

STAT 255 Notes

Andrew J. Sage

2024-08-26

Table of contents

Preface	4
1 Visualizing and Summarizing Data	6
1.1 Getting Started in R	6
1.2 Summary Statistics	11
1.3 Data Visualization	14
2 Introduction to Statistical Models	26
2.1 Fitting Models to Data	26
2.2 Variability Explained by a Model	37
2.3 Models with Interaction	50
2.4 Least Squares Estimation (LSE)	58
2.5 ANalysis Of VAriance	63
3 Simulation-Based Inference	73
3.1 Sampling Distributions	73
3.2 Confidence Intervals	85
3.3 Bootstrapping	96
3.4 More Bootstrap Examples	109
3.5 Hypothesis Testing	161
3.6 More Hypothesis Test Examples	176
3.7 Responsible Hypothesis Testing	194
4 Inference from Models	197
4.1 The Normal Error Regression Model	197
4.2 Standard Error and Confidence Intervals	209
4.3 t-tests	222
4.4 F-tests	231
4.5 lm Summary in R	236
4.6 Intervals for Expected Response	248
4.7 The Regression Effect	259
4.8 Responsible Statistical Inference	264
5 Building and Assessing Models	272
5.1 Checking Regression Assumptions	272
5.2 Transformations	283

5.3 Building Models for Interpretation: Confounding, Multicollinearity, and Polynomial Regression	292
6 Logistic Regression	324
6.1 Logistic Regression	324
6.2 Interpreting Coefficients in Logistic Regression	331
6.3 Multiple Logistic Regression	333
7 Predictive Modeling	337
7.1 Modeling for Prediction	337
7.2 Variance-Bias Tradeoff	348
7.3 Ridge Regression	355
7.4 Decision Trees	365
7.5 Regression Splines	377
7.6 Summary and Comparison	385
7.7 Assessing a Classifier's Performance	393
7.8 Receiver Operating Characteristic Curve	401
7.9 Ethical Considerations in Predictive Modeling	407

Preface

These notes serve as the primary textual resource for Stat 255: Statistics for Data Science at Lawrence University.

What is this course about?

Stat 255 provides an introduction to essential statistical tasks including modeling, inference, prediction, and computation. The course employs a modern approach, intended to equip students with skills needed for working with today's complex data. Traditional concepts, like interval estimation and hypothesis testing, are introduced through the lens of multivariate models and simulation. Data computation in R plays a central role throughout the course.

The course's overarching learning outcomes are:

1. Visualize and wrangle data using statistical software R.
2. Build and assess multivariate models to predict future outcomes.
3. Use statistics from samples to draw inferences about larger populations or processes.
4. Quantify uncertainty associated with estimates and predictions.
5. Explain the assumptions associated with statistical models, and evaluate whether these assumptions are reasonably satisfied in context.
6. Write reproducible analyses, using statistical software.
7. Make ethical decisions based on data.

More specific learning tasks, related to these outcomes are provided in each chapter.

Who is this course intended for?

This course is intended for students who are interested in learning statistical modeling and data computation skills that might prove useful in further courses, research, or career.

Stat 255 can serve as either:

- a first course in statistics for students with a strong quantitative background, typically including calculus.

- a second course in statistics, building on introductory topics taught in courses like Lawrence's Stat 107: Principles of Statistics or AP Statistics.

At Lawrence, this course is required for the statistics track of the mathematics major, the economics and mathematics-economics majors, the business analytics track of the business and entrepreneurship major, and the statistics and data science minor. It also satisfies the statistics requirement for several other majors and minors.

The prerequisite for the course is either 1) a prior college-level course in statistics (i.e. STAT 107, BIOL 170 or 280, ANTH 207, AP Stats) OR 2) Calculus. (Math 140, AP Calculus, or equivalent).

The course does not assume any prior knowledge of statistics, but does move more rapidly than a typical introductory statistics course. Students engage rigorously in statistical thinking and computation, intended to equip them with essential skills for further study in statistics and data science.

1 Visualizing and Summarizing Data

Learning Outcomes:

1. Distinguish between categorical and quantitative variables.
2. Interpret data graphics (histogram, density plot, box plot, violin plot, scatterplot, bar graph).
3. Interpret summary statistics (mean, median, standard deviation, and IQR).
4. Use R to create data graphics and calculate summary statistics.

1.1 Getting Started in R

We'll work with data on houses that sold in King County, WA, (home of Seattle) between 2014 and 2015.

We begin by loading the `tidyverse` package which can be used to create professional data graphics and summaries.

```
library(tidyverse)
```

1.1.1 Previewing the Data

```
head()
```

The `head()` function displays the first 5 rows of the dataset.

```
head(Houses)
```

```
# A tibble: 6 x 9
  Id price bedrooms bathrooms sqft_living sqft_lot condition waterfront
  <int> <dbl>      <dbl>       <dbl>      <dbl> <fct>    <fct>
1     1   1225        4         4.5       5420 average   No
2     2   885.        4         2.5       2830 average   No
```

```

3     3  385.        4      1.75       1620      4980  good      No
4     4  253.        2      1.5        1070      9643  average   No
5     5  468.        2      1          1160      6000  good      No
6     6  310.        3      1          1430      19901 good     No
# i 1 more variable: yr_built <dbl>

```

The rows of the dataset are called **observations**. In this case, the observations are the houses.

The columns of the dataset, which contain information about the houses, are called **variables**.

`glimpse`

The `glimpse()` command shows the number of observations (rows), and the number of variables, (columns). We also see the name of each variable and its type. Variable types include

- **Categorical variables**, which take on groups or categories, rather than numeric values. In R, these might be coded as logical `<logi>`, character `<chr>`, factor `<fct>` and ordered factor `<ord>`.
- **Quantitative variables**, which take on meaningful numeric values. These include numeric `<num>`, integer `<int>`, and double `<dbl>`.
- **Date and time variables** take on values that are dates and times, and are denoted `<dttm>`

```
glimpse(Houses)
```

```

Rows: 100
Columns: 9
$ Id           <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, ~
$ price         <dbl> 1225.00, 885.00, 385.00, 252.70, 468.00, 310.00, 550.00, 4~
$ bedrooms      <dbl> 4, 4, 4, 2, 2, 3, 4, 4, 3, 3, 3, 4, 5, 3, 4, 4, 3, 4, 3, 3~
$ bathrooms     <dbl> 4.50, 2.50, 1.75, 1.50, 1.00, 1.00, 1.00, 1.00, 1.00, 2.25~
$ sqft_living   <dbl> 5420, 2830, 1620, 1070, 1160, 1430, 1660, 1600, 960, 1660, ~
$ sqft_lot       <dbl> 101930, 5000, 4980, 9643, 6000, 19901, 34848, 4300, 6634, ~
$ condition     <fct> average, average, good, average, good, good, poor, good, a~
$ waterfront    <fct> No, No~
$ yr_built      <dbl> 2001, 1995, 1947, 1985, 1942, 1927, 1933, 1916, 1952, 1979~

```

There are 100 houses in the dataset, and 9 variables on each house.

`summary`

`summary` displays the mean, minimum, first quartile, median, third quartile, and maximum for each numeric variable, and the number of observations in each category, for categorical variables.

```
summary(Houses)
```

	Id	price	bedrooms	bathrooms	
Min.	: 1.00	Min. : 180.0	Min. :1.00	Min. :0.750	
1st Qu.	: 25.75	1st Qu.: 322.9	1st Qu.:3.00	1st Qu.:1.500	
Median	: 50.50	Median : 507.5	Median :3.00	Median :2.000	
Mean	: 50.50	Mean : 735.4	Mean :3.39	Mean :2.107	
3rd Qu.	: 75.25	3rd Qu.: 733.8	3rd Qu.:4.00	3rd Qu.:2.500	
Max.	:100.00	Max. :5300.0	Max. :6.00	Max. :6.000	
	sqft_living	sqft_lot	condition	waterfront	yr_built
Min.	: 440	Min. : 1044	poor : 1	No :85	Min. :1900
1st Qu.	:1410	1st Qu.: 5090	fair : 1	Yes:15	1st Qu.:1948
Median	:2000	Median : 7852	average :59		Median :1966
Mean	:2291	Mean : 13205	good :30		Mean :1965
3rd Qu.	:2735	3rd Qu.: 12246	very_good: 9		3rd Qu.:1991
Max.	:8010	Max. :101930			Max. :2014

1.1.2 Modifying the Data

Next we'll look at how to manipulate the data and create new variables.

Adding a New Variable

We can use the `mutate()` function to create a new variable based on variables already in the dataset.

Let's add a variable giving the age of the house, as of 2015.

```
Houses <- Houses |> mutate(age = 2015-yr_built)
head(Houses)
```

```
# A tibble: 6 x 10
  Id price bedrooms bathrooms sqft_living sqft_lot condition waterfront
  <int> <dbl>     <dbl>      <dbl>      <dbl> <fct>    <fct>
1     1 1225        4       4.5      5420   average   No
2     2  885.       4       2.5      2830   average   No
3     3  385.       4       1.75     1620    good     No
```

```

4     4 253.        2      1.5          1070    9643 average   No
5     5 468.        2       1            1160    6000 good     No
6     6 310.        3       1            1430   19901 good     No
# i 2 more variables: yr_built <dbl>, age <dbl>

```

Selecting Columns

If the dataset contains a large number of variables, narrow down to the ones you are interested in working with. This can be done with the `select()` command. If there are not very many variables to begin with, or you are interested in all of them, then you may skip this step.

Let's create a smaller version of the dataset, with only the columns `price`, `sqft_living`, and `waterfront`. We'll call this `Houses_3var`.

```
Houses_3var <- Houses |> select(price, sqft_living, waterfront)
head(Houses_3var)
```

```

# A tibble: 6 x 3
  price sqft_living waterfront
  <dbl>      <dbl> <fct>
1 1225          5420 No
2 885.          2830 No
3 385.          1620 No
4 253.          1070 No
5 468.          1160 No
6 310.          1430 No

```

Filtering by Row

The `filter()` command narrows a dataset down to rows that meet specified conditions.

We'll filter the data to include only houses built after 2000.

```
New_Houses <- Houses |> filter(yr_built>=2000)
head(New_Houses)
```

```

# A tibble: 6 x 10
  Id price bedrooms bathrooms sqft_living sqft_lot condition waterfront
  <int> <dbl>      <dbl>      <dbl>      <dbl> <fct>     <fct>
1     1 1225          4        4.5        5420  101930 average   No
2    16 3075          4        5           4550  18641 average Yes

```

```

3    23  862.      5    2.75      3595    5639 average  No
4    24  360.      4    2.5       2380    5000 average  No
5    25  625.      4    2.5       2570    5520 average  No
6    27  488.      3    2.5       3160   13603 average  No
# i 2 more variables: yr_built <dbl>, age <dbl>

```

Now, we'll filter the data to include only houses on the waterfront.

```

New_Houses <- Houses |> filter(waterfront == "Yes")
head(New_Houses)

```

```

# A tibble: 6 x 10
  Id price bedrooms bathrooms sqft_living sqft_lot condition waterfront
  <int> <dbl>     <dbl>     <dbl>     <dbl> <fct>    <fct>
1 16 3075        4      5       4550  18641 average Yes
2 19 995.        3     4.5      4380  47044 average Yes
3 34 825.        2      1       1150  12775 good   Yes
4 40 2400.       4     2.5      3650  8354  average Yes
5 42 290.        2     0.75     440   8313  good   Yes
6 46 5111.       5     5.25     8010  45517 average Yes
# i 2 more variables: yr_built <dbl>, age <dbl>

```

1.2 Summary Statistics

1.2.1 Measures of Center

Common ways to characterize the center of a distribution include mean, median, and mode.

For a set of n values y_1, \dots, y_n :

- **mean** (\bar{y}) represents the numerical average and is calculated by $\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$.
- **median** represents the middle number when the values are arranged from least to greatest. If there are an even number of values in the dataset, then the median is given by the average of the middle two numbers.
 - The median of the upper half of the values is called the **upper (or 3rd) quartile**. This represents the 75th percentile in the distribution.
 - The median of the lower half of the values is called the **lower (or 1st) quartile**. This represents the 25th percentile in the distribution.
- **mode** is the most frequently occurring number in the data.

1.2.2 Measures of Spread

Common ways of measuring the amount of spread, or variability, in a variable include:

- **range**: the difference between the maximum and minimum values
- **interquartile range**: the difference between the upper and lower quartiles (i.e. the range of the middle 50% of the values).
- **standard deviation** (s): standard deviation is approximately the average deviation between an observation and the mean. It is calculated by

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

The square of the standard deviation, called the **variance** is denoted s^2 .

1.2.3 Calculating Summary Statistics in R

Let's calculate the mean, median, and standard deviation, in prices.

```
Houses_Summary <- Houses |> summarize(Mean_Price = mean(price, na.rm=TRUE),  
                                         Median_Price = median(price, na.rm=TRUE),  
                                         StDev_Price = sd(price, na.rm = TRUE),  
                                         Number_of_Houses = n())  
  
Houses_Summary
```

```
# A tibble: 1 x 4  
  Mean_Price Median_Price StDev_Price Number_of_Houses  
    <dbl>        <dbl>       <dbl>           <int>  
1     735.        507.       835.          100
```

Notes:

1. The `n()` command calculates the number of observations.
2. The `na.rm=TRUE` command removes missing values, so that summary statistics can be calculated. It's not needed here, since this dataset doesn't include missing values, but if the dataset does include missing values, you will need to include this, in order to do the calculation.

The `kable()` function in the `knitr()` package creates tables with professional appearance.

```
library(knitr)  
kable(Houses_Summary)
```

Mean_Price	Median_Price	StDev_Price	Number_of_Houses
735.3525	507.5	835.1231	100

1.2.4 Grouped Summaries

`group_by()`

The `group_by()` command allows us to calculate summary statistics, with the data broken down by category. We'll compare waterfront houses to non-waterfront houses.

```

Houses_Grouped_Summary <- Houses |> group_by(waterfront) |>
  summarize(Mean_Price = mean(price, na.rm=TRUE),
            Median_Price = median(price, na.rm=TRUE),
            StDev_Price = sd(price, na.rm = TRUE),
            Number_of_Houses = n())
kable(Houses_Grouped_Summary)

```

waterfront	Mean_Price	Median_Price	StDev_Price	Number_of_Houses
No	523.7595	450	295.7991	85
Yes	1934.3800	1350	1610.7959	15

Note: `arrange(desc(Mean_Gross))` arranges the table in descending order of Mean_Gross.
 To arrange in ascending order, use `arrange(Mean_Gross)`.

1.3 Data Visualization

Next, we'll create graphics to help us visualize the distributions and relationships between variables. We'll use the `ggplot()` function, which is part of the `tidyverse` package.

1.3.1 Histogram

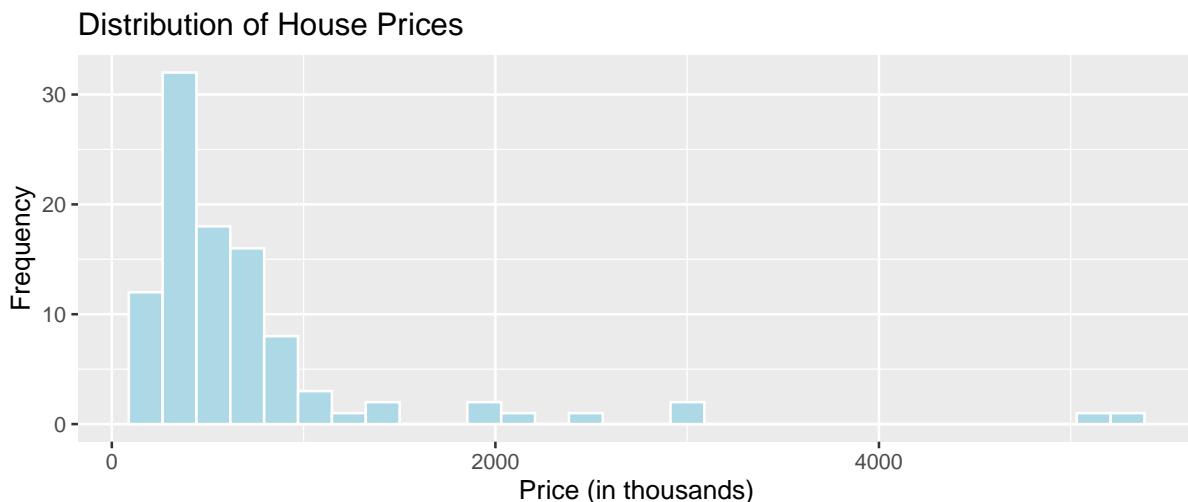
Histograms are useful for displaying the distribution of a single quantitative variable. In a histogram, the x-axis breaks the variable into ranges of values, and the y-axis displays the number of observations with a value falling in that category (frequency).

General Template for Histogram

```
ggplot(data=DatasetName, aes(x=VariableName)) +  
  geom_histogram(fill="colorchoice", color="colorchoice") +  
  ggtitle("Plot Title") +  
  xlab("x-axis label") +  
  ylab("y-axis label")
```

Histogram of House Prices

```
ggplot(data=Houses, aes(x=price)) +  
  geom_histogram(fill="lightblue", color="white") +  
  ggtitle("Distribution of House Prices") +  
  xlab("Price (in thousands)") +  
  ylab("Frequency")
```



We see that the distribution of house prices is right-skewed. Most houses cost less than \$1,000,000, though there are a few houses that are much more expensive. The most common price range is around \$400,000 to \$500,000.

1.3.2 Density Plot

Density plots show the distribution for a quantitative variable price. Scores can be compared across categories, like whether or not the house is on a waterfront.

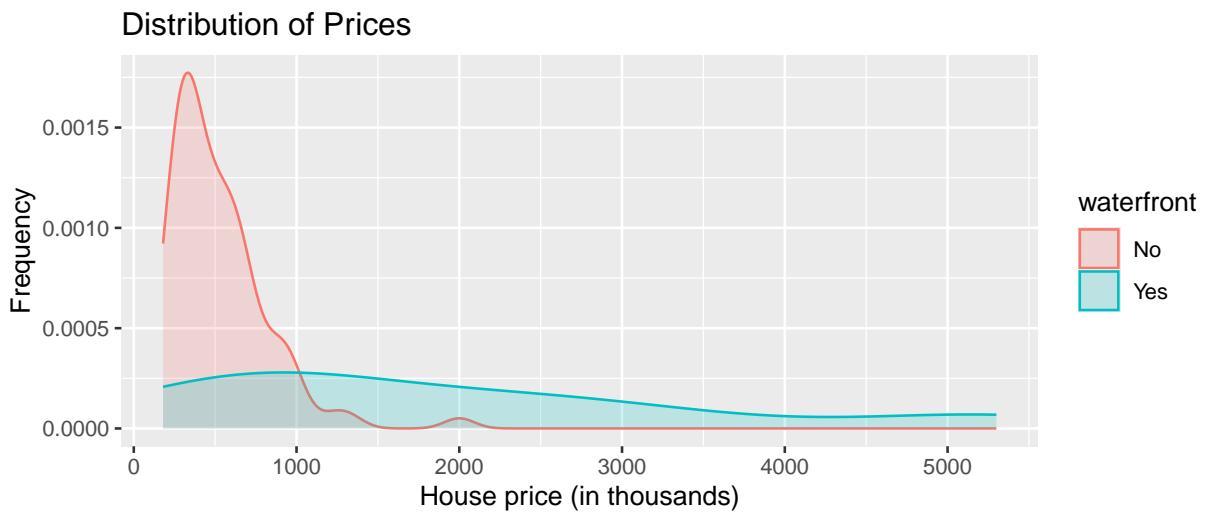
General Template for Density Plot

```
ggplot(data=DatasetName, aes(x=QuantitativeVariable,
                             color=CategoricalVariable, fill=CategoricalVariable)) +
  geom_density(alpha=0.2) +
  ggtitle("Plot Title") +
  xlab("Axis Label") +
  ylab("Frequency")
```

`alpha`, ranging from 0 to 1 dictates transparency.

Density Plot of House Prices

```
ggplot(data=Houses, aes(x=price, color=waterfront, fill=waterfront)) +
  geom_density(alpha=0.2) +
  ggtitle("Distribution of Prices") +
  xlab("House price (in thousands)") +
  ylab("Frequency")
```



We see that on average, houses on the waterfront tend to be more expensive and have a greater price range than houses not on the waterfront.

1.3.3 Boxplot

Boxplots can be used to compare a quantitative variable with a categorical variable. The middle 50% of observations are contained in the “box”, with the upper and lower 25% of the observations in each tail.

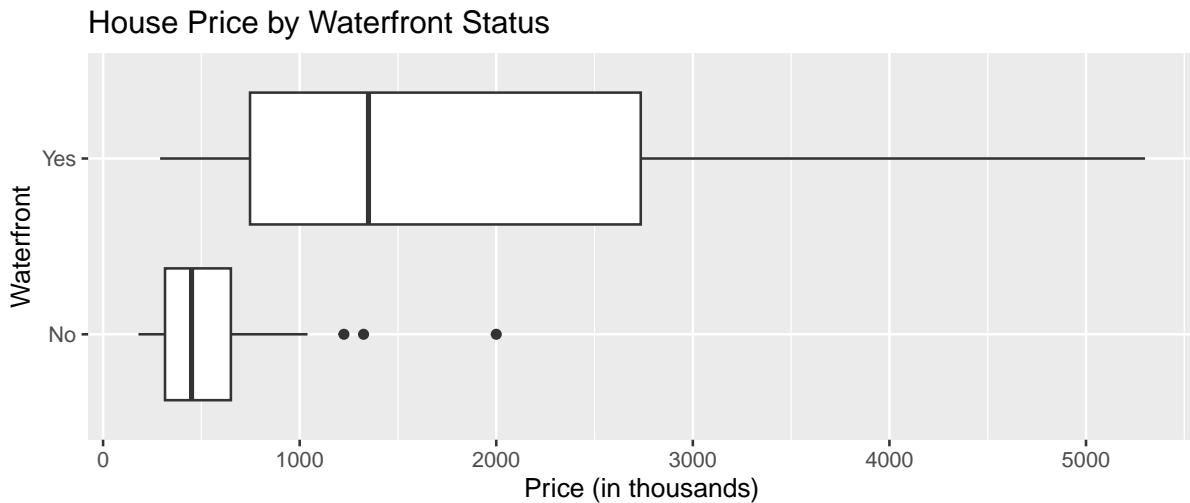
General Template for Boxplot

```
ggplot(data=DatasetName, aes(x=CategoricalVariable,
                               y=QuantitativeVariable)) +
  geom_boxplot() +
  ggtitle("Plot Title") +
  xlab("Variable Name") + ylab("Variable Name")
```

You can make the plot horizontal by adding `+ coordflip()`. You can turn the axis text vertical by adding `theme(axis.text.x = element_text(angle = 90))`.

Boxplot Comparing Price by Waterfront Status

```
ggplot(data=Houses, aes(x=waterfront, y=price)) + geom_boxplot() +
  ggtitle("House Price by Waterfront Status") +
  xlab("Waterfront") + ylab("Price (in thousands)") + coord_flip()
```



For houses not on the waterfront, the median price is about \$400,000, and the middle 50% of prices range from about \$300,000 to \$600,000.

For waterfront houses, the median price is about \$1,500,000, and the middle 50% of prices range from about \$900,000 to \$1,900,000.

1.3.4 Violin Plot

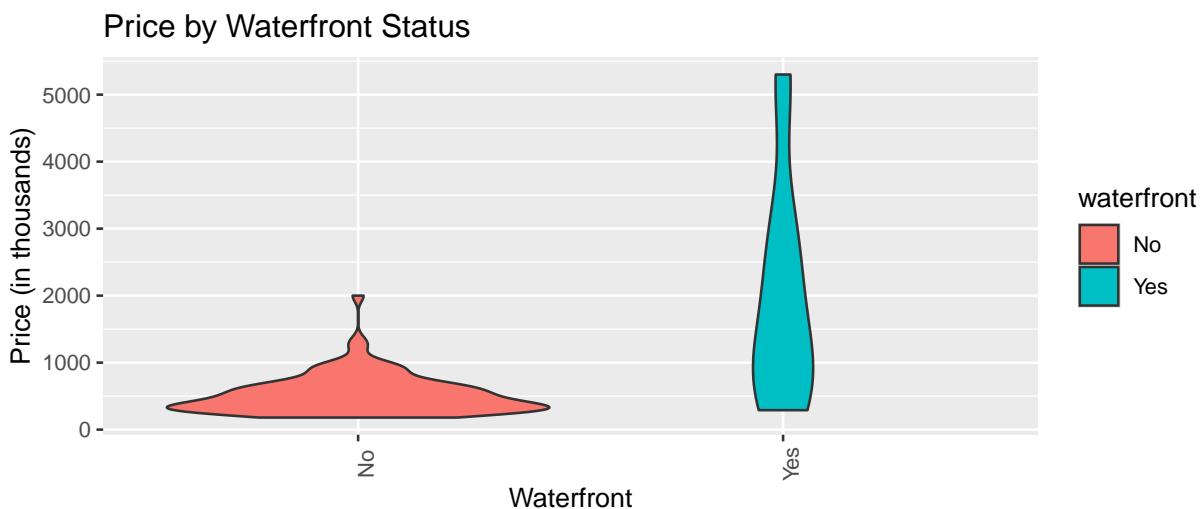
Violin plots are an alternative to boxplots. The width of the violin tells us the density of observations in a given range.

General Template for Violin Plot

```
ggplot(data=DatasetName, aes(x=CategoricalVariable, y=QuantitativeVariable,
                               fill=CategoricalVariable)) +
  geom_violin() +
  ggtitle("Plot Title") +
  xlab("Variable Name") + ylab("Variable Name")
```

Violin Plot Comparing Prices by Waterfront

```
ggplot(data=Houses, aes(x=waterfront, y=price, fill=waterfront)) +
  geom_violin() +
  ggtitle("Price by Waterfront Status") +
  xlab("Waterfront") + ylab("Price (in thousands)") +
  theme(axis.text.x = element_text(angle = 90))
```



Again, we see that houses on the waterfront tend to be more expensive than those not on the waterfront, and have a wider range in prices.

1.3.5 Scatterplot

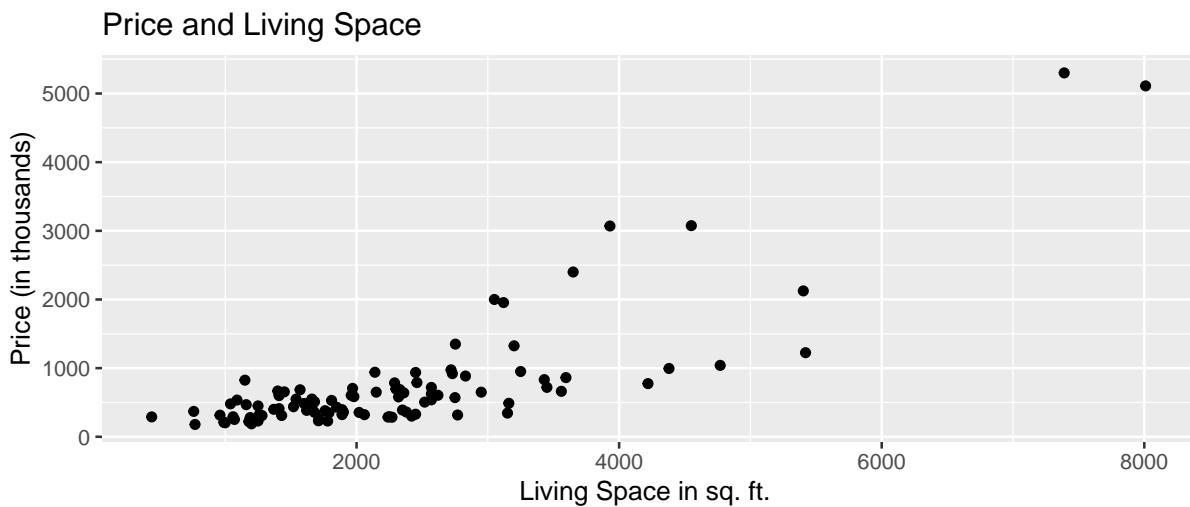
Scatterplots are used to visualize the relationship between two quantitative variables.

Scatterplot Template

```
ggplot(data=DatasetName, aes(x=CategoricalVariable, y=QuantitativeVariable)) +  
  geom_point() +  
  ggtitle("Plot Title") +  
  ylab("Axis Label") +  
  xlab("Axis Label")
```

Scatterplot Comparing Price and Square Feet of Living Space

```
ggplot(data=Houses, aes(x=sqft_living, y=price)) +  
  geom_point() +  
  ggtitle("Price and Living Space") +  
  ylab("Price (in thousands)") +  
  xlab("Living Space in sq. ft. ")
```

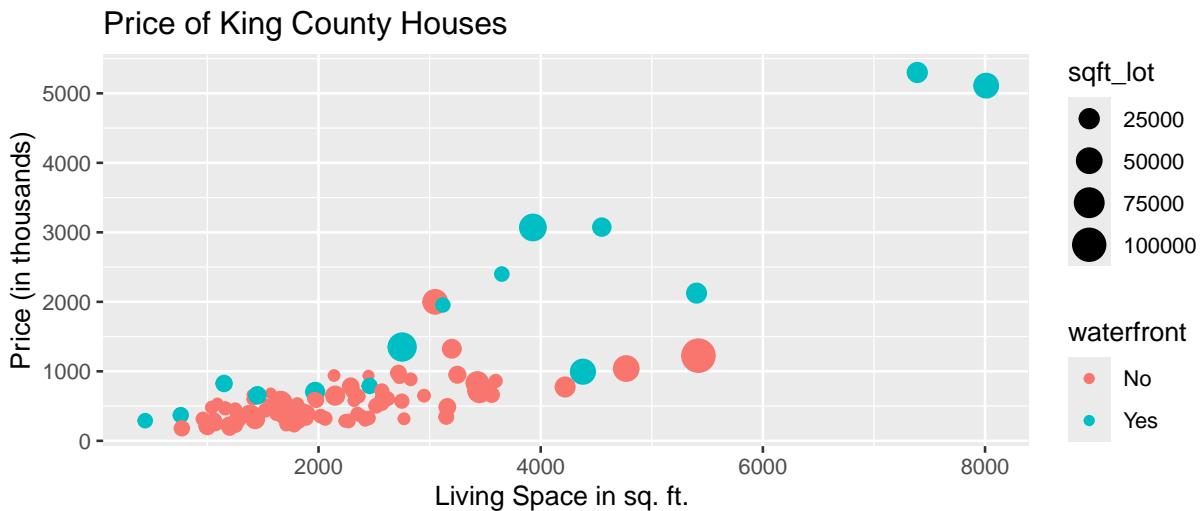


We see that there is an upward trend, indicating that houses with more living space tend to, on average, be higher priced than those with less living space. The relationship appears to be roughly linear, though there might be some curvature, as living space gets very large. There are some exceptions to this trend, most notably a house with more than 7,000 square feet, priced just over \$1,000,000.

We can also add color, size, and shape to the scatterplot to display information about other variables.

We'll use color to illustrate whether the house is on the waterfront, and size to represent the square footage of the entire lot (including the yard and the house).

```
ggplot(data=Houses,
       aes(x=sqft_living, y=price, color=waterfront, size=sqft_lot)) +
  geom_point() +
  ggtitle("Price of King County Houses") +
  ylab("Price (in thousands)") +
  xlab("Living Space in sq. ft. ")
```



We notice that many of the largest and most expensive houses are on the waterfront.

1.3.6 Bar Graph

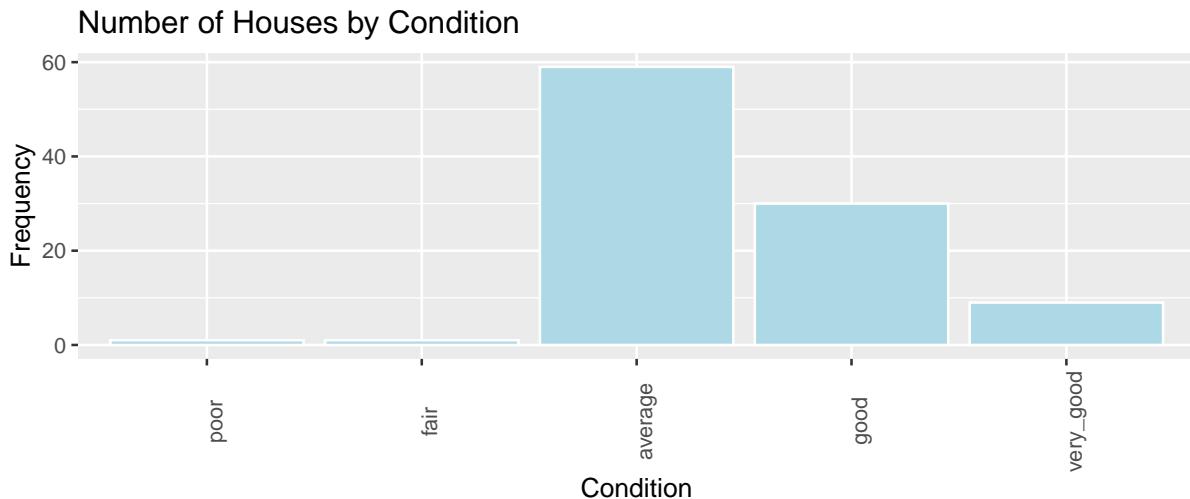
Bar graphs can be used to visualize one or more categorical variables. A bar graph is similar to a histogram, in that the y-axis again displays frequency, but the x-axis displays categories, instead of ranges of values.

Bar Graph Template

```
ggplot(data=DatasetName, aes(x=CategoricalVariable)) +
  geom_bar(fill="colorchoice",color="colorchoice") +
  ggtitle("Plot Title") +
  xlab("Variable Name") +
  ylab("Frequency")
```

Bar Graph by Condition

```
ggplot(data=Houses, aes(x=condition)) +
  geom_bar(fill="lightblue",color="white") +
  ggtitle("Number of Houses by Condition") +
  xlab("Condition") +
  ylab("Frequency") +
  theme(axis.text.x = element_text(angle = 90))
```



We see that the majority of houses are in average condition. Some are in good or very good condition, while very few are in poor or very poor condition.

1.3.7 Stacked and Side-by-Side Bar Graphs

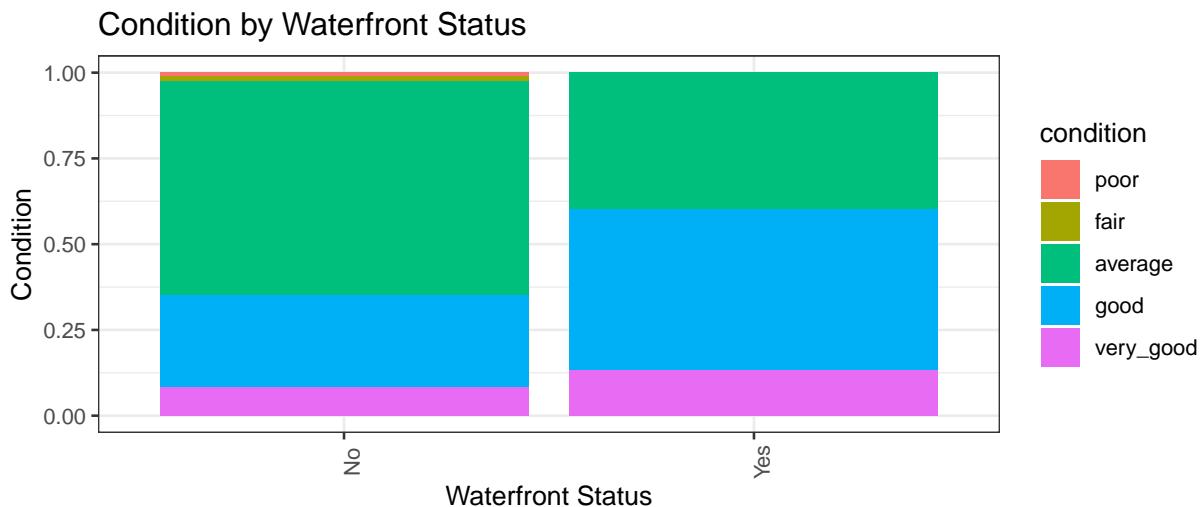
Stacked Bar Graph Template

```
ggplot(data = DatasetName, mapping = aes(x = CategoricalVariable1,
                                         fill = CategoricalVariable2)) +
  stat_count(position="fill") +
  theme_bw() + ggtitle("Plot Title") +
  xlab("Variable 1") +
  ylab("Proportion of Variable 2") +
  theme(axis.text.x = element_text(angle = 90))
```

Stacked Bar Graph Example

The `stat_count(position="fill")` command creates a stacked bar graph, comparing two categorical variables. Let's explore whether waterfront status is related to condition.

```
ggplot(data = Houses, mapping = aes(x = waterfront, fill = condition)) +
  stat_count(position="fill") +
  theme_bw() + ggtitle("Condition by Waterfront Status") +
  xlab("Waterfront Status") +
  ylab("Condition") +
  theme(axis.text.x = element_text(angle = 90))
```



We see that a higher proportion of waterfront houses are in good or excellent condition than non-waterfront houses.

Side-by-side Bar Graph Template

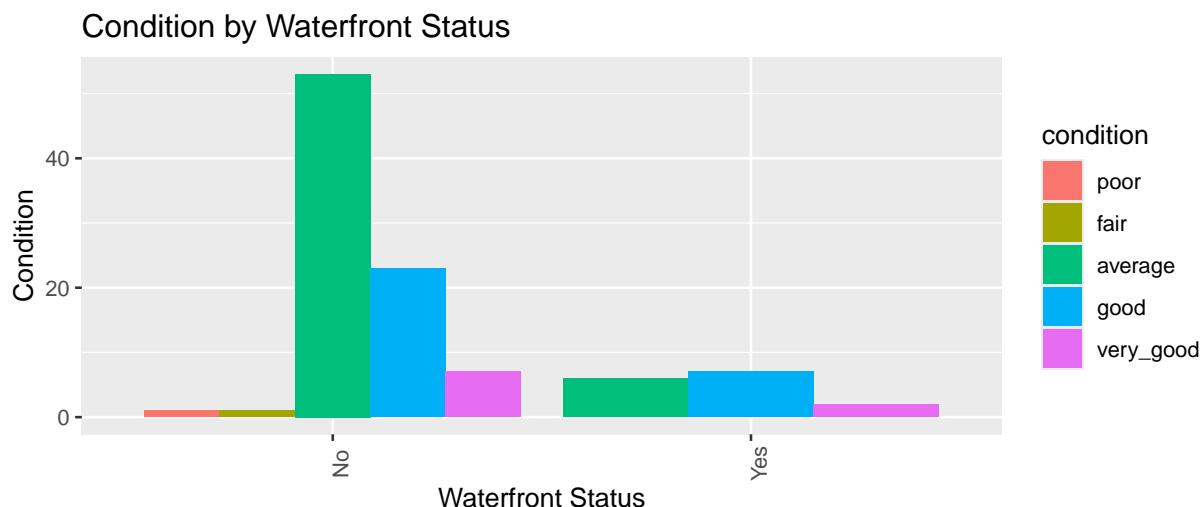
We can create a side-by-side bar graph, using `position=dodge`.

```
ggplot(data = DatasetName, mapping = aes(x = CategoricalVariable1,
                                         fill = CategoricalVariable2)) +
  geom_bar(position = "dodge") +
  ggtitle("Plot Title") +
  xlab("Genre") +
  ylab("Frequency")
```

Side-by-side Bar Graph Example

```
ggplot(data = Houses, mapping = aes(x = waterfront, fill = condition)) +
  geom_bar(position = "dodge") +
  ggtitle("Condition by Waterfront Status") +
  xlab("Waterfront Status") +
```

```
ylab("Condition") +
  theme(axis.text.x = element_text(angle = 90))
```



In this case, since there are so few waterfront houses, the graph is hard to read and not very useful.

The stacked bar graph is a better way to convey information in this instance, though you may find that for a different dataset, the side-by-side bar graph could be a better choice.

1.3.8 Correlation Plot

Correlation plots can be used to visualize relationships between quantitative variables. Correlation is a number between -1 and 1, describing the strength of the linear relationship between two variables. Variables with strong positive correlations will have correlation close to +1, while variables with strong negative correlations will have correlations close to -1. Variables with little to no relationship will have correlation close to 0.

The `cor()` function calculates correlations between quantitative variables. We'll use `select_if` to select only numeric variables. The ‘use=“complete.obs” command tells R to ignore observations with missing data.

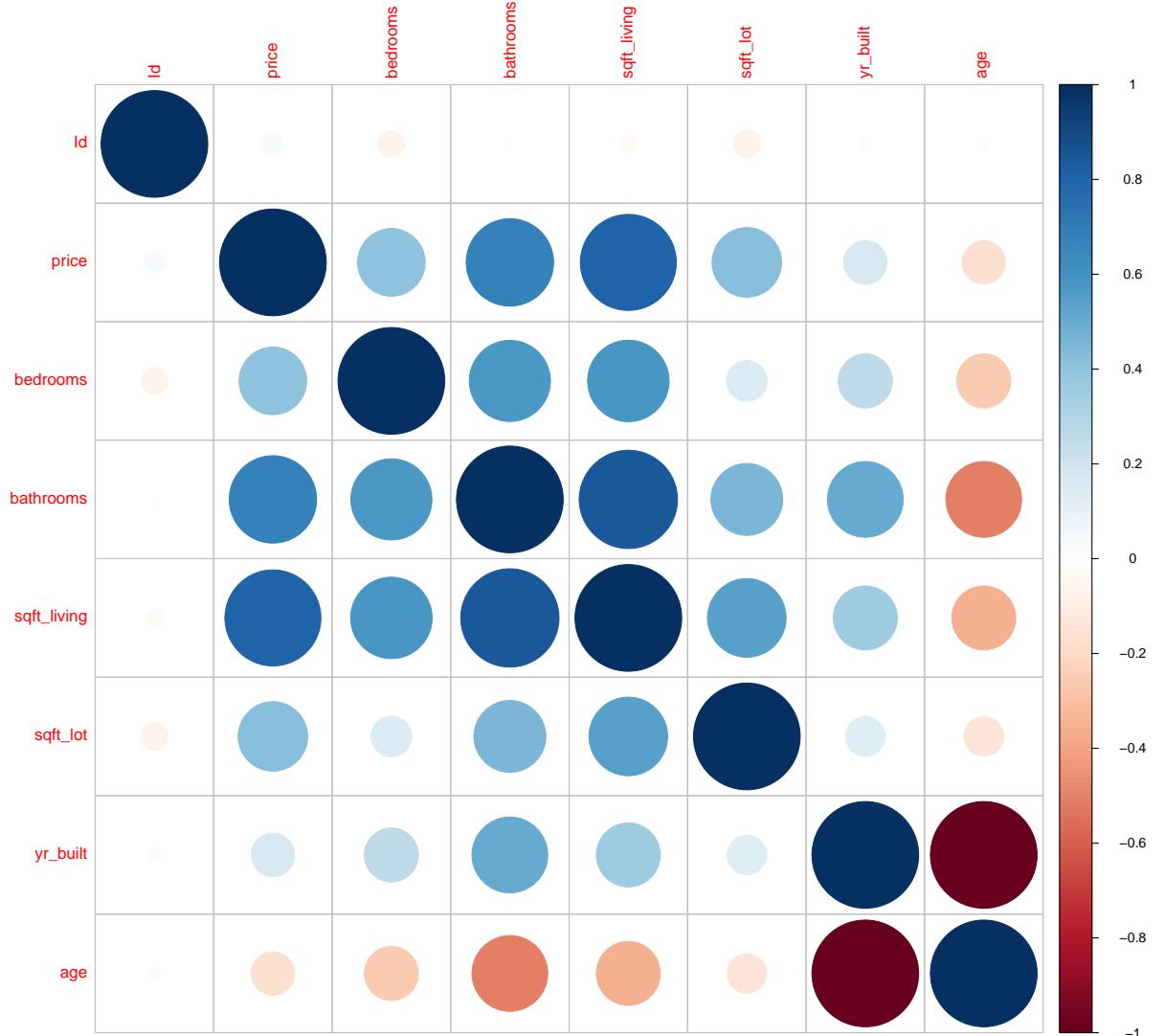
```
cor(select_if(Houses, is.numeric), use="complete.obs") |> round(2)
```

	Id	price	bedrooms	bathrooms	sqft_living	sqft_lot	yr_built	age
Id	1.00	0.03	-0.06	-0.01	-0.03	-0.07	-0.02	0.02
price	0.03	1.00	0.40	0.67	0.81	0.42	0.17	-0.17
bedrooms	-0.06	0.40	1.00	0.58	0.58	0.15	0.26	-0.26

bathrooms	-0.01	0.67	0.58	1.00	0.85	0.45	0.50	-0.50
sqft_living	-0.03	0.81	0.58	0.85	1.00	0.54	0.36	-0.36
sqft_lot	-0.07	0.42	0.15	0.45	0.54	1.00	0.14	-0.14
yr_built	-0.02	0.17	0.26	0.50	0.36	0.14	1.00	-1.00
age	0.02	-0.17	-0.26	-0.50	-0.36	-0.14	-1.00	1.00

The `corrplot()` function in the `corrplot()` package provides a visualization of the correlations. Larger, thicker circles indicate stronger correlations.

```
library(corrplot)
Corr <- cor(select_if(Houses, is.numeric), use="complete.obs")
corrplot(Corr)
```



We see that price has a strong positive correlation with square feet of living space, and is also positively correlated with number of bedrooms and bathrooms. Living space, bedrooms, and bathrooms are all positively correlated, which makes sense, since we would expect bigger houses to have more bedrooms and bathrooms. Price does not show much correlation with the other variables. We notice that bathrooms is negatively correlated with age, which means older houses tend to have fewer bathrooms than newer ones. Not surprisingly, age is very strongly correlated with year built.

1.3.9 Scatterplot Matrix

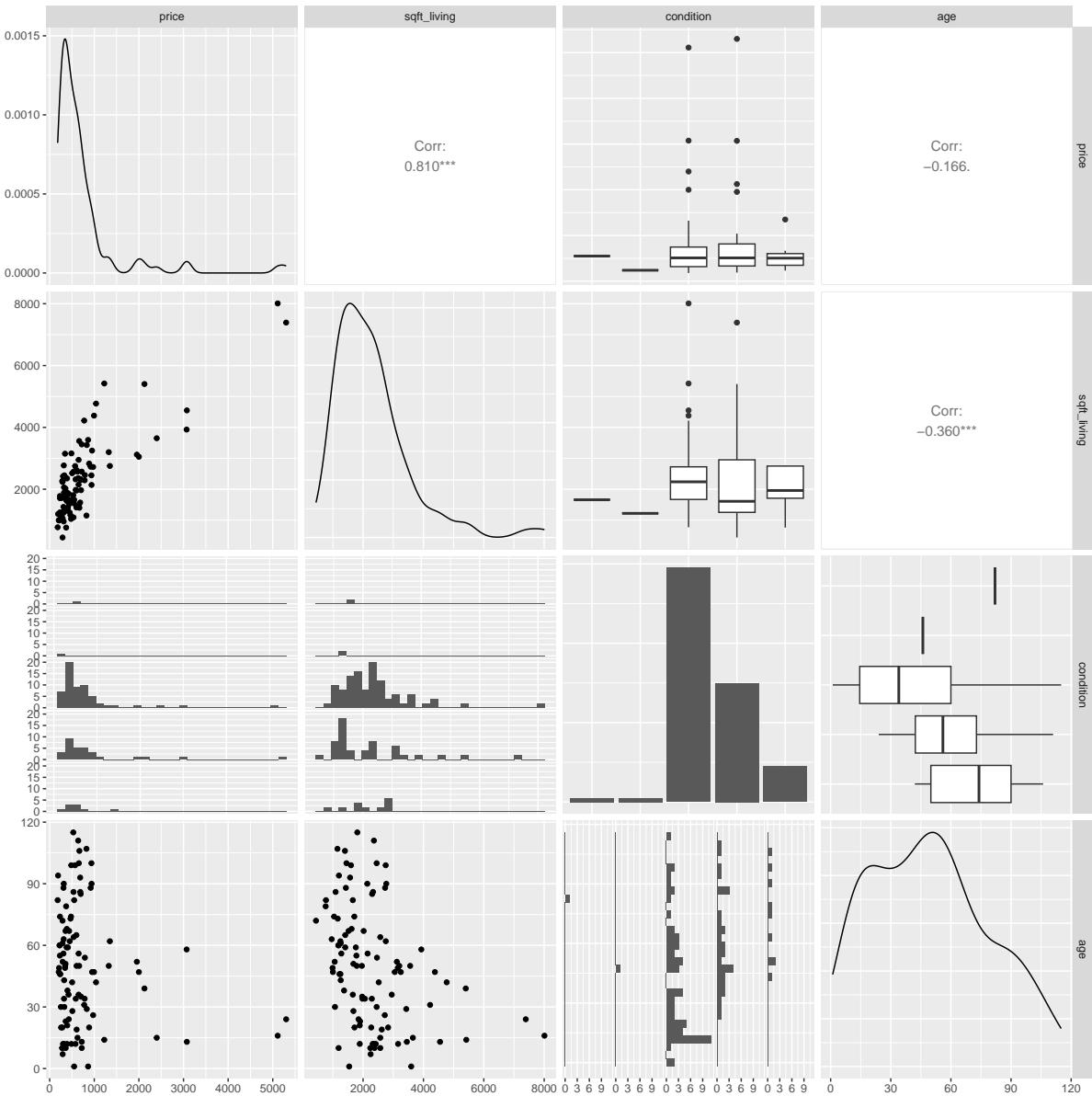
A scatterplot matrix is a grid of plots. It can be created using the `ggpairs()` function in the `GGally` package.

The scatterplot matrix shows us:

1. Along the diagonal are density plots for quantitative variables, or bar graphs for categorical variables, showing the distribution of each variable.
2. Under the diagonal are plots showing the relationships between the variables in the corresponding row and column. Scatterplots are used when both variables are quantitative, bar graphs are used when both variables are categorical, and boxplots are used when one variable is categorical, and the other is quantitative.
3. Above the diagonal are correlations between quantitative variables.

Including too many variables can make these hard to read, so it's a good idea to use `select` to narrow down the number of variables.

```
library(GGally)
ggpairs(Houses |> select(price, sqft_living, condition, age))
```



The scatterplot matrix is useful for helping us notice key trends in our data. However, the plot can hard to read as it is quite dense, especially when there are a large number of variables. These can help us look for trends from a distance, but we should then focus in on more specific plots.

2 Introduction to Statistical Models

Learning Outcomes:

5. Use regression models to calculate predicted values, changes, and differences.
6. Interpret regression coefficients for simple linear and multiple regression models, including models with interaction.
7. Calculate and interpret regression sums of squares SSR, SSM, and SST, and coefficient of determination, R^2 .
8. Determine how changes to sample size, number of terms in a model, or regression coefficients impact SST, SSR, and R^2 .
9. Explain what interactions between variables mean, in context.
10. Explain the least-squares estimation process.
11. Show how to calculate F-statistics.
12. Draw conclusions based on F-statistics.
13. Fit regression models using R and report conclusions.

2.1 Fitting Models to Data

2.1.1 Terminology

In this section, we'll use statistical models to predict the prices of houses in King County, WA.

In a statistical model,

- The variable we are trying to predict (price) is called the **response variable** (denoted Y).

- Variable(s) we use to help us make the prediction is(are) called **explanatory variables** (denoted X). These are also referred to as **predictor variables** or **covariates**.

In this section, we'll attempt to predict the price of a house, using information about its size (in square feet), and whether or not it is on the waterfront. The price is our response variable, while size and waterfront location are explanatory variables.

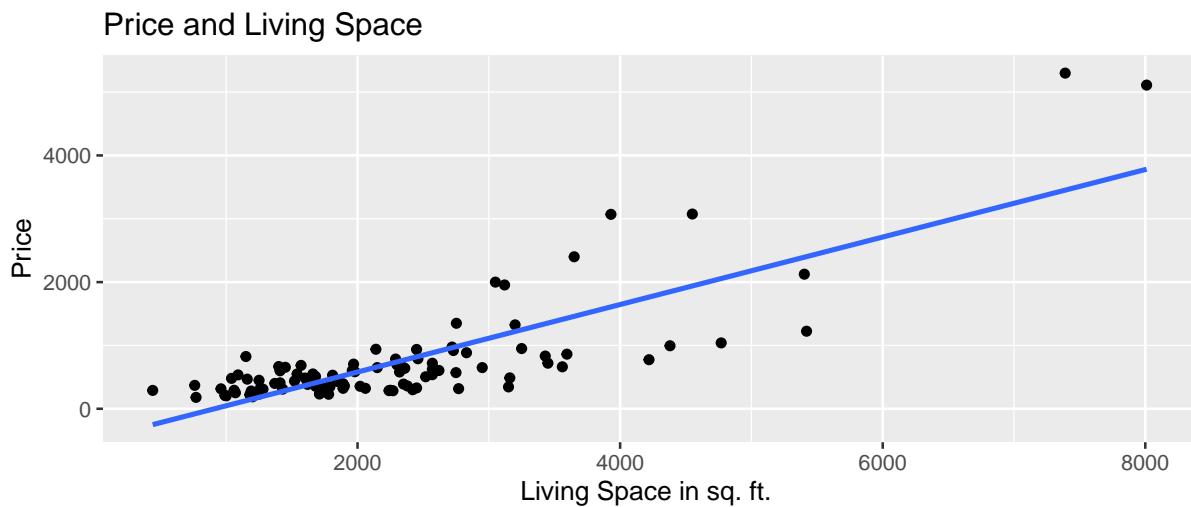
- **Categorical variables** are variables that take on groups or categories, rather than numeric values, for example, whether or not the house is on the waterfront.
- **Quantitative variables** take on meaningful numeric values, for example the number of square feet in the house.

2.1.2 Quantitative Explanatory Variable

We'll first predict the price of the house, using the number of square feet of living space as our explanatory variable.

We'll assume that price changes linearly with square feet, and fit a trend line to the data.

```
ggplot(data=Houses, aes(x=sqft_living, y=price)) +
  geom_point() +
  stat_smooth(method="lm", se=FALSE) +
  ggtitle("Price and Living Space") +
  ylab("Price") +
  xlab("Living Space in sq. ft. ")
```



The model equation is

$$\widehat{\text{Price}} = b_0 + b_1 \times \text{Sq.Ft.}$$

Note, the symbol over the response variable (Price) is read as “hat”, and means “predicted price”.

We fit the model in R, using the `lm` (linear model) command. The output gives the estimates of b_0 and b_1 .

```
M_House_sqft <- lm(data=Houses, price~sqft_living)
M_House_sqft
```

```
Call:
lm(formula = price ~ sqft_living, data = Houses)

Coefficients:
(Intercept)  sqft_living
-484.9575      0.5328
```

The estimates are $b_0 = -484.9575$ and $b_1 = 0.5328$.

The model equation is

$$\widehat{\text{Price}} = -484.9575 + 0.5328 \times \text{Sq.Ft.}$$

Interpretations

The intercept b_0 represents the expected (or average) value of the response variable, when the explanatory variable is equal to 0. This is not always a meaningful interpretation in context.

The slope b_1 represents the expected (or average) change in the response variable for each one-unit increase in the explanatory variable.

- On average, a house with 0 square feet is expected to cost -485 thousand dollars. This is not a sensible interpretation, as there are no houses with 0 square feet.
- For each additional square foot in living space, the price of the house is expected to increase by 0.5328 thousand dollars (or \$533).
 - Since a 1 square ft. increase is very small, it makes more sense to give the interpretation in terms of a 100-square foot increase. For each additional 100 square feet in living space, the price of the house is expected to increase by 53.28 thousand dollars.

Prediction

We can predict the price of a house with a given number of square feet by plugging the square feet into the model equation.

The predicted price of a house with 1,500 square feet is

$$\widehat{\text{Price}} = -484.9575 + 0.5328 \times 1500 = \$314 \text{ thousand}$$

We can calculate this directly in R using the `predict` command.

```
predict(M_House_sqft, newdata=data.frame(sqft_living=1500))
```

```
1  
314.1803
```

We should only try to make predictions on houses within the range of the observed data. Since the largest house in the dataset is 8,000 square feet we should not try to predict the price of house with 10,000 square feet.

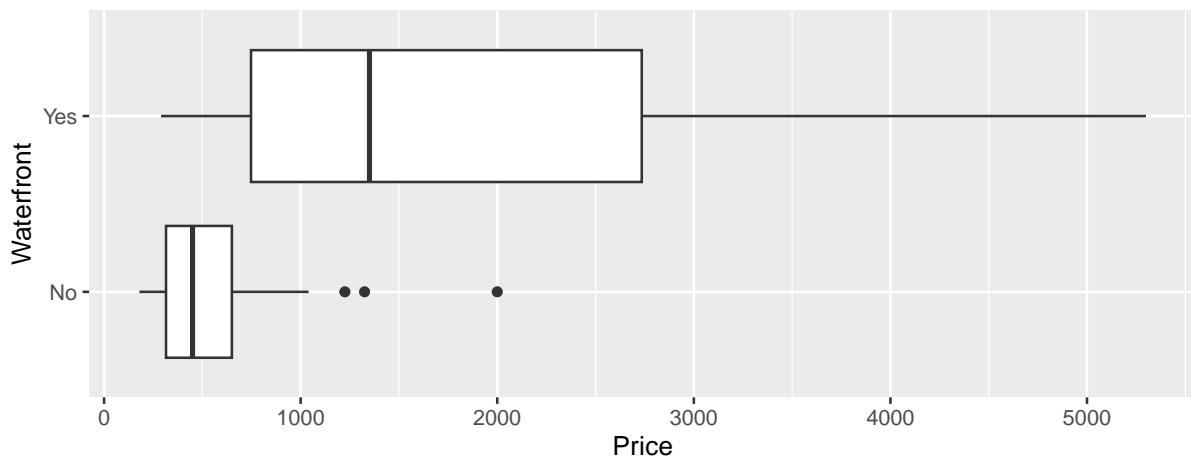
2.1.3 Categorical Explanatory Variable

Next, we'll predict the price of a house based on whether or not it is on the waterfront.

The boxplot shows the distribution of prices for waterfront and nonwaterfront houses. The red dots indicate the mean.

```
ggplot(data=Houses, aes(x=waterfront, y=price)) + geom_boxplot() +  
  ggtitle("House Price by Waterfront Status") +  
  xlab("Waterfront") + ylab("Price") + coord_flip() +  
  stat_summary(fun.y=mean, geom="point", shape=20, color="red", fill="red")
```

House Price by Waterfront Status



The table displays the price summary by waterfront status.

```
Houses_Grouped_Summary <- Houses %>% group_by(waterfront) %>%
  summarize(Mean_Price = mean(price, na.rm=TRUE),
            Median_Price = median(price, na.rm=TRUE),
            StDev_Price = sd(price, na.rm = TRUE),
            Number_of_Houses = n())
kable(Houses_Grouped_Summary)
```

waterfront	Mean_Price	Median_Price	StDev_Price	Number_of_Houses
No	523.7595	450	295.7991	85
Yes	1934.3800	1350	1610.7959	15

The model equation is

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

The waterfront variable takes on value of 1 if the house is on the waterfront, and 0 otherwise.

```
M_House_wf <- lm(data=Houses, price~waterfront)
M_House_wf
```

```
Call:
lm(formula = price ~ waterfront, data = Houses)
```

Coefficients:

(Intercept)	waterfrontYes
523.8	1410.6

The estimates are $b_0 = 523.8$ and $b_1 = 1410.6$.

The model equation is

$$\widehat{\text{Price}} = 523.8 + 1410.6 \times \text{Waterfront}$$

Interpretations

The intercept b_0 represents the expected (or average) value of the response variable in the “baseline” category (in this case non-waterfront).

The coefficient b_1 represents the expected (or average) difference in response between the a category and the “baseline” category.

- On average, a house that is not on the waterfront is expected to cost 523.8 thousand dollars.
- On average a house that is on the waterfront is expected to cost 1410.6 thousand (or 1.4 million) dollars more than a house that is not on the waterfront.

Prediction

We can predict the price of a house with a given number of square feet by plugging in either 1 or 0 for the waterfront variable.

The predicted price of a house on the waterfront is:

$$\widehat{\text{Price}} = 523.8 + 1410.6 \times 1 = \$1934.6 \text{ thousand (or 1.9 million)}$$

The predicted price of a house not on the waterfront is:

$$\widehat{\text{Price}} = 523.8 + 1410.6 \times 0 = \$523.8 \text{ thousand}$$

Calculations in R:

```
predict(M_House_wf, newdata=data.frame(waterfront="Yes"))
```

```
1  
1934.38
```

```
predict(M_House_wf, newdata=data.frame(waterfront="No"))
```

```
1  
523.7595
```

Notice that the predicted prices for each category correspond to the average price for that category.

2.1.4 Multiple Explanatory Variables

We've used square feet and waterfront status as explanatory variables individually. We can also build a model that uses both of these variables at the same time.

A model with two or more explanatory variables is called a **multiple regression model**.

The model equation is

$$\widehat{\text{Price}} = b_0 + b_1 \times \text{Sq. Ft} + b_2 \times \text{Waterfront}$$

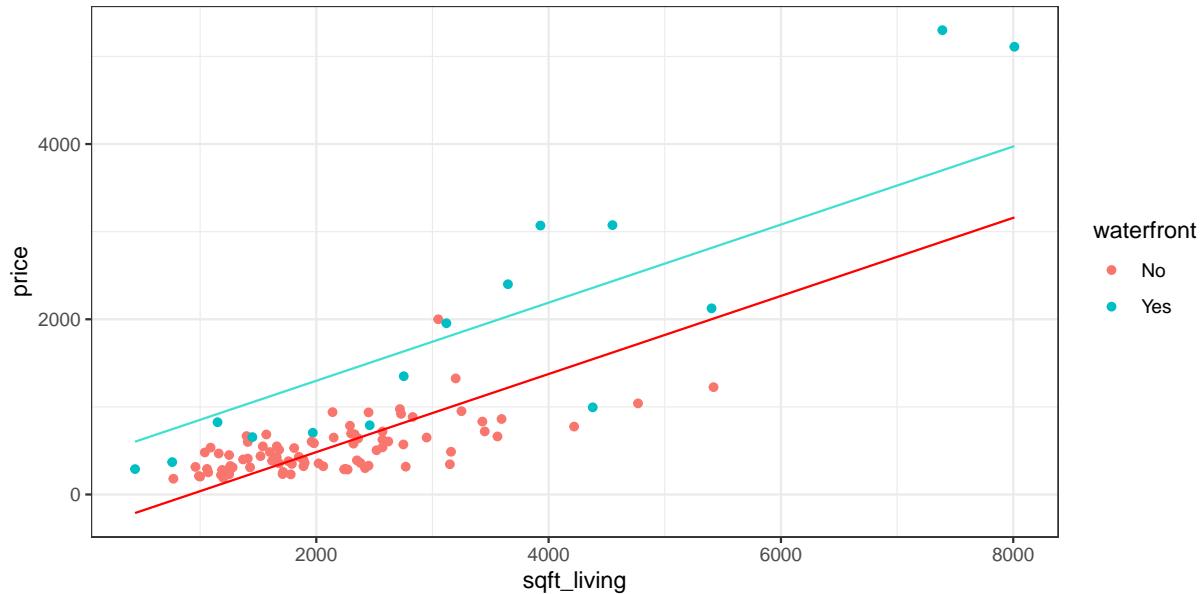
For a house not on the waterfront, $b_2 = 0$, so the model equation is:

$$\widehat{\text{Price}} = b_0 + b_1 \text{Sq. Ft}$$

For a house on the waterfront, $b_2 = 1$, so the model equation is:

$$\widehat{\text{Price}} = (b_0 + b_2) + b_1 \times \text{Sq. Ft}$$

Notice that the slope is the same, regardless of whether the house is on the waterfront (b_1). The intercept, however, is different (b_0 for houses not on the waterfront, and $b_0 + b_2$ for houses on the waterfront). Thus, the model assumes that price increases at the same rate, with respect to square feet, regardless of whether or not it is on the waterfront, but allows the predicted price for a waterfront house to differ from a non-waterfront house of the same size.



We fit the model in R.

```
M_wf_sqft <- lm(data=Houses, price~sqft_living+waterfront)
M_wf_sqft
```

Call:
`lm(formula = price ~ sqft_living + waterfront, data = Houses)`

Coefficients:

(Intercept)	sqft_living	waterfrontYes
-407.6549	0.4457	814.3613

The model equation is

$$\widehat{\text{Price}} = -407.7 + 0.4457 \times \text{Sq. Ft} + 814.36 \times \text{Waterfront}$$

Interpretations

The intercept b_0 represents the expected (or average) value of the response variable, when all quantitative explanatory variables are equal to 0, and all categorical variables are in the “baseline” category. This interpretation is not always sensible.

We interpret coefficients b_j for categorical or quantitative variables, the same way we would in a regression model with only one variable, but we need to state that all other explanatory variables are being held constant.

- On average, a house that is not on the waterfront with 0 square feet is expected to cost -407.7 thousand dollars. This is not a sensible interpretation, since there are no houses with 0 square feet.
- For each 1-square foot increase in size, the price of a house is expected to increase by 0.4457 thousand (or 446 hundred) dollars, assuming waterfront status is the same. Equivalently, for each 100-square foot increase in size, the price of a house is expected to increase by 44.57 thousand dollars, assuming waterfront status is the same.
- On average, a house on the waterfront is expected to cost 814 thousand dollars more than a house that is not on the waterfront, assuming square footage is the same.

Prediction

The predicted price of a 1,500 square foot house on the waterfront is:

$$\widehat{\text{Price}} = -407.7 + 0.4457 \times 1500 + 814.36 \times 1 = \$1075 \text{ thousand (or 1.075 million)}$$

The predicted price of a 1,500 square foot not on the waterfront is:

$$\widehat{\text{Price}} = -407.7 + 0.4457 \times 1500 = \$260.9 \text{ thousand}$$

Calculations in R:

```
predict(M_wf_sqft, newdata=data.frame(waterfront="Yes", sqft_living=1500))
```

```
1  
1075.227
```

```
predict(M_wf_sqft, newdata=data.frame(waterfront="No", sqft_living=1500))
```

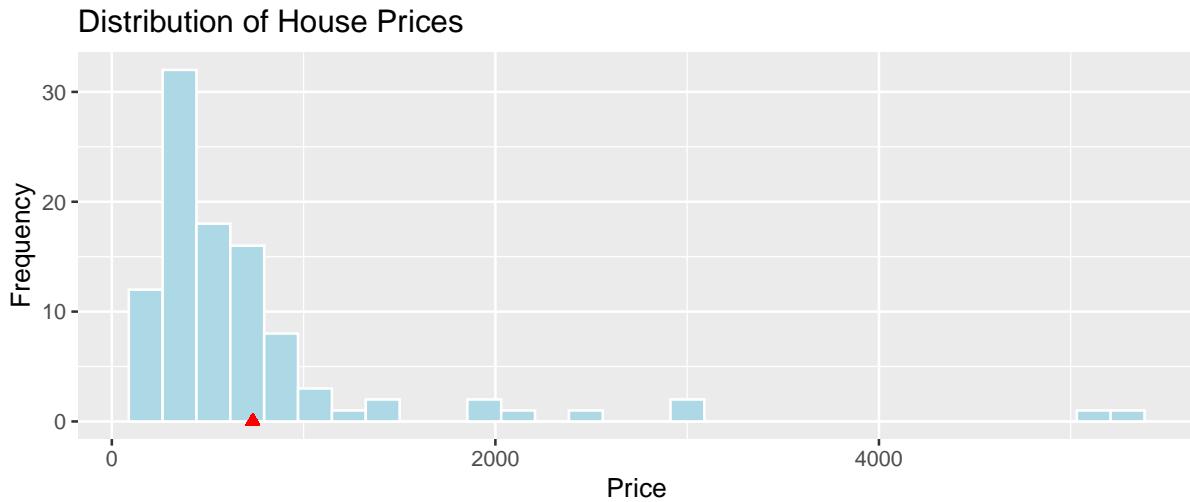
```
1  
260.8657
```

2.1.5 No Explanatory Variable

Finally, we'll consider a model that makes use of no explanatory variables at all. Although this might seem silly, its relevance will be seen in the next section.

The histogram shows the distribution of prices, without any information about explanatory variables. The mean price is indicated in red.

```
ggplot(data=Houses, aes(x=price)) +  
  geom_histogram(fill="lightblue", color="white") +  
  ggtitle("Distribution of House Prices") + xlab("Price") + ylab("Frequency") +  
  geom_point(aes(x=mean(Houses$price), y=0), color="red", shape=24, fill="red")
```



The mean, median, and standard deviation in prices is shown below.

```
library(knitr)  
kable(Houses_Summary)
```

Mean_Price	Median_Price	StDev_Price	Number_of_Houses
735.3525	507.5	835.1231	100

Suppose we know that a house sold in King County during this time, and want to predict the price, without knowing anything else about the house.

The best we can do is to use the mean price for our prediction. (We'll define what we mean by "best" later in the chapter.)

The model equation is

$$\widehat{\text{Price}} = b_0$$

We fit a statistical model in R using the `lm` command.

```
# syntax for lm command
# lm(data=DatasetName, ResponseVariable~ExplanatoryVariable(s))

M0_House <- lm(data=Houses, price ~ 1) # when there are no explanatory variables, use ~1
M0_House
```

Call:

```
lm(formula = price ~ 1, data = Houses)
```

Coefficients:

```
(Intercept)
    735.4
```

The model equation is

$$\widehat{\text{Price}} = 735.4$$

Interpretation

The expected price of a house in King County is 735.4 thousand dollars.

Predictions

Without knowing anything about any explanatory variables, we would predict the price of any house sold in King County, WA to cost 735.4 thousand dollars.

2.2 Variability Explained by a Model

We've seen four different models for predicting house price. It would be nice to have a way to assess how well the models are predicting prices, and determine which model appears to be the best.

Of course we won't know the price of the house we are trying to predict, so we can't be sure how close or far our prediction is. We do, however, know the prices of the original 100 houses in our dataset. We can assess the models by measuring how far the actual prices of the 100 houses differ from the predicted (mean) price, and by calculating the proportion of total variation in sale price explained by each model.

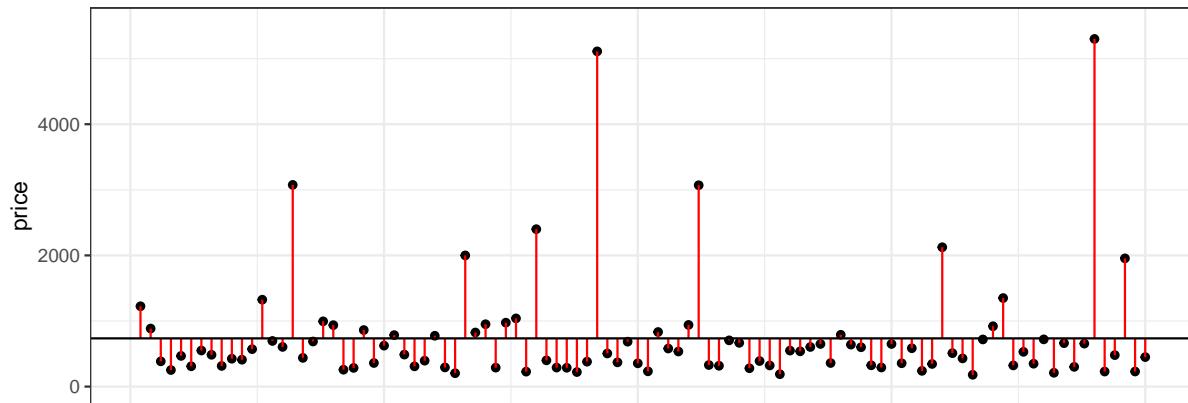
2.2.1 Total Variability

Let's start with our most basic model, which uses no explanatory variables and predicts the price of each simply using the average of all houses in the dataset.

We measure the total variability in the response variable by calculating the square difference between each individual response value and the overall average. This quantity is called the total sum of squares (SST).

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

The plot below shows a horizontal line at the mean sale price (785 thousand). The points represent prices of individual houses, and the red lines represent the differences between the price of each house and the overall average.



The first three houses in the dataset are shown below.

```
First3Houses <- Houses %>% select(Id, price, waterfront, sqft_living) %>% head(3)
kable(First3Houses)
```

Id	price	waterfront	sqft_living
1	1225	No	5420
2	885	No	2830
3	385	No	1620

$$\begin{aligned} SST &= \sum_{i=1}^{100} (y_i - \bar{y})^2 \\ &= (1225 - 785)^2 + (885 - 785)^2 + (385 - 785)^2 + \dots \end{aligned}$$

We could calculate SST by hand for small datasets. For larger datasets, we'll use R to perform the calculation.

```
meanprice <- mean(Houses$price) #calculate mean price
SST <- sum(( Houses$price - meanprice)^2) ## calculate SST
SST
```

```
[1] 69045634
```

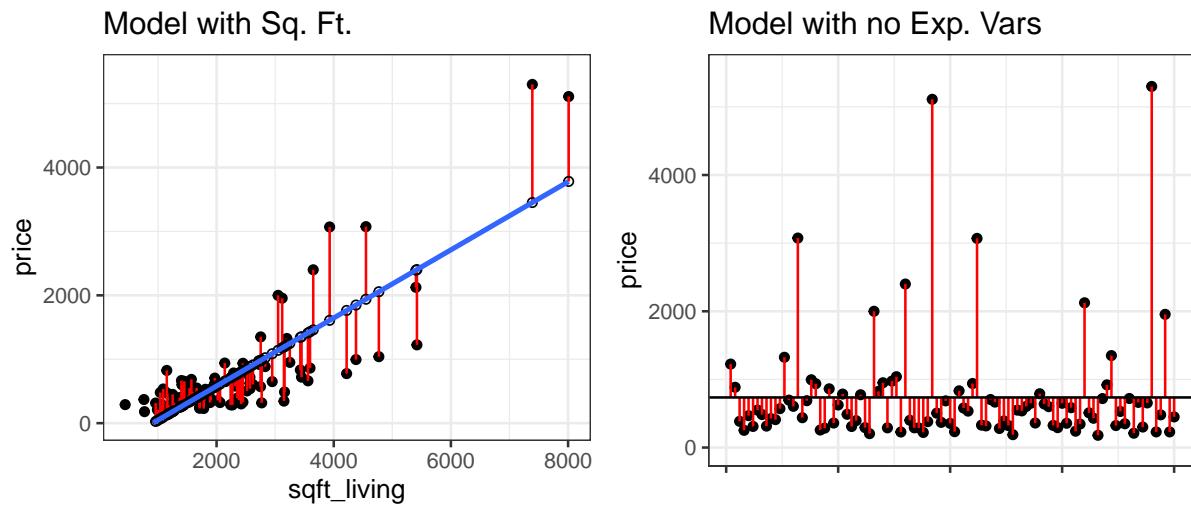
By itself, the size of SST does not have much meaning. We cannot say whether a SST value like the one we see here is large or small, since it depends on the size and scale of the variable being measured. An SST value that is very large in one context might be very small in another.

SST does, however, give us a baseline measure of the total variability in the response variable. We'll assess the performance of a model with a given explanatory variable by measuring how much of this variability the model accounts for.

2.2.2 Residuals

Now let's consider our model that uses the size of the house in square feet as the explanatory variable. The figure on the left shows difference between actual and predicted prices, using this linear model. We compare the size of the differences to those resulting from the basic model that does not use any explanatory variables, and predicts each price using the overall average (shown on the right).

```
Residplot_sqft <- ggplot(data=Houses, aes(x = sqft_living, y = price)) + geom_point() +
  geom_segment(aes(xend = sqft_living, yend = M_House_sqft$fitted.values), color="red") +
  geom_point(aes(y = M_House_sqft$fitted.values), shape = 1) +
  stat_smooth(method="lm", se=FALSE) + ylim(c(0,5500)) +
  theme_bw()
```



Notice that the red lines are shorter in the figure on the left, indicating the predictions are closer to the actual values.

The difference between the actual and predicted values is called the **residual**. The residual for the i th case is

$$r_i = (y_i - \hat{y}_i)$$

We'll calculate the residuals for the first three houses in the dataset, shown below.

```
kable(First3Houses)
```

	Id	price	waterfront	sqft_living
	1	1225	No	5420
	2	885	No	2830
	3	385	No	1620

The model equation is

$$\widehat{\text{Price}} = -484.9575 + 0.5328 \times \text{Sq. Ft}$$

The predicted prices for these three houses are:

$$\widehat{\text{Price}_1} = -484.9575 + 0.5328 \times 5420 = 2402.6 \text{ thousand dollars}$$

$$\widehat{\text{Price}_2} = -484.9575 + 0.5328 \times 2830 = 1022.7 \text{ thousand dollars}$$

$$\widehat{\text{Price}_3} = -484.9575 + 0.5328 \times 1620 = 378.1 \text{ thousand dollars}$$

To calculate the residuals, we subtract the predicted price from the actual price.

$$r_1 = y_1 - \hat{y}_1 = 1225 - 2402.6 = -1177.6 \text{ thousand dollars}$$

$$r_2 = y_2 - \hat{y}_2 = 885 - 1022.7 = -137.7 \text{ thousand dollars}$$

$$r_3 = y_3 - \hat{y}_3 = 385 - 378.1 = 6.9 \text{ thousand dollars}$$

The fact that the first two residuals are negative indicates that these houses sold for less than the model predicts.

The predicted values and residuals from a model can be calculated automatically in R. The predicted values and residuals for the first 5 houses are shown below.

```
Predicted <- predict(M_House_sqft)
head(Predicted, 3)
```

```
1           2           3
2402.5937  1022.7491  378.1113
```

```
Residual <- M_House_sqft$residuals
head(Residual, 3)
```

```
1           2           3
-1177.593665 -137.749128    6.888668
```

2.2.3 Variability Explained by Sq. Ft. Model

The **sum of squared residuals (SSR)** measures the amount of unexplained variability in the response variable after accounting for all explanatory variables in the model.

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

Note that SSR is similar to SST, except we subtract the model's predicted values, rather than the overall average. In the special case of a model with no explanatory variables, the predicted values are equal to the overall average, so SSR is equal to SST.

We calculate SSR for the model using square feet as the explanatory variable.

$$\begin{aligned}\text{SSR} &= \sum_{i=1}^n (y_i - \hat{y}_i)^2 \\ &= (1225 - 2402.6)^2 + (885 - 1022.7)^2 + (385 - 378.1)^2 + \dots\end{aligned}$$

We can calculate the model's SSR directly in R.

```
SSR_sqft <- sum(M_House_sqft$residuals^2)  
SSR_sqft
```

```
[1] 23767280
```

SSR represents the amount of total variability in saleprice remaining after accounting for the house's size in square feet.

The $\text{SSR}=23,767,290$ value is about one third of the SST value of 69,045,634. This means that about 2/3 of the total variability in sale price is explained by the model that accounts for sale price.

The difference ($\text{SST}-\text{SSR}$) represents the variability in the response variable that is explained by the model. This quantity is called the **model sum of squares (SSM)**.

$$\text{SSM} = \text{SST} - \text{SSR}$$

It can be shown that $\text{SSM} = \sum_{i=1}^n (\hat{y}_i - \bar{y})^2$.

The proportion of total variability in the response variable explained by the model is called the **coefficient of determination**, denoted R^2 . We calculate this by dividing SSM by SST.

$$R^2 = \frac{SSM}{SST} = \frac{SST - SSR}{SST}$$

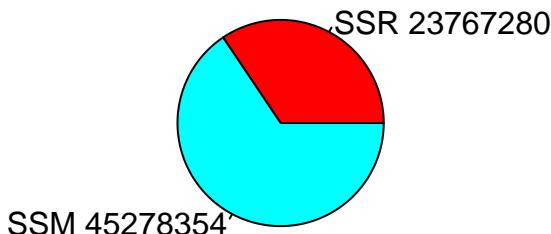
Example: For the model with square feet as the explanatory variable,

$$SSM = SST - SSR = 69,045,634 - 23,767,290 = 45,278,344.$$

$$R^2 = \frac{45,278,344}{69,045,634} = 0.6557.$$

Approximately 65.6% of the total variability in sale price is explained by the model using square feet as the explanatory variable.

Total Variability in House Prices (SST)



We calculate R^2 directly in R.

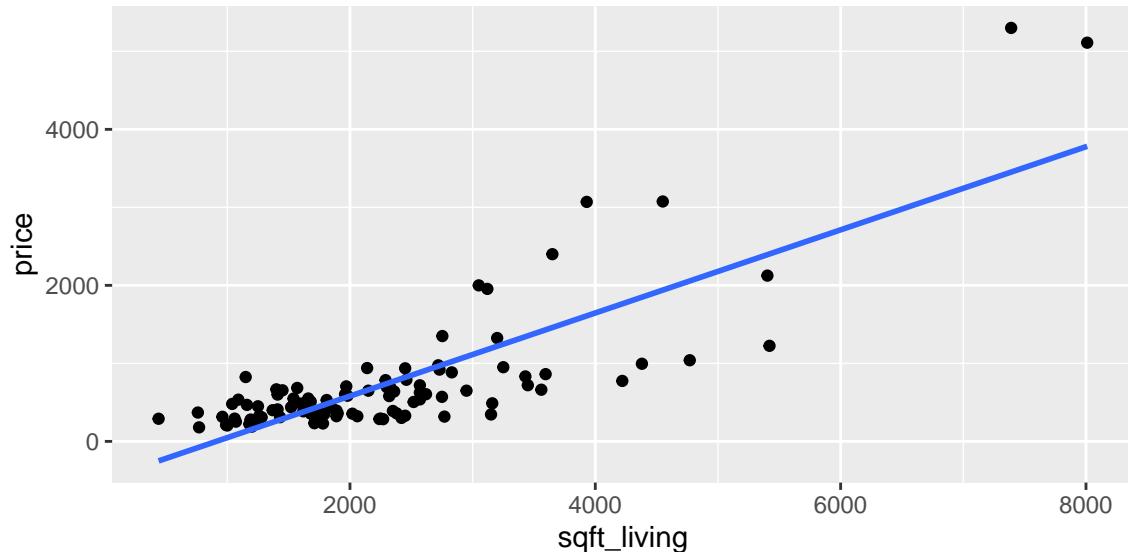
```
summary(M_House_sqft)$r.squared
```

```
[1] 0.6557743
```

2.2.4 Linear Correlation Coefficient

For models with a single quantitative explanatory variable, the coefficient of determination is equal to the square of the correlation coefficient r , discussed in Chapter 1.

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



- 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.

```
cor(Houses$price, Houses$sqft_living)
```

[1] 0.8097989

- 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.

2.2.5 Variability Explained by Waterfront Model

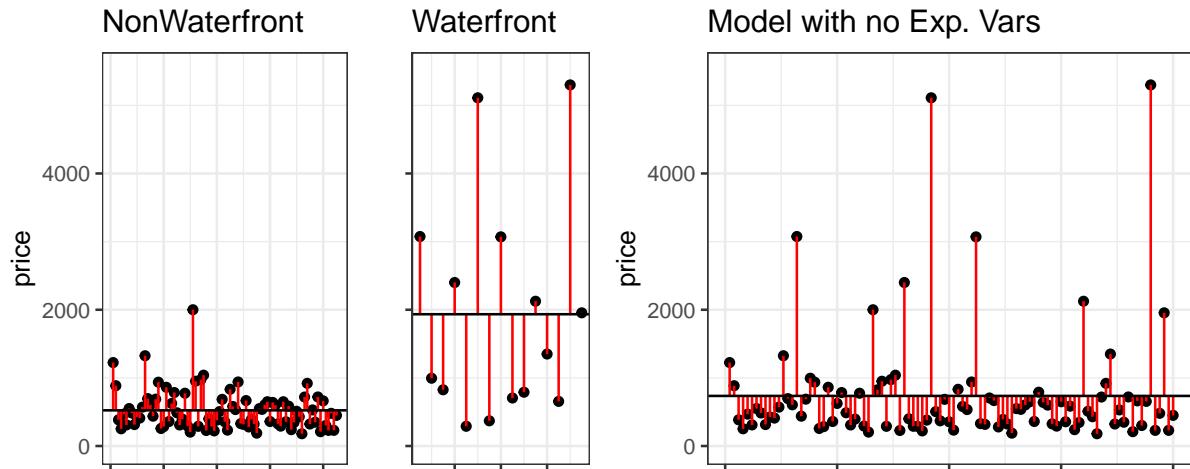
We can similarly calculate the proportion of variability explained by the model using waterfront as an explanatory variable.

Recall that in this model, the predicted price of a house with a waterfront is given by the average price of all waterfront houses, and the predicted price of a non-waterfront house is given by the average price of all non-waterfront houses.

We can calculate residuals using these predicted values, and compare them to the residuals resulting from a model with no explanatory variables, which uses the overall average price for all predictions.

