5.11 Simulation for Comparing Models

```

set.seed(11082020)
RMSPE1 <- rep(NA, 10)
RMSPE2 <- rep(NA, 10)
for(i in 1:10){
  samp <- sample(1:1000, 200)
  Train2 <- Train[-samp, ]
  Validation <- Train[samp, ]
  M1 <- lm(data=Train2, price ~ bedrooms + accommodates + bathrooms + beds +
            city +room_type + cancellation_policy + cleaning_fee)
  M2 <- lm(data=Train2, log(price) ~ bedrooms + accommodates + bathrooms + beds +
            city +room_type + cancellation_policy + cleaning_fee)
  pred1 <- predict(M1, newdata=Validation)
  pred2 <- exp(predict(M2, newdata=Validation))
  RMSPE1[i] <- sqrt(mean((Validation$price - pred1)^2))
  RMSPE2[i] <- sqrt(mean((Validation$price - pred2)^2))
}

```

7.5.12 Comparison of Results

```

mean(RMSPE1)

## [1] 112.0109

mean(RMSPE2)

## [1] 113.7282

```

We see that we do slightly better when we do not use the log transform.

7.5.13 Fit Model to Full Training Data Set

```

M <- lm(data=Train, price ~ bedrooms + accommodates + bathrooms + beds +
         city +room_type + cancellation_policy + cleaning_fee)

```

7.5.14 Check of Model Assumptions

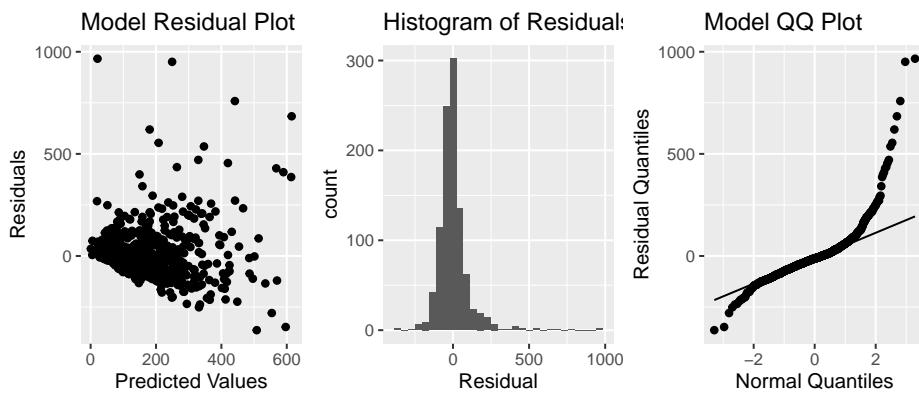
We'll create residual plots for this model.

```

library(gridExtra)
P1 <- ggplot(data=data.frame(M$residuals), aes(y=M$residuals, x=M$fitted.values)) + geom_point()
  ggttitle("Model Residual Plot") + xlab("Predicted Values") + ylab("Residuals")

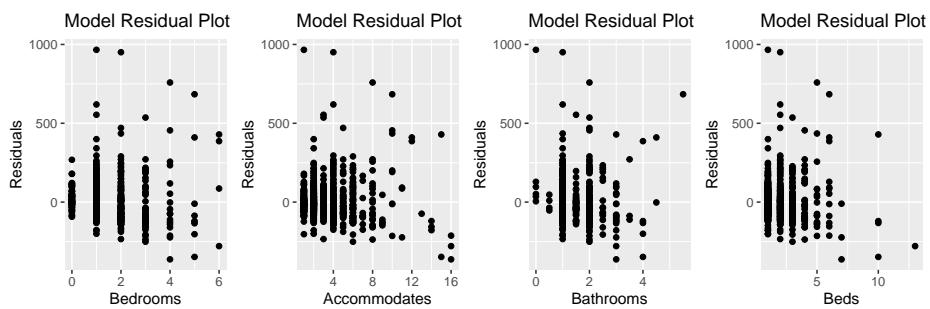
```

```
P2 <- ggplot(data=data.frame(M$residuals), aes(x=M$residuals)) + geom_histogram() +
  ggtitle("Histogram of Residuals") + xlab("Residual")
P3 <- ggplot(data=data.frame(M$residuals), aes(sample = M$residuals)) + stat_qq() +
  stat_qq_line() + xlab("Normal Quantiles") + ylab("Residual Quantiles") + ggtitle("Model QQ Plot")
grid.arrange(P1, P2, P3, ncol=3)
```



7.5.15 Residul by Predictor Plots

```
P1 <- ggplot(data=data.frame(M$residuals), aes(y=M$residuals, x=M$model$bedrooms)) + geom_point() +
  ggtitle("Model Residual Plot") + xlab("Bedrooms") + ylab("Residuals")
P2 <- ggplot(data=data.frame(M$residuals), aes(y=M$residuals, x=M$model$accommodates)) +
  ggtitle("Model Residual Plot") + xlab("Accommodates") + ylab("Residuals")
P3 <- ggplot(data=data.frame(M$residuals), aes(y=M$residuals, x=M$model$bathrooms)) +
  ggtitle("Model Residual Plot") + xlab("Bathrooms") + ylab("Residuals")
P4 <- ggplot(data=data.frame(M$residuals), aes(y=M$residuals, x=M$model$beds)) + geom_point() +
  ggtitle("Model Residual Plot") + xlab("Beds") + ylab("Residuals")
grid.arrange(P1, P2, P3, P4, ncol=4)
```



7.5.16 Predictions on Test Data

```
M <- lm(data=Train, log(price) ~ bedrooms + accommodates + bathrooms + beds +
         city +room_type + cancellation_policy + cleaning_fee)
Predictions <- predict(M, newdata=Test)

## Error in model.frame.default(Terms, newdata, na.action = na.action, xlev = object$xlevels): fa
```

7.5.17 Modification of Test Data

We'll change the super_strict_60 entry to "super_strict_30", which seems like to closest match in the training data

```
Test$cancellation_policy <- recode(Test$cancellation_policy, super_strict_60="super_strict_30")
```

Now, we make the predictions on the new data.

7.5.18 Predictions on Test Data

```
M <- lm(data=Train, price ~ bedrooms + accommodates + bathrooms + beds +
         city +room_type + cancellation_policy + cleaning_fee)
Predictions <- predict(M, newdata=Test)
```

7.5.19 Check Predictions

```
head(Predictions)

##      1     2     3     4     5     6
## 140.8920 155.7997 263.7413 279.2066 106.4915 164.2310
summary(Predictions)

##    Min. 1st Qu. Median    Mean 3rd Qu.    Max.
## 13.36   83.46 141.99 157.70 203.21 828.16
```

7.5.20 Save Predictions

```
Test$price <- Predictions

write.csv(Test, file="Test_predictions.csv")
```


Chapter 8

Advanced Regression and Nonparametric Approaches

8.1 Penalized Regression: Ridge and Lasso

8.1.1 Regression on Air BnB Dataset - Some Exp. Vars.

```
M1 <- lm(data=Train, price ~ bedrooms + accommodates + bathrooms + beds +
           city + room_type + cancellation_policy + cleaning_fee)
coef(M1)

##                               (Intercept)                bedrooms
##                         58.140180             48.279110
##                   accommodates                bathrooms
##                         11.891802              49.027248
##                           beds                  cityChicago
##                         -7.272873            -14.859591
##                           cityDC                 cityLA
##                         31.798209               6.897891
##                           cityNYC                citySF
##                         17.213827              69.611515
##       room_typePrivate room    room_typeShared room
##                         -85.387718            -96.893873
##   cancellation_policymoderate cancellation_policystrict
##                         -19.243073             -16.066733
## cancellation_policysuper_strict_30      cleaning_feeTRUE
##                         88.824633             -20.323137
```

8.1.2 Regression on Air BnB Dataset - All Exp. Vars.

```
M2 <- lm(data=Train, price ~ .)
head(coef(M2))

##              (Intercept)          id property_typeCondominium
##      -15242.3300816057399 0.0000007952311      -10.0250181873609
##      property_typeHouse   property_typeTownhouse    property_typeOther
##      2.1010429305294     -9.1810124499946      28.0377312977006
```

8.1.3 Complexity in Model Coefficients

- We've thought about complexity in terms of the number of terms we include in a model, as well as whether we include quadratic terms and interactions
- We can also think about model complexity in terms of the coefficients b_1, \dots, b_p .
- Larger values of b_1, \dots, b_p are associated with more complex models. Smaller values of b_0, b_1, \dots, b_p are associated with less complex models. When $b_j = 0$, this mean variable j is not used in the model.