The left two figures show the residuals resulting from a model that accounts for waterfront status. The figure on the right shows the residuals resulting from the model with no explanatory variables.

```
grid.arrange(arrangeGrob(M1aResid,M1bResid, Residplot_M0 + ggtitle("Model with no Exp. Vars")))
```



Notice that after accounting for waterfront status, the differences between observed and predicted values are bigger than they were in the model that accounted for square feet, though not as big as for the model that doesn't use any explanatory variables.

We use R to calculate SSR for the waterfront model.

```
SSR_wf <- sum(M_House_wf$residuals^2)
SSR_wf
```

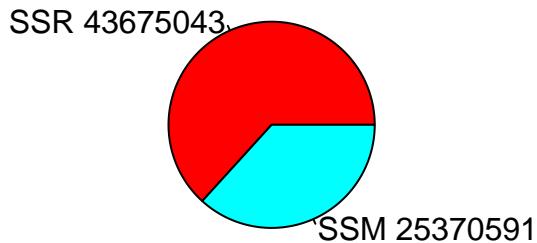
```
[1] 43675043
```

$$SSM = SST - SSR = 69,045,634 - 43,675,043 = 25,370,591.$$

$$R^2 = \frac{25,370,591}{69,045,634} = 0.3674.$$

Approximately 36.7% of the total variability in sale price is explained by the model using waterfront status as the explanatory variable.

Total Variability in House Prices (SST)



We calculate R^2 directly in R.

```
summary(M_House_wf)$r.squared
```

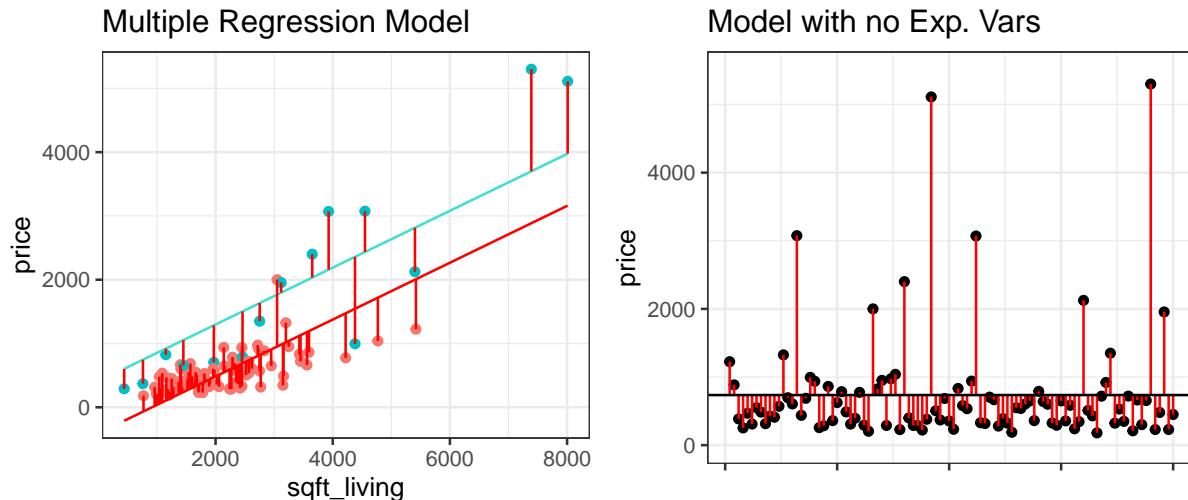
```
[1] 0.3674467
```

2.2.6 Variability Explained by Multiple Regression Model

We've seen that the model using square feet accounts for about 2/3 of the total variability in house prices, while the model using waterfront status accounts for about 1/3 of the total variability. Let's see if we can do better by using both variables together.

The left figure shows the residuals resulting from a model that accounts for both waterfront status and square feet. The figure on the right shows the residuals resulting from the model with no explanatory variables.

```
grid.arrange(Residplot_MR + ggtitle("Multiple Regression Model") , Residplot_M0 + ggtitle("Model with no Exp. Vars"))
```



We use R to calculate SSR for the waterfront model.

```
SSR_wf_sqft <- sum(M_wf_sqft$residuals^2)  
SSR_wf_sqft
```

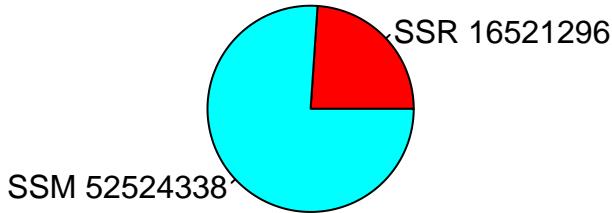
```
[1] 16521296
```

$$SSM = SST - SSR = 69,045,634 - 16,521,296 = 52,524,338.$$

$$R^2 = \frac{52,524,338}{69,045,634} = 0.761.$$

Approximately 76.1% of the total variability in sale price is explained by the model using square feet and waterfront status as the explanatory variables.

Total Variability in House Prices (SST)



We calculate R^2 directly in R.

```
summary(M_wf_sqft)$r.squared
```

```
[1] 0.7607192
```

Including both square feet and waterfront status allows us to explain more variability in sale price than models that include one but not both of these variables.

2.2.7 Summary: SST, SSR, SSM, R^2

- the total variability in the response variable is the sum of the squared differences between the observed values and the overall average.

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

- the variability remaining unexplained even after accounting for explanatory variable(s) in a model 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

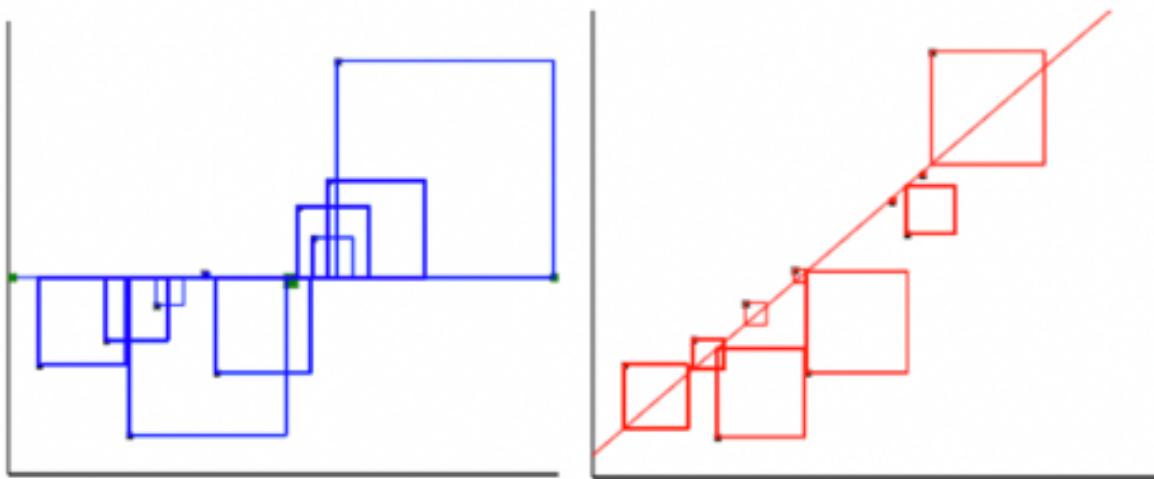
$$\text{SSM} = \text{SST} - \text{SSR}$$

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

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

Note that some texts use different abbreviations than the ones used here. When working with resources outside this class, be sure to carefully check the notation being used.

For the model with a single quantitative explanatory variable.



Model with a single categorical explanatory variable with 3 categories:



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

2.2.8 Model Comparison Summary

Model	Variables	Unexplained Variability	Variability Explained	R^2
0	None	69045634.1341747	0	0
1	Sq. Ft.	23767280.3817707	45278353.752404	0.6557743
2	Waterfront	43675043.0897012	25370591.0444735	0.3674467

Model	Variables	Unexplained Variability	Variability Explained	R^2
3	Sq. Ft. and Waterfront	16521296.4889025	52524337.6452723	0.7607192

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.
- On some datasets, there is a lot of “natural” variability in the response variable, and no model will achieve a high R^2 . That’s okay. Even a model with $R^2 = 0.10$ or less can provide useful information.
- The goal is not to achieve a model that makes perfect predictions, but rather to be able to quantify the amount of uncertainty associated with the predictions we make.

2.3 Models with Interaction

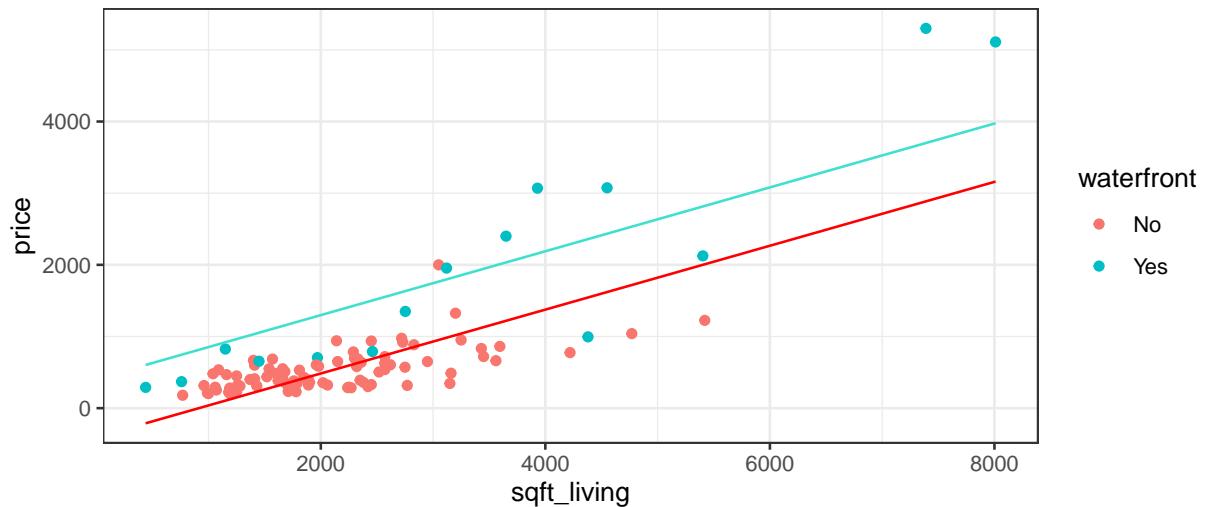
2.3.1 Definition of Interaction

We previously used a multiple regression model of the form

$$\widehat{\text{Price}} = b_0 + b_1 \times \text{SqFt} + b_2 \times \text{Waterfront}$$

Recall that this model assumes the slope relating price and square footage is the same (b_1) for houses on the waterfront as for houses not on the waterfront. An illustration of the model is shown below.

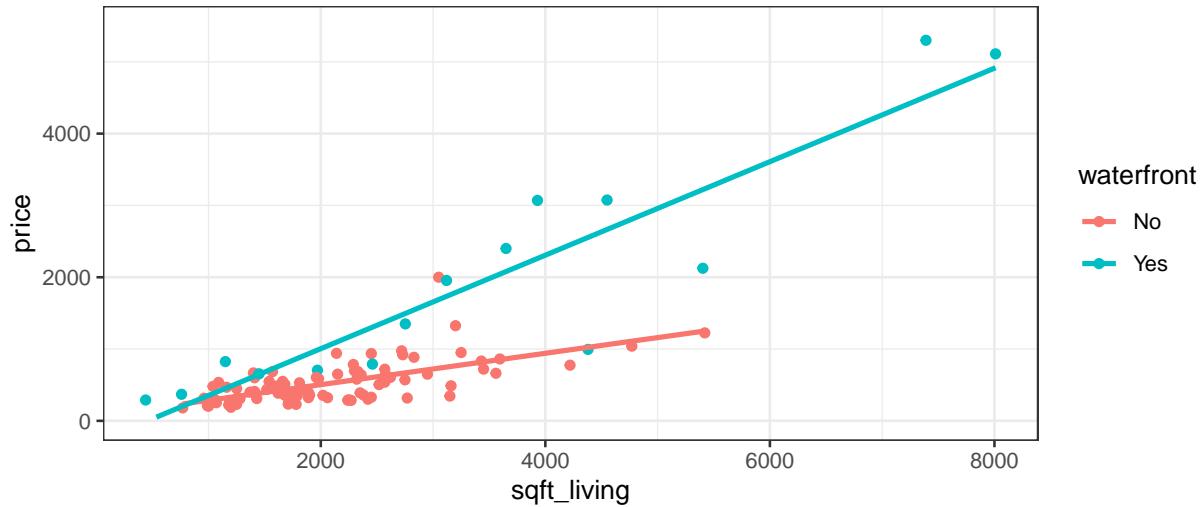
PM3



This assumption of the rate of change in price with respect to living space being the same for waterfront houses, as for non-waterfront houses might be unrealistic.

Let's fit separate lines for waterfront and non-waterfront houses, without requiring them to have the same slope.

```
ggplot(data=Houses, aes(x=sqft_living, y=price, color=waterfront)) + geom_point() + stat_smooth()
```



It appears that the prices of the houses on the waterfront are increasing more rapidly, with respect to square feet of living space, than the non-waterfront houses. The effect of additional square feet on the price of the house appears to depend on whether or not the house is on the waterfront. This is an example of an **interaction** between square footage and waterfront status.

An **interaction** between two explanatory variables occurs when the effect of one explanatory variable on the response depends on the other explanatory variable.

2.3.2 Interaction Term

If we want to allow for different slopes between waterfront and non-waterfront houses, we'll need to change the mathematical equation of our model. To do that, we'll add a coefficient b_3 , multiplied by the product of our two explanatory variables.

The model equation is

$$\widehat{\text{Price}} = b_0 + b_1 \times \text{Sq. Ft.} + b_2 \times \text{waterfront} + b_3 \times \text{Sq.Ft} \times \text{Waterfront}$$

The last term is called an **interaction term**.

For a house on the waterfront ($\text{waterfront} = 1$), the equation relating price to square feet is

$$\begin{aligned}\widehat{\text{Price}} &= b_0 + b_1 \times \text{Sq. Ft.} + b_2 \times 1 + b_3 \times \text{Sq.Ft} \times 1 \\ &= (b_0 + b_2) + (b_1 + b_3) \times \text{Sq. Ft.}\end{aligned}$$

For a house not on the waterfront ($\text{waterfront} = 0$), the equation relating price to square feet is

$$\begin{aligned}\widehat{\text{Price}} &= b_0 + b_1 \times \text{Sq. Ft.} + b_2 \times 0 + b_3 \times \text{Sq.Ft} \times 0 \\ &= b_0 + b_1 \times \text{Sq. Ft}\end{aligned}$$

The intercept is b_0 for non-waterfront houses, and $b_0 + b_2$ for waterfront houses.

The slope is b_1 for non-waterfront houses, and $b_1 + b_3$ for waterfront houses.

Thus, the model allows both the slope and intercept to differ between waterfront and non-waterfront houses.

2.3.3 Interaction Models in R

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

```
M_House_Int <- lm(data=Houses, price~sqft_living*waterfront)
M_House_Int
```

Call:

```
lm(formula = price ~ sqft_living * waterfront, data = Houses)
```

Coefficients:

	(Intercept)	sqft_living
	67.3959	0.2184
waterfrontYes	sqft_living:waterfrontYes	
	-364.5950	0.4327

The regression equation is

$$\widehat{\text{Price}} = 67.4 + 0.2184 \times \text{Sq. Ft.} - 364.6 \times \text{waterfront} + 0.4327 \times \text{Sq.Ft} \times \text{Waterfront}$$

For a house on the waterfront (waterfront = 1), the equation is

$$\begin{aligned}\widehat{\text{Price}} &= 67.4 + 0.2184 \times \text{Sq. Ft.} - 364.6 \times 1 + 0.4327 \times \text{Sq.Ft} \times 1 \\ &= (67.4 - 364.6) + (0.2184 + 0.4327) \times \text{Sq. Ft.} \\ &= -297.2 + 0.6511 \times \text{Sq. Ft.}\end{aligned}$$

For a house not on the waterfront (waterfront = 0), the equation is

$$\begin{aligned}\widehat{\text{Price}} &= 67.4 + 0.2184 \times \text{Sq. Ft.} - 364.6 \times 0 + 0.4327 \times \text{Sq.Ft} \times 0 \\ &= 67.40 + 0.2184 \times \text{Sq. Ft.}\end{aligned}$$

Interpretation

When interpreting b_0 and b_1 , we need to state that the interpretations apply only to the “baseline” category (in this case non-waterfront houses).

In a model with interaction, it does not make sense to talk about holding one variable constant when interpreting the effect of the other, since the effect of one variable depends on the value or category of the other. Instead, we must state the value or category of one variable when interpreting the effect of the other.

Interpretations:

- b_0 - On average, a house with 0 square feet that is not on the waterfront is expected to cost 67 thousand dollars. This is not a sensible interpretation since there are no houses with 0 square feet.
- b_1 - For each additional square foot in size, the price of a non-waterfront house is expected to increase by 0.2184 thousand dollars.
- b_2 - On average, the price of a waterfront house with 0 square feet is expected to be 364.6 thousand dollars less than the price of a non-waterfront house with 0 square feet. This is not a sensible interpretation in this case.
- b_3 - For each additional square foot in size, the price of a waterfront house is expected to increase by 0.4327 thousand dollars more than a non-waterfront house.

Alternatively, we could interpret $b_0 + b_2$ and $b_1 + b_3$ together.

- $b_0 + b_2$ - On average, a house with 0 square feet that is on the waterfront is expected to cost -297.2 thousand dollars. This is not a sensible interpretation since there are no houses with 0 square feet.
- $b_1 + b_3$ - For each additional square foot in size, the price of a waterfront house is expected to increase by 0.6511 thousand dollars.

Prediction

We calculate predicted prices for the following houses:

```
Houses[c(1,16), ] %>% select(Id, price, sqft_living, waterfront)
```

```
# A tibble: 2 x 4
  Id price sqft_living waterfront
  <int> <dbl>      <dbl> <fct>
1     1   1225       5420 No
2    16   3075       4550 Yes
```

$$\widehat{Price}_1 = 67.4 + 0.2184 \times 5420 - 364.6 \times 0 + 0.4327 \times 5420 \times 0 = 1191 \text{ thousand dollars}$$

$$\widehat{Price}_{16} = 67.4 + 0.2184 \times 4450 - 364.6 \times 1 + 0.4327 \times 4450 \times 1 = 2600 \text{ thousand dollars}$$

2.3.4 R^2 for Interaction Model

We can calculate residuals, as well as SSR, SSM, SST, and R^2 , in the same manner we've seen previously.

We'll perform these calculations using R.

```
SSR_int <- sum(M_House_Int$residuals^2)
SSR_int
```

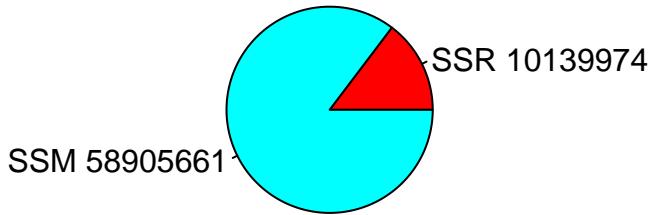
```
[1] 10139974
```

$$SSM = SST - SSR = 69,045,634 - 10,139,974 = 58,905,660.$$

$$R^2 = \frac{58,905,660}{69,045,634} = 0.8531.$$

Approximately 85.3% of the total variability in sale price is explained by the model using square feet and waterfront status, as well as an interaction between them as the explanatory variables.

Total Variability in House Prices (SST)



We calculate R^2 directly in R.

```
summary(M_House_Int)$r.squared
```

```
[1] 0.853141
```

We see that adding an interaction term improved the proportion of variability in house price explained by the model from 0.76 to 0.85. This is a fairly notable increase.

2.3.5 Considerations for Using Interactions

It might be tempting to think we should always add an interaction term to a model when using two or more explanatory variables. After all, an interaction term is just another term added to the model, meaning that R^2 will never go down.

Adding an interaction term is not always a good idea, though. We saw that doing so makes interpretations more complicated. Increasing the complexity of a model also increases the risk of overfitting, potentially hurting predictive performance on new data.

We should only add an interaction term if we have strong reason to believe that the rate of change in the response variable with respect to one explanatory variable really does depend on the other variable. This might come from background knowledge about the subject, or consultation with an expert in the area. It could also come from data visualization, and the increase in variability in the response variable explained when an interaction term is added to the model.

In the house price dataset, we might expect that the price of waterfront houses might increase more rapidly as they get bigger than the price of non-waterfront houses. The fact that the lines shown in the scatterplot are not close to being parallel provides further evidence of a difference in rate of increase, providing justification for the use of an interaction term in the model. Furthermore, R^2 increases notably (from 0.76 to 0.85), when an interaction term is added. All of these reasons support using an interaction term in this context.

When examining a scatterplot, we should note that even if there is truly no interaction among all houses, the lines probably won't be exactly parallel, due to random deviations among the sample of houses chosen. If the lines are reasonably close to parallel, then an interaction term is likely not needed.

We'll look more at criteria for determining whether to add an interaction term to a model in the coming sections.

2.3.6 Interaction vs Correlation

It is easy to confuse the concept of interaction with that of correlation. These are, in fact, very different concepts.

A correlation between two variables means that as one increases, the other is more likely to increase or decrease. We only use the word correlation to describe two quantitative variables, but we could discuss the similar notion of a relationship between categorical variables.

An interaction between two explanatory variables means that the effect of one on the response depends on the other.

Examples of Correlations (or relationships)

1. Houses on the waterfront tend to be bigger than houses not on the waterfront, so there is a relationship between square feet and waterfront status.
2. Houses with large amounts of living space in square feet are likely to have more bedrooms, so there is a correlation between living space and bedrooms.
3. Suppose that some genres of movies (drama, comedy, action, etc.) tend to be longer than others. This is an example of a relationship between genre and length.

The fact that there is a correlation between explanatory variables is NOT a reason to add an interaction term involving those variables in a model. Correlation is something entirely different than interaction!

Examples of Interactions

1. As houses on the waterfront increase in size, their price increases more rapidly than for houses not on the waterfront. This means there is an interaction between size and waterfront location.
2. Suppose that the effect of additional bedrooms on price is different for houses with lots of living space than for houses with little living space. This would be an example of an interaction between living space and number of bedrooms.

3. Suppose that audiences become more favorable to dramas as they get longer, but less favorable to comedies as they get longer. In this scenario, the effect of movie length on audience rating depends on the genre of the movie, indicating an interaction between length and genre.

2.4 Least Squares Estimation (LSE)

2.4.1 Estimating Regression Coefficients

We've already used R to determine the estimates of b_0, b_1, b_2 , and b_3 in various kinds of linear models. At this point, it is natural to wonder where these estimates are come from.

Regression coefficients b_0, b_1, \dots, b_p are chosen in a way that minimizes the sum of the squared differences between the observed and predicted values. That is, we minimize

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

Because \hat{y}_i is a function of b_0, b_1, \dots, b_p , we can choose the values of b_0, b_1, \dots, b_p in a way that minimizes SSR.

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

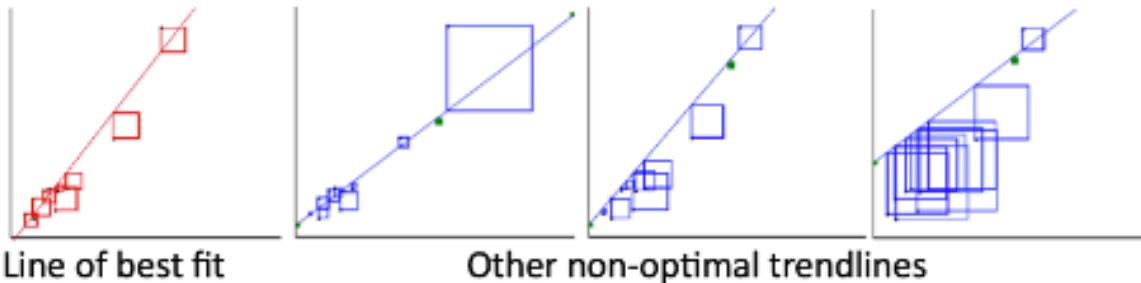
The process of estimating regression coefficients b_0, b_1, \dots, b_p in a way that minimizes SSR is called **least-squares estimation**.

Example: Model with one quantitative variable

We start with an example of estimating the regression coefficients for a model with a single explanatory variable. This is easy to illustrate, since we can draw a scatter plot displaying our explanatory and response variable.

The figure below illustrates four possible trend lines that could be fit to a set of 10 points in a scatter plot. The first line is the line of best fit, in that it makes the sum of the squared residuals the smallest of all possible lines that could be drawn. The second through fourth plots all show examples of other trend lines that are not the line of best fit. The sum of squared residuals for each of these models is bigger than for the first one.

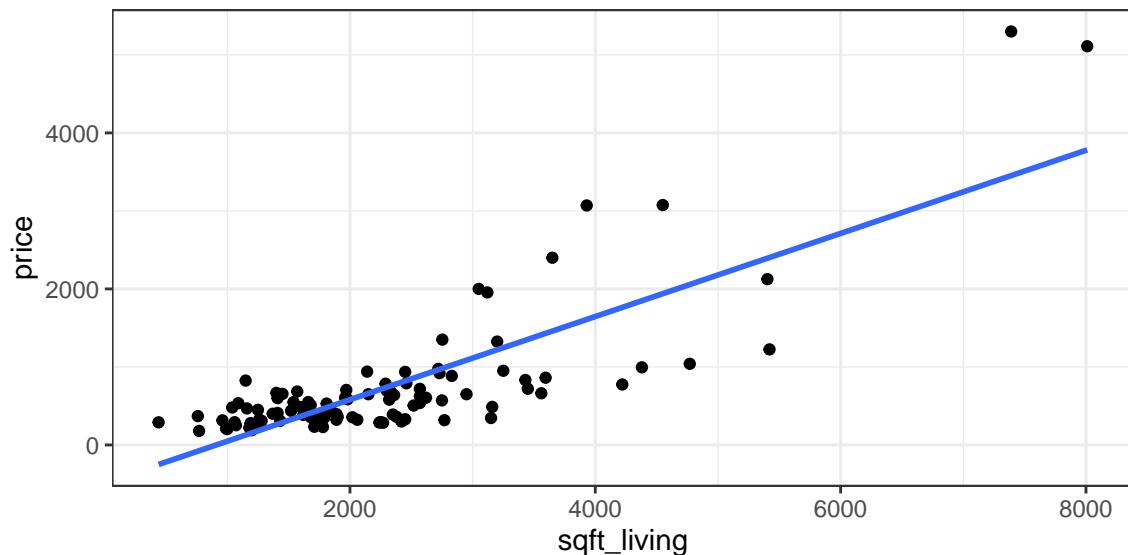
In the illustration, SSR is represented by the total area of the squares. The line of best fit is the one that make the intercept the smallest.



- This [Rossman-Chance applet](#) provides an illustration of the line of best fit.

Returning to the model for predicting price of a house, using only size in square feet as an explanatory variable, the scatter plot, along with the slope and intercept of the regression line are shown below.

```
ggplot(data=Houses, aes(x=sqft_living, y=price)) + geom_point() +
  stat_smooth(method="lm", se=FALSE) + theme_bw()
```



```
M_House_sqft
```

```
Call:
lm(formula = price ~ sqft_living, data = Houses)

Coefficients:
(Intercept)  sqft_living
-484.9575      0.5328
```

The line $\text{Price} = -485 + 0.5328 \times \text{Square Feet}$ is the “line of best fit” in the sense that it minimizes the sum of the squared residuals (SSR). Any other choices for the slope or intercept of the regression line would result in larger SSR than this line.

2.4.2 Mathematics of LSE for SLR

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

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

We setup the equation by substituting in the values of y_i and x_i seen in the data.

Recall the first 3 houses in the dataset:

```
kable(First3Houses)
```

Id	price	waterfront	sqft_living
1	1225	No	5420
2	885	No	2830
3	385	No	1620

$$\begin{aligned} \sum_{i=1}^{100} (y_i - \hat{y}_i)^2 &= \sum_{i=1}^n (y_i - (b_0 + b_1 x_i))^2 \\ &= (1225 - (b_0 + b_1(5420)))^2 + (885 - (b_0 + b_1(2830)))^2 + (385 - (b_0 + b_1(1620)))^2 + \dots \end{aligned}$$

We need to find the values of b_0 and b_1 that minimize this expression. This is a 2-dimensional optimization problem that can be solved using multivariable calculus or numerical or graphical methods.

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

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

2.4.3 LSE for Categorical Variable

- Consider a model with a single categorical variable (such as waterfront), 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
 - $b_j = \bar{y}_j - \bar{y}_0$

2.4.4 LSE More Generally

- For multiple regression models, including those involving interaction, 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](#).

2.5 ANalysis Of VAriance

2.5.1 Submodels

We've seen 5 different models for predicting house price using some combination of square feet and waterfront status.

A model A is defined to be a **submodel** of another model B, if every term in model A is also included in model B.

Model	Variables	Unexplained Variability	Variability Explained	R^2
0	None	69045634	0	0
1	Sq. Ft.	23767280	45278354	0.656
2	Waterfront	43675043	25370591	0.367
3	Sq. Ft. and Waterfront	16521296	52524338	0.761
4	Sq. Ft., Waterfront, and Interaction	10139974	58905661	0.853

- 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.
- Models 1, 2, and 3 are all submodels of Model 4.
- Model 0 is a submodel of Models 1, 2, 3, and 4.
- 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.

2.5.2 F-Statistics

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 = # terms in Full Model, not including the intercept

q = # terms in Reduced Model, not including the intercept

n = number of observations

We calculate a statistic called F that measures the amount of variability explained by adding additional variable(s) to the model, relative to the total amount of unexplained variability.

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

- 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” depends on n , p , and q . We will revisit this later in the course.

Example 1

Let's Calculate an ANOVA F-Statistic to compare Models 1 and 3.

Reduced Model:

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

Full Model:

$$\widehat{\text{Price}} = b_0 + b_1 \times \text{sqft_living} + b_2 \times \text{Waterfront}$$

$$\begin{aligned} F &= \frac{\frac{\text{SSR}_{\text{Reduced}} - \text{SSR}_{\text{Full}}}{p-q}}{\frac{\text{SSR}_{\text{Full}}}{n-(p+1)}} \\ &= \frac{\frac{23,767,280 - 16,521,296}{2-1}}{\frac{16,521,296}{100-(2+1)}} \end{aligned}$$

```
((SSR_sqft-SSR_wf_sqft)/(2-1))/((SSR_wf_sqft)/(100-(2+1)))
```

```
[1] 42.54269
```

We can calculate the statistic directly in R, using the `anova` command.

```
anova(M_House_sqft, M_wf_SqFt)$F[2]
```

```
[1] 42.54269
```

In the coming chapters, we'll talk about what to conclude from an F-statistic of 42.5. Is this big enough to say that adding waterfront status to a model already including square feet helps better explain variability in sale price? (Spoiler alert: YES - an F-statistic of 42.5 is quite large and indicative that the full model is a better choice than the reduced model.) We previously saw that the model including both square feet and waterfront status had a R^2 value considerably higher than the one including only square feet. This large F-statistic is further evidence to the benefit of considering both variables in our model.

Example 2

We'll calculate an F-statistic to compare Models 3 and 4. This can help us determine whether it is worthwhile to include an interaction term in our model.

Reduced Model:

$$\widehat{\text{Price}} = b_0 + b_1 \times \text{sqft_living} + b_2 \times \text{Waterfront}$$

Full Model:

$$\widehat{\text{Price}} = b_0 + b_1 \times \text{sqft_living} + b_2 \times \text{Waterfront} + b_3 \times \text{sqft_living} \times \text{Waterfront}$$

$$F = \frac{\frac{\text{SSR}_{\text{Reduced}} - \text{SSR}_{\text{Full}}}{p-q}}{\frac{\text{SSR}_{\text{Full}}}{n-(p+1)}}$$
$$= \frac{\frac{16,521,296 - 10,139,974}{3-2}}{\frac{10,139,974}{100-(3+1)}}$$

```
((SSR_wf_sqft-SSR_int)/(3-2))/((SSR_int)/(100-(3+1)))
```

```
[1] 60.41505
```

We can calculate the statistic directly in R, using the `anova` command.

```
anova(M_wf_SqFt, M_House_Int)$F[2]
```

```
[1] 60.41505
```

We observe an F-statistic of 60, which is even bigger than the one seen previously! This suggests that adding the interaction term does indeed improve the model's ability to account for variability in prices.

2.5.3 Comparing 3 or More Categories

F-statistics are commonly used when making comparisons involving categorical variables with 3 or more categories.

One variable in the houses dataset, which we haven't looked at yet, is the condition of the house at the time of sale. The table shows the number of houses in each condition listed.

```
summary(Houses$condition)
```

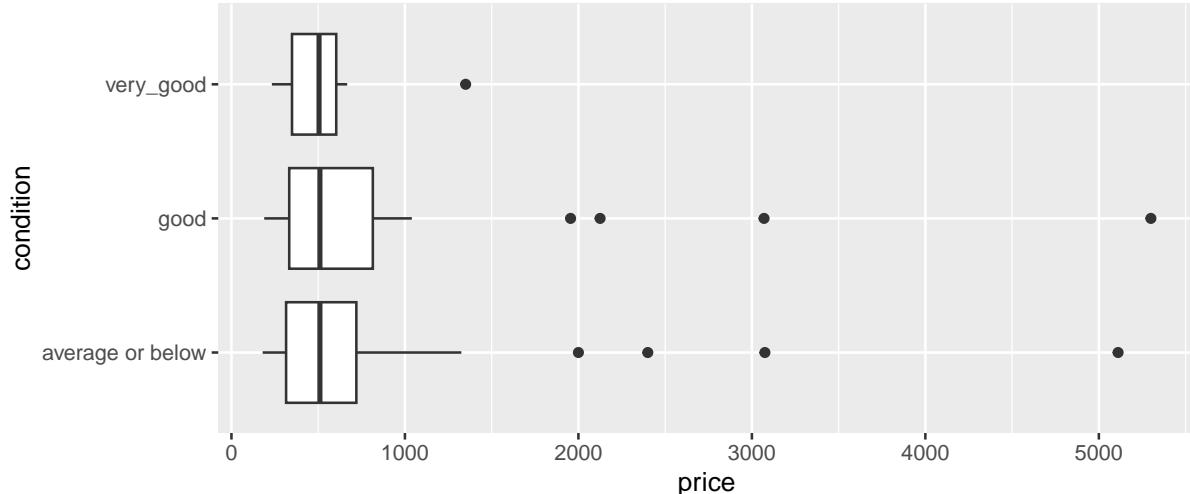
average or below	good	very_good
61	30	9

We notice that there is only one house in poor condition and one house in fair condition. These sample sizes are too small to analyze. We'll combine these two houses with those in the "average" category, creating a new category called "average or below".

```
Houses$condition <- fct_collapse(Houses$condition, "average or below" = c("poor","fair", "av
```

The boxplot shows the distribution of houses in each category, and the table below it provides a numerical summary.

```
ggplot(data=Houses, aes(x=condition, y=price)) + geom_boxplot() +coord_flip()
```



```
Cond_Tab <- Houses %>% group_by(condition) %>% summarize(Mean_Price = mean(price),  
SD_Price= sd (price),  
N= n())  
kable(Cond_Tab)
```

condition	Mean_Price	SD_Price	N
average or below	700.6349	768.1179	61
good	861.0000	1048.9521	30
very_good	551.8361	332.8597	9

It can be helpful to calculate a single statistic that quantifies the size of the differences between the conditions. If we were just comparing two different categories, we could simply find the difference in mean prices between them. But, with three or more categories, we need a way to represent the size of the differences with a single number. An F-statistic can serve this purpose.

We'll calculate an F-statistic for a model that includes condition, compared to a model with only an intercept term.

Reduced Model:

$$\widehat{\text{Price}} = b_0$$

Full Model:

$$\widehat{\text{Price}} = b_0 + b_1 \times \text{good condition} + b_2 \times \text{very good condition}$$

Notice that the equation includes separate variables for the “good” and “very” good conditions. These variables take on value 0 if the house is not in that condition, and 1 if the house is in that condition. Here, houses in “average or below” condition are considered the “baseline” category.

We'll fit the model in R. The coefficient estimates for b_0 , b_1 and b_2 are shown below.

```
M_House_Cond <- lm(data=Houses, price~condition)
M_House_Cond
```

Call:
`lm(formula = price ~ condition, data = Houses)`

Coefficients:

(Intercept)	conditiongood	conditionvery_good
700.6	160.4	-148.8

The model equation is

$$\widehat{\text{Price}} = b_0 + b_1 \times \text{good condition} + b_2 \times \text{very good condition}$$

Interpretations

- On average, houses in average or below condition cost 700.6 thousand dollars.
- On average, houses in good condition cost 160.4 thousand dollars more than those in average or below condition.
- On average, houses in very good condition cost 148.8 thousand dollars less than those in average or below condition.

This last sentence is surprising and merits further investigation. We'll leave that for future consideration.

For now, we'll calculate an F-statistic based on the models.

Note that in this case, the reduced model does not include any explanatory variables, so SSR is equal to SST, which we calculate previously.

```
SST
```

```
[1] 69045634
```

We calculate SSR for the full model.

```
SSR_cond <- sum(M_House_Cond$residuals^2)
SSR_cond
```

```
[1] 68195387
```

$$F = \frac{\frac{SSR_{Reduced} - SSR_{Full}}{p-q}}{\frac{SSR_{Full}}{n-(p+1)}}$$

$$= \frac{\frac{69,045,634 - 68,195,387}{2-0}}{\frac{68,195,387}{100-(2+1)}}$$

```
((SST - SSR_cond)/(2-0))/(SSR_cond/(100-(2+1)))
```

```
[1] 0.6046888
```

We perform the calculation directly in R.

```
anova(M_House_Cond, M0_House)$F[2]
```

```
[1] 0.6046888
```

Notice that the F-statistic of 0.6 is considerably smaller than the F-statistics we've seen previously.

This indicates that adding condition to a model with no other explanatory variables doesn't seem to help improve the model's ability to account for variation in price. Put another way, there doesn't appear to be much evidence of difference in price between houses in the different conditions.

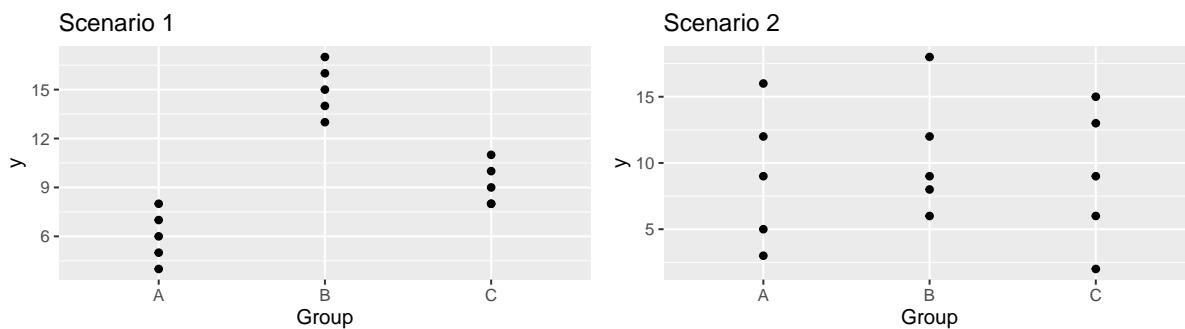
2.5.4 F-Statistic Illustration

The figure below gives an illustration of data that would produce a large F-statistic (Scenario 1), and also data that would produce a small F-statistic (Scenario 2), like the one seen in the house condition data.

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

In scenario 1, we notice considerable differences between the groups, relative to the amount of variability within groups. In this scenario, knowing the group an observation is in will help us predict the response for that group, so we should include account for the groups in our model. We would obtain a large F-statistic when comparing a model that includes group to one that contains only an intercept term.

In scenario 2, there is little difference between the overall averages in each group, and more variability between individual observations within each group. In a scenario like this, knowing the group an observation lies in does little to help us predict the response. In this scenario, predictions from a model that includes group as an explanatory variable would not be much better than those from a model that does not. Hence, we would obtain a small F-statistic.



	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

2.5.5 Alternative F-Statistic Formula

The above illustration suggests alternative (and mathematically equivalent) way to calculate the F-statistic. We calculate the ratio of variability between different groups, relative to the amount of variability within each group

For a categorical variable with g groups,

- let $\bar{y}_1, \dots, \bar{y}_g$ represent the mean response for each group.
- let n_1, \dots, n_g represent the sample size for each group

- Then $\frac{\sum_{i=1}^g \sum_{j=1}^{n_i} n_i (\bar{y}_{i\cdot} - \bar{y}_{..})^2}{g-1}$ 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
- An alternative formula for this F-statistic is:

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

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

Example

Let's recalculate the F-statistic for the conditions of the houses, using this alternate formula. The first 3 houses are shown.

```
kable(head(Houses %>% select(Id, price, condition), 3))
```

Id	price	condition
1	1225	average or below
2	885	average or below
3	385	good

We have seen previously that:

- $\bar{y}_{..} = 735.3526$ (overall average price), and $n = 10$
- $\bar{y}_1 = 700.6349$ (average price for average or below houses), and $n_1 = 61$
- $\bar{y}_2 = 861.0$ (average price for good houses), and $n_2 = 30$
- $\bar{y}_3 = 551.8361$ (average price for very good houses), and $n_3 = 9$

Then,

$$\begin{aligned} \bullet \frac{\sum_{i=1}^g \sum_{j=1}^{n_i} (y_{ij} - \bar{y}_{..})^2}{\frac{850247.3}{2}}, \text{ and} \\ \bullet \frac{\sum_{i=1}^g \sum_{j=1}^{n_i} (y_{ij} - \bar{y}_i)^2}{\frac{68195387}{97}} = \frac{(1225.0 - 700.6349)^2 + (885.0 - 700.6349)^2 + (385.0 - 861.0)^2 + \dots}{100-3} = \frac{68195387}{97} \end{aligned}$$

$$F = \frac{\frac{\sum_{i=1}^g \sum_{j=1}^{n_i} n_i (\bar{y}_i - \bar{y}_{..})^2}{\frac{61(700.6349 - 735.3526)^2 + 30(861.0 - 735.3526)^2 + 9(551.8361 - 735.3526)^2}{3-1}}}{\frac{\sum_{i=1}^g \sum_{j=1}^{n_i} (y_{ij} - \bar{y}_i)^2}{\frac{(1225.0 - 700.6349)^2 + (885.0 - 700.6349)^2 + (385.0 - 861.0)^2 + \dots}{100-3}}} = \frac{\frac{850247.3}{2}}{\frac{68195387}{97}}$$

- 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:

```
((61*(700.6349-735.3526)^2+30*(861.0-735.3526)^2+9*(551.8361-735.3526)^2)/(3-1))/((SSR_cond))
```

```
[1] 0.6046889
```

For models with only one categorical explanatory variable, “variability within vs variability between” interpretation of an F-statistic is very popular. This statistic is often relevant in studies in the natural and social sciences. Such studies are often referred to as One-Way ANOVA’s. In fact, these are just a special case of the full vs reduced model interpretation of the F-statistic, which can be applied to any two models, as long as one is a submodel of the other.

3 Simulation-Based Inference

Learning Outcomes:

14. Define population, sample, and sampling distribution, and a parameter and statistic.
15. Interpret standard deviation and standard error in context.
16. Explain how to use bootstrapping to calculate confidence intervals.
17. Interpret confidence intervals in context.
18. Calculate confidence intervals in R.
19. Write null and alternative hypotheses, identify test statistics, and state conclusions in context.
20. Interpret p-values in context.
21. Explain how to use simulation to perform hypothesis tests.
22. Compare and contrast the conclusions we can draw from confidence intervals and hypothesis tests.
23. Perform hypothesis tests in R.

3.1 Sampling Distributions

3.1.1 Population and Sample

In statistics, we often do not have the time, money, or means to collect data on all individuals or units on which we want to draw conclusions. Instead, we might collect data on only a subset of the individuals, and then make inferences about all individuals we are interested in, using the information we collected.

Vocabulary:

1. A **population** is the entire set of individuals that we want to draw conclusions about.

2. A **sample** is a subset of a population.
3. A **parameter** is a numerical quantity pertaining to an entire population or process.
4. A **statistic** is a numerical quantity calculated from a sample.

We'll work with a dataset containing information on all 20,591 flights from New York to Chicago in 2013. Our population of interest is all 20,591 flights. We're interested in the proportion of flights that arrive on time, and the average arrival delay (in minutes). Arrival delay represents how much earlier/later did the flight arrive than expected. Whether or not the flight arrived on time is a categorical variable, while arrival delay is a quantitative variable.

In this situation, we have information on the entire population. In statistics, this is rare. It is more common to have information on only subset of flights contained in a sample. If the sample is collected in a way that is representative of the population, such as by sampling at random, then we can use the sample to draw conclusions about the population.

We'll begin by studying the behavior of sample statistics when we know the population parameters, and then use what we learn to handle more real situations where we don't know about the entire population.

The parameter of interest is the proportion of on-time arrivals out of all flights in the population of 20,591. When the parameter is proportion, we'll denote it with the letter p .

The first 10 flights, all of which occurred on January 1 are shown below. We see that most of those flights were not on-time.

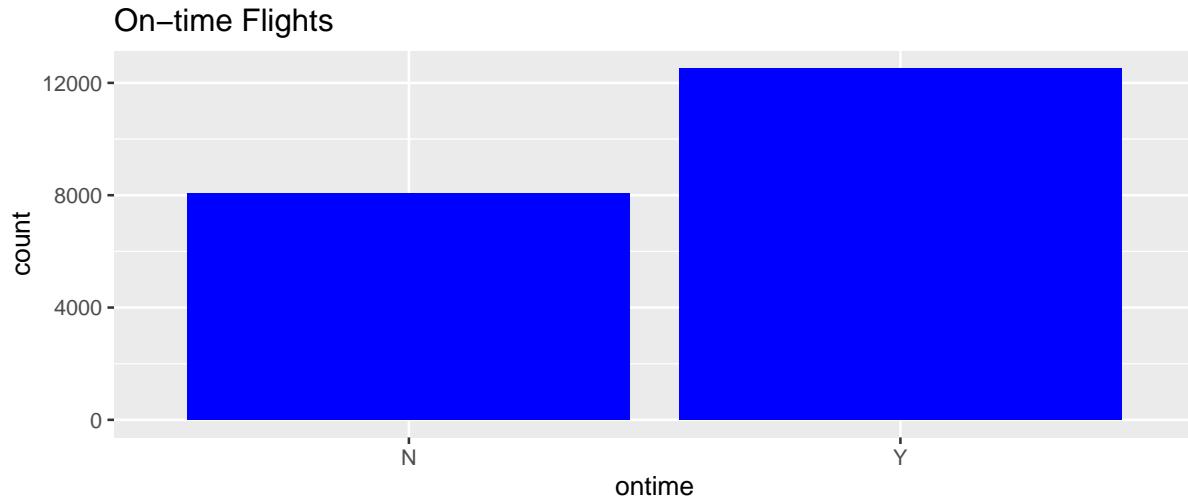
```
head(Flights_NY_CHI, 10)
```

```
# A tibble: 10 x 9
  year month   day carrier origin dest sched_dep_time arr_delay ontime
  <int> <int> <int> <chr>   <chr> <chr>          <int>     <dbl> <chr>
1 2013     1     1  UA      EWR    ORD        558       12  N
2 2013     1     1  AA      LGA    ORD        600        8  N
3 2013     1     1  MQ      EWR    ORD        600       32  N
4 2013     1     1  AA      LGA    ORD        630       14  N
5 2013     1     1  AA      LGA    ORD        700        4  N
6 2013     1     1  UA      LGA    ORD        700       20  N
7 2013     1     1  UA      EWR    ORD        713       21  N
8 2013     1     1  AA      LGA    ORD        745      -12  Y
9 2013     1     1  MQ      EWR    ORD        710       49  N
10 2013    1     1  B6     JFK    ORD        830       15  N
```

Note that a negative arrival delay denotes a flight that arrived before expected, thus on time.

The bar graph shows the number of flights that arrived on time throughout the year.

```
on_time_plot_POP <- ggplot(data=Flights_NY_CHI, aes(x=ontime)) +  
  geom_bar(fill="blue") +  
  ggtitle("On-time Flights")  
on_time_plot_POP
```



We see that the majority of flights did arrive on time.

We'll calculate the proportion of flights arriving on time, among all 20,591 flights in the population.

```
#proportion of flights on time  
p <- sum(Flights_NY_CHI$ontime=="Y")/20591  
p
```

```
[1] 0.6079841
```

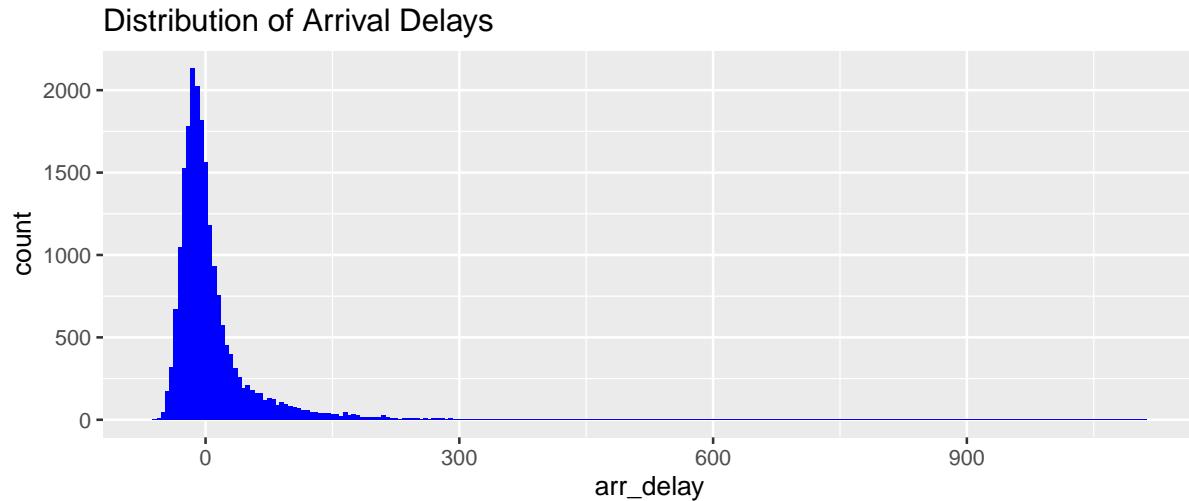
When the population parameter is a proportion, we'll denote it with the letter p . Here $p=0.6079841$. Keep in mind that in a real situation, we typically won't know the value of the population parameter p , and will need to estimate it from a sample.

The histogram shows the distribution of arrival delay times. Negative delays indicate the flight arriving ahead of schedule.

```

Delay_plot_POP <- ggplot(data=Flights_NY_CHI, aes(x=arr_delay)) +
  geom_histogram(fill="blue", binwidth=5) +
  ggtitle("Distribution of Arrival Delays")
Delay_plot_POP

```



We see that the distribution of arrival delays is heavily right-skewed. While most flights arrive around the scheduled time, a few were late by 3 or more hours.

We'll calculate the mean arrival delay.

```

#proportion of flights on time
mu <- mean(Flights_NY_CHI$arr_delay)
mu

```

```
[1] 7.144772
```

When the population parameter represents a mean, we'll denote it using μ . Here $\mu = 7.144772$.

We also calculate the standard deviation of arrival delays.

```

# mean arrival delay in sample
SD_delay <- sd(S1$arr_delay)
SD_delay

```

```
[1] 60.10419
```

3.1.2 Sampling Variability

We typically won't have data on the full population and won't know the values of parameters like p and μ . Instead, we'll have data on just a sample taken from the population.

To illustrate, we'll take a sample of 75 flights. The first 6 flights in the sample are shown below. The `ontime` variable tells whether or not the flight arrived on time.

```
# take sample of 75 flights
set.seed(08082023)
S1 <- sample_n(Flights_NY_CHI, 75)
head(S1)
```

```
# A tibble: 6 x 9
  year month   day carrier origin dest sched_dep_time arr_delay ontime
  <int> <int> <int> <chr>   <chr>  <chr>          <int>      <dbl> <chr>
1 2013     3     8 AA       LGA    ORD        1720        106 N
2 2013    12    15 AA       JFK    ORD        1715       -24 Y
3 2013    10    22 UA       EWR    ORD        1300       -12 Y
4 2013     8    26 UA       EWR    ORD        2110        15 N
5 2013     7    23 AA       LGA    ORD        1359        66 N
6 2013     5    13 UA       EWR    ORD        900       -21 Y
```

We'll calculate the number, and proportion of flights that arrived on time.

```
num_ontime <- sum(S1$ontime == "Y") # count number of on-time arrivals
num_ontime
```

```
[1] 39
```

Proportion of on-time arrivals in the sample.

```
# proportion of on-time flights in sample
p_hat <- num_ontime/75
p_hat
```

```
[1] 0.52
```

When the sample statistic is a proportion, it is commonly denoted \hat{p} .

In our sample $\hat{p} = 52$ percent of flights arrived on-time. The sample statistic \hat{p} is an estimate of the population proportion p , the proportion of all flights arriving on time.

We also calculate the mean arrival delay in minutes.

```
# mean arrival delay in sample
y_bar <- mean(S1$arr_delay)
y_bar
```

```
[1] 19.2
```

We'll denote this sample mean \bar{y} . It is an estimate of the population mean, representing the arrival mean delay for all flights, which we'll denote μ .

Of course, this was just one sample of 75 flights. If we took different samples of 75 flights, we would expect the statistics \hat{p} and \bar{x} to vary from sample to sample.

Here's a different sample of 75 flights.

```
S2 <- sample_n(Flights_NY_CHI, 75)
```

Proportion arriving on time in second sample:

```
# proportion arriving on time in second sample
num_ontime2 <- sum(S2$ontime == "Y") # count number of on-time arrivals
p_hat2 <- num_ontime2/75
p_hat2
```

```
[1] 0.5066667
```

Mean arrival delay in second sample:

```
# mean arrival delay in second sample
y_bar2 <- mean(S2$arr_delay)
y_bar2
```

```
[1] 15.28
```

Sample statistics will vary from sample to sample, thus it is not realistic to expect them to exactly match their corresponding population parameters.

Nevertheless we can use the sample to estimate the proportion of all flights in the population that arrive on time.

Let's take 10,000 more random samples of 75 flights and record the proportion of on-time arrivals in each sample.

```
nreps <- 10000 # number of repetitions
p_hat_val <- rep(NA, nreps) # create vector to hold proportion of on-time arrivals
y_bar_val <- rep(NA, nreps) # create vector to hold mean arrival delay

Sample <- 1:nreps

for(i in 1:nreps){
  S <- sample_n(Flights_NY_CHI, 75) # take sample of 75
  N_ontime <- sum(S$ontime == "Y") # count number of on-time arrivals
  p_hat_val[i] <- N_ontime/75 # record proportion on-time
  y_bar_val[i] <- mean(S$arr_delay) # record mean arrival delay
}

Samples_df <- data.frame(Sample, p_hat_val, y_bar_val) # store results in a data frame
```

The table shows the proportion of on-time arrivals in the first 20 samples of 75 flights.

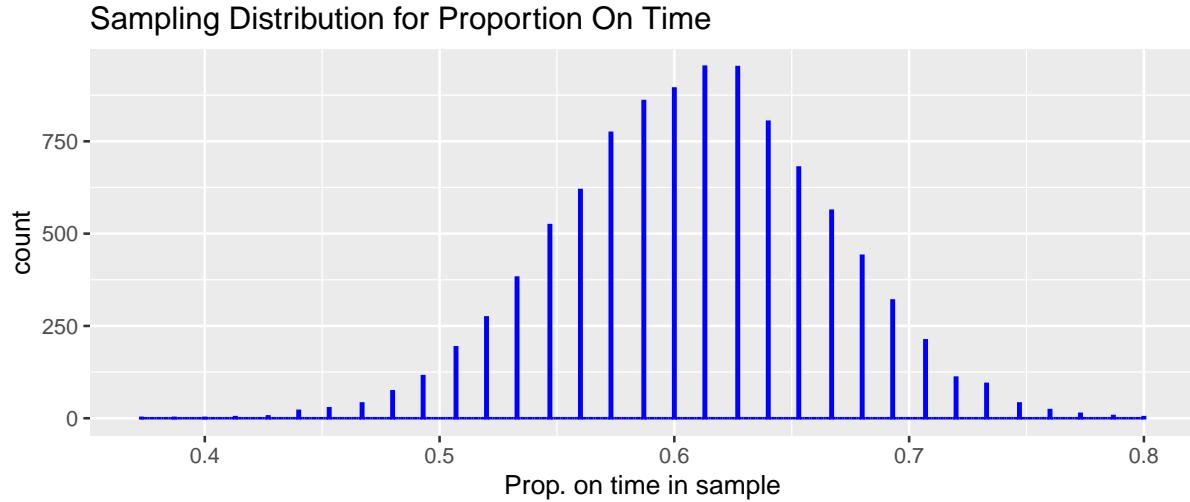
```
kable(head(Samples_df, 20) |> round(2))
```

Sample	p_hat_val	y_bar_val
1	0.65	6.16
2	0.53	10.75
3	0.59	12.41
4	0.69	2.24
5	0.59	7.33
6	0.67	1.28
7	0.61	3.25
8	0.59	8.93
9	0.65	-3.16
10	0.65	5.73
11	0.61	12.32
12	0.59	8.79
13	0.61	5.53

Sample	p_hat_val	y_bar_val
14	0.48	10.55
15	0.60	5.40
16	0.44	24.11
17	0.49	15.93
18	0.57	4.45
19	0.64	6.76
20	0.56	3.65

The histogram below shows the distribution of the proportion of on-time arrivals in the 10,000 different samples.

```
Prop_Samp_Dist<- ggplot(data=Samples_df, aes(x=p_hat_val)) +
  geom_histogram(color="blue", fill="blue", binwidth=0.001) +
  ggtitle("Sampling Distribution for Proportion On Time") +
  xlab("Prop. on time in sample")
Prop_Samp_Dist
```



We notice that most of our 10,000 samples yielded proportions of on-time arrivals between 0.5 and 0.7. The distribution of proportion of on-time arrivals is roughly symmetric and bell-shaped.

The distribution shown in this histogram is called the **sampling distribution for \hat{p}** . The sampling distribution for a statistic shows the distribution of the statistic over many samples.

We'll calculate the mean of the sampling distribution for \hat{p} . How does it compare to the true population parameter p ?

```
Mean_p_hat <- mean(Samples_df$p_hat_val)
Mean_p_hat
```

```
[1] 0.60848
```

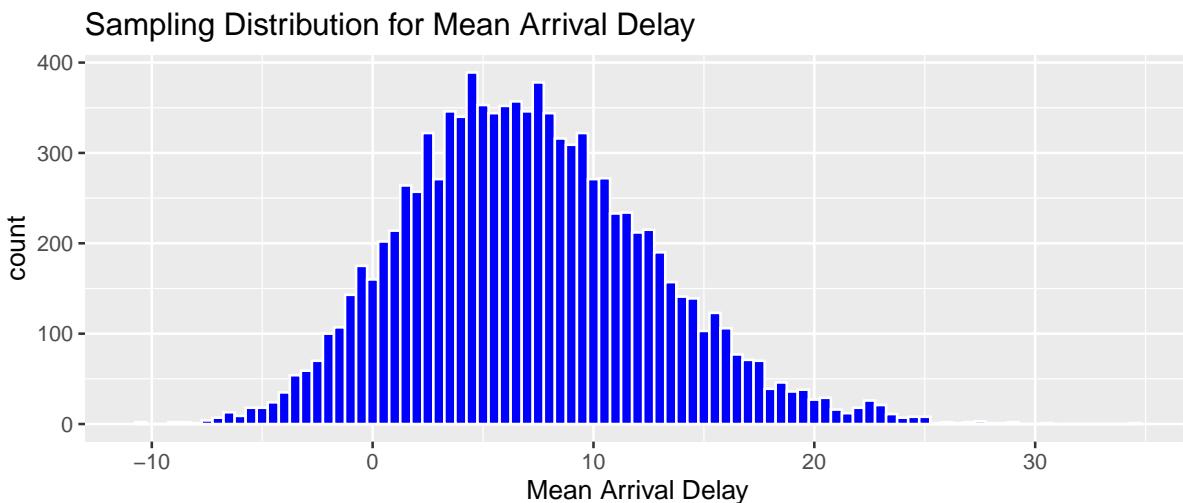
We can gauge how much the proportion of on-time arrivals varies between samples by calculating the standard deviation of this sampling distribution. The standard deviation of a sampling distribution for a statistic is also called the **standard error** of the statistic. In this case it represents the standard error \hat{p} (the proportion of on-time arrivals), and is denoted $SE(\hat{p})$. This standard error is shown below.

```
SE_p_hat <- sd(Samples_df$p_hat_val)
SE_p_hat
```

```
[1] 0.05659102
```

Now, we'll examine the sampling distribution of the mean arrival time \bar{y} .

```
Mean_Samp_Dist<- ggplot(data=Samples_df, aes(x=y_bar_val)) +
  geom_histogram(color="white", fill="blue", binwidth=0.5) +
  ggtitle("Sampling Distribution for Mean Arrival Delay") +
  xlab("Mean Arrival Delay")
Mean_Samp_Dist
```



How does the sampling distribution for mean arrival delays compare to the distribution of arrival delays for individual flights? Think about the shape and the variability of the distributions.

```
mean_y_bar <- mean(Samples_df$y_bar_val)
mean(mean_y_bar)
```

```
[1] 7.081397
```

The standard error of the mean, $SE(\bar{y})$ is shown calculated below.

```
SE_y_bar <- sd(Samples_df$y_bar_val)
SE_y_bar
```

```
[1] 5.563925
```

What does this standard error represent? How is it different than the standard deviation of flight times, which we previously saw was 60.1 minutes?

Vocabulary:

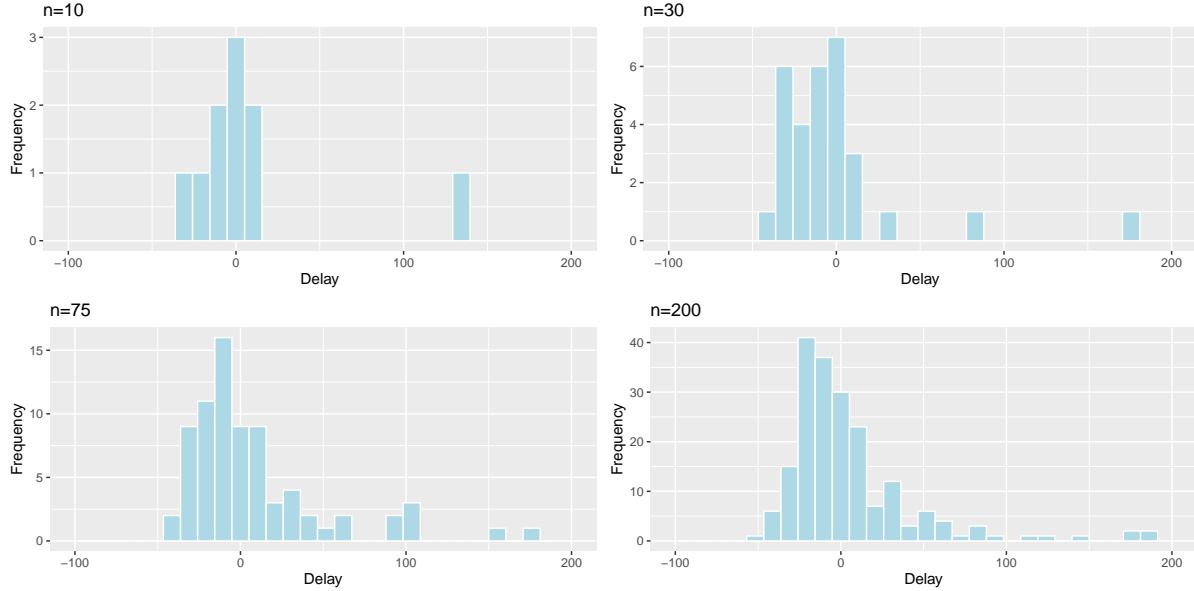
- The **sampling distribution** of a statistic is the distribution of values the statistic takes on across many different samples of a given size.
- The **standard error** of a statistic is the standard deviation of that statistic's sampling distribution. It measures how much the statistic varies between different samples of a given size.

3.1.3 Sample Size and Standard Error

Question:

Suppose the sample consisted of 10, or 30, or 500 flights, instead of 75? Would you expect the standard deviation of individual flight times to increase, decrease or stay about the same? What about the standard error of the mean delay?

The histogram shows the distribution of individual flight delays in random samples of each size.



For each sample, most of the flights have delays slightly below or above 0, though a small percentage of the flights in each sample have much larger delays. The variability in delays is about the same, regardless of sample size.

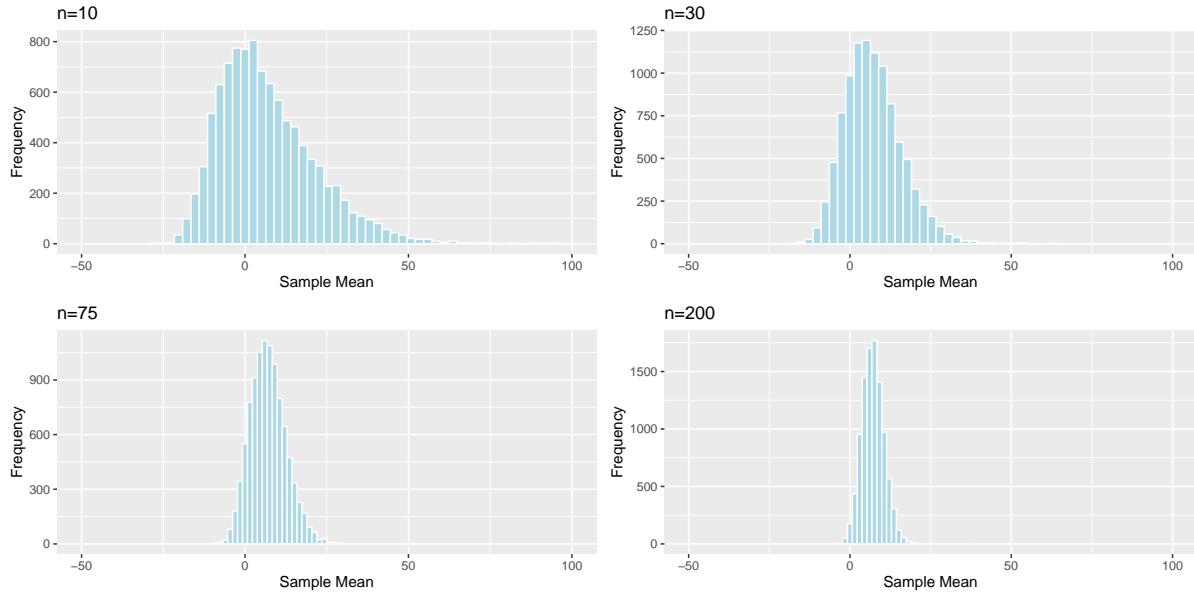
The table shows the standard deviation in each of the samples.

Sample_Size	SD
10	45.38673
30	40.64390
75	43.40787
200	48.01117

Sample size does not impact the amount of variability between individual flights. Standard deviation in delay times does not systematically increase or decrease based on sample size (of course it varies a little based on the lakes randomly chosen in the sample).

Now, we'll examine what happens to the standard error of the mean as the sample size changes.

Distributions of Mean Between Different Samples



Notice that as the sample size increases, the sampling distribution of the mean becomes more symmetric and bell-shaped, and also more concentrated around the population mean μ .

The table shows the standard error of the mean for samples of different size:

Sample_Size	SE
10	15.027787
30	8.832375
75	5.461011
200	3.371757

As sample size increases, variability between means of different samples decreases. Standard error of the mean decreases. This is also true of standard errors for other statistics (i.e. difference in means, regression slopes, etc.)

3.2 Confidence Intervals

3.2.1 Constructing Confidence Intervals

We saw that while statistics calculated from individual samples deviate from population parameters, over many samples, they approximately average to the population parameter (assuming the samples are chosen randomly).

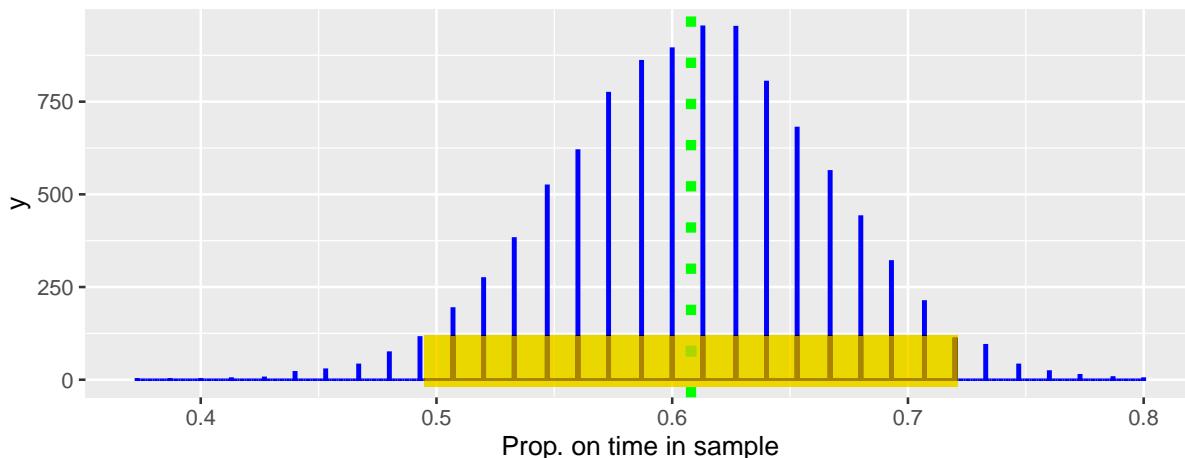
Thus, when we have only a single sample, we can use the sample statistic as an estimate of the population parameter, provided we allow for a certain margin of error. The question is how much margin of error do we need?

The sampling distribution for the proportion of on-time flights is shown again below. The true proportion of on-time flights ($p = 0.607984$) is marked by the green dotted line. The gold bar at the bottom of the histogram represents the range of sample proportions that lie within ± 2 standard errors of the true population proportion of flights that arrived on time:

$$0.608 - 2(0.057) = 0.495 \text{ to } 0.608 + 2(0.057) = 0.721$$

```
Prop_Samp_Dist + geom_vline(xintercept=p, color="green", linetype="dotted", linewidth=2) + g
```

Sampling Distribution for Proportion On Time



We calculate the proportion of samples whose proportion of on-time arrivals lies within ± 2 standard errors of the true proportion.

```
Lower <- p - 2*SE_p_hat  
Upper <- p + 2*SE_p_hat  
sum((Samples_df$p_hat_val >= Lower) & (Samples_df$p_hat_val <= Upper))
```

[1] 9539

Approximately 95% 10,000 samples produced proportions within ± 2 standard errors of the true population proportion of on-time flights.

In a real situation, we won't have access to the entire population of flights, only the flights in a single sample. For example, recall our original sample of 75 flights, in which we observed a proportion of on-time arrivals of $\hat{p} = 0.52$.

Since we now know that 95% of all samples produce proportions that lie within two standard errors of the population proportion, we can obtain an estimate of the population proportion p by adding and subtracting $2 \times \text{SE}(\hat{p})$ from our observed sample proportion \hat{p} .

Using probability theory, it can be shown generally that if the sampling distribution of a statistic is symmetric and bell shaped, then approximately 95% of all samples will produce sample statistics that lie within two standard errors of the corresponding population parameter. Such an interval is called an approximate **95% confidence interval** for the population parameter.

Approximate 95% confidence interval: If the sampling distribution of a statistic is symmetric and bell-shaped, a 95% confidence interval for the population parameter is:

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

More generally, if we want to use a level of confidence that is different than 95%, we can adjust the value we multiply the standard error by. In general, a standard error confidence interval has the form:

$$\text{Statistic} \pm m \times \text{Standard Error},$$

where the value of m depends on the desired level of confidence.

Confidence intervals that are calculated by adding and subtracting a certain number of standard errors from the sample statistic are called **standard error confidence intervals**. This approach works as long as the sampling distribution is symmetric and bell-shaped. Probability theory tells us that in a symmetric and bell-shaped distribution, approximately 95% of the area lies within two standard errors of the center of the distribution, given by the true parameter value. We will, however, see that this approach will not work in all cases. Not all statistics produce sampling distributions that are symmetric and bell-shaped, and we will need an alternative way to calculate confidence intervals in these situations.

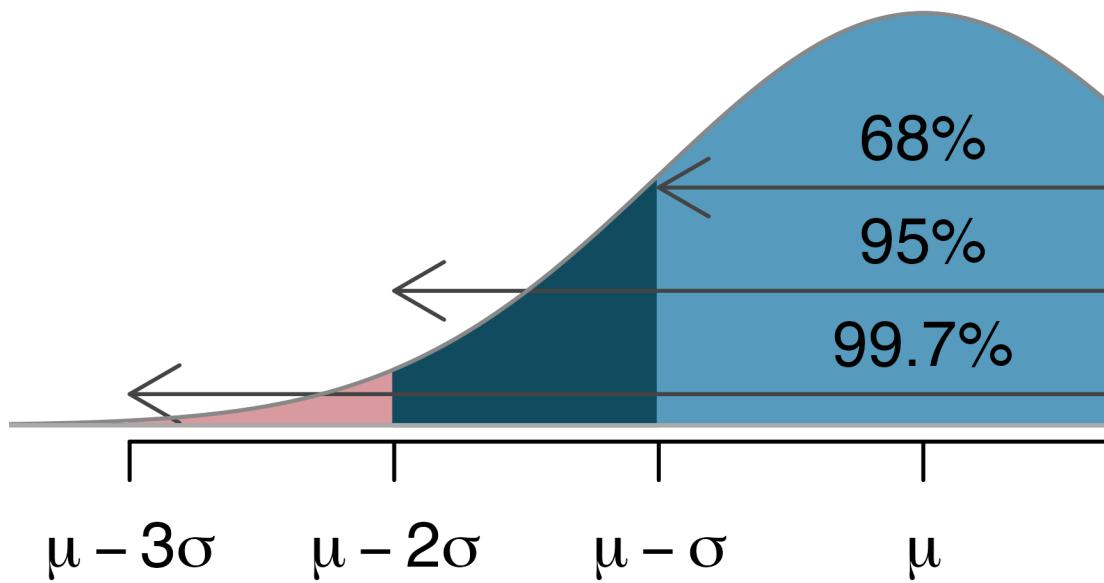


Figure 3.1: Image from <https://openintro-ims.netlify.app/foundations-mathematical>

3.2.1.1 Example: 95% Confidence Interval for p

We'll calculate a 95% confidence interval for the proportion of on-time flights, using our original sample where $\hat{p} = 0.52$. The 95% confidence interval is:

$$\begin{aligned}\hat{p} &\pm 2 \times \text{SE}(\hat{p}) \\ &= 0.52 \pm 2(0.056591)\end{aligned}$$

The confidence interval is calculated below.

```
c(p_hat - 2*SE_p_hat, p_hat + 2*SE_p_hat)
```

```
[1] 0.406818 0.633182
```

Based on our sample of 75 flights, we can be 95% confident that the true proportion of on-time arrivals among all 2013 flights from New York to Chicago is between 0.407 and 0.633.

3.2.1.2 Example: 95% Confidence Interval for μ

Likewise, we calculate a 95% confidence interval for average arrival delay using the formula:

$$\bar{y} \pm 2 \times \text{SE}(\bar{y}) \\ = 19.2 \pm 2(5.5639246)$$

```
c(y_bar - 2*SE_y_bar, y_bar + 2*SE_y_bar)
```

```
[1] 8.072151 30.327849
```

Based on our sample of 75 flights, we can be 95% confident that the mean arrival delay among all 2013 flights from New York to Chicago is between 8.1 and 30.3 minutes.

Note that this is a statement about what we think is true of the mean overall flight time, not the time of an individual flight. It would be incorrect to say that we are 95% confident that an individual flight would be expected to have a delay in this interval. You might think about whether the interval for the delay time of an individual flight should be wider or narrower than this. We'll talk about such an interval later in the term.

3.2.2 What does 95% Confidence Mean?

Knowing what we do about the true value of the population parameters p and μ , we can see that our interval for p , which was (0.407, 0.633) does indeed contain the true population value of $p = 0.6079841$. However, the interval for μ , which was (8.1, 30.3) does not contain the true value of $\mu = 7.144772$.

Does this mean we did something wrong when we calculated the interval for μ , the average flight delay among all flights in the population? The answer is “no”. Notice we claimed to be only “95%” confident that our interval contains the true value of the population parameter μ . This means that we should expect 5% of samples taken randomly to yield a sample mean

\bar{y} so different from the population mean μ , that the resulting confidence interval would not contain the true value of μ . This does not mean we did anything wrong, just that we obtained an unusual sample just by chance. Since our procedure, namely adding and subtracting two standard errors, is designed to work 95% of the time, we can expect such samples to be rare.

In a real situation, we won't know the true value of the population parameter, so we won't know for sure whether or not our confidence interval contains this true parameter value.

To further understand the meaning of "95% confidence", let's explore what happens when we calculate confidence intervals based on estimates \bar{y} obtained from many different samples. For each of our 10,000 different samples taken from our population, we'll add and subtract two standard errors from the sample proportion \hat{p} corresponding to that sample.

The table below displays the value of \hat{p} , for the first 20 samples we took, along with the lower and upper bounds of the confidence interval, and whether or not the confidence interval contains the true parameter value p (either 1=TRUE or 0=FALSE).

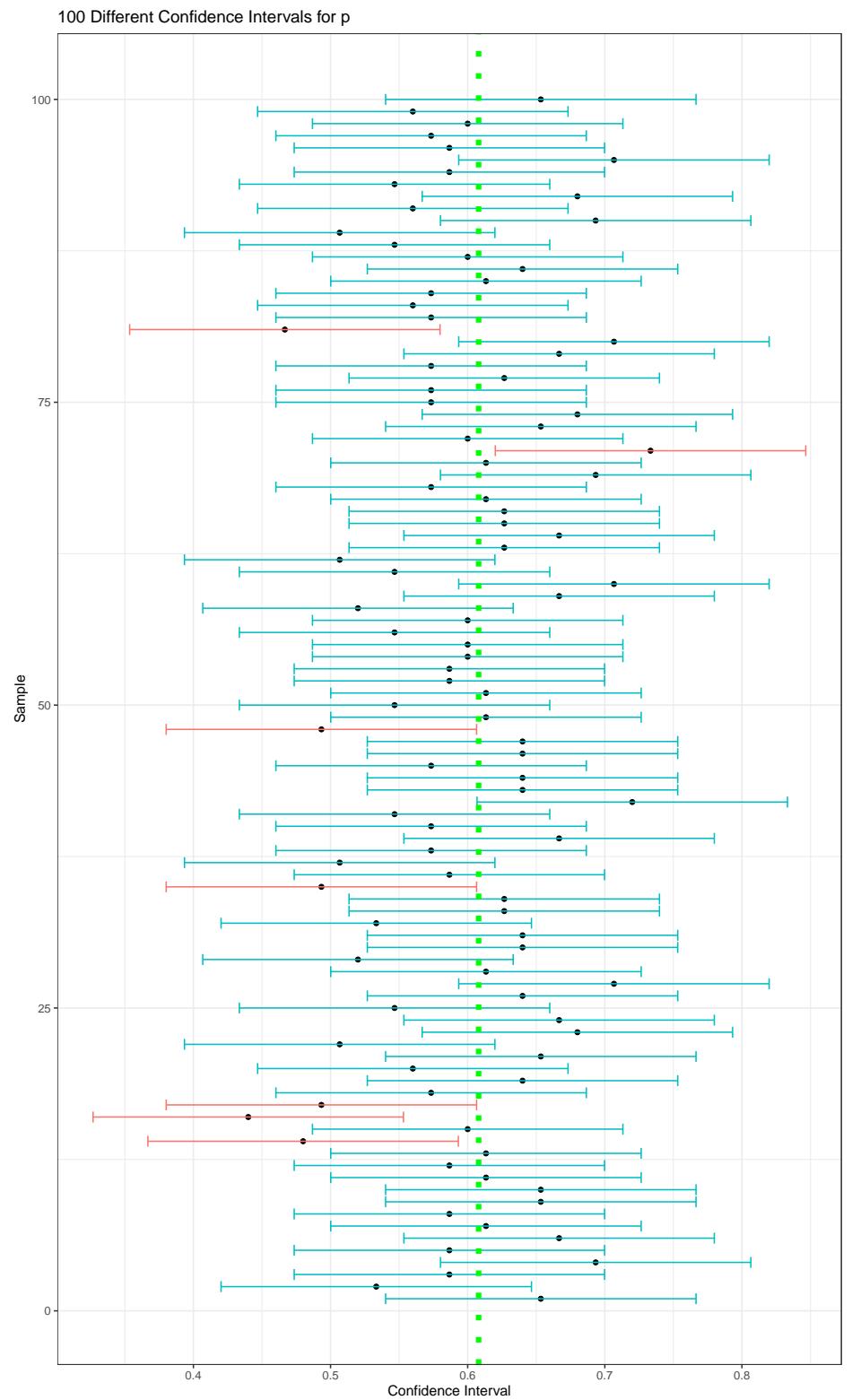
```
Samples_df_p <- Samples_df %>% mutate(Lower = p_hat_val - 2*SE_p_hat,
                                         Upper = p_hat_val + 2*SE_p_hat,
                                         Containsp = p >= Lower & p <= Upper) |>
  select(Sample, p_hat_val, Lower, Upper, Containsp)
kable(head(Samples_df_p |> round(2), 20))
```

Sample	p_hat_val	Lower	Upper	Containsp
1	0.65	0.54	0.77	1
2	0.53	0.42	0.65	1
3	0.59	0.47	0.70	1
4	0.69	0.58	0.81	1
5	0.59	0.47	0.70	1
6	0.67	0.55	0.78	1
7	0.61	0.50	0.73	1
8	0.59	0.47	0.70	1
9	0.65	0.54	0.77	1
10	0.65	0.54	0.77	1
11	0.61	0.50	0.73	1
12	0.59	0.47	0.70	1
13	0.61	0.50	0.73	1
14	0.48	0.37	0.59	0
15	0.60	0.49	0.71	1
16	0.44	0.33	0.55	0
17	0.49	0.38	0.61	0
18	0.57	0.46	0.69	1
19	0.64	0.53	0.75	1

Sample	p_hat_val	Lower	Upper	Containsp
20	0.56	0.45	0.67	1

The graphic below visualizes the confidence intervals produced using the estimates from the first 100 samples. The green dotted line indicates the true value of p . The black dots indicate the value of \hat{p} for each sample. Intervals that do in fact contain the true value of p are shown in blue, and intervals that do not contain the true value of p are shown in green.

```
ggplot(data=Samples_df_p[1:100,], aes(y=Sample, x=p_hat_val)) +
  geom_point() +
  geom_errorbar(aes(xmin = Lower, xmax = Upper, color=Containsp)) +
  xlab("Confidence Interval") +
  ylab("Sample") +
  geom_vline(xintercept = p, color="green", linetype="dotted", size=2) +
  ggtitle("100 Different Confidence Intervals for p") +
  theme_bw()
```



Out of these 100 samples, 93 contain the true value of the population parameter p . This is close to the desired 95% confidence level.

The picture shows confidence intervals produced by the first 100 samples, but we actually took 10,000 different samples of 75 flights. Let's calculate how many of these samples produced confidence intervals that contain the true value of p .

```
sum(Samples_df_p$Contains == TRUE)
```

```
[1] 9539
```

Again, notice that close to 95% of the samples produced confidence intervals contain the true population parameter p . Note that for the red intervals that do not contain p nothing was done incorrectly. The sample was taken at random, and the confidence interval was calculated using the correct formula. It just happened that by chance, we obtained a sample proportion \hat{p} that was unusually high or low, leading to an interval that did not capture the true population parameter. This, of course, happens rarely, and approximately 95% of the samples do, in fact, result in intervals that contain the true value of p .