8.1.4 Penalized Regression

- We've seen that in ordinary least-squares regression, b_0, b_1, \dots, b_p are chosen in a way that minimizes

$$\sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n (y_i - (b_0 + b_1 x_{i1} + b_2 x_{i2} + \dots + b_p x_{ip}))^2$$

- When p is large, and we want to be careful of overfitting, a common approach is to add a “penalty term” to this function, to incentivize choosing low values of b_1, \dots, b_p .

Specifically, we minimize:

$$\sum_{i=1}^n (y_i - \hat{y}_i)^2 + \sum_{j=1}^p p(b_j) = \sum_{i=1}^n (y_i - (b_0 + b_1 x_{i1} + b_2 x_{i2} + \dots + b_p x_{ip}))^2 + \sum_{j=1}^p p(b_j)$$

, where p is a function that increases as b_j gets farther from 0.

8.1.5 Ridge Regression

In Ridge regression, we let $p_j(b_j) = b_j^2$.

Thus, b_0, b_1, \dots, b_p are chosen in a way that minimizes

$$\begin{aligned} & \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^p b_j^2 \\ &= \sum_{i=1}^n (y_i - (b_0 + b_1 x_{i1} + b_2 x_{i2} + \dots + b_p x_{ip}))^2 + \lambda \sum_{j=1}^p b_j^2 \end{aligned}$$

where λ is a pre-determined positive constant.

- Larger values of b_j typically help the model better fit the training data, thereby making the first term smaller, but also make the second term larger.
- The idea is to find optimal values of b_0, b_1, \dots, b_p that are large enough to allow the model to fit the data well, thus keeping the first term (RSS) small, while also keeping the penalty term small as well.

8.1.6 Simple Example

```
head(df)
```

```
##      x1        x2        x3 x4 x5     y
## 1 4.139206 2.4498870 4.834715  0  1 18.1
## 2 2.850317 -0.2666719 5.219334  2  1  1.8
## 3 9.869605 -0.8287838 7.676558  2  0  8.9
## 4 1.423671  0.4906483 7.779525  4  1  7.3
## 5 8.244600  0.2575340 6.912677  4  1 15.2
## 6 5.404841  1.0488419 8.405193  3  0 10.4
```

8.1.7 Standardizing

- It is important to standardize each explanatory variable (i.e. subtract the mean and divide by the standard deviation).
- This ensures each variable has mean 0 and standard deviation 1.
- Without standardizing the optimal choice of b_j 's would depend on scale, with variables with larger absolute measurements having more influence.
- We'll standardize the response variable too. Though this is not strictly necessary, it doesn't hurt. We can always transform back if necessary.

- Standardization is performed using the `scale` command in R.

8.1.8 Standardizing Dataframe

```
df <- data.frame(scale(df))
head(df)

##           x1          x2          x3          x4          x5          y
## 1 -0.45459759  1.83361148 -0.7135063 -1.5278568  0.58554 1.50064128
## 2 -0.88764549 -0.52845565 -0.5195898 -0.1909821  0.58554 -1.30170193
## 3  1.47073150 -1.01721597  0.7192874 -0.1909821 -1.46385 -0.08104937
## 4 -1.36697736  0.13003961  0.7712009  1.1458926  0.58554 -0.35612600
## 5  0.92475383 -0.07265499  0.3341557  1.1458926  0.58554  1.00206488
## 6 -0.02936316  0.61539287  1.0866488  0.4774553 -1.46385  0.17683498
```

8.1.9 Ordinary Least-Squares Model

```
M_OLS <- lm(data=df, y~.)
summary(M_OLS)

##
## Call:
## lm(formula = y ~ ., data = df)
##
## Residuals:
##      1       2       3       4       5       6       7 
## 0.01093 -0.15412  0.02664  0.11262 -0.04872 -0.02664  0.07929
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 0.0000000000000001327 0.0816209130227057295 0.000 1.0000  
## x1          0.8474116247443916894 0.1147677904948321836 7.384 0.0857 .
## x2          0.9136565883555646250 0.1417118779920441829 6.447 0.0980 .
## x3          0.5430015911020581454 0.3171076433134487060 1.712 0.3365  
## x4          -0.1626972125210692788 0.2460100351057598889 -0.661 0.6280  
## x5          0.5781111710181862096 0.2229383124515697534 2.593 0.2343  
## ---        
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2159 on 1 degrees of freedom
## Multiple R-squared: 0.9922, Adjusted R-squared: 0.9534 
## F-statistic: 25.53 on 5 and 1 DF, p-value: 0.1491
```

8.1.10 Ridge Regression Model ($\lambda = 1$)

```
M_Ridge1 <- lm.ridge(data=df, y~, lambda = 1)
head(M_Ridge1$coef)

##           x1           x2           x3           x4           x5
## 0.545775362 0.745085871 0.209240885 -0.002114925 0.231880867
```

8.1.11 Comparison of Coefficients

```
b_OLS <- round(coef(M_OLS),3)
b_Ridge1 <- round(c(0,M_Ridge1$coef),3)
data.frame(b_OLS, b_Ridge1)

##           b_OLS   b_Ridge1
## (Intercept) 0.000 0.000
## x1          0.847 0.546
## x2          0.914 0.745
## x3          0.543 0.209
## x4         -0.163 -0.002
## x5          0.578 0.232
```

8.1.12 Predictions and Residuals

y	Pred_OLS	Pred_Ridge1	OLS_Resid	Ridge1_Resid
1.5006413	1.4897102	1.1965629	0.0109311	0.3040784
-1.3017019	-1.1475867	-0.9189041	-0.1541152	-0.3827978
-0.0810494	-0.1076928	-0.1552730	0.0266434	0.0742236
-0.3561260	-0.4687439	-0.3828531	0.1126179	0.0267271
1.0020649	1.0507862	0.7062330	-0.0487214	0.2958318
0.1768350	0.2034784	0.3558117	-0.0266434	-0.1789767
-0.9406638	-1.0199514	-0.8015775	0.0792875	-0.1390863

- The ridge predictions tend to be closer to the mean y-value of 0.

8.1.13 Choosing b_j in OLS

In OLS, we choose b_0, b_1, \dots, b_p are chosen in a way that minimizes

$$\sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n (y_i - (b_0 + b_1x_{i1} + b_2x_{i2} + \dots + b_px_{ip}))^2$$

```
sum((y-Pred_OLS)^2)

## [1] 0.04663381
```

```
sum((y-Pred_Ridge1)^2)
```

```
## [1] 0.3841155
```

Not surprisingly the OLS model achieves smaller $\sum_{i=1}^n (y_i - \hat{y}_i)^2$. This has to be true, since the OLS coefficients are chosen to minimize this quantity.

8.1.14 Penalty Term

In Ridge regression, we penalize estimates that are far from 0.

```
sum(coef(M_OLS)[2:6]^2)
```

```
## [1] 2.208408
```

```
sum(coef(M_Ridge1)[2:6]^2)
```

```
## [1] 1.109008
```

8.1.15 Choosing b_j in Ridge Regression ($\lambda = 1$)

In ridge regression, we choose b_0, b_1, \dots, b_p are chosen in a way that minimizes

$$\begin{aligned} Q &= \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^p b_j^2 \\ &= \sum_{i=1}^n (y_i - (b_0 + b_1 x_{i1} + b_2 x_{i2} + \dots + b_p x_{ip}))^2 + \lambda \sum_{j=1}^p b_j^2 \end{aligned}$$

- Using the OLS coefficients, $\sum_{i=1}^n (y_i - \hat{y}_i)^2 = 0.04663381$, and $\lambda \sum_{j=1}^p b_j^2 = 2.208408$, thus $Q = 2.255042$.
- Using the Ridge coefficients, $\sum_{i=1}^n (y_i - \hat{y}_i)^2 = 0.3841155$, and $\lambda \sum_{j=1}^p b_j^2 = 1.109008$, thus $Q = 1.493124$.

It makes sense that Q is smaller, using the ridge coefficients. This must be true, since these coefficients were chosen to minimize Q .

8.1.16 Other values of λ

The value of λ is predetermined by the user. The larger the value of λ , the more heavily large b_j 's are penalized. A value of $\lambda = 0$ corresponds to ordinary least-squares.

$$\begin{aligned} Q &= \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^p b_j^2 \\ &= \sum_{i=1}^n (y_i - (b_0 + b_1 x_{i1} + b_2 x_{i2} + \dots + b_p x_{ip}))^2 + \lambda \sum_{j=1}^p b_j^2 \end{aligned}$$

8.1.17 Predictions using $\lambda = 0.5, \lambda = 1, \lambda = 2$

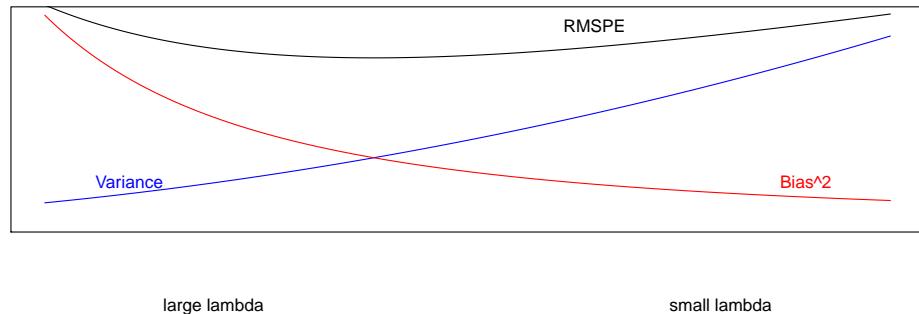
```
M_Ridge0.5 <- lm.ridge(data=df, y~, lambda = 0.5)
M_Ridge1 <- lm.ridge(data=df, y~, lambda = 1)
M_Ridge2 <- lm.ridge(data=df, y~, lambda = 2)
```

y	Pred_OLS_0	Pred_Ridge0.5	Pred_Ridge1	Pred_Ridge2
1.5006413	1.4897102	1.3131737	1.1965629	1.0266306
-1.3017019	-1.1475867	-1.0259091	-0.9189041	-0.7624705
-0.0810494	-0.1076928	-0.1619903	-0.1552730	-0.1338746
-0.3561260	-0.4687439	-0.4312111	-0.3828531	-0.3086787
1.0020649	1.0507862	0.8518971	0.7062330	0.5157487
0.1768350	0.2034784	0.3284327	0.3558117	0.3636789
-0.9406638	-1.0199514	-0.8743930	-0.8015775	-0.7010345

- The smaller the value of λ , the more complex the model. As λ gets large, the model becomes less complex, and predictions are pulled closer to $\bar{y} = 0$.

8.1.18 Optimal value of λ

- Small values of λ lead to more complex models, with larger $|b_j|$'s.
- As λ increases, $|b_j|$'s shrink toward 0. The model becomes less complex, thus bias increases, but variance decreases.
- We can use cross validation to determine the optimal value of λ



8.1.19 Ridge Regression on AirBnB Dataset

```
Train_sc <- Train %>% mutate_if(is.numeric, scale)

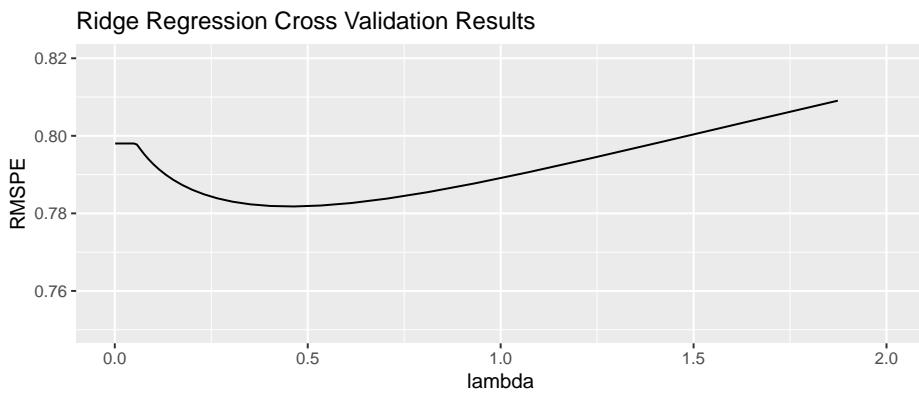
control = trainControl("repeatedcv", number = 5, repeats=5)
l_vals = 10^seq(-3, 3, length = 100)

set.seed(11162020)
AirBnB_ridge <- train(price ~., data = Train_sc, method = "glmnet",
                        tuneGrid=expand.grid(alpha=0, lambda=l_vals))

AirBnB_ridge$bestTune$lambda

## [1] 0.4641589
```

8.1.20 Ridge Regression MSPE by λ



8.1.21 Ridge Regression Coefficients for Optimal Model

```

OLS_AirBnB <- lm(data=Train_sc, price~.)
OLS_coef <- coef(OLS_AirBnB)
Ridge_coef <- coef(AirBnB_ridge$finalModel, AirBnB_ridge$bestTune$lambda) [,1]
head(data.frame(OLS_coef, Ridge_coef),10)

##                               OLS_coef     Ridge_coef
## (Intercept)            30.04775580  0.229367076
## id                     0.03375381  0.019037906
## property_typeCondominium -0.06888227 -0.053765129
## property_typeHouse       0.01443634 -0.003516517
## property_typeTownhouse   -0.06308308 -0.132078666
## property_typeOther        0.19264829  0.117498437
## room_typePrivate room    -0.56281326 -0.351720684
## room_typeShared room      -0.59937318 -0.361384788
## accommodates             0.25176666  0.143761719
## bathrooms                0.11843735  0.117465535

```

8.1.22 Lasso Regression

Lasso regression is very similar to ridge regression. We let $p_j(b_j) = |b_j|$.