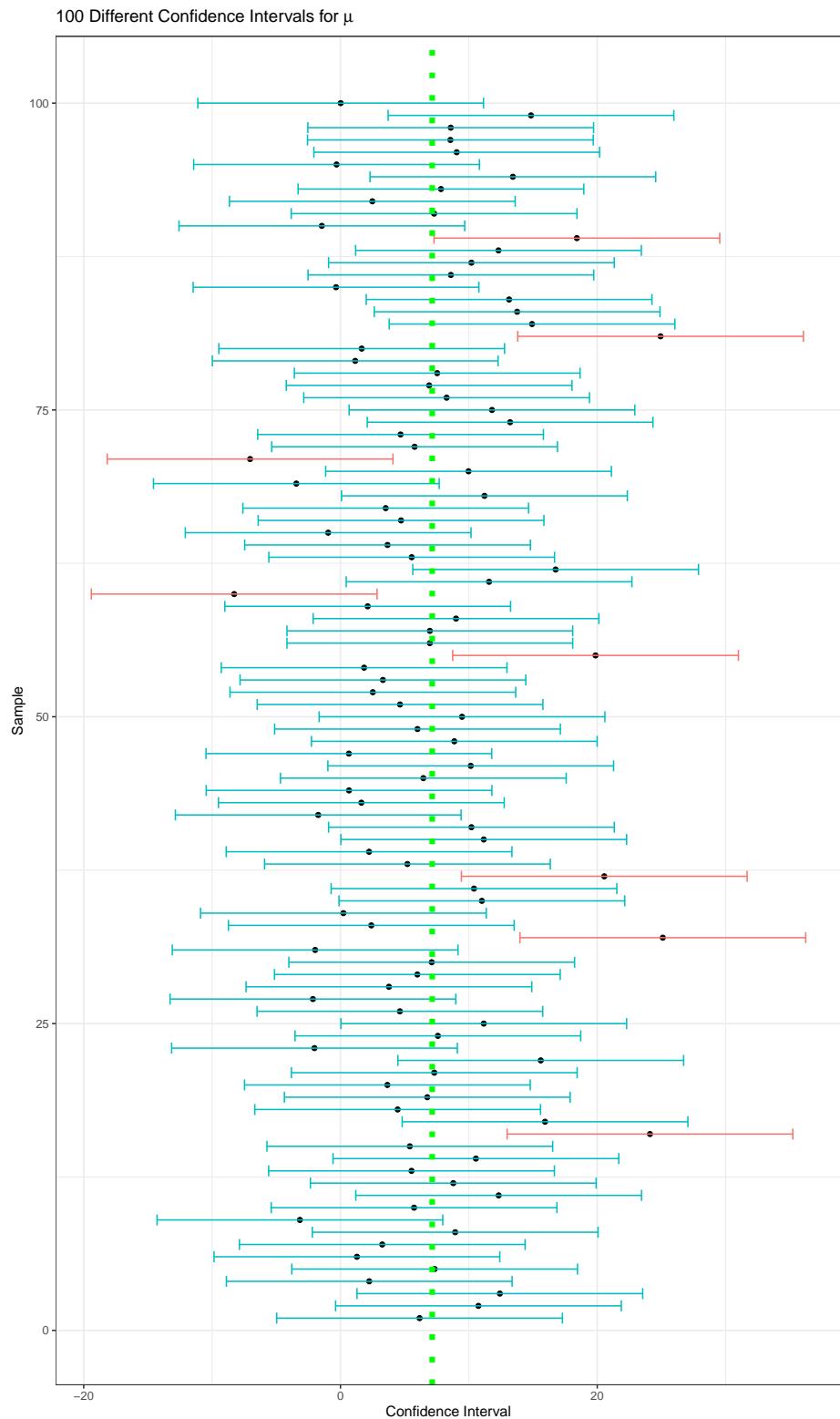
```
Samples_df_mu <- Samples_df %>% mutate(Lower = y_bar_val - 2*SE_y_bar,
                                         Upper = y_bar_val + 2*SE_y_bar,
                                         Containsmu = mu >= Lower & mu <= Upper) |>
                                         select(Sample, y_bar_val, Lower, Upper, Containsmu)
kable(head(Samples_df_mu |> round(2), 20))
```

Sample	y_bar_val	Lower	Upper	Containsmu
1	6.16	-4.97	17.29	1
2	10.75	-0.38	21.87	1
3	12.41	1.29	23.54	1
4	2.24	-8.89	13.37	1
5	7.33	-3.79	18.46	1
6	1.28	-9.85	12.41	1
7	3.25	-7.87	14.38	1
8	8.93	-2.19	20.06	1
9	-3.16	-14.29	7.97	1
10	5.73	-5.39	16.86	1
11	12.32	1.19	23.45	1
12	8.79	-2.34	19.91	1
13	5.53	-5.59	16.66	1
14	10.55	-0.58	21.67	1

Sample	y_bar_val	Lower	Upper	Containsmu
15	5.40	-5.73	16.53	1
16	24.11	12.98	35.23	0
17	15.93	4.81	27.06	1
18	4.45	-6.67	15.58	1
19	6.76	-4.37	17.89	1
20	3.65	-7.47	14.78	1

The graphic below visualizes the confidence intervals produced using the estimates from the first 100 samples. The green dotted line indicates the true value of p . The black dots indicate the value of \hat{p} for each sample. Intervals that do in fact contain the true value of p are shown in blue, and intervals that do not contain the true value of p are shown in green.

```
ggplot(data=Samples_df_mu[1:100,], aes(y=Sample, x=y_bar_val)) +
  geom_point() +
  geom_errorbar(aes(xmin = Lower, xmax = Upper, color=Containsmu)) +
  xlab("Confidence Interval") +
  ylab("Sample") +
  geom_vline(xintercept = mu, color="green", linetype="dotted", size=2) +
  ggtitle(expression(paste("100 Different Confidence Intervals for ", mu))) +
  theme_bw()
```



Out of these 100 samples, 92 contain the true value of the population parameter μ .

Out of all 10,000 samples, the proportion containing the true population value of μ is:

```
sum(Samples_df_mu$Containsmu == TRUE)
```

```
[1] 9570
```

This brings us back to the question “what does 95% confidence mean?”. An approximate 95% confidence interval means that if we take a large number of samples and calculate confidence intervals from each of them, then approximately 95% of the samples will produce intervals containing the true population parameter. In reality, we’ll only have one sample, and won’t know whether or not our interval contains the true parameter value. Assuming we have taken the sample and calculated the interval correctly, we can rest assured in the knowledge that that 95% of all intervals taken would contain the true parameter value, and hope that ours is among that 95%.

It might be tempting to say that “there is approximately a 95% chance” that the population parameter lies within the confidence interval, but this is incorrect. In the statistical framework used here (known as classical, or frequentist statistics), the population parameter is assumed to be a fixed, but (typically) unknown number. It either is within the interval, or it isn’t. We just (typically) don’t know which. There’s nothing random about whether or not the parameter value is in our interval, so it doesn’t make sense to speak of it in terms of chance or randomness. Randomness comes into play due to the fact that we selected a random sample, which will produce a statistic likely to differ from the population parameter due to sampling variability. A different statistical framework, known as *Bayesian statistics* approaches this differently, and would allow us to use randomness and chance to describe our beliefs about any uncertain quantity, including a population proportion. In this class, however, we’ll stick to the classical frequentist interpretation.

Of course, you might ask why we needed to calculate a confidence interval for the proportion of on-time flights in the first place, since we actually have data on all 20,591 flights in the population and already know the true proportion of on-time arrivals and mean arrival delay. The answer is that we don’t. But, in most real situations, we will only have data from a single sample, not the entire population, and we won’t know the true population parameter. We’ll be able to build on the ideas of sampling distributions and standard error that we learned about in this section to calculate confidence intervals in those scenarios.

3.3 Bootstrapping

3.3.1 Mercury Concentration 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.

In our sample, we have data on 53 lakes, out of more than 30,000 lakes in the state of Florida. We'll attempt to draw conclusions about the entire population, consisting of all lakes in Florida, using data from our sample of 53. It is not clear how the lakes in this sample of 53 were selected, or how representative they are of all lakes in the state of Florida. Let's assume for our purposes that the lakes in the sample can be reasonably thought of as being representative of all lakes in Florida.



Figure 3.2: <https://www.maine.gov/ifw/fish-wildlife/fisheries/species-information/largemouth-bass.html>

```
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, 0, 1, 1~
```

We are 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 3.3: from Google Maps

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","N","N","S","N","S"  
FloridaLakes <- FloridaLakes %>% rename(Mercury = AvgMercury)
```

Our data come from a sample of 53 lakes, out of more than 30,000 in the entire state of Florida. The mercury levels of the 53 lakes in the sample are shown in the table below.

```
print.data.frame(data.frame(FloridaLakes%>% select(Lake, Location, Mercury)), row.names = FALSE)
```

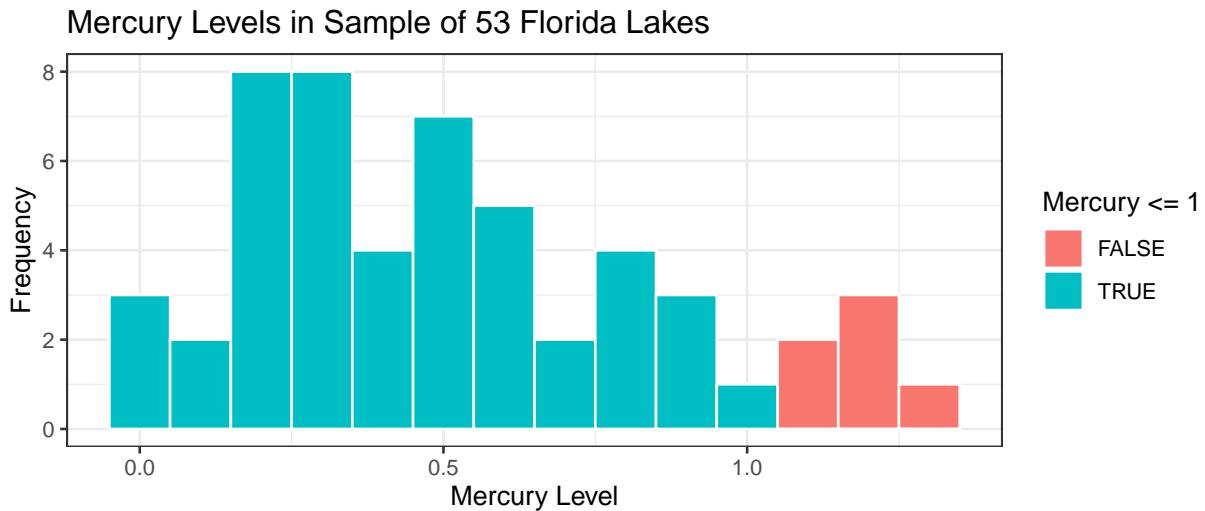
Lake	Location	Mercury
Alligator	S	1.23
Annie	S	1.33
Apopka	N	0.04

Blue Cypress	S	0.44
Brick	S	1.20
Bryant	N	0.27
Cherry	N	0.48
Crescent	N	0.19
Deer Point	N	0.83
Dias	N	0.81
Dorr	N	0.71
Down	S	0.50
Eaton	N	0.49
East Tohopekaliga	S	1.16
Farm-13	N	0.05
George	N	0.15
Griffin	N	0.19
Harney	N	0.77
Hart	S	1.08
Hatchineha	S	0.98
Iamonia	N	0.63
Istokpoga	S	0.56
Jackson	N	0.41
Josephine	S	0.73
Kingsley	N	0.34
Kissimmee	S	0.59
Lochloosa	N	0.34
Louisa	S	0.84
Miccasukee	N	0.50
Minneola	N	0.34
Monroe	N	0.28
Newmans	N	0.34
Ocean Pond	N	0.87
Ocheese Pond	N	0.56
Okeechobee	S	0.17
Orange	N	0.18
Panasoffkee	N	0.19
Parker	S	0.04
Placid	S	0.49
Puzzle	N	1.10
Rodman	N	0.16
Rousseau	N	0.10
Sampson	N	0.48
Shipp	S	0.21
Talquin	N	0.86
Tarpon	S	0.52

Tohopekaliga	S	0.65
Trafford	S	0.27
Trout	S	0.94
Tsala Apopka	N	0.40
Weir	N	0.43
Wildcat	N	0.25
Yale	N	0.27

The histogram shows the distribution of mercury levels in the 53 lakes in the sample. Lakes exceeding the US standard of 1 ppm are shown in red.

```
Lakes_Hist <- ggplot(data=FloridaLakes, aes(x=Mercury)) +
  geom_histogram(aes(fill=Mercury<=1), color="white", binwidth = 0.1) +
  ggtitle("Mercury Levels in Sample of 53 Florida Lakes") +
  xlab("Mercury Level") + ylab("Frequency") + theme_bw()
Lakes_Hist
```



The proportion of lakes with mercury levels exceeding 1 ppm is calculated below.

```
p_hat <- sum(FloridaLakes$Mercury > 1)/53
p_hat
```

```
[1] 0.1132075
```

We see that in our sample of 53 lakes, approximately 11% have mercury levels exceeding the US standard of 1 ppm. Suppose we want to estimate the proportion of all Florida Lakes whose

mercury level exceeds this standard. As we saw in the previous section, we would not expect the population proportion to exactly match the sample, due to random variability between samples. We can use the sample proportion as an estimate ($\hat{p} = 0.1132$), and construct a confidence interval for the unknown population proportion p .

In order to construct the confidence interval, we need to know how much the sample proportion of lakes exceeding 1 ppm \hat{p} could vary between different samples of size 53. That is, we need to know the standard error of \hat{p} . In the previous section, we calculated the standard error by taking 10,000 different samples of the same size as ours from the population, calculating the proportion for each sample, and then calculating the standard deviation of the proportions obtained from these 10,000 different samples. This procedure will not work here, however, because unlike the previous example where we really did have data on the entire population of all flights from New York to Chicago, we do not have data on all 30,000+ lakes in Florida. We cannot take a lot of different samples of size 53 from the population of all lakes, and thus, cannot obtain the sampling distribution for the the proportion of lakes exceeding 1 ppm, or estimate the standard error of \hat{p} .

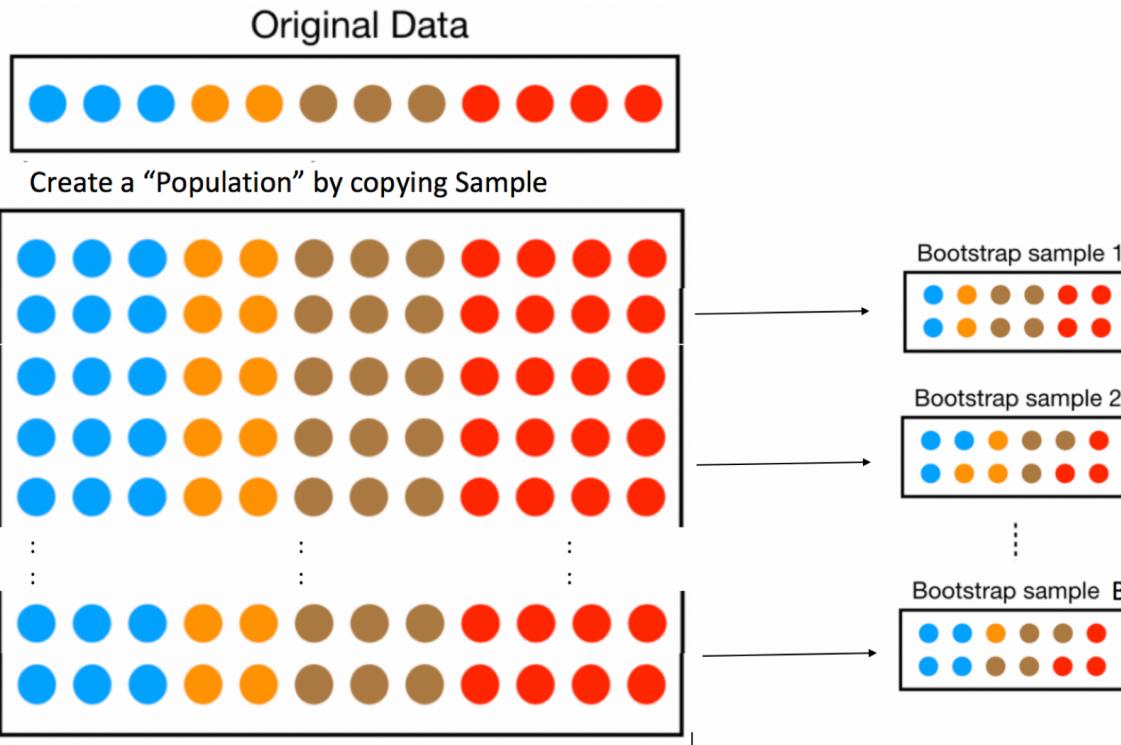
3.3.2 Bootstrap Sampling

All we have is a single sample of 53 lakes. We need to figure out how much the proportion of lakes with mercury levels exceeding 1 ppm would vary between different samples of size 53, using only the information contained in our one sample.

To do this, we'll implement a popular simulation-based strategy, known as **bootstrapping**.

Let's assume our sample is representative of all Florida lakes. Then, we'll duplicate the sample many times to create a large set that will look like the population of all Florida Lakes. We can then draw samples of 53 from that large population, and record the mean mercury level for each sample of 53.

An illustration of the bootstrapping procedure is shown below, using a sample of 12 colored dots, instead of the 53 lakes.



In fact, duplicating the sample many times and selecting new samples of size n has the same effect as drawing samples of size n from the original sample, by putting the item drawn back in each time, a procedure called **sampling with replacement**. Thus, we can skip the step of copying/pasting the sample many times, and instead draw our samples with replacement.

This means that in each new sample, some lakes will be drawn multiple times and others not at all. It also ensures that each sample is different, allowing us to estimate variability in the sample mean between the different samples of size 53.

An illustration of the concept of bootstrapping, using sampling with replacement is shown below.

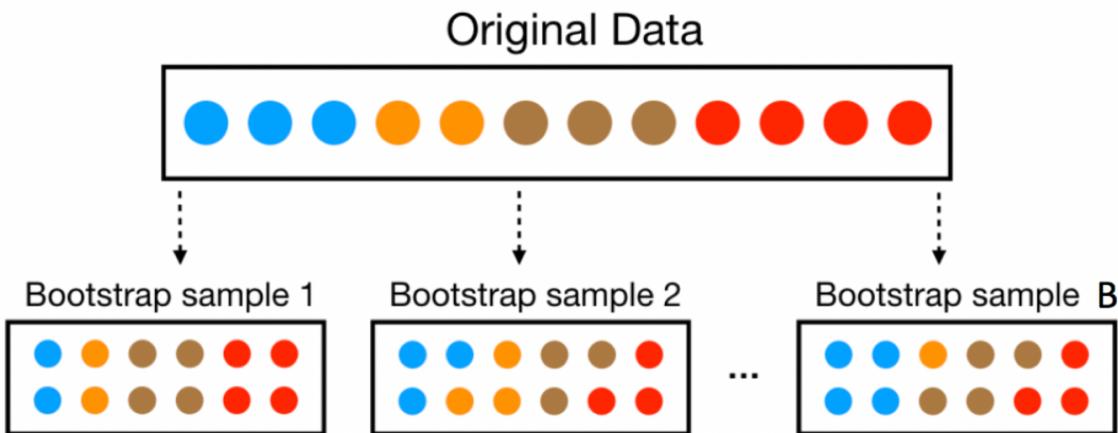


Image from <https://bradleyboehmke.github.io/HOML/process.html>

The variability in sample means in our newly drawn samples is used to approximate the variability in proportion \hat{p} that would occur between different samples of 53 lakes, drawn from the population of all Florida Lakes.

The point of bootstrapping is to observe how much a statistic (in this case the proportion of lakes with Mercury levels exceeding 1 ppm) varies between bootstrap samples. This can act as an estimate of how much that statistic would vary between different samples of size n , drawn from the population.

The steps of bootstrap sampling can be summarized in the following algorithm.

Bootstrap Algorithm

For an original sample of size n :

1. Take a sample size n by randomly sampling from the original, with replacement. Thus, some observations will show up multiple times, and others not at all. This sample is called a **bootstrap sample**.
2. Calculate the statistic of interest in the bootstrap sample (in this case \hat{p} , the proportion of lakes whose mercury levels exceed 1 ppm).
3. Repeat steps 1 and 2 many (say 10,000) times, keeping track of the statistic of interest that is calculated in each bootstrap sample.
4. Look at the distribution of the statistic across bootstrap samples. The variability in this bootstrap distribution can be used to approximate the variability in the sampling distribution for the statistic of interest.

3.3.3 Bootstrap Samples of Lakes

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

The lakes in the first sample are shown below. Notice that some lakes occur multiple times, and others not at all.

Bootstrap Sample 1

```
BootstrapSample1 <- sample_n(FloridaLakes, 53, replace=TRUE) %>% arrange(Lake)
BootstrapSample1 %>% select(ID, Lake, Mercury) |> kable()
```

ID	Lake	Mercury
3	Apopka	0.04
4	Blue Cypress	0.44
4	Blue Cypress	0.44
5	Brick	1.20
5	Brick	1.20
5	Brick	1.20
6	Bryant	0.27
6	Bryant	0.27
7	Cherry	0.48
10	Dias	0.81
10	Dias	0.81
10	Dias	0.81
13	Eaton	0.49
17	Griffin	0.19
18	Harney	0.77
18	Harney	0.77
19	Hart	1.08
20	Hatchineha	0.98
22	Istokpoga	0.56
22	Istokpoga	0.56
23	Jackson	0.41
25	Kingsley	0.34
25	Kingsley	0.34
26	Kissimmee	0.59
27	Lochloosa	0.34
30	Minneola	0.34
32	Newmans	0.34
33	Ocean Pond	0.87

ID	Lake	Mercury
34	Ocheese Pond	0.56
35	Okeechobee	0.17
37	Panasoffkee	0.19
37	Panasoffkee	0.19
38	Parker	0.04
38	Parker	0.04
40	Puzzle	1.10
40	Puzzle	1.10
40	Puzzle	1.10
41	Rodman	0.16
41	Rodman	0.16
42	Rousseau	0.10
43	Sampson	0.48
43	Sampson	0.48
44	Shipp	0.21
44	Shipp	0.21
45	Talquin	0.86
51	Tohopekaliga	0.65
51	Tohopekaliga	0.65
51	Tohopekaliga	0.65
48	Trout	0.94
48	Trout	0.94
52	Wildcat	0.25
52	Wildcat	0.25
53	Yale	0.27

We calculate the proportion of lakes with mercury levels exceeding 1 ppm in this bootstrap sample. Note that if a lake shows up more than once in the bootstrap sample, then it is counted however many times it shows up.

```
sum(BootstrapSample1$Mercury > 1) / 53
```

```
[1] 0.1320755
```

Bootstrap Sample #2

We take a second bootstrap sample. We display only the first 10 lakes, though the bootstrap sample still has 53 lakes.

Notice that the lakes chosen and omitted differ from the first sample.

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

```
# A tibble: 53 x 3
  ID   Lake      Mercury
  <int> <chr>     <dbl>
1     1 Alligator  1.23
2     3 Apopka     0.04
3     4 Blue Cypress 0.44
4     5 Brick       1.2
5     6 Bryant      0.27
6     7 Cherry      0.48
7     9 Deer Point  0.83
8    10 Dias        0.81
9    10 Dias        0.81
10   11 Dorr        0.71
# i 43 more rows
```

Proportion exceeding 1 ppm:

```
sum(BootstrapSample2$Mercury > 1) / 53
```

```
[1] 0.0754717
```

Bootstrap Sample #3

We'll take one more bootstrap sample and calculate the proportion of lakes with mercury levels exceeding 1 ppm. The first 10 lakes in this third sample are shown.

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

```
# A tibble: 53 x 3
  ID   Lake      Mercury
  <int> <chr>     <dbl>
1     1 Alligator  1.23
2     3 Apopka     0.04
3     3 Apopka     0.04
4     5 Brick       1.2
5     5 Brick       1.2
```

```

6      7 Cherry      0.48
7      7 Cherry      0.48
8      8 Crescent    0.19
9      9 Deer Point  0.83
10     10 Dias       0.81
# i 43 more rows

```

Proportion exceeding 1 ppm:

```
sum(BootstrapSample3$Mercury > 1) / 53
```

```
[1] 0.1132075
```

3.3.4 Bootstrap Distribution

Now that we have seen how bootstrap sampling works, we'll take a large number (10,000) different bootstrap samples and examine how the proportion of lakes with mercury levels exceeding 1 ppm varies between samples.

We'll use a `for-loop` to take many different bootstrap samples and record the observed proportion in a vector called `p_hat_b`

```

p_hat <- sum(FloridaLakes$Mercury > 1)/53 #calculate sample statistic
Bootstrap_prop <- rep(NA, 10000)   #setup vector to hold bootstrap statistics

for (i in 1:10000){
  BootstrapSample <- sample_n(FloridaLakes, 53, replace=TRUE) #take bootstrap sample
  Bootstrap_prop[i] <- sum(BootstrapSample$Mercury > 1)/53 # calc. prop exceeding 1
}
Lakes_Bootstrap_Prop <- data.frame(Bootstrap_prop)  #store values in a dataframe

```

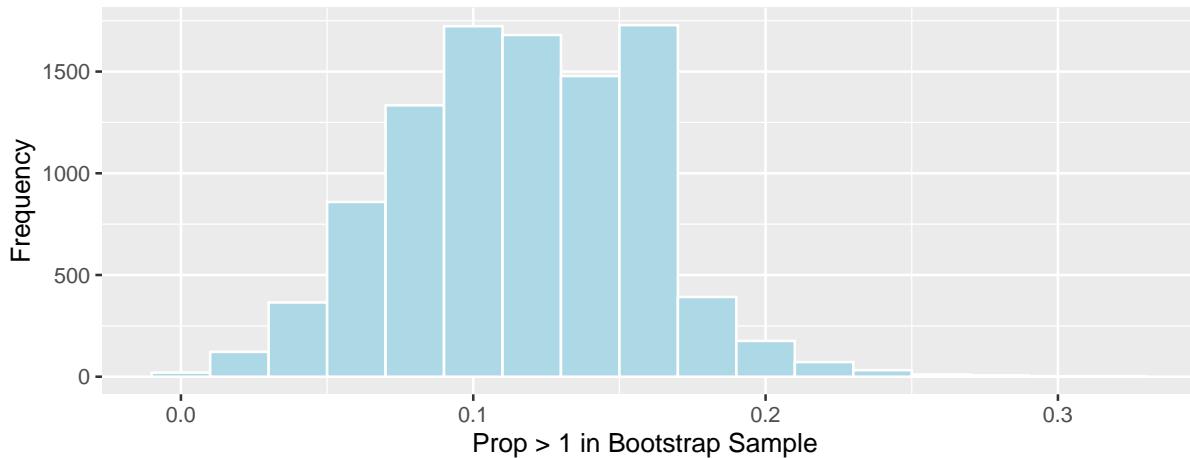
The distribution of proportions observed in the 10,000 different bootstrap samples is shown below. This distribution is called the **bootstrap distribution**.

```

Lakes_Bootstrap_Prop_plot <- ggplot(data=Lakes_Bootstrap_Prop, aes(x=Bootstrap_prop)) +
  geom_histogram(color="white", fill="lightblue", binwidth=0.02) +
  xlab("Prop > 1 in Bootstrap Sample ") + ylab("Frequency") +
  ggtitle("Bootstrap Distribution for Prop. of Lakes Exceeding 1 ppm Hg") +
  theme(legend.position = "none")
Lakes_Bootstrap_Prop_plot

```

Bootstrap Distribution for Prop. of Lakes Exceeding 1 ppm Hg



The bootstrap distribution is meant to approximate the sampling distribution of the statistic of interest (in this case the proportion exceeding 1 ppm). Because it is based on the sample, the bootstrap distribution will be centered at the sample statistic (\hat{p} in this case) while the sampling distribution would have been centered at the population parameter (p), which is unknown. The important things, however, is that the variability in the bootstrap distribution gives a good approximation of the amount of variability in the sampling distribution, so we can use the standard deviation of the bootstrap distribution (called **bootstrap standard error**) in our confidence interval calculation.

3.3.5 Bootstrap SE Confidence Interval

We calculate the standard deviation of this bootstrap distribution, which is an estimate of the standard error of \hat{p} . It measures how much the proportion of lakes exceeding 1 ppm varies between samples of size 53.

Bootstrap Standard Error:

```
SE_p_hat <- sd(Lakes_Bootstrap_Prop$Bootstrap_prop)
```

Since the bootstrap distribution is roughly symmetric and bell-shaped, we can calculate a 95% confidence interval for the proportion of all Florida lakes with mercury levels exceeding 1 ppm, using bootstrap standard error confidence interval method.

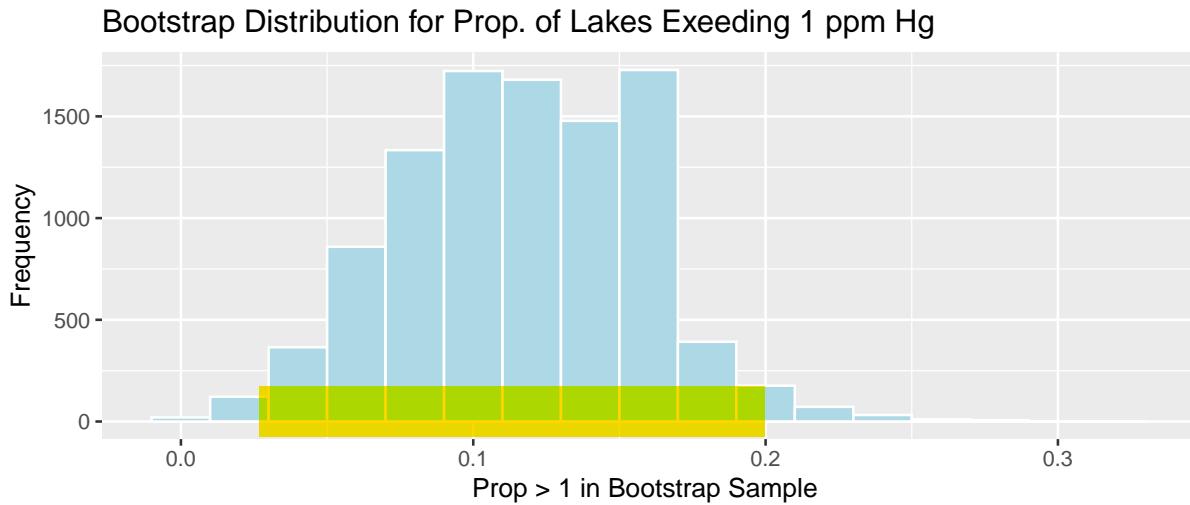
$$\hat{p} \pm 2 \times \text{SE}(\hat{p})$$

```
c(p_hat - 2*SE_p_hat, p_hat + 2*SE_p_hat)
```

```
[1] 0.02684273 0.19957237
```

The gold bar at the bottom of the bootstrap distribution represents this 95% confidence interval.

```
Lakes_Bootstrap_Prop_plot +  
  geom_segment(aes(x=p_hat - 2*SE_p_hat, xend=p_hat + 2*SE_p_hat, y=50, yend=50),  
    color="gold", size=10, alpha=0.01)
```



We are 95% confident that the proportion of all Florida lakes with mercury levels exceeding 1 ppm is between 0.0268 and 0.1996.

3.4 More Bootstrap Examples

3.4.1 Bootstrapping Other Statistics

We've seen how to use bootstrapping to calculate confidence intervals for an unknown population parameter p , using an estimate \hat{p} , calculated from a sample of size n . This procedure can be applied to calculate confidence intervals for a wide range of population parameters, using statistics calculated from a sample.

For example, we could calculate confidence intervals any of the following parameters, using the corresponding sample statistic.

Context	Parameter	Statistic
Proportion	p	\hat{p}
Mean	μ	\bar{x}
Standard Deviation	σ	s
Median	no common abbreviations	
Difference in Means	$\mu_2 - \mu_1$	$\bar{x}_2 - \bar{x}_1$
Regression Coefficient	β_j	b_j
Estimated Regression Response	$\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}$	$b_0 + b_1 x_{i1} + \dots + b_p x_{ip}$

We follow the same algorithm as we did when working with a proportion, and simply calculate whatever statistic we are interested in step 2, in place of \hat{p} , as we did previously.

The bootstrap algorithm is given again, below.

Bootstrap Algorithm

For an original sample of size n :

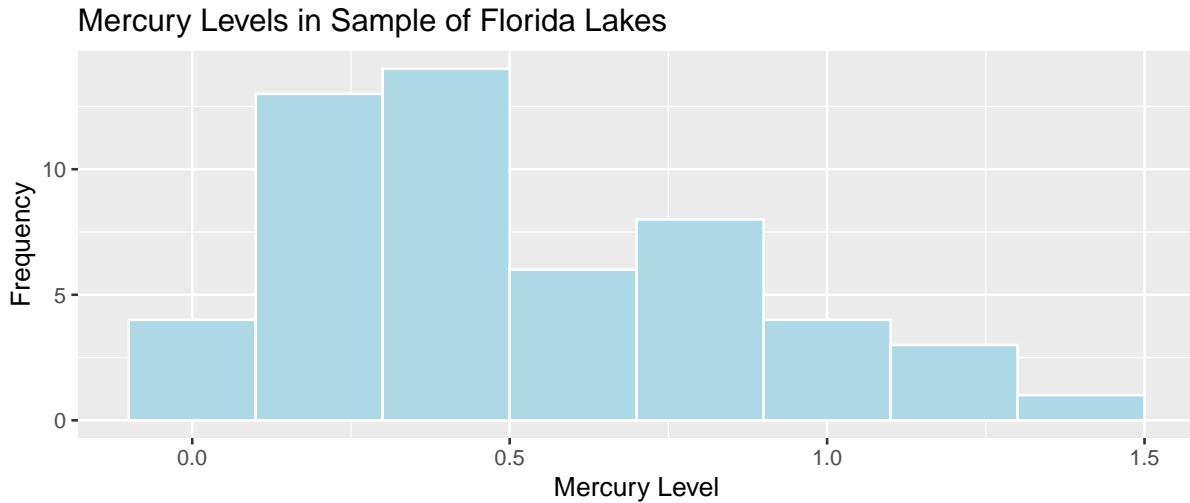
1. Take a sample of size n by randomly sampling from the original sample with replacement. (Thus some observations will show up multiple times and others not at all.)
2. Calculate the statistic of interest in the bootstrap sample.
3. Repeat steps 1 and 2 many (say 10,000) times, keeping track of the statistic of interest that is calculated in each bootstrap sample.
4. Look at the distribution of the statistic across bootstrap samples. The variability in this bootstrap distribution can be used to approximate the variability in the sampling distribution for the statistic of interest.

We'll now go through examples, calculating bootstrap confidence intervals for each of the parameters listed above.

3.4.2 CI for Mean

The histogram shows the distribution of mercury levels of the 53 lakes in our sample. The mean and standard deviation in mercury levels for these 53 lakes is shown.

```
Lakes_Hist <- ggplot(data=FloridaLakes, aes(x=Mercury)) +  
  geom_histogram(color="white", fill="lightblue", binwidth = 0.2) +  
  ggtitle("Mercury Levels in Sample of Florida Lakes") +  
  xlab("Mercury Level") + ylab("Frequency")  
Lakes_Hist
```



We'll calculate the mean and median mercury level for the 53 lakes in the sample.

```
Lakes_Stats <- FloridaLakes %>% summarize(MeanHg = mean(Mercury),  
                                              StDevHG = sd(Mercury),  
                                              N=n())  
kable(Lakes_Stats)
```

MeanHg	StDevHG	N
0.5271698	0.3410356	53

We want to calculate a 95% confidence interval for the mean mercury level among all Florida lakes. We'll use bootstrapping again, this time using the sample mean, rather than the proportion exceeding 1 ppm, as our statistic of interest.

Bootstrap Steps

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

We'll illustrate the procedure on 3 bootstrap samples.

Bootstrap Sample 1

The first 10 lakes in the bootstrap sample are shown below. Notice again that some lakes occur multiple times, and others not at all.

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

```
# A tibble: 53 x 3
  ID   Lake      Mercury
  <int> <chr>     <dbl>
1     1 Alligator  1.23
2     2 Annie      1.33
3     2 Annie      1.33
4     3 Apopka     0.04
5     3 Apopka     0.04
6     3 Apopka     0.04
7     3 Apopka     0.04
8     6 Bryant     0.27
9     9 Deer Point 0.83
10    9 Deer Point 0.83
# i 43 more rows
```

We calculate the mean mercury level among the lakes in the bootstrap sample.

```
mean(BootstrapSample1$Mercury)
```

```
[1] 0.4790566
```

Bootstrap Sample #2

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

```
# A tibble: 53 x 3
  ID     Lake   Mercury
  <int> <chr>    <dbl>
1 1     Alligator 1.23
2 1     Alligator 1.23
3 1     Alligator 1.23
4 2     Annie     1.33
5 5     Brick      1.2
6 6     Bryant    0.27
7 8     Crescent   0.19
8 8     Crescent   0.19
9 9     Deer Point 0.83
10 9    Deer Point 0.83
# i 43 more rows
```

Mean Mercury Level:

```
mean(BootstrapSample2$Mercury)
```

```
[1] 0.5479245
```

Bootstrap Sample #3

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

```
# A tibble: 53 x 3
  ID     Lake   Mercury
  <int> <chr>    <dbl>
1 1     Alligator 1.23
2 3     Apopka    0.04
3 3     Apopka    0.04
4 4     Blue Cypress 0.44
5 5     Brick      1.2
6 5     Brick      1.2
```

```

7      6 Bryant          0.27
8      7 Cherry           0.48
9      8 Crescent          0.19
10     9 Deer Point        0.83
# i 43 more rows

```

Mean Mercury Level:

```
mean(BootstrapSample3$Mercury)
```

```
[1] 0.5437736
```

Now, we'll take 10,000 bootstrap samples, and record the mean mercury concentration in each sample.

```

mean <- mean(FloridaLakes$Mercury) #calculate sample statistic
Bootstrap_Mean <- rep(NA, 10000) # setup vector to hold bootstrap statistics

for (i in 1:10000){
  BootstrapSample <- sample_n(FloridaLakes, 53, replace=TRUE) # take bootstrap sample
  Bootstrap_Mean[i] <- mean(BootstrapSample$Mercury) # calculate mean in bootstrap sample
}
Lakes_Bootstrap_Results_Mean <- data.frame(Bootstrap_Mean) #store results in data frame

```

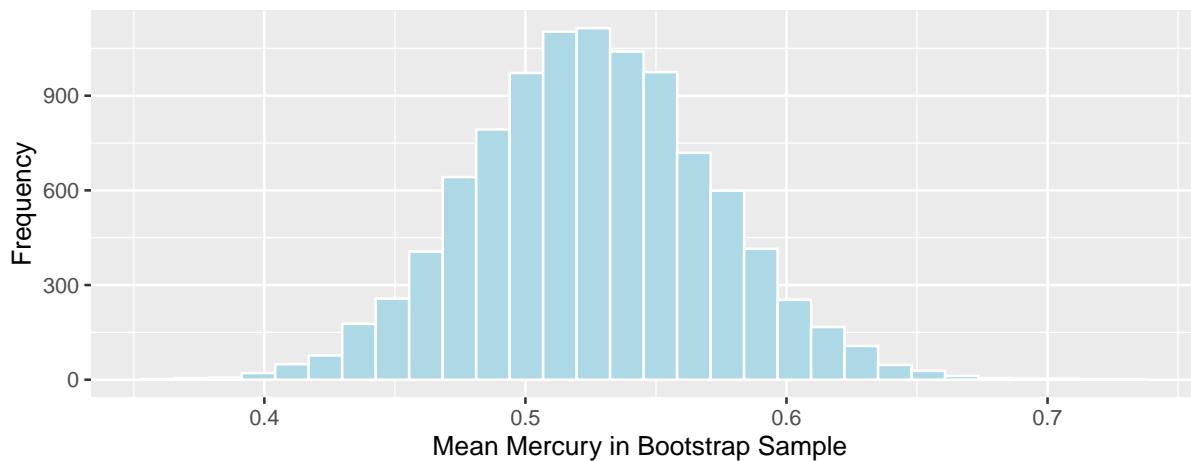
The bootstrap distribution for the mean mercury level is shown below, along with its standard error.

```

Lakes_Bootstrap_Mean_Plot <- ggplot(data=Lakes_Bootstrap_Results_Mean,
                                      aes(x=Bootstrap_Mean)) +
  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_Plot

```

Bootstrap Distribution for Sample Mean in Florida Lakes



Bootstrap Standard Error

We'll calculate the bootstrap standard error of the mean. This is a measure of how much the mean varies between samples of size 53.

```
SE_mean <- sd(Lakes_Bootstrap_Results_Mean$Bootstrap_Mean)  
SE_mean
```

```
[1] 0.04595372
```

Notice that the standard error of the mean is much less than the sample standard deviation of 0.341.

Interpretations of sample standard deviation and standard error of the mean

- The sample standard deviation measures the amount of variability in mercury levels between the 53 individual lakes in our sample.
- The standard error of the mean measures the amount of variability in sample mean mercury levels between different samples of size 53.

There is more variability between mercury levels in individual lakes than there is between average mercury levels in different samples of size 53.

Since the bootstrap distribution is roughly symmetric and bell-shaped, we can use the bootstrap standard error method to calculate an approximate 95% confidence interval for the mean mercury level among all Florida lakes.

Statistic $\pm 2 \times$ Standard Error

In this case, the statistic of interest is the sample mean $\bar{x} = 0.527$. The confidence interval is

$$\begin{aligned}\bar{x} \pm 2 \times \text{SE}(\bar{x}) \\ = 0.527 \pm 2 \times 0.0459537\end{aligned}$$

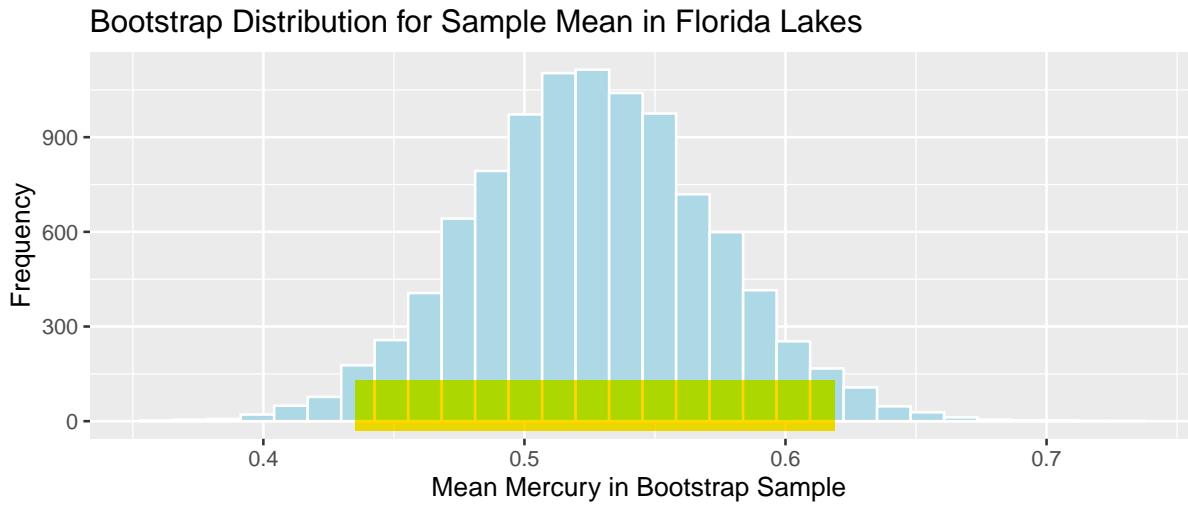
95% Confidence Interval:

```
c(mean - 2*SE_mean, mean + 2*SE_mean)
```

```
[1] 0.4352624 0.6190773
```

The 95% confidence interval is shown by the gold bar on the graph of the bootstrap distribution below.

```
Lakes_Bootstrap_Mean_Plot +
  geom_segment(aes(x=mean - 2*SE_mean, xend=mean + 2*SE_mean, y=50, yend=50),
               color="gold", size=10, alpha=0.01)
```



We are 95% confident that the average mercury level among all Florida lakes is between 0.44 and 0.62 parts per million.

It is important to note that we are not saying that we are 95% confident that an individual lake lie in this range, or that 95% of all individual lakes lie in this range. We are only saying that we are confident that the *average* mercury level among all lakes lies in this range. A confidence interval is a statement about a population parameter (in this case the average mercury level), rather than about individual lakes in the population. Since there is more variability about individual lakes than overall averages, we'll need to make a wider interval when talking about the mercury level for an individual lake.

3.4.3 CI for Standard Deviation

Now, we'll calculate a confidence interval for the standard deviation in mercury levels among all Florida lakes. Recall that the sample standard deviation (s) was:

```
Sample_SD <- sd(FloridaLakes$Mercury)  
Sample_SD
```

```
[1] 0.3410356
```

We'll use this estimate to calculate a confidence interval for the population standard deviation σ .

This time, our statistic of interest is the sample standard deviation s .

Bootstrap Steps

1. Take a sample of 53 lakes by randomly sampling from the original sample of 53 lakes, with replacement.
2. Calculate the standard deviation in mercury level in the bootstrap sample.
3. Repeat steps 1 and 2 many (say 10,000) times, keeping track of the standard deviation mercury level in each bootstrap sample.
4. Look at the distribution of the standard deviations across bootstrap samples. The variability in this bootstrap distribution can be used to approximate the variability in the sampling distribution for the standard deviation in mercury level.

We'll illustrate the procedure on 3 bootstrap samples.

Bootstrap Sample 1

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

```
# A tibble: 53 x 3  
  ID     Lake       Mercury  
  <int> <chr>      <dbl>  
1     1 Alligator   1.23  
2     3 Apopka      0.04  
3     4 Blue Cypress 0.44  
4     5 Brick        1.2  
5     6 Bryant       0.27
```

```
6     6 Bryant      0.27
7     6 Bryant      0.27
8     6 Bryant      0.27
9     7 Cherry       0.48
10    8 Crescent     0.19
# i 43 more rows
```

We calculate the standard deviation in mercury levels among the lakes in the bootstrap sample.

```
sd(BootstrapSample1$Mercury)
```

```
[1] 0.3157289
```

Bootstrap Sample #2

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

```
# A tibble: 53 x 3
  ID   Lake        Mercury
  <int> <chr>      <dbl>
1 1    Alligator   1.23
2 1    Alligator   1.23
3 3    Apopka      0.04
4 4    Blue Cypress 0.44
5 5    Brick        1.2
6 5    Brick        1.2
7 6    Bryant       0.27
8 7    Cherry       0.48
9 7    Cherry       0.48
10 8   Crescent     0.19
# i 43 more rows
```

Standard Deviation in Mercury Level:

```
sd(BootstrapSample2$Mercury)
```

```
[1] 0.3466327
```

Bootstrap Sample #3

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

```
# A tibble: 53 x 3
  ID     Lake   Mercury
  <int> <chr>    <dbl>
1 2     Annie     1.33
2 3     Apopka    0.04
3 5     Brick      1.2
4 6     Bryant    0.27
5 6     Bryant    0.27
6 6     Bryant    0.27
7 7     Cherry    0.48
8 8     Crescent   0.19
9 9     Deer Point 0.83
10 9    Deer Point 0.83
# i 43 more rows
```

Standard Deviation Mercury Level:

```
sd(BootstrapSample3$Mercury)
```

```
[1] 0.3294457
```

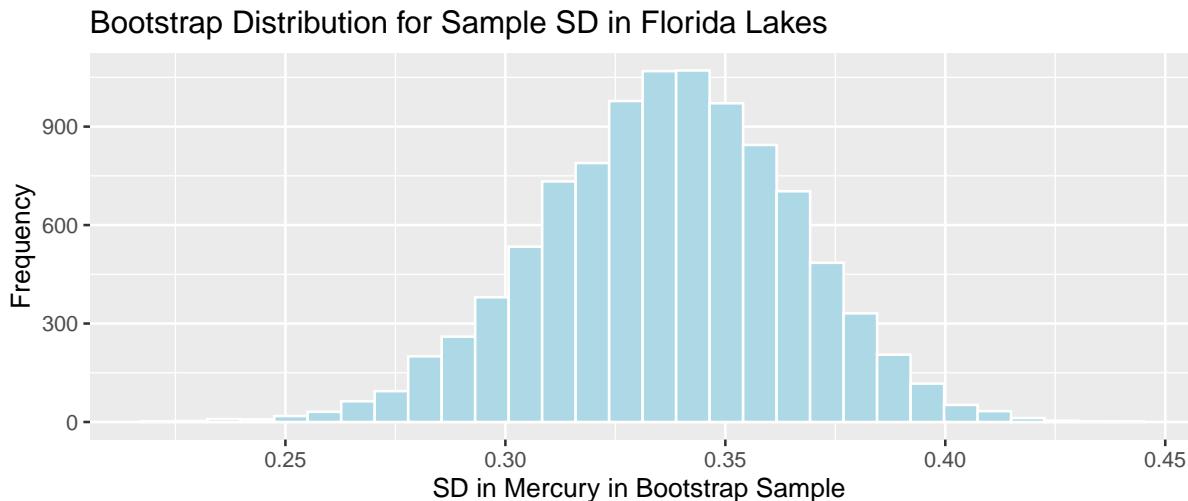
Now, we'll take 10,000 bootstrap samples, and record the standard deviation in mercury concentration in each sample.

```
Sample_SD <- sd(FloridaLakes$Mercury) #calculate sample statistic
Bootstrap_SD <- rep(NA, 10000) # setup vector to hold bootstrap statistics

for (i in 1:10000){
  BootstrapSample <- sample_n(FloridaLakes, 53, replace=TRUE) # take bootstrap sample
  Bootstrap_SD[i] <- sd(BootstrapSample$Mercury) # calculate standard deviation in bootstrap s
}
Lakes_Bootstrap_Results_SD <- data.frame(Bootstrap_SD) #store results in data frame
```

The bootstrap distribution for the mean mercury level is shown below, along with its standard error.

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



Bootstrap Standard Error:

We'll calculate the bootstrap standard error of the standard deviation. This is a measure of how much the standard deviation varies between samples.

```
SE_SD <- sd(Lakes_Bootstrap_Results_SD$Bootstrap_SD)
SE_SD
```

[1] 0.02867548

Since the bootstrap distribution is roughly symmetric and bell-shaped, we can use the bootstrap standard error method to calculate an approximate 95% confidence interval for the standard deviation in mercury levels among all Florida lakes.

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

In this case, the statistic of interest is the sample standard deviation $s = 0.341$. The confidence interval is

$$s \pm 2 \times \text{SE}(s) \\ = 0.341 \pm 2 \times 0.029$$

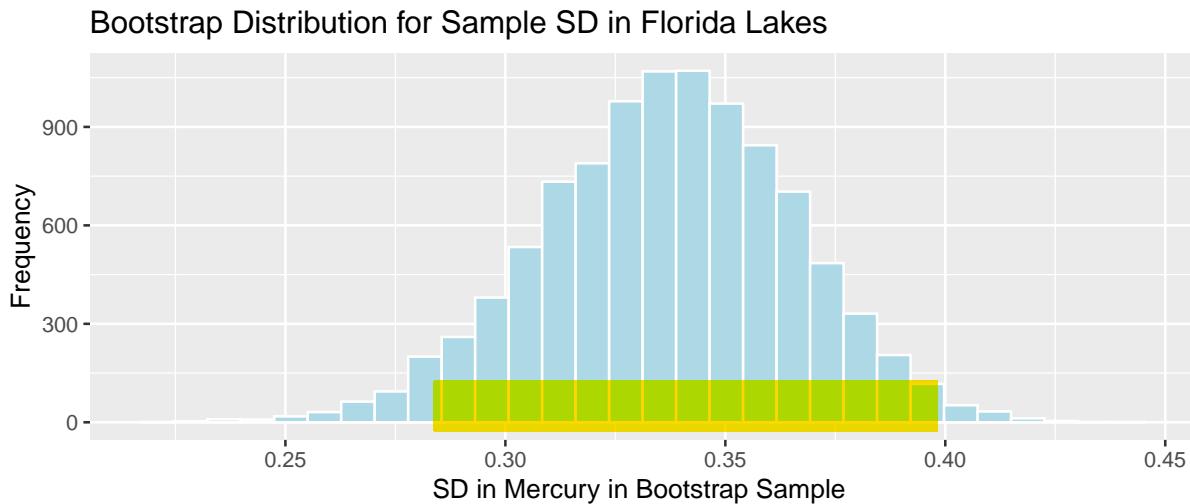
95% Confidence Interval:

```
c(Sample_SD - 2*SE_SD, Sample_SD + 2*SE_SD )
```

```
[1] 0.2836847 0.3983866
```

The 95% confidence interval is shown by the gold bar on the graph of the bootstrap distribution below.

```
Lakes_Bootstrap_SD_Plot +
  geom_segment(aes(x=Sample_SD - 2*SE_SD,xend=Sample_SD + 2*SE_SD, y=50, yend=50),
               color="gold", size=10, alpha=0.01)
```



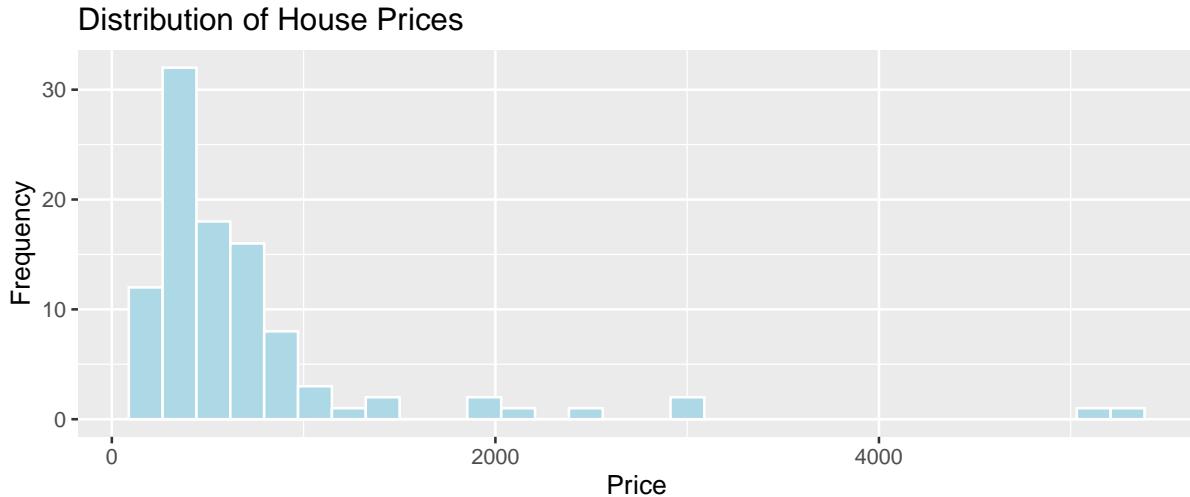
We are 95% confident that the standard deviation in mercury levels among all Florida lakes is between 0.28 and 0.4 parts per million.

3.4.4 CI for Median

We already calculated a confidence interval for the mean mercury level among all Florida lakes. We could calculate a bootstrap confidence interval for the median mercury level as well, but since the distribution of mercury levels in the lakes is roughly symmetric, the mean is a reasonable measure of center, and there is not a clear reason for using the median instead.

When a distribution is skewed or contains large outliers, however, the median is a more robust measure of center than the mean. Recall the distribution of 100 Seattle house prices seen in Chapters 1 and 2.

```
ggplot(data=Houses, aes(x=price)) +  
  geom_histogram(fill="lightblue", color="white") +  
  ggtitle("Distribution of House Prices") +  
  xlab("Price") +  
  ylab("Frequency")
```



These 100 houses are a sample of all houses sold in Seattle in 2014 and 2015, so we can use statistics from our sample to draw conclusions about all houses sold in Seattle in this time period.

In this subsection, we'll use bootstrapping to calculate a 95% confidence interval for the median price among all houses sold in Seattle in this time period.

We calculate the sample median price.

```
Sample_Median <- median(Houses$price)  
Sample_Median
```

[1] 507.5

Bootstrap Steps

1. Take a sample of 100 houses by randomly sampling from the original sample of 100 houses, with replacement.

2. Calculate the median price in the bootstrap sample.
3. Repeat steps 1 and 2 many (say 10,000) times, keeping track of the median price in each bootstrap sample.
4. Look at the distribution of the median price across bootstrap samples. The variability in this bootstrap distribution can be used to approximate the variability in the sampling distribution for the median price.

We'll illustrate the procedure on 3 bootstrap samples.

Bootstrap Sample 1

```
BootstrapSample1 <- sample_n(Houses, 100, replace=TRUE) %>% arrange(Id)
BootstrapSample1 %>% select(Id, price)
```

```
# A tibble: 100 x 2
  Id   price
  <int> <dbl>
1     2    885.
2     3    385.
3     4    253.
4     4    253.
5     5    468.
6     6    310.
7     8    485.
8     9    315.
9     9    315.
10    10   425
# i 90 more rows
```

We calculate the median price among the houses in the bootstrap sample.

```
median(BootstrapSample1$price)
```

```
[1] 400
```

Bootstrap Sample #2

```
BootstrapSample2 <- sample_n(Houses, 100, replace=TRUE) %>% arrange(Id)
BootstrapSample2 %>% select(Id, price)
```

```
# A tibble: 100 x 2
  Id   price
  <int> <dbl>
1     1 1225
2     3 385.
3     4 253.
4     4 253.
5     4 253.
6     6 310.
7     7 550.
8     8 485.
9     8 485.
10    9 315.
# i 90 more rows
```

Median Price:

```
median(BootstrapSample2$price)
```

```
[1] 427.5
```

Bootstrap Sample #3

```
BootstrapSample3 <- sample_n(Houses, 100, replace=TRUE) %>% arrange(Id)
BootstrapSample3 %>% select(Id, price)
```

```
# A tibble: 100 x 2
  Id   price
  <int> <dbl>
1     2 885.
2     4 253.
3     5 468.
4     7 550.
5    10 425
6    13 1325
7    16 3075
8    17 438
9    18 688.
10   19 995.
# i 90 more rows
```

Median Price:

```
median(BootstrapSample3$price)
```

```
[1] 488
```

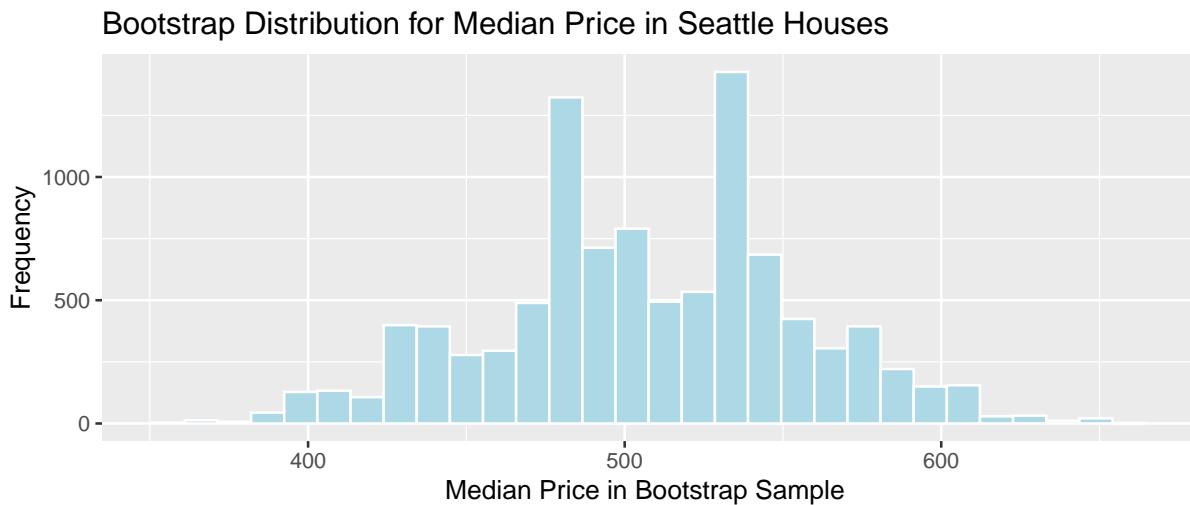
Now, we'll take 10,000 bootstrap samples, and record the median price in each sample.

```
Sample_Med <- median(Houses$price) #calculate sample median
Bootstrap_Med <- rep(NA, 10000) # setup vector to hold bootstrap statistics

for (i in 1:10000){
  BootstrapSample <- sample_n(Houses, 100, replace=TRUE) # take bootstrap sample
  Bootstrap_Med[i] <- median(BootstrapSample$price) # calculate standard deviation in bootstrap
}
Houses_Bootstrap_Results_Med <- data.frame(Bootstrap_Med) #store results in data frame
```

The bootstrap distribution for the median price is shown below, along with its standard error.

```
Houses_Bootstrap_Med_Plot <- ggplot(data=Houses_Bootstrap_Results_Med,
                                       aes(x=Bootstrap_Med)) +
  geom_histogram(color="white", fill="lightblue") +
  xlab("Median Price in Bootstrap Sample ") + ylab("Frequency") +
  ggttitle("Bootstrap Distribution for Median Price in Seattle Houses") +
  theme(legend.position = "none")
Houses_Bootstrap_Med_Plot
```



Bootstrap Standard Error:

We'll calculate the bootstrap standard error of the median. This is a measure of how much the median varies between samples.

```
SE_Med <- sd(Houses_Bootstrap_Results_Med$Bootstrap_Med)  
SE_Med
```

```
[1] 48.11411
```

The standard error measures the amount of variability in median house price between different samples of size 100.

Note that this is different than the sample standard deviation, which represents the standard deviation in prices between the 100 different houses in the sample.

Notice that the bootstrap distribution for the median is not symmetric and bell-shaped. Thus, we cannot be assured that 95% of samples will produce a statistic within two standard errors of the mean, so the standard error confidence interval method is not appropriate here. Instead, we'll calculate a confidence interval by taking the middle 95% of the values in the bootstrap distribution. A confidence interval calculated this way is called a **percentile bootstrap interval**.

We'll calculate the 0.025 quantile and the 0.975 quantile of this bootstrap distribution. These are the points below which lie 2.5% and 97.5% of the medians in the bootstrap distribution. Thus, the middle 95% of the medians lie between these values.

```
q.025 <- quantile(Houses_Bootstrap_Results_Med$Bootstrap_Med, 0.025)  
q.025
```

```
2.5%  
410
```

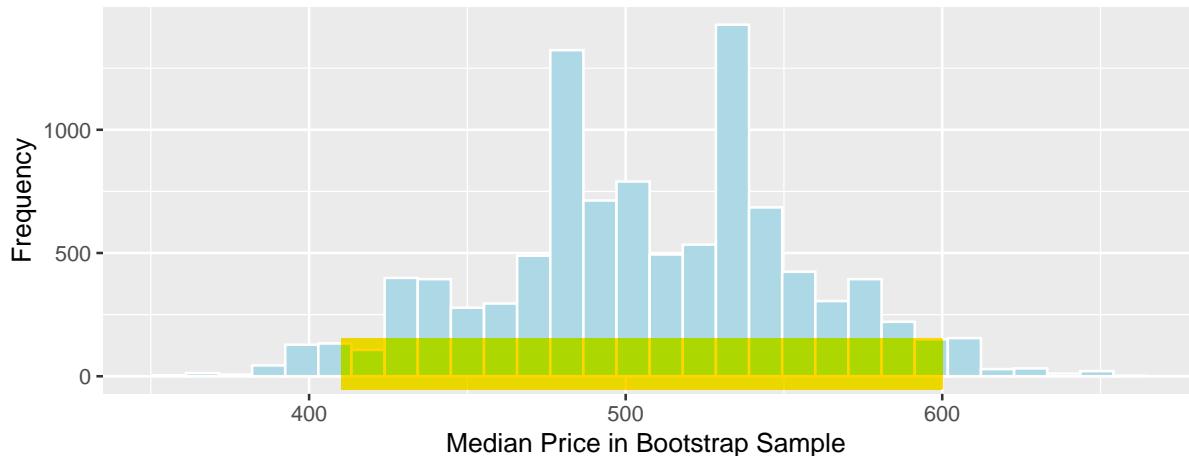
```
q.975 <- quantile(Houses_Bootstrap_Results_Med$Bootstrap_Med, 0.975)  
q.975
```

```
97.5%  
600.05
```

The 95% confidence interval is shown by the gold bar on the graph of the bootstrap distribution below.

```
Houses_Bootstrap_Med_Plot +
  geom_segment(aes(x=q.025,xend=q.975, y=50, yend=50),
               color="gold", size=10, alpha=0.01)
```

Bootstrap Distribution for Median Price in Seattle Houses



We are 95% confident that the median price among all houses that sold in Seattle between 2014 and 2015 is between 410 and 600 thousand dollars.

3.4.5 CI for Difference in Means

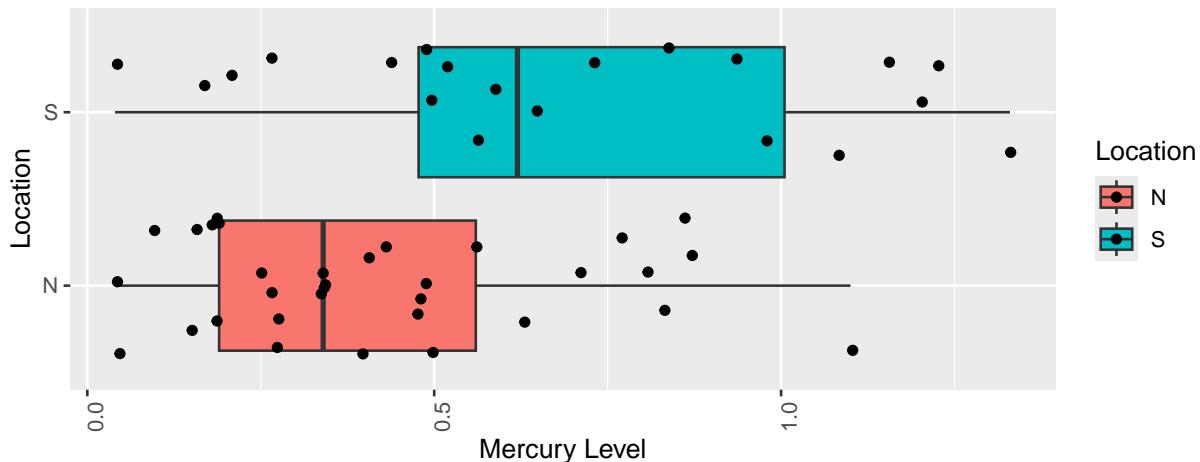
We previously calculated a confidence interval for the average mercury level among all lakes in Florida.

Now, we'll calculate an interval for the difference in average mercury levels between lakes in Northern Florida, compared to Southern Florida.

The boxplot shows and table below describe the distribution of mercury levels for lakes in Northern Florida, compared to Southern Florida.

```
LakesBP <- ggplot(data=FloridaLakes, aes(x=Location, y=Mercury, 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
```

Mercury Levels in Florida Lakes



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

In our sample of 33 Northern Lakes and 20 Southern Lakes, we saw a difference of 0.27 ppm. We'll calculate a confidence interval to estimate how big or small this difference could be among all Florida lakes.

We'll use a statistical model to calculate the average mercury levels in Northern and Southern Florida.

$$\widehat{\text{Mercury}} = b_0 + b_1 \times \text{South}$$

- 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

The estimates for corresponding to the original sample are shown below.

```
M <- lm(data=FloridaLakes, Mercury~Location)
```

```
M
```

Call:

```
lm(formula = Mercury ~ Location, data = FloridaLakes)
```

Coefficients:

(Intercept)	LocationS
0.4245	0.2720

Thus, we can obtain a confidence interval for the difference in average mercury levels by fitting a regression model to each of our bootstrap samples and recording the value of the sample statistic b_1 , which represents this difference. Alternatively, we could calculate the mean from each group separately and subtract.

When comparing groups, we make one modification in Step #1 of the bootstrap process. Rather than drawing a sample of size n at random, with replacement, we'll draw the same number of observations from each group as were in the original sample. In this case, we had 33 northern lakes, and 20 southern lakes.

Bootstrap Steps

1. Take a sample of 33 northern lakes and 20 southern lakes by randomly sampling from the original sample, with replacement.
2. Fit a regression model with location as the explanatory variable and record the value of b_1 , representing the difference between the means for each group (South-North).
3. Repeat steps 1 and 2 many (say 10,000) times, keeping track of the difference in means in each bootstrap sample.
4. Look at the distribution of the differences in means across bootstrap samples. The variability in this bootstrap distribution can be used to approximate the variability in the sampling distribution for the difference in means between mercury levels in Northern and Southern Florida.

We'll illustrate the procedure on 3 bootstrap samples.

Bootstrap Sample 1

```

NLakes <- sample_n(FloridaLakes %>% filter(Location=="N"), 33, replace=TRUE) ## sample 33
SLakes <- sample_n(FloridaLakes %>% filter(Location=="S"), 20, replace=TRUE) ## sample 20
BootstrapSample1 <- rbind(NLakes, SLakes) %>% arrange(ID) %>%
  select(ID, Lake, Location, Mercury) ## combine Northern and Southern Lakes
BootstrapSample1

```

```

# A tibble: 53 x 4
  ID Lake      Location Mercury
  <int> <chr>    <fct>     <dbl>
1 1 Alligator S          1.23
2 2 Annie     S          1.33
3 2 Annie     S          1.33
4 3 Apopka    N          0.04
5 4 Blue Cypress S          0.44
6 4 Blue Cypress S          0.44
7 4 Blue Cypress S          0.44
8 7 Cherry    N          0.48
9 8 Crescent   N          0.19
10 10 Dias     N          0.81
# i 43 more rows

```

We fit a regression model to the bootstrap sample and calculate the regression coefficients. We're interested in the second coefficient, b_1 , which represents the mean difference between lakes in Southern and Northern Florida

```

Mb1 <- lm(data=BootstrapSample1, Mercury ~ Location) ## fit linear model
Mb1

```

Call:
`lm(formula = Mercury ~ Location, data = BootstrapSample1)`

Coefficients:
`(Intercept) LocationS`
`0.3894 0.2471`

```

NLakes <- sample_n(FloridaLakes %>% filter(Location=="N"), 33, replace=TRUE) ## sample 33
SLakes <- sample_n(FloridaLakes %>% filter(Location=="S"), 20, replace=TRUE) ## sample 20
BootstrapSample2 <- rbind(NLakes, SLakes) %>% arrange(ID) %>%
  select(ID, Lake, Location, Mercury) ## combine Northern and Southern Lakes
BootstrapSample2

```

```

# A tibble: 53 x 4
  ID Lake      Location Mercury
  <int> <chr>    <fct>     <dbl>
1 1 Alligator S        1.23
2 1 Alligator S        1.23
3 1 Alligator S        1.23
4 1 Alligator S        1.23
5 3 Apopka   N        0.04
6 3 Apopka   N        0.04
7 3 Apopka   N        0.04
8 4 Blue Cypress S    0.44
9 4 Blue Cypress S    0.44
10 4 Blue Cypress S   0.44
# i 43 more rows

```

Bootstrap Sample 2

```

Mb2 <- lm(data=BootstrapSample2, Mercury ~ Location) ## fit linear model
Mb2

```

Call:

```
lm(formula = Mercury ~ Location, data = BootstrapSample2)
```

Coefficients:

(Intercept)	LocationS
0.4585	0.2755

Bootstrap Sample 3

```

NLakes <- sample_n(FloridaLakes %>% filter(Location=="N"), 33, replace=TRUE) ## sample 33
SLakes <- sample_n(FloridaLakes %>% filter(Location=="S"), 20, replace=TRUE) ## sample 20
BootstrapSample3 <- rbind(NLakes, SLakes) %>% arrange(ID) %>%
  select(ID, Lake, Location, Mercury) ## combine Northern and Southern Lakes
BootstrapSample3

```

```

# A tibble: 53 x 4
  ID Lake      Location Mercury
  <int> <chr>    <fct>     <dbl>
1 1 Alligator S        1.23
2 1 Alligator S        1.23

```

```

3    2 Annie      S      1.33
4    2 Annie      S      1.33
5    2 Annie      S      1.33
6    2 Annie      S      1.33
7    3 Apopka     N      0.04
8    3 Apopka     N      0.04
9    6 Bryant     N      0.27
10   6 Bryant     N      0.27
# i 43 more rows

```

```

Mb3 <- lm(data=BootstrapSample3, Mercury ~ Location) ## fit linear model
Mb3

```

Call:
`lm(formula = Mercury ~ Location, data = BootstrapSample3)`

Coefficients:
`(Intercept) LocationS
 0.3342 0.4178`

We'll now take 10,000 different bootstrap samples and look at the bootstrap distribution for b_1 , the difference in mean mercury levels between lakes in Southern and Northern Florida

```

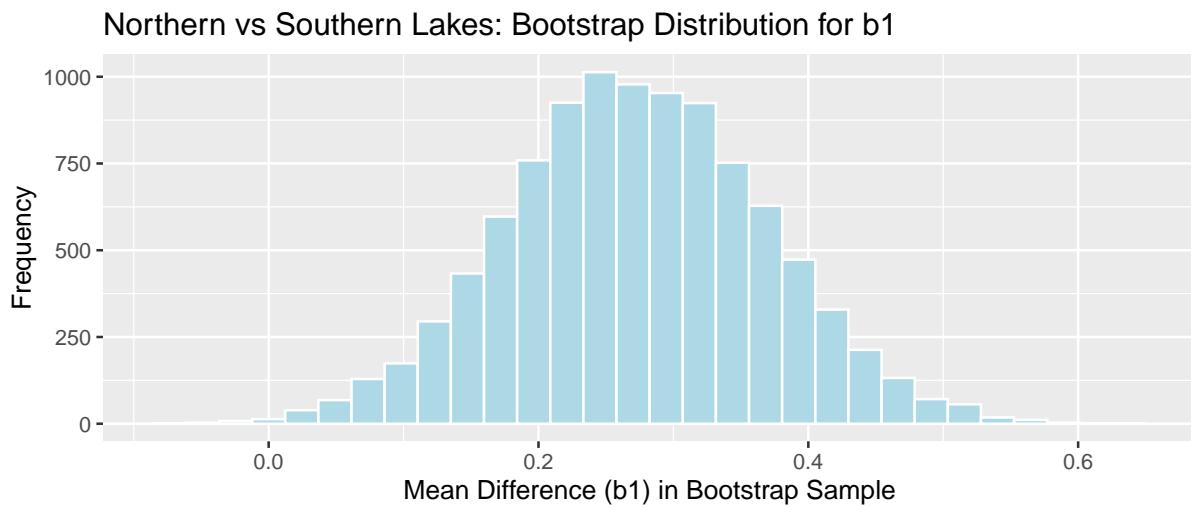
M <- lm(data=FloridaLakes, Mercury~Location) #fit model to original sample
Sample_b1 <- M$coefficients[2] # record b1 value (second coefficient)
Bootstrap_b1 <- rep(NA, 10000) #vector to store b1 values

for (i in 1:10000){
  NLakes <- sample_n(FloridaLakes %>% filter(Location=="N"), 33, replace=TRUE) ## sample 33
  SLakes <- sample_n(FloridaLakes %>% filter(Location=="S"), 20, replace=TRUE) ## sample 20
  BootstrapSample <- rbind(NLakes, SLakes) ## combine Northern and Southern Lakes
  M <- lm(data=BootstrapSample, Mercury ~ Location) ## fit linear model
  Bootstrap_b1[i] <- M$coefficients[2] ## record b1
}
NS_Lakes_Bootstrap_Results <- data.frame(Bootstrap_b1) #save results as dataframe

```

The bootstrap distribution for the difference in means, b_1 , is shown below, along with the standard error for the difference.

```
NS_Lakes_Bootstrap_Plot_b1 <- ggplot(data=NS_Lakes_Bootstrap_Results, aes(x=Bootstrap_b1)) +
  geom_histogram(color="white", fill="lightblue") +
  xlab("Mean Difference (b1) in Bootstrap Sample") + ylab("Frequency") +
  ggtitle("Northern vs Southern Lakes: Bootstrap Distribution for b1")
NS_Lakes_Bootstrap_Plot_b1
```



Bootstrap Standard Error:

We'll calculate the bootstrap standard error of the difference in means b_1 . This is a measure of how much the difference in means varies between samples.

```
SE_b1 <- sd(NS_Lakes_Bootstrap_Results$Bootstrap_b1)
SE_b1
```

```
[1] 0.09549244
```

The bootstrap distribution is symmetric and bell-shaped, so we can use the standard error method to calculate a 95% confidence interval.

$$b_1 \pm 2 \times SE(b_1)$$

$$= 0.271 \pm 2 \times 0.095$$

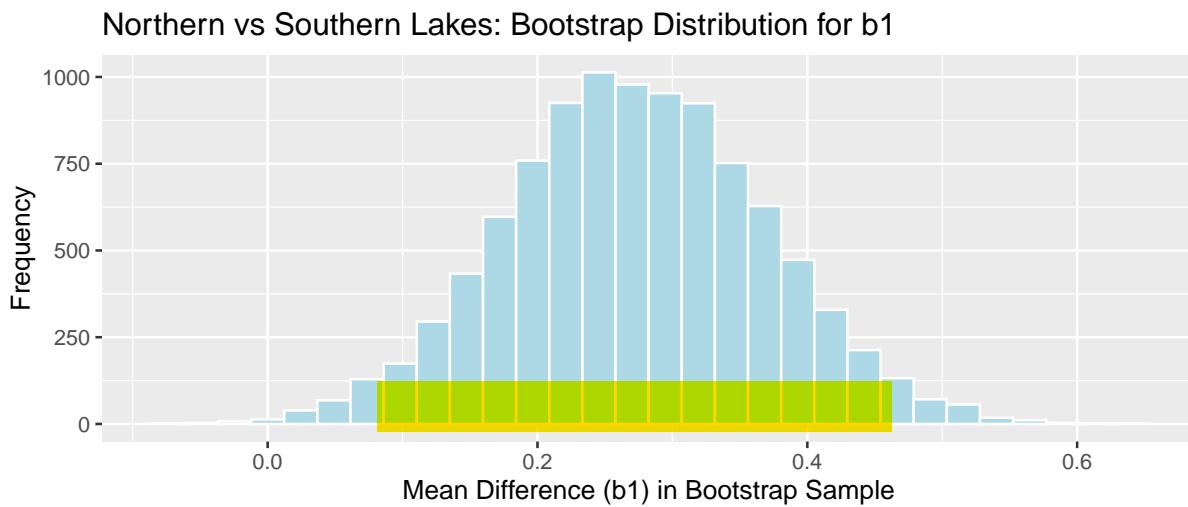
95% Confidence Interval:

```
c(Sample_b1 - 2*SE_b1, Sample_b1 + 2*SE_b1)
```

```
LocationS  LocationS
0.08096967 0.46293942
```

The 95% confidence interval is shown by the gold bar on the graph of the bootstrap distribution below.

```
NS_Lakes_Bootstrap_Plot_b1 +
  geom_segment(aes(x=Sample_b1 - 2*SE_b1, xend=Sample_b1 + 2*SE_b1, y=50, yend=50),
               color="gold", size=10, alpha=0.01)
```

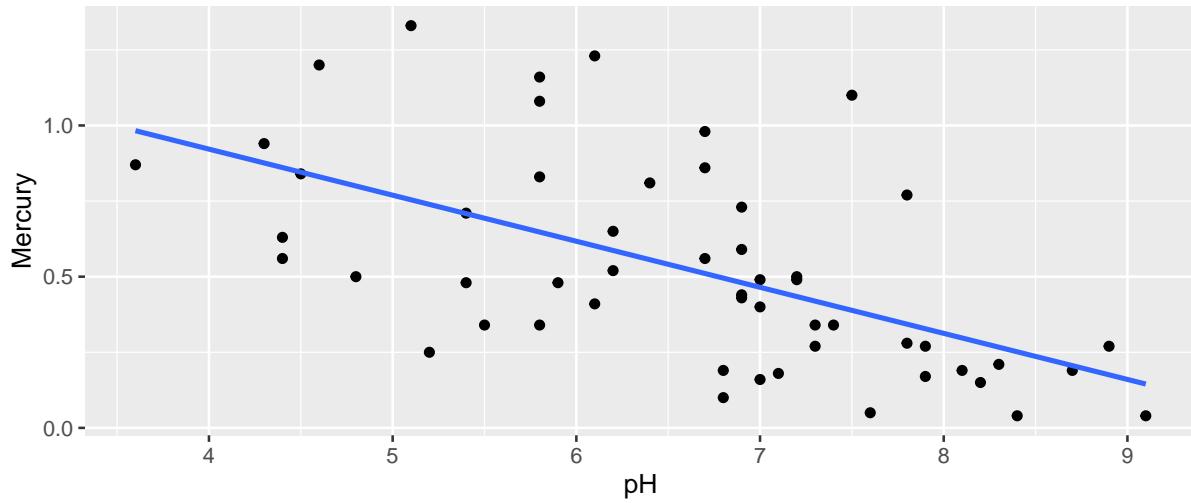


We are 95% confident that the mean mercury level among all lakes in Southern Florida is between 0.08 and 0.46 higher than the mean mercury level among all lakes in Northern Florida.

3.4.6 CI for Regression Slope

Now, we'll examine the relationship between mercury concentration and pH in Florida lakes. The scatterplot displays these variables, along with the least squares regression line.

```
ggplot(data=FloridaLakes, aes(y=Mercury, x=pH)) +
  geom_point() + stat_smooth(method="lm", se=FALSE)
```



The regression equation is

$$\widehat{\text{Mercury}} = b_0 + b_1 \times \text{pH}$$

Regression estimates b_0 and b_1 are shown below.

```
M <- lm(data=FloridaLakes, Mercury~pH)
M
```

```
Call:
lm(formula = Mercury ~ pH, data = FloridaLakes)
```

Coefficients:

(Intercept)	pH
1.5309	-0.1523

- On average, lakes with pH level 0 are expected to have a mercury level of 1.53 ppm.
- For each one-unit increase in pH, mercury level is expected to decrease by 0.15 ppm.

These estimates are sample statistics, calculated from our sample of 53 lakes. We can think of our regression equation estimates b_0 and b_1 as estimates of parameters β_0 and β_1 , which pertain to the slope and intercept of the regression line pertaining to the entire population of all lakes in Florida. We'll use b_0 and b_1 to estimate β_0 and β_1 in the same way that we used sample proportion \hat{p} to estimate population proportion p and sample mean \bar{x} to estimate population mean μ .

The intercept, β_0 has little meaning here, but the slope β_1 represents the average change in mercury level for each one-unit increase in pH, among all Florida lakes. We'll use bootstrapping to find a confidence interval for this quantity.

Bootstrap Steps

1. Take a sample of 53 lakes by randomly sampling from the original sample, with replacement.
2. Fit a regression model with pH as the explanatory variable and record the value of slope b_1 .
3. Repeat steps 1 and 2 many (say 10,000) times, keeping track of slope of the regression line for each bootstrap sample.
4. Look at the distribution of the slopes across bootstrap samples. The variability in this bootstrap distribution can be used to approximate the variability in the sampling distribution for the slope relating mercury and pH levels.

We'll illustrate the procedure on 3 bootstrap samples.

Bootstrap Sample 1

```
BootstrapSample1 <- sample_n(FloridaLakes , 53, replace=TRUE) %>% arrange(ID) %>%
  select(ID, Lake, pH, Mercury) # take bootstrap sample
BootstrapSample1
```

	ID	Lake	pH	Mercury
	<int>	<chr>	<dbl>	<dbl>
1	1	Alligator	6.1	1.23
2	1	Alligator	6.1	1.23
3	2	Annie	5.1	1.33
4	2	Annie	5.1	1.33
5	3	Apopka	9.1	0.04
6	4	Blue Cypress	6.9	0.44
7	5	Brick	4.6	1.2
8	9	Deer Point	5.8	0.83
9	9	Deer Point	5.8	0.83
10	10	Dias	6.4	0.81
		# i 43 more rows		

We fit a regression model to the bootstrap sample and calculate the regression coefficients. We're again interested in the second coefficient, b_1 , which now represents the slope of the regression line.

```
Mb1 <- lm(data=BootstrapSample1, Mercury ~ pH) # fit linear model  
Mb1
```

Call:
lm(formula = Mercury ~ pH, data = BootstrapSample1)

Coefficients:
(Intercept) pH
1.7542 -0.1774

Bootstrap Sample 2

```
BootstrapSample2 <- sample_n(FloridaLakes , 53, replace=TRUE) %>% arrange(ID) %>%  
  select(ID, Lake, pH, Mercury)  
BootstrapSample2
```

```
# A tibble: 53 x 4  
  ID Lake          pH Mercury  
  <int> <chr>     <dbl>   <dbl>  
1 1 Alligator    6.1    1.23  
2 2 Annie        5.1    1.33  
3 2 Annie        5.1    1.33  
4 3 Apopka       9.1    0.04  
5 3 Apopka       9.1    0.04  
6 5 Brick         4.6    1.2  
7 5 Brick         4.6    1.2  
8 6 Bryant        7.3    0.27  
9 6 Bryant        7.3    0.27  
10 7 Cherry       5.4    0.48  
# i 43 more rows
```

```
Mb2 <- lm(data=BootstrapSample2, Mercury ~ pH) # fit linear model  
Mb2
```

Call:
lm(formula = Mercury ~ pH, data = BootstrapSample2)

Coefficients:
(Intercept) pH
1.752 -0.189

Bootstrap Sample 3

```
BootstrapSample3 <- sample_n(FloridaLakes , 53, replace=TRUE) %>% arrange(ID) %>%
  select(ID, Lake, pH, Mercury)
BootstrapSample3
```

```
# A tibble: 53 x 4
  ID     Lake      pH  Mercury
  <int> <chr>    <dbl>   <dbl>
1 2     Annie     5.1    1.33
2 5     Brick     4.6    1.2 
3 6     Bryant    7.3    0.27
4 7     Cherry    5.4    0.48
5 10    Dias      6.4    0.81
6 11    Dorr      5.4    0.71
7 14    East Tohopekaliga 5.8    1.16
8 14    East Tohopekaliga 5.8    1.16
9 15    Farm-13   7.6    0.05
10 15   Farm-13   7.6    0.05
# i 43 more rows
```

```
Mb3 <- lm(data=BootstrapSample3, Mercury ~ pH) # fit linear model
Mb3
```

Call:

```
lm(formula = Mercury ~ pH, data = BootstrapSample3)
```

Coefficients:

(Intercept)	pH
1.2629	-0.1102

We'll now take 10,000 different bootstrap samples and look at the bootstrap distribution for b_1 , the slope of the regression line relating mercury level and pH.