Thus, b_0, b_1, \dots, b_p are chosen in a way that minimizes

$$\begin{aligned} & \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^p |b_j| \\ &= \sum_{i=1}^n (y_i - (b_0 + b_1 x_{i1} + b_2 x_{i2} + \dots + b_p x_{ip}))^2 + \lambda \sum_{j=1}^p |b_j| \end{aligned}$$

8.1.23 Lasso Regression on AirBnB Dataset

```

l_vals = 10^seq(-3, 3, length = 100)

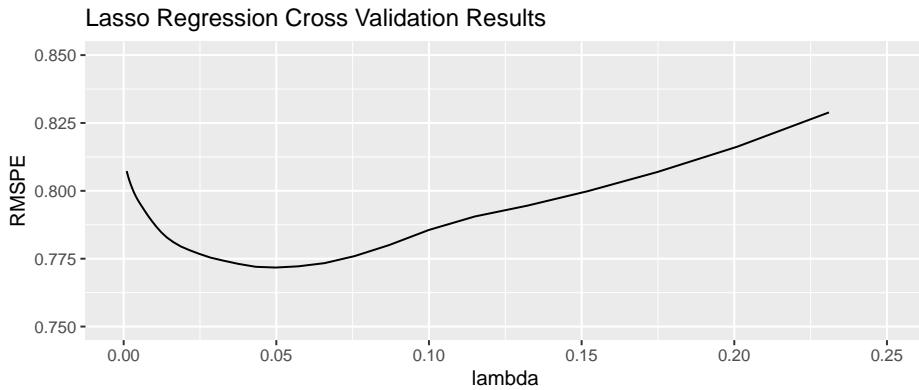
set.seed(11162020)
AirBnB_lasso <- train(price ~., data = Train_sc, method = "glmnet", trControl=control,
                      tuneGrid=expand.grid(alpha=1, lambda=l_vals))

AirBnB_lasso$bestTune$lambda

## [1] 0.04977024

```

8.1.24 Lasso Regression MSPE by λ



8.1.25 OLS, Ridge, Lasso Coefficients

```

OLS_AirBnB <- lm(data=Train_sc, price~.)
OLS_coef <- coef(OLS_AirBnB)
Lasso_coef <- coef(AirBnB_lasso$finalModel, AirBnB_lasso$bestTune$lambda)[,1]
head(data.frame(OLS_coef, Ridge_coef, Lasso_coef),10)

##                                     OLS_coef     Ridge_coef   Lasso_coef
## (Intercept)           30.04775580  0.229367076  0.3209095
## id                   0.03375381  0.019037906  0.0000000
## property_typeCondominium -0.06888227 -0.053765129  0.0000000
## property_typeHouse      0.01443634 -0.003516517  0.0000000
## property_typeTownhouse   -0.06308308 -0.132078666  0.0000000
## property_typeOther       0.19264829  0.117498437  0.0000000
## room_typePrivate room    -0.56281326 -0.351720684 -0.4584811
## room_typeShared room     -0.59937318 -0.361384788 -0.3930319
## accommodates            0.25176666  0.143761719  0.1409742
## bathrooms               0.11843735  0.117465535  0.1063063

```

8.1.26 Comparing RMSPE for OLS, Lasso, Ridge

```

set.seed(11162020)
AirBnB_OLS <- train(data=Train_sc, price ~ ., method="lm", trControl=control)

min(AirBnB_OLS$results$RMSE)

## [1] 0.9898462

```

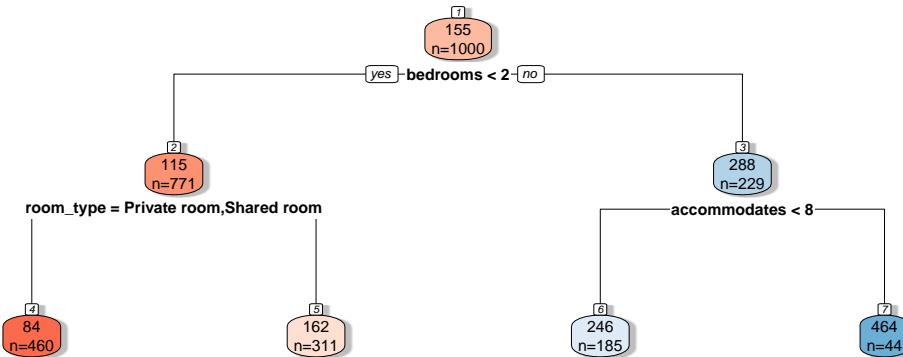
```
min(AirBnB_ridge$results$RMSE)
## [1] 0.7817871
min(AirBnB_lasso$results$RMSE)
## [1] 0.7717481
```

8.2 Decision Trees

8.2.1 Basics of Decision Trees

- A decision tree is a flexible alternative to a regression model. It is said to be **nonparametric** because it does not involve parameters like $\beta_0, \beta_1, \dots, \beta_p$.
- A tree makes no assumption about the nature of the relationship between the response and explanatory variables, and instead allows us to learn this relationship from the data.
- A tree makes prediction by repeatedly grouping together like observations in the training data.
- We can make predictions for a new case, by tracing it through the tree, and averaging responses of training cases in the same terminal node.

8.2.2 Decision Tree Example



- The predicted price of an Airbnb with 3 bedrooms that accommodates 5 is \$246.
- The predicted price of an Airbnb with 1 bedroom that is a full house or apartment (rather than private or shared room) is \$162.

8.2.3 Partitioning in A Decision Tree

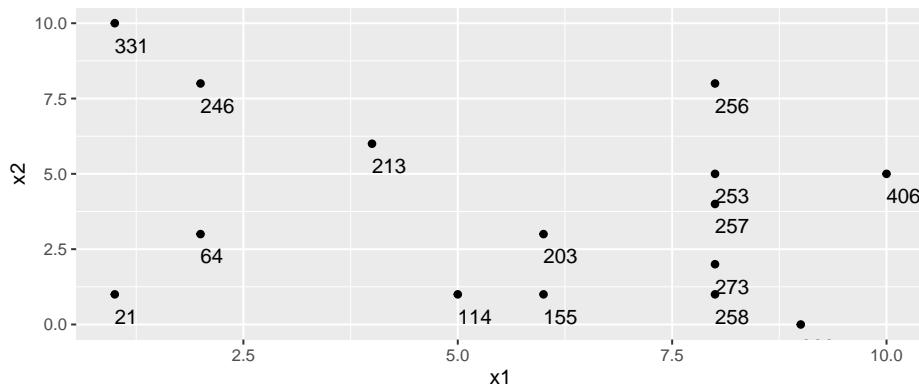
- For a quantitative response variable, data are split into two nodes so that responses in the same node are as similar as possible, while responses in the different nodes are as different as possible.
- Let L and R represent the left and right nodes from a possible split. Let n_L and n_R represent the number of observations in each node, and \bar{y}_L and \bar{y}_R represent the mean of the training data responses in each node.
- For each possible split, involving an explanatory variable, we calculate:

$$\sum_{i=1}^{n_L} (y_i - \bar{y}_L)^2 + \sum_{i=1}^{n_R} (y_i - \bar{y}_R)^2$$

- We choose the split that minimizes this quantity.

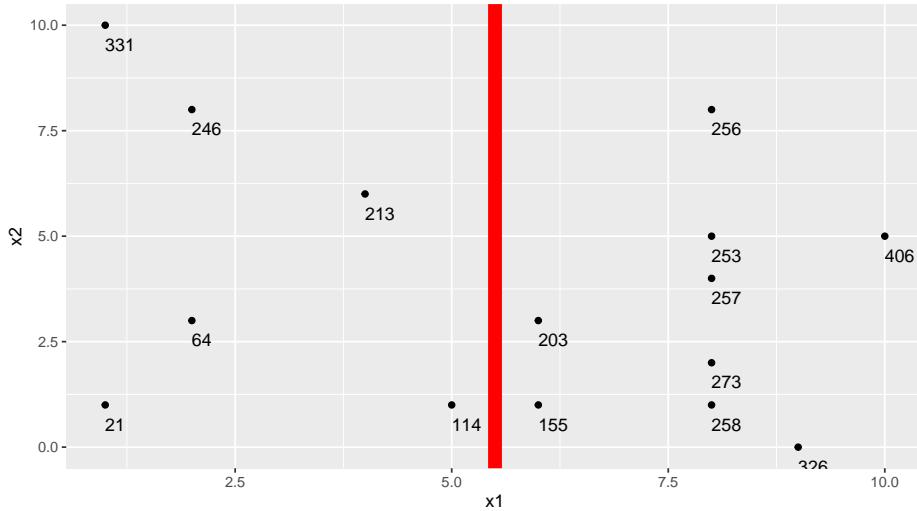
8.2.4 Partitioning Example

```
## [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
## x1     8     2     8     1     8     6     2     5     1     8     4     10     9     8
## x2     5     3     1     1     4     3     8     1    10     8     6     5     0     2
## y    253    64   258    21   257   203   246   114   331   256   213   406   326   273
## [,15]
## x1      6
## x2      1
## y      155
```



8.2.5 One Possible Split ($x_1 < 5.5$)

We could split the data into 2 groups depending on whether $x_1 < 5.5$.



8.2.6 One Possible Split ($x_1 < 5.5$) (cont.)

- $\bar{y}_L = (331 + 246 + 213 + 21 + 64 + 114)/6 \approx 164.84$
- $\bar{y}_R = (203 + 155 + 256 + 253 + 257 + 273 + 258 + 326 + 406)/9 \approx 265.22$

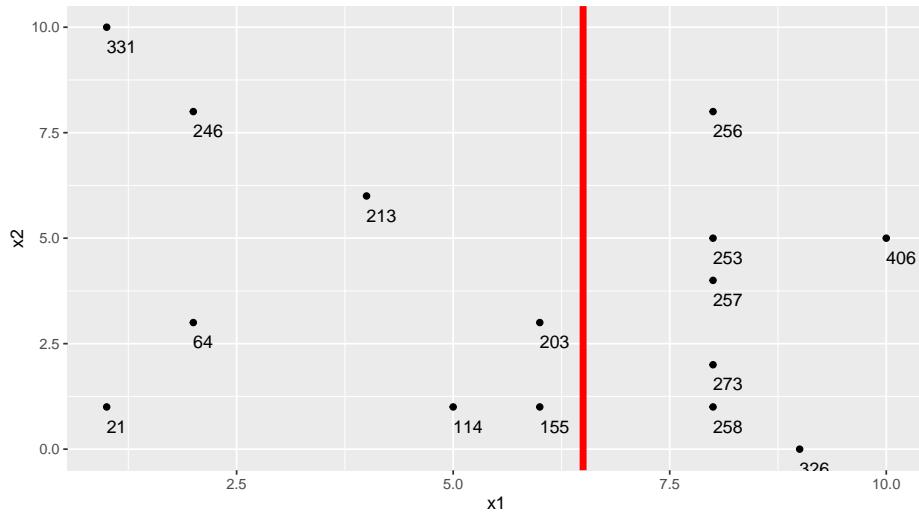
$$\begin{aligned} & \sum_{i=1}^{n_L} (y_i - \bar{y}_L)^2 \\ &= (331 - 164.83)^2 + (246 - 164.83)^2 + \dots + (114 - 164.83)^2 \\ &= 69958.83 \end{aligned}$$

$$\begin{aligned} & \sum_{i=1}^{n_R} (y_i - \bar{y}_R)^2 \\ &= (203 - 265.22)^2 + (155 - 265.22)^2 + \dots + (406 - 265.22)^2 \\ &= 39947.56 \end{aligned}$$

- $69958.83 + 39947.56 = 109906.4$

8.2.7 Second Possible Split ($x_1 < 6.5$)

We could split the data into 2 groups depending on whether $x_1 < 6.5$.



8.2.8 Second Possible Split ($x_1 < 6.5$) (cont.)

- $\bar{y}_L = (331 + 246 + 213 + 21 + 64 + 114 + 203 + 155)/8 \approx 168.375$
- $\bar{y}_R = (256 + 253 + 257 + 273 + 258 + 326 + 406)/7 \approx 289.857$

$$\begin{aligned} & \sum_{i=1}^{n_L} (y_i - \bar{y}_L)^2 \\ &= (331 - 168.375)^2 + (246 - 168.375)^2 + \dots + (203 - 168.375)^2 \\ &= 71411.88 \end{aligned}$$

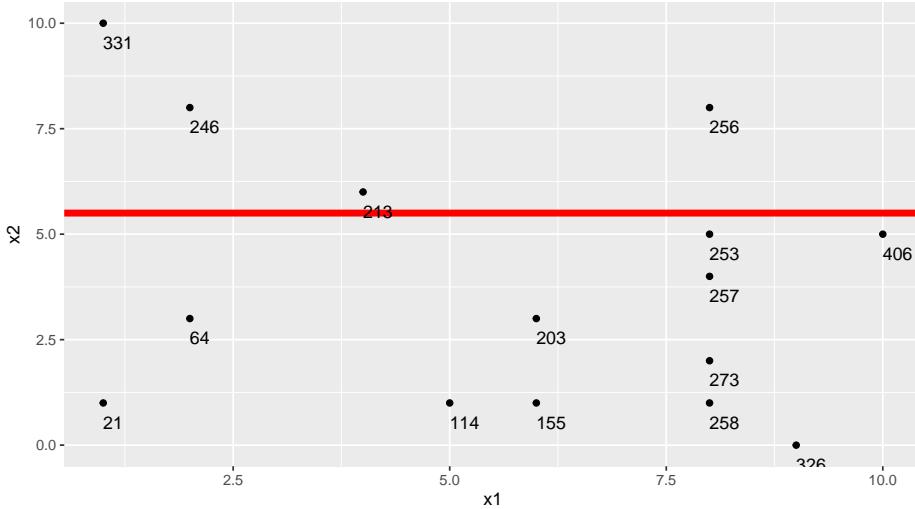
$$\begin{aligned} & \sum_{i=1}^{n_R} (y_i - \bar{y}_R)^2 \\ &= (256 - 289.857)^2 + (253 - 289.857)^2 + \dots + (406 - 289.857)^2 \\ &= 19678.86 \end{aligned}$$

- $71411.88 + 19678.86 = 91090.74$

The split at $x_1 < 6.5$ is better than $x_1 < 5.5$

8.2.9 Third Possible Split ($x_2 < 5.5$)

We could split the data into 2 groups depending on whether $x_2 < 5.5$.



8.2.10 Third Possible Split ($x_2 < 5.5$) (cont.)

- $\bar{y}_L = (331 + 246 + 213 + 256)/4 \approx 261.5$
- $\bar{y}_R = (21 + 64 + \dots + 406)/11 \approx 211.82$

$$\begin{aligned} & \sum_{i=1}^{n_L} (y_i - \bar{y}_L)^2 \\ &= (331 - 261.5)^2 + (246 - 261.5)^2 + (213 - 261.5)^2 + (256 - 261.5)^2 \\ &= 7453 \end{aligned}$$

$$\begin{aligned} & \sum_{i=1}^{n_R} (y_i - \bar{y}_R)^2 \\ &= (21 - 211.82)^2 + (64 - 211.82)^2 + \dots + (406 - 211.82)^2 \\ &= 131493.6 \end{aligned}$$