```
M <- lm(data=FloridaLakes, Mercury~pH) #fit model to original sample
Sample_b1 <- M$coefficients[2] # record b1 value (second coefficient)
Bootstrap_b1 <- rep(NA, 10000) #vector to store b1 values

for (i in 1:10000){
  BootstrapSample <- sample_n(FloridaLakes , 53, replace=TRUE) #take bootstrap sample
```

```

M <- lm(data=BootstrapSample, Mercury ~ pH) # fit linear model
Bootstrap_b1[i] <- M$coefficients[2] # record b1
}
Lakes_Bootstrap_Slope_Results <- data.frame(Bootstrap_b1) #save results as dataframe

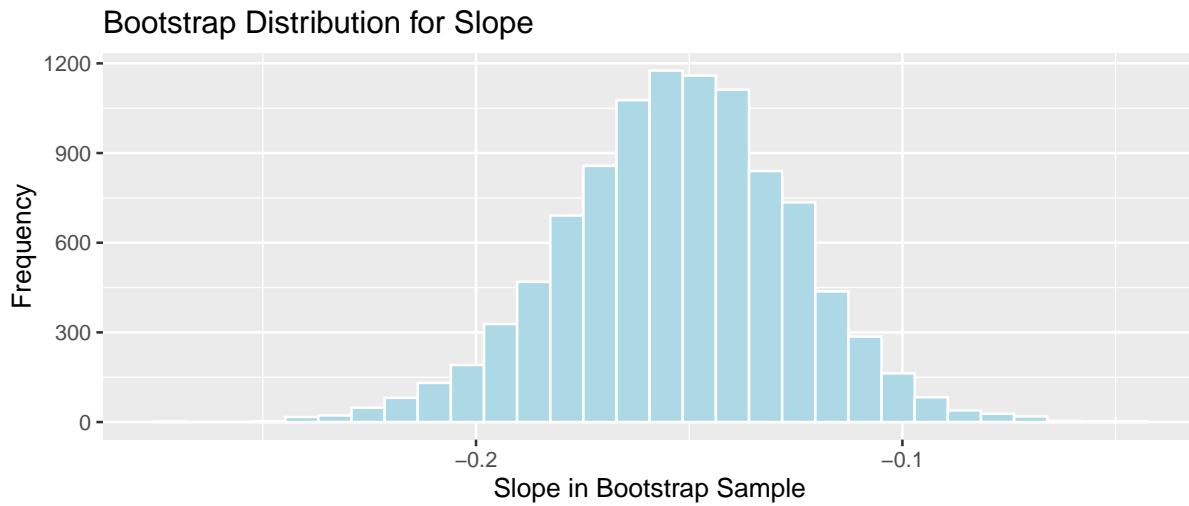
```

The bootstrap distribution for the slopes, b_1 , is shown below, along with the standard error for the difference.

```

Lakes_Bootstrap_Plot_Slope <- ggplot(data=Lakes_Bootstrap_Slope_Results, aes(x=Bootstrap_b1))
  geom_histogram(color="white", fill="lightblue") +
  xlab("Slope in Bootstrap Sample") + ylab("Frequency") +
  ggtitle("Bootstrap Distribution for Slope")
Lakes_Bootstrap_Plot_Slope

```



Bootstrap Standard Error:

We'll calculate the bootstrap standard error of the slope b_1 . This is a measure of how much the slope varies between samples.

```

SE_b1 <- sd(Lakes_Bootstrap_Slope_Results$Bootstrap_b1)
SE_b1

```

[1] 0.02694529

The bootstrap distribution is symmetric and bell-shaped, so we can use the standard error method to calculate a 95% confidence interval.

$$b_1 \pm 2 \times \text{SE}(b_1) \\ = -0.1523 \pm 2 \times 0.027$$

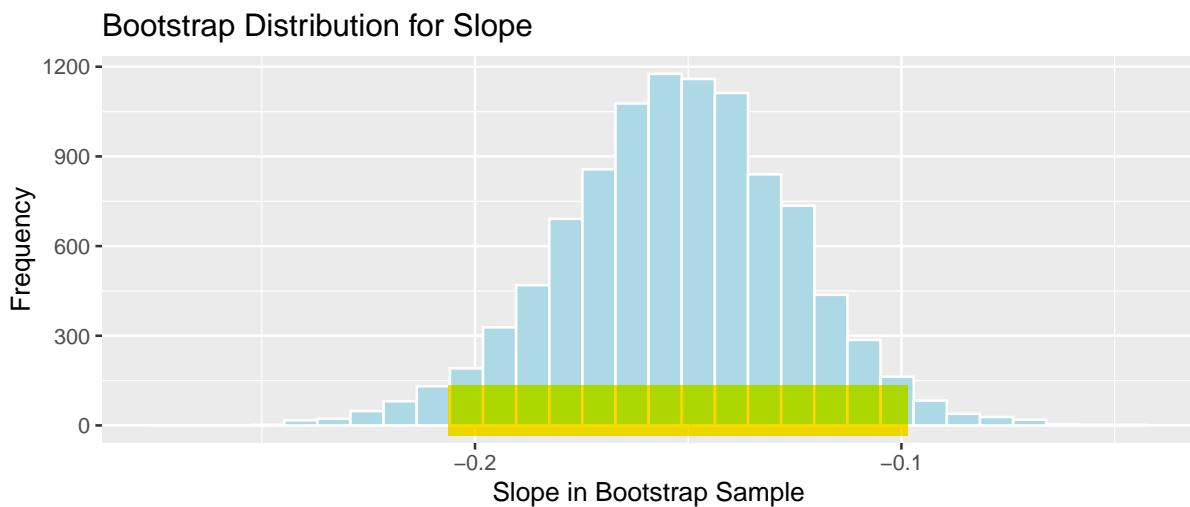
95% Confidence Interval:

```
c(Sample_b1 - 2*SE_b1, Sample_b1 + 2*SE_b1)
```

pH	pH
-0.20619145	-0.09841028

The 95% confidence interval is shown by the gold bar on the graph of the bootstrap distribution below.

```
Lakes_Bootstrap_Plot_Slope +
  geom_segment(aes(x=Sample_b1 - 2*SE_b1,xend=Sample_b1 + 2*SE_b1, y=50, yend=50),
               color="gold", size=10, alpha=0.01)
```



We are 95% confident that among all Florida lakes, for each 1 unit increase in pH, mercury level decreases between 0.2 and 0.1, on average.

3.4.7 CI for Regression Response

In addition to calculating a confidence interval for the slope of the regression line relating mercury and pH levels in a lake, we can also calculate a confidence interval for the average mercury level among all lakes with a given pH.

We'll calculate a confidence interval for the average mercury level among all lakes with a neutral pH level of 7.

The regression equation is

$$\begin{aligned}\widehat{\text{Mercury}} &= b_0 + b_1 \times \text{pH} \\ &= 1.5309 - 0.1523 \times \text{pH}\end{aligned}$$

so the expected mercury level among all lakes with pH = 7 is $b_0 + 7b_1 = 1.5309 - 0.1523(7) = 0.4648$ ppm.

This quantity is a statistic calculated from a sample of 53 lakes, so we would not expect the average mercury level among all lakes in the population to be exactly equal to 0.4648. Again, we'll use this sample statistic as an estimate of the population parameter, and use bootstrapping to estimate the variability associated with this statistic, in order to make a confidence interval.

Bootstrap Steps

1. Take a sample of 53 lakes by randomly sampling from the original sample, with replacement.
2. Fit a regression model with location as the explanatory variable and record the values of b_0 and b_1 . Use these to calculate $b_0 + 7b_1$.
3. Repeat steps 1 and 2 many (say 10,000) times, keeping track of b_0 and b_1 , and calculating $b_0 + 7b_1$ in each bootstrap sample.
4. Look at the distribution of the expected response, $b_0 + 7b_1$, across bootstrap samples. The variability in this bootstrap distribution can be used to approximate the variability in the sampling distribution for the expected mercury level among all lakes with pH level of 7.

We'll illustrate the procedure on 3 bootstrap samples.

Bootstrap Sample 1

```
BootstrapSample1 <- sample_n(FloridaLakes , 53, replace=TRUE) %>% arrange(ID) %>%
  select(ID, Lake, pH, Mercury)    # take bootstrap sample
BootstrapSample1
```

```
# A tibble: 53 x 4
  ID   Lake          pH  Mercury
  <int> <chr>      <dbl>    <dbl>
1     1 Alligator    6.1     1.23
```

```

2      3 Apopka      9.1    0.04
3      3 Apopka      9.1    0.04
4      6 Bryant       7.3    0.27
5      7 Cherry       5.4    0.48
6      8 Crescent     8.1    0.19
7      8 Crescent     8.1    0.19
8      9 Deer Point   5.8    0.83
9      9 Deer Point   5.8    0.83
10     10 Dias        6.4    0.81
# i 43 more rows

```

We fit a regression model to the bootstrap sample and calculate the regression coefficients. We're interested in the second coefficient, b_1 , which represents the mean difference between lakes in Southern and Northern Florida

```

Mb1 <- lm(data=BootstrapSample1, Mercury ~ pH) ## fit linear model
b0 <- Mb1$coefficients[1] # record value of b0 (first coefficient)
b1 <- Mb1$coefficients[2] # record value of b1 (second coefficient)
b0+7*b1 #calculate b0+7*b1

```

```
(Intercept)
0.4353724
```

Bootstrap Sample 2

```

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

```

```

# A tibble: 53 x 4
  ID Lake          pH Mercury
  <int> <chr>     <dbl>   <dbl>
1 2 Annie        5.1    1.33
2 2 Annie        5.1    1.33
3 3 Apopka       9.1    0.04
4 4 Blue Cypress 6.9    0.44
5 6 Bryant        7.3    0.27
6 6 Bryant        7.3    0.27
7 10 Dias         6.4    0.81
8 10 Dias         6.4    0.81
9 12 Down         7.2    0.5

```

```
10      12 Down          7.2      0.5  
# i 43 more rows
```

```
Mb2 <- lm(data=BootstrapSample2, Mercury ~ pH) # fit linear model  
b0 <- Mb2$coefficients[1] # record value of b0 (first coefficient)  
b1 <- Mb2$coefficients[2] # record value of b1 (second coefficient)  
b0+7*b1 #calculate b0+7*b1
```

```
(Intercept)  
0.5738289
```

Bootstrap Sample 3

```
BootstrapSample3 <- sample_n(FloridaLakes , 53, replace=TRUE) %>% arrange(ID) %>%  
  select(ID, Lake, pH, Mercury)  
BootstrapSample3
```

```
# A tibble: 53 x 4  
  ID Lake      pH Mercury  
  <int> <chr>    <dbl>   <dbl>  
1 2 Annie     5.1    1.33  
2 3 Apopka    9.1    0.04  
3 6 Bryant    7.3    0.27  
4 6 Bryant    7.3    0.27  
5 7 Cherry    5.4    0.48  
6 7 Cherry    5.4    0.48  
7 8 Crescent   8.1    0.19  
8 10 Dias     6.4    0.81  
9 10 Dias     6.4    0.81  
10 10 Dias    6.4    0.81  
# i 43 more rows
```

```
Mb3 <- lm(data=BootstrapSample3, Mercury ~ pH) # fit linear model  
b0 <- Mb3$coefficients[1] # record value of b0 (first coefficient)  
b1 <- Mb3$coefficients[2] # record value of b1 (second coefficient)  
b0+7*b1 #calculate b0+7*b1
```

```
(Intercept)  
0.4223812
```

We'll now take 10,000 different bootstrap samples and record the values of b_0 , b_1 , which we'll then use to calculate $b_0 + 7b_1$.

```
M <- lm(data=FloridaLakes, Mercury~pH) #fit model to original sample
Sample_b0 <- M$coefficients[1] # record b0 value (second coefficient)
Sample_b1 <- M$coefficients[2] # record b1 value (second coefficient)
Sample_Exp7 <- Sample_b0 + 7*Sample_b1 # calculate sample expected mercury when pH=7
Bootstrap_b0 <- rep(NA, 10000) #vector to store b0 values
Bootstrap_b1 <- rep(NA, 10000) #vector to store b1 values

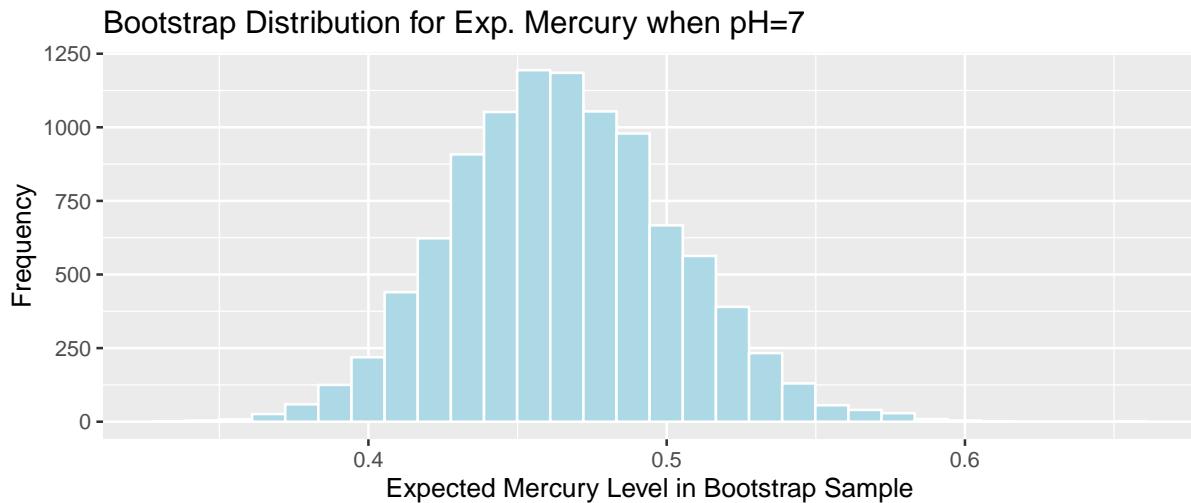
for (i in 1:10000){
  BootstrapSample <- sample_n(FloridaLakes , 53, replace=TRUE) #take bootstrap sample
  M <- lm(data=BootstrapSample, Mercury ~ pH) # fit linear model
  Bootstrap_b0[i] <- M$coefficients[1] # record b0
  Bootstrap_b1[i] <- M$coefficients[2] # record b1
}

Bootstrap_Exp7 <-  Bootstrap_b0 + 7*Bootstrap_b1 # calcualte expected response for each boot

Lakes_Bootstrap_Exp7_Results <- data.frame(Bootstrap_b0, Bootstrap_b1, Bootstrap_Exp7) #save
```

The bootstrap distribution for the expected mercury level among all lakes with pH level 7, $b_0 + 7b_1$, is shown below, along with the standard error for this quantity.

```
Lakes_Bootstrap_Plot_Exp7 <- ggplot(data=Lakes_Bootstrap_Exp7_Results, aes(x=Bootstrap_Exp7))
  geom_histogram(color="white", fill="lightblue") +
  xlab("Expected Mercury Level in Bootstrap Sample") + ylab("Frequency") +
  ggtitle( "Bootstrap Distribution for Exp. Mercury when pH=7")
Lakes_Bootstrap_Plot_Exp7
```



Bootstrap Standard Error:

We'll calculate the bootstrap standard error of expected mercury concentration $b_0 + 7b_1$. This is a measure of how much the estimated expected concentration varies between samples.

```
SE_Exp7 <- sd(Lakes_Bootstrap_Exp7_Results$Bootstrap_Exp7)
SE_Exp7
```

```
[1] 0.03702503
```

Again, the bootstrap distribution is symmetric and bell-shaped, so we can use the standard error method to calculate a 95% confidence interval.

$$b_1 \pm 2 \times \text{SE}(b_1)$$

$$= 0.4648 \pm 2 \times 0.037$$

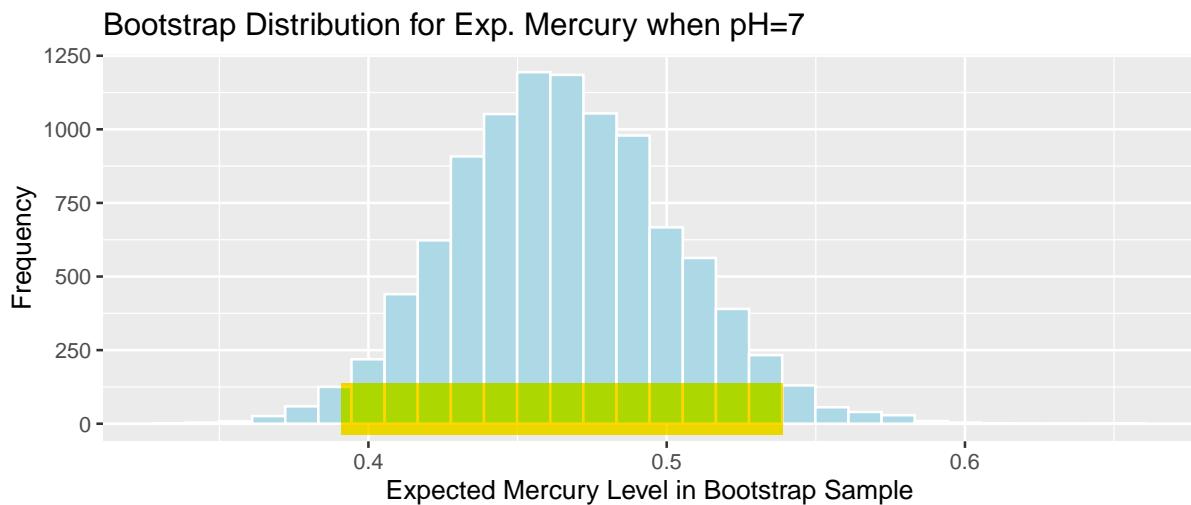
95% Confidence Interval:

```
c(Sample_Exp7 - 2*SE_Exp7, Sample_Exp7 + 2*SE_Exp7)
```

```
(Intercept) (Intercept)
0.3907626 0.5388627
```

The 95% confidence interval is shown by the gold bar on the graph of the bootstrap distribution below.

```
Lakes_Bootstrap_Plot_Exp7 +
  geom_segment(aes(x=Sample_Exp7 - 2*SE_Exp7, xend=Sample_Exp7 + 2*SE_Exp7, y=50, yend=50),
               color="gold", size=10, alpha=0.01)
```



We are 95% confident that average mercury level among all Florida lakes with pH level 7 is between 0.4 and 0.5 ppm.

Again, we are not saying that we think an individual like with a pH level of 7 will lie in this range, only that the average mercury level among all such lakes lies in this range.

3.4.8 More CI's in Regression

We saw in the previous two examples how to calculate a confidence interval for the slope of a regression line, and for an expected response in regression. In fact, we can calculate confidence intervals for any function involving regression coefficients $\beta_0, \beta_1, \dots, \beta_p$, in a similar manner.

For example, let's consider the model for Seattle house prices that involved square feet, whether or not the house was on the waterfront, and an interaction term between these variables.

The model is

$$\widehat{Price} = b_0 + b_1 \times \text{Sq. Ft.} + b_2 \times \text{waterfront} + b_3 \times \text{Sq.Ft} \times \text{Waterfront}$$

We fit the model and obtain the parameter estimates shown below.

```

M <- lm(data= Houses, price~sqft_living + waterfront +
          sqft_living:waterfront) #fit model to original sample
Sample_b0 <- M$coefficients[1] # record b0 value (second coefficient)
Sample_b1 <- M$coefficients[2] # record b1 value (second coefficient)
Sample_b2 <- M$coefficients[3] # record b1 value (second coefficient)
Sample_b3 <- M$coefficients[4] # record b1 value (second coefficient)
M

```

Call:

```
lm(formula = price ~ sqft_living + waterfront + sqft_living:waterfront,
   data = Houses)
```

Coefficients:

(Intercept)	sqft_living
67.3959	0.2184
waterfrontYes	sqft_living:waterfrontYes
-364.5950	0.4327

Consider the following quantities that we might be interested in estimating:

1. The expected price of a 2,000 square foot waterfront house.
2. The expected price of a 1,500 square foot non-waterfront house.
3. The difference between the expected price of a house 1,800 square foot house on the waterfront, compared to a house the same size that is not on the waterfront.
4. The difference in the rate of change in house prices for each additional 100 square feet for houses on the waterfront, compared to houses not on the waterfront.

Each of these quantities can be expressed as a linear function of our regression coefficients b_0, b_1, b_2, b_3 . We just need to find the appropriate function of the b_j 's, and then calculate a bootstrap confidence interval for that quantity, using the same steps we've seen in the previous examples.

Substituting into the regression equation, we see that:

1. The expected price of a 2,000 square foot waterfront house is given by

$$b_0 + 2000b_1 + b_2 + 2000b_3$$

We calculate this estimate from the model, based on our sample of 100 houses:

```
2000*Sample_b1 +Sample_b2+2000*Sample_b3 # calculate b0+2000b1+b2+2000b3
```

```
sqft_living  
937.4777
```

We estimate that the average price of all 2,000 square foot waterfront houses in Seattle is 937 thousand dollars.

2. The expected price of a 1,500 square foot non-waterfront house is given by

$$b_0 + 1500b_1$$

```
Sample_b0 + 1500*Sample_b1 # calculate b0+1500b1+
```

```
(Intercept)  
394.9499
```

We estimate that the average price of all 1,500 square foot non-waterfront houses in Seattle is 395 thousand dollars.

3. The difference between the expected price of a house 1,800 square foot house on the waterfront, compared to a house the same size that is not on the waterfront is given by:

$$\begin{aligned}(b_0 + 1800b_1 + b_2 + 1800b_3) - (b_0 + 1800b_1) \\= b_2 + 1800b_3\end{aligned}$$

```
Sample_b2+1800*Sample_b3 # calculate b2+1800b3
```

```
waterfrontYes  
414.2058
```

We estimate that on average a 1,800 square foot house on the waterfront will cost 414 thousand dollars more than a 1,800 square foot house not on the waterfront.

4. The difference in the rate of change in house prices for each additional 100 square feet for houses on the waterfront, compared to houses not on the waterfront.

This question is asking about the difference in slopes of the regression lines relating price and square feet for houses on the waterfront, compared to those not on the waterfront.

For houses on the waterfront, the regression equation is

$$\widehat{\text{Price}} = (b_0 + b_2) + (b_1 + b_3) \times \text{Sq. Ft.},$$

so the slope is $b_1 + b_3$.

For houses not on the waterfront, the regression equation is

$$\widehat{\text{Price}} = b_0 + b_1 \times \text{Sq. Ft.},$$

so the slope is b_1 .

These slope pertain to the expected change in price for each additional 1 square foot. So, for a 100-square foot increase, the price of a waterfront house is expected to increase by $100(b_1 + b_3)$, compared to an increase of $100b_1$ for a non-waterfront house. Thus, the difference in the rates of change is $100b_3$.

```
100*Sample_b3 # calculate 100b3
```

```
sqft_living:waterfrontYes  
43.26671
```

We estimate that the price of waterfront houses increases by 43 thousand dollars more for each additional 100 square feet than the price of non-waterfront houses.

These estimates calculated from the sample are statistics, which, like all the other statistics we've seen are likely to vary from the true values of the corresponding population parameters, due to variability between samples. We can use bootstrapping to calculate confidence intervals for the relevant population parameters, using these sample statistics (the functions of b_j 's), just as we've done for the other statistics we've seen.

Bootstrap Steps

1. Take a sample of 100 houses by randomly sampling from the original sample, with replacement.
2. Fit a regression model with location as the explanatory variable and record the values of regression coefficients b_0, b_1, b_2, b_3 . Use these to calculate each of the four desired quantities (i.e. $b_0 + 2000b_1 + b_2 + 2000b_3$)
3. Repeat steps 1 and 2 many (say 10,000) times, keeping track of the regression coefficients and calculating the desired quantities in each bootstrap sample.

4. Look at the distribution of the quantities of interest, across bootstrap samples. The variability in this bootstrap distribution can be used to approximate the variability in the sampling distribution for each of these quantities.

We'll illustrate the procedure on 3 bootstrap samples.

Bootstrap Sample 1

We take the first bootstrap sample and fit a model with interaction. For brevity, we won't list out the houses in each of the bootstrap samples, as the idea should be clear by now. Model coefficients are shown below.

```
BootstrapSample1 <- sample_n(Houses , 100, replace=TRUE) %>% arrange(Id) %>%
  select(Id, price, sqft_living, waterfront)

Mb1 <- lm(data=BootstrapSample1, price ~ sqft_living + waterfront + sqft_living:waterfront)
b0 <- Mb1$coefficients[1] # record value of b0 (first coefficient)
b1 <- Mb1$coefficients[2] # record value of b1 (second coefficient)
b2 <- Mb1$coefficients[3] # record value of b2 (third coefficient)
b3 <- Mb1$coefficients[4] # record value of b3 (fourth coefficient)
Mb1
```

Call:

```
lm(formula = price ~ sqft_living + waterfront + sqft_living:waterfront,
   data = BootstrapSample1)
```

Coefficients:

(Intercept)	sqft_living
69.4180	0.2308
waterfrontYes	sqft_living:waterfrontYes
-444.1795	0.4229

We calculate each of the four desired quantities.

```
b0+2000*b1 + b2 + 2000*b3
```

```
(Intercept)
932.7184
```

```
b0+1500*b1
```

```
(Intercept)
415.6311
```

```
b2+1800*b3
```

```
waterfrontYes
317.0968
```

```
100*b3
```

```
sqft_living:waterfrontYes
42.29312
```

Bootstrap Sample 2

```
BootstrapSample2 <- sample_n(Houses , 100, replace=TRUE) %>% arrange(Id) %>%
  select(Id, price, sqft_living, waterfront)

Mb2 <- lm(data=BootstrapSample2, price ~ sqft_living + waterfront + sqft_living:waterfront)
b0 <- Mb2$coefficients[1] # record value of b0 (first coefficient)
b1 <- Mb2$coefficients[2] # record value of b1 (second coefficient)
b2 <- Mb2$coefficients[3] # record value of b2 (third coefficient)
b3 <- Mb2$coefficients[4] # record value of b3 (fourth coefficient)
Mb2
```

Call:

```
lm(formula = price ~ sqft_living + waterfront + sqft_living:waterfront,
  data = BootstrapSample2)
```

Coefficients:

	(Intercept)	sqft_living
	118.2805	0.1840
waterfrontYes	sqft_living:waterfrontYes	
	-337.1774	0.4875

We calculate each of the four desired quantities.

```
b0+2000*b1 + b2 + 2000*b3
```

```
(Intercept)  
1124.13
```

```
b0+1500*b1
```

```
(Intercept)  
394.2376
```

```
b2+1800*b3
```

```
waterfrontYes  
540.3978
```

```
100*b3
```

```
sqft_living:waterfrontYes  
48.75418
```

Bootstrap Sample 3

```
BootstrapSample3 <- sample_n(Houses , 100, replace=TRUE) %>% arrange(Id) %>%  
  select(Id, price, sqft_living, waterfront)  
  
Mb3 <- lm(data=BootstrapSample3, price ~ sqft_living + waterfront + sqft_living:waterfront)  
b0 <- Mb3$coefficients[1] # record value of b0 (first coefficient)  
b1 <- Mb3$coefficients[2] # record value of b1 (second coefficient)  
b2 <- Mb3$coefficients[3] # record value of b2 (third coefficient)  
b3 <- Mb3$coefficients[4] # record value of b3 (fourth coefficient)  
Mb3
```

Call:

```
lm(formula = price ~ sqft_living + waterfront + sqft_living:waterfront,  
  data = BootstrapSample3)
```

Coefficients:

	(Intercept)	sqft_living
	52.4142	0.2223
waterfrontYes	sqft_living:waterfrontYes	
	-260.6892	0.3865

We calculate each of the four desired quantities.

```
b0+2000*b1 + b2 + 2000*b3
```

(Intercept)

1009.22

```
b0+1500*b1
```

(Intercept)

385.811

```
b2+1800*b3
```

waterfrontYes

434.9802

```
100*b3
```

```
sqft_living:waterfrontYes
```

38.6483

We'll now take 10,000 different bootstrap samples and record the values of b_0 , b_1 , b_3 , and b_4 , which we'll then use to calculate each of our four desired quantities.

```
M <- lm(data=Houses, price~sqft_living + waterfront +
          sqft_living:waterfront) #fit model to original sample
Sample_b0 <- M$coefficients[1] # record b0 value (second coefficient)
Sample_b1 <- M$coefficients[2] # record b1 value (second coefficient)
Sample_b2 <- M$coefficients[3] # record b1 value (second coefficient)
Sample_b3 <- M$coefficients[4] # record b1 value (second coefficient)
Sample_Q1 <- Sample_b0 + 2000*Sample_b1 + Sample_b2+2000*Sample_b3 # calculate b0+2000b1+b2+2000b3
Sample_Q2 <- Sample_b0 + 1500*Sample_b1 # calculate b0+1500b1+
Sample_Q3 <- Sample_b2+1800*Sample_b3 # calculate b2+1800b3
Sample_Q4 <- 100*Sample_b3 # calculate 100b3

Bootstrap_b0 <- rep(NA, 10000) #vector to store b0 values
Bootstrap_b1 <- rep(NA, 10000) #vector to store b1 values
Bootstrap_b2 <- rep(NA, 10000) #vector to store b2 values
Bootstrap_b3 <- rep(NA, 10000) #vector to store b3 values
```

```

for (i in 1:10000){
  BootstrapSample <- sample_n(Houses, 1000, replace=TRUE)    #take bootstrap sample
  Mb <- lm(data=BootstrapSample, price ~ sqft_living +
            waterfront + sqft_living:waterfront) # fit linear model with interaction
  Bootstrap_b0[i] <- Mb$coefficients[1] # record value of b0 (first coefficient)
  Bootstrap_b1[i] <- Mb$coefficients[2] # record value of b1 (second coefficient)
  Bootstrap_b2[i] <- Mb$coefficients[3] # record value of b2 (third coefficient)
  Bootstrap_b3[i] <- Mb$coefficients[4] # record value of b3 (fourth coefficient)
}

Bootstrap_Q1 <- Bootstrap_b0 + 2000*Bootstrap_b1 + Bootstrap_b2 + 2000*Bootstrap_b3
Bootstrap_Q2 <- Bootstrap_b0 + 1500*Bootstrap_b1
Bootstrap_Q3 <- Bootstrap_b2 + 1800*Bootstrap_b3
Bootstrap_Q4 <- 100*Bootstrap_b3

Houses_Bootstrap_Results <- data.frame(Bootstrap_b0, Bootstrap_b1, Bootstrap_b2, Bootstrap_b3, Bootstrap_Q1, Bootstrap_Q2, Bootstrap_Q3, Bootstrap_Q4)

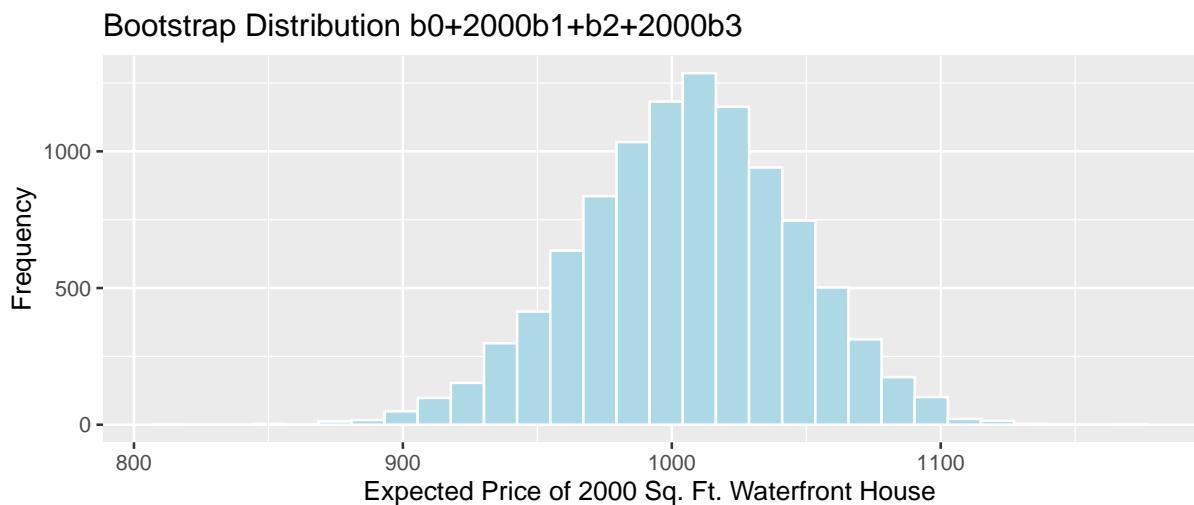
```

Bootstrap Distribution for $b_0 + 2000b_1 + b_2 + 2000b_3$

```

Houses_Bootstrap_Plot_Q1 <- ggplot(data=Houses_Bootstrap_Results,
                                      aes(x=Bootstrap_Q1)) +
  geom_histogram(color="white", fill="lightblue") +
  xlab("Expected Price of 2000 Sq. Ft. Waterfront House") + ylab("Frequency") +
  ggtitle( "Bootstrap Distribution b0+2000b1+b2+2000b3")
Houses_Bootstrap_Plot_Q1

```



Standard Error:

```
SE_Q1 <- sd(Houses_Bootstrap_Results$Bootstrap_Q1)  
SE_Q1
```

```
[1] 40.03112
```

The bootstrap distribution is symmetric and bell-shaped, so we can use the standard error method to calculate a 95% confidence interval.

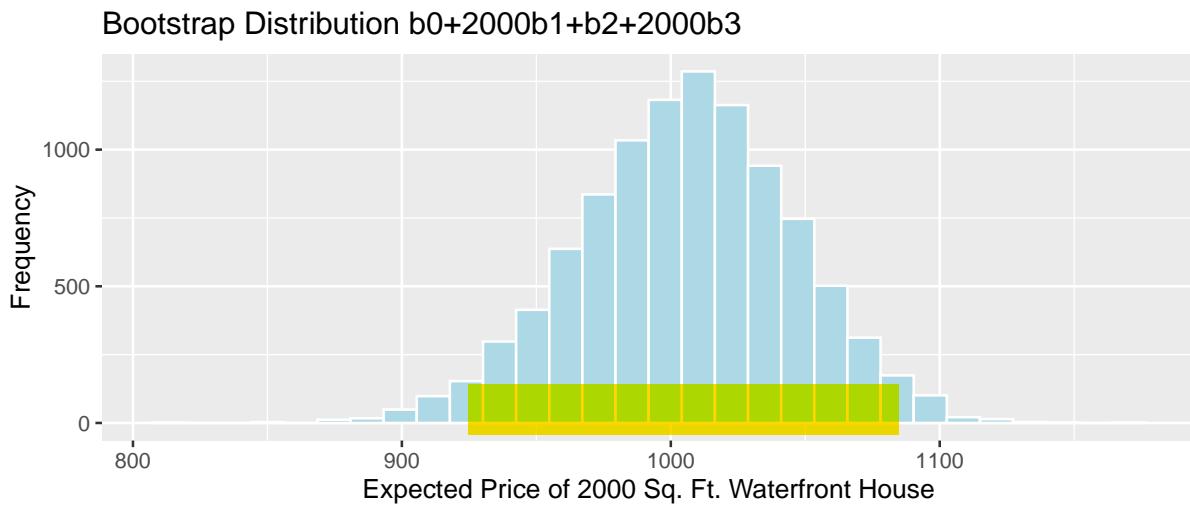
95% Confidence Interval:

```
c(Sample_Q1 - 2*SE_Q1, Sample_Q1 + 2*SE_Q1)
```

```
(Intercept) (Intercept)  
924.8114 1084.9359
```

The 95% confidence interval is shown by the gold bar on the graph of the bootstrap distribution below.

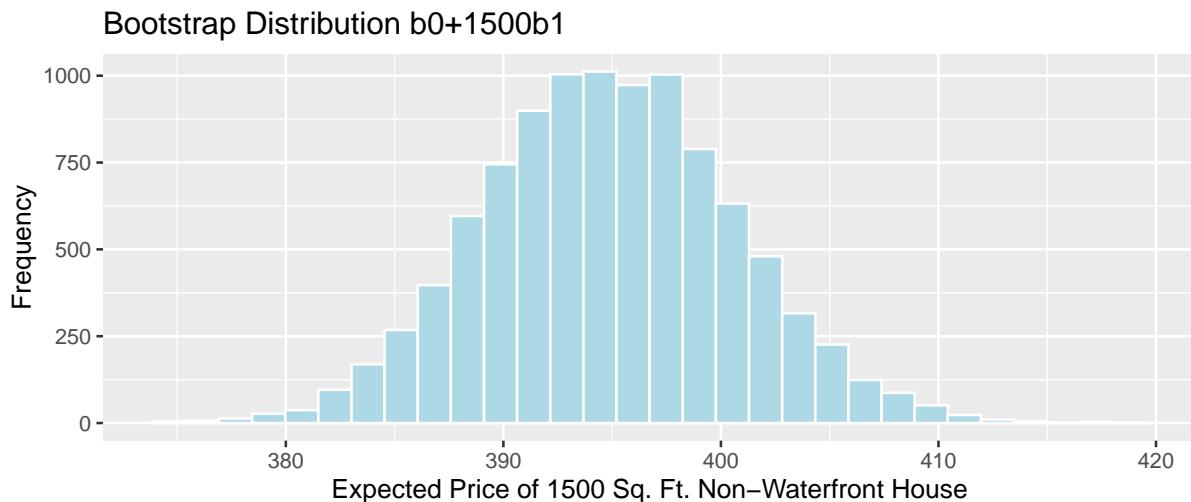
```
Houses_Bootstrap_Plot_Q1 +  
  geom_segment(aes(x=Sample_Q1 - 2*SE_Q1,xend=Sample_Q1 + 2*SE_Q1, y=50, yend=50),  
               color="gold", size=10, alpha=0.01)
```



We are 95% confident that average price among all 2,000 square foot Seattle waterfront houses is between 924.8114379 and 1084.9359281 thousand dollars.

Bootstrap Distribution for $b_0 + 1500b_1$

```
Houses_Bootstrap_Plot_Q2 <- ggplot(data=Houses_Bootstrap_Results,
  aes(x=Bootstrap_Q2)) +
  geom_histogram(color="white", fill="lightblue") +
  xlab("Expected Price of 1500 Sq. Ft. Non-Waterfront House") + ylab("Frequency") +
  ggtitle( "Bootstrap Distribution b0+1500b1")
Houses_Bootstrap_Plot_Q2
```



Standard Error:

```
SE_Q2 <- sd(Houses_Bootstrap_Results$Bootstrap_Q2)
SE_Q2
```

```
[1] 5.812557
```

The bootstrap distribution is symmetric and bell-shaped, so we can use the standard error method to calculate a 95% confidence interval.

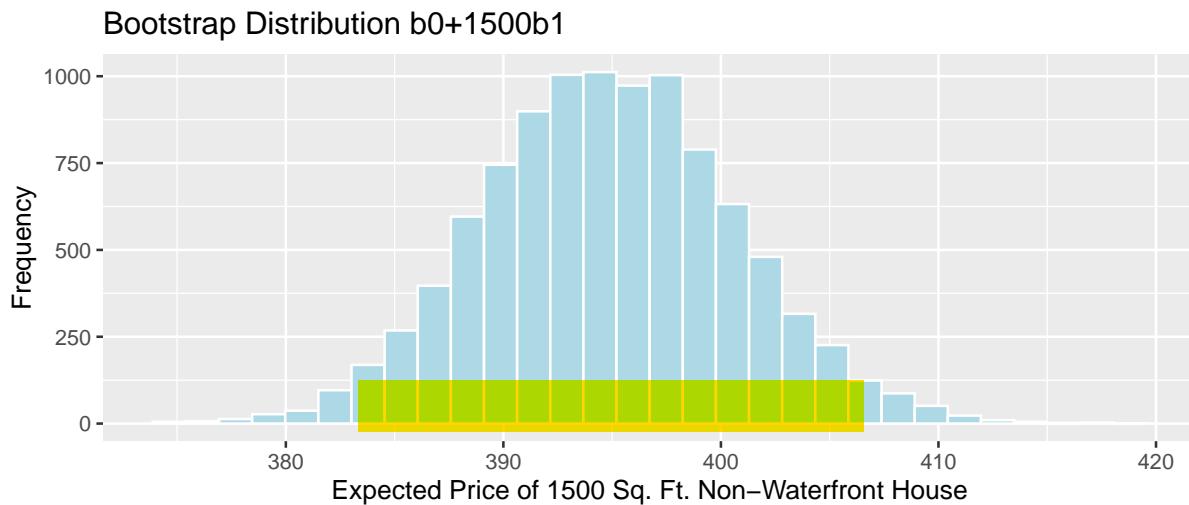
95% Confidence Interval:

```
c(Sample_Q2 - 2*SE_Q2, Sample_Q2 + 2*SE_Q2)
```

```
(Intercept) (Intercept)
383.3247    406.5750
```

The 95% confidence interval is shown by the gold bar on the graph of the bootstrap distribution below.

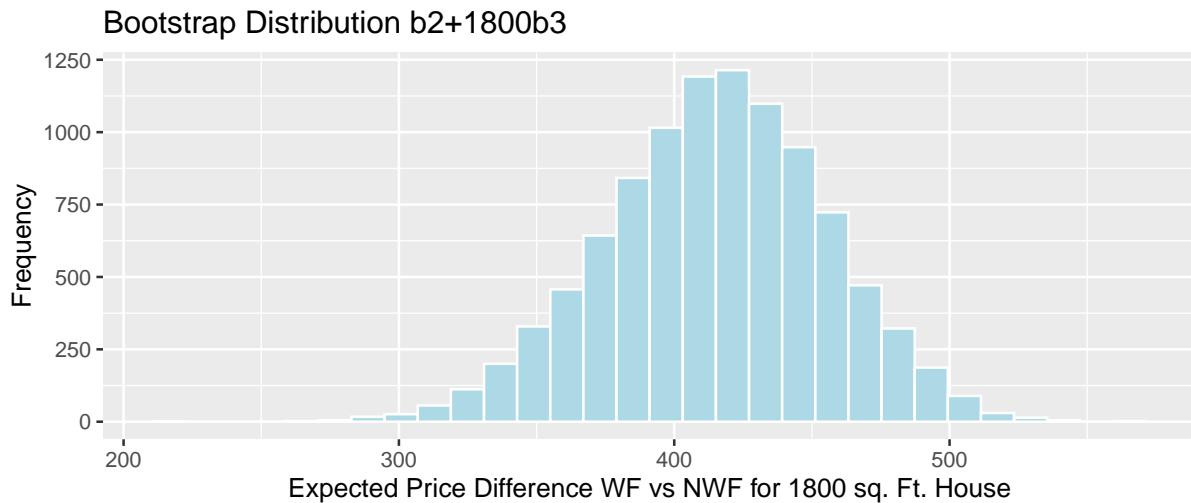
```
Houses_Bootstrap_Plot_Q2 +
  geom_segment(aes(x=Sample_Q2 - 2*SE_Q2,xend=Sample_Q2 + 2*SE_Q2, y=50, yend=50),
               color="gold", size=10, alpha=0.01)
```



We are 95% confident that average price among all 1,500 square foot Seattle non-waterfront houses is between 383.3247413 and 406.5749705 thousand dollars.

Bootstrap Distribution for $b_2 + 1800b_3$

```
Houses_Bootstrap_Plot_Q3 <- ggplot(data=Houses_Bootstrap_Results,
                                      aes(x=Bootstrap_Q3)) +
  geom_histogram(color="white", fill="lightblue") +
  xlab("Expected Price Difference WF vs NWF for 1800 sq. Ft. House") + ylab("Frequency") +
  ggttitle( "Bootstrap Distribution b2+1800b3")
Houses_Bootstrap_Plot_Q3
```



Standard Error:

```
SE_Q3 <- sd(Houses_Bootstrap_Results$Bootstrap_Q3)
SE_Q3
```

```
[1] 40.38204
```

The bootstrap distribution is symmetric and bell-shaped, so we can use the standard error method to calculate a 95% confidence interval.

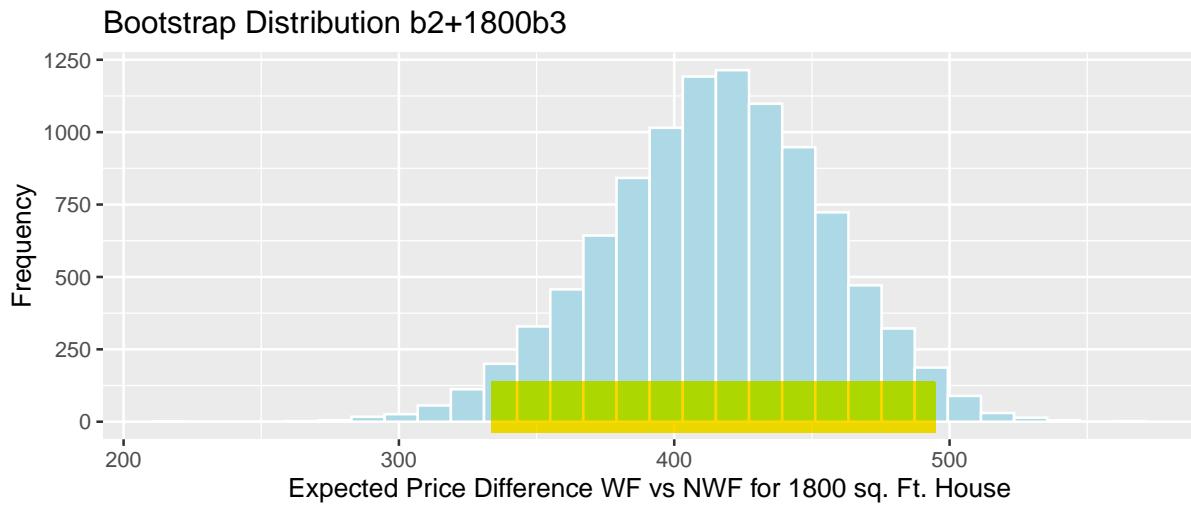
95% Confidence Interval:

```
c(Sample_Q3 - 2*SE_Q3, Sample_Q3 + 2*SE_Q3)
```

```
waterfrontYes      waterfrontYes
      333.4417        494.9698
```

The 95% confidence interval is shown by the gold bar on the graph of the bootstrap distribution below.

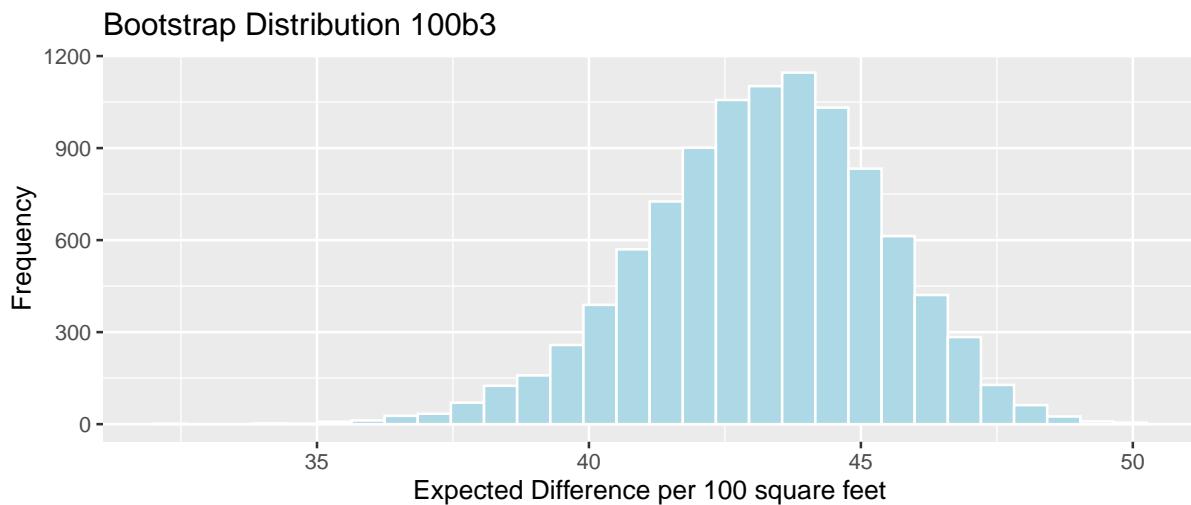
```
Houses_Bootstrap_Plot_Q3 +
  geom_segment(aes(x=Sample_Q3 - 2*SE_Q3,xend=Sample_Q3 + 2*SE_Q3, y=50, yend=50),
               color="gold", size=10, alpha=0.01)
```



We are 95% confident that the average price among all 1800 square feet waterfront houses in Seattle is between 333.441699 and 494.9698429 thousand dollars more than the average price among all non-waterfront houses of the same size.

Bootstrap Distribution for $100b_3$

```
Houses_Bootstrap_Plot_Q4 <- ggplot(data=Houses_Bootstrap_Results,
                                     aes(x=Bootstrap_Q4)) +
  geom_histogram(color="white", fill="lightblue") +
  xlab("Expected Difference per 100 square feet") + ylab("Frequency") +
  ggtitle( "Bootstrap Distribution 100b3")
Houses_Bootstrap_Plot_Q4
```



Bootstrap Standard Error: We'll calculate the bootstrap standard error of the slope $100b_3$. This is a measure of how much the slope varies between samples.

```
SE_Q4 <- sd(Houses_Bootstrap_Results$Bootstrap_Q4)  
SE_Q4
```

```
[1] 2.175838
```

The bootstrap distribution is symmetric and bell-shaped, so we can use the standard error method to calculate a 95% confidence interval.

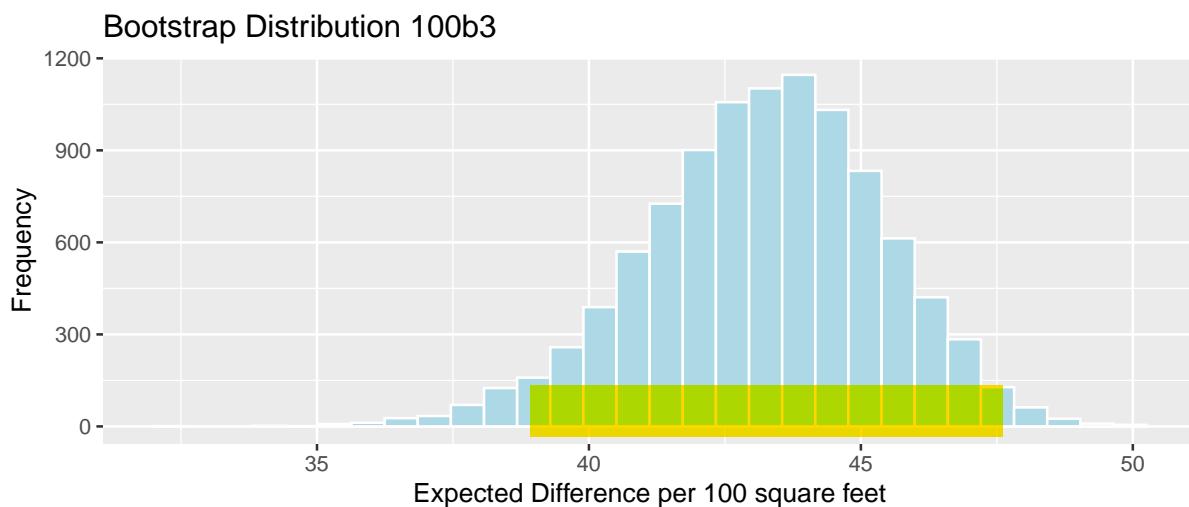
95% Confidence Interval:

```
c(Sample_Q4 - 2*SE_Q4, Sample_Q4 + 2*SE_Q4)
```

```
sqft_living:waterfrontYes sqft_living:waterfrontYes  
38.91503 47.61838
```

The 95% confidence interval is shown by the gold bar on the graph of the bootstrap distribution below.

```
Houses_Bootstrap_Plot_Q4 +  
  geom_segment(aes(x=Sample_Q4 - 2*SE_Q4, xend=Sample_Q4 + 2*SE_Q4, y=50, yend=50),  
                color="gold", size=10, alpha=0.01)
```



We are 95% confident that for each 100 square foot increase, the average price among all waterfront houses increases by between 38.9150334 and 47.6183836 thousand dollars more than the increase in average price among all non-waterfront.

3.4.9 Bootstrapping Cautions

While bootstrapping is a popular and robust procedure for calculating confidence intervals, it does have cautions and limitations. We should be sure to use the bootstrap procedure appropriate for our context. A standard-error bootstrap interval is appropriate when the sampling distribution for our statistic is roughly symmetric and bell-shaped. When this is not true, a percentile bootstrap interval can be used as long as there are no gaps or breaks in the bootstrap distribution. In situations where there are gaps and breaks in the bootstrap distribution, then the bootstrap distribution may not be a reasonable approximation of the sampling distribution we are interested in.

3.5 Hypothesis Testing

3.5.1 Mercury Levels in Florida Lakes

Recall the 2004 study by Lange, T., Royals, H. and Connor, L., which 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.

In our sample, we have data on 53 lakes, out of more than 30,000 lakes in the state of Florida.

We are 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 Orlando.



Figure 3.4: from Google Maps

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", "N", "N", "S", "N", "S"))
FloridaLakes <- FloridaLakes %>% rename(Mercury = AvgMercury)
print.data.frame(data.frame(FloridaLakes %>% select(Lake, Location, Mercury)), row.names = FALSE)
```

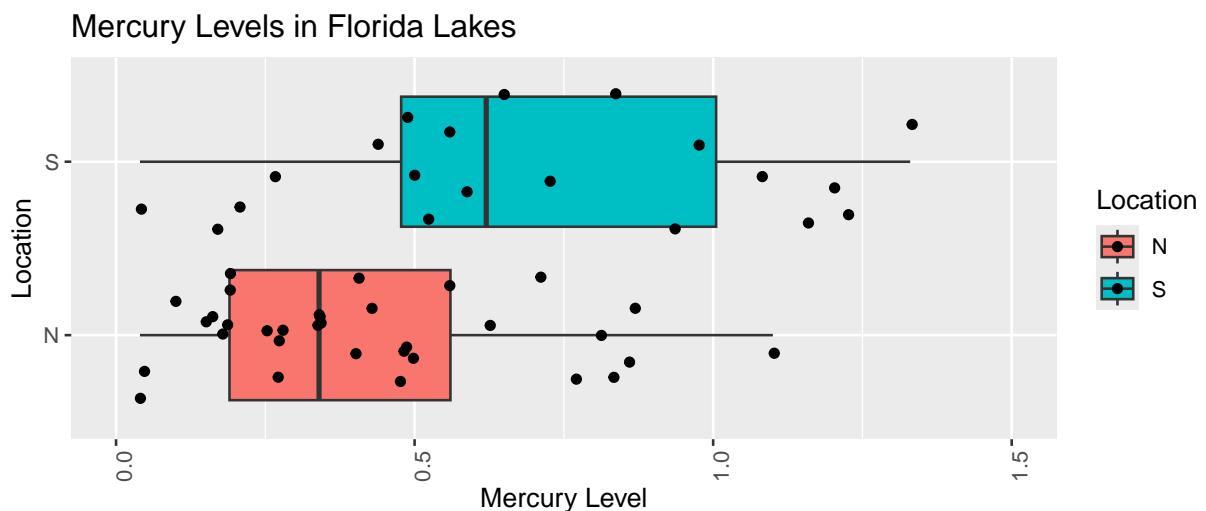
Lake	Location	Mercury
Alligator	S	1.23
Annie	S	1.33
Apopka	N	0.04
Blue Cypress	S	0.44
Brick	S	1.20
Bryant	N	0.27
Cherry	N	0.48
Crescent	N	0.19
Deer Point	N	0.83
Dias	N	0.81
Dorr	N	0.71
Down	S	0.50
Eaton	N	0.49
East Tohopekaliga	S	1.16
Farm-13	N	0.05
George	N	0.15
Griffin	N	0.19
Harney	N	0.77
Hart	S	1.08
Hatchineha	S	0.98
Iamonia	N	0.63
Istokpoga	S	0.56
Jackson	N	0.41
Josephine	S	0.73
Kingsley	N	0.34
Kissimmee	S	0.59
Lochloosa	N	0.34
Louisa	S	0.84
Miccasukee	N	0.50
Minneola	N	0.34
Monroe	N	0.28
Newmans	N	0.34
Ocean Pond	N	0.87
Ocheese Pond	N	0.56
Okeechobee	S	0.17
Orange	N	0.18
Panasoffkee	N	0.19
Parker	S	0.04
Placid	S	0.49
Puzzle	N	1.10
Rodman	N	0.16
Rousseau	N	0.10

Sampson	N	0.48
Shipp	S	0.21
Talquin	N	0.86
Tarpon	S	0.52
Tohopekaliga	S	0.65
Trafford	S	0.27
Trout	S	0.94
Tsala Apopka	N	0.40
Weir	N	0.43
Wildcat	N	0.25
Yale	N	0.27

We are interested in investigating whether average mercury levels are higher in either Northern Florida or Southern Florida than the other.

The boxplot and table below show the distribution of mercury levels among the 33 northern and 20 southern lakes in the sample.

```
LakesBP <- ggplot(data=FloridaLakes, aes(x=Location, y=Mercury, 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
```



```
LakesTable <- FloridaLakes %>% group_by(Location) %>% summarize(MeanHg=mean(Mercury), StDevHg=sd(Mercury))
kable(LakesTable)
```

Location	MeanHg	StDevHg	N
N	0.4245455	0.2696652	33
S	0.6965000	0.3838760	20

We see that on average mercury levels were higher among the southern lakes than the northern ones, a difference of $0.697 - 0.445 = 0.272$ ppm.

3.5.2 Model for Mercury Level

We can use a statistical model to estimate a lake's mercury level, using its location (N or S) as our explanatory variable.

The model equation is

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

- 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

Fitting the model in R, we obtain the estimates for b_0 and b_1 .

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

Call:

```
lm(formula = Mercury ~ Location, data = FloridaLakes)
```

Coefficients:

(Intercept)	LocationS
0.4245	0.2720

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

- $b_1 = 0.272 = 0.6965 - 0.4245$ 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.

3.5.3 Hypotheses and Key Question

Since the lakes we observed are only a sample of 53 lakes out of more than 30,000, we cannot assume the difference in mercury concentration for **all** Northern vs Southern Florida lakes is exactly 0.272. Instead, we need to determine whether a difference of this size in our sample is large enough to provide evidence of a difference in average mercury level between **all** Northern and Southern lakes in Florida.

One possible explanation for us getting the results we did in our sample is that there really is no difference in average mercury levels between all lakes in Northern and Southern Florida, and we just happened, by chance, to select more lakes with higher mercury concentrations in Southern Florida than in Northern Florida. A different possible explanation is that there really is a difference in average mercury level between lakes in Northern and Southern Florida.

In a statistical investigation, the **null hypothesis** is the one that says there is no difference between groups , or no relationship between variables in the larger population, and that any difference/relationship observed in our sample occurred merely by chance. The **alternative hypothesis** contradicts the null hypothesis, stating that there is a difference/relationship.

Stated formally, the hypotheses are:

Null Hypothesis: There is no difference in average mercury level between all lakes in Northern Florida and all lakes in Southern Florida.

Alternative Hypothesis: There is a difference in average mercury level between all lakes in Northern Florida and all lakes in Southern Florida.

A statistician's job is to determine whether the data provide strong enough evidence to rule out the null hypothesis.

The question we need to investigate is:

“How likely is it that we would have observed a difference in means (i.e. a value of b_1) as extreme as $0.6965 - 0.4245 = 0.272$ ppm, merely by chance, if there is really no relationship between location and mercury level?”

3.5.4 Permutation Test for Difference in Means

We can answer the key question using a procedure known as a **permutation test**. In a permutation test, we randomly permute (or shuffle) the values of our explanatory variable to simulate a situation where there is no relationship between our explanatory and response variable. We observe whether it is plausible to observe values of a statistic (in this case the difference in means) as extreme or more extreme than what we saw in the actual data.

We'll simulate situations where there is no relationship between location and mercury level, and see how often we observe a difference in means (b_1) as extreme as 0.272.

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. The statistic used to measure the size of the difference or relationship in the sample is called the **test statistic**.
3. Repeat steps 1 and 2 many (say 10,000) times, recording the test statistic (difference in means, b_1) each time.
4. Analyze the distribution of the test statistic (mean difference), simulated under the assumption that there is no relationship between location and mercury level. Look whether the value of the test statistic we observed in the sample (0.272) is consistent with values simulated under the assumption that the null hypothesis is true.

This simulation can be performed using this [Rossman-Chance App](#).

3.5.5 Five Permutations in R

We'll use R to perform permutation test.

First Permutation

Recall these groups were randomly assigned, so the only differences in averages are due to random chance.

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

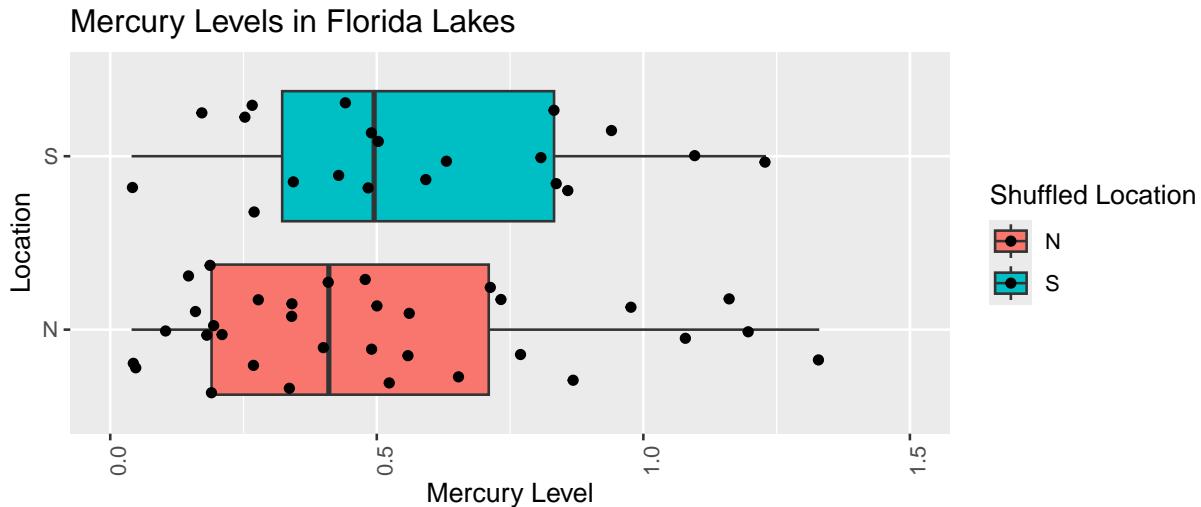
Shuffle1df <- data.frame(FloridaLakes$Lake, FloridaLakes$Location,
                         FloridaLakes$Mercury, ShuffledLakes$Location)
names(Shuffle1df) <- c("Lake", "Location", "Mercury", "Shuffled Location")
kable(head(Shuffle1df))
```

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

Notice that the locations of the lakes have now been mixed up and assigned randomly. So, any relationship between location and mercury level will have occurred merely by chance.

We create a boxplot and calculate the difference in mean mercury levels for the shuffled data.

```
LakesPerm <- ggplot(data=Shuffle1df, aes(x=`Shuffled Location`,
                                         y=Mercury, fill=`Shuffled Location`)) +
  geom_boxplot() + geom_jitter() + ggtitle("Mercury Levels in Florida Lakes") +
  xlab("Location") + ylab("Mercury Level") + theme(axis.text.x = element_text(angle = 90)) +
LakesPerm
```



```
LakesPermTable <- Shuffle1df %>% group_by(`Shuffled Location`) %>% summarize(MeanHg=mean(Mercury))
kable(LakesPermTable)
```

Shuffled Location	MeanHg	StDevHg	N
N	0.4978788	0.3536308	33
S	0.5755000	0.3220898	20

Notice that the sample means are not identical. We observe a difference of -0.07762121212121 just by chance associated with the assignment of the lakes to their random location groups.

This difference is considerably smaller than the difference of 0.272 that we saw in the actual data, suggesting that perhaps a difference as big as 0.272 would not be likely to occur by chance. Before we can be sure of this, however, we should repeat our simulation many times to get a better sense for how big of a difference we might reasonable expect to occur just by chance.

Second Permutation

```

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

```

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

```

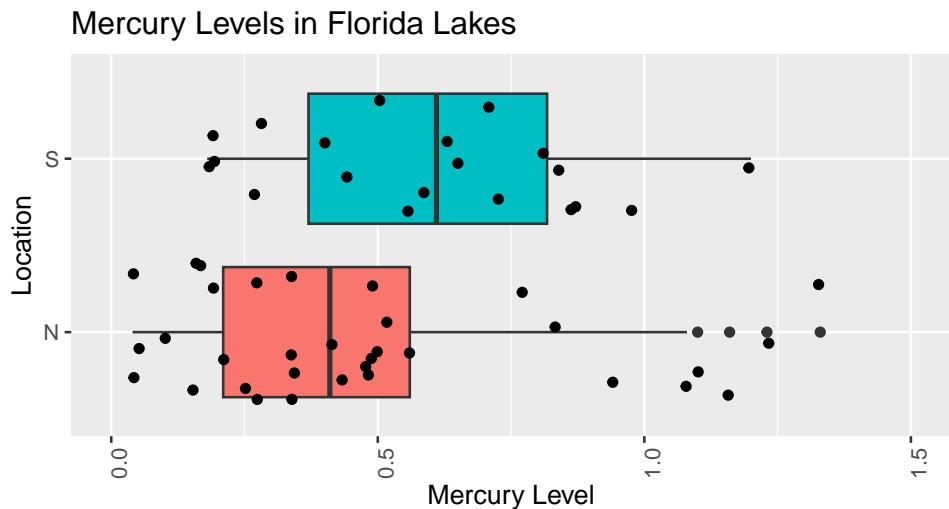
Shuffle1df <- data.frame(FloridaLakes$Lake, FloridaLakes$Location, FloridaLakes$Mercury, Shu
names(Shuffle1df) <- c("Lake", "Location", "Mercury", "Shuffled Location")

```

```

LakesPerm <- ggplot(data=Shuffle1df, aes(x=`Shuffled Location`, y=Mercury, fill=`Shuffled Loc
  geom_boxplot() +  geom_jitter() + ggtitle("Mercury Levels in Florida Lakes") +
  xlab("Location") + ylab("Mercury Level") + theme(axis.text.x = element_text(angle = 90)) +
LakesPerm

```



```

LakesPermTable <- Shuffle1df %>% group_by(`Shuffled Location`) %>% summarize(MeanHg=mean(Mer
kable(LakesPermTable)

```

Shuffled Location	MeanHg	StDevHg	N
N	0.4866667	0.3676332	33
S	0.5940000	0.2883236	20

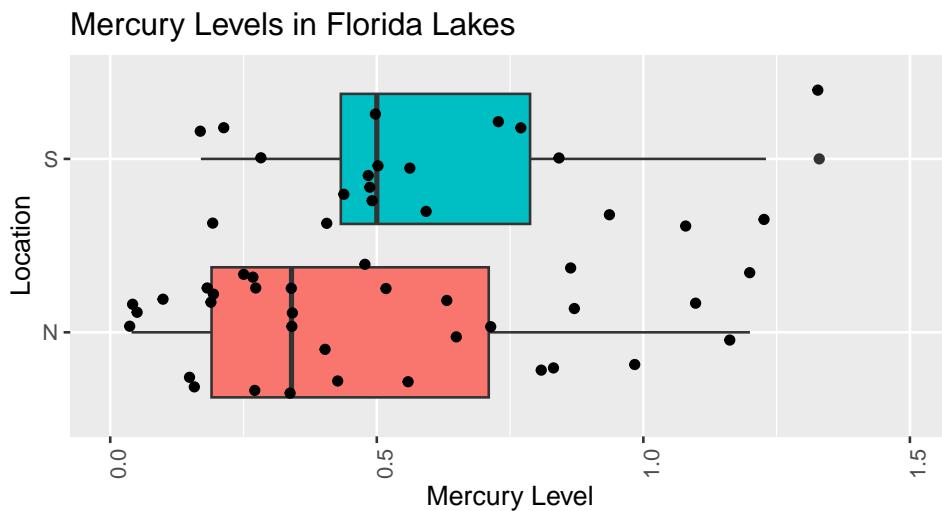
Third Permutation

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

Lake	Location	Mercury	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

```
Shuffle1df <- data.frame(FloridaLakes$Lake, FloridaLakes$Location, FloridaLakes$Mercury, Shu
names(Shuffle1df) <- c("Lake", "Location", "Mercury", "Shuffled Location")
```

```
LakesPerm <- ggplot(data=Shuffle1df, aes(x=`Shuffled Location`, y=Mercury, fill=`Shuffled Loc
  geom_boxplot() +  geom_jitter() + ggttitle("Mercury Levels in Florida Lakes") +
  xlab("Location") + ylab("Mercury Level") + theme(axis.text.x = element_text(angle = 90)) +
LakesPerm
```



Shuffled Location

- N
- S

```
LakesPermTable <- Shuffle1df %>% group_by(`Shuffled Location`) %>% summarize(MeanHg=mean(Mercury))
kable(LakesPermTable)
```

Shuffled Location	MeanHg	StDevHg	N
N	0.4760606	0.3406514	33
S	0.6115000	0.3329339	20

Fourth Permutation

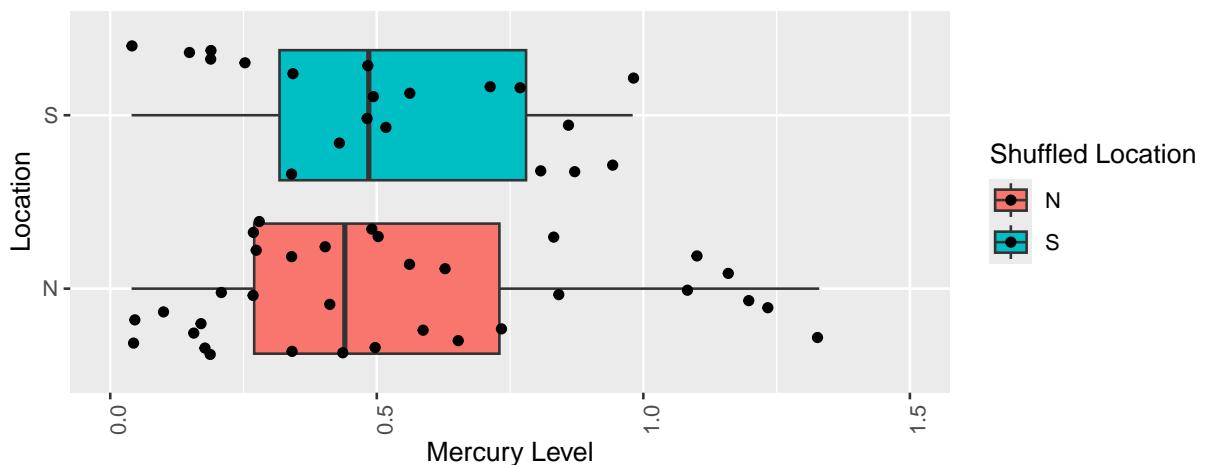
```
ShuffledLakes <- FloridaLakes      ## create copy of dataset
ShuffledLakes$Location <- ShuffledLakes$Location[sample(1:nrow(ShuffledLakes))]
kable(head(Shuffle1df))
```

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

```
Shuffle1df <- data.frame(FloridaLakes$Lake, FloridaLakes$Location, FloridaLakes$Mercury, ShuffledLakes$Shuffled Location)
names(Shuffle1df) <- c("Lake", "Location", "Mercury", "Shuffled Location")
```

```
LakesPerm <- ggplot(data=Shuffle1df, aes(x=`Shuffled Location`, y=Mercury, fill=`Shuffled Location`)) +
  geom_boxplot() + geom_jitter() + ggtitle("Mercury Levels in Florida Lakes") +
  xlab("Location") + ylab("Mercury Level") + theme(axis.text.x = element_text(angle = 90)) +
  LakesPerm
```

Mercury Levels in Florida Lakes



```
LakesPermTable <- Shuffle1df %>% group_by(`Shuffled Location`) %>% summarize(MeanHg=mean(Mercury), N=n(), StDevHg=sd(Mercury))
kable(LakesPermTable)
```

Shuffled Location	MeanHg	StDevHg	N
N	0.5315152	0.3750343	33
S	0.5200000	0.2851961	20

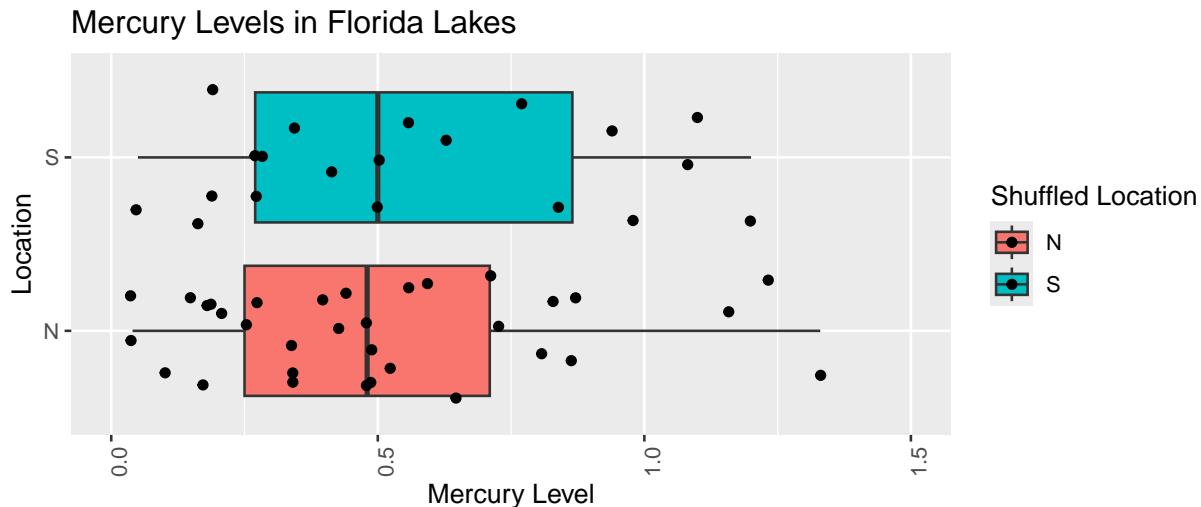
Fifth Permutation

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

Lake	Location	Mercury	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

```
Shuffle1df <- data.frame(FloridaLakes$Lake, FloridaLakes$Location, FloridaLakes$Mercury, ShuffledLakes$Shuffled Location)
names(Shuffle1df) <- c("Lake", "Location", "Mercury", "Shuffled Location")
```

```
LakesPerm <- ggplot(data=Shuffle1df, aes(x=`Shuffled Location`, y=Mercury, fill=`Shuffled Location`)) +  
  geom_boxplot() +  
  geom_jitter() +  
  ggttitle("Mercury Levels in Florida Lakes") +  
  xlab("Location") +  
  ylab("Mercury Level") +  
  theme(axis.text.x = element_text(angle = 90)) +  
  LakesPerm
```



```
LakesPermTable <- Shuffle1df %>% group_by(`Shuffled Location`) %>% summarize(MeanHg=mean(Mercury),  
kable(LakesPermTable))
```

Shuffled Location	MeanHg	StDevHg	N
N	0.5054545	0.3339357	33
S	0.5630000	0.3582281	20

3.5.6 R Code for Permutation Test

We'll write a `for` loop to perform 10,000 permutations and record the value of b_1 (the difference in sample means) for each simulation.

```
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, Mercury ~ 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

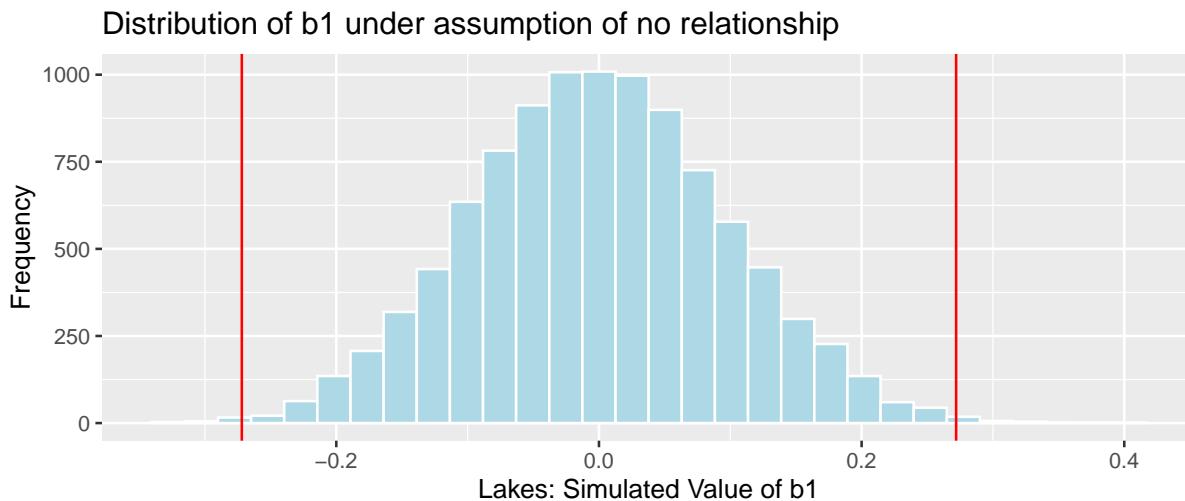
```

The histogram shows the distribution of differences in the group means observed in our simulation. The red lines indicate the difference we actually observed in the data (0.272), as well as an equally large difference in the opposite direction (-0.272).

```

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

```



The red lines are quite extreme, relative to the simulated values shown in the histogram. Based on the simulation, it is rare to obtain a difference as extreme as the 0.272 value we saw in the actual data, by chance when there is actually no difference in average mercury levels between Northern and Southern Florida lakes.

We calculate the precise number of simulations (out of 10,000) resulting in difference in means more extreme than 0.27195.

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

```
[1] 39
```

The proportion of simulations resulting in difference in means more extreme than 0.272 is:

```
sum(abs(b1Sim) > abs(b1))/10000
```

```
[1] 0.0039
```

We only observed a difference between the groups as extreme or more extreme than the 0.272 difference we saw in the sample in a proportion of 0.0039 of our simulations (less than 1%).

The probability of getting a difference in means as extreme or more extreme than 0.272 ppm by chance, when there is no relationship between location and mercury level is about 0.0039. In other words, it is very unlikely that we would have observed a result like we did by chance alone. Thus, we have strong evidence that there is a difference in average mercury level between lakes in Northern and Southern Florida. In this case, there is strong evidence that mercury level is higher in Southern Florida lakes than Northern Florida lakes.

Recall that in the previous chapter, we found that we could be 95% confident that the mean mercury level among all lakes in Southern Florida is between 0.08 and 0.46 higher than the mean mercury level among all lakes in Northern Florida.

3.5.7 p-values

The **p-value** represents the probability of getting a test statistic as extreme or more extreme than we did in our sample when the null hypothesis is true.

In this situation, the p-value represents the probability of observing a difference in sample means as extreme or more extreme than 0.272 if there is actually no difference in average mercury level among all lakes in Northern Florida, compared to Southern Florida.

In our study, the p-value was 0.0039, which is very low. This provides strong evidence against the null hypothesis that there is no difference in average mercury levels between all Northern and Southern Florida lakes.

A low p-value tells us that the difference in average Mercury levels that we saw in our sample is unlikely to have occurred by chance, providing evidence that there is indeed a difference in average Mercury levels between Northern and Southern lakes.

The p-value does not tell us anything about the size of the difference! If the difference is really small (say 0.001 ppm), perhaps there is no need to worry about it. It is possible to get a small p-value even when the true difference is very small (especially when our sample size is large). In addition to a p-value, we should consider whether a difference is big enough to be meaningful in a practical way, before making any policy decisions.

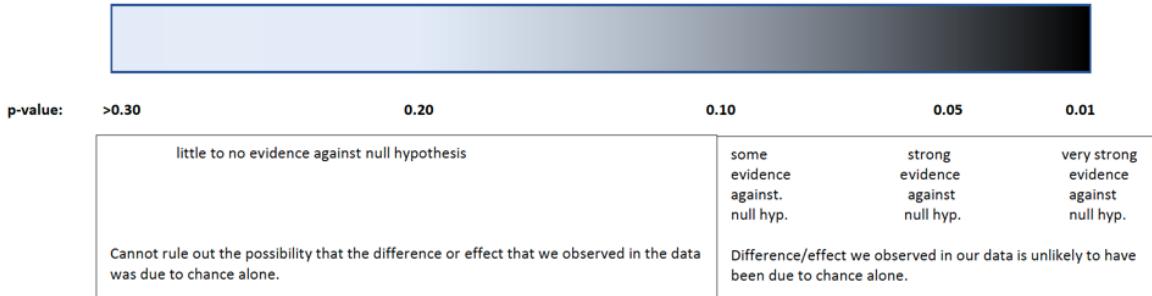
For now, we can use the difference in sample means of 0.272 ppm as an estimate of the size of the difference. Based on our limited knowledge of mercury levels, this does seem big enough to merit further investigation, and possible action.

At this point, a reasonable question is “how small must a p-value be in order to provide evidence against the null hypothesis?” While it is sometimes common to establish strict cutoffs for what counts as a small p-value (such as < 0.05), the American Statistical Association does not recommend this. In reality, a p-value of 0.04 is practically no different than a p-value of 0.06. Rather than using strict cutoffs for what counts as small, it is better to interpreting p-values on a sliding scale, as illustrated in the diagram below. A p-value of 0.10 or less provides at least some evidence against a null hypothesis, and the smaller the p-value is, the stronger the evidence gets.

```
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:



3.6 More Hypothesis Test Examples

3.6.1 Other Test Statistics

The permutation test procedure can be used to test hypotheses involving lots of different test statistics, in addition to testing for a difference in means, as we saw seen in the previous section. For example we could test whether there is evidence of:

1. a difference in the median mercury level between lakes in Northern Florida, compared to southern Florida
2. a difference in the amount of variability in mercury levels between lakes in Northern Florida, compared to southern Florida
3. a difference in the proportion of lakes whose mercury level exceeds 1 ppm between lakes in Northern Florida, compared to southern Florida
4. a difference in mean price between King County houses in very good, good, and average or below conditions
5. a relationship between mercury level and pH level among all Florida lakes

For each of these investigations, the null hypothesis will be that there is no difference or relationship among all lakes (that is, whatever difference or relationship occurred in the sample occurred just by random chance). We'll need to find a test statistic that measures the quantity we're interested in (for example, difference in means). Then, we use the permutation procedure to simulate a scenario where our null hypothesis is true, and see if test statistic we saw in our data is consistent with the ones we simulate under the null hypothesis.

3.6.2 General Permutation Test Procedure

Procedure:

1. Randomly shuffle the values or categories of the explanatory variable, so that any relationship between the explanatory and response variable occurs just by chance.
2. Calculate the test statistic on the shuffled data.
3. Repeat steps 1 and 2 many (say 10,000) times, recording the test statistic each time.

- Analyze the distribution of the test statistic, simulated under the assumption that the null hypothesis is true. Look whether the value of the test statistic we observed in the sample is consistent with values simulated under the assumption that the null hypothesis is true. (We might calculate a p-value, which represents the proportion of simulations in which we observed a test statistic as extreme or more extreme than the one we saw in our actual sample.)

Next, we'll apply these steps to questions 2, 4, and 5 from the previous subsection.

3.6.3 Difference in Standard Deviation

We'll test whether there is evidence of a difference in variability between lakes in Northern Florida, compared to Southern Florida. Since standard deviation is a measure of variability, we'll use the difference in standard deviation in Northern vs Southern lakes as our test statistic.