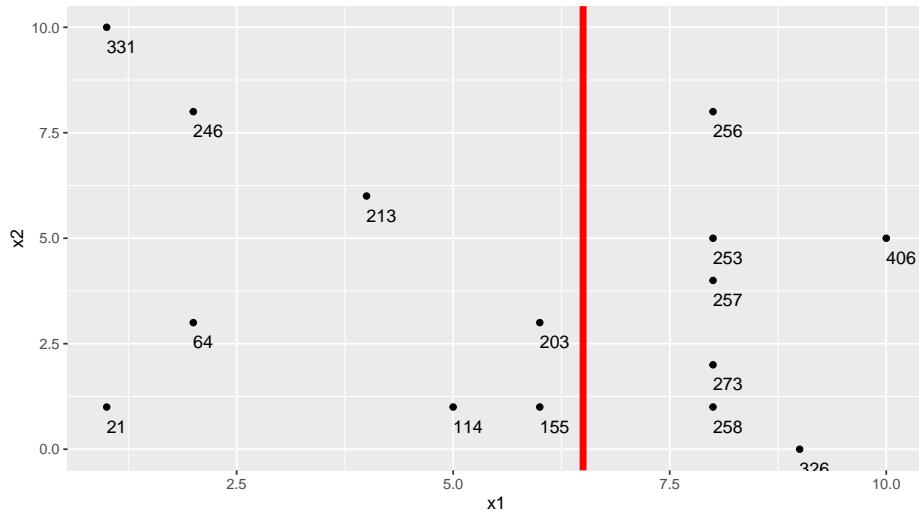
- $7453 + 131493.6 = 138946.6$

8.2.11 Comparison of Splits

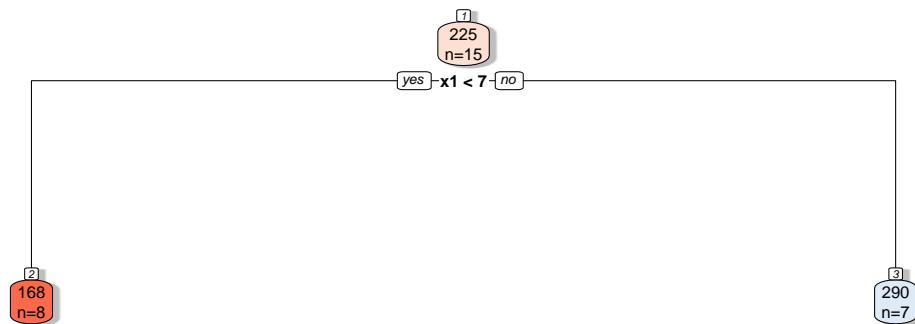
- Of the three split's we've calculated, $\sum_{i=1}^{n_L} (y_i - \bar{y}_L)^2 + \sum_{i=1}^{n_R} (y_i - \bar{y}_R)^2$ is minimized using $x_1 < 6.5$.

- In fact, if we calculate all possible splits over x_1 and x_2 , $\sum_{i=1}^{n_L} (y_i - \bar{y}_L)^2 + \sum_{i=1}^{n_R} (y_i - \bar{y}_R)^2$ is minimized by splitting on $x_1 < 6.5$

8.2.12 First Split

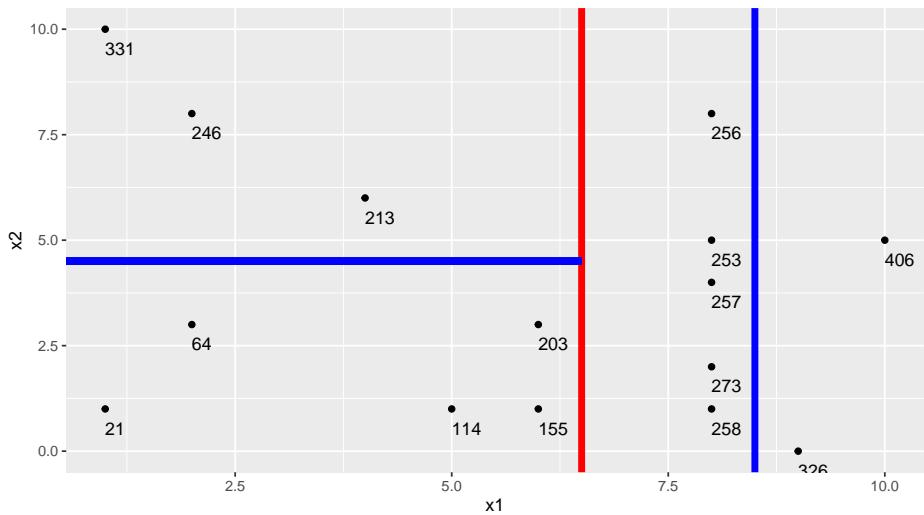


8.2.13 First Split in the Tree

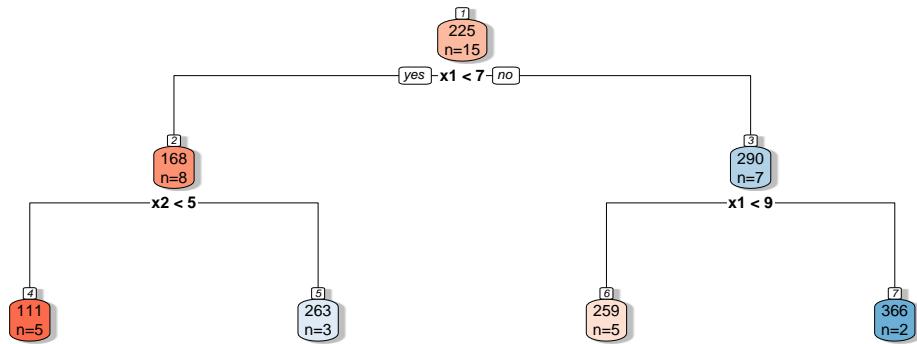


8.2.14 Next Splits

- Next, we find the best splits on the resulting two nodes. It turns out that the left node is best split on $x_2 < 4.5$, and the right node is best split on $x_1 < 8.5$.



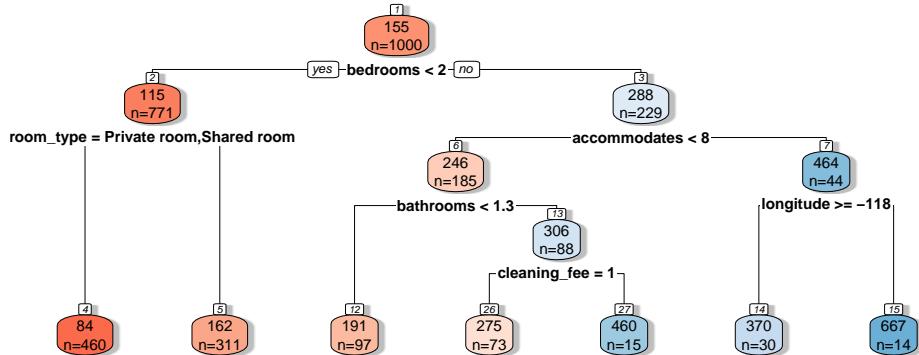
8.2.15 Next Splits in Tree



8.2.16 Recursive Partitioning

- Splitting continues until nodes reach a certain predetermined minimal size, or until change improvement in model fit drops below a predetermined value

8.2.17 Tree on AirBnB Data



8.2.18 Model Complexity in Trees

- The more we partition data into smaller nodes, the more complex the model becomes.
- As we continue to partition, bias decreases, as cases are grouped with those that are more similar to themselves. On the other hand, variance increases, as there are fewer cases in each node to be averaged, putting more weight on each individual observation.
- Splitting into too small of nodes can lead to drastic overfitting. In the extreme case, if we split all the way to nodes of size 1, we would get RMSE of 0 on the training data, but should certainly not expect RMSPE of 0 on the test data.
- The optimal depth of the tree, or minimal size for terminal nodes can be determined using cross-validation.
- the `rpart` package uses a complexity parameter `cp`, which determines how much a split must improve model fit in order to be made. Smaller values of `cp` are associated with more complex tree models.