Recall that the standard deviation among the 53 Northern Florida Lakes in our sample was 0.270 ppm, which is lower than the 0.384 ppm in Southern Florida.

```
kable(LakesTable)
```

Location	MeanHg	StDevHg	N
N	0.4245455	0.2696652	33
S	0.6965000	0.3838760	20

The test statistic we observe in our sample is $0.2696 - 0.3839 = -0.1142$ ppm.

We need to determine whether a difference this large could have plausibly occurred in our sample, just by chance, if there is really no difference in standard deviation among all lakes in Northern Florida, compared to Southern Florida.

Null Hypothesis: There is no difference in standard deviation of mercury levels between all lakes in Northern Florida and all lakes in Southern Florida.

Alternative Hypothesis: There is a difference in standard deviation of mercury levels between all lakes in Northern Florida and all lakes in Southern Florida.

We'll apply the general hypothesis testing procedure, using standard deviation as our test statistic.

Procedure:

- Randomly shuffle the locations of the lakes, so that any relationship between the location and mercury level occurs just by chance.

2. Calculate the difference in standard deviation between lakes in the two samples of the shuffled data.
3. Repeat steps 1 and 2 many (say 10,000) times, recording the difference in standard deviations each time.
4. Analyze the distribution of difference in standard deviations, simulated under the assumption that there is no difference in standard deviations between North and South. Look whether the value of the test statistic we observed in the sample is consistent with values simulated under the assumption that there is no difference in standard deviations.

R Code for Permutation Test

We'll write a `for` loop to perform 10,000 permutations and record the value of b_1 (the difference in sample means) for each simulation.

```
SDTab <- FloridaLakes %>% group_by(Location) %>% summarize(SD=sd(Mercury))
DiffSD <- SDTab$SD[2] - SDTab$SD[1]

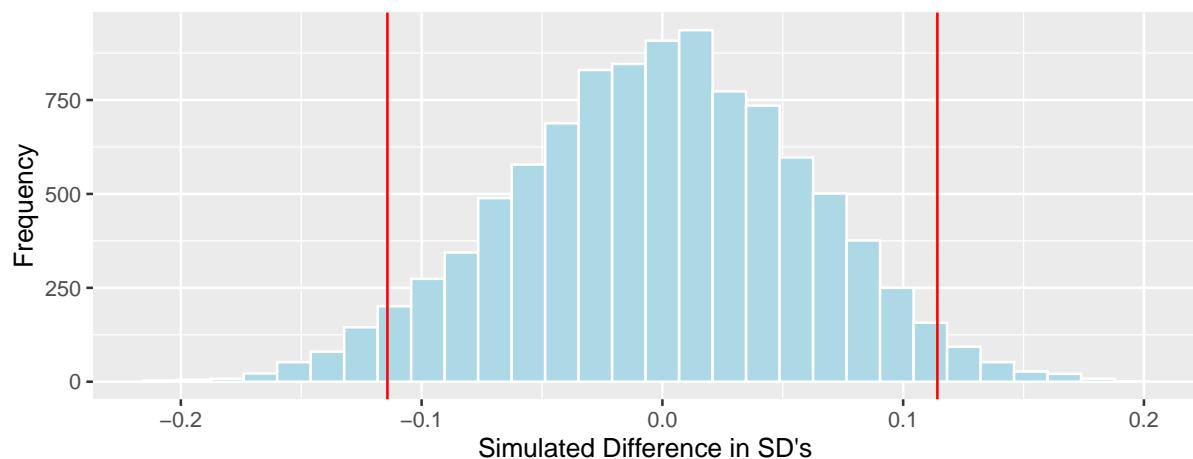
## perform simulation
DiffSim <- rep(NA, 10000)      ## vector to hold results
ShuffledLakes <- FloridaLakes    ## create copy of dataset
for (i in 1:10000){
  #randomly shuffle locations
  ShuffledLakes$Location <- ShuffledLakes$Location[sample(1:nrow(ShuffledLakes))]
  SDTabSim <- ShuffledLakes %>% group_by(Location) %>% summarize(SD=sd(Mercury))
  DiffSim[i] <- SDTabSim$SD[2] - SDTabSim$SD[1] #record difference in SD for simulated data
}
NSLakes_SDSimResults <- data.frame(DiffSim) #save results in dataframe
```

The distribution of the simulated differences in standard deviation is shown below. Recall that these were simulated assuming that the null hypothesis, that there is no difference in standard deviation of mercury levels among all lakes in Northern Florida, compared to Southern Florida is true.

The red lines represent differences as extreme as -0.1142 that we saw in our sample.

```
NSLakes_SDSimResultsPlot <- ggplot(data=NSLakes_SDSimResults, aes(x=DiffSim)) +
  geom_histogram(fill="lightblue", color="white") +
  geom_vline(xintercept=c(DiffSD, -1*DiffSD), color="red") +
  xlab("Simulated Difference in SD's") + ylab("Frequency") +
  ggtitle("Distribution of Difference in SD under assumption of no relationship")
NSLakes_SDSimResultsPlot
```

Distribution of Difference in SD under assumption of no relationship



We calculate the number of simulations (out of 10,000) resulting in standard deviations greater than the 0.1142.

```
sum(abs(DiffSim) > abs(DiffSD))
```

```
[1] 614
```

p-value: Proportion of simulations (out of 10,000) resulting in difference in standard deviations greater than the 0.1142.

```
mean(abs(DiffSim) > abs(DiffSD))
```

```
[1] 0.0614
```

This p-value represents the probability of observing a difference in sample standard deviations as extreme as 0.1142 in samples of size 33 and 20 by chance, if in fact, the standard deviation in mercury concentration levels is the same for lakes in Northern Florida as in Southern Florida.

Since the p-value is small, it is unlikely that we would observe a difference in standard deviations as extreme as 0.1142 by chance. There is evidence that lakes in Southern Florida exhibit more variability in mercury levels than lakes in Northern Florida (though the evidence is not as strong as it was when we were testing for a difference in means).

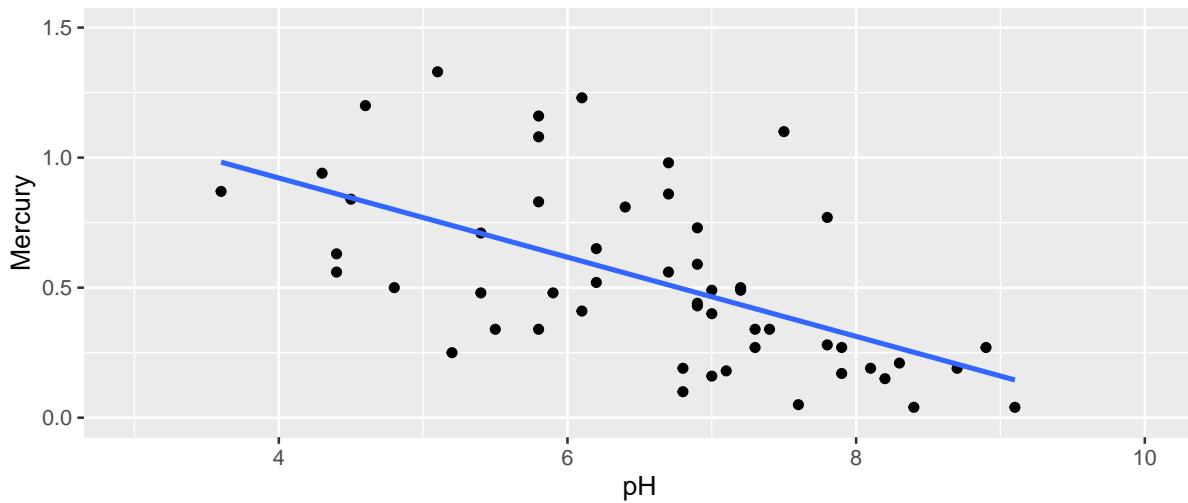
Note that we have avoided the fallacy of using 0.05 as a strict cutoff for rejecting the null hypothesis.

Although the difference in standard deviations is statistically discernible, it is hard to say whether it is practically meaningful. Without knowing a lot about mercury levels, and their impact on the ecosystem, it's harder to tell whether an estimated difference in standard deviations of 0.11 ppm is meaningful or not. It would be good to consult a biologist before making any decisions based on these results.

3.6.4 Slope of Regression Line

In addition to the mercury levels of the Florida lakes, we have data on the pH level of each lake. pH level measures the acidity of a lake, ranging from 0 to 14, with 7 being neutral, and lower levels indicating more acidity. We plot the pH level against the mercury level in our sample of 53 lakes.

```
ggplot(data=FloridaLakes, aes(y=Mercury, x=pH)) +
  geom_point() + stat_smooth(method="lm", se=FALSE) +
  xlim(c(3, 10)) + ylim(c(0,1.5))
```



The regression equation is

$$\widehat{\text{Mercury}} = b_0 + b_1 \times \text{pH}$$

Regression estimates b_0 and b_1 are shown below.

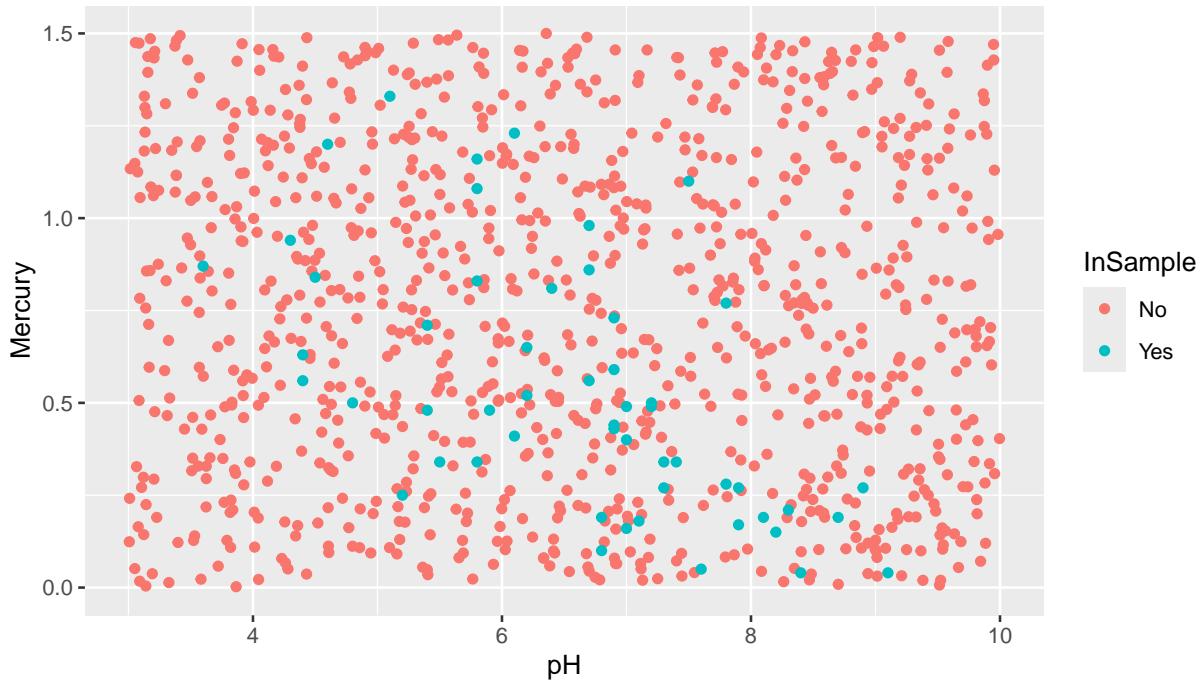
```
Lakes_M_pH <- lm(data=FloridaLakes, Mercury~pH)
Lakes_M_pH
```

```
Call:  
lm(formula = Mercury ~ pH, data = FloridaLakes)
```

```
Coefficients:  
(Intercept)          pH  
      1.5309        -0.1523
```

We can use the slope of the regression line b_1 to measure the strength relationship between Mercury and pH. Based on our sample, each one-unit increase in pH, mercury level is expected to decrease by 0.15 ppm.

If there was really no relationship, then the slope among all lakes would be 0. But, of course, we would not expect the slope in our sample to exactly match the slope for all lakes. Our question of interest is whether it is plausible that we could have randomly selected a sample resulting in a slope as extreme as 0.15 by chance, when there is actually no relationship between mercury and pH levels, among all lakes. In other words, could we plausible have drawn the sample of 53 lakes shown in blue from a population like the one in red, shown below?



Key Question:

- How likely is it that we would have observed a slope (i.e. a value of b_1) as extreme as 0.15 by chance, if there is really no relationship between mercury level and pH?

Null Hypothesis: Among all Florida lakes, there is no relationship between mercury level and pH.

Alternative Hypothesis: Among all Florida lakes, there is a relationship between mercury level and pH.

Procedure:

1. Randomly shuffle the pH values, so that any relationship between acceleration mercury and pH 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 mercury and pH. Look whether the actual slope we observed is consistent with the simulation results.

We'll illustrate the first three permutations.

First Permutation

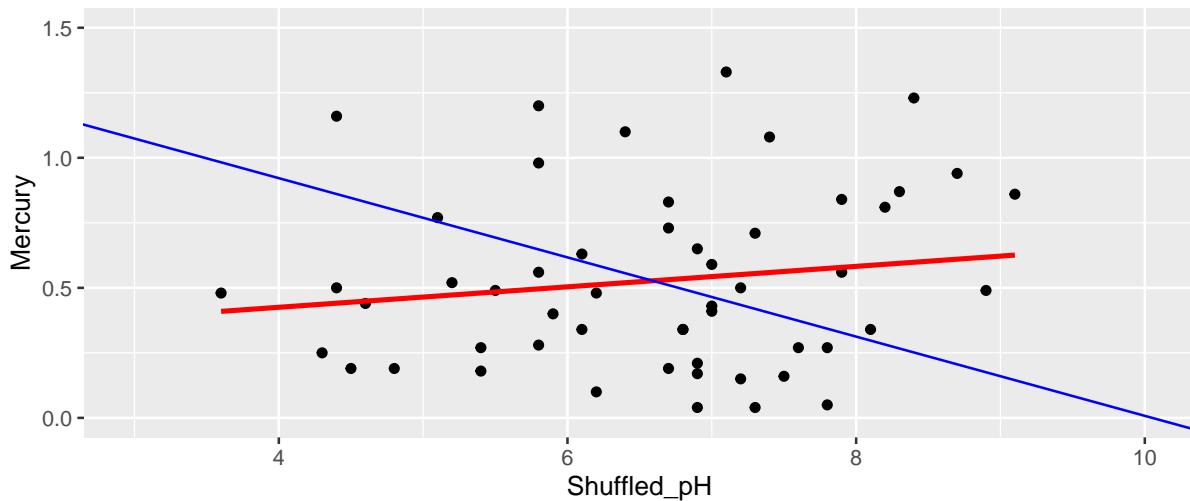
```
ShuffledLakes <- FloridaLakes      ## create copy of dataset
ShuffledLakes$pH <- ShuffledLakes$pH[sample(1:nrow(ShuffledLakes))]
```

```
Shuffle1df <- data.frame(ShuffledLakes$Lake, FloridaLakes$Mercury, FloridaLakes$pH, ShuffledLakes$pH)
names(Shuffle1df) <- c("Lake", "Mercury", "pH", "Shuffled_pH")
kable(head(Shuffle1df))
```

Lake	Mercury	pH	Shuffled_pH
Alligator	1.23	6.1	8.4
Annie	1.33	5.1	7.1
Apopka	0.04	9.1	6.9
Blue Cypress	0.44	6.9	4.6
Brick	1.20	4.6	5.8
Bryant	0.27	7.3	7.8

The red line indicates the slope of the regression line fit to the shuffled data. The blue line indicates the regression line for the actual lakes in the sample, which has a slope of -0.15.

```
ggplot(data=Shuffle1df, aes(x=Shuffled_pH, y=Mercury)) +
  geom_point() + stat_smooth(method="lm", se=FALSE, color="red") +
  xlim(c(3, 10)) + ylim(c(0,1.5)) +
  geom_abline(slope=-0.1523, intercept=1.5309, color="blue")
```



Slope of regression line from permuted data:

```
M_Lakes_Shuffle <- lm(data=Shuffle1df, Mercury~Shuffled_pH)
summary(M_Lakes_Shuffle)$coef[2]
```

```
[1] 0.03934635
```

Second Permutation

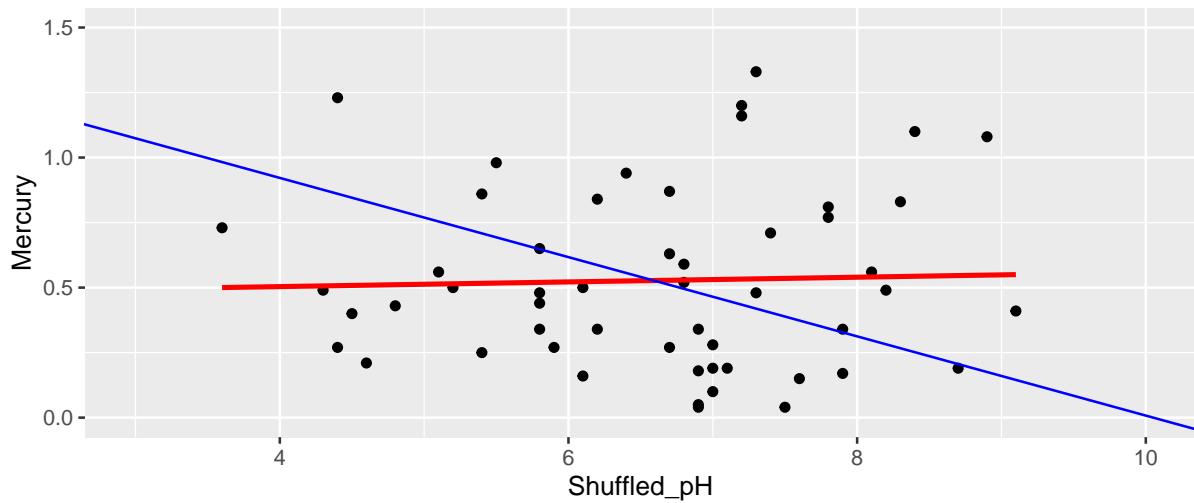
```
ShuffledLakes <- FloridaLakes      ## create copy of dataset
ShuffledLakes$pH <- ShuffledLakes$pH[sample(1:nrow(ShuffledLakes))]

Shuffle2df <- data.frame(ShuffledLakes$Lake, FloridaLakes$Mercury, FloridaLakes$pH, ShuffledLakes$Shuffled_pH)
names(Shuffle2df) <- c("Lake", "Mercury", "pH", "Shuffled_pH")
kable(head(Shuffle2df))
```

Lake	Mercury	pH	Shuffled_pH
Alligator	1.23	6.1	4.4
Annie	1.33	5.1	7.3
Apopka	0.04	9.1	7.5

Lake	Mercury	pH	Shuffled_pH
Blue Cypress	0.44	6.9	5.8
Brick	1.20	4.6	7.2
Bryant	0.27	7.3	6.7

```
ggplot(data=Shuffle2df, aes(x=Shuffled_pH, y=Mercury)) +
  geom_point() + stat_smooth(method="lm", se=FALSE, color="red") +
  xlim(c(3, 10)) + ylim(c(0,1.5)) +
  geom_abline(slope=-0.1523, intercept=1.5309, color="blue")
```



Slope of regression line from permuted data:

```
M_Lakes_Shuffle <- lm(data=Shuffle2df, Mercury~Shuffled_pH)
summary(M_Lakes_Shuffle)$coef[2]
```

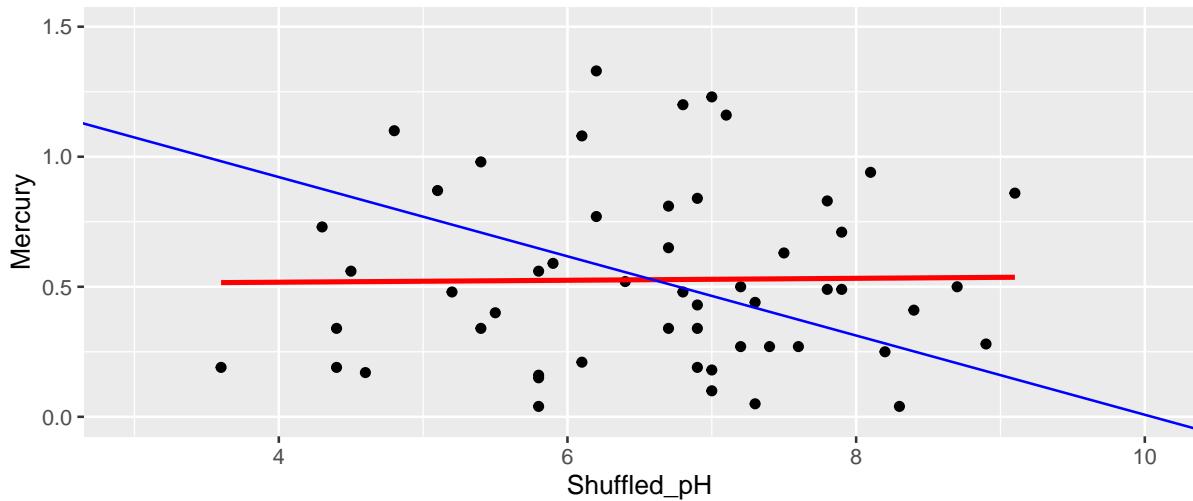
```
[1] 0.009030783
```

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

```
Shuffle3df <- data.frame(ShuffledLakes$Lake, FloridaLakes$Mercury, FloridaLakes$pH, ShuffledLakes$Shuffled_pH)
names(Shuffle3df) <- c("Lake", "Mercury", "pH", "Shuffled_pH")
kable(head(Shuffle3df))
```

Lake	Mercury	pH	Shuffled_pH
Alligator	1.23	6.1	7.0
Annie	1.33	5.1	6.2
Apopka	0.04	9.1	8.3
Blue Cypress	0.44	6.9	7.3
Brick	1.20	4.6	6.8
Bryant	0.27	7.3	7.2

```
ggplot(data=Shuffle3df, aes(x=Shuffled_pH, y=Mercury)) +
  geom_point() + stat_smooth(method="lm", se=FALSE, color="red") +
  xlim(c(3, 10)) + ylim(c(0,1.5)) +
  geom_abline(slope=-0.1523, intercept=1.5309, color="blue")
```



Slope of regression line from permuted data:

```
M_Lakes_Shuffle <- lm(data=Shuffle3df, Mercury~Shuffled_pH)
summary(M_Lakes_Shuffle)$coef[2]
```

```
[1] 0.003748437
```

None of our three simulations resulted in a slope near as extreme as the -0.15 that we saw in the actual data. This seems to suggest that it is unlikely that we would have observed a slope as extreme as -0.15 if there is actually no relationship between mercury and pH among all lakes.

That said, we should repeat the simulation many more times to see whether getting a slope as extreme as -0.15 is plausible.

```

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

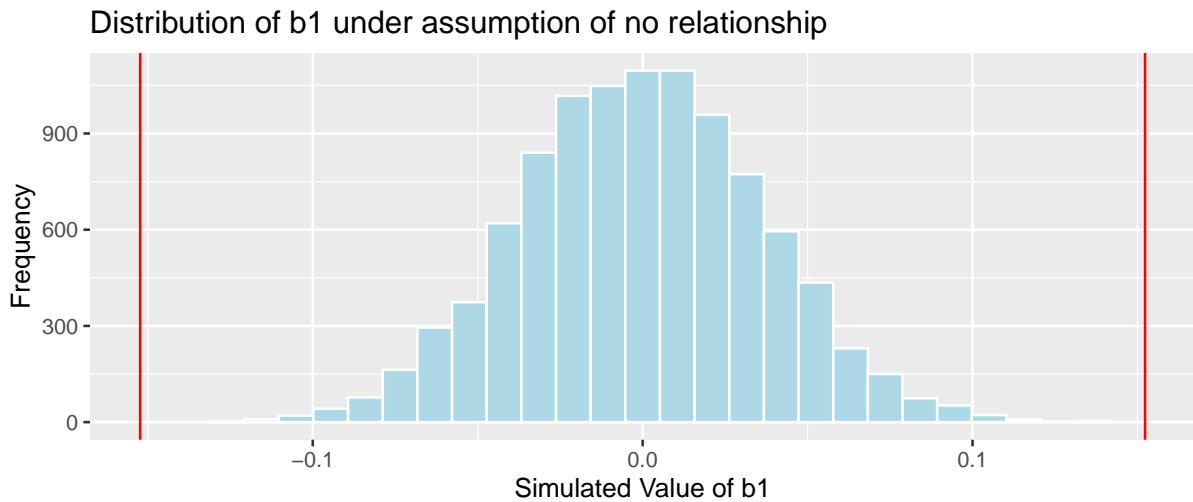
## perform simulation
b1Sim <- rep(NA, 10000)           ## vector to hold results
ShuffledLakes <- FloridaLakes    ## create copy of dataset
for (i in 1:10000){
  #randomly shuffle acceleration times
  ShuffledLakes$pH <- ShuffledLakes$pH[sample(1:nrow(ShuffledLakes))]
  ShuffledLakes_M<- lm(data=ShuffledLakes, Mercury ~ pH)  #fit model to shuffled data
  b1Sim[i] <- ShuffledLakes_M$coef[2] ## record b1 from shuffled model
}
Lakes_pHSimulationResults <- data.frame(b1Sim) #save results in dataframe

```

```

b1 <- Lakes_M_pH$coef[2] ## record value of b1 from actual data
Lakes_pHSimulationResultsPlot <- ggplot(data=Lakes_pHSimulationResults, 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")
Lakes_pHSimulationResultsPlot

```



p-value: Proportion of simulations resulting in value of b_1 more extreme than -0.15

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

```
[1] 0
```

The p-value represents the probability of observing a slope as extreme or more extreme than -0.15 by chance when there is actually no relationship between mercury level and pH.

It is extremely unlikely that we would observe a value of b_1 as extreme as -0.15 by chance, if there is really no relationship between mercury level and pH. In fact, this never happened in any of our 10,000 simulations!

There is very strong evidence of a relationship mercury level and pH.

A low p-value tells us only that there is evidence of a relationship, not that it is practically meaningful. We have seen that for each one-unit increase in pH, mercury level is expected to decrease by 0.15 ppm on average, which seems like a pretty meaningful decrease, especially considering that mercury levels typically stay between 0 and 1.

We used the slope as our test statistic to measure the evidence of the relationship between the explanatory and response variables. In fact, we could have also used the correlation coefficient r as our test statistic, and we would have gotten the same p-value. Either slope or correlation may be used for a hypothesis test involving two quantitative variables, but we will use slope in this class.

3.6.5 F-Statistic

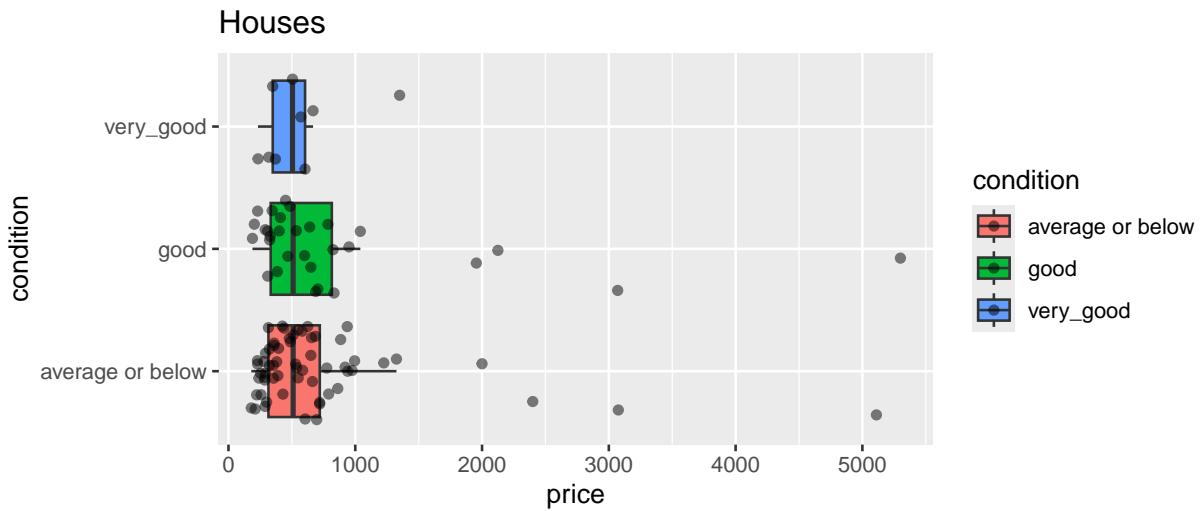
Recall when we examined the prices of houses in King County, WA, whose conditions were rated as either very good, good, or average or below. Suppose we want to test the hypotheses:

Null Hypothesis: There is no difference in average prices between houses of the three different conditions, among all houses in King County, WA.

Alternative Hypothesis: There is a difference in average prices between houses of the three different conditions, among all houses in King County, WA.

Comparative boxplots are shown below.

```
ggplot(data=Houses, aes(x=condition, y=price, fill=condition)) +  
  geom_boxplot(outlier.shape = NA) + geom_jitter(alpha=0.5) + coord_flip() + ggtitle("Houses")
```



```
Cond_Tab <- Houses %>% group_by(condition) %>% summarize(Mean_Price = mean(price),
  SD_Price= sd (price),
  N= n())
kable(Cond_Tab)
```

condition	Mean_Price	SD_Price	N
average or below	700.6349	768.1179	61
good	861.0000	1048.9521	30
very_good	551.8361	332.8597	9

We notice differences in price. Surprisingly, houses in good condition cost more than 300 thousand dollars more than those in very good condition, on average.

If we were only comparing two groups, we could use the difference in average price between them as a test statistic. But since we're comparing three, we need a statistic that can measure the size of differences between all three groups. An F-statistic can do this, so we'll use the F-statistic as our test statistic here.

We calculated the F-statistic in Chapter 2.

```
M_House_Cond <- lm(data=Houses, price~condition)
M0_House <- lm(data=Houses, price~1)
anova(M_House_Cond, M0_House)$F[2]
```

[1] 0.6046888

Our question of interest is “How likely is it to observe an F-statistic as extreme or more extreme than 0.605 if there is actually no difference in average price between houses of the three conditions?”

We'll use a permutation-based hypothesis test to investigate this question.

Procedure:

1. Randomly shuffle the conditions of the houses, so that any relationship between condition and price is due only to chance.
2. Using the shuffled data, calculate an F-statistic for a predicting price, comparing a full model that uses condition as an explanatory variable, to a reduced model with no explanatory variables.
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 condition and price. Look whether the actual F-statistic we observed is consistent with the simulation results.

We'll illustrate the first three permutations.

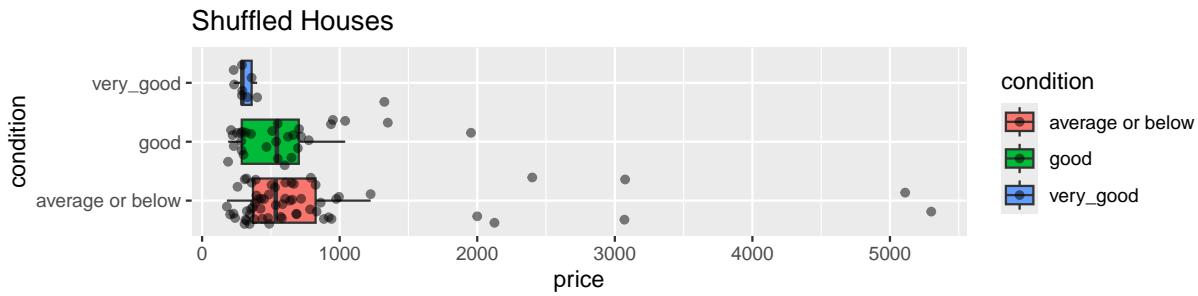
First Permutation

```
ShuffledHouses <- Houses      ## create copy of dataset
ShuffledHouses$condition <- ShuffledHouses$condition[sample(1:nrow(ShuffledHouses))]

Shuffle1df <- data.frame(Houses@Id, Houses$price, Houses$condition, ShuffledHouses$condition)
names(Shuffle1df) <- c("Id", "price", "condition", "Shuffled_Condition")
kable(head(Shuffle1df))
```

	Id	price	condition	Shuffled_Condition
1	1225.0	average or below	average or below	average or below
2	885.0	average or below	average or below	average or below
3	385.0	good	average or below	average or below
4	252.7	average or below	good	good
5	468.0	good	good	good
6	310.0	good	average or below	average or below

```
ggplot(data=ShuffledHouses, aes(x=condition, y=price, fill=condition)) +
  geom_boxplot(outlier.shape = NA) + geom_jitter(alpha=0.5) + coord_flip() + ggtitle("Shuffle")
```



We fit a model to predict price using condition, and compare it to one that predicts price without using condition, and calculate the F-statistic.

```
M1_Shuffled_Houses <- lm(data=ShuffledHouses, price~condition)
M0_Shuffled_Houses <- lm(data=ShuffledHouses, price~1)
anova(M1_Shuffled_Houses, M0_Shuffled_Houses)$F[2]
```

[1] 1.831639

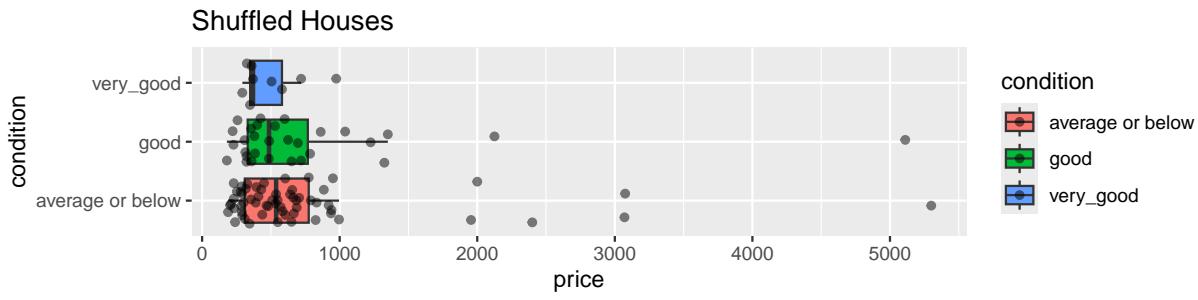
Second Permutation

```
ShuffledHouses <- Houses      ## create copy of dataset
ShuffledHouses$condition <- ShuffledHouses$condition[sample(1:nrow(ShuffledHouses))]

Shuffle2df <- data.frame(Houses@Id, Houses$price, Houses$condition, ShuffledHouses$condition)
names(Shuffle2df) <- c("Id", "price", "condition", "Shuffled_Condition")
kable(head(Shuffle2df))
```

Id	price	condition	Shuffled_Condition
1	1225.0	average or below	good
2	885.0	average or below	average or below
3	385.0	good	good
4	252.7	average or below	average or below
5	468.0	good	average or below
6	310.0	good	average or below

```
ggplot(data=ShuffledHouses, aes(x=condition, y=price, fill=condition)) +
  geom_boxplot(outlier.shape = NA) + geom_jitter(alpha=0.5) + coord_flip() + ggtitle("Shuffle")
```



We fit a model to predict price using condition, and compare it to one that predicts price without using condition, and calculate the F-statistic.

```
M1_Shuffled_Houses <- lm(data=ShuffledHouses, price~condition)
M0_Shuffled_Houses <- lm(data=ShuffledHouses, price~1)
anova(M1_Shuffled_Houses, M0_Shuffled_Houses)$F[2]
```

```
[1] 0.3999225
```

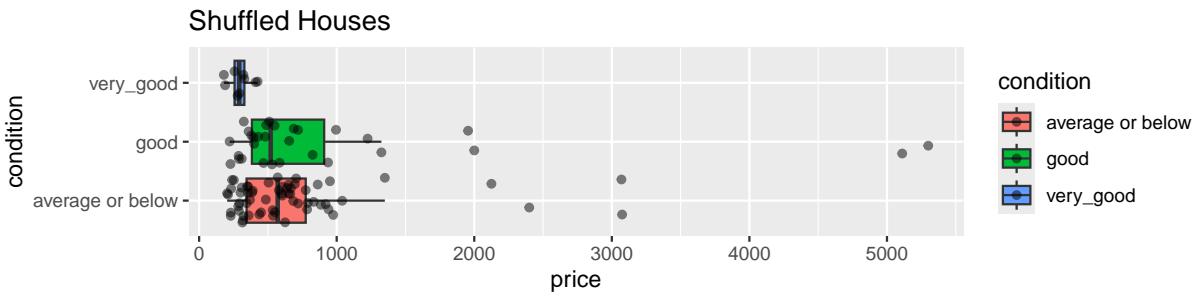
Third Permutation

```
ShuffledHouses <- Houses      ## create copy of dataset
ShuffledHouses$condition <- ShuffledHouses$condition[sample(1:nrow(ShuffledHouses))]

Shuffle3df <- data.frame(Houses@Id, Houses$price, Houses$condition, ShuffledHouses$condition)
names(Shuffle3df) <- c("Id", "price", "condition", "Shuffled_Condition")
kable(head(Shuffle1df))
```

	Id	price	condition	Shuffled_Condition
1	1225.0	average or below	average or below	average or below
2	885.0	average or below	average or below	average or below
3	385.0	good	average or below	average or below
4	252.7	average or below	good	good
5	468.0	good	good	good
6	310.0	good	average or below	average or below

```
ggplot(data=ShuffledHouses, aes(x=condition, y=price, fill=condition)) +
  geom_boxplot(outlier.shape = NA) + geom_jitter(alpha=0.5) + coord_flip() + ggtitle("Shuffle")
```



We fit a model to predict price using condition, and compare it to one that predicts price without using condition, and calculate the F-statistic.

```
M1_Shuffled_Houses <- lm(data=ShuffledHouses, price~condition)
M0_Shuffled_Houses <- lm(data=ShuffledHouses, price~1)
anova(M1_Shuffled_Houses, M0_Shuffled_Houses)$F[2]
```

```
[1] 2.55569
```

We'll simulate 10,000 permutations and record the F-statistic for each set of permuted data.

```
Fstat <- anova(M_House_Cond, M0_House)$F[2] ## record value of F-statistic from actual data

## perform simulation
FSim <- rep(NA, 10000) ## vector to hold results
ShuffledHouses <- Houses ## create copy of dataset
for (i in 1:10000){
  #randomly shuffle acceleration times
  ShuffledHouses$condition <- ShuffledHouses$condition[sample(1:nrow(ShuffledHouses))]
  ShuffledHouses_M1<- lm(data=ShuffledHouses, price ~ condition) #fit full model to shuffled data
  ShuffledHouses_M0<- lm(data=ShuffledHouses, price ~ 1) #fit reduced model to shuffled data
  FSim[i] <- anova(ShuffledHouses_M1, ShuffledHouses_M0)$F[2] ## record F from shuffled model
}
House_Cond_SimulationResults <- data.frame(FSim) #save results in dataframe
```

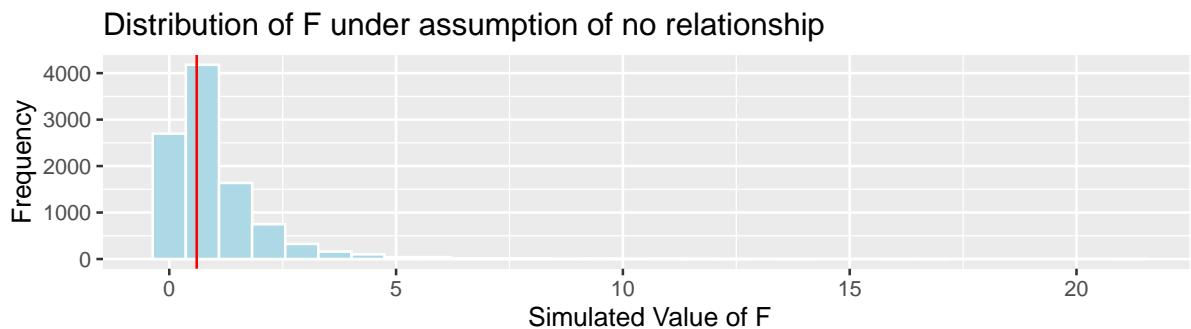
The distribution of the F-statistics is shown below. Recall that these are simulated under the assumption that there is no difference in average price between houses of the three different conditions, i.e. no relationship between price and condition.

The red line shows the location of the F-statistic we saw in our data (0.60). Since F-statistics cannot be negative, we don't need to worry about finding an F-statistic as extreme in the opposite direction.

```

House_Cond_SimulationResults_Plot <- ggplot(data=House_Cond_SimulationResults,
                                              aes(x=FSim)) +
  geom_histogram(fill="lightblue", color="white") + geom_vline(xintercept=c(Fstat), color="red")
  xlab("Simulated Value of F") + ylab("Frequency") + ggtitle("Distribution of F under assumption of no relationship")
House_Cond_SimulationResults_Plot

```



The F-statistic in our actual we observed does not appear to be very extreme.

p-value: Proportion of simulations resulting in value of F more extreme than 0.60.

```
mean(FSim > Fstat)
```

[1] 0.5548

The p-value represents the probability of observing an F-statistic as extreme as 0.60 by chance, in samples of size 61, 30, and 9, if in fact there is no relationship between price and size of car.

More than half of our simulations resulted in an F-statistic as extreme or more extreme than the one we saw in our actual data, even though the simulation was performed in a situation where there was no relationship between price and condition. Thus, it is very plausible that we would observe an F-statistic as extreme or more extreme than we saw in our data, even if there is no relationship between price and condition (or no difference in average price between the conditions), among all houses.

Since the p-value is large, we cannot reject the null hypothesis. We do not have evidence to say that average price differs between houses of the different condition types.

It is important to note that we are not saying that we believe the average price is the same for each condition. Recall that the average prices among the conditions in our sample differed by more than 300 thousand dollars! It's just that given the size of our samples, and the amount of variability in our data, we cannot rule out the possibility that this difference occurred purely by chance.

3.7 Responsible Hypothesis Testing

While hypothesis tests are a powerful tool in statistics, they are also one that has been widely misused, to the detriment of scientific research. The hard caused by these misuses caused the American Statistical Association to release a [2016 statement](#), intended to provide guidance and clarification to scientists who use hypothesis testing and p-values in their research.

The statement provides the following six principles for responsible use of hypothesis tests and p-values.

1. P-values can indicate how incompatible the data are with a specified statistical model.
2. P-values do not measure the probability that the studied hypothesis is true, or the probability that the data were produced by random chance alone.
3. Scientific conclusions and business or policy decisions should not be based only on whether a p-value passes a specific threshold.
4. Proper inference requires full reporting and transparency.
5. A p-value, or statistical significance, does not measure the size of an effect or the importance of a result.
6. By itself, a p-value does not provide a good measure of evidence regarding a model or hypothesis.

The statement provides important guidance for us to consider as we work with hypothesis testing in this class, as well as in future classes and potentially in our own research.

- 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.

What to conclude from p-values and what not to:

- A low p-value provides evidence against the null hypothesis. It suggests the test statistic we observed is inconsistent with the null hypothesis.
- A low p-value does **not** tell us that the difference or relationship we observed is meaningful in a practical sense. Researchers should look at the size of the difference or strength of the relationship in the sample before deciding whether it merits being acted upon.

- A high p-value means that the data could have plausibly been obtained when the null hypothesis is true. The test statistic we observed is consistent with what we would have expected to see when the null hypothesis is true, and thus we cannot rule out the null hypothesis.
- 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.

Just because our result is consistent with the null hypothesis does not mean that we should believe that null hypothesis is true. Lack of evidence against a claim does not necessarily mean that the claim is true.

In this scenario, we got a small p-value, but we should also be aware of what we should conclude if the p-value is large. Remember that the p-value only measures the strength of evidence *against* the null hypothesis. A large p-value means we lack evidence against the null hypothesis. This does not mean, however, that we have evidence *supporting* null hypothesis.

A hypothesis test can be thought of as being analogous to a courtroom trial, where the null hypothesis is that the defendant did not commit the crime. Suppose that after each side presents evidence, the jury remains unsure whether the defendant committed the crime. Since the jury does not have enough evidence to be sure, they must, under the law of the United States find the defendant “not guilty.” This does not mean that the jury thinks the defendant is innocent, only that they do not have enough evidence to be sure they are guilty. Similarly in a hypothesis test, a large p-value indicates a lack of evidence against the null hypothesis, rather than evidence supporting it. As such, we should avoid statements suggesting we “support”, “accept”, or “believe” the null hypothesis, and simply state that we lack evidence against it.

Things to say when the p-value is large:

- The data are consistent with the null hypothesis.
- We do not have enough evidence against the null hypothesis.
- We cannot reject the null hypothesis.
- The null hypothesis is plausible.

Things NOT to say when the p-value is large:

- The data support the null hypothesis.
- The data provide evidence for the null hypothesis.
- We accept the null hypothesis.

- We conclude that the null hypothesis is true.

Thus, if we had obtained a large p-value in the comparison in mercury levels between Northern and Southern lakes, the appropriate conclusion would be

“We do not have evidence that the average mercury level differs between lakes in Northern Florida, compared to Southern Florida.”

Even if we got a large p-value it would be incorrect to say “There is no difference in average mercury levels between lakes in Northern Florida and Southern Florida.”

We would just be saying that given the size of our sample and the amount of variability on the date, we cannot rule out the possibility of observing a difference like we did by chance, when there really is no difference.

4 Inference from Models

Learning Outcomes:

24. State the assumptions of the normal error regression model in context.
25. Calculate confidence intervals and t-statistics using standard error formulas.
26. State the hypotheses associated with each p-value in the lm summary table and explain what we should conclude from each test.
27. Show how to calculate standard errors, t-statistics, residual standard error, Multiple R², and F-Statistic in the lm summary output in R.
28. Interpret confidence and prediction intervals for expected response, as well as intervals for parameter estimates, and identify the sources of variability affecting each.
29. Explain the regression effect in context.
30. Make responsible decisions from data, considering issues such as representation and randomization, multiple testing, and the difference between statistical significance and practical importance.

4.1 The Normal Error Regression Model

You've probably noticed that most of the sampling distributions of statistics we've seen were symmetric and bell-shaped in nature. When working with statistics that have symmetric and bell-shaped distributions and know standard error formulas, it is possible to use well-known probability facts to obtain confidence intervals and perform hypothesis tests without actually performing simulation.

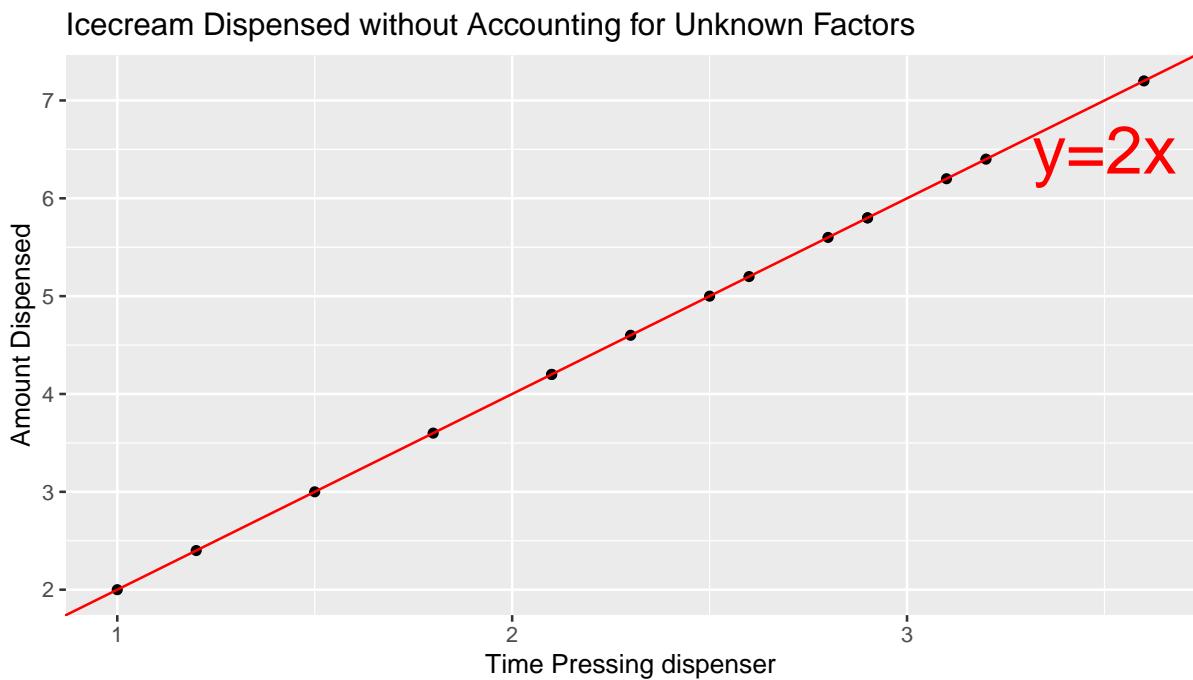
In this chapter, we'll examine a set of assumptions that, if true, would ensure that statistics like means and differences in means, and regression coefficients follow symmetric and bell-shaped distributions. We'll learn how to use facts from probability to calculate confidence intervals and p-values without actually performing simulation in these instances.

4.1.1 Example: Ice Cream dispenser



Image from: <https://www.partstown.com/about-us/spaceman-ice-cream-machine-troubleshooting>

Suppose an ice cream machine is manufactured to dispense 2 oz. of ice cream per second, on average. If 15 people used the ice cream machine, holding the dispenser for different amounts of time, and each person got exactly 2 oz. per second, the relationship between time holding the dispenser and amount dispensed would look like this:

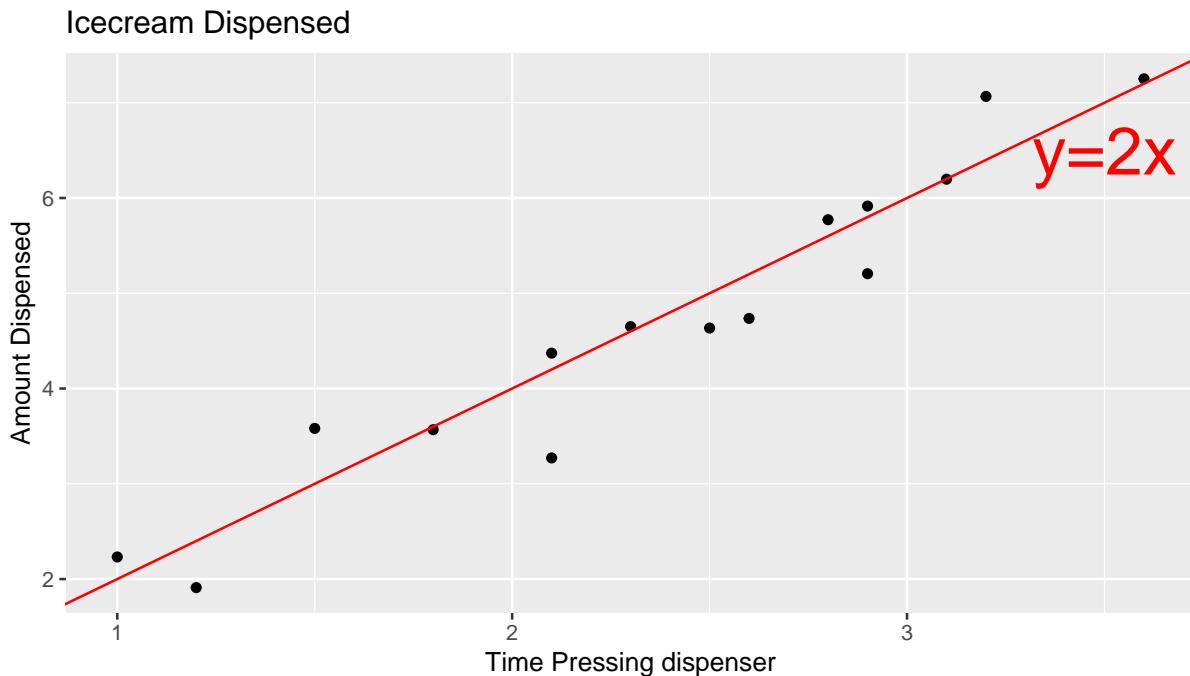


In reality, however, the actual amount dispensed each time it is used will vary due to unknown factors like:

- force applied to dispenser

- temperature
- build-up of ice cream
- other unknown factors

Thus, if 15 real people held the dispenser and recorded the amount of ice cream they got, the scatter plot we would see would look something like this:



We can think of the amount of ice cream a person receives as being a result of two separate components, often referred to as **signal** and **noise**.

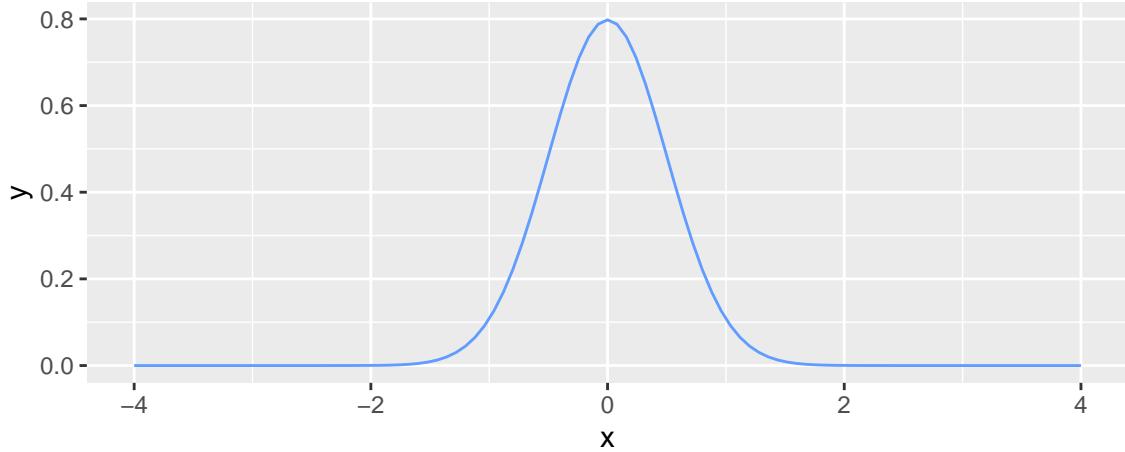
Signal represents the average amount of ice cream a person is expected to receive based on the amount of time holding the dispenser. In this case, signal is given by the function Expected Amount = $2 \times$ Time. Everyone who holds the dispenser for t seconds is expected to receive $2t$ ounces of ice cream.

Noise represents how much each person's actual amount of ice cream deviates from their expected amount. For example, a person who holds the dispenser for 1.5 seconds and receives 3.58 oz. of ice cream will have received 0.58 ounces more than expected due to noise (i.e. factors beyond time holding the dispenser).

In a statistical model, we assume that the response value of a response variable we observe is the sum of the signal, or expected response, which is a function of the explanatory variables in the model, and noise, which results from deviations due to factors beyond those accounted for in the model.

4.1.2 Normal Distribution

It is common to model noise using a symmetric, bell-shaped distribution, known as a **normal distribution**.

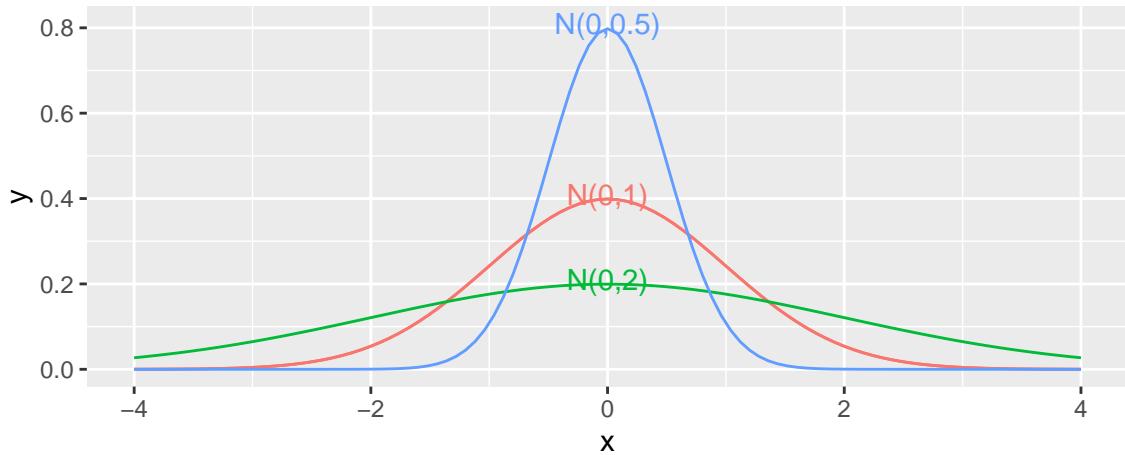


We can think of the error term as a random draw from somewhere in the area below the bell-curve. For example, in the above illustration, most of the area under the curve lies between $-1 \leq x \leq 1$. If this curve represented the noise term in the ice cream example, it would mean that most people's actual amount of ice cream dispensed would be within ± 1 ounce of their expected amount (or signal). Notice that the normal distribution is centered at 0, indicating that on average, a person would be expected to get an amount exactly equal to their signal, but that they might deviate above or below this amount by unexplained factors, which can be modeled by random chance.

A normal distribution is defined by two parameters:

- μ representing the center of the distribution
- σ representing the standard deviation

This distribution is denoted $\mathcal{N}(\mu, \sigma)$.



When the standard deviation is small, such as for the blue curve, noise tends to be close to 0, meaning the observed values will be close to their expectation. On the other hand, the green curve, which has higher standard deviation, would often produce noise values as extreme as ± 2 or more.

Note that the square of the standard deviation σ^2 is called the variance. Some books denote the normal distribution as $\mathcal{N}(\mu, \sigma^2)$, instead of $\mathcal{N}(\mu, \sigma)$. We will use the $\mathcal{N}(\mu, \sigma)$ here, which is consistent with R.

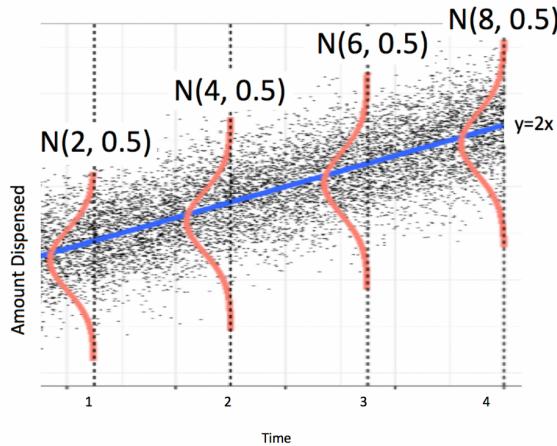
4.1.3 Modeling Ice Cream Dispenser

In this example, we'll simulate the amount of ice cream dispensed for each person by adding a random number from a normal distribution with mean 0 and standard deviation 0.5 to the expected amount dispensed, which is given by $2x$, where x represents time pressing the dispenser. We'll let ϵ_i represent the random noise term for the i th person.

Thus, amount dispensed (Y_i) for person i is given by

$$Y_i = 2x_i + \epsilon_i, \text{ where } \epsilon_i \sim \mathcal{N}(0, 0.5)$$

Illustration



Modified from image at <https://bookdown.org/robback/bookdown-bysh/ch-poissonreg.html>

Figure 4.1: The amount dispensed follows a normal distribution with mean equal to twice the time holding the dispenser and standard deviation 0.5

We simulate the amount dispensed for a sample of 15 people below. The `rnorm(n, μ , σ)` function generates n random numbers from a normal distribution with mean μ , and standard deviation σ . The code below simulates 15 random numbers and adds them to the expected amount dispensed for 15 people who hold the dispenser for the times shown.

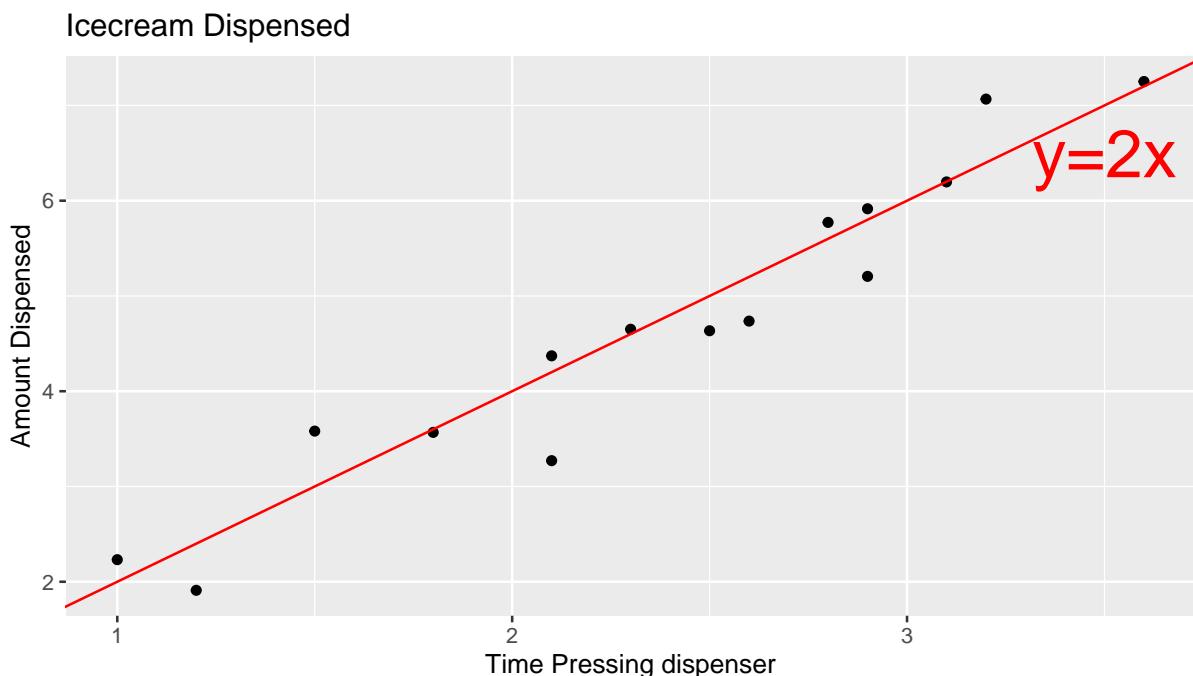
```
set.seed(10082020)
# set times
time <- c(1, 1.2, 1.5, 1.8, 2.1, 2.1, 2.3, 2.5, 2.6, 2.8, 2.9, 2.9, 3.1, 3.2, 3.6)
expected <- 2*time # expected amount
noise <- rnorm(15, 0, 0.5) %>% round(2) #generate noise from normal distribution
amount <- 2*time + noise # calculate observed amounts
Icecream <- data.frame(time, signal, noise, amount) # set up data table
kable((Icecream)) #display table
```

time	signal	noise	amount
1.0	2.0	0.23	2.23
1.2	2.4	-0.49	1.91
1.5	3.0	0.58	3.58
1.8	3.6	-0.03	3.57
2.1	4.2	0.17	4.37
2.1	4.2	-0.93	3.27
2.3	4.6	0.05	4.65
2.5	5.0	-0.37	4.63

time	signal	noise	amount
2.6	5.2	-0.46	4.74
2.8	5.6	0.17	5.77
2.9	5.8	-0.59	5.21
2.9	5.8	0.12	5.92
3.1	6.2	0.00	6.20
3.2	6.4	0.67	7.07
3.6	7.2	0.05	7.25

The scatterplot displays the amount dispensed, compared to the time pressing the dispenser. The red line indicates the line $y = 2x$. If there was no random noise, then each person's amount dispensed would lie exactly on this line.

```
ggplot(data=Icecream1, aes(x=time, y=amount)) + geom_point() + ggtitle("Icecream Dispensed")
  annotate("text", label="y=2x", x= 3.5, y=6.5, size=10, color="red")
```

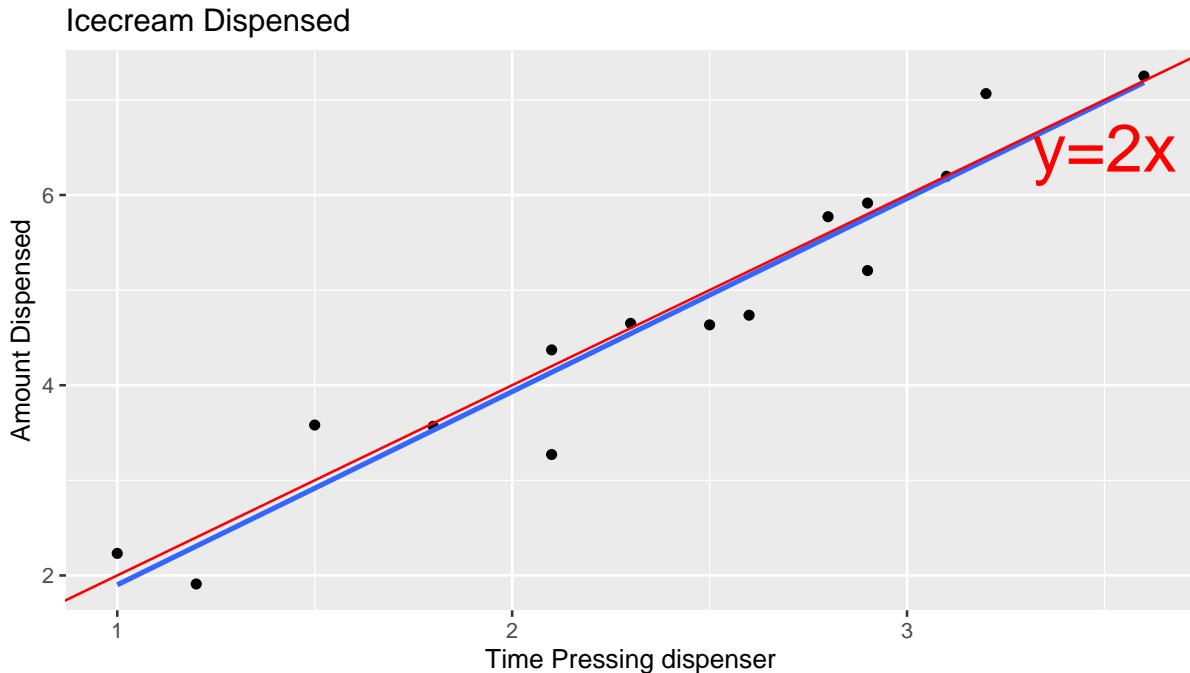


Estimating Regression Line

In a real situation, we would not see the signal and noise columns in the table or the red line on the graph. We would only see the time and amount, and points on the scatter plot. From these, we would need to estimate the location of the red line by fitting a least squares regression line to the data, as we've done before.

The blue line represents the location of the least squares regression line fit to the time and amounts observed.

```
ggplot(data=Icecream1, aes(x=time, y=amount)) + geom_point() + ggtitle("Icecream Dispensed")
  annotate("text", label="y=2x", x= 3.5, y=6.5, size=10, color="red")
```



The blue line is close, but not identical to the red line, representing the true (usually unknown) signal.

The slope and intercept of the blue line are given by:

```
IC_Model <- lm(data=Icecream1, lm(amount~time))
IC_Model
```

```
Call:
lm(formula = lm(amount ~ time), data = Icecream1)
```

```
Coefficients:
(Intercept)      time
-0.1299        2.0312
```

Notice that these estimates are close, but not identical to the intercept and slope of the red line, which are 0 and 2, respectively.

The equation of the red line is given by:

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i, \text{ where } \epsilon_i \sim \mathcal{N}(0, \sigma),$$

where Y_i represents amount dispensed, and X_i represents time. β_0 , β_1 , and σ are the unknown model parameters associated with the ice cream machine's process.

Using the values of b_0 and b_1 obtained by fitting a model to our observed data as estimates of β_0 and β_1 , our estimated regression equation is

$$\$ \$ Y_i = b_0 + b_1 X_i + \epsilon_i = -0.1299087 + 2.0312489 X_i + \epsilon_i$$

$\$ \$$

$$\text{where } \epsilon_i \sim \mathcal{N}(0, \sigma).$$

An estimate for σ is given by

$$s = \sqrt{\frac{\text{SSR}}{n-(p+1)}} = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{(n-(p+1))}}.$$

We calculate s , using R.

```
s <- sqrt(sum(IC_Model$residuals^2)/(15-2))
s
```

[1] 0.4527185

In statistics it is common to use Greek letters like β_j and σ , to represent unknown model parameters, pertaining to a population or process (such as the ice cream dispenser), and English letters like b_j and s to represent statistics calculated from data.

The estimates of $b_0 = -0.1299087$, $b_1 = 2.0312489$, and $s = 0.4527185$ are reasonably close estimates to the values $\beta_0 = 0$, $\beta_1 = 2$, and $\sigma = 0.5$, that we used to generate the data.

In a real situation, we'll have only statistics b_0 , b_1 , and s , and we'll need to use them to draw conclusions about parameters $\beta_0 = 0$, $\beta_1 = 2$, and $\sigma = 0.5$.

4.1.4 Normal Error Regression Model

In the ice cream example, the relationship between expected amount and time holding the dispenser was given by a linear equation involving a single numeric explanatory variable. We can generalize this to situations with multiple explanatory variables, which might be numeric or categorical.

Individual observations are then assumed to vary from their expectation in accordance with a normal distribution, representing random noise (or error).

The mathematical form of a normal error linear regression model is

$$Y_i = \beta_0 + \beta_1 X_{i1} + \dots + \beta_p X_{ip} + \epsilon_i, \text{ with } \epsilon_i \sim \mathcal{N}(0, \sigma).$$

Note that in place of X_{ip} , we could have indicators for categories, or functions of X_{ip} , such as X_{ip}^2 , $\log(X_{ip})$, or $\sin(X_{ip})$.

- The quantities $\beta_0, \beta_1, \dots, \beta_p$ are parameters, pertaining to the true but unknown data generating mechanism.
- The estimates b_0, b_1, \dots, b_p , are statistics, calculated from our observed data.
- We use statistics b_0, b_1, \dots, b_p to obtain confidence intervals and hypothesis tests to make statements about parameters $\beta_0, \beta_1, \dots, \beta_p$.

Assumptions of Normal Error Regression Model

When generating data for the ice cream machine, we assumed four things:

1. **Linearity** - the expected amount of ice cream dispensed (i.e the signal) is a linear function of time the dispenser was pressed.
2. **Constant Variance** - individual amounts dispensed varied from their expected amount with equal variability, regardless of the amount of time. That is, the amount of variability in individual amounts dispensed was the same for people who held the dispenser for 1 s. as for people who held it for 2 s. or 3 s., etc.
3. **Normality** - individual amounts dispensed varied from their expected amount in accordance with a normal distribution.
4. **Independence** - the amount of ice cream dispensed for one person was not affected by the amount dispensed for anyone else.

More generally, the normal error regression model assumes:

1. **Linearity** - the expected response is a linear function of the explanatory variable(s).

2. **Constant Variance** - the variance between individual values of the response variable are the same for any values/categories of the explanatory variable(s)
3. **Normality** - for any values/categories of the explanatory variable(s) individual response values vary from their expectation in accordance with a normal distribution.
4. **Independence** - individual response values are not affected by one another

4.1.5 Examples of Normal Error Regression Model

We can formulate all of the examples we've worked with so far in terms of the normal error regression model.

In the house price example, consider the following models:

Model 1:

$$\text{Price}_i = \beta_0 + \beta_1 \text{Sq.Ft.}_i + \epsilon_i, \text{ where } \epsilon_i \sim \mathcal{N}(0, \sigma).$$

If we use this model, we're saying that we believe the expected price of a house is a linear function of its size, and that for any given size, the distribution of actual prices are normally distributed around their expected value of $\beta_0 + \beta_1 \text{Sq.Ft.}_i$. Furthermore, we're saying that the amount of variability in house prices is the same for houses of any size.

Model 2:

$$\text{Price}_i = \beta_0 + \beta_2 \text{Waterfront}_i + \epsilon_i, \text{ where } \epsilon_i \sim \mathcal{N}(0, \sigma).$$

If we use this model, we're saying that we believe the expected price of a house depends only on whether or not it is on the waterfront, and that prices of both waterfront and non-waterfront houses follow normal distributions. Though these distributions may have different means (β_0 for non-waterfront houses, and β_1 for waterfront houses), the amount of variability in prices should be the same for waterfront as non-waterfront houses.

Model 3:

$$\text{Price}_i = \beta_0 + \beta_1 \text{Sq.Ft.}_i + \beta_2 \text{Waterfront}_i + \beta_3 \times \text{Sq.Ft.}_i \times \text{Waterfront}_i + \epsilon_i, \text{ where } \epsilon_i \sim \mathcal{N}(0, \sigma).$$

and

Model 4:

$$\text{Price}_i = \beta_0 + \beta_1 \text{Sq.Ft.}_i + \beta_2 \text{Waterfront}_i + \beta_3 \times \text{Sq.Ft.}_i \times \text{Waterfront}_i + \epsilon_i, \text{ where } \epsilon_i \sim \mathcal{N}(0, \sigma).$$

Both of Models 3 and 4 assume that actual prices of houses with the same size and waterfront status are normally distributed, and that the mean of the normal distribution is a linear function of its size. Model 3 allows for the intercept of the lines to differ between waterfront and non-waterfront houses, while Model 4 allows both the intercept and slope to differ. Both

assume that the amount of variability among houses of the same size and waterfront status is the same.

4.1.6 Implications of Normal Error Regression Model

If we really believe that data come about as the normal error regression model describes, then probability theory tells us that regression coefficients b_j 's, representing differences between categories for categorical variables and rates of change for quantitative variables, follow symmetric and bell-shaped distributions. We can use this fact to create confidence intervals and perform hypothesis tests, without needing to perform simulation. This is, in fact what R does in its model summary output.

These methods are only valid, however, if data can reasonably be thought of as having come from a process consistent with the assumptions of the normal error regression model process. If we don't believe that our observed data can be reasonably thought of as having come from such a process, then the confidence intervals and p-values produced by R, and other places that rely on probability-based methods will not be reliable.

We close the section with a philosophical question:

Do data really come about from processes like the normal error regression model? That is, do you think it is reasonable to believe that data we see in the real world (perhaps the amount of ice cream dispensed by an ice cream machine) represent independent outcomes of a process in which expected outcomes are a linear function of explanatory variables, and deviate from their expectation according to a normal distribution with constant variance?

We won't attempt to answer that question here, but it is worth thinking about. After all, it is an assumption on which many frequently employed methods of statistical inference depends.

4.2 Standard Error and Confidence Intervals

4.2.1 Common Standard Error Formulas

In the normal error regression model, σ is an unknown model parameter representing the standard deviation in response values among cases with the same values/categories of explanatory variable(s). We estimate σ using the statistic

$$s = \sqrt{\frac{\text{SSR}}{n-(p+1)}} = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{(n-(p+1))}}$$

s is not the same as the standard errors of statistics we estimated using bootstrapping in Chapter 3. It is, however, related to these standard errors. In fact, standard errors for common statistics like sample means, differences in means, and regression coefficients can be approximated using formulas involving s .

So far, we've used simulation (permutation tests and bootstrap intervals) to determine the amount of variability associated with a test statistic or estimate, in order to perform hypotheses tests and create confidence intervals. In special situations, there are mathematical formulas, based on probability theory, that can be used to approximate these standard errors without having to perform the simulations.

Theory-Based Standard Error Formulas

Statistic	Standard Error
Single Mean	$SE(b_0) = \frac{s}{\sqrt{n}}$
Difference in Means	$SE(b_j) = s \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}$
Single Proportion	$SE(\hat{p}) = \sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$
Difference in Proportions	$SE(\hat{p}) = \sqrt{\left(\frac{\hat{p}_1(1-\hat{p}_1)}{n_1} + \frac{\hat{p}_2(1-\hat{p}_2)}{n_2} \right)}$
Intercept in Simple Linear Regression	$SE(b_0) = s \sqrt{\frac{1}{n} + \frac{\bar{x}^2}{\sum(x_i - \bar{x})^2}}$
Slope in Simple Linear Regression	$SE(b_1) = \sqrt{\frac{s^2}{\sum(x_i - \bar{x})^2}} = \sqrt{\frac{1}{n-2} \frac{\sum(\hat{y}_i - y_i)^2}{\sum(x_i - \bar{x})^2}}$

- $s = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{(n-(p+1))}} = \sqrt{\frac{\text{SSR}}{(n-(p+1))}}$, (p is number of regression coefficients not including b_0). s is an estimate of the variability in the response variable among cases where the explanatory variable(s) are the same. Note that in the one-sample case, this simplifies to the standard deviation formula we've seen previously.

- In the 2nd formula, the standard error estimate $s\sqrt{\frac{1}{n_1+n_2}}$ is called a “pooled” estimate since it combines information from all groups. When there is reason to believe standard deviation differs between groups, we often use an “unpooled” standard error estimate of $\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}$, where s_1, s_2 represent the standard deviation for groups 1 and 2. This can be used in situations where the explanatory variable is categorical and we have doubts about the constant variance assumption.

There is no theory-based formula for standard error associated with the median or standard deviation. For these, and many other statistics, we rely on simulation to estimate variability between samples.

There are formulas for standard errors associated with coefficients in multiple regression, but these require mathematics beyond what is assumed in this class. They involve linear algebra and matrix inversion, which you can read about [here](#) if you are interested.

When the sampling distribution is symmetric and bell-shaped, approximate 95% confidence intervals can be calculated using the formula,

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

where the standard error is estimated using a formula, rather than through bootstrapping.

We'll go through some examples to illustrate how to calculate and interpret s and $SE(b_j)$.

4.2.2 Example: Difference in Means

We'll use the normal error regression model to predict a lake's mercury level, using location (N vs S) as the explanatory variable.

The regression model is:

$$\text{Mercury} = \beta_0 + \beta_1 \times \text{Location}_{\text{South}} + \epsilon_i, \text{ where } \epsilon_i \sim \mathcal{N}(0, \sigma)$$

This model assumes:

1. **Linearity** - there is no linearity assumption when the explanatory variable is categorical.
2. **Constant Variance** - the variance between mercury levels of individual lakes is the same for Northern Florida, as for Southern Florida.
3. **Normality** - mercury levels are normally distributed in Northern Florida and also in Southern Florida.

4. **Independence** - mercury levels of individual lakes are not affected by those of other lakes.

The `lm` summary command in R returns information pertaining to the model.

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

```
Call:
lm(formula = Mercury ~ 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 estimates column returns the estimates of b_0 and b_1 . The estimated regression equation is

$$\widehat{\text{Mercury}} = 0.42455 + 0.27185 \times \text{Location}_{\text{South}}$$

Estimating σ

The **Residual Standard Error** in the output gives the value of s , and estimate of σ .

$$s = \sqrt{\frac{\text{SSR}}{n-(p+1)}} = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{(n-(p+1))}}$$

SSR is:

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

```
[1] 5.126873
```

Since LocationSouth is our only explanatory variable, $p = 1$, and since we have 53 lakes in the sample, $n = 53$.

Thus,

$$s = \sqrt{\frac{5.126873}{53-(1+1)}} = 0.317.$$

The standard deviation in mercury concentrations among lakes in the same part of the state is estimated to be 0.317 ppm.

The second column in the coefficients table gives standard errors associated with b_0 and b_1 . These tell us how much these statistics are expected to vary between samples of the given size.

Estimating $SE(b_0)$ and $SE(b_1)$

We'll use the theory-based formulas to calculate the standard errors for b_0 and b_1 .

In this case, b_0 represents a single mean, the mean mercury level for lakes in Northern Florida. Since there are 33 such lakes, the calculation is:

$$SE(b_0) = \frac{s}{\sqrt{n}} = \frac{0.317}{\sqrt{33}} \approx 0.055$$

The standard error of intercept b_0 is 0.055. This represents the variability in average mercury level in different samples of 33 lakes from Northern Florida.

$$SE(b_1) = s \sqrt{\frac{1}{n_1} + \frac{1}{n_2}} = 0.317 \sqrt{\frac{1}{20} + \frac{1}{33}} = 0.0898$$

The standard error of slope b_1 is 0.0898. This represents the variability in average difference in mercury levels between samples of 33 lakes from Northern Florida and 20 lakes from Southern Florida.

These numbers match the values in the `Std. Error` column of the coefficients table of the `lm` summary output.

A 95% confidence interval for β_0 is given by

$$\begin{aligned} b_0 &\pm 2 \times SE(b_0) \\ &= 0.42455 \pm 2 \times 0.055 \\ &= (0.314, 0.535) \end{aligned}$$

A 95% confidence interval for β_1 is given by

$$\begin{aligned}
 b_1 &\pm 2 \times \text{SE}(b_1) \\
 &= 0.272 \pm 2 \times 0.0898 \\
 &= (0.09, 0.45)
 \end{aligned}$$

We are 95% confident that the average mercury level in Southern Lakes is between 0.09 ppm and 0.45 ppm higher than in Northern Florida.

These intervals can be obtained directly in R using the `confint` command.

```
confint(Lakes_M, level=0.95)
```

2.5 %	97.5 %
(Intercept)	0.31374083 0.5353501
LocationS	0.09157768 0.4523314

Comparison to Bootstrap

We previously calculated $\text{SE}(b_1)$ and a 95% confidence interval for β_1 , which are shown below.

Bootstrap Standard Error

```
SE_b1 <- sd(NS_Lakes_Bootstrap_Results$Bootstrap_b1)
SE_b1
```

```
[1] 0.09549244
```

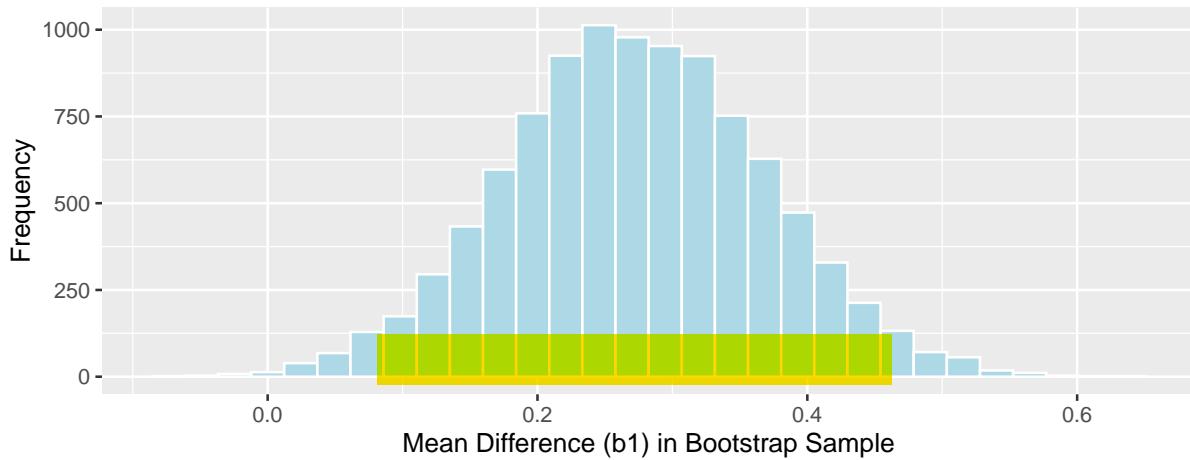
Bootstrap 95% Confidence Interval:

```
Sample_b1 <- Lakes_M$coefficients[2]
c(Sample_b1 - 2*SE_b1, Sample_b1 + 2*SE_b1)
```

```
LocationS LocationS
0.08096967 0.46293942
```

```
NS_Lakes_Bootstrap_Plot_b1 +
  geom_segment(aes(x=Sample_b1 - 2*SE_b1, xend=Sample_b1 + 2*SE_b1, y=50, yend=50),
               color="gold", size=10, alpha=0.01)
```

Northern vs Southern Lakes: Bootstrap Distribution for b1



The formula-based standard error and confidence interval and those produced by bootstrapping are both approximations, so they are not expected to be identical, but should be close.

4.2.3 Example: Regression Slope and Intercept

We'll use the normal error regression model to predict a lake's mercury level, using pH as the explanatory variable.

The regression model is:

$$\text{Mercury} = \beta_0 + \beta_1 \times \text{pH} + \epsilon_i, \text{ where } \epsilon_i \sim \mathcal{N}(0, \sigma)$$

This model assumes:

1. **Linearity** - the mercury level is a linear function of pH.
2. **Constant Variance** - the variance between mercury levels of individual lakes is the same for each pH level.
3. **Normality** - for each pH, mercury levels are normally distributed.
4. **Independence** - mercury levels of individual lakes are not affected by those of other lakes.

We might have doubts about some of these assumptions. For example, if we believe there might be more variability in mercury levels among lakes with higher pH levels than lower ones (a violation of the constant variance assumption), or that lakes closer together are likely to have similar mercury levels (a violation of the independence assumption) then the results of the model might not be reliable.

Later in this chapter, we'll learn ways to check the appropriateness of these assumptions. For now, we'll assume that the model is a reasonable enough approximation of reality and use it accordingly.

The `lm` summary command in R returns information pertaining to the model.