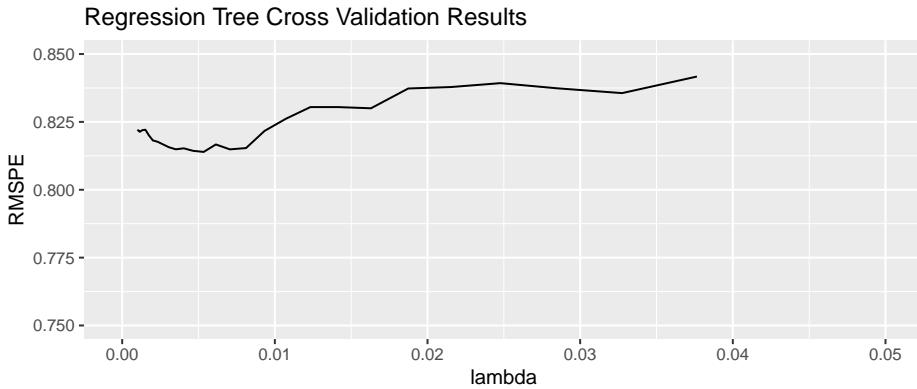
8.2.19 Cross-Validation on AirBnB Data

```

cp_vals = 10^seq(-3, 3, length = 100)
set.seed(11162020)
AirBnB_Tree <- train(data=Train_sc, price ~ ., method="rpart", trControl=control,
                      tuneGrid=expand.grid(cp=cp_vals))
AirBnB_Tree$bestTune

##          cp
  
```

```
## 13 0.005336699
```



8.2.20 Comparing OLS, Lasso, Ridge, and Tree

```
min(AirBnB_OLS$results$RMSE)
## [1] 0.9898462
min(AirBnB_ridge$results$RMSE)
## [1] 0.7817871
min(AirBnB_lasso$results$RMSE)
## [1] 0.7717481
min(AirBnB_Tree$results$RMSE)
## [1] 0.8139371
```

In this situation, the tree outperforms OLS, but does not do as well as lasso. The best model will vary depending on the nature of the data. We can use cross-validation to determine which model is likely to perform best in prediction.

8.2.21 Random Forest

- A popular extension of a decision tree is a random forest.
- A random forest consists of many (often $\sim 10,000$) trees. Predictions are made by averaging predictions from individual trees.
- In order to ensure the trees are different from each other:
 1. each tree is grown from a different bootstrap sample of the training data.

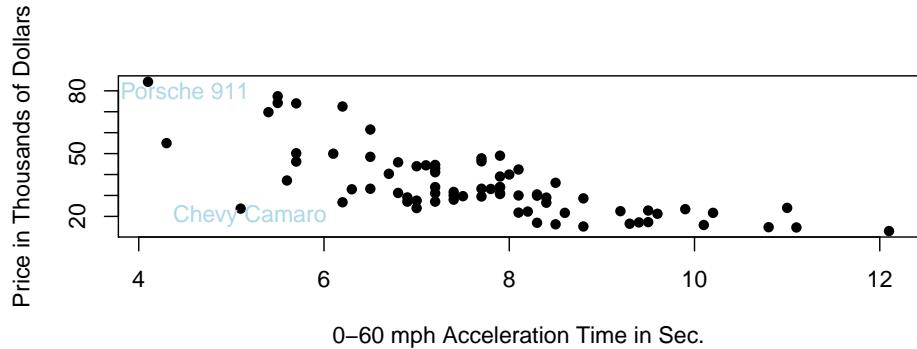
- 2. when deciding on a split, only a random subset of explanatory variables are considered.
- Growing deep trees ensures low bias. In a random forest, averaging across many deep trees decreases variance, while maintaining low bias.

8.3 Regression Splines

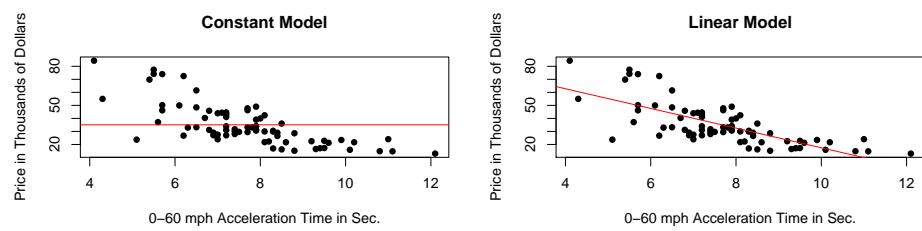
8.3.1 Regression Splines

- We've seen that we can use polynomial regression to capture nonlinear trends in data.
- A **regression spline** is a piecewise function of polynomials.
- Here we'll keep thing simple by focusing on a spline with a single explanatory variable. Splines can also be used for multivariate data.