```
M_pH <- lm(data=FloridaLakes, Mercury ~ pH)
summary(M_pH)
```

```
Call:
lm(formula = Mercury ~ pH, data = FloridaLakes)

Residuals:
    Min      1Q  Median      3Q     Max 
-0.48895 -0.19188 -0.05774  0.09456  0.71134 

Coefficients:
            Estimate Std. Error t value   Pr(>|t|)    
(Intercept) 1.53092   0.20349   7.523 0.000000000814 ***
pH          -0.15230   0.03031  -5.024 0.000006572811 ***  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2816 on 51 degrees of freedom
Multiple R-squared:  0.3311,    Adjusted R-squared:  0.318 
F-statistic: 25.24 on 1 and 51 DF,  p-value: 0.000006573
```

The estimated regression equation is

$$\widehat{\text{Mercury}} = 1.53 - 0.15 \times \text{pH}$$

Estimating σ

As we saw in the ice cream example, an estimate for σ is given by

$$s = \sqrt{\frac{\text{SSR}}{n-(p+1)}} = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{(n-(p+1))}}$$

We calculate s , using R.

SSR is:

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

```
[1] 4.045513
```

Since pH is our only explanatory variable, $p = 1$, and since we have 53 lakes in the sample, $n = 53$.

Thus,

$$s = \sqrt{\frac{4.045513}{53-(1+1)}} = 0.2816.$$

The standard deviation in mercury concentrations among lakes with the same pH is estimated to be 0.2816 ppm.

The **residual standard error** in the summary output returns the value of s .

Estimating $SE(b_0)$ and $SE(b_1)$

Now, we'll use the theory-based formulas to calculate standard error associated with the intercept and slope of the regression line relating mercury level and pH in Florida lakes.

We calculate \bar{x} , the mean pH, and $\sum(x_i - \bar{x})^2$.

```
xbar <- mean(FloridaLakes$pH)
xbar
```

```
[1] 6.590566
```

```
Sxx <- sum((FloridaLakes$pH - xbar)^2)
Sxx
```

```
[1] 86.32528
```

$$SE(b_0) = s \sqrt{\frac{1}{n} + \frac{\bar{x}^2}{\sum(x_i - \bar{x})^2}} = 0.2816 \sqrt{\frac{1}{53} + \frac{6.59^2}{86.325}} \approx 0.2034$$

The standard error of intercept b_0 is 0.2034. This represents the variability in estimated mercury level in lakes with pH of 0, between different samples of size 53. Since none of the lakes have pH levels of 0, this is not a meaningful statistic.

$$SE(b_1) = \sqrt{\frac{s^2}{\sum(x_i - \bar{x})^2}} = \sqrt{\frac{0.2816^2}{86.325}} \approx 0.0303$$

The standard error of slope b_1 is 0.0303. This represents the variability in estimated rate of change in mercury level per one unit increase in pH, between different samples of size 53.

These numbers match the values in the `Std. Error` column of the coefficients table of the `lm` summary output.

A 95% confidence interval for β_1 is given by

$$\begin{aligned} b_1 &\pm 2 \times \text{SE}(b_1) \\ &= -0.15 \pm 2 \times 0.0303 \\ &= (-0.09, -0.21) \end{aligned}$$

We are 95% confident that for each one unit increase in pH, mercury concentration is expected to decrease between 0.09 and 0.21 ppm, on average.

We can also use the `confint()` command to obtain these intervals.

```
confint(M_pH, level=0.95)
```

	2.5 %	97.5 %
(Intercept)	1.1223897	1.93944769
pH	-0.2131573	-0.09144445

Comparison to Bootstrap

We previously calculated $\text{SE}(b_1)$ and a 95% confidence interval for β_1 , which are shown below.

Bootstrap Standard Error for b_1 :

```
Sample_b1 <- M_pH$coefficients[2]
SE_b1 <- sd(Lakes_Bootstrap_Slope_Results$Bootstrap_b1)
SE_b1
```

```
[1] 0.02694529
```

Bootstrap 95% Confidence Interval for β_1 :

```
c(Sample_b1 - 2*SE_b1, Sample_b1 + 2*SE_b1)
```

pH	pH
-0.20619145	-0.09841028

The formula-based standard error and confidence interval and those produced by bootstrapping are both approximations, so they are not expected to be identical, but should be close.

4.2.4 Example: Single Mean

Finally, we'll use a model to estimate the average mercury level among all lakes in Florida. For this, we use an "Intercept-only" model, with no explanatory variables.

The model equation is:

The regression model is:

$$\text{Mercury} = \beta_0 + \epsilon_i, \text{ where } \epsilon_i \sim \mathcal{N}(0, \sigma)$$

This model assumes:

1. **Linearity** - there is no linearity assumption since there are no explanatory variables.
2. **Constant Variance** - there is no linearity assumption since there are no explanatory variables.
3. **Normality** - mercury levels among all lakes in Florida are normally distributed.
4. **Independence** - mercury levels of individual lakes are not affected by those of other lakes.

The `lm` summary command in R returns information pertaining to the model.

```
Lakes_MO <- lm(data=FloridaLakes, Mercury ~ 1)
summary(Lakes_MO)
```

Call:

```
lm(formula = Mercury ~ 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

The model estimates that the average mercury level is 0.527, which matches the sample mean, as we know it should.

Estimating σ

An estimate for σ is given by

$$s = \sqrt{\frac{\text{SSR}}{n-(p+1)}} = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{(n-(p+1))}}.$$

We calculate s , using R.

SSR is:

```
sum(Lakes_M0$residuals^2)
```

```
[1] 6.047875
```

Since there are no explanatory variables, $p = 0$, and since we have 53 lakes in the sample, $n = 53$.

When there are no explanatory variables, s is simply an estimate of the standard deviation of individual observations of the response variable.

Thus,

$$s = \sqrt{\frac{6.047875}{53-(0+1)}} = 0.341.$$

The standard deviation in mercury concentrations among lakes in Florida is estimated to be 0.341 ppm.

This matches **residual standard error** in the summary output returns the value of s .

Estimating $\text{SE}(b_0)$

Now, we'll use the theory-based formulas to calculate standard error associated with b_0 , the average mercury level among all lakes in Florida.

$$\text{SE}(b_0) = \frac{s}{\sqrt{n}} = \frac{0.341}{\sqrt{53}} \approx 0.04684$$

The standard error of mean b_0 is 0.04684. This represents the variability in mean mercury level among samples of 53 lakes.

Notice that the standard error of the mean is less than the standard deviation of individual observations, and that the standard error of the mean decreases as sample size increases, as we've seen previously.

A 95% confidence interval for β_0 , is:

$$\begin{aligned}
 b_0 &\pm 2 \times \text{SE}(b_0) \\
 &= 0.527 \pm 2 \times 0.0484 \\
 &= (0.43, 0.62)
 \end{aligned}$$

We are 95% confident that the average mercury level among all lakes in Florida is between 0.43 and 0.62 ppm.

We can also obtain this interval using the `confint()` command.

```
confint(Lakes_M0, level=0.95)
```

```
2.5 %      97.5 %
(Intercept) 0.4331688 0.6211709
```

Comparison to Bootstrap

Here are the bootstrap standard error and confidence interval we calculated previously.

Bootstrap Standard Error

```
SE_mean <- sd(Lakes_Bootstrap_Results_Mean$Bootstrap_Mean)
SE_mean
```

```
[1] 0.04595372
```

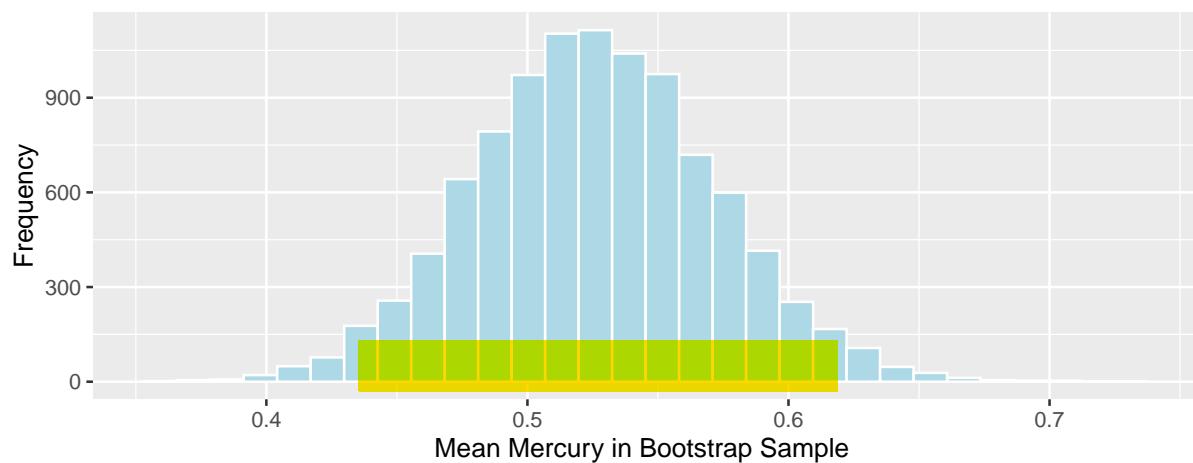
Bootstrap 95% Confidence Interval:

```
c(mean - 2*SE_mean, mean + 2*SE_mean)
```

```
[1] 0.4352624 0.6190773
```

```
Lakes_Bootstrap_Mean_Plot +
  geom_segment(aes(x=mean - 2*SE_mean, xend=mean + 2*SE_mean, y=50, yend=50),
               color="gold", size=10, alpha=0.01)
```

Bootstrap Distribution for Sample Mean in Florida Lakes



The standard error and interval are very close to those obtained using the approximation formulas.

4.2.5 CI Method Comparison

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

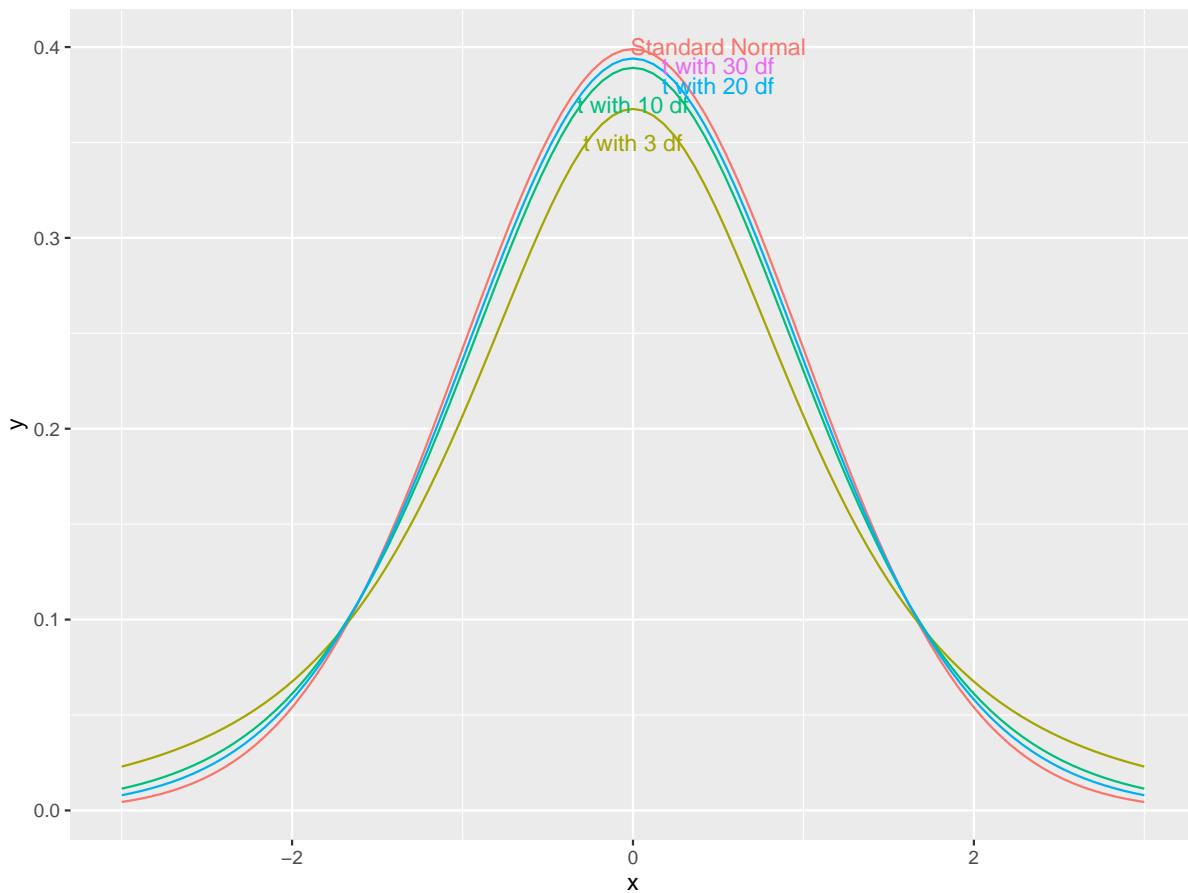
4.3 t-tests

4.3.1 t-Distribution

A t-distribution is a symmetric, bell-shaped curve. The t-distribution depends on a parameter called **degrees of freedom**, which determines the thickness of the distribution's tails.

When the sampling distribution for a statistic is symmetric and bell-shaped, it can be approximated by a t-distribution. As the sample size increases, so do the degrees of freedom in the t-distribution.

As the degrees of freedom grow large, we see that the t-distributions get closer together, converging to a bell-shaped curve. This distribution is called a **standard normal distribution**.



4.3.2 t-tests

When the sampling distribution of a statistic is symmetric and bell-shaped, then the ratio

$$t = \frac{\text{Statistic}}{\text{SE}(\text{Statistic})}$$

approximately follows a t-distribution.

The statistic t is called a t-statistic.

We'll use this t-statistic as the test statistic in our hypothesis test.

The degrees of freedom are given by $n - (p + 1)$, where p represents the number of terms in the model, not including the intercept.

To find a p-value, we use a t-distribution to find the probability of obtaining a t-statistic as or more extreme than the one calculated from our data.

A hypothesis test based on the t-statistic and t-distribution is called a **t-test**.

4.3.3 t-test Examples

4.3.3.1 Example 1: N vs S Lakes

The equation of the model is:

$$\widehat{\text{Mercury}} = b_0 + b_1 \times \text{South}$$

We fit the model in R and display its summary output below.

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

Call:

```
lm(formula = Mercury ~ 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

```

Looking at the coefficients table, we've already seen where the first two columns, Estimate and Std. Error come from. The last two columns, labeled t value and “Pr(>|t|)” represent the t-statistic and p-value for the t-test associated with the that the regression parameter on that line is zero. (i.e. $\beta_j = 0$).

t-test for line Locations

To test whether there is evidence of a difference in mercury levels between Northern and Southern Florida, we'll test the null hypothesis ($\beta_1 = 0$). This corresponds to the locations line of the R output.

Null Hypothesis: There is no difference in average mercury levels between Northern and Southern Florida ($\beta_1 = 0$).

Alternative Hypothesis: There is a difference in average mercury levels in Northern and Southern Florida ($\beta_1 \neq 0$).

In the previous section, we calculated the standard error for b_1 to be 0.0898, respectively.

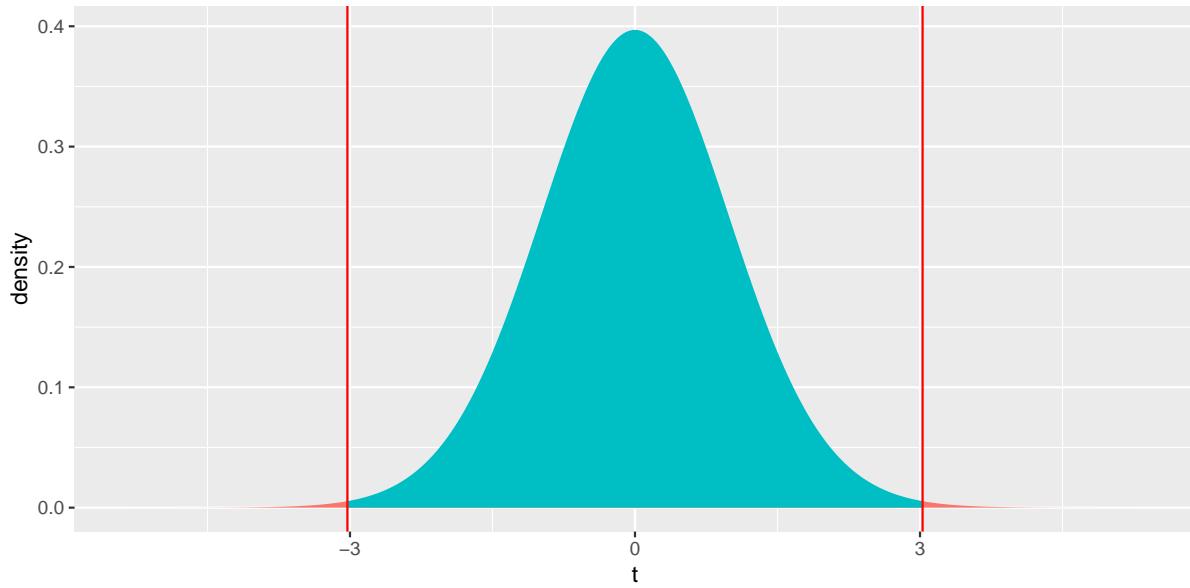
Test Statistic: $t = \frac{b_j}{SE(b_j)} = \frac{0.27195}{0.0898} = 3.027$

To calculate the p-value, we plot the t-statistic of 3.027 that we observed in our data and observe where it lies on a t-distribution.

```

ts=3.027
gf_dist("t", df=51, geom = "area", fill = ~ (abs(x)< abs(ts)), show.legend=FALSE) + geom_vline

```



The `pt` function returns the probability of getting a t-statistic higher than 3.027. We multiply by 2 to also get the area in the left tail.

p-value:

```
2*pt(-abs(ts), df=51)
```

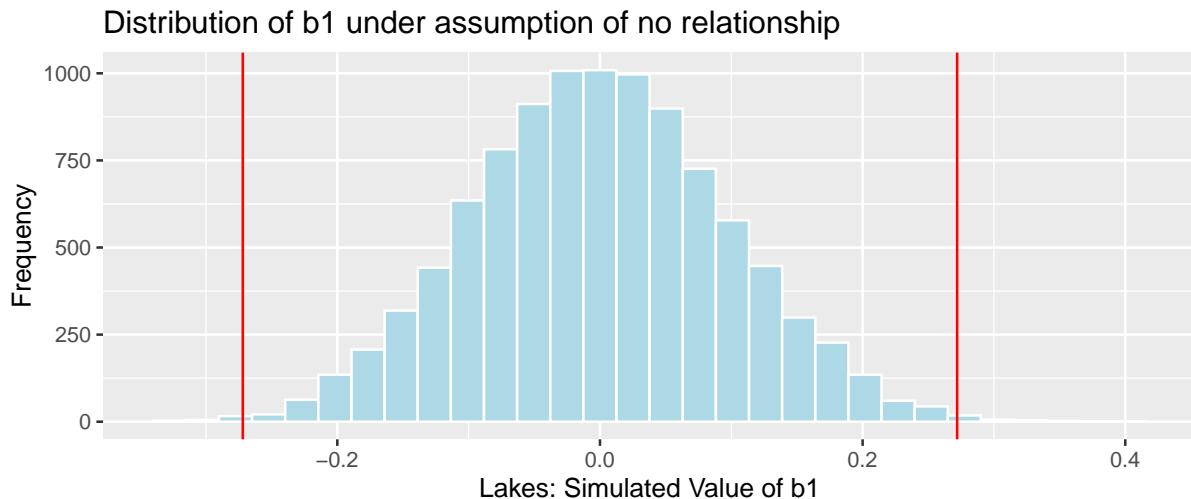
```
[1] 0.003866374
```

The low p-value gives us strong evidence of a difference in average mercury levels between lakes in Northern and Southern Florida.

Comparison to Permutation Test

Recall the permutation test we previously performed.

```
NSLakes_SimulationResultsPlot
```



p-value:

```
b1 <- Lakes_M$coef[2] ## record value of b1 from actual data
mean(abs(NSLakes_SimulationResults$b1Sim) > abs(b1))
```

[1] 0.0039

The p-value from the permutation test is similar to the one from the t-test. Both lead us to the same conclusion.

Hypothesis Test for line (intercept)

Notice that there is also a t-statistic and p-value on the line (intercept). This is testing the following hypotheses.

Null Hypothesis: The average mercury level among all lakes in North Florida is 0 ($\beta_0 = 0$).

Alternative Hypothesis: The average mercury level among all lakes in Northern Florida is not 0 ($\beta_0 \neq 0$).

We could carry out this test by dividing the estimate by its standard error and finding the p-value using a t-distribution, just like we did for β_1 . However, we already know that the average mercury level among all lakes in North Florida is certainly not 0, so this is a silly test, and we shouldn't conclude much from it.

Not every p-value that R provides is actually meaningful or informative.

4.3.3.2 Example 2: Regression Slope

We revisit the model predicting a lake's mercury level, using pH as the explanatory variable.

```
M_pH <- lm(data=FloridaLakes, Mercury~pH)
summary(M_pH)
```

```
Call:
lm(formula = Mercury ~ pH, data = FloridaLakes)

Residuals:
    Min      1Q  Median      3Q     Max 
-0.48895 -0.19188 -0.05774  0.09456  0.71134 

Coefficients:
            Estimate Std. Error t value   Pr(>|t|)    
(Intercept) 1.53092   0.20349   7.523 0.000000000814 ***
pH          -0.15230   0.03031  -5.024 0.000006572811 ***  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2816 on 51 degrees of freedom
Multiple R-squared:  0.3311,    Adjusted R-squared:  0.318 
F-statistic: 25.24 on 1 and 51 DF,  p-value: 0.000006573
```

The estimated regression equation is

$$\text{Mercury} = 1.53 - 0.15 \times \text{pH}, \text{where } \epsilon_i \sim \mathcal{N}(0, \sigma)$$

We can use b_1 to test whether there is evidence of a relationship between mercury concentration and pH level.

Null Hypothesis: There is no relationship between mercury and pH level among all Florida lakes. ($\beta_1 = 0$).

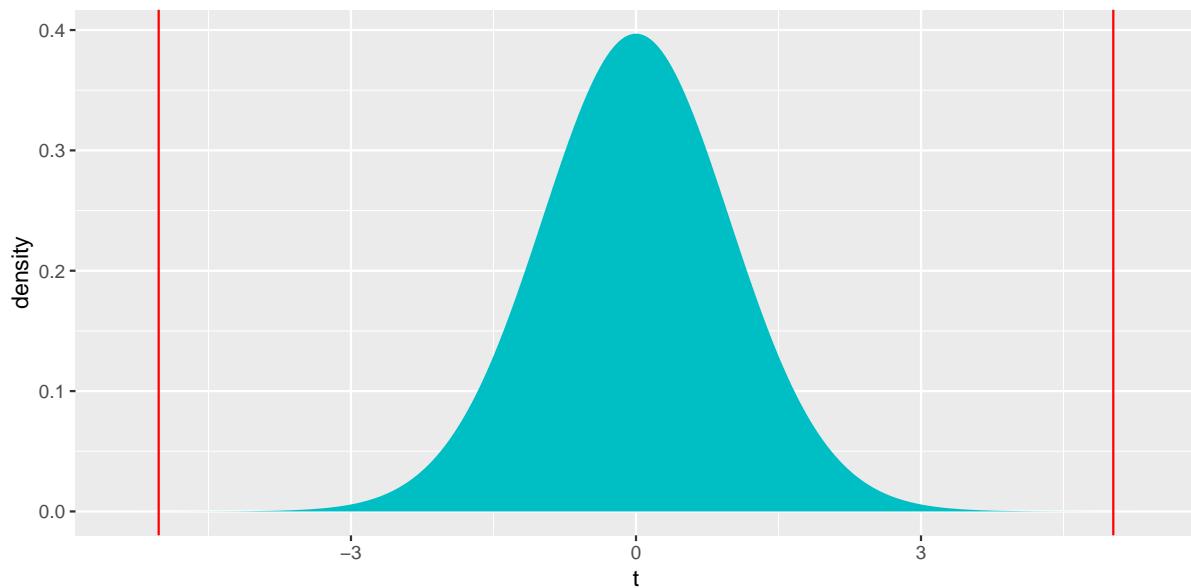
Alternative Hypothesis: There is a relationship between mercury and pH level among all Florida lakes. ($\beta_1 \neq 0$).

In the last section, we calculated $SE(b_1)$ to be 0.0303.

Test Statistic: $t = \frac{b_j}{SE(b_j)} = \frac{-0.15230}{0.03031} = -5.024$

```
ts=5.024
```

```
gf_dist("t", df=51, geom = "area", fill = ~ (abs(x)< abs(ts)), show.legend=FALSE) + geom_vline
```



p-value:

```
2*pt(-abs(ts), df=51)
```

```
[1] 0.000006578117
```

The p-value is extremely small, providing strong evidence of a relationship between pH level and mercury concentration.

Hypothesis Test for Intercept Line

Null Hypothesis: The average mercury level among all Florida lakes with $\text{pH} = 0$ is 0. ($\beta_0 = 0$).

Alternative Hypothesis: The average mercury level among all Florida lakes with $\text{pH} = 0$ is not 0. ($\beta_0 \neq 0$).

Since there are no lakes with pH level 0, this is not a meaningful test.

4.3.4 Limitations of Model-Based Inference

We've seen that in situations where the sampling distribution for a regression coefficient b_j is symmetric and bell-shaped, we can create confidence intervals and perform hypothesis tests using the t-distribution without performing permutation for hypothesis tests, or bootstrapping for confidence intervals.

There are, however, limitations to this approach, which underscore the importance of the simulation-based approaches seen in Chapters 3 and 4.

These include:

1. There are lots of statistics, like medians and standard deviations, that do not have known standard error formulas, and do not follow symmetric bell-shaped distributions. In more advanced and complicated models, it is common to encounter statistics of interest with unknown sampling distributions. In these cases, we can estimate p-values and build confidence intervals via simulation, even if we cannot identify the distribution by name.
2. Even for statistics with known standard error formulas, the t-test is only appropriate when the sampling distribution for b_j is symmetric and bell-shaped. While there is probability theory that shows this will happen when the sample size is "large enough," there is no set sample size that guarantees this. Datasets with heavier skewness in the response variable will require larger sample sizes than datasets with less skewness in the response.
3. The simulation-based approaches provide valuable insight to the logic behind hypothesis tests. When we permute values of an explanatory variable in a hypothesis test it is clear that we are simulating a situation where the null hypothesis is true. Likewise, when we simulate taking many samples in bootstrapping, it is clear that we are assessing the variability in a statistic across samples. Simply jumping to the t-based approximations of these distributions makes it easy to lose our sense of what they actually represent, and thus increases the likelihood of interpreting them incorrectly.

In fact prominent statistician R.A. Fisher wrote of simulation-based methods in 1936:

"Actually, the statistician does not carry out this very simple and very tedious process, but his conclusions have no justification beyond the fact that they agree with those which could have been arrived at by this elementary method."

Fisher's comment emphasizes the fact that probability-based tests, like the t-test are simply approximations to what we would obtain via simulation-based approaches, which were not possible in his day, but are now.

Proponents of simulation-based inference include Tim Hesterberg, Senior Statistician at Instacart, and former Senior Statistician at Google, which heavily used simulation-based tests associated with computer experiments associated with their search settings. Hesterberg wrote a [2015 paper](#), arguing for the use and teaching of simulation-based techniques.

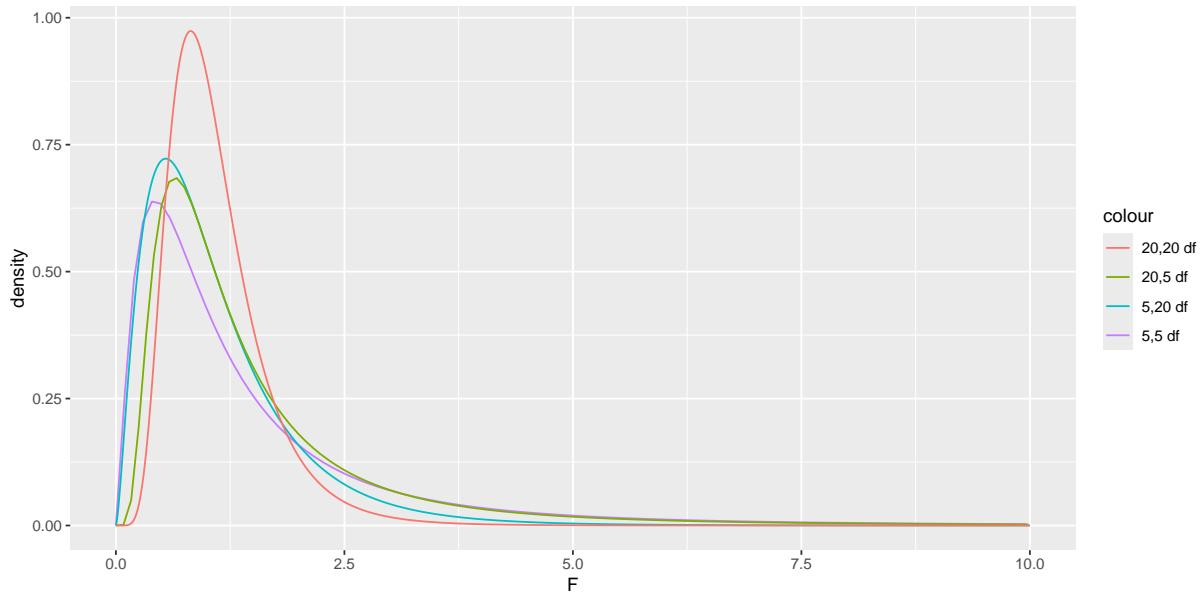
We will move forward by using probability-based inference where appropriate, while understanding that we are merely approximating what we would obtain via simulation. Meanwhile, we'll continue to employ simulation-based approaches where probability-based techniques are inappropriate or unavailable.

4.4 F-tests

Just as we've seen that the ratio of a regression statistic to its standard error follows a t-distribution when can be thought of as having come from a process that can be approximated with the normal error regression model, F-statistics also follow a known probability distribution under this assumption.

4.4.1 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.



Under certain conditions (which we'll examine in the next chapter), the F-statistic

$$F = \frac{\frac{\text{Unexplained Variability in Reduced Model} - \text{Unexplained Variability in Full Model}}{p-q}}{\frac{\text{Unexplained Variability in Full Model}}{n-(p+1)}}$$

follows an F-distribution.

The numerator and denominator degrees of freedom are given by $p - q$ and $n - (p + 1)$, respectively. These are the same values we divided by when computing the F-statistic.

4.4.2 House Condition Example

Recall the F-statistic for comparing prices of houses in either very good, good, or average condition, and the p-value associated with the simulation-based F-test we performed previously.

Null Hypothesis: There is no difference in average prices between houses of the three different conditions, among all houses in King County, WA.

Alternative Hypothesis: There is a difference in average prices between houses of the three different conditions, among all houses in King County, WA.

Reduced Model: Price = $\beta_0 + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \sigma)$

Full Model: Price = $\beta_0 + \beta_1 \times \text{good condition} + \beta_2 \times \text{very good condition} + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \sigma)$

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

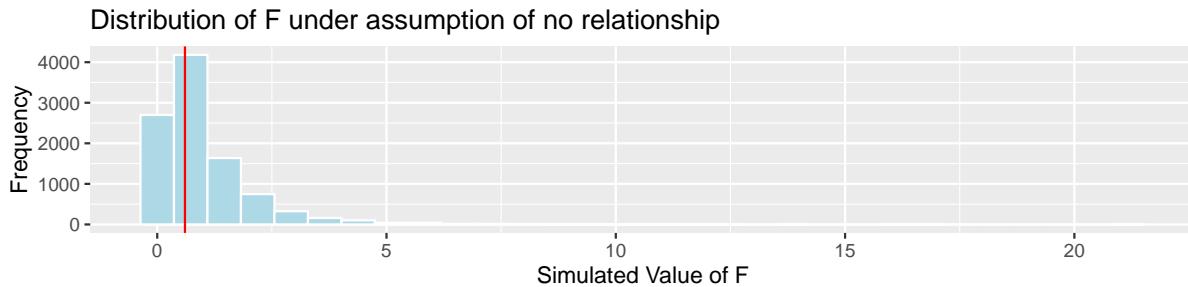
$$= \frac{\frac{69,045,634 - 68,195,387}{2-0}}{\frac{68,195,387}{100-(2+1)}}$$

```
((SST - SSR_cond)/(2-0))/(SSR_cond/(100-(2+1)))
```

```
[1] 0.6046888
```

The results of the simulation-based hypothesis test are shown below.

```
House_Cond_SimulationResults_Plot
```



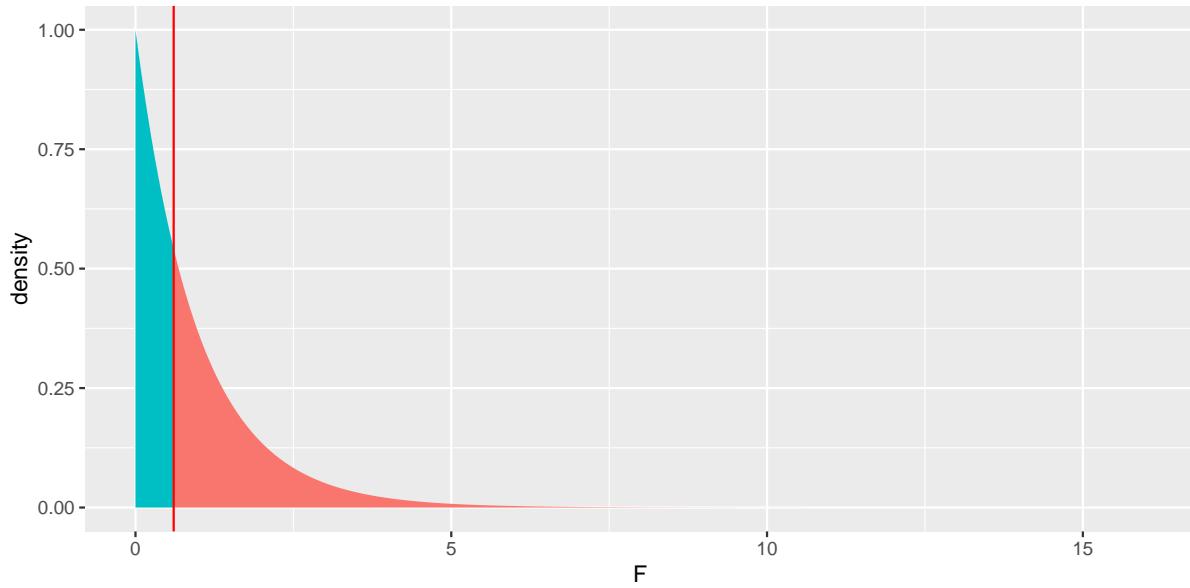
simulation-based p-value:

```
mean(FSim > Fstat)
```

```
[1] 0.5548
```

Now, we calculate the p-value using the probability-based F-distribution.

```
ts=0.605  
gf_dist("f", df1=2, df2=97, geom = "area", fill = ~ (abs(x)< abs(ts)), show.legend=FALSE) +
```



p-value:

```
1-pf(ts, df1=2, df2=97)
```

```
[1] 0.5481219
```

The p-value we obtained is very similar to the one we obtained using the simulation-based test.

We can obtain this p-value directly using the `anova` command.

```
M_cond <- lm(data=Houses, price ~ condition)  
M0 <- lm(data=Houses, price ~ 1)  
anova(M0, M_cond)
```

Analysis of Variance Table

```

Model 1: price ~ 1
Model 2: price ~ condition
  Res.Df    RSS Df Sum of Sq    F Pr(>F)
1     99 69045634
2     97 68195387  2     850247 0.6047 0.5483

```

Since the p-value is large, we do not have enough evidence to say that average price differs between conditions of the houses.

4.4.3 Interaction Example

We can also use an F-test to compare a model predicting house prices with an interaction term to one without one.

Reduced Model: Price = $\beta_0 + \beta_1 \times \text{sqft_living} + \beta_2 \times \text{Waterfront} + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \sigma)$

Full Model: Price = $\beta_0 + \beta_1 \times \text{sqft_living} + \beta_2 \times \text{Waterfront} + \beta_3 \times \text{sqft_living} \times \text{Waterfront} + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, \sigma)$

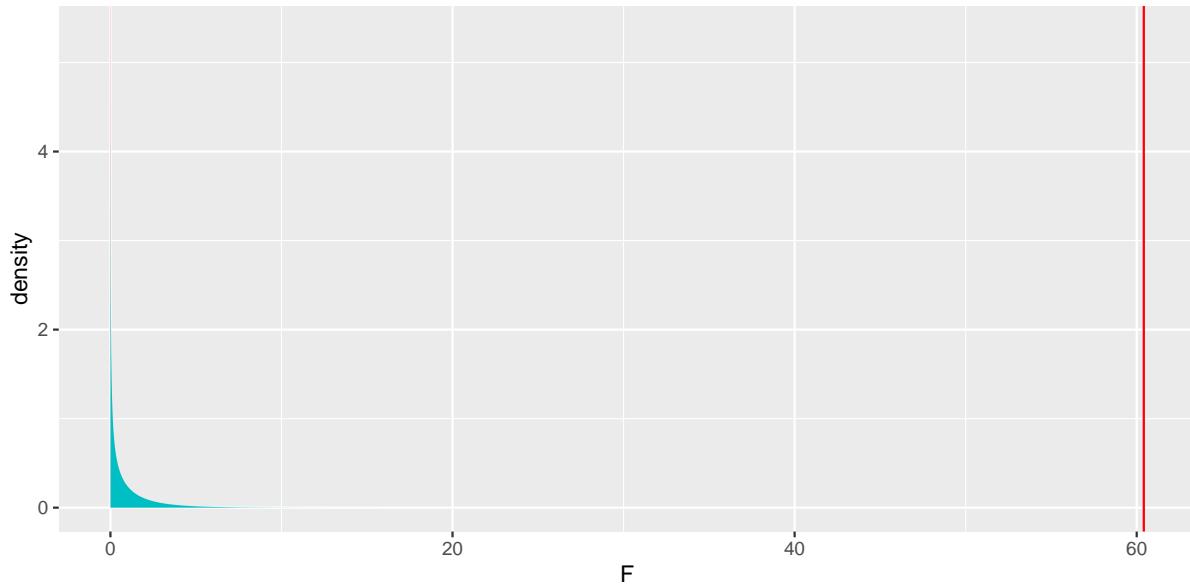
$$F = \frac{\frac{\text{SSR}_{\text{Reduced}} - \text{SSR}_{\text{Full}}}{p-q}}{\frac{\text{SSR}_{\text{Full}}}{n-(p+1)}}$$

$$= \frac{\frac{16,521,296 - 10,139,974}{3-2}}{\frac{10,139,974}{100-(3+1)}}$$

```
((SSR_wf_sqft-SSR_int)/(3-2))/((SSR_int)/(100-(3+1)))
```

```
[1] 60.41505
```

```
ts=60.41505
gf_dist("f", df1=1, df2=96, geom = "area", fill = ~ (abs(x)> abs(ts)), show.legend=FALSE) +
```



p-value:

```
1-pf(ts, df1=1, df2=96)
```

```
[1] 0.000000000008572476
```

The probability-based F-test is used in the `anova` command.

```
M_wf_SqFt <- lm(data=Houses, price~sqft_living + waterfront)
M_House_Int <- lm(data=Houses, price~sqft_living * waterfront)
anova(M_wf_SqFt, M_House_Int)
```

Analysis of Variance Table

Model	price ~ sqft_living + waterfront	price ~ sqft_living * waterfront									
Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)						
1	97	16521296									
2	96	10139974	1	6381323	60.415 0.000000000008572 ***						

Signif. codes:	0	'***'	0.001	'**'	0.01	'*'	0.05	'. '	0.1	' '	1

Since the p-value is small, we have enough evidence to say that there is evidence of an interaction between size and whether or not the house is on the waterfront.

Notice that this p-value is identical to the one we obtained in the previous section, using the t-test for interaction associated with the `lm` command.

4.5 `lm` Summary in R

We've now seen how to obtain all of the quantities shown in the `lm` summary output in R. This section does not provide any new information, but brings together all of the information from the previous sections and chapters to show, in one place, how the `lm` summary output is obtained.

4.5.1 `lm` summary Output

The `summary` command for a linear model in R displays a table including 4 columns.

Linear Model `summary()` Output in R

- **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. It tells us how the estimate is expected to vary between different samples of the given size. These are computed using the formulas in Section 4.2.
- **t value** is the estimate divided by its standard error.
- **Pr(>|t|)** is a p-value for the hypothesis test associated with the null hypothesis $\beta_j = 0$, where β_j is the regression coefficient pertaining to the given line. Note that β_j is the unknown population parameter estimated by b_j .

- The **Residual Standard Error** is $s = \sqrt{\frac{SSR}{n-(p+1)}} = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{(n-(p+1))}}$. This is an estimate of σ , which represents the standard deviation in the distribution of the response variable for given value(s) or category(ies) of explanatory variable(s). It tells us how much variability is expected in the response variable between different individuals with the same values/categories of the explanatory variables.
- The **degrees of freedom** are $n - (p + 1)$.
- The **Multiple R-Squared** value is the R^2 value seen in Chapter 2. $R^2 = \frac{SST-SSR}{SST} = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y}_i)^2}$

- We know that R^2 can never decrease when additional variables are added to a model. The **Adjusted-R²** value is an alternate version of R^2 that is designed to penalize adding variables that do little to explain variation in the response.
- The F-statistic on the bottom line of the R-output corresponds to an F-test of the given model against a reduced model that include no explanatory variables. The p-value on this line is associated with the test of the null hypothesis that there is no relationship between the response variable and any of the explanatory variables. Since SSR for this reduced model is equal to SST, the F-statistic calculation simplifies to:

$$F = \frac{\frac{SST - SSR}{p}}{\frac{SSR}{n-(p+1)}}$$

The degrees of freedom associated with the F-statistic are given by p and $(n - (p + 1))$.

Example: Northern vs Southern Florida Lakes

Recall our linear model for mercury levels of lakes in Northern Florida, compared to Southern Florida.

The equation of the model is:

$$\text{Mercury} = \beta_0 + \beta_1 \times \text{South} + \epsilon_i, \text{ where } \epsilon_i \sim \mathcal{N}(0, \sigma)$$

We fit the model in R and display its summary output below.

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

```
Call:
lm(formula = Mercury ~ 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 estimated regression equation is

$$\text{Mercury} = 0.42455 + 0.27195 \times \text{South}, \text{ where } \epsilon_i \sim \mathcal{N}(0, \sigma)$$

- SSR is:

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

```
[1] 5.126873
```

- SST is:

```
sum((FloridaLakes$Mercury - mean(FloridaLakes$Mercury))^2)
```

```
[1] 6.047875
```

- The residual standard error s is our estimate of the standard deviation among lakes in the same location (either Northern or Southern Florida).

$$s = \sqrt{\frac{\text{SSR}}{n - (p + 1)}} = \sqrt{\frac{\text{SSR}}{n - (p + 1)}} = \sqrt{\frac{5.126873}{53 - (1 + 1)}} = 0.3171$$

The degrees of freedom associated with this estimate is $53 - (1 + 1) = 51$.

- The Multiple R-Squared is:

$$R^2 = \frac{6.047875 - 5.126873}{6.047875} = 0.1523$$

- The F-statistic is

$$F = \frac{\frac{SST-SSR}{p}}{\frac{SSR}{n-(p+1)}} = \frac{\frac{6.047875-5.126873}{1}}{\frac{5.126873}{53-(1+1)}} = 9.162$$

This F-statistic is associated with 1 and 51 degrees of freedom.

Using formulas in Section 4.2, we obtain the standard error estimates for b_0 and b_1 , given in the second column of the table.

$$SE(b_0) = SE(\bar{x}_N) = s \frac{1}{\sqrt{n_{North}}} = \frac{0.3171}{\sqrt{33}} = 0.0552$$

$SE(b_0)$ represents the variability in average mercury levels between different samples of 33 Northern Florida lakes.

$$SE(b_1) = SE(\bar{x}_South - \bar{x}_{North}) = s \sqrt{\frac{1}{n_{North}} + \frac{1}{n_{South}}} = 0.3171 \sqrt{\frac{1}{20} + \frac{1}{33}} = 0.0898$$

$SE(b_1)$ represents the variability in average difference in mercury levels between northern and southern lakes between different samples of 33 Northern Florida lakes and 20 Southern Florida lakes.

The last column, labeled “Pr(>|t|)” is, in fact a p-value associated with associated with the null hypothesis that the regression parameter on that line is zero. (i.e. $\beta_j = 0$).

Hypothesis Test for line (intercept)

Null Hypothesis: The average mercury level among all lakes in North Florida is 0 ($\beta_0 = 0$).

Alternative Hypothesis: The average mercury level among all lakes in Northern Florida is not 0 ($\beta_0 \neq 0$).

We already know the average mercury level among all lakes in North Florida is not 0, so this is a silly test.

Hypothesis Test for line Locations

Null Hypothesis: There is no difference in average mercury levels between Northern and Southern Florida ($\beta_1 = 0$).

Alternative Hypothesis: There is a difference in average mercury levels in Northern and Southern Florida ($\beta_1 \neq 0$).

4.5.2 Difference in Means Example

Recall the hypothesis test we performed to investigate whether there is a difference in average mercury level between lakes in Northern Florida and Southern Florida.

Null Hypothesis: There is no difference in average mercury levels between Northern and Southern Florida ($\beta_1 = 0$).

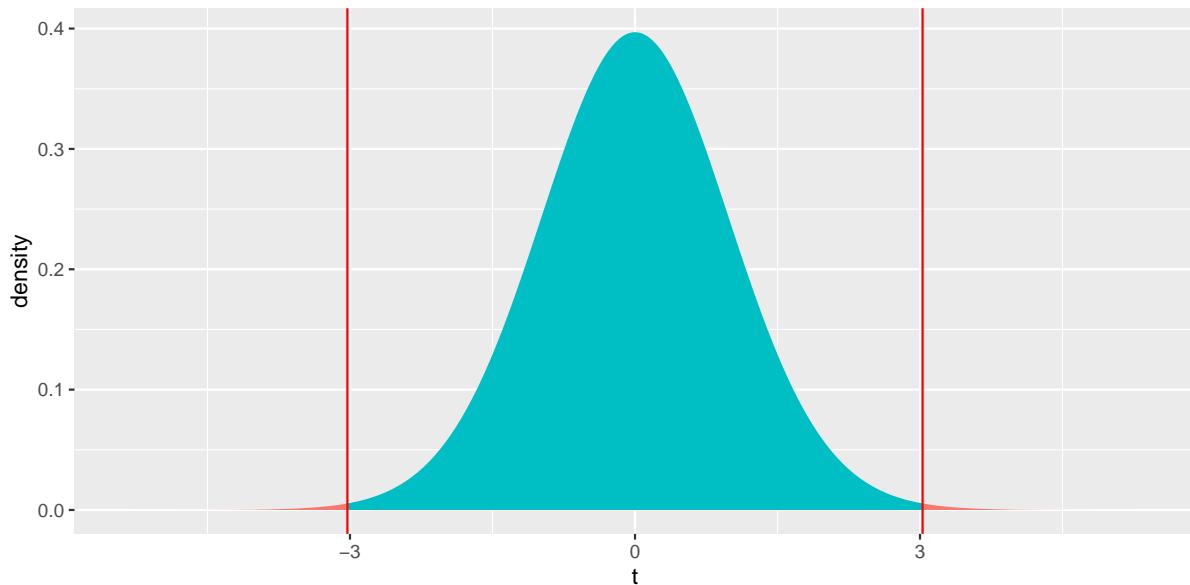
Alternative Hypothesis: There is a difference in average mercury levels in Northern and Southern Florida ($\beta_1 \neq 0$).

Test Statistic: $t = \frac{b_j}{\text{SE}(b_j)} = \frac{0.27195}{0.08985} = 3.027$

Key Question: What is the probability of getting a t-statistic as extreme as 3.027 if $\beta_1 = 0$ (i.e. there is no difference in mercury levels between northern and southern lakes).

We plot the t-statistic of 3.027 that we observed in our data and observe where it lies on a t-distribution.

```
ts=3.027  
gf_dist("t", df=51, geom = "area", fill = ~ (abs(x)< abs(ts)), show.legend=FALSE) + geom_vline
```



```
2*pt(-abs(ts), df=51)
```

```
[1] 0.003866374
```

The low p-value gives us strong evidence of a difference in average mercury levels between lakes in Northern and Southern Florida.

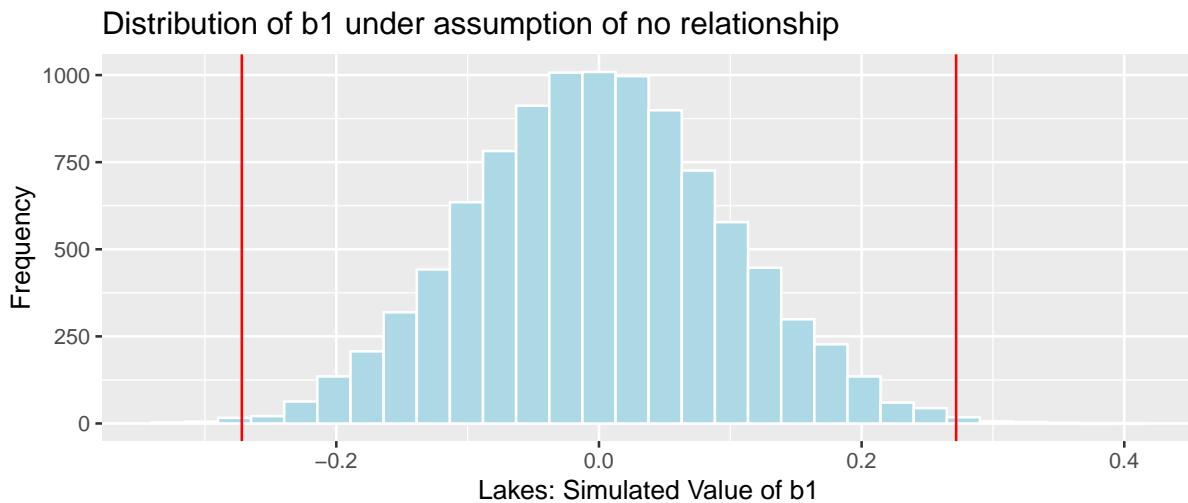
This is the p-value reported in R's `lm` `summary()` output.

A t-statistic more extreme than ± 2 will roughly correspond to a p-value less than 0.05.

Permutation Test

Let's compare these results to those given by the permutation test.

```
NSLakes_SimulationResultsPlot
```



p-value:

```
b1 <- Lakes_M$coef[2] ## record value of  $b_1$  from actual data  
mean(abs(NSLakes_SimulationResults$b1Sim) > abs(b1))
```

```
[1] 0.0039
```

4.5.3 Simple Linear Regression Example

We examine the model summary output for the model predicting a lake's mercury level, using pH as the explanatory variable.

```
M_pH <- lm(data=FloridaLakes, Mercury~pH)  
summary(M_pH)
```

```

Call:
lm(formula = Mercury ~ pH, data = FloridaLakes)

Residuals:
    Min      1Q  Median      3Q     Max 
-0.48895 -0.19188 -0.05774  0.09456  0.71134 

Coefficients:
            Estimate Std. Error t value    Pr(>|t|)    
(Intercept) 1.53092   0.20349   7.523 0.000000000814 ***
pH          -0.15230   0.03031  -5.024 0.000006572811 ***  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2816 on 51 degrees of freedom
Multiple R-squared:  0.3311,    Adjusted R-squared:  0.318 
F-statistic: 25.24 on 1 and 51 DF,  p-value: 0.000006573

```

The estimated regression equation is

$$\text{Mercury} = 1.53 - 0.15 \times \text{pH}, \text{ where } \epsilon_i \sim \mathcal{N}(0, \sigma)$$

- SSR is:

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

[1] 4.045513

- SST is:

```
sum((FloridaLakes$Mercury - mean(FloridaLakes$Mercury))^2)
```

[1] 6.047875

- The residual standard error s is our estimate of σ , the standard deviation among lakes with the same pH.

$$s = \sqrt{\frac{\text{SSR}}{n - (p + 1)}} = \sqrt{\frac{\text{SSR}}{n - (p + 1)}} = \sqrt{\frac{4.045513}{53 - (1 + 1)}} = 0.2816$$

The degrees of freedom associated with this estimate is $53 - (1 + 1) = 51$.

- The Multiple R-Squared is:

$$R^2 = \frac{6.047875 - 4.045513}{6.047875} = 0.3311$$

- The F-statistic is

$$F = \frac{\frac{SST-SSR}{p}}{\frac{SSR}{n-(p+1)}} = \frac{\frac{6.047875-4.045513}{1}}{\frac{4.045513}{53-(1+1)}} = 25.24$$

This F-statistic is associated with 1 and 51 degrees of freedom.

Using formulas in Section 4.2, we obtain the standard error estimates for b_0 and b_1 , given in the second column of the table.

To do this, we need to calculate \bar{x} and $\sum(x_i - \bar{x})^2$, where x represents the explanatory variable, pH .

```
mean(FloridaLakes$pH)
```

```
[1] 6.590566
```

```
sum((FloridaLakes$pH-mean(FloridaLakes$pH))^2)
```

```
[1] 86.32528
```

$$SE(b_0) = s \sqrt{\frac{1}{n} + \frac{\bar{x}^2}{\sum(x_i - \bar{x})^2}} = 0.2816 \sqrt{\frac{1}{53} + \frac{6.59^2}{86.32528}} = 0.2034$$

$SE(b_0)$ represents the variability in mercury levels among lakes with pH of 0 between different samples of size 53. Since we don't have any lakes with pH of 0, this is not a meaningful calculation.

$$SE(b_1) = \sqrt{\frac{s^2}{\sum(x_i - \bar{x})^2}} = \sqrt{\frac{0.2816^2}{86.32528}} = 0.0303$$

$SE(b_1)$ represents the variability in rate of change in mercury level for each additional one unit increase in pH, between different samples of size 53.

Hypothesis Test for Intercept Line

Null Hypothesis: The average mercury level among all Florida lakes with pH = 0 is 0. ($\beta_0 = 0$).

Alternative Hypothesis: The average mercury level among all Florida lakes with pH = 0 not 0. ($\beta_0 \neq 0$).

Since there are no lakes with pH level 0, this is not a meaningful test.

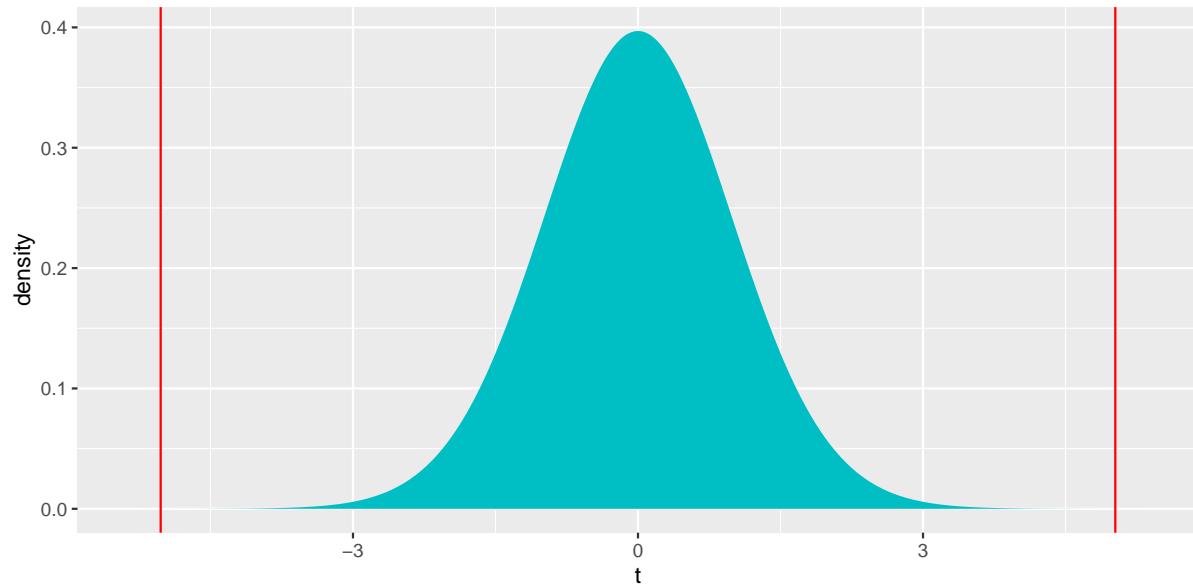
Hypothesis Test for pH Line

Null Hypothesis: There is no relationship between mercury and pH level among all Florida lakes. ($\beta_1 = 0$).

Alternative Hypothesis: There is a relationship between mercury and pH level among all Florida lakes. ($\beta_1 \neq 0$).

Test Statistic: $t = \frac{b_j}{\text{SE}(b_j)} = \frac{-0.15230}{0.03031} = -5.024$

```
ts=5.024
gf_dist("t", df=51, geom = "area", fill = ~ (abs(x)< abs(ts)), show.legend=FALSE) + geom_vline
```



```
2*pt(-abs(ts), df=51)
```

```
[1] 0.000006578117
```

The p-value is extremely small, just as the simulation-based p-value we saw in Chapter 3.

4.5.4 Multiple Regression Example

We perform hypothesis tests on a model predicting house price using square feet and waterfront status as explanatory variables.

```
M_wf_sqft <- lm(data=Houses, price~sqft_living+waterfront)
summary(M_wf_sqft)
```

Call:

```
lm(formula = price ~ sqft_living + waterfront, data = Houses)
```

Residuals:

Min	1Q	Median	3Q	Max
-1363.79	-251.55	59.28	177.58	1599.72

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-407.6549	86.2868	-4.724	0.00000779668 ***
sqft_living	0.4457	0.0353	12.626	< 0.0000000000000002 ***
waterfrontYes	814.3613	124.8546	6.522	0.00000000313 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 412.7 on 97 degrees of freedom

Multiple R-squared: 0.7607, Adjusted R-squared: 0.7558

F-statistic: 154.2 on 2 and 97 DF, p-value: < 0.0000000000000022

We won't go through the standard error calculations here, though the details are given in the link provided in Section 4.2.

Intercept Line:

Null Hypothesis The average price among all non-waterfront houses with 0 square feet is 0 dollars. ($\beta_0 = 0$)

This is not a sensible hypothesis to test.

sqft_living Line:

Null Hypothesis There is no relationship between price and square feet in a house, after accounting for waterfront status. ($\beta_1 = 0$)

The large t-statistic (12.626) and small p-value provide strong evidence against this null hypothesis.

We know that a small p-value alone does not provide evidence of a relationship that is practically meaningful, but since our model estimates an expected 45 thousand dollar increase for each additional 100 square feet, this seems like a meaningful relationship.

waterfrontYes Line:

Null Hypothesis On average, there is no difference between average price of waterfront and non-waterfront houses, assuming they are the same size. ($\beta_2 = 0$)

The large t-statistic (6.522) and small p-value provide strong evidence against this null hypothesis. Waterfront houses are estimated to cost 814 thousand dollars more, on average, than non-waterfront houses of the same size.

4.5.5 MR with Interaction Example

```
M_House_Int <- lm(data=Houses, price ~ sqft_living * waterfront)
summary(M_House_Int)
```

```
Call:
lm(formula = price ~ sqft_living * waterfront, data = Houses)

Residuals:
    Min      1Q  Median      3Q     Max 
-1559.34 -114.93 -30.24  131.09 1266.58 

Coefficients:
            Estimate Std. Error t value    Pr(>|t|)    
(Intercept) 67.39594   91.39267   0.737    0.4627    
sqft_living  0.21837   0.04035   5.412 0.00000045752269 ***
waterfrontYes -364.59498  180.75875  -2.017    0.0465 *  
sqft_living:waterfrontYes  0.43267   0.05566   7.773 0.00000000000857 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 325 on 96 degrees of freedom
Multiple R-squared:  0.8531,    Adjusted R-squared:  0.8486 
F-statistic: 185.9 on 3 and 96 DF,  p-value: < 0.0000000000000022
```

Intercept Line:

Null Hypothesis The average price among all non-waterfront houses with 0 square feet is 0 dollars. ($\beta_0 = 0$)

This is not a sensible hypothesis to test.

sqft_living Line:

Null Hypothesis There is no relationship between price and square feet among non-waterfront houses. ($\beta_1 = 0$)

The large t-statistic (5.412) and small p-value provide strong evidence against this null hypothesis.

waterfrontYes Line:

Null Hypothesis On average, there is no difference between average price of waterfront and non-waterfront houses with 0 square feet. ($\beta_2 = 0$)

This is not a sensible hypothesis to test.

sqft_living:waterfrontYes

Null Hypothesis: There is no interaction between square feet and waterfront. ($\beta_3 = 0$)
(That is, the effect of size on price is the same for waterfront and non-waterfront houses).

The large t-statistic (7.773) and small p-value provide strong evidence against this null hypothesis. It appears there really is evidence of an interaction between price and waterfront status, as we previously suspected, based on graphical representation and background knowledge.

Note that if the interaction term had yielded a large p-value, indicating a lack of evidence of an interaction, we might have wanted to drop the interaction term from the model, in order to make the interpretations of the other estimates and hypothesis tests simpler.

4.6 Intervals for Expected Response

4.6.1 Parameter Values and Expected Responses

Recall that in Chapter 4, we saw two different types of confidence intervals. One type was for regression parameters, $\beta_0, \beta_1, \dots, \beta_p$, using estimate b_0, b_1, \dots, b_p . The other type was for expected responses, which involved estimating linear functions of these parameters, for example $\beta_0 + 7\beta_1$.

Under the assumptions of a normal error regression model, an approximate 95% confidence interval for regression parameter β_j is given by

$$b_j + \pm t^* \text{SE}(b_j),$$

where $t^* \approx 2$.

We've seen that in R, confidence intervals for regression parameters can be obtained through the `confint()` command.

A 95% confidence interval for an expected response $E(Y_i|X_{i1} = x_{i1}, \dots, X_{ip} = x_{ip}) = \beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}$ is estimated by

$$b_0 + b_1 x_{i1} + \dots + b_p x_{ip} \pm t^* \text{SE}(b_0 + b_1 x_{i1} + \dots + b_p x_{ip}),$$

In this section, we'll look more into confidence intervals for expected responses and also another kind of interval involving expected responses, called a prediction interval.

Well sometimes write $E(Y_i|X_{i1} = x_{i1}, \dots, X_{ip} = x_{ip})$ as $E(Y|X)$.

4.6.2 Estimation and Prediction

Recall the ice cream dispenser that is known to dispense ice cream 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 ice cream 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 ice cream will be dispensed?

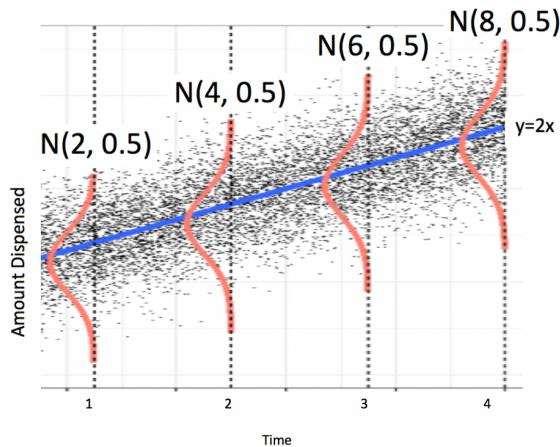
The first question is one of estimation. The second pertains to prediction.

When estimating expected responses and making predictions on new observations, there are two sources of variability we must consider.

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 $E(Y|X)$.



Modified from image at <https://bookdown.org/roback/bookdown-bysh/ch-poissonreg.html>

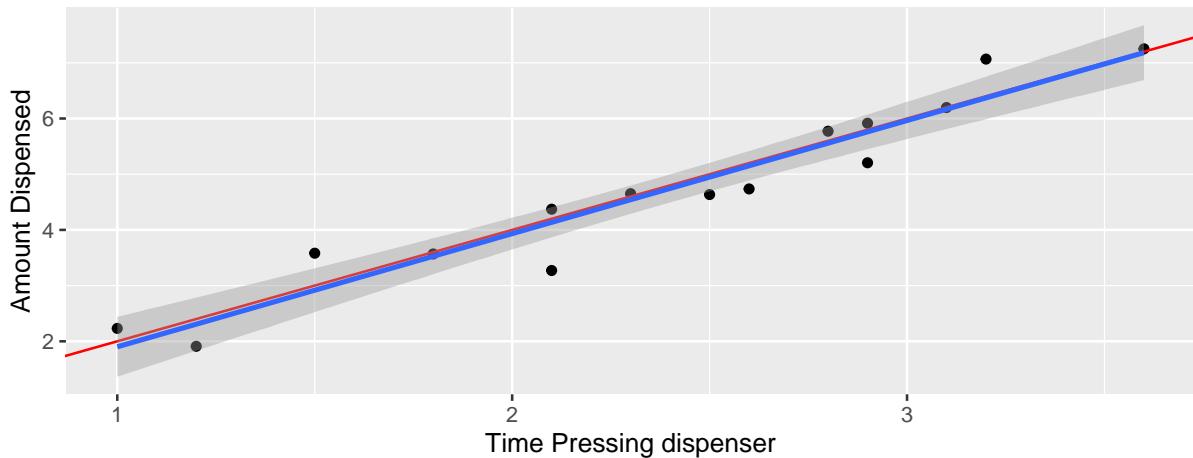
4.6.2.1 Example: Ice Cream Machine

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.

In the ice cream example, we can see that the blue line, fit to our data, is a good approximation of the “true” regression line that pertains to the mechanism from which the data were generated. It does, however, vary from the red line slightly due to sampling variability.

```
ggplot(data=Icecream1, aes(x=time, y=amount)) + geom_point() + ggtitle("Icecream Dispensed")
```

Icecream Dispensed



```
summary(IC_Model)
```

Call:

```
lm(formula = 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

```
b0 <- IC_Model$coefficients[1]
b1 <- IC_Model$coefficients[2]
s <- sigma(IC_Model)
```

The first question:

“On average, how much ice cream will be dispensed for people who press the dispenser 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 ice cream 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.

Confidence interval for $E(Y|(X = x))$:

$$\begin{aligned} b_0 + b_1 x^* &\pm t^* SE(\hat{Y}|X = x^*) \\ b_0 + b_1 x^* &\pm t^* \sqrt{\widehat{Var}(\hat{Y}|X = x^*)} \end{aligned}$$

The second question is a question of prediction. Even if we knew the true values of β_0 and β_1 , we would not be able to give the exact amount dispensed for an individual user, since this varies between users.

Prediction interval for $E(Y|(X = x))$:

$$b_0 + b_1 x^* \pm t^* \sqrt{\widehat{Var}(\hat{Y}|X = x^*) + s^2}$$

The extra s^2 in the calculation of prediction variance comes from the uncertainty associated with individual observations.

4.6.3 Intervals in R

In R, we can obtain confidence intervals for an expected response and prediction intervals for an individual response using the `predict` command, with either `interval="confidence"` or `interval="prediction"`.

```
predict(IC_Model, newdata=data.frame(time=1.5), interval = "confidence", level=0.95)
```

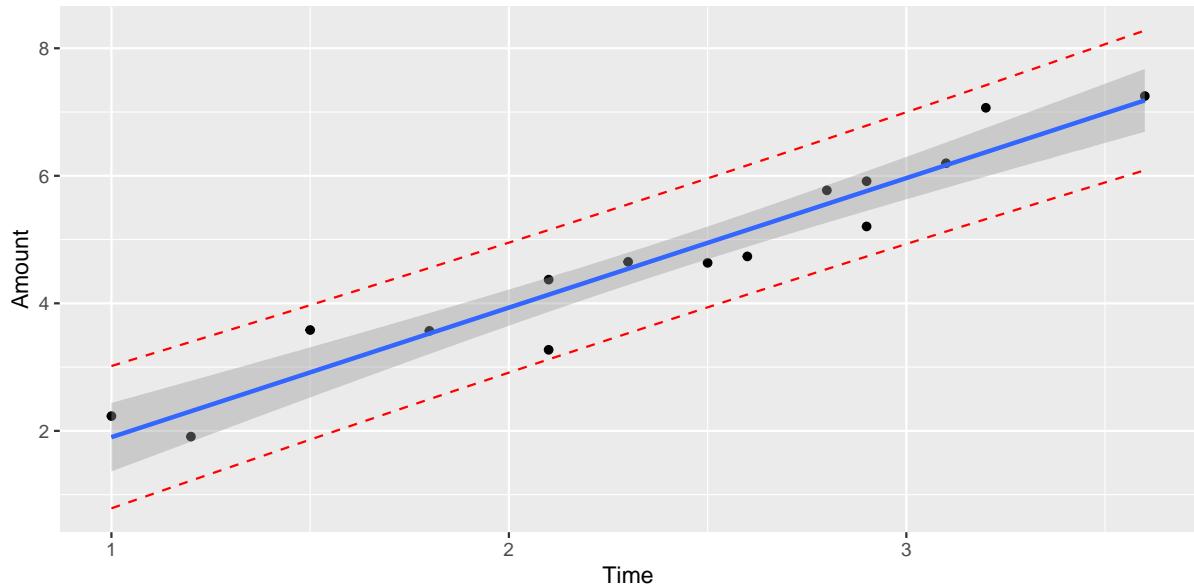
	fit	lwr	upr
1	2.916965	2.523728	3.310201

We are 95% confident that the mean amount of ice cream dispensed when the dispenser is held for 1.5 seconds is between 2.52 and 3.31 oz.

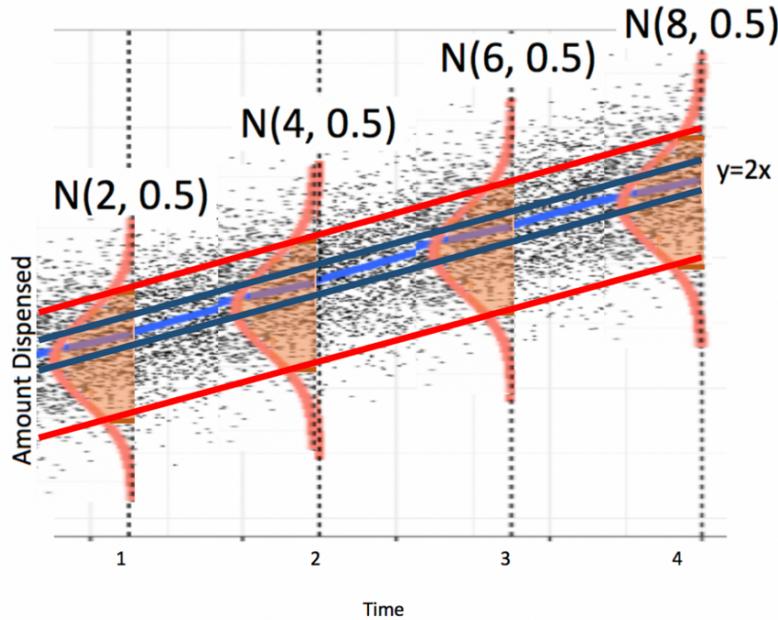
```
predict(IC_Model, newdata=data.frame(time=1.5), interval = "prediction", level=0.95)
```

	fit	lwr	upr
1	2.916965	1.862832	3.971097

We are 95% confident that individual who holds the dispenser for 1.5 seconds will get between 1.86 and 3.97 oz of ice cream.



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.



Modified from image at <https://bookdown.org/roback/bookdown-bysh/ch-poissonreg.html>

4.6.4 SLR Calculations (Optional)

In simple linear regression,

$$SE(\hat{Y}|X = x^*) = \sqrt{\frac{1}{n} + \frac{(x^* - \bar{x})^2}{\sum_{i=1}^n (x_i - \bar{x})^2}}$$

Thus a confidence interval for $E(Y|(X = x))$ is:

$$\begin{aligned} b_0 + b_1 x^* \pm t^* SE(\hat{Y}|X = x^*) \\ = b_0 + b_1 x^* \pm t^* s \sqrt{\frac{1}{n} + \frac{(x^* - \bar{x})^2}{\sum_{i=1}^n (x_i - \bar{x})^2}} \end{aligned}$$

A prediction interval for $E(Y|(X = x))$ is:

$$\beta_0 + \beta_1 x^* \pm t^* s \sqrt{\left(\frac{1}{n} + \frac{(x^* - \bar{x})^2}{\sum_{i=1}^n (x_i - \bar{x})^2} \right) + 1}$$

Calculations in Ice cream example

For $x = 1.5$, a confidence interval is:

$$\begin{aligned} b_0 + b_1 x^* \pm t^* SE(\hat{Y}|X = x^*) \\ = b_0 + b_1 x^* \pm 2s \sqrt{\frac{1}{n} + \frac{(x^* - \bar{x})^2}{\sum_{i=1}^n (x_i - \bar{x})^2}} \\ = -0.1299087 + 2.0312489 \pm 20.4527185 \sqrt{\frac{1}{15} + \frac{(1.5 - 2.3733)^2}{8.02933}} \end{aligned}$$

A prediction interval is:

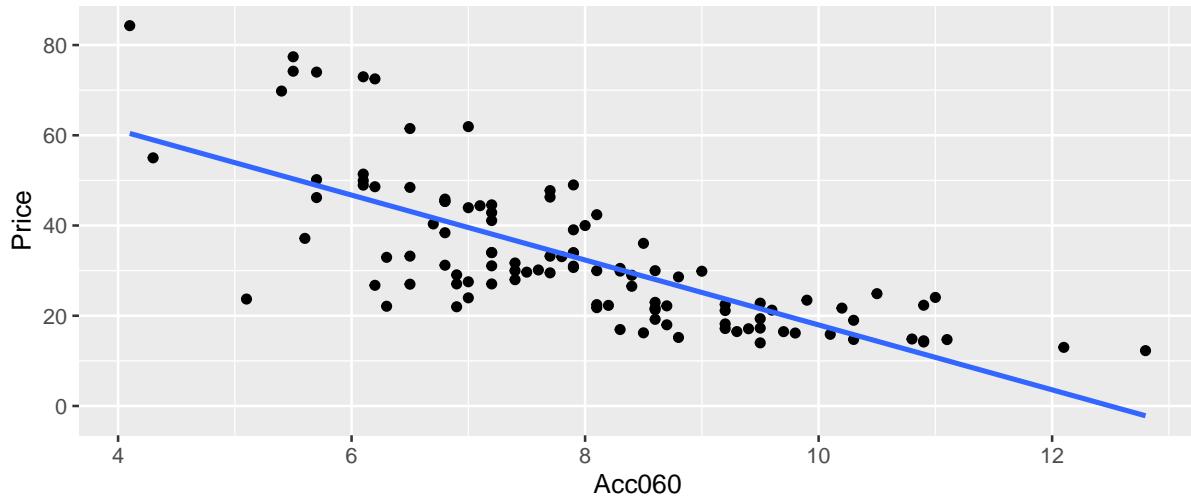
$$\begin{aligned} b_0 + b_1 x^* \pm t^* SE(\hat{Y}|X = x^*) \\ = b_0 + b_1 x^* \pm 2s \sqrt{\frac{1}{n} + \frac{(x^* - \bar{x})^2}{\sum_{i=1}^n (x_i - \bar{x})^2}} \\ = -0.1299087 + 2.0312489 \pm 20.4527185 \sqrt{\left(\frac{1}{15} + \frac{(1.5 - 2.3733)^2}{8.02933} \right) + 1} \end{aligned}$$

4.6.5 Car Price and Acceleration Time

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.

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

```
CarsA060 <- ggplot(data=Cars2015, aes(x=Acc060, y=Price)) + geom_point()
CarsA060 + stat_smooth(method="lm", se=FALSE)
```



$$Price = \beta_0 + \beta_1 \times \text{Acc. Time}, \text{ where } \epsilon_i \sim \mathcal{N}(0, \sigma)$$

The model assumes expected price is a linear function of acceleration time.

Parameter Interpretations:

β_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.

β_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.

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

Call:

```
lm(formula = Price ~ 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: < 0.00000000000000022

```

** 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.
 - $b_1 = -7.1933$ 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. 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?

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.

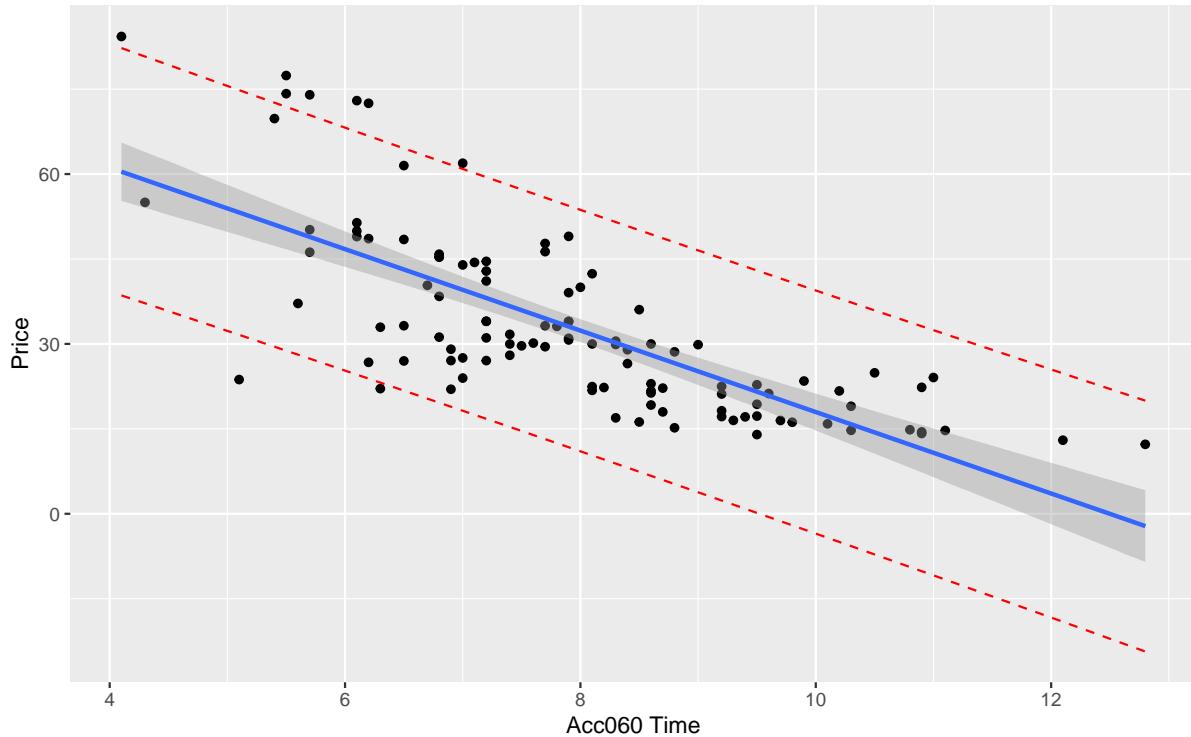
Note: this is a confidence interval for $\beta_0 - 7\beta_1$.

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.



4.6.6 Florida Lakes Est. and Pred.

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.

```
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.

```
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.

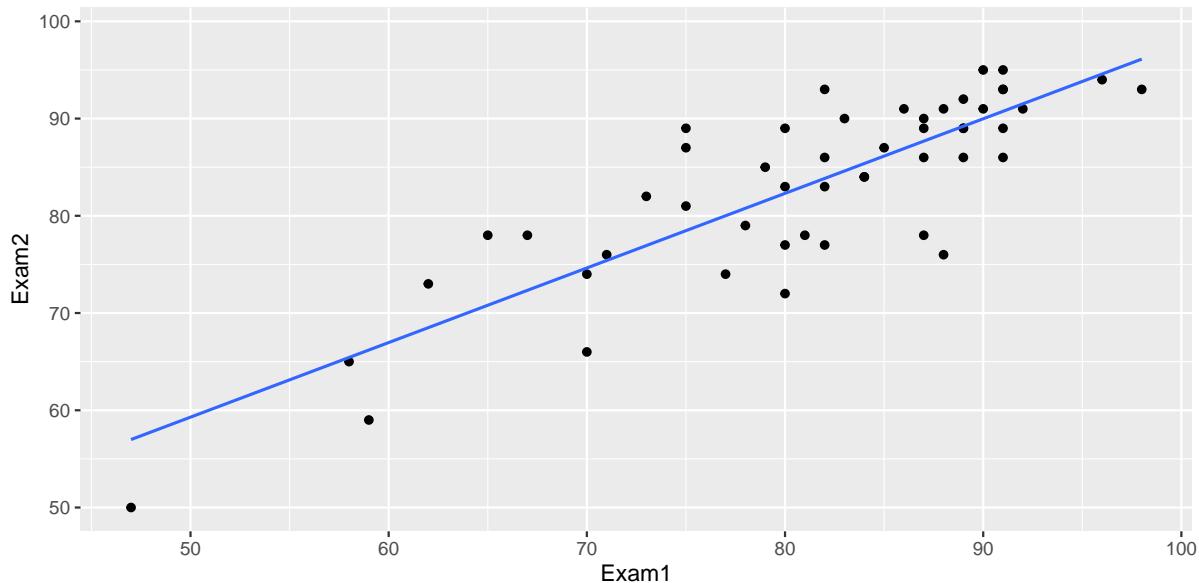
Note that the normality assumption, which allows for negative mercury levels leads to a somewhat nonsensical result.

4.7 The Regression Effect

4.7.1 The Regression Effect

You might be wondering how regression gets its name. It comes from a well known phenomenon, known as “regression to the mean”, or the “regression effect”. While the word “regression” is often construed with a negative context (i.e. getting worse), it could also refer to movement in the positive direction.

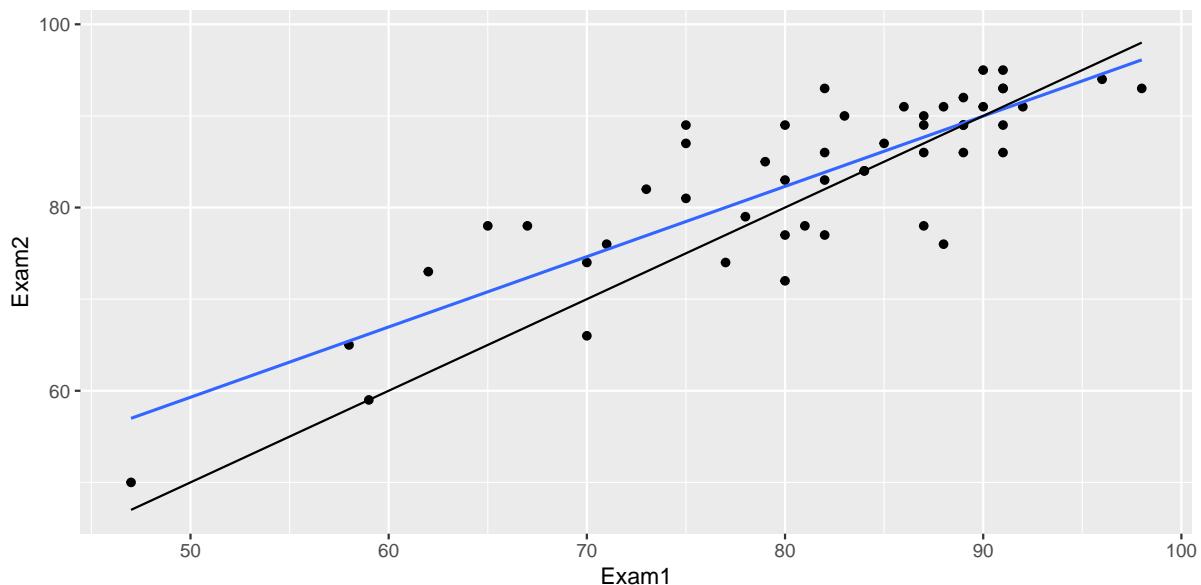
Exam 1 vs Exam 2 scores for intro stat students at another college



What is the relationship between scores on the two exams?

4.7.2 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?

4.7.3 The Regression Effect

A low score on an exam is often the result of both poor preparation and bad luck.

A high score often results from both good preparation and good luck.

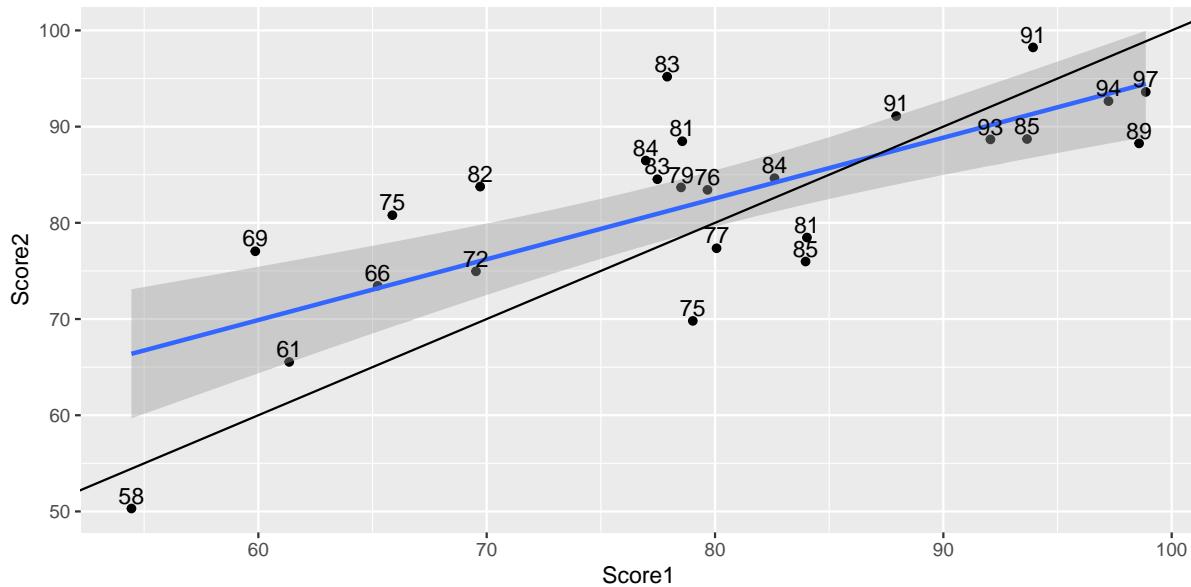
While changes in study habits and preparation likely explain some improvement in low scores, we would also expect the lowest performers to improve simply because of better luck.

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.

4.7.4 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**.

4.7.5 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.

4.7.6 Test Scores Simulation - Lowest Scores

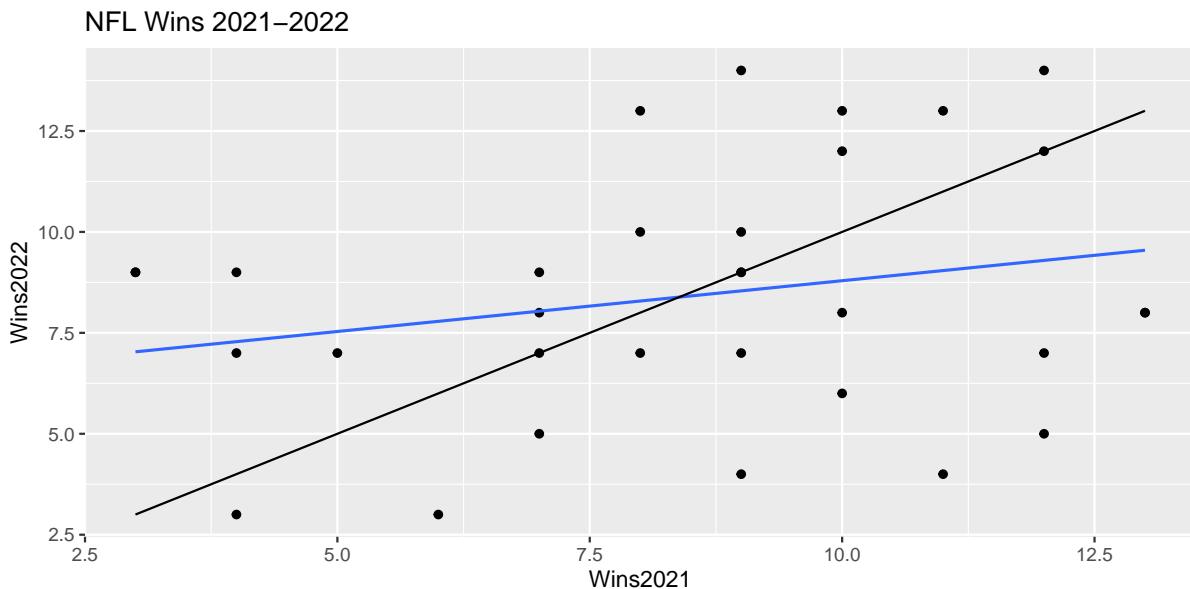
```
kable(head(TestSim%>%arrange(Score1)))
```

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.

4.7.7 Another Example

Wins by NFL teams in 2021 and 2022



4.7.8 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 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.”

4.8 Responsible Statistical Inference

While statistics offer insight that have lead to great advances in science, medicine, business and economics and many other areas, they have also been misused, (either intentionally or unintentionally) in ways that have resulted in harm. In an effort to better educate the public, the American Statistical Association released a report in 2016, highlighting common misuses of p-values, and the notion of statistical significance. Kenneth Rothman, Professor of Epidemiology at Boston University School of Public Health wrote:

- “(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

In this section, we'll look at some real scenarios where it might be tempting to make an improper conclusion based on data. Think carefully about each scenario and identify possible misinterpretations that might arise. Then think about what conclusions or actions would be appropriate in each scenario.

Keep in mind the following guidelines for responsible statistical inference.

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.
- We should only generalize results from a hypothesis test performed on a sample to a larger population or process if the sample is representative of the population or process (such as when the sample is randomly selected).
- A correlation between two variables does not necessarily imply a causal relationship.

4.8.1 Breakfast Cereals and Sex of Babies

The title of a [2008 article by New Scientist](#) makes a surprising claim: “Breakfast cereals boost chances of conceiving boys.” Although the article lacks details on the underlying study, researchers, in fact, tracked the eating habits of women who were attempting to become pregnant. They tracked 133 different food items, and tested whether there was a difference in the proportion of baby boys conceived between women who ate the food, compared to those who didn’t. Of the 133 foods tested, only breakfast cereal showed a significant difference.

Question: Why is the researcher’s conclusion that eating breakfast cereal increases the chances of conceiving a boy inappropriate? Hint: think about how this situation is similar to the one in the [xkcd comic](#)?

A statistical procedure called the **Bonferroni Correction** suggests that when more than one test is performed simultaneously, we should multiply observed p-values by the number of tests performed. Why might this help prevent errors? Do you see any downsides of doing this?

4.8.2 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.

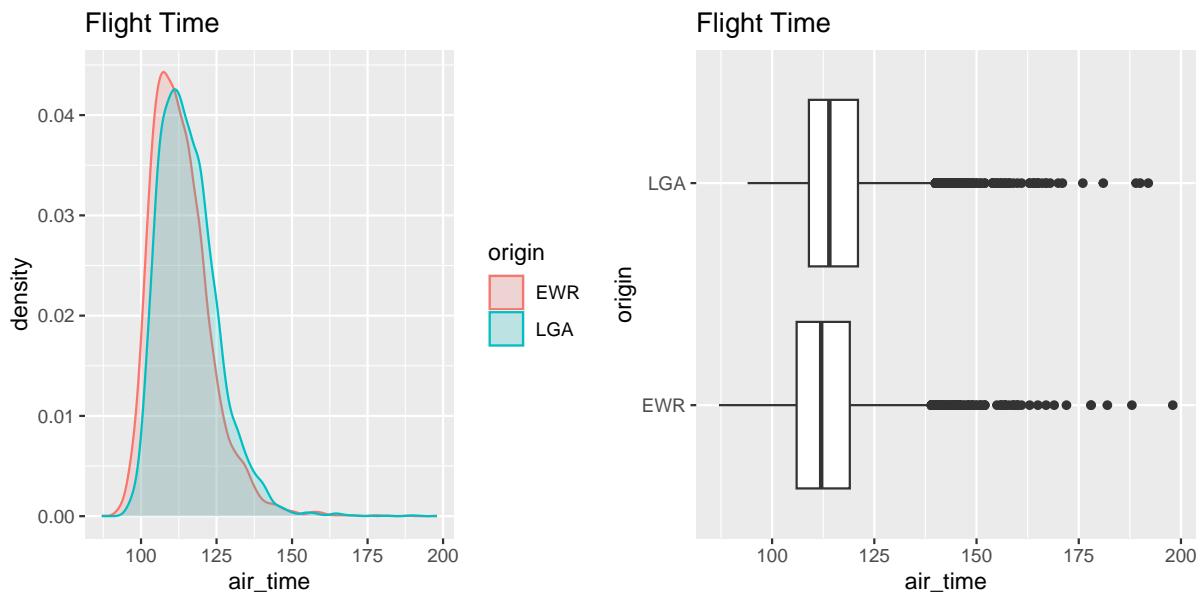
```
library(nycflights13)
data(flights)
flights$origin <- as.factor(flights$origin)
flights$dest <- as.factor(flights$dest)
```

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)
```

The plot and table compare the duration of flights from New York to Chicago from each airport.

```
p1 <- ggplot(data=Flights_NY_CHI, aes(x=air_time, fill=origin, color=origin)) + geom_density()
p2 <- ggplot(data=Flights_NY_CHI, aes(x=air_time, y=origin)) + geom_boxplot() + ggtitle("Flight Time")
grid.arrange(p1, p2, ncol=2)
```



```
library(knitr)
T <- Flights_NY_CHI %>% group_by(origin) %>%
  summarize(Mean_Airtime = mean(air_time, na.rm=TRUE),
            SD = sd(air_time, na.rm=TRUE), n=sum(!is.na(air_time)))
kable(T)
```

origin	Mean_Airtime	SD	n
EWR	113.2603	9.987122	5828
LGA	115.7998	9.865270	8507

We fit a model to test whether there is evidence of a difference in average flight time.

```
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: < 0.0000000000000022

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.

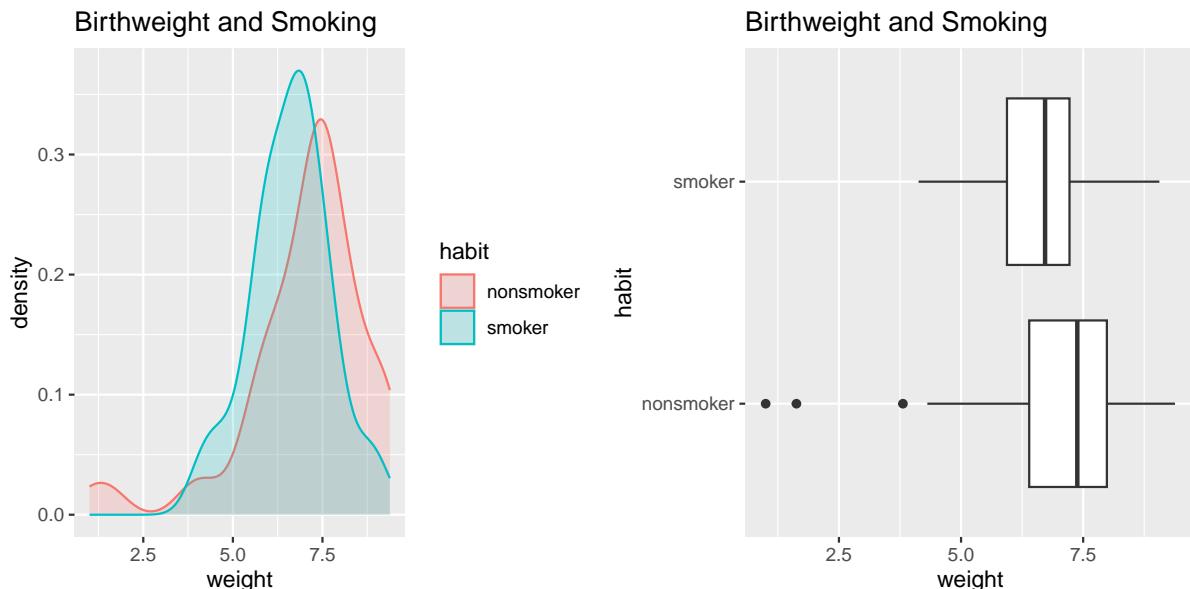
Question: If you were planning a trip from New York to Chicago, how much of a factor would this information play in your decision? Why do you think the p-value is so small in this scenario?

4.8.3 Smoking and Birthweight Example

We consider data on the relationship between a pregnant mother's smoking and the birth weight 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.

The plot and table show the distribution of birth weights among babies whose mothers smoked, compared to those who didn't.

```
p1 <- ggplot(data=NCBirths, aes(x=weight, fill=habit, color=habit)) + geom_density(alpha=0.2)
p2 <- ggplot(data=NCBirths, aes(x=weight, y=habit)) + geom_boxplot() + ggtitle("Birthweight and Smoking")
grid.arrange(p1, p2, ncol=2)
```



```
library(knitr)
T <- NCBirths %>% group_by(habit) %>% summarize(Mean_Weight = mean(weight), SD = sd(weight),
kable(T)
```

habit	Mean_Weight	SD	n
nonsmoker	7.039200	1.709388	50
smoker	6.616333	1.106418	30

We fit a model and test for differences in average birth weight.

```
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

```
confint(M_Birthwt)
```

	2.5 %	97.5 %
(Intercept)	6.613070	7.4653303
habitsmoker	-1.118735	0.2730012

- The average birth weight 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 birth weights. 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.

Question: 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?

4.8.3.1 Larger Study

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.

We'll now fit a model to the larger dataset.

```
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

Confidence interval based on larger dataset:

```
confint(M_Birthwt_Full)
```

	2.5 %	97.5 %
(Intercept)	7.0444697	7.24407557
habitsmoker	-0.5965648	-0.03452013

Notice that the estimated difference in birth weights is actually smaller in the full dataset than in the subset (though 0.3 lbs is still a pretty big difference for babies), and yet now the p-value is much smaller. Why do you think this happened? What should we learn from this?

4.8.4 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?

5 Building and Assessing Models

Learning Outcomes:

31. Check the validity of model assumptions using plots and other information.
32. Interpret regression coefficients in models involving log transformations.
33. Calculate predicted values and confidence/prediction intervals for models involving log transformations.
34. Identify instances of confounding and Simpson's paradox and draw conclusions in these situations.
35. Explain how multicollinearity impacts predictions and confidence/prediction intervals.
36. Build and interpret multiple and polynomial regression models in R.

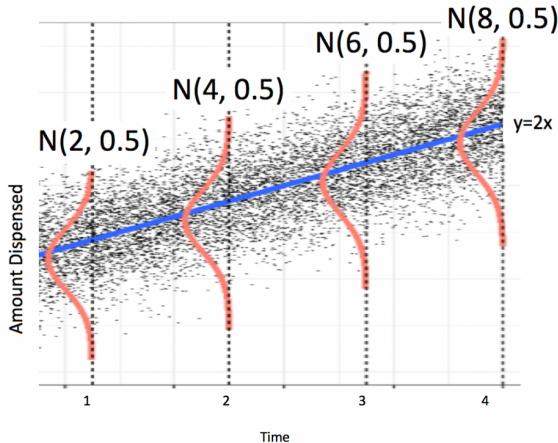
5.1 Checking Regression Assumptions

We've seen that tests and intervals based on the normal error regression model depend on four assumptions. If these assumptions are not reasonable then the tests and intervals may not be reliable.

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 . (This assumption is only relevant for models including at least one quantitative explanatory variable.)
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: The response value for each observation is not affected by any of the other observations (expect due to explanatory variables included in the model).

Illustration of Model Assumptions



Modified from image at <https://bookdown.org/robact/bookdown-bysh/ch-poissonreg.html>

We know that these assumptions 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.

Of course, these assumptions will almost never be truly satisfied, but they should at least be a reasonable approximation if we are to draw meaningful conclusions.

5.1.1 Checking Model Assumptions

The following plots are useful when assessing the appropriateness of the normal error regression model.

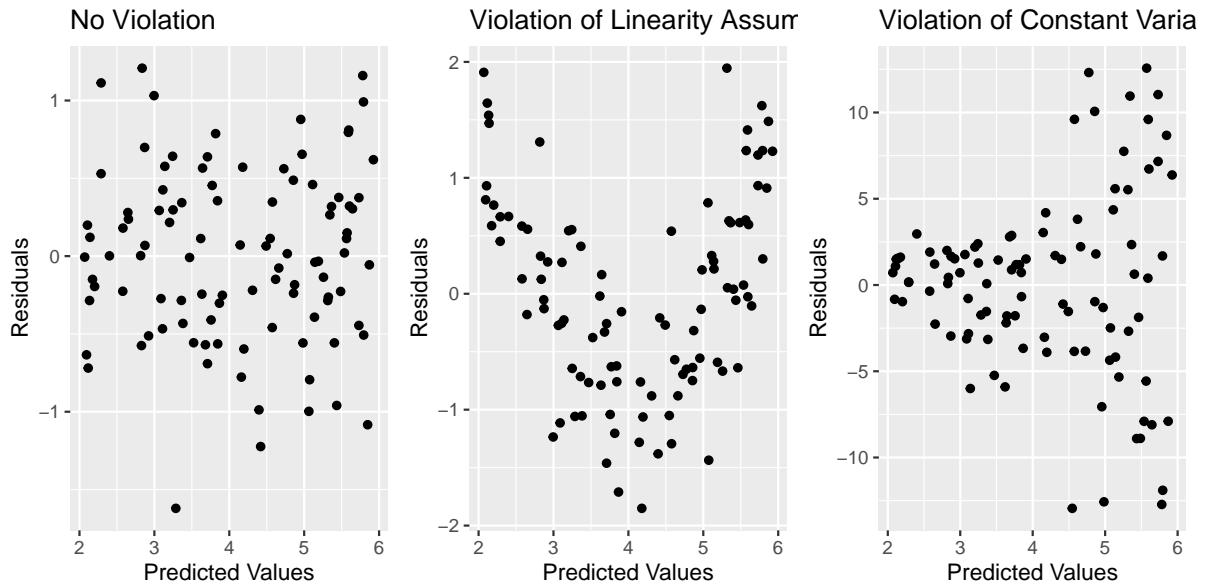
1. Scatterplot of residuals against predicted values
2. Histogram of standardized residuals
 - heavy skewness indicates a problem with normality assumption
3. Normal quantile plot
 - severe departures from diagonal line indicate problem with normality assumption

Residual vs Predicted Plots

A residual vs predicted plot is 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)
```



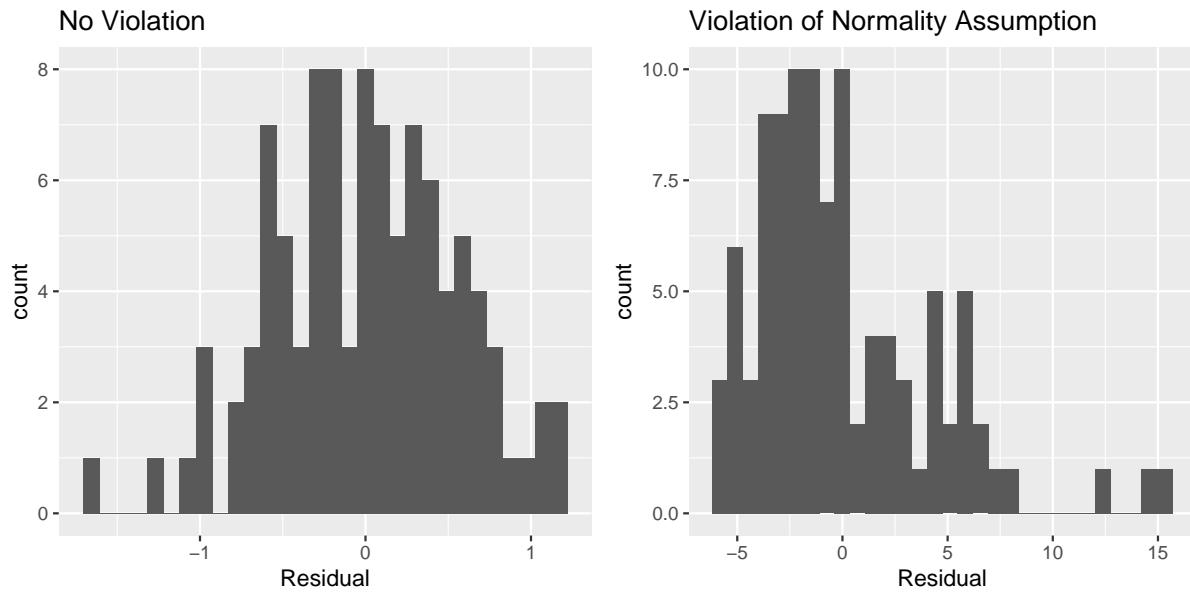
If there is only one explanatory variable, plotting the residuals against that variable reveals the same information as a residual vs predicted plot.

Histogram of Residuals

A histogram of the residuals is useful for assessing the 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)
```



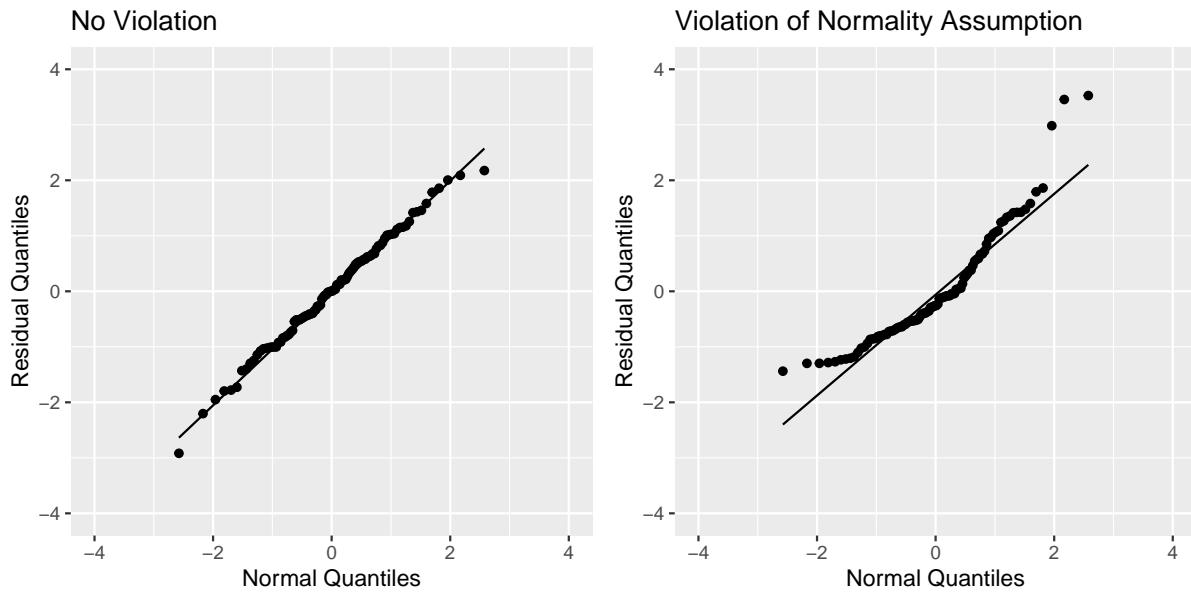
Normal Quantile-Quantile (QQ) Plot

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 = scale(no_viol_Model$residuals))) + stat_qq() + stat_qq_line()
P2 <- ggplot(data=Violations, aes(sample = scale(norm_viol_Model$residuals))) + stat_qq() + stat_qq_line()
grid.arrange(P1, P2, ncol=2)
```



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.

5.1.2 Summary of Checks for Model Assumptions

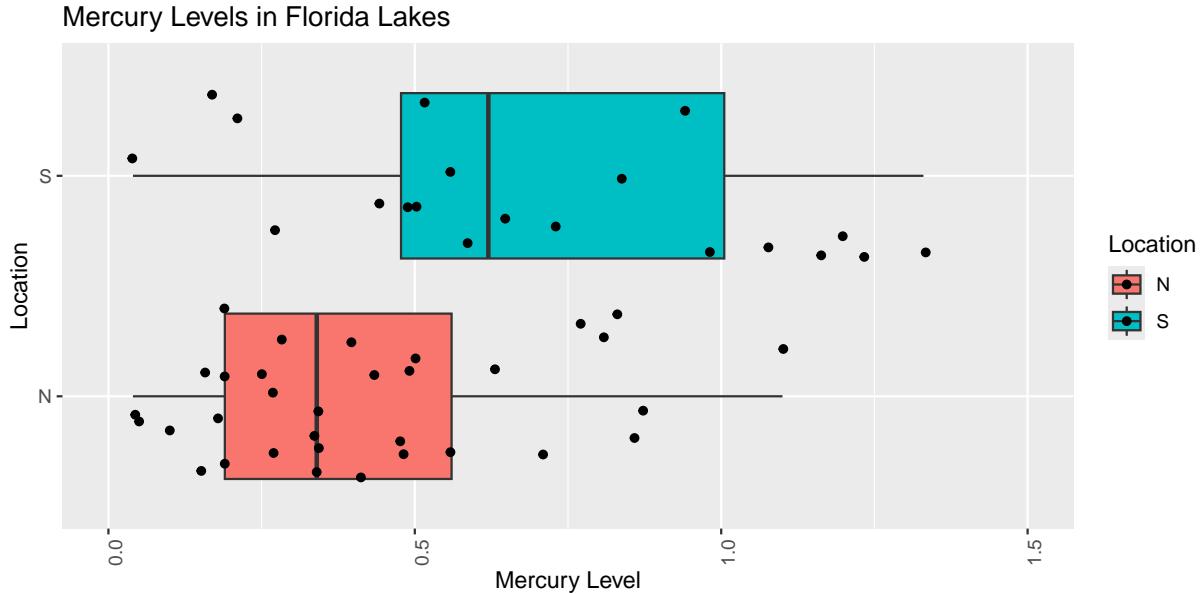
Model assumption	How to detect violation
Linearity	Curvature in residual plot
Constant Variance	Funnel shape in residual plot
Normality	Skewness in histogram of residuals or departure from diag. line in QQ plot
Independence	No graphical check, carefully examine data collection

5.1.3 Example: N v S Lakes

Recall our sample of 53 Florida Lakes, 33 in the north, and 20 in the south.

$$\text{Mercury}_i = \beta_0 + \beta_1 \times \text{South}_i + \epsilon_i, \text{ where } \epsilon_i \sim \mathcal{N}(0, \sigma).$$

LakesBP



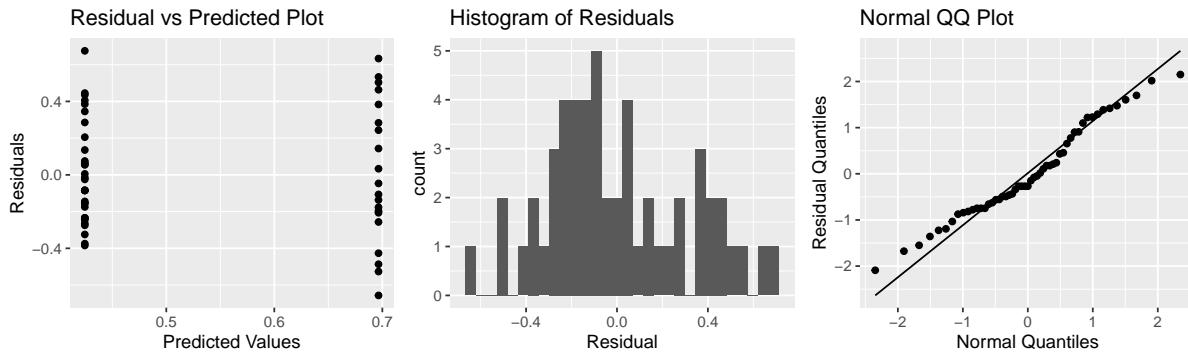
When we use the normal error regression model, we are assuming the following:

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, except for being in the north or south, which we account for in the model. 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.

A residual by predicted plot, histogram of residuals, and normal quantile-quantile plot are shown below.

```
P1 <- ggplot(data=FloridaLakes, aes(y=Lakes_M$residuals, x=Lakes_M$fitted.values)) + geom_point()
P2 <- ggplot(data=FloridaLakes, aes(x=Lakes_M$residuals)) + geom_histogram() + ggtitle("Histogram of Residuals")
P3 <- ggplot(data=FloridaLakes, aes(sample = scale(Lakes_M$residuals))) + stat_qq() + stat_qq_line()
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.

We shouldn't be concerned about using theory-based hypothesis tests or confidence intervals for the mean mercury level or difference in mean mercury levels. There might be some concern that prediction intervals could be either too wide or too narrow, but this is not a major concern, since the constant variance assumption is not severe.

5.1.4 Example: pH Model

Recall the regression line estimating the relationship between a lake's mercury level and pH.

$$\text{Mercury}_i = \beta_0 + \beta_1 \times \text{pH}_i + \epsilon_i, \text{ where } \epsilon_i \sim \mathcal{N}(0, \sigma).$$

The model assumes:

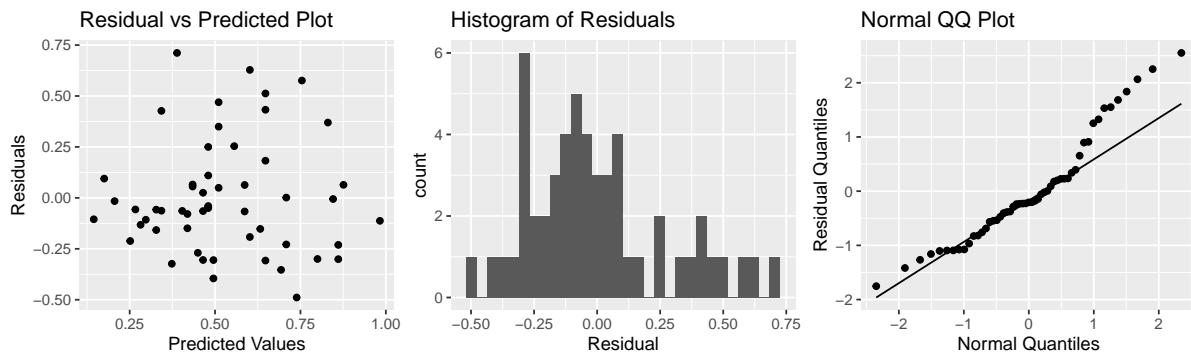
1. Linearity: the expected mercury level of a lake is a linear function of pH.
2. Normality: for any given pH, the mercury levels of lakes with that pH follow a normal distribution. For example, mercury levels for lakes with pH of 6 are normally distributed, and mercury levels for lakes with pH of 9 are normally distributed, though these normal distributions may have different means.

3. Constant Variance: the variance (or standard deviation) in the normal distribution for mercury level is the same for each pH. For example, there is the same amount of variability associated with lakes with pH level 6, as pH level 8.
4. Independence: no two lakes are any more alike than any others, except with respect to pH, which is accounted for in the model. This may not be a reasonable assumption, but it's unclear what the effects of such a violation would be.

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.

The plots for checking these assumptions are shown below.

```
P1 <- ggplot(data=FloridaLakes, aes(y=M_pH$residuals, x=M_pH$fitted.values)) + geom_point()
P2 <- ggplot(data=FloridaLakes, aes(x=M_pH$residuals)) + geom_histogram() + ggtitle("Histogram of Residuals")
P3 <- ggplot(data=FloridaLakes, aes(sample = scale(M_pH$residuals))) + stat_qq() + stat_qq_line()
grid.arrange(P1, P2, P3, ncol=3)
```



The residual vs predicted plot does not show any linear trend, and variability appears to be about the same for low predicted values as for high ones. Thus, the linearity and constant variance assumptions appear reasonable.

The histogram shows some right-skewness, and the right-most points on the normal-qq plot are above the line, indicating a possible concern with the normality assumption. There is some evidence of right-skewness, which might impact the appropriateness of the normal error regression model.

Nevertheless, we obtained similar results using the simulation-based results as the normal error regression model, suggesting that the concern about normality did not have much impact on the estimation of β_1 . It is possible that this concern could have implications for other kinds of inference, such as confidence intervals for an expected response, and prediction intervals, which we'll explore later in the chapter.

5.1.5 Example: House Prices

Recall the model for estimating price of a house, using size, waterfront status, and an interaction term.

$$\text{Price}_i = \beta_0 + \beta_1 \text{Sq.Ft.}_i + \beta_2 \text{Waterfront}_i + \beta_3 \times \text{Sq.Ft.}_i \times \text{Waterfront}_i + \epsilon_i, \text{ where } \epsilon_i \sim \mathcal{N}(0, \sigma).$$

The model assumes:

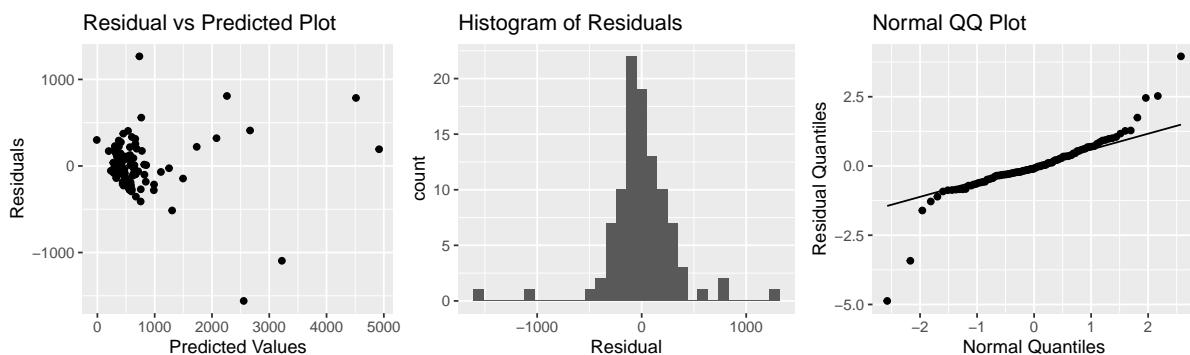
1. Linearity: the expected price of a house is a linear function of its size. The slope and intercept of this function may be different for houses on the waterfront, compared to houses not on the waterfront.
2. Normality: prices of houses of a given size and waterfront status are normally distributed.
3. Constant Variance: the variance (or standard deviation) in the normal distribution for prices is the same for all sizes and waterfront statuses.
4. Independence: no two houses are any more alike than any others, except with respect to size and waterfront status.

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.

Several reasons come to mind that might cause us to doubt the validity of these assumptions, but let's investigate them empirically, using our data on 100 houses.

The plots for checking these assumptions are shown below.

```
P1 <- ggplot(data= Houses, aes(y=M_House_Int$residuals, x=M_House_Int$fitted.values)) + geom_point()
P2 <- ggplot(data= Houses, aes(x=M_House_Int$residuals)) + geom_histogram() + ggtitle("Histogram of Residuals")
P3 <- ggplot(data= Houses, aes(sample = scale(M_House_Int$residuals))) + stat_qq() + stat_qq_line()
grid.arrange(P1, P2, P3, ncol=3)
```



Although we might have had some initial concerns about the model assumptions, the plots do not raise any serious concerns. There is no sign of a nonlinear relationship in the residual vs predicted plot, so the linearity assumption appears reasonable.

There is possibly more variability associated with prices or more expensive houses than less expensive ones, so we might have some concerns about constant variance, but since there are only a few very high-priced houses, and the increasing variance is not too severe, this may not be much of a concern.

There are a few houses on each end of the normal qq plot that deviate from their expected line, but not very many. It's not uncommon to have a few points deviate from the line on the end, so we do not have severe concerns about normality. The histogram of residuals is roughly symmetric.

Thus, the normal error regression model appears to be reasonable for these data.

5.1.6 Impact of Model Assumption Violations

In this chapter, we've studied the normal error regression model and its underlying assumptions. We've seen that when these assumptions are realistic, we can use distributions derived from probability theory, such as t and F distributions to approximate sampling distributions, in place of the simulation-based methods seen in Chapters 3 and 4.

Of course, real data don't come exactly from processes like the fictional ice cream dispenser described in Section 5.1, so it's really a question of whether this model is a realistic approximation (or simplification) of the true mechanism that led to the data we observe. We can use diagnostics like residual and Normal-QQ plots, as well as our intuition and background knowledge to assess whether the normal error regression model is a reasonable approximation.

The p-values provided by the `lm` summary output, and `anova` commands, and the and intervals produced by the `confint`, and `predict` command, as well as many other R commands, depend on the assumptions of the normal error regression model, and should only be used when these assumptions are reasonable.

In situations where some model assumptions appear to be violated, we might be okay using certain tests/intervals, but not others. In general, we should proceed with caution in these situations.

The table below provides guidance on the potential impact of model assumption violation on predicted values, confidence intervals, and prediction intervals.

Model assumption Violated	Predicted Values	Confidence Intervals	Prediction Intervals
Linearity	Unreliable	Unreliable	Unreliable

Model assumption Violated	Predicted Values	Confidence Intervals	Prediction Intervals
Constant Variance	Reliable	Somewhat unreliable - Some too wide, others too narrow	Very unreliable - Some too wide, others too narrow
Normality	Reliable	Possibly unreliable - might be symmetric when they shouldn't be. Might be okay when skewness isn't bad and sample size is large.	Very unreliable - will be symmetric when they shouldn't be
Independence	might be reliable	unreliable - either too wide or too narrow	unreliable - either too wide or too narrow

When model assumptions are a concern, consider using a transformation of the data, a more advanced model, or a more flexible technique, such as a nonparametric approach or statistical machine learning algorithm.

5.2 Transformations

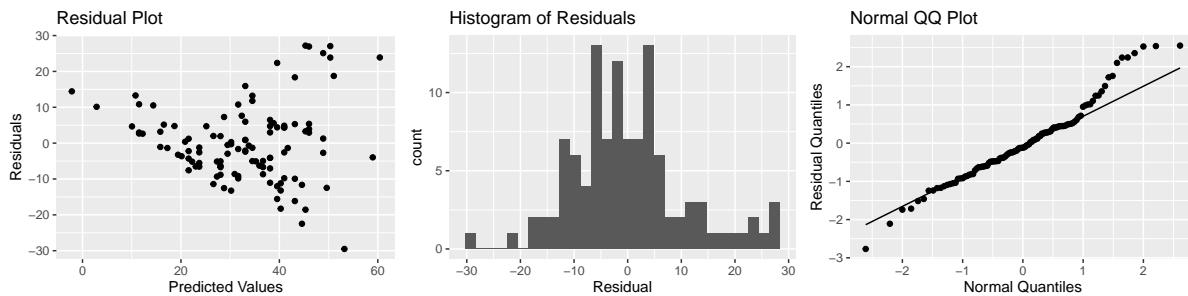
When there are violations of model assumptions, we can sometimes correct for these by modeling a function of the response variable, rather than the response variable itself. When the histogram of residuals and normal qq plot show signs of right-skewness, modeling $\log(Y)$ is often helpful.

5.2.1 Example: Modeling Car Prices

We'll look at the assumptions associated with the model for predicting car price, using acceleration time as the explanatory variable.

Diagnostic plots are shown below.

```
P1 <- ggplot(data=Cars2015, aes(y=Cars_M_A060$residuals, x=Cars_M_A060$fitted.values)) + geom_point()
P2 <- ggplot(data=Cars2015, aes(x=Cars_M_A060$residuals)) + geom_histogram() + ggtitle("Histogram of Residuals")
P3 <- ggplot(data=Cars2015, aes(sample = scale(Cars_M_A060$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.

5.2.2 Log Transformation

When residual plots yield model inadequacy, we might try to correct these by applying a transformation to the response variable.

When working a nonnegative, right-skewed response variable, it is often helpful to work with the logarithm of the response variable.

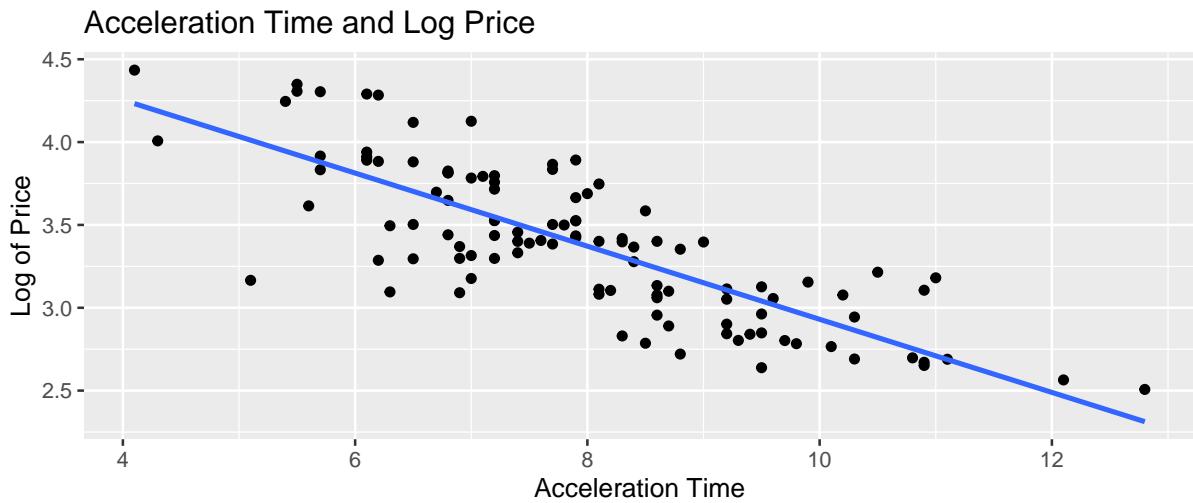
Note: In R, `log()` denotes the natural (base e) logarithm, often denoted `ln()`. We can actually use any logarithm, but the natural logarithm is commonly used.

We'll use the model:

$$\text{Log Price} = \beta_0 + \beta_1 \times \text{Acc060} + \epsilon_i, \text{ where } \epsilon_i \sim \mathcal{N}(0, \sigma)$$

The plot shows $\log(\text{price})$ on the y-axis. We see that the relationship appears more linear than when we plot price itself.

```
ggplot(data=Cars2015, aes(x=Acc060, y=log(Price))) + geom_point() +
  xlab("Acceleration Time") + ylab("Log of Price") +
  ggtitle("Acceleration Time and Log Price") + stat_smooth(method="lm", se=FALSE)
```



Log Transformation 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.

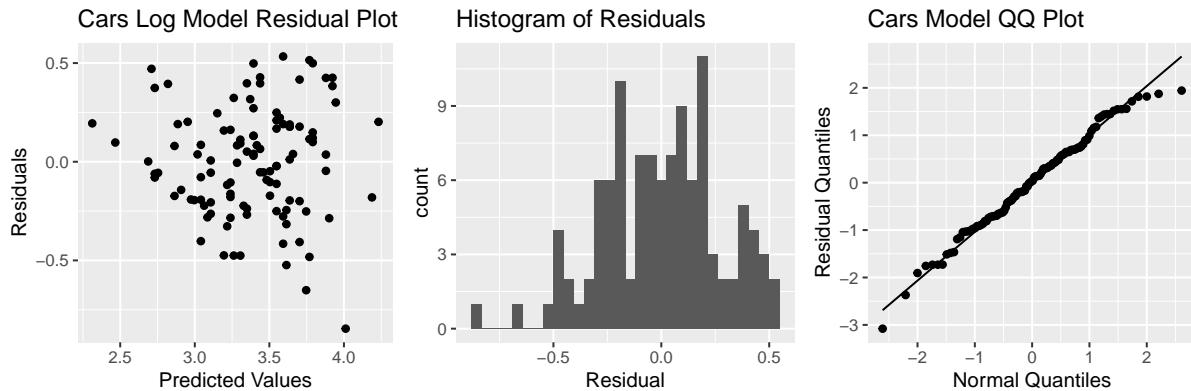
We should only use the p-values and confidence intervals provided by R, which depend on the t-distribution approximation, if we believe these assumptions are reasonable.

Assumption Check for Model on Log Price

```

P1 <- ggplot(data=Cars2015, aes(y=Cars_M_Log$residuals, x=Cars_M_Log$fitted.values)) + geom_point()
P2 <- ggplot(data=Cars2015, aes(x=Cars_M_Log$residuals)) + geom_histogram() + ggtitle("Histogram of Residuals")
P3 <- ggplot(data=Cars2015, aes(sample = scale(Cars_M_Log$residuals))) + stat_qq() + stat_qq_line()
grid.arrange(P1, P2, P3, ncol=3)

```



There is still some concern about constant variance, though perhaps not as much. The normality assumption appears more reasonable.

5.2.3 Inference for Log Model

R output for the model is shown below.

```

Cars_M_Log <- lm(data=Cars2015, log(Price)~Acc060)
summary(Cars_M_Log)

```

```

Call:
lm(formula = log(Price) ~ 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

Prediction Equation:

$$\widehat{\text{Price}} = e^{5.13582 - 0.22064 \times \text{Acc060}}$$

Predicted price for car that takes 7 seconds to accelerate:

$$\widehat{\text{Price}} = e^{5.13582 - 0.22064 \times 7} = 36.3$$

Predicted price for car that takes 10 seconds to accelerate:

$$\widehat{\text{Price}} = e^{5.13582 - 0.22064 \times 10} = 18.7$$

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.

5.2.4 Log Model Interpretations

Log of Expected Price = $\beta_0 + \beta_1 \times \text{Acc060}$, Thus:

$$\text{Expected Price} = e^{\beta_0 + \beta_1 \times \text{Acc060}}$$

$$e^{\beta_0} e^{\beta_1 \times \text{Acc060}}$$

$$e^{\beta_0} (e^{\beta_1})^{\text{Acc060}}$$

- e^{β_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^{β_1} .

Exponentiating the model coefficients gives:

```
exp(Cars_M_Log$coefficients)
```

(Intercept)	Acc060
170.1730148	0.8020062

- 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.

Confidence Intervals for β_0 and β_1

```
confint(Cars_M_Log)
```

	2.5 %	97.5 %
(Intercept)	4.8787105	5.3949208
Acc060	-0.2524862	-0.1887916

```
exp(confint(Cars_M_Log))
```

	2.5 %	97.5 %
(Intercept)	131.4610408	220.284693
Acc060	0.7768669	0.827959

- 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.

Log Model CI for Expected Response

If we just use the `predict` function, we get a confidence interval for `log(price)`.

```
predict(Cars_M_Log, newdata=data.frame(Acc060=c(7)), interval="confidence")
```

```
fit      lwr      upr  
1 3.592343 3.53225 3.652436
```

To get an interval for price itself, we exponentiate, using `exp`.

```
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.

Log Model Prediction Interval

```
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.

5.2.5 Model Comparisons

We'll compare the intervals we obtain using the `log` transformation to those from the model without the transformation.

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
```

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
```

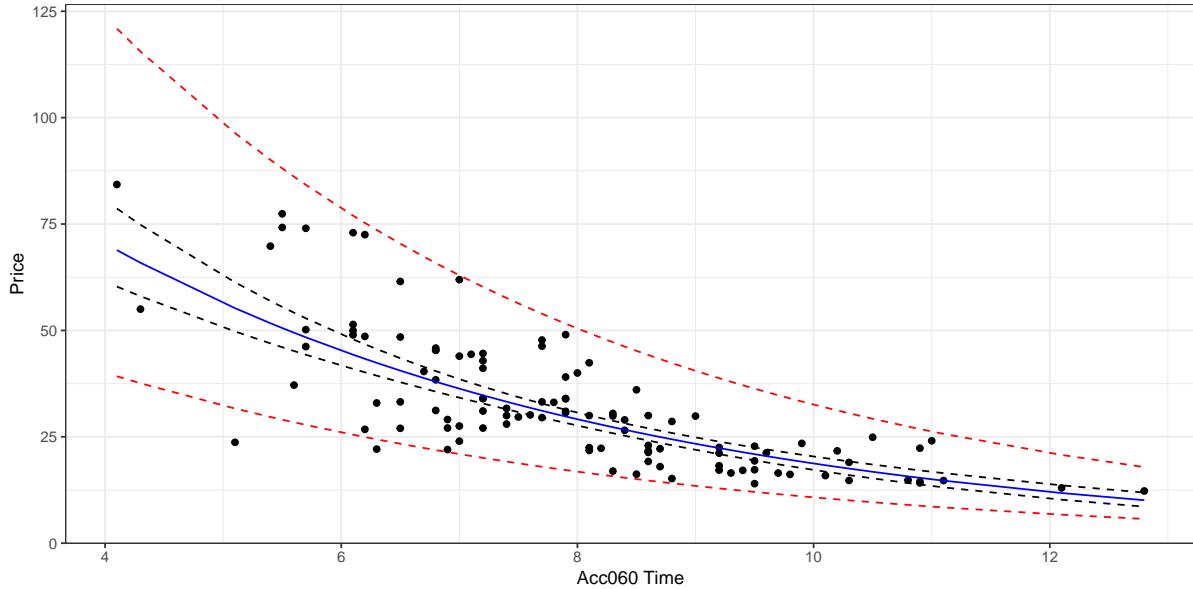
Transformed Model:

```
exp(predict(Cars_M_Log, newdata=data.frame(Acc060=c(7)), interval="prediction", level=0.95))

    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.

5.2.6 Log Model Visualization



The log model suggests a nonlinear trend in price with respect to acceleration time and gives wider confidence and prediction intervals for cars that accelerate faster and tend to be more expensive. It also gives non-symmetric intervals. These results appear to be consistent with the observed data.

5.2.7 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^{β_0} represents the expected response in the baseline category
 - e^{β_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.

5.3 Building Models for Interpretation: Confounding, Multicollinearity, and Polynomial Regression

So far, we've dealt with models with 2 or fewer variables. Some real questions require accounting for more than two variables. In these situations, we'll need to develop a model that is complex enough to capture the important aspects of the mechanism we're modeling, but also simple enough for us to be able to explain and interpret. We'll need to decide how many variables to include in the model, and whether to use transformations, or to include interaction terms.

We'll examine strategies for modeling in two different contexts. In this chapter, we'll focus on building models for situations when we want to make interpretations and draw conclusions about relationships between variables. In Chapter 7, we focus on modeling solely for the purpose of prediction, when we are not interested in making interpretations or conclusions about relationships between variables.

When building a model for the purpose of interpretation, we are typically interested in investigating a research question pertaining to relationships between explanatory and response variables. We'll need to think about things like:

- which explanatory variables should we include in the model, and how many?
- should we include any interaction terms?
- should we use any nonlinear terms?
- should we use a transformation of the response variable?

We'll go through a couple example to see how we can address these questions in building a model.

Keep in mind, there is no single correct model, but there are common characteristics of a good model. While two statisticians might use different models for a given set of data, they will hopefully lead to reasonably similar conclusions if constructed carefully.

5.3.1 Modeling SAT Scores

We'll now look at a dataset containing education data on all 50 states. It includes the following variables.

`state` - a factor with names of each state

`expend` - expenditure per pupil in average daily attendance in public elementary and secondary schools, 1994-95 (in thousands of US dollars)

ratio - average pupil/teacher ratio in public elementary and secondary schools, Fall 1994

salary - estimated average annual salary of teachers in public elementary and secondary schools, 1994-95 (in thousands of US dollars)

frac - percentage of all eligible students taking the SAT, 1994-95

sat - average total SAT score, 1994-95

region - region of the country

```
library(mosaicData)
data(SAT)
SAT <- SAT %>% dplyr::select(-c(verbal, math))
library(Lock5Data)
data("USStates")
SAT <- SAT %>% left_join(USStates %>% select(State, Region), by=c("state"="State")) %>% renam
```

SAT

		state	expend	ratio	salary	frac	sat	region
1		Alabama	4.405	17.2	31.144	8	1029	S
2		Alaska	8.963	17.6	47.951	47	934	W
3		Arizona	4.778	19.3	32.175	27	944	W
4		Arkansas	4.459	17.1	28.934	6	1005	S
5		California	4.992	24.0	41.078	45	902	W
6		Colorado	5.443	18.4	34.571	29	980	W
7		Connecticut	8.817	14.4	50.045	81	908	NE
8		Delaware	7.030	16.6	39.076	68	897	NE
9		Florida	5.718	19.1	32.588	48	889	S
10		Georgia	5.193	16.3	32.291	65	854	S
11		Hawaii	6.078	17.9	38.518	57	889	W
12		Idaho	4.210	19.1	29.783	15	979	W
13		Illinois	6.136	17.3	39.431	13	1048	MW
14		Indiana	5.826	17.5	36.785	58	882	MW
15		Iowa	5.483	15.8	31.511	5	1099	MW
16		Kansas	5.817	15.1	34.652	9	1060	MW
17		Kentucky	5.217	17.0	32.257	11	999	MW
18		Louisiana	4.761	16.8	26.461	9	1021	S
19		Maine	6.428	13.8	31.972	68	896	NE
20		Maryland	7.245	17.0	40.661	64	909	NE
21		Massachusetts	7.287	14.8	40.795	80	907	NE
22		Michigan	6.994	20.1	41.895	11	1033	MW
23		Minnesota	6.000	17.5	35.948	9	1085	MW

24	Mississippi	4.080	17.5	26.818	4	1036	S
25	Missouri	5.383	15.5	31.189	9	1045	MW
26	Montana	5.692	16.3	28.785	21	1009	W
27	Nebraska	5.935	14.5	30.922	9	1050	MW
28	Nevada	5.160	18.7	34.836	30	917	W
29	New Hampshire	5.859	15.6	34.720	70	935	NE
30	New Jersey	9.774	13.8	46.087	70	898	NE
31	New Mexico	4.586	17.2	28.493	11	1015	W
32	New York	9.623	15.2	47.612	74	892	NE
33	North Carolina	5.077	16.2	30.793	60	865	S
34	North Dakota	4.775	15.3	26.327	5	1107	MW
35	Ohio	6.162	16.6	36.802	23	975	MW
36	Oklahoma	4.845	15.5	28.172	9	1027	S
37	Oregon	6.436	19.9	38.555	51	947	W
38	Pennsylvania	7.109	17.1	44.510	70	880	NE
39	Rhode Island	7.469	14.7	40.729	70	888	NE
40	South Carolina	4.797	16.4	30.279	58	844	S
41	South Dakota	4.775	14.4	25.994	5	1068	MW
42	Tennessee	4.388	18.6	32.477	12	1040	S
43	Texas	5.222	15.7	31.223	47	893	S
44	Utah	3.656	24.3	29.082	4	1076	W
45	Vermont	6.750	13.8	35.406	68	901	NE
46	Virginia	5.327	14.6	33.987	65	896	S
47	Washington	5.906	20.2	36.151	48	937	W
48	West Virginia	6.107	14.8	31.944	17	932	S
49	Wisconsin	6.930	15.9	37.746	9	1073	MW
50	Wyoming	6.160	14.9	31.285	10	1001	W

Note that the dataset is quite old (from 1994-95), so the financial information may be out of date. Nevertheless, it is useful for exploring relationships between SAT scores and other variables.

Research Question

A good statistical research question should be one that has practical implications that people would care about. It should be complex enough to be worth investigating. If the answer is obvious, then there would be no need to use statistics, or scientific reasoning in general.

For the SAT score dataset, we'll focus on the question:

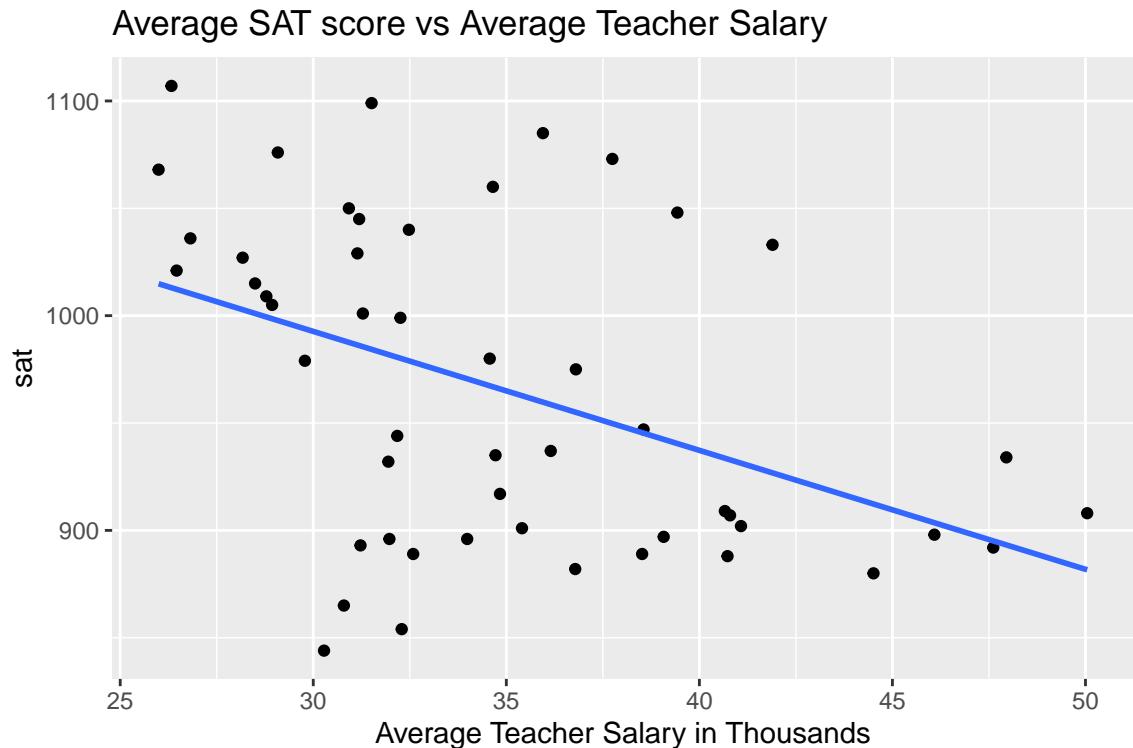
Do students in states that prioritize education spending achieve better SAT scores?

While this may seem like a straightforward question, we'll see that answering it properly requires careful thought and analysis.

5.3.1.1 Initial Model

One way to measure a state's investment in education is in how much it pays its teachers. 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")
```



Fitting a simple linear regression model to the data, we obtain the following:

```
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 < 0.0000000000000002 ***
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

```

On average, SAT score is expected to decrease by about 5.5 points for each additional one thousand dollars in average teacher salary in the state. The low p-value suggests a relationship like this is unlikely to occur by chance, though the practical importance of a 5-point decrease in SAT score (out of 1600) seems minimal. Furthermore, only 19% of the total variation in SAT score is explained by teaching salary. Nevertheless, a person looking to argue against raising teacher salaries might use the negative estimate and low p-value as a justification for their position.

5.3.1.2 A Deeper Investigation

Notice that there are large discrepancies in the `frac` variable, representing the percentage of students taking the SAT. In Connecticut, 81% of high school students took the SAT, compared to only 6% in Arkansas.

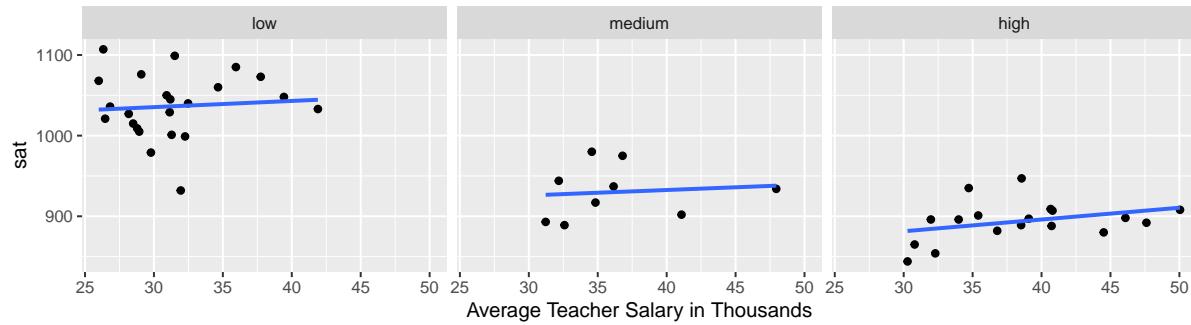
Let's break the data down by the percentage of students who take the SAT. We'll (somewhat arbitrarily), divide the states into

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")))
```

Plotting SAT score against average teacher salary in each state, we see that the picture changes.

```
ggplot(data=SAT, aes( y=sat, x=salary )) +geom_point() + facet_wrap(facets = ~fracgrp) +
stat_smooth(method="lm", se=FALSE) + xlab("Average Teacher Salary in Thousands")
```



There appears to be a slight positive relationship between teacher salary and SAT score in each state.

While breaking up the data into these three groups helps us visualize, we'll simply add the `frac` variable to the model as a quantitative variable, rather than breaking it into these arbitrary categories.

```
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: < 0.0000000000000002
```

For each one thousand dollar increase in average teacher salary, a state's average SAT score is expected to increase by 2.18 points, assuming percentage of students taking the test is the same.

For each one percent increase in percentage of students taking the SAT, a state's average score is expected to decrease by 2.78 points, assuming average teacher salary is the same.

Both of these estimates are associated with low p-values. While the effect of a 2 point increase per \$1,000 in average teacher salary might seem small, the ~3 point decrease for each percentage point of students taking the exam is quite meaningful. According to the model, if the percentage of students taking the SAT is 10 percentage points higher than another, and the states pay their teachers the same, then the state with more people taking the exam is expected to have an average score almost 30 points lower.

Adding percentage of students taking the exam increased the R^2 value substantially.

We see that the relationship between SAT score and salary appears to reverse when we account for percentage of students taking the test. States with low percentages of people taking the SAT tend to get higher scores, as the people taking the test tend to be those who are best prepared and have strong incentive for taking it, perhaps because they are trying to get into an elite college. At the same time, states that pay their teachers more tend to have higher percentages of people taking the SAT. This may be because states that prioritize education are more likely to cover the cost of students taking the test, or even to require it. It may also be that many of the states that require the SAT are coastal states, where cost of living, and thus teacher salaries, tend to be higher in general. Thus, it appears initially that teacher salaries are negatively correlated with SAT scores, but after accounting for percentage taking the test, the trend reverses. Situations where an apparent trend disappears or reverses after accounting for another variable are called **Simpson's Paradox**.

5.3.1.3 Student-to-Teacher Ratio

Let's see what other possible explanatory variables we might want to add to the model. Keep in mind that our goal is to understand the relationship between teacher salary and SAT scores in the state, so we should only use variables that help us understand this relationship. In addition to teacher salaries, student-to-teacher ratio might be an indication of a state's investment in education. We'll add student-to-teacher ratio to the model and explore whether there is evidence that hiring enough teachers to keep student-to-teacher ratio low has a benefit, in terms of SAT score.

```
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	<0.0000000000000002 ***							
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: < 0.0000000000000002

Interpretations

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 drop in average SAT score, assuming average teacher salary, and percentage of students taking the SAT are held constant.

We see that student to teacher ratio is negatively associated with SAT score, with an expected drop of about 4.6 points in average SAT score for each additional student per teacher, assuming average teacher salary and percentage of students taking the exam are held constant. This suggests that states should try to keep student to teacher ratios low. We see teacher salary remains positively correlated with SAT score and percentage taking the test remains negatively correlated, after accounting for student to teacher ratio.

5.3.1.4 Multicollinearity

Next, let's add the variable `expend`, which measures the state's expenditure per pupil.

```
SAT_M4 <- lm(data=SAT, sat~salary+frac+ratio+expend)
summary(SAT_M4)
```

Call:
`lm(formula = sat ~ salary + frac + ratio + expend, data = SAT)`

Residuals:

Min	1Q	Median	3Q	Max
-90.531	-20.855	-1.746	15.979	66.571

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	1045.9715	52.8698	19.784	< 0.0000000000000002 ***
salary	1.6379	2.3872	0.686	0.496
frac	-2.9045	0.2313	-12.559	0.0000000000000261 ***
ratio	-3.6242	3.2154	-1.127	0.266
expend	4.4626	10.5465	0.423	0.674

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 32.7 on 45 degrees of freedom
Multiple R-squared: 0.8246, Adjusted R-squared: 0.809
F-statistic: 52.88 on 4 and 45 DF, p-value: < 0.0000000000000022

It may be surprising to see that after accounting for expenditure per student, teacher salary is still positively correlated with SAT score, but that the p-value associated with teacher salary is quite large. Likewise, while student-to-teacher ratio is still negatively associated, it too has a large p-value. Also notice that R^2 barely increased when accounting for total expenditures.

This happens because expenditures are highly correlated with teacher salary. States that pay their teacher more also spend more on education per pupil. The scatterplot matrix below shows a strong correlation of 0.87 between teacher salary and expenditures.

```
SAT_Num <- select_if(SAT, is.numeric)
C <- cor(SAT_Num, use = "pairwise.complete.obs")
round(C,2)
```

	expend	ratio	salary	frac	sat
expend	1.00	-0.37	0.87	0.59	-0.38
ratio	-0.37	1.00	0.00	-0.21	0.08

```

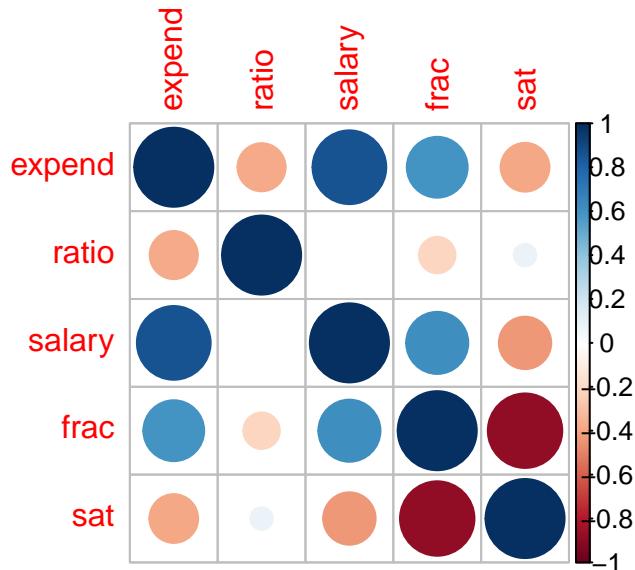
salary    0.87  0.00   1.00  0.62 -0.44
frac      0.59 -0.21   0.62  1.00 -0.89
sat       -0.38  0.08  -0.44 -0.89  1.00

```

```

library(corrplot)
corrplot(C)

```



Because these variables are highly correlated, it doesn't make sense to talk about the effect of increasing teacher salary, while holding expenditure constant, or vice-versa. Notice the standard error on the `salary` line in model `SAT_M4` (which includes expenditures) is more than twice as high as in `SAT_M3`, which did not. This happens because the model is not able to separate the effect of salary from the effect of expenditure, and thus becomes very uncertain of the effect of both, resulting in high standard errors. In addition to reducing the t-statistic, and increasing the p-value, this leads to much wider and less informative confidence intervals associated with the effect of teacher salary.

Confidence intervals for model involving teacher salary, percentage taking the test, and student-to-teacher ratio.

```

confint(SAT_M3)

```

	2.5 %	97.5 %
(Intercept)	968.6691802	1147.1271438
salary	0.5304797	4.5744605
frac	-3.3727807	-2.4539197
ratio	-8.9098147	-0.3690414

Confidence intervals for model with above variables plus expenditure.

```
confint(SAT_M4)
```

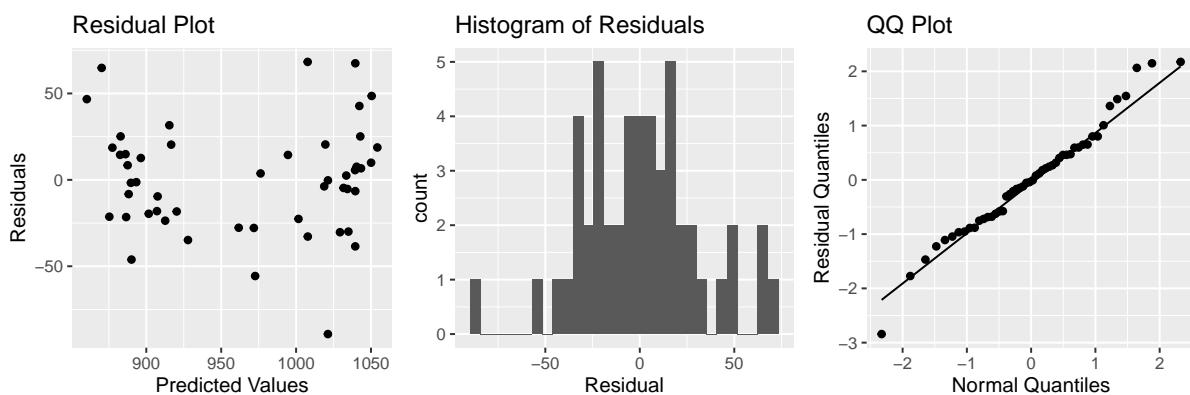
	2.5 %	97.5 %
(Intercept)	939.486374	1152.456698
salary	-3.170247	6.446081
frac	-3.370262	-2.438699
ratio	-10.100417	2.851952
expend	-16.779204	25.704393

Models with highly correlated explanatory variables suffer from **multicollinearity**, which increases standard errors, making the effect of variables harder to discern. When we have explanatory variables that are highly correlated (usually with correlation greater than 0.8), we should pick out just one to include in the model. In this case, we'll stick with teacher salary.

5.3.1.5 Check Model Assumptions

Let's return to the model with salary, ratio, and fraction taking test. We use residual plots to assess model assumptions.

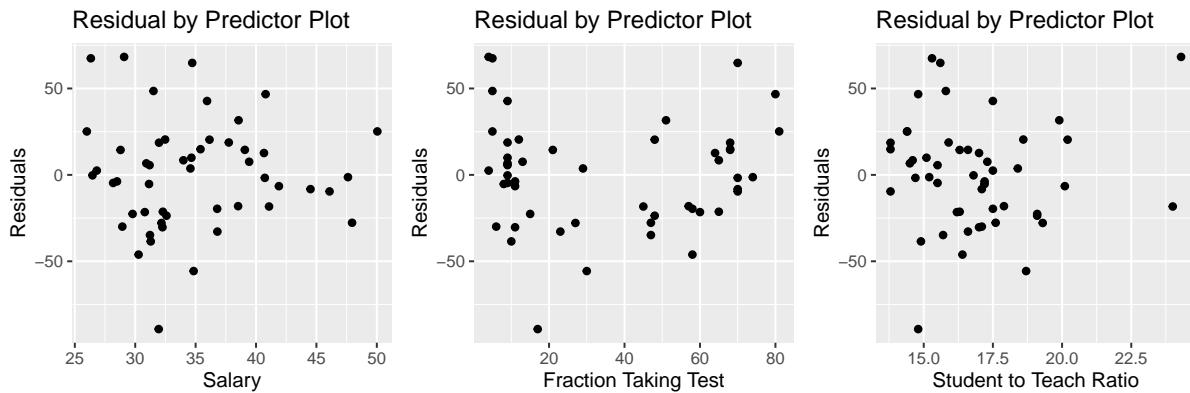
```
P1 <- ggplot(data=data.frame(SAT_M3$residuals), aes(y=SAT_M3$residuals, x=SAT_M3$fitted.value))
P2 <- ggplot(data=data.frame(SAT_M3$residuals), aes(x=SAT_M3$residuals)) + geom_histogram()
P3 <- ggplot(data=data.frame(SAT_M3$residuals), aes(sample = scale(SAT_M3$residuals))) + stat_qq()
grid.arrange(P1, P2, P3, ncol=3)
```



There is some sign of a quadratic trend in the residual plot, creating concern about the linearity assumption.

In models with multiple explanatory variables, it is helpful to also plot our residuals against the explanatory variables to see whether the model is properly accounting for relationships involving each variable. If we see nonlinear trends, we should consider adding a nonlinear function of that explanatory variable.

```
P1 <- ggplot(data=data.frame(SAT_M3$residuals), aes(y=SAT_M3$residuals, x=SAT_M3$model$salary))
P2 <- ggplot(data=data.frame(SAT_M3$residuals), aes(y=SAT_M3$residuals, x=SAT_M3$model$frac))
P3 <- ggplot(data=data.frame(SAT_M3$residuals), aes(y=SAT_M3$residuals, x=SAT_M3$model$ratio))
grid.arrange(P1, P2, P3, ncol=3)
```



There is also a quadratic trend in the plot involving the fraction variable. We might account for this by adding a quadratic term for `frac` to the model.

5.3.1.6 Quadratic Term

```
SAT_M5 <- lm(data=SAT, sat~salary+frac+I(frac^2)+ratio )
summary(SAT_M5)
```

```
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

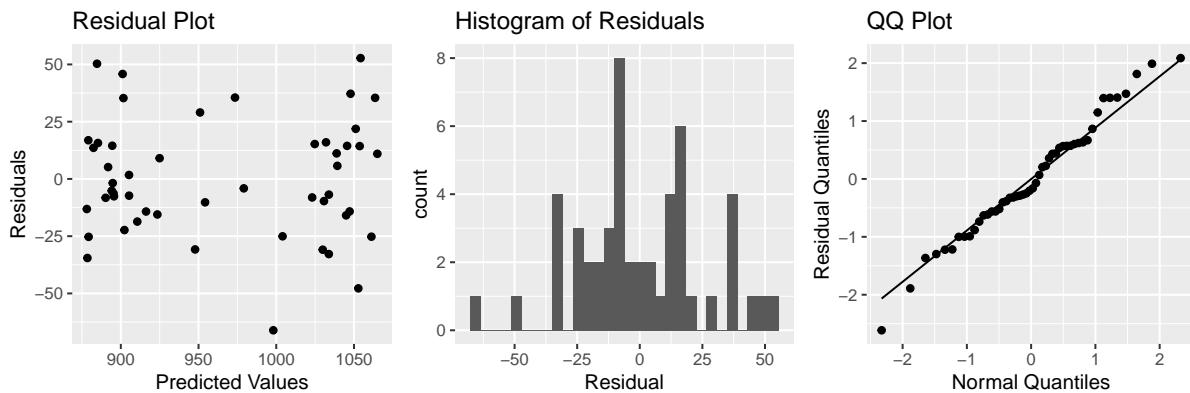
```

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: < 0.00000000000000022

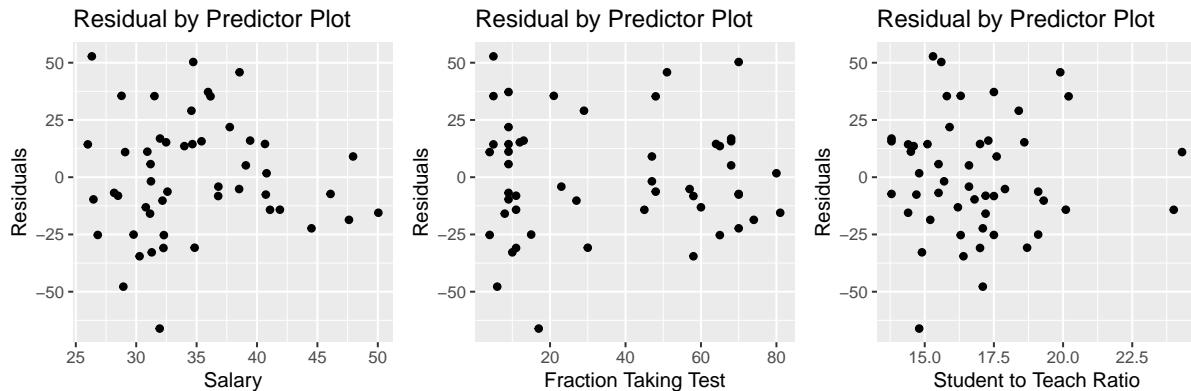
We notice a small p-value associated with the quadratic term, indicating SAT scores do indeed show evidence of a quadratic trend with respect to the percentage of students taking the test.

We now examine residual plots for the model that includes the quadratic term for `frac`.

```
P1 <- ggplot(data=data.frame(SAT_M5$residuals), aes(y=SAT_M5$residuals, x=SAT_M5$fitted.value))
P2 <- ggplot(data=data.frame(SAT_M5$residuals), aes(x=SAT_M5$residuals)) + geom_histogram()
P3 <- ggplot(data=data.frame(SAT_M5$residuals), aes(sample = scale(SAT_M5$residuals))) + stat_qq()
grid.arrange(P1, P2, P3, ncol=3)
```



```
P1 <- ggplot(data=data.frame(SAT_M3$residuals), aes(y=SAT_M3$residuals, x=SAT_M3$model$salary))
P2 <- ggplot(data=data.frame(SAT_M3$residuals), aes(y=SAT_M3$residuals, x=SAT_M3$model$frac))
P3 <- ggplot(data=data.frame(SAT_M3$residuals), aes(y=SAT_M3$residuals, x=SAT_M3$model$ratio))
grid.arrange(P1, P2, P3, ncol=3)
```



The quadratic trend in the residual by predicted plot and second residual by fraction plot appear to have disappeared, suggesting this model has properly accounted for the quadratic trend.

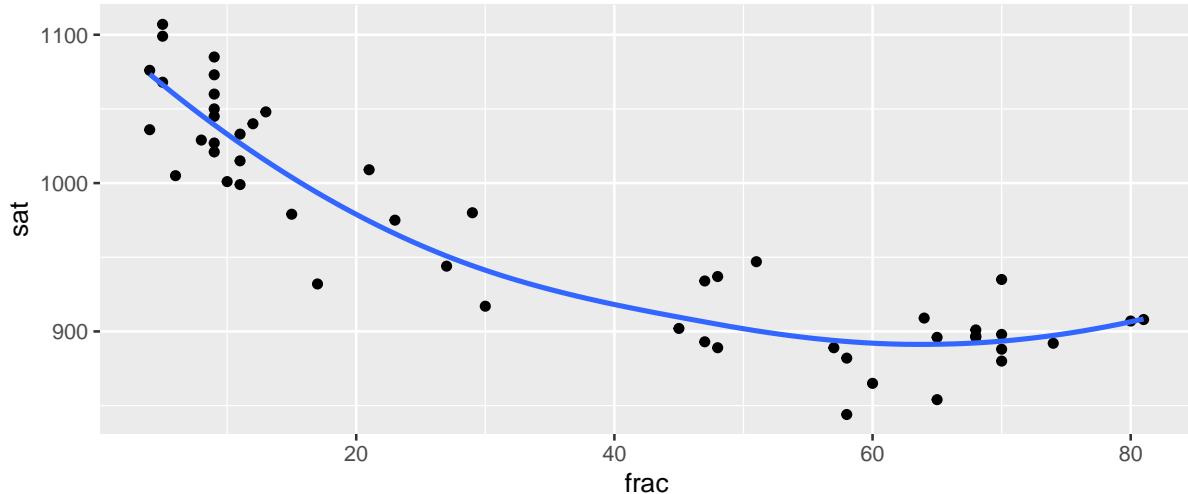
Interpretations for Model with Quadratic Term

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.

```
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. Although the trend does seem to level off in a quadratic way, we wouldn't expect SAT scores to start to increase if more than 80 percent of students took the exam!

5.3.1.7 Account for Region?

So far, we've considered only quantitative explanatory variables. What if we add region of the country to the model.

```
SAT_M6 <- lm(data=SAT, sat~salary+frac+I(frac^2)+ratio + region )
summary(SAT_M6)
```

```
Call:
lm(formula = sat ~ salary + frac + I(frac^2) + ratio + region,
  data = SAT)

Residuals:
    Min      1Q  Median      3Q     Max 
-45.309 -15.407 -1.996  13.852  41.859 

Coefficients:
            Estimate Std. Error t value    Pr(>|t|)    
(Intercept) 1085.46629   37.25849   29.133 < 0.000000000000002 ***
```

```

salary      0.19485   0.86256   0.226      0.822378
frac       -6.12514   0.84697  -7.232      0.00000000679 *** 
I(frac^2)    0.04762   0.01217   3.914      0.000327 *** 
ratio       0.83366   1.99687   0.417      0.678452
regionNE   -11.53616  18.18986  -0.634      0.529384
regionS     -40.07482  11.14606  -3.595      0.000845 *** 
regionW     -15.89290  11.77634  -1.350      0.184386
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Residual standard error: 23.58 on 42 degrees of freedom
 Multiple R-squared: 0.9148, Adjusted R-squared: 0.9007
 F-statistic: 64.46 on 7 and 42 DF, p-value: < 0.0000000000000022

We find that on average, SAT scores were lower in the NE, S, and W regions, compared to the baseline region of MW, though only in the S is the difference large enough to yield a small p-value.

Notice that the effect of teacher salary and student-to-teacher ratio are no longer statistically significant. This happens because now we are only comparing states in the same region of the country. The p-value associated with teacher salary is now testing the null hypothesis “There is no relationship between average teacher salary and SAT score among states in the same region of the country, with the same percentage of students taking the test, and same student to teacher ratio.”

Because we only have 50 states to begin with, breaking down by region results in small sample sizes, which contributes to the large p-values. Furthermore, it is unclear why we would need to account for region here. If our goal is to assess the impact of educational spending on SAT scores, it is probably okay to compare states in different regions of the country. Unless we have some reason for wanting to compare states in the same region, we shouldn’t include region as an explanatory variable (even though including it did raise our R^2 value above 0.9). In general, we should only include variables if they help us address our research question. In this case, it’s not clear that accounting for region helps us better understand the relationship between a state’s investment in education, and its students average SAT scores.

It is important to note that we **should not** decide whether to include a variable based on whether or not it yielded a small p-value. Adding or deleting variables from a model until we get a desired p-value on a variable we’re interested in can lead to **Confirmation bias** (that is choosing our model in a way that intentionally confirms what we expected or hoped to be true), and to detecting spurious correlations that will not be replicable in future studies. This phenomenon, known as **p-hacking** has led to incorrect and unethical conclusions. We should make modeling decisions about which variables to include in a model before looking at the p-values and then draw conclusions based on the results we see, keeping in mind that p-values are only a part of the picture.

Predictions and Intervals

Going back to Model M5, which did not include region, we can make confidence and predictions intervals corresponding to hypothetical states.

```
newstate <- data.frame(salary = 45, frac=0.5, ratio=15)

predict(SAT_M5, newdata = newstate, interval="confidence", conf.level=0.95)

    fit      lwr      upr
1 1116.615 1085.93 1147.3
```

We are 95% confident that the average of average SAT scores among all states with average teacher salary of 45 thousand dollars, where 50% of students take the SAT and having student-to-teacher ratio of 15 is between 1085 and 1147.

```
predict(SAT_M5, newdata = newstate, interval="prediction", conf.level=0.95)

    fit      lwr      upr
1 1116.615 1055.256 1177.973
```

We are 95% confident that an individual state with average teacher salary of 45 thousand dollars, where 50% of students take the SAT and having student-to-teacher ratio of 15 will have an average SAT score between 1055 and 1178.