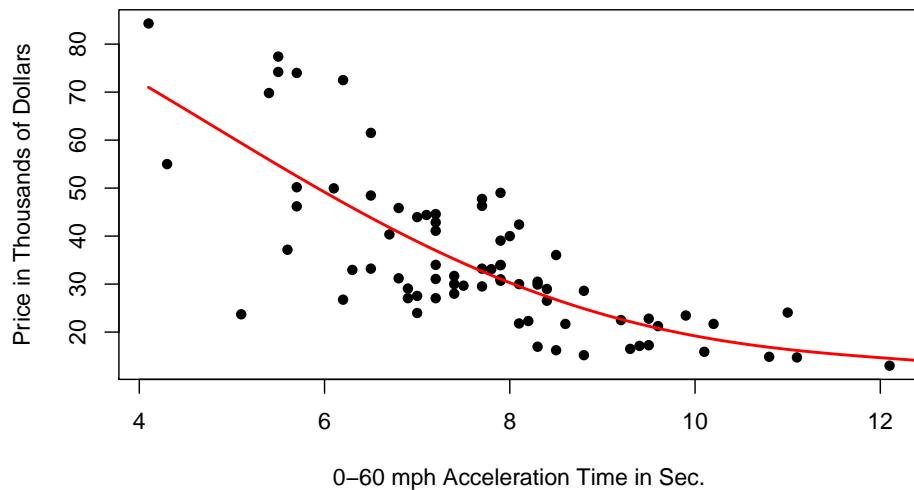
8.3.2 Price of 2015 New Cars



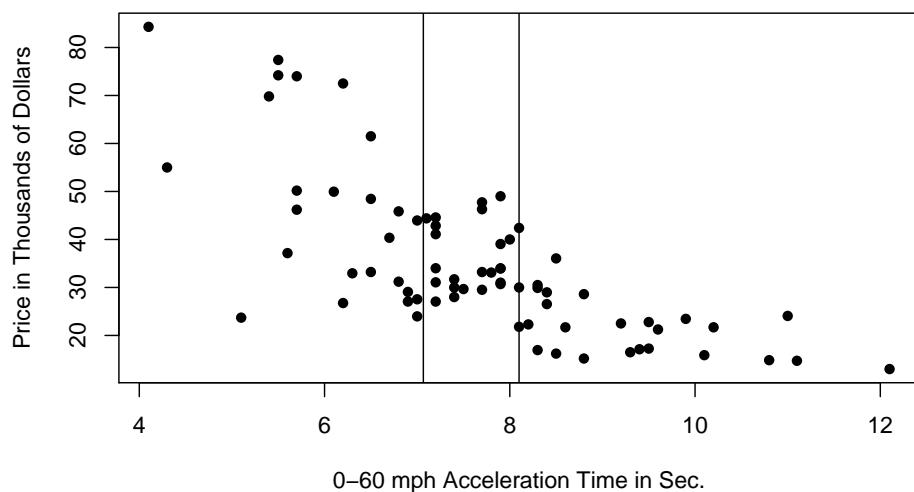
8.3.3 Two Models with High Bias



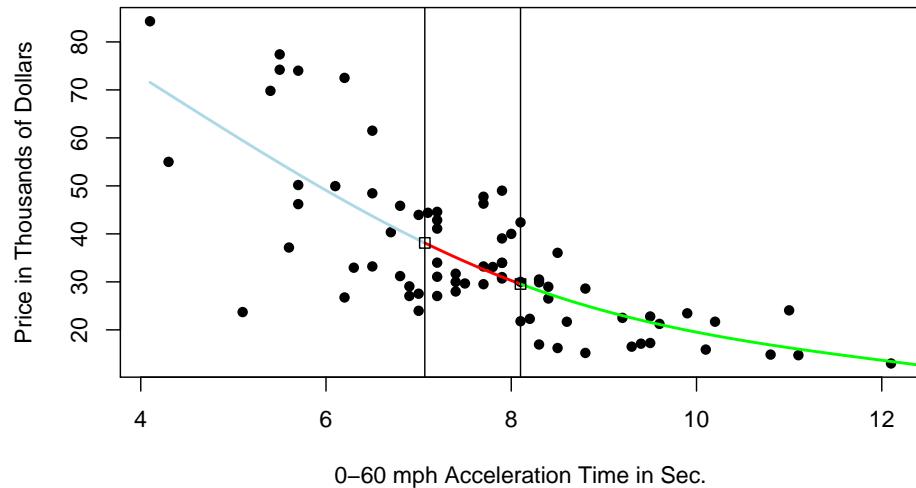
8.3.4 Cubic Model



8.3.5 Cubic Splines

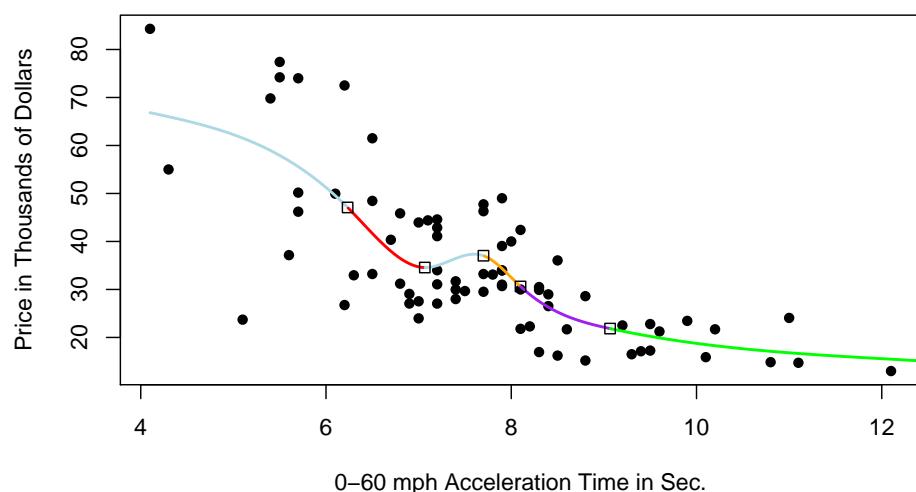


8.3.6 Cubic Splines

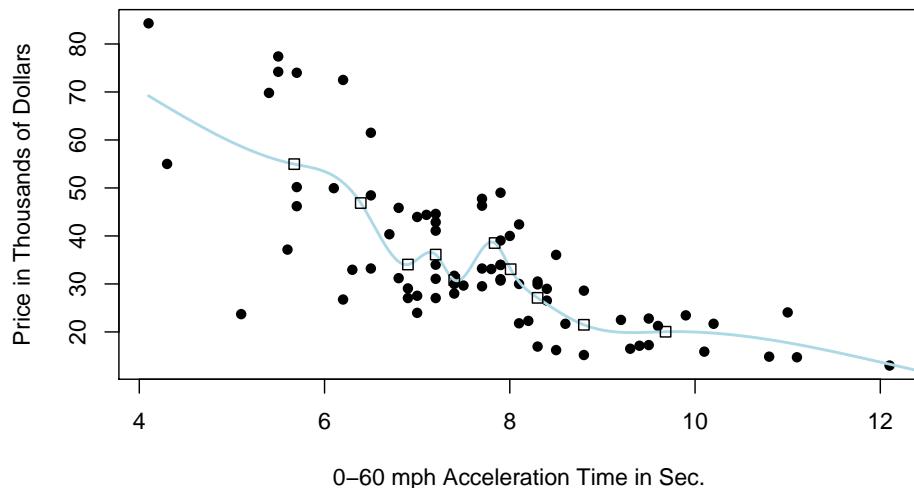


- region boundaries are called **knots**

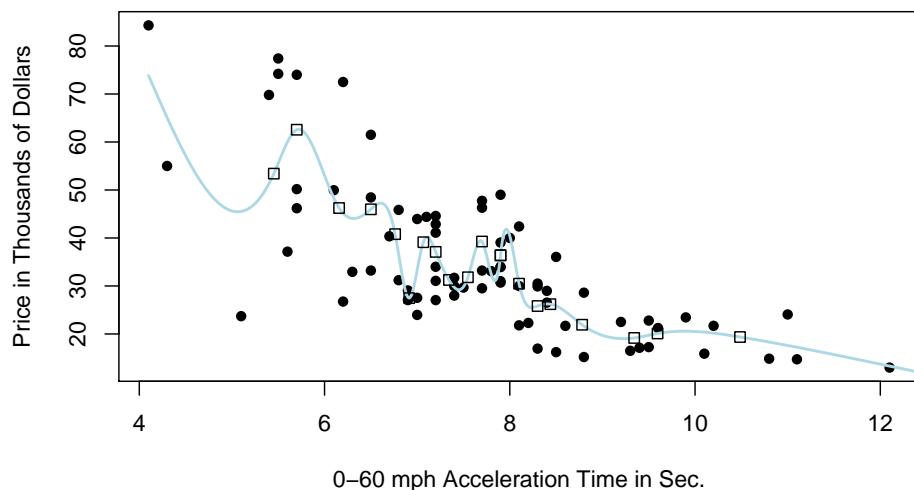
8.3.7 Cubic Spline with 5 Knots



8.3.8 Cubic Spline with 10 Knots

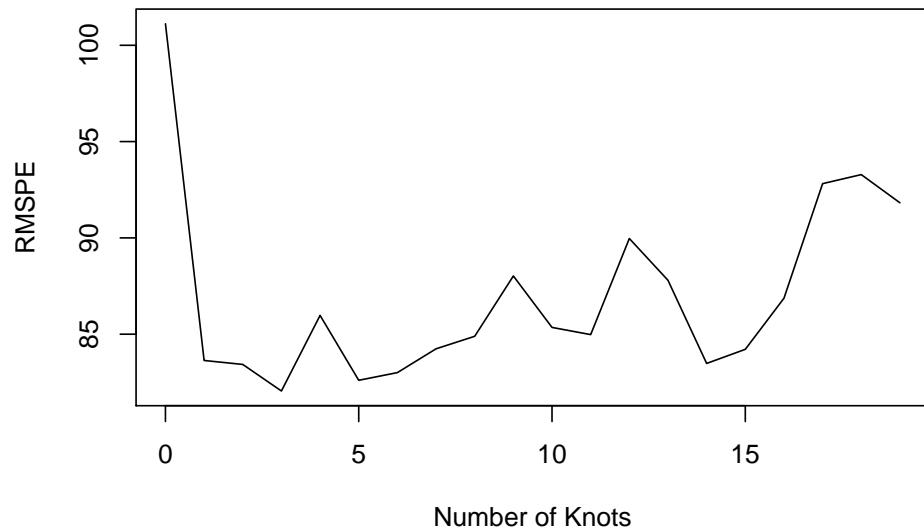


8.3.9 Cubic Spline with 20 Knots



8.3.10 Model Evaluation

- predicted price for 35 new 2015 cars not in original dataset
- calculated mean square prediction error (MSPE)



8.3.11 Implementation of Splines

Important Considerations:

- how many knots
- where to place knots
- degree of polynomial

The best choices for all of these will vary between datasets and can be assessed through cross-validation.

—>