5.3.2 Modeling Car Price

We'll build a model for the price of a new 2015 car, to help us understand what factors are related to the price of a car.

```
data(Cars2015)
Cars2015 <- Cars2015 %>% rename(Price=LowPrice)
Cars2015 <- Cars2015 %>% select(-HighPrice)
glimpse(Cars2015)

Rows: 110
Columns: 19
$ 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-
$ Price     <dbl> 12.270, 14.745, 13.990, 12.995, 14.180, 16.495, 16.170, 19.3-
$ Drive     <fct> 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, 3.9, 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-

```

Exploratory Analysis

We'll look at a summary of the categorical variables in the dataset.

```
Cars_Cat <- select_if(Cars2015, is.factor)
summary(Cars_Cat)
```

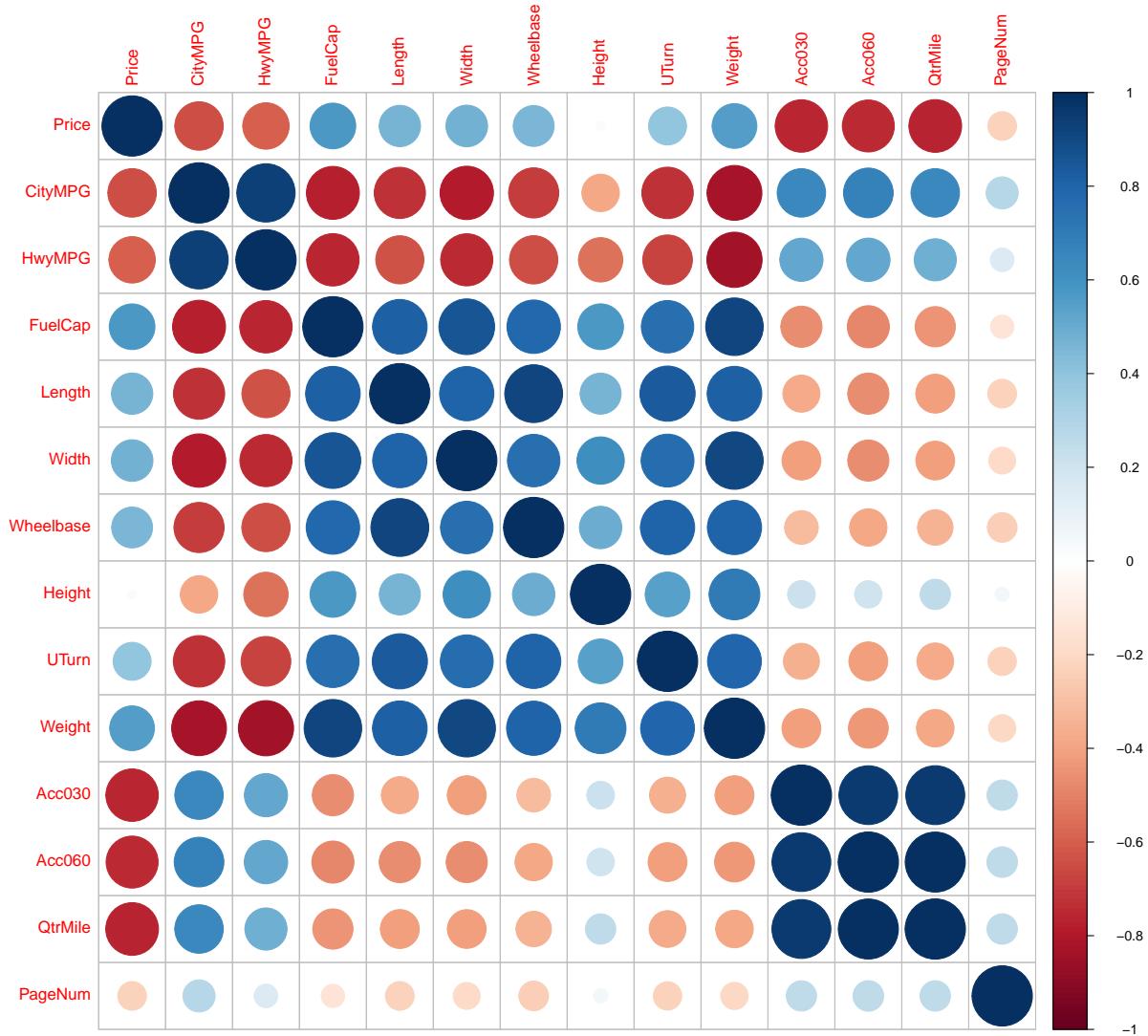
	Make	Model	Type	Drive	Size
Chevrolet:	8	CTS : 2	7Pass :15	AWD:25	Large :29
Ford :	7	2 Touring : 1	Hatchback:11	FWD:63	Midsized:34
Hyundai :	7	200 : 1	Sedan :46	RWD:22	Small :47
Toyota :	7	3 i Touring: 1	Sporty :11		
Audi :	6	3 Series GT: 1	SUV :18		
Nissan :	6	300 : 1	Wagon : 9		
(Other) :	69	(Other) :103			

We examine the correlation matrix of quantitative variables.

```
Cars_Num <- select_if(Cars2015, is.numeric)
C <- cor(Cars_Num, use = "pairwise.complete.obs")
round(C,2)
```

	Price	CityMPG	HwyMPG	FuelCap	Length	Width	Wheelbase	Height	UTurn	
Price	1.00	-0.65	-0.59	0.57	0.47	0.48		0.46	0.02	0.40
CityMPG	-0.65	1.00	0.93	-0.77	-0.72	-0.78		-0.69	-0.39	-0.73
HwyMPG	-0.59	0.93	1.00	-0.75	-0.64	-0.75		-0.64	-0.54	-0.68
FuelCap	0.57	-0.77	-0.75	1.00	0.82	0.85		0.79	0.58	0.76
Length	0.47	-0.72	-0.64	0.82	1.00	0.81		0.92	0.46	0.84
Width	0.48	-0.78	-0.75	0.85	0.81	1.00		0.76	0.62	0.77
Wheelbase	0.46	-0.69	-0.64	0.79	0.92	0.76		1.00	0.49	0.81
Height	0.02	-0.39	-0.54	0.58	0.46	0.62		0.49	1.00	0.55
UTurn	0.40	-0.73	-0.68	0.76	0.84	0.77		0.81	0.55	1.00
Weight	0.55	-0.83	-0.84	0.91	0.82	0.91		0.81	0.71	0.80
Acc030	-0.76	0.64	0.51	-0.47	-0.38	-0.41		-0.31	0.21	-0.36
Acc060	-0.74	0.68	0.52	-0.49	-0.47	-0.46		-0.38	0.21	-0.41
QtrMile	-0.76	0.65	0.49	-0.45	-0.42	-0.41		-0.35	0.25	-0.37
PageNum	-0.23	0.28	0.15	-0.15	-0.23	-0.20		-0.24	0.06	-0.22
	Weight	Acc030	Acc060	QtrMile	PageNum					
Price	0.55	-0.76	-0.74	-0.76	-0.23					
CityMPG	-0.83	0.64	0.68	0.65	0.28					
HwyMPG	-0.84	0.51	0.52	0.49	0.15					
FuelCap	0.91	-0.47	-0.49	-0.45	-0.15					
Length	0.82	-0.38	-0.47	-0.42	-0.23					
Width	0.91	-0.41	-0.46	-0.41	-0.20					
Wheelbase	0.81	-0.31	-0.38	-0.35	-0.24					
Height	0.71	0.21	0.21	0.25	0.06					
UTurn	0.80	-0.36	-0.41	-0.37	-0.22					
Weight	1.00	-0.41	-0.43	-0.39	-0.20					
Acc030	-0.41	1.00	0.95	0.95	0.25					
Acc060	-0.43	0.95	1.00	0.99	0.26					
QtrMile	-0.39	0.95	0.99	1.00	0.26					
PageNum	-0.20	0.25	0.26	0.26	1.00					

```
library(corrplot)
C <- corrplot(C)
```



Note the high correlation between many variables in the dataset. We'll need to be careful about multicollinearity.

5.3.2.1 Acc. and Qrt. Mile Time

We saw in Section 5.2 that it was better to model $\log(\text{Price})$ than price itself, so we'll continue modeling $\log\text{price}$ here.

Model Using Just Acceleration Time

First, we fit a model using only the time it takes to accelerate from 0 to 60 mph as an explanatory variable.

```
Cars_M1 <- lm(data=Cars2015, log(Price) ~ Acc060)
summary(Cars_M1)
```

Call:
lm(formula = log(Price) ~ 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: < 0.0000000000000002

Confidence Interval for Effect of Acceleration Time:

```
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%.

Model Using Just Quarter Mile Time

Now, let's fit a different model using only the time it takes to drive a quarter mile as an explanatory variable.

```
Cars_M2 <- lm(data=Cars2015, log(Price) ~ QtrMile)
summary(Cars_M2)
```

```

Call:
lm(formula = log(Price) ~ 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: < 0.0000000000000022

```

Confidence Interval for Effect of 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.

Model Using Both Acceleration and Quarter Mile Time

```
Cars_M3 <- lm(data=Cars2015, log(Price) ~ QtrMile + Acc060)
summary(Cars_M3)
```

```

Call:
lm(formula = log(Price) ~ QtrMile + Acc060, 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 *** 
Acc060       -0.00018    0.00012  -1.50    0.1348    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 

Residual standard error: 0.275 on 107 degrees of freedom
Multiple R-squared:  0.6385,    Adjusted R-squared:  0.6351 
F-statistic: 190.7 on 2 and 107 DF,  p-value: < 0.0000000000000022

```

```
-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: < 0.0000000000000022

Confidence Intervals from 2-variable Model

```
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.

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.

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.

5.3.2.2 Adding Weight to Model

We could use either quarter mile time or acceleration time as an explanatory variable, but we shouldn't use both. We'll proceed with quarter mile time.

```
Cars_M4 <- lm(data=Cars2015, log(Price) ~ QtrMile + Weight)  
summary(Cars_M4)
```

Call:

```
lm(formula = log(Price) ~ 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

```

R^2 went up from 0.64 to 0.76!

We might consider adding an interaction term between quarter mile time and weight. This would mean that we think the effect of quarter mile time on price of a car is different for heavier cars than for lighter cars. It's not clear to me why that would be the case.

```
Cars_M5 <- lm(data=Cars2015, log(Price) ~ QtrMile * Weight)
summary(Cars_M5)
```

```

Call:
lm(formula = log(Price) ~ 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. There doesn't seem to be much reason to complicate the model by adding an interaction term.

5.3.2.3 Adding More Variables

We'll consider adding highway MPG to the model.

```
Cars_M6 <- lm(data=Cars2015, log(Price) ~ QtrMile + Weight + HwyMPG)
summary(Cars_M6)
```

Call:

```
lm(formula = log(Price) ~ QtrMile + Weight + HwyMPG, data = Cars2015)
```

Residuals:

Min	1Q	Median	3Q	Max
-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

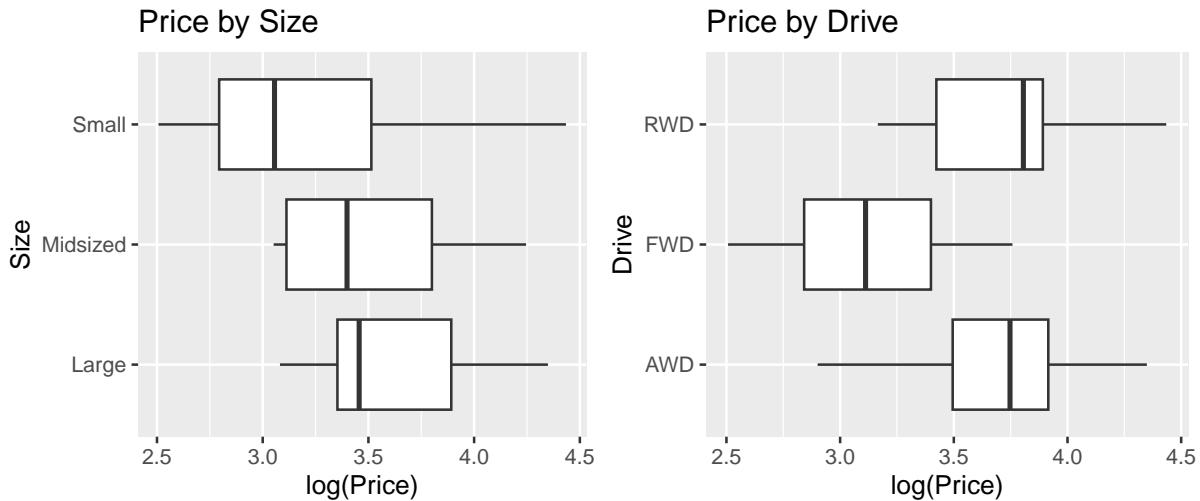
Multiple R-squared: 0.7732, Adjusted R-squared: 0.7668

F-statistic: 120.5 on 3 and 106 DF, p-value: < 0.0000000000000022

HwyMPG doesn't make change R^2 much, and has a high correlation with weight. Let's not include it.

Next, we'll consider adding categorical explanatory variables Size, and Drive.

```
P1 <- ggplot(data=Cars2015, aes(x=log(Price), y=Size)) + geom_boxplot() + ggtitle("Price by Size")
P2 <- ggplot(data=Cars2015, aes(x=log(Price), 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.

```
Cars_M7 <- lm(data=Cars2015, log(Price) ~ QtrMile + Weight + Drive)
summary(Cars_M7)
```

Call:

```
lm(formula = log(Price) ~ 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

We found evidence of differences in price between front-wheel drive and rear-wheel drive, compared to all wheel drive cars.

Next, we'll explore adding size to the model.

```
Cars_M8 <- lm(data=Cars2015, log(Price) ~ QtrMile + Weight + Drive + Size)
summary(Cars_M8)
```

```
Call:
lm(formula = log(Price) ~ QtrMile + Weight + Drive + Size, data = Cars2015)

Residuals:
    Min      1Q  Median      3Q     Max 
-0.71092 -0.12126  0.01355  0.11831  0.44439 

Coefficients:
            Estimate Std. Error t value     Pr(>|t|)    
(Intercept) 5.66594310  0.37169625 15.243 < 0.0000000000000002 ***  
QtrMile     -0.19256547  0.02000711 -9.625 0.00000000000000505 ***  
Weight       0.00023978  0.00004101  5.847 0.00000059416930182 ***  
DriveFWD    -0.21598794  0.05780323 -3.737 0.000306 ***  
DriveRWD    -0.15259183  0.06410851 -2.380 0.019142 *    
SizeMidsize  0.04699095  0.06271499  0.749 0.455398      
SizeSmall    0.08875861  0.08105810  1.095 0.276071      
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2085 on 103 degrees of freedom
Multiple R-squared:  0.8018,   Adjusted R-squared:  0.7903 
F-statistic: 69.46 on 6 and 103 DF,  p-value: < 0.0000000000000022
```

Adding size barely increased R^2 at all. We find no evidence of differences in price between the three sizes, after accounting for the other variables.

Note: Information about car size is already being taken into account through the `Weight` variable.

We could keep looking at other variables to add, but at this point, we have a model that gives us a good sense of the factors related to price of a car, capturing 80% of total variability in car price, and is still easy to interpret.

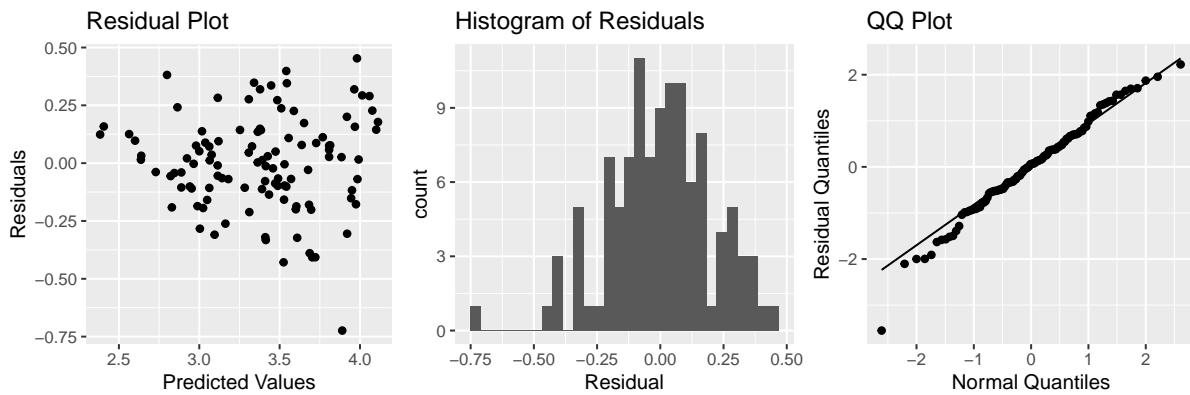
For our research purposes, this model is good enough.

5.3.2.4 Check of Model Assumptions

We'll use residuals to check the model assumptions.

Residual by Predicted Plot, Histogram of Residuals, and Normal Quantile-Quantile Plot

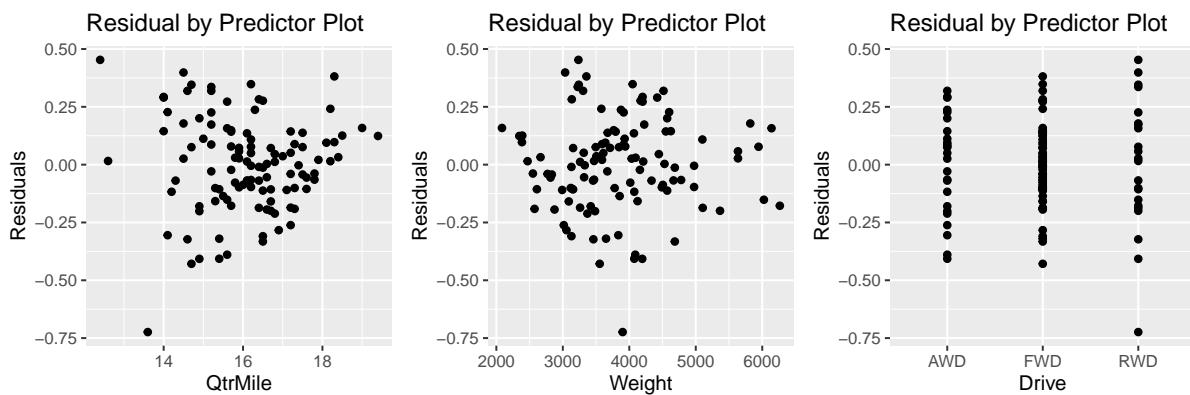
```
P1 <- ggplot(data=data.frame(Cars_M7$residuals), aes(y=Cars_M7$residuals, x=Cars_M7$fitted.value))
P2 <- ggplot(data=data.frame(Cars_M7$residuals), aes(x=Cars_M7$residuals)) + geom_histogram()
P3 <- ggplot(data=data.frame(Cars_M7$residuals), aes(sample = scale(Cars_M7$residuals))) + stat_qq()
grid.arrange(P1, P2, P3, ncol=3)
```



There is slight concern about constant variance, but otherwise, the model assumptions look good.

Residual by Predictor Plots

```
P1 <- ggplot(data=data.frame(Cars_M7$residuals), aes(y=Cars_M7$residuals, x=Cars_M7$model$QtrMile))
P2 <- ggplot(data=data.frame(Cars_M7$residuals), aes(y=Cars_M7$residuals, x=Cars_M7$model$Weight))
P3 <- ggplot(data=data.frame(Cars_M7$residuals), aes(y=Cars_M7$residuals, x=Cars_M7$model$Drive))
grid.arrange(P1, P2, P3, ncol=3)
```



These plots don't raise any concerns.

5.3.2.5 Coefficients and Exponentiation

The model coefficients are shown below.

```
Cars_M7$coefficients
```

(Intercept)	QtrMile	Weight	DriveFWD	DriveRWD
5.8140678915	-0.1900743859	0.0002049586	-0.2240322171	-0.1388439916

Since we used a log transformation, we should interpret e^{b_j} rather than b_j itself.

```
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.

5.3.2.6 Confidence and Prediction Intevals

We'll use our model to estimate the average price with the following characteristics, and also to predict the price of a new car with the given characteristics.

```
newcar <- data.frame(QtrMile = 18, Weight=2400, Drive = "AWD")
```

This is an interval for $\log(\text{Price})$.

```
predict(Cars_M7, newdata=newcar, interval="confidence", level=0.95)
```

```
fit      lwr      upr  
1 2.884629 2.741423 3.027836
```

Exponentiating, we obtain

```
exp(predict(Cars_M7, newdata=newcar, interval="confidence", level=0.95))
```

```
fit      lwr      upr  
1 17.89693 15.50904 20.65249
```

We are 95% Confident that the average price of all new 2015 cars that weigh 2400 lbs, drive a quarter mile in 18 seconds on a fast track, and have all wheel drive is between 15.5 thousand and 20.7 thousand dollars.

Next, we calculate a prediction interval for an individual car with these characteristics.

```
exp(predict(Cars_M7, newdata=newcar, interval="prediction", level=0.95))
```

```
fit      lwr      upr  
1 17.89693 11.57309 27.6763
```

We are 95% Confident that the price of an individual new 2015 car that weighs 2400 lbs, drives a quarter mile in 18 seconds on a fast track, and has all wheel drive will be between 11.6 thousand and 27.7 thousand dollars.

5.3.2.7 Model Building Summary

Consider the following when building a model for the purpose of interpreting parameters and understanding and drawing conclusions about a population or process.

- Model driven by research question
- Include variables of interest
- Include potential confounders (like in SAT example)
- Avoid including highly correlated explanatory variables

- Avoid messy transformations and interactions where possible
- Use residual plots to assess appropriateness of model assumptions
- Aim for high R^2 but not highest
- Aim for model complex enough to capture nature of data, but simple enough to give clear interpretations

6 Logistic Regression

Learning Outcomes:

37. Identify situations where it is appropriate to use a logistic regression model.
38. Calculate probabilities and odds using logistic regression.
39. Interpret logistic regression coefficients in context.
40. Implement logistic regression models in R.

6.1 Logistic Regression

6.1.1 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). Problems with categorical response variables are sometimes called **classification** problems, while problems with numeric response variables are sometimes called **regression** problems.

6.1.2 Credit Card Dataset

We'll work with 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)
```

```

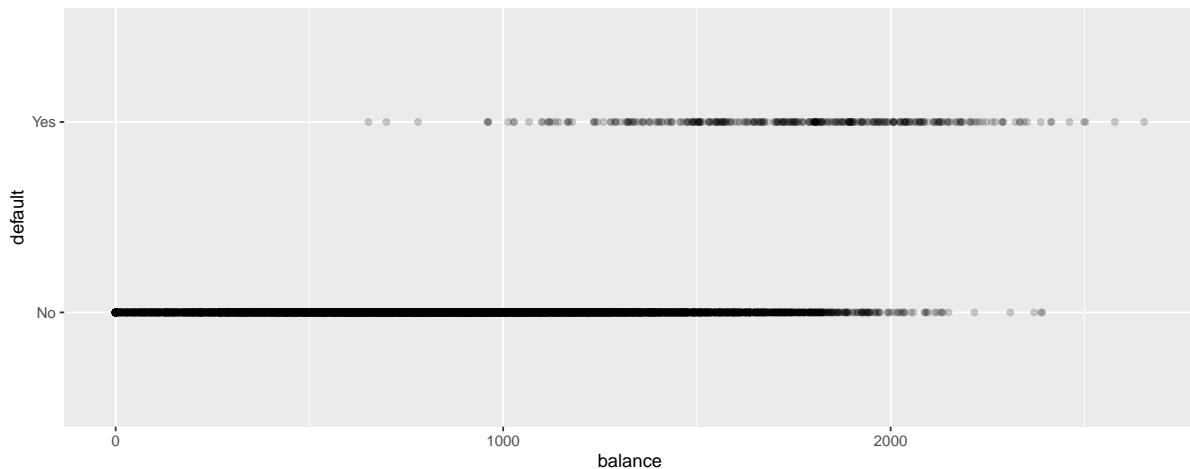
default    student      balance        income
No :9667   No :7056   Min.   : 0.0   Min.   : 772
Yes: 333   Yes:2944  1st Qu.:481.7  1st Qu.:21340
            Median :823.6  Median :34553
            Mean   :835.4  Mean   :33517
            3rd Qu.:1166.3 3rd Qu.:43808
            Max.   :2654.3  Max.   :73554

```

Default and Balance

The plot displays each person's credit card balance on the x-axis, and whether or not they defaulted (a 0 or 1) on the y-axis.

```
ggplot(data=Default, aes(y=default, x=balance)) + geom_point(alpha=0.2)
```

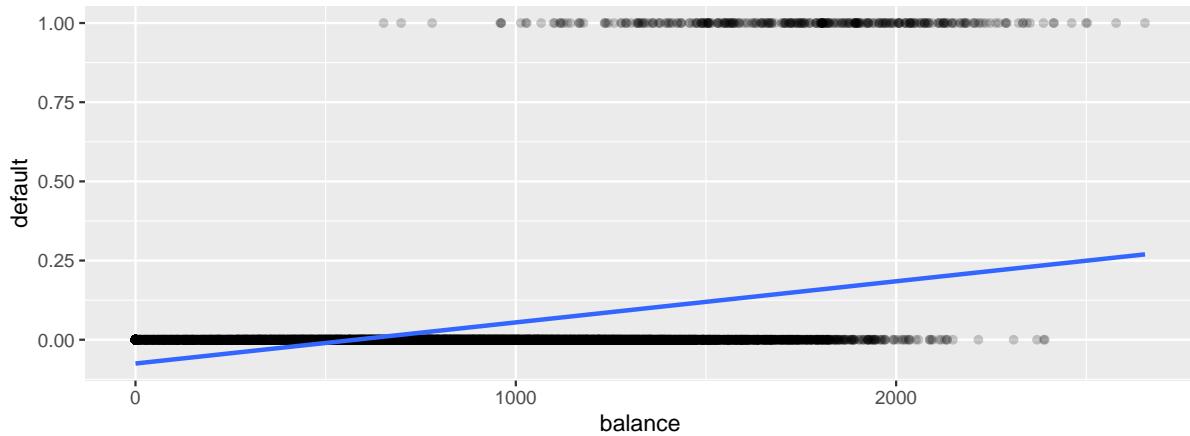


We see that defaults are rare when the balance is less than \$1,000, and more common for balances above \$2,000.

We'll first try fitting a linear regression model to the data to try to estimate the probability of a person defaulting on a loan, using the size of their balance as the explanatory variable.

```
#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(method="lm")
```

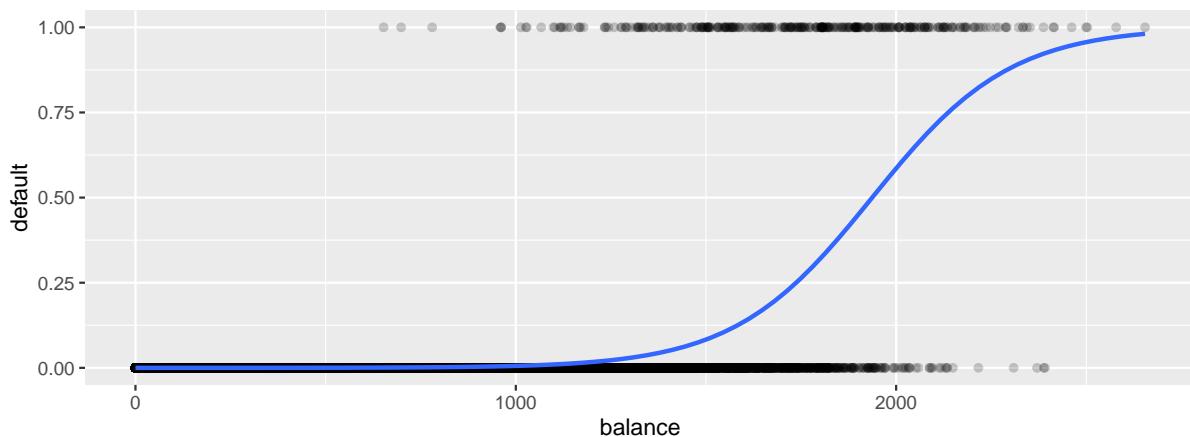


There are a lot of problems with this model!

It allows the estimated probability of default to be negative. It also assumes a linear trend that doesn't seem to fit the data very well.

A sigmoidal curve, like the one below, seems like a better model for default probabilities. This curve stays between 0 and 1, and its curved nature seems like a better fit for the 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))
```



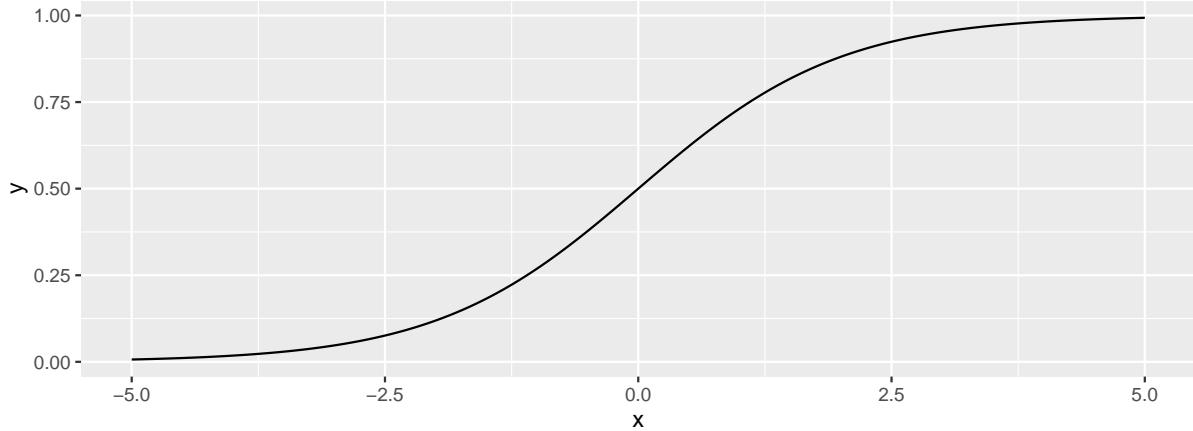
6.1.3 Logistic Regression Model

A logistic regression model uses a sigmoidal curve like the one we saw to model default probabilities, using balance as an explanatory variable.

The model makes use of the function

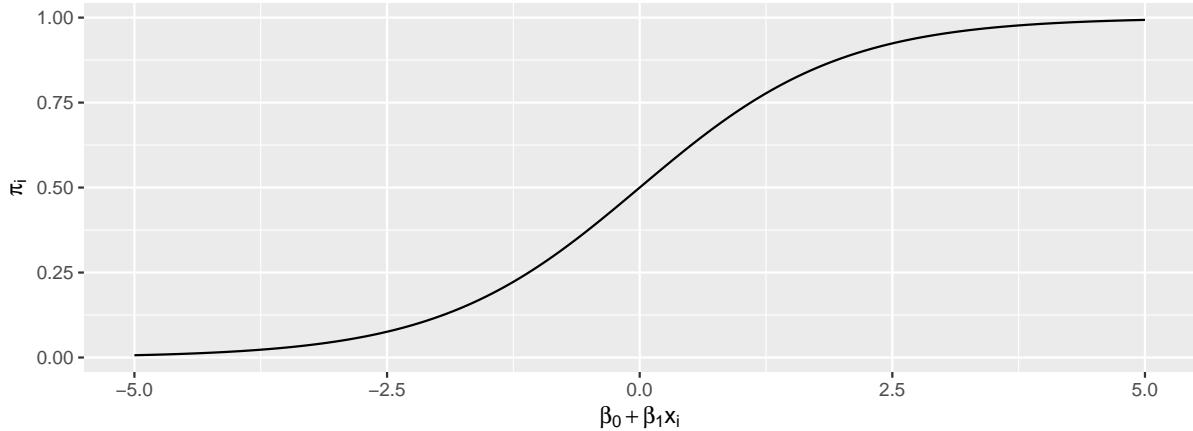
$$f(x) = \frac{e^x}{1 + e^x},$$

whose graph is shown below. This function is called an **inverse logit** function.



Starting with our linear model $E(Y_i) = \beta_0 + \beta_1 x_{i1}$, we need to transform $\beta_0 + \beta_1 x_{i1}$ into the interval $(0,1)$.

- Let $\pi_i = \frac{e^{\beta_0 + \beta_1 x_{i1}}}{1 + e^{\beta_0 + \beta_1 x_{i1}}}$.
- Then $0 \leq \pi_i \leq 1$, and π_i represents an estimate of $P(Y_i = 1)$.
- This function maps the values of $\beta_0 + \beta_1 x_{i1}$ into the interval $(0,1)$.



The **logistic regression model** assumes that:

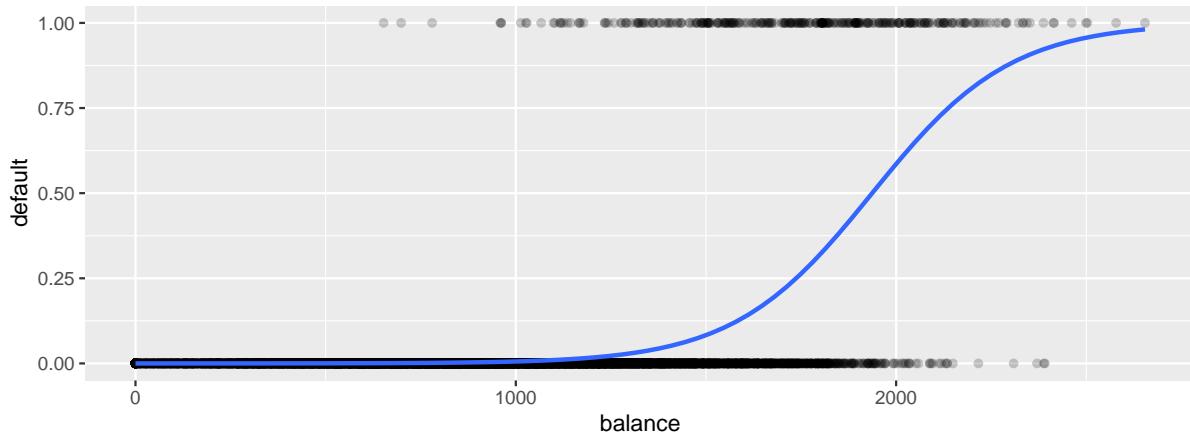
- $Y_i \in \{0, 1\}$

- $E(Y_i) = P(Y_i = 1) = \pi_i = \frac{e^{\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}}}{1 + e^{\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}}}$ i.e. $\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip} = \log\left(\frac{\pi_i}{1 - \pi_i}\right)$. (This is called the logit function and can be written $\text{logit}(\pi_i)$).

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.

We fit the logistic curve to the 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))
```



To fit the logistic regression model in R, we use the function `glm`, instead of `lm`. The function is specified the same way as before, and we add `family = binomial(link = "logit")`.

```
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)
```

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

The regression equation is:

$$P(\text{Default}) = \hat{\pi}_i = \frac{e^{-10.65+0.0055 \times \text{balance}}}{1 + e^{-10.65+0.0055 \times \text{balance}}}$$

Predictions

- 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 $\frac{e^{-10.65+0.0055(2000)}}{1+e^{-10.65+0.0055(2000)}} \approx 0.59$

We confirm these, using the `predict` command 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

```

6.1.3.1 Where do 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**. We won't say anything more about that topic here, but it is a prominent technique, widely used in statistic modeling. It is explored in more detail in advanced statistics courses, such as STAT 445:Mathematical Statistics.

6.2 Interpreting Coefficients in Logistic Regression

6.2.1 Odds and Odds Ratio

For an event with probability p , the **odds** of the event occurring are $\frac{p}{1-p}$.

Examples: 1. The odds of a fair coin landing heads are $\frac{0.5}{1-0.5} = 1$, sometimes written 1:1.

2. The odds of a fair 6-sided die landing on a 1 are $\frac{1/6}{1-1/6} = \frac{1}{5}$, sometimes written 1:5.

In the credit card example, the odds of default are:

- For a \$1,000 balance - odds of default are $\frac{0.005752145}{1-0.005752145} \approx 1 : 173$.
- For a \$1,500 balance - odds of default are $\frac{0.08294762}{1-0.08294762} \approx 1 : 11$.
- For a \$2,000 balance - odds of default are $\frac{0.5857694}{1-0.5857694} \approx 1.414 : 1$.

The quantity $\frac{\pi_i}{\pi_j}$ is called the **odds ratio** and represents the odds ratio of a default for user i , compared to user j .

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.

6.2.2 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 $\log\left(\frac{\pi_j}{1-\pi_j}\right) = \beta_0 + \beta_1 x$, and $\log\left(\frac{\pi_j}{1-\pi_j}\right) = \beta_0 + \beta_1(x + 1)$.

$$\log\left(\frac{\frac{\pi_j}{1-\pi_j}}{\frac{\pi_i}{1-\pi_i}}\right) = \log\left(\frac{\pi_j}{1-\pi_j}\right) - \log\left(\frac{\pi_i}{1-\pi_i}\right) = \beta_0 + \beta_1(x + 1) - (\beta_0 + \beta_1(x)) = \beta_1.$$

For every 1-unit increase in x we expect the log odds of “success” to multiply by a factor of β_1 .

For every 1-unit increase in x we expect the odds of “success” to multiply by a factor of e^{β_1} .

Interpretation in Credit Card Example

$$b_1 = 0.0055$$

For each 1-dollar increase in balance on the credit card., the odds of default are estimated to multiply by $e^{0.0055} \approx 1.0055$.

That is, for each additional dollar on the card balance, the odds of default are estimated to increase by 0.55%

For each increase of d dollars in credit card balance, odds of default are estimated to multiply by a factor of $e^{0.0055d}$.

For every \$1,000 increase in balance, the odds of default are expected to multiply by a factor of $e^{0.0055 \times 1000} \approx 244$.

Thus, 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. This matches our result when we actually calculated out the probabilities and odds.

Hypothesis test for $\beta_1 = 0$

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.

Confidence Intervals for β_1

```
confint(CCDefault_M, level = 0.95)
```

	2.5 %	97.5 %
(Intercept)	-11.383288936	-9.966565064
balance	0.005078926	0.005943365

We are 95% confident that for each 1 dollar increase in credit card balance, the odds of default are expected to multiply 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](#).

6.3 Multiple Logistic Regression

6.3.1 Logistic Regression Models with Multiple Explanatory Variables

We can also perform logistic regression in situations where there are multiple explanatory variables. We'll estimate probability of default, using both balance and whether or not the person is a student (a categorical variable) as explanatory variables.

```
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)
```

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

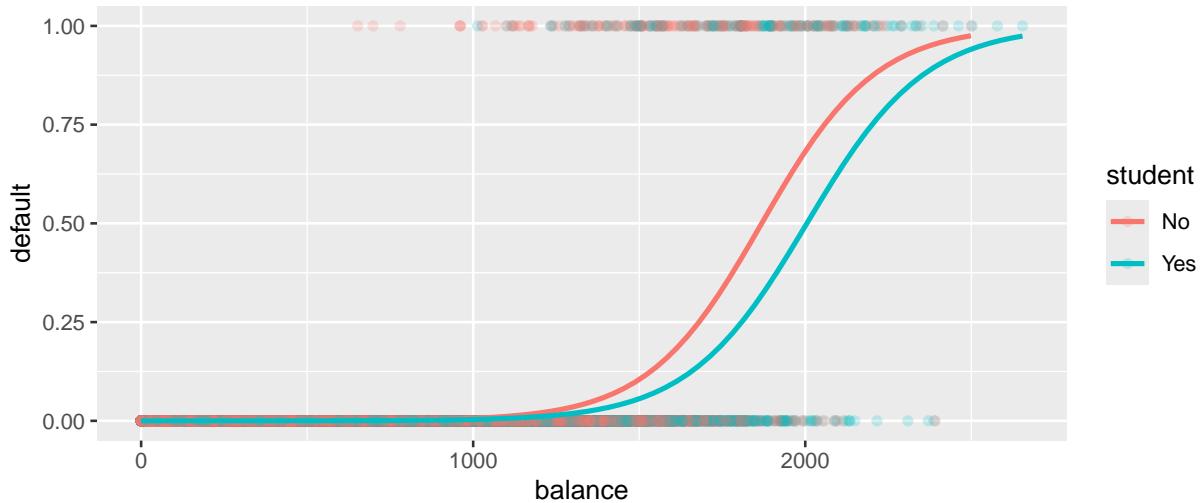
(Dispersion parameter for binomial family taken to be 1)

```
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

The following graph gives an illustration of the model.

```
ggplot(data=Default, aes(y=default, x= balance, color=student)) + geom_point(alpha=0.2) + stat
```



The regression equation is:

$$P(\text{Default}) = \hat{\pi}_i = \frac{e^{-10.75 + 0.005738 \times \text{balance} - 0.7149 \times \text{student}_i}}{1 + e^{-10.75 + 0.005738 \times \text{balance} - 0.7149 \times \text{student}_i}}$$

- For each 1 dollar increase in balance, the odds of default are estimated to multiply by a factor $e^{0.005738} \approx 1.00575$, assuming whether or not the person is a student is held constant. Thus, the estimated odds of default increase by about 0.5%, for each 1-dollar increase in balance..
- For every \$100 increase in balance, the odds of default are estimated to multiply by $e^{0.005738 \times 100} \approx 1.775$, assuming whether or not the person is a student is held constant. 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.

Hypothesis Tests in Multiple Logistic Regression Model

- Since the p-value associated with **balance** is very small, there is strong evidence of a relationship between balance and odds of default, after accounting for whether or not the person is a student.
- Since the p-value associated with **StudentYes** is very small, there is strong evidence that students are less likely to default than nonstudents, provided the balance on the card is the same.

6.3.2 Multiple Logistic Regression Model with Interaction

The previous model assumes the effect of balance on default probability is the same for students as for nonstudents. If we suspect that the effect of having a larger balance might be different for students than for nonstudents, then we could use a model with interaction between the balance and student variables.

```
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)
```

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

Interpretations for Logistic Model with Interaction

- The regression equation is:

$$P(\text{Default}) = \hat{\pi}_i = \frac{e^{-10.87 + 0.0058 \times \text{balance} - 0.35 \times I_{\text{student}} - 0.0002 \times \text{balance} \times I_{\text{student}}}}{1 + e^{-10.87 + 0.0058 \times \text{balance} - 0.35 \times I_{\text{student}} - 0.0002 \times \text{balance} \times I_{\text{student}}}}$$

Equation for Students

$$P(\text{Default}) = \hat{\pi}_i = \frac{e^{-10.52+0.0056 \times \text{balance}}}{1 + e^{-10.52+0.0056 \times \text{balance}}}$$

Assuming a person is a student, for every \$100 increase in balance, the odds of default are expected to multiply by a factor of $e^{0.0056 \times 100} = 1.75$, a 75% increase.

Equation for Non-Students

$$P(\text{Default}) = \hat{\pi}_i = \frac{e^{-10.87+0.0058 \times \text{balance}}}{1 + e^{-10.87+0.0058 \times \text{balance}}}$$

Assuming a person is a student, for every \$100 increase in balance, the odds of default are expected to multiply by a factor of $e^{0.0058 \times 100} = 1.786$, a 78.6% increase.

- 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.

6.3.3 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 be a factor of e^{β_j} 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^{β_j} 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.)

7 Predictive Modeling

Learning Outcomes:

41. Explain how model complexity relates to training and test error, prediction variance and bias, and overfitting.
42. Explain how to use cross-validation in model selection.
43. Explain how complexity parameters associated with ridge regression, decision trees, and splines impact variance, bias, and likelihood of overfitting.
44. Make predictions from a decision tree.
45. Calculate classification accuracy, sensitivity, specificity, confusion matrix, and receiver operating characteristic curves. Identify ethical considerations associated with predictive models in context.

7.1 Modeling for Prediction

7.1.1 Overview

We've previously learned how to build models for the purpose of interpretation, when our primary focus is on understanding relationships between variables in the model. In this chapter, we'll examine how to build models for situations when we are not interested in understanding relationships between variables, and instead care only about making the most accurate predictions possible.

We've seen that when we model for interpretation, we encounter a tradeoff between model complexity and interpretability. We wanted to choose a model that is complex enough to reasonably approximate the structure of the underlying data, but at the same time, not so complicated that it becomes hard to interpret. When modeling for prediction, we don't need to worry about interpretability, which can sometimes make more complex models more desirable. Nevertheless, we'll encounter a different kind of tradeoff, involving model complexity, that we'll have to think about, and we'll see that more complex models do not always lead to better predictions.

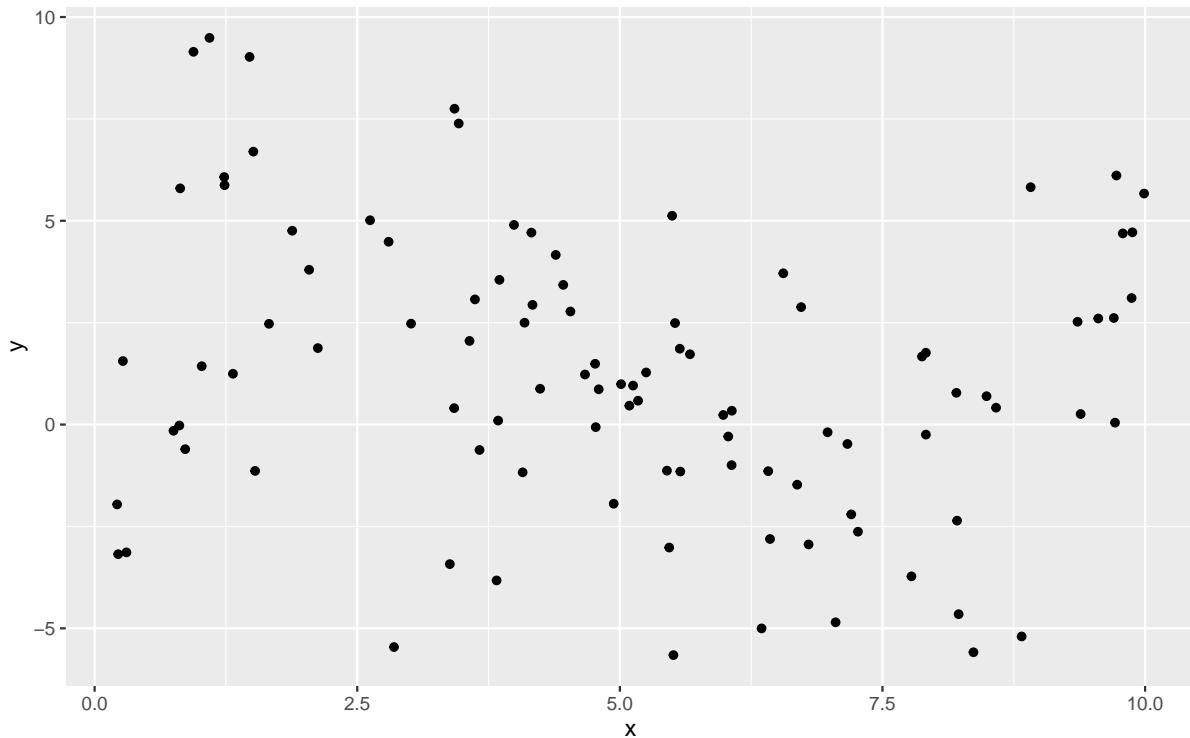
Predictive Modeling Vocabulary

- 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.

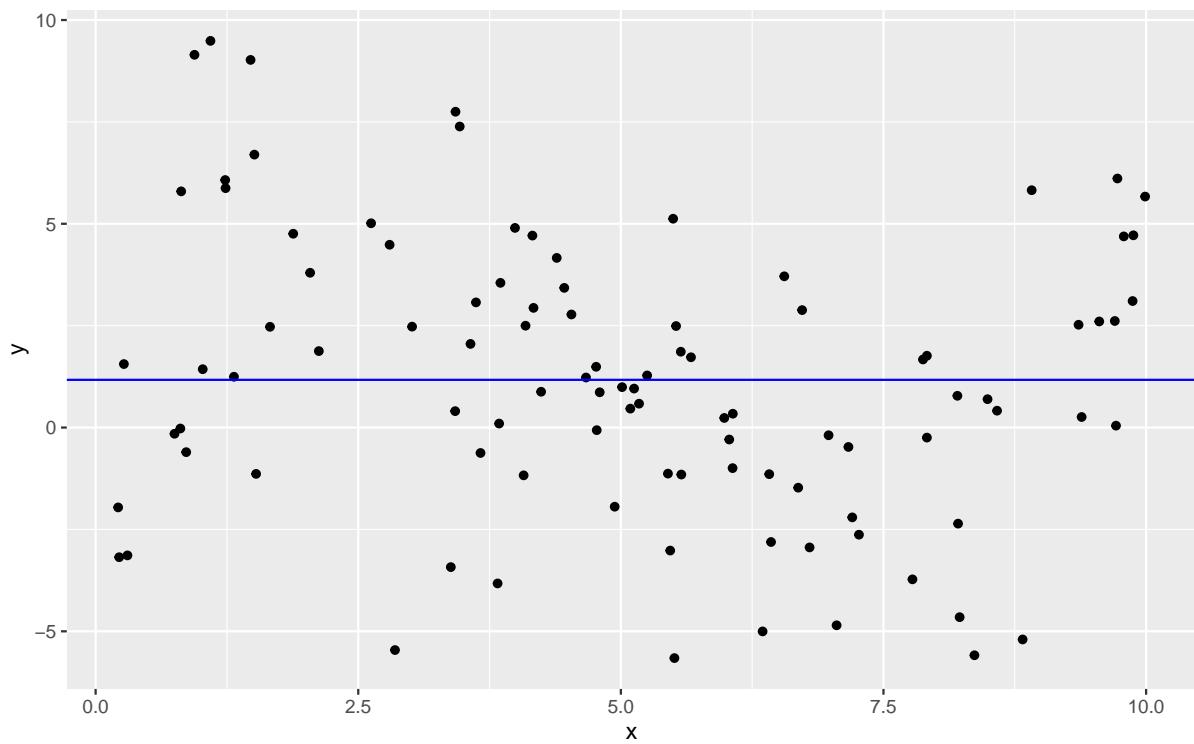
7.1.2 Illustration of Predictive Modeling

The illustration shows observations from a simulated dataset consisting of 100 observations of a single explanatory variable x , and response variable y . We want to find a model that captures the trend in the data and will be best able to predict new values of y , for given x .

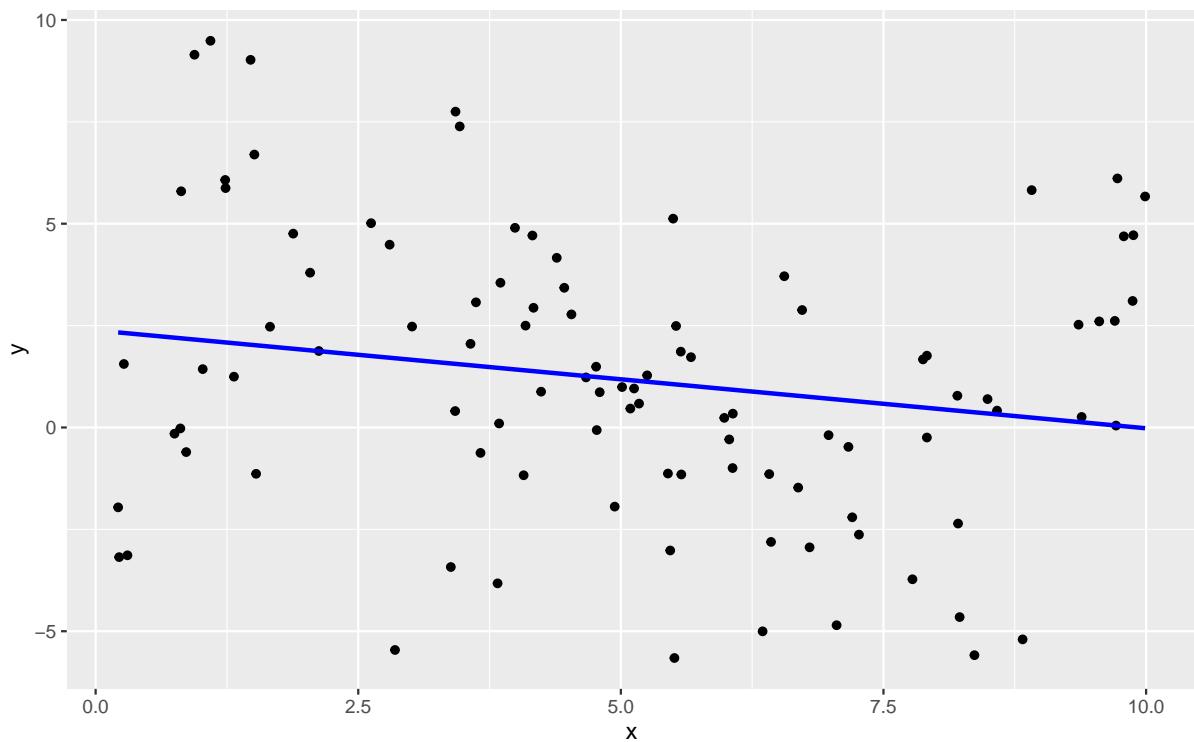


We'll fit several different polynomial models to the data, increasing in complexity from the most simple model we could possibly use, a constant model, to a very complex eighth degree polynomial model.

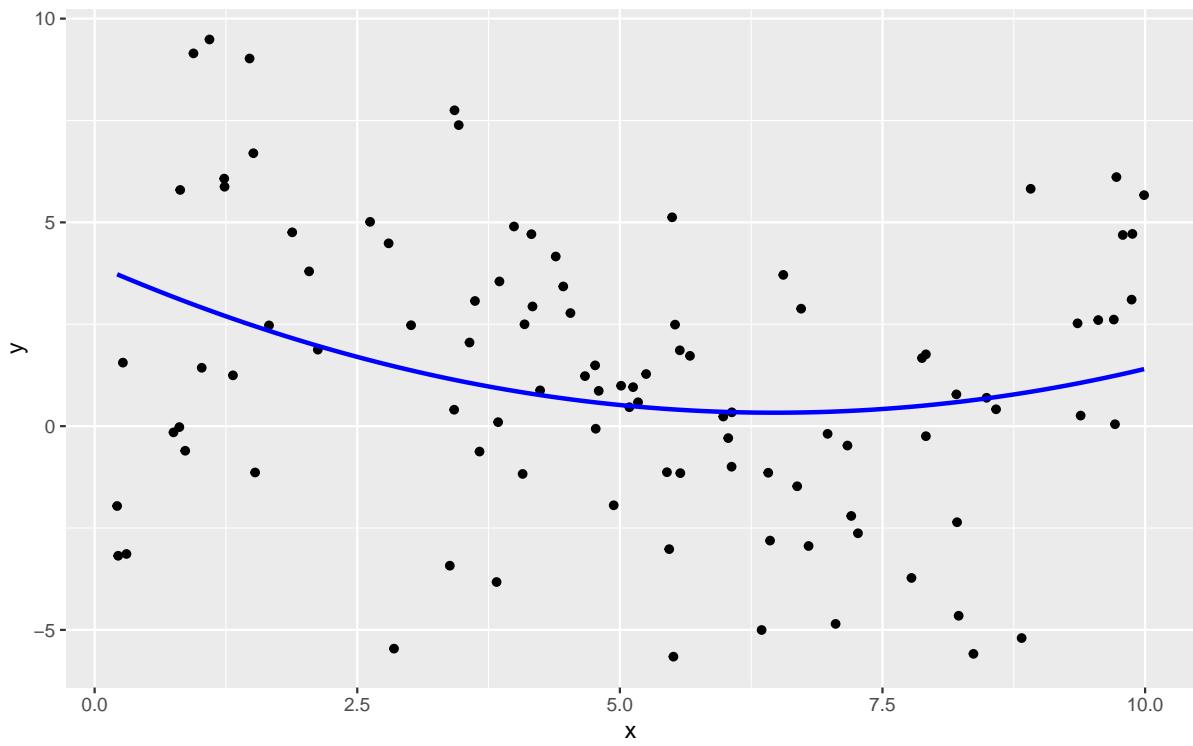
Constant Model to Sample Data



Linear Model to Sample Data

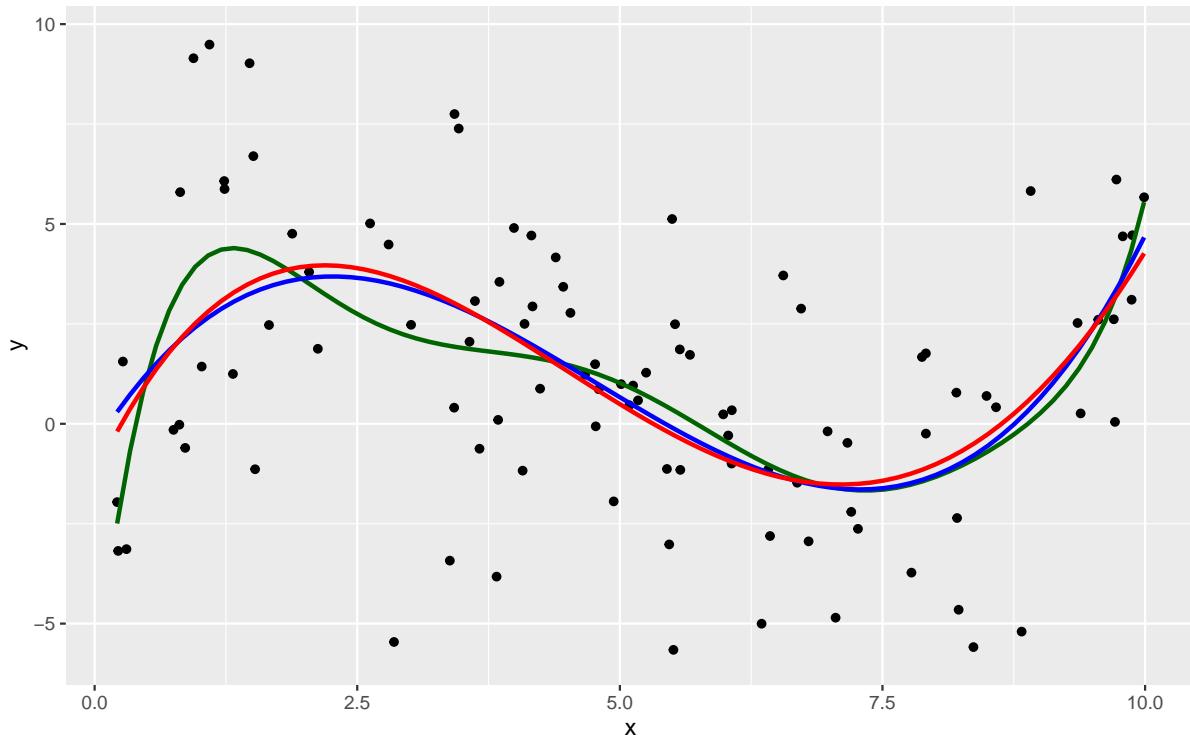


Quadratic Model



Degree 3, 4, and 8 Models

We continue exploring higher order polynomial models. The blue curve represents a third degree (cubic) polynomial model, while the red curve represents a fourth degree (quartic) model and the green represents an eighth degree model.



We see that the flexibility of the model increases as we add higher-order terms. 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. This can be both a good and bad thing, as it allows the model to better bend and fit the data, but also makes it susceptible to the influence of outliers.

7.1.3 Predicting New Data

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	?

We fit polynomial models of degree 0 through 8 to the data. Note that although we did not show the 5th through 7th degree models in our illustrations, we'll still fit these to the data.

```

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))

```

We predict the values of the new observations, using each of the 9 models.

```

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)

```

In fact, since these data were simulated, we know the true value of y , so we can compare the predicted values to the true ones.

```
kable(Newdf %>% dplyr::select(-c(samp)) %>% round(2) %>% head(5))
```

	x	y	Deg0Pred	Deg1Pred	Deg2Pred	Deg3Pred	Deg4Pred	Deg5Pred	Deg6Pred	Deg7Pred	Deg8Pred
108	5.53	5.49	1.17	1.05	0.41	-0.14	-0.30	0.10	0.40	0.25	0.31
4371	2.05	3.92	1.17	1.89	2.02	3.66	3.95	4.26	3.82	3.37	3.49
4839	3.16	1.46	1.17	1.63	1.29	3.24	3.35	2.78	2.24	2.15	2.07
6907	2.06	6.79	1.17	1.89	2.02	3.66	3.95	4.25	3.80	3.36	3.47
7334	2.92	-	1.17	1.68	1.42	3.43	3.60	3.14	2.51	2.31	2.25
			1.03								

7.1.4 Evaluating Predictions - RMSPE

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.

We calculate RMSPE for each of the 9 models.

```
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))
```

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

The third degree model did the best at predicting the new data.

Notice that making the model more complex beyond third degree not only didn't help, but actually hurt prediction accuracy.

7.1.5 Training Data Error

Now, let's examine the behavior if we had fit the models to the data, instead of the test data.

```

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))

```

```

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

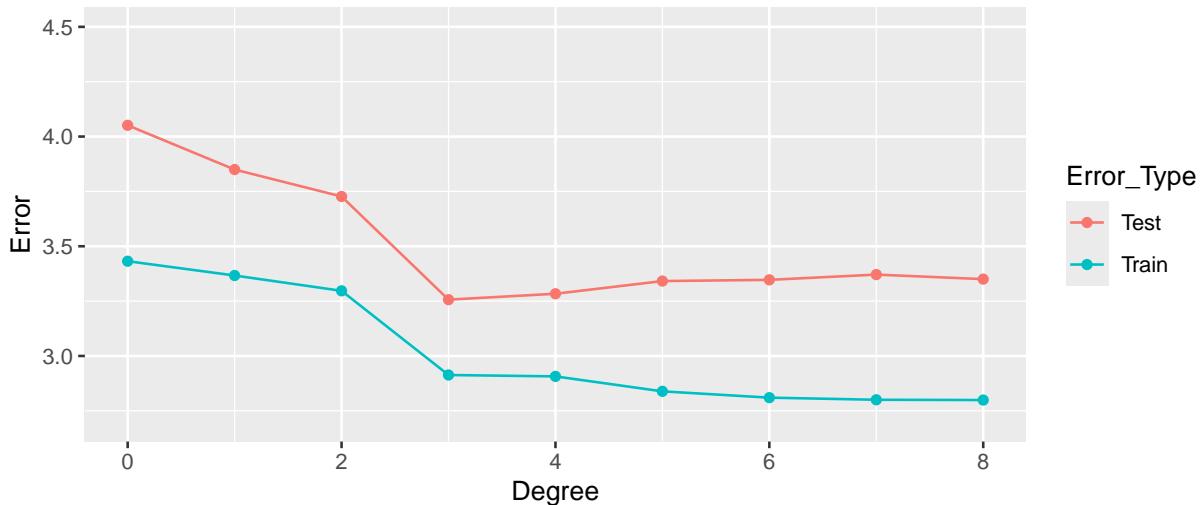
```

	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

Notice that the most complex model achieves the best performance on the training data, but not on the test data.

As the model complexity grows, the model will always fit the training data better, but that does not mean it will perform better on new data. It is possible to start modeling noise, rather than true signal in the training data, which hurts the accuracy of the model when applied to new data.

7.1.6 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

7.1.7 Best Model

Of the models we looked at, the third degree model does the best. The estimates of its coefficients are shown below.

```
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 ***
```

```

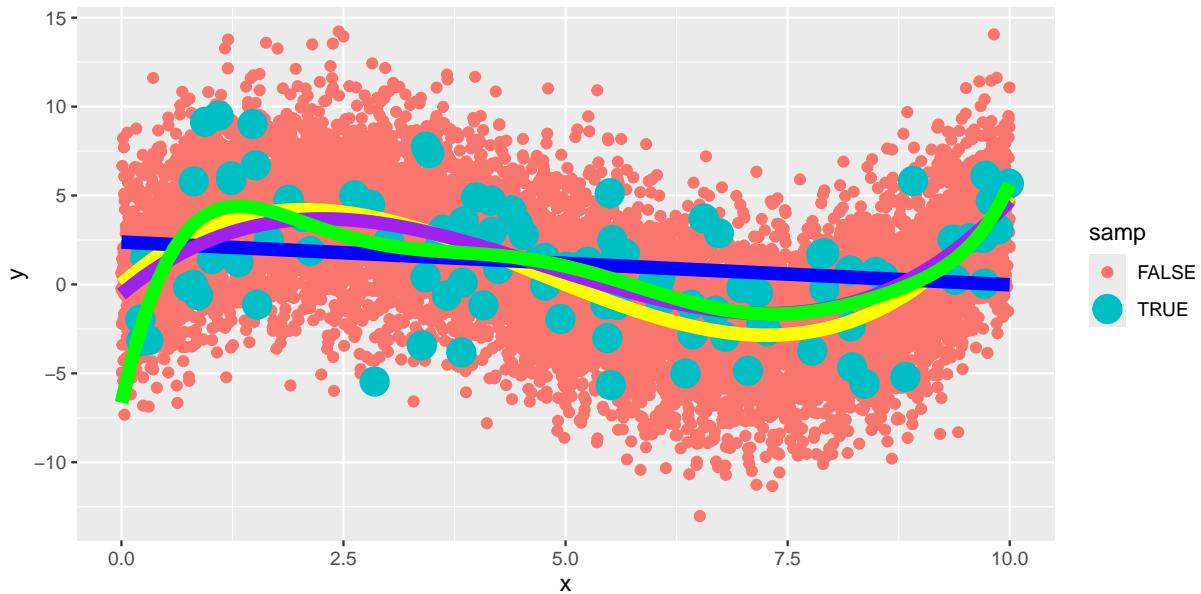
I(x^3)      0.08419    0.01622   5.191 0.00000117 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

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

```

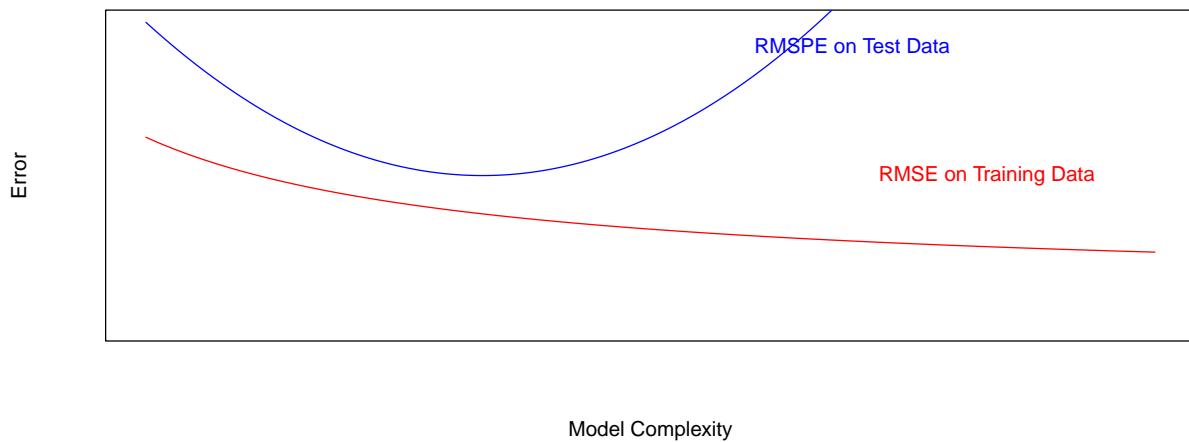
In fact, the data were generated from the model $y_i = 4.5x - 1.4x^2 + 0.1x^3 + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, 3)$

We compare the true expected response curve (in yellow) to the estimates from the various polynomial models.



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**.

7.1.8 Model Complexity, Training Error, and Test Error



7.2 Variance-Bias Tradeoff

7.2.1 What Contributes 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 $(y^* - \hat{y})^2 = (y^* - \hat{f}(x^*))^2$.

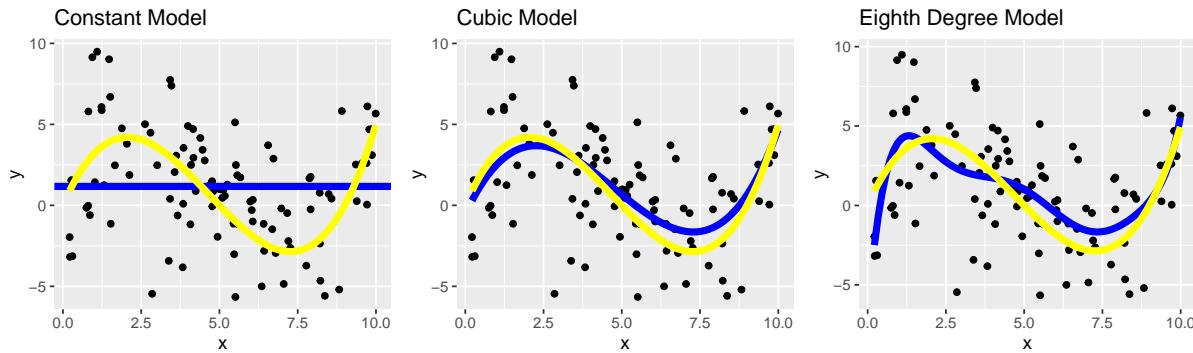
1. **Bias associated with fitting model:** Model bias pertains to the difference between the true response function value $f(x^*)$, and the average value of $\hat{f}(x^*)$ that would be obtained in the long run over many samples.
 - for example, if the true response function f is cubic, then using a 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...

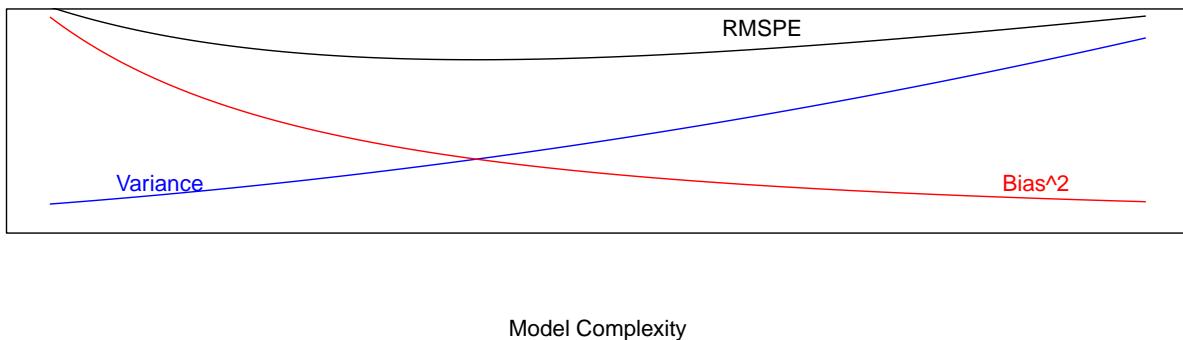
The constant model suffers from high bias. Since it does not include a linear, quadratic, or cubic term, it cannot accurately approximate the true regression function.

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.

7.2.4 Modeling for 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 in the training (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)

7.2.5 Cross-Validation

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.

7.2.6 Cross-Validation Illustration

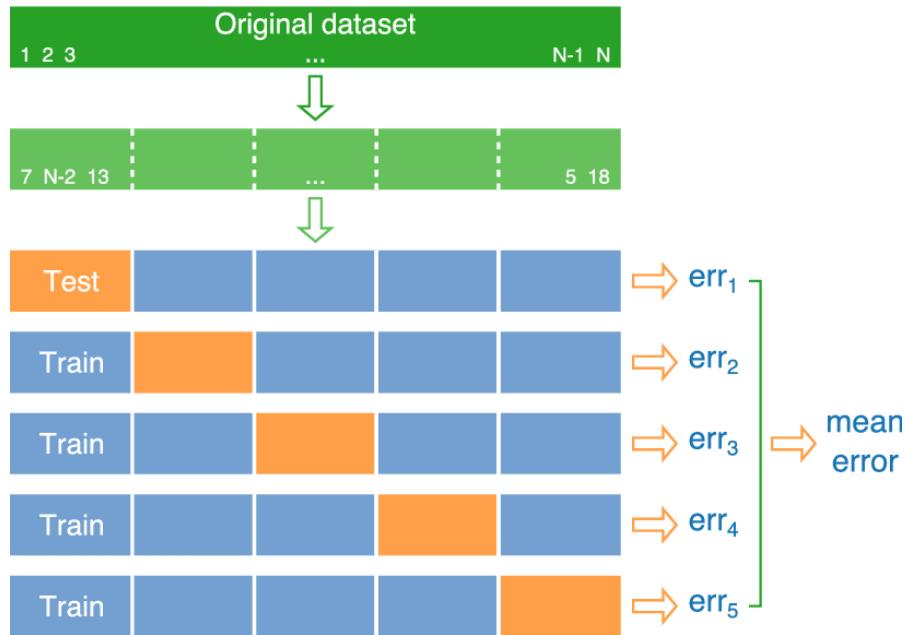


Figure 7.1: https://www.researchgate.net/figure/A-schematic-illustration-of-K-fold-cross-validation-for-K-5-Original-dataset-shown_fig5_311668395

7.2.7 CV in R

The `train` function in the `caret` R package performs cross validation automatically. We'll use it to compare five different models for house prices among a dataset of 1,000 houses sold in Ames, IA between 2006 and 2010.

We'll consider six different models of (mostly) increasing complexity.

```
library(tidyverse)
Train_Data <- read_csv("Ames_Train_Data.csv") # Load data
library(caret) # load caret package
```

```
# set cross-validation settings - use 10 repeats of 10-fold CV
control <- trainControl(method="repeatedcv", number=10, repeats=10, savePredictions = "all")

# define models
# set same random seed before each model to ensure same partitions are used in CV, making the results comparable
set.seed(10302023)
```

```

model1 <- train(data=Train_Data,
                  SalePrice ~ `Overall Qual` ,
                  method="lm", trControl=control)

set.seed(10302023)
model2 <- train(data=Train_Data,
                  SalePrice ~ `Overall Qual` + `Gr Liv Area` + `Garage Area` ,
                  method="lm", trControl=control)

set.seed(10302023)
model3 <- train(data=Train_Data, SalePrice ~ `Overall Qual` +
                  `Gr Liv Area` + `Garage Area` +
                  `Neighborhood` + `Bldg Type` ,
                  method="lm", trControl=control)

set.seed(10302023)
model4 <- train(data=Train_Data, SalePrice ~ `Overall Qual` +
                  + `Gr Liv Area` + `Garage Area` +
                  + `Neighborhood` + `Bldg Type` + `Year Built` ,
                  method="lm", trControl=control)

set.seed(10302023)
model5 <- train(data=Train_Data, SalePrice ~ `Overall Qual` +
                  `Gr Liv Area` + `Garage Area` + `Neighborhood` +
                  `Bldg Type` + `Year Built` + I(`Overall Qual`^2) +
                  I(`Gr Liv Area`^2) + I(`Garage Area`^2) +
                  I(`Year Built`^2), method="lm", trControl=control)

set.seed(10302023)
model6 <- train(data=Train_Data, SalePrice ~ ., method="lm", trControl=control) # include e

# Calculate RMSPE for each model
RMSPE1 <- sqrt(mean((model1$pred$obs-model1$pred$pred)^2))
RMSPE2 <- sqrt(mean((model2$pred$obs-model2$pred$pred)^2))
RMSPE3 <- sqrt(mean((model3$pred$obs-model3$pred$pred)^2))
RMSPE4 <- sqrt(mean((model4$pred$obs-model4$pred$pred)^2))
RMSPE5 <- sqrt(mean((model5$pred$obs-model5$pred$pred)^2))
RMSPE6 <- sqrt(mean((model6$pred$obs-model6$pred$pred)^2))

```

```
RMSPE1
```

```
[1] 51710.58
```

```
RMSPE2
```

```
[1] 44192.34
```

```
RMSPE3
```

```
[1] 38212.96
```

```
RMSPE4
```

```
[1] 38016.67
```

```
RMSPE5
```

```
[1] 38245.46
```

```
RMSPE6
```

```
[1] 40586.14
```

We see that in this case, model M4 performed the best on the hold-out data. We should use Model 4 to make predictions on new data over the other models seen here. It is likely that there are better models out there than model 4, likely with complexity somewhere between that of model 4 and models 5 and 6. Perhaps you can find one.

Once we have our preferred model, we can read in our test data and make predictions, and display the first 10 predicted values.

```
TestData <- read_csv("Ames_Test_Data.csv")
predictions <- predict(model4, newdata=TestData) # substitute your best model
head(data.frame(predictions), 10)
```

```
predictions
1      156767.46
2      252017.22
3      222084.65
4      247418.75
5      116156.38
6      161242.97
7      114106.22
8      46470.89
9      344009.51
10     190402.70
```

We create a csv file containing the predictions, using the code below.

```
write.csv(predictions, file = "predictions.csv")
```

7.3 Ridge Regression

7.3.1 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 higher order 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_1, \dots, b_p are associated with less complex models. When $b_j = 0$, this means variable j is not used in the model.

To illustrate, we fit a regression model to the Ames housing dataset, which includes 71 possible explanatory variables, in addition to price.

```
set.seed(10302021)
samp <- sample(1:nrow(ames_raw), 1000)
Train_Data <- ames_raw[samp,]
```

The full list of coefficient estimates is shown below.

```
M_OLS <- lm(data=Train_Data, SalePrice ~ .)
M_OLS$coefficients
```

(Intercept)	`Overall Qual`	`Year Built`
-14136063.8944780	6975.9249580	494.5819845
`Mas Vnr Area`	`Central Air`Y	`Gr Liv Area`
33.3816934	-3745.3629147	37.2290416
`Lot Frontage`	`1st Flr SF`	`Bedroom AbvGr`
-14.9594204	13.3690338	-2353.0379012
`TotRms AbvGrd`	Order	PID
914.5594852	9.0730317	0.7938731
`MS SubClass`030	`MS SubClass`040	`MS SubClass`045
1747.3877587	5867.2885074	7182.7066393
`MS SubClass`050	`MS SubClass`060	`MS SubClass`070
-1330.0097462	-7400.8589386	-953.6999925
`MS SubClass`075	`MS SubClass`080	`MS SubClass`085
-10585.7402995	-6516.1314579	-6527.2337681
`MS SubClass`090	`MS SubClass`120	`MS SubClass`160
-21891.2260114	-20296.4699853	-34837.6663542
`MS SubClass`180	`MS SubClass`190	`MS Zoning`C (all)
-18822.1874094	-12421.4283083	-38785.6865317
`MS Zoning`FV	`MS Zoning`I (all)	`MS Zoning`RH
-23114.7182938	-17213.3331514	-13306.2219908

`MS Zoning`RL	`MS Zoning`RM	`Lot Area`
-17643.5478889	-24783.4465053	0.7407590
StreetPave	AlleyPave	AlleyNA
37262.6393526	2527.1840221	29.0268338
`Lot Shape`IR2	`Lot Shape`IR3	`Lot Shape`Reg
8575.5874307	10397.8001841	2676.2691253
`Land Contour`HLS	`Land Contour`Low	`Land Contour`Lvl
11318.1617440	-22452.1871122	12230.7109977
`Lot Config`CulDSac	`Lot Config`FR2	`Lot Config`FR3
10427.6521788	-12317.5423512	-14557.2814204
`Lot Config`Inside	`Land Slope`Mod	`Land Slope`Sev
-1168.2308323	10550.3766984	-24213.7593573
NeighborhoodBlueste	NeighborhoodBrDale	NeighborhoodBrkSide
19001.1401867	20558.0541049	14314.5790198
NeighborhoodClearCr	NeighborhoodCollgCr	NeighborhoodCrawfor
7379.3465594	-46.6307558	28775.4384686
NeighborhoodEdwards	NeighborhoodGilbert	NeighborhoodGreens
-7491.2919144	1951.6205024	4667.7592928
NeighborhoodIDOTRR	NeighborhoodMeadowV	NeighborhoodMitchel
12473.1492830	18891.1264802	-9274.3206260
NeighborhoodNAmes	NeighborhoodNoRidge	NeighborhoodNPkVill
1781.9098984	32109.4882191	19601.4137971
NeighborhoodNridgHt	NeighborhoodNWAmes	NeighborhoodOldTown
34407.2004203	-3367.8475918	9857.0794600
NeighborhoodSawyer	NeighborhoodSawyerW	NeighborhoodSomerst
6761.6510470	3945.8508858	20453.1722509
NeighborhoodStoneBr	NeighborhoodSWISU	NeighborhoodTimber
44556.0799048	8518.9310629	1455.9163924
NeighborhoodVeenker	`Condition 1`Feedr	`Condition 1`Norm
5759.5517197	-315.5372848	10041.9749570
`Condition 1`PosA	`Condition 1`PosN	`Condition 1`RR Ae
49498.8746234	15873.9313786	-11599.3284625
`Condition 1`RRAn	`Condition 1`RRNn	`Condition 2`Feedr
7894.0460905	6045.1253614	7639.7933280
`Condition 2`Norm	`Condition 2`PosA	`Condition 2`PosN
6804.6578647	1496.1296933	-224695.9129765
`Condition 2`RRNn	`Overall Cond`	`Year Remod/Add`
20826.2553462	5652.4500122	115.1029835
`Roof Style`Gable	`Roof Style`Gambrel	`Roof Style`Hip
-983.4965913	-3775.4699187	-1020.9768277
`Roof Style`Mansard	`Roof Style`Shed	`Roof Matl`CompShg
18711.8071355	-9552.0308918	671041.0670584
`Roof Matl`Membran	`Roof Matl`Tar&Grv	`Roof Matl`WdShngl

738250.2085208	653237.7170668	757954.9462501
`Mas Vnr Type`BrkFace	`Mas Vnr Type`None	`Mas Vnr Type`Stone
-14297.1433973	-5178.2753573	-11764.3736400
`Exter Qual`Fa	`Exter Qual`Gd	`Exter Qual`TA
-21903.6214492	-37435.0319282	-40396.2768748
`Exter Cond`Fa	`Exter Cond`Gd	`Exter Cond`Po
-2499.1726078	9625.5181637	-9090.3355705
`Exter Cond`TA	FoundationCBlock	FoundationPConc
10976.5039644	2136.6596942	4341.3771422
FoundationSlab	FoundationStone	FoundationWood
-8011.2100540	5209.4079083	-21499.8677282
`Bsmt Qual`Fa	`Bsmt Qual`Gd	`Bsmt Qual`Po
-19118.1141774	-13338.1127108	58476.8071349
`Bsmt Qual`TA	`Bsmt Qual`NA	`Bsmt Exposure`Gd
-14904.6825425	29237.6694620	8445.8558434
`Bsmt Exposure`Mn	`Bsmt Exposure`No	`Bsmt Exposure`NA
-5636.1528491	-5726.8655933	-28286.6797423
`BsmtFin SF 1`	`BsmtFin SF 2`	`Bsmt Unf SF`
33.8127701	24.9524651	13.2017599
HeatingGasW	HeatingWall	`Heating QC`Fa
3554.5996405	16477.8270033	-7277.0668485
`Heating QC`Gd	`Heating QC`Po	`Heating QC`TA
-760.5324990	-17982.8274261	-5710.2836368
ElectricalFuseF	ElectricalFuseP	ElectricalMix
728.4351710	25345.9011297	51715.0815893
ElectricalSBrkr	`2nd Flr SF`	`Bsmt Full Bath`
-2872.3538220	12.4886691	-498.4665953
`Bsmt Half Bath`	`Full Bath`	`Half Bath`
3026.8853409	3781.8172133	3333.8342115
`Kitchen AbvGr`	`Kitchen Qual`Fa	`Kitchen Qual`Gd
-12956.8475513	-7789.8981158	-13554.1557391
`Kitchen Qual`TA	FunctionalMaj2	FunctionalMin1
-13298.5375890	-24570.4189413	-8664.3164291
FunctionalMin2	FunctionalMod	FunctionalTyp
-12709.0598893	-14394.7176699	4206.1036385
Fireplaces	`Fireplace Qu`Fa	`Fireplace Qu`Gd
11467.2013154	-12766.1232883	-14204.2309228
`Fireplace Qu`Po	`Fireplace Qu`TA	`Fireplace Qu`NA
-23140.9430388	-17684.2617792	-5555.0116338
`Garage Type`Attchd	`Garage Type`Basment	`Garage Type`BuiltIn
-1098.5463523	-698.4291295	-4596.8780051
`Garage Type`CarPort	`Garage Type`Detchd	`Garage Yr Blt`
-10478.5627774	-138.9775476	-61.6633788

`Garage Finish`RFn	-4343.2684170	`Garage Finish`Unf	-1482.5096317	`Garage Cars`	2990.4177680
`Garage Area`	24.3013657	`Garage Qual`Fa	-70420.9028312	`Garage Qual`Gd	-51114.4969380
`Garage Qual`Po	-134052.9764560	`Garage Qual`TA	-67768.2607876	`Garage Cond`Fa	73986.1047213
`Garage Cond`Gd	64104.4268776	`Garage Cond`Po	112712.4021077	`Garage Cond`TA	71259.2488661
`Paved Drive`P	5257.0317668	`Paved Drive`Y	2077.1863821	`Wood Deck SF`	7.6414867
`Open Porch SF`	-16.2882422	`Enclosed Porch`	-11.3738346	`3Ssn Porch`	16.1217441
`Screen Porch`	38.2573038	`Pool Area`	-106.8682816	`Pool QC`TA	24551.0371793
`Pool QC`NA	-100661.2221692	FenceGdWo	3.7225602	FenceMnPrv	1567.2382267
FenceMnWw	-880.4035636	FenceNA	1297.2148680	`Misc Feature`Gar2	545161.5427966
`Misc Feature`Othr	569599.2024080	`Misc Feature`Shed	551469.2569906	`Misc Feature`NA	547892.2454606
`Misc Val`	1.6361319	`Mo Sold`	-262.1972861	`Yr Sold`	5937.2902795
`Sale Type`Con	7985.9754307	`Sale Type`ConLD	23972.1096774	`Sale Type`ConLI	9525.1931028
`Sale Type`ConLw	11056.3865076	`Sale Type`CWD	-9551.0936247	`Sale Type`New	23118.1311002
`Sale Type`Oth	39733.0338449	`Sale Type`VWD	-11729.6976216	`Sale Type`WD	4819.9397983
`Sale Condition`AdjLand	41276.9513339	`Sale Condition`Alloca	24461.6278440	`Sale Condition`Family	404.8499384
`Sale Condition`Normal	1592.2625631	`Sale Condition`Partial	-4641.7581345		

Let's focus on the first 10 rows.

```
head(coef(M_OLS), 10) %>% round(3)
```

(Intercept)	`Overall Qual`	`Year Built`	`Mas Vnr Area`	`Central Air`Y
-14136063.894	6975.925	494.582	33.382	-3745.363
`Gr Liv Area`	`Lot Frontage`	`1st Flr SF`	`Bedroom AbvGr`	`TotRms AbvGrd`
37.229	-14.959	13.369	-2353.038	914.559

If all coefficients in the model were 0, then we would be using the most simple constant model, and the prediction for the price of each house would be exactly the same as the overall mean. As b_j 's get farther from 0, predictions begin move away from the overall mean and depend more and more on the values or categories of the explanatory variable(s) associated with individual houses. This creates a risk, however, of overfitting.

A way to combat this, other than dropping variables from the model, is to shrink some or all of the regression coefficients closer to 0, pushing predictions closer to the overall mean.

A statistical technique for doing this is called **ridge regression**.

7.3.2 Ridge Regression Penalty

We've seen that in ordinary least-squares regression, b_0, b_1, \dots, b_p are chosen in a way that to 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 incentive choosing values of b_1, \dots, b_p that are closer to 0, thereby "shrinking" the predictions toward the overall mean house price.

Specifically, we minimize:

$$\begin{aligned} & \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^p b_j^2 \\ &= \sum_{i=1}^n (y_i - (b_0 + b_1 x_{i1} + b_2 x_{i2} + \dots + b_p x_{ip}))^2 + \lambda \sum_{j=1}^p b_j^2 \end{aligned}$$

where λ is a pre-determined positive constant.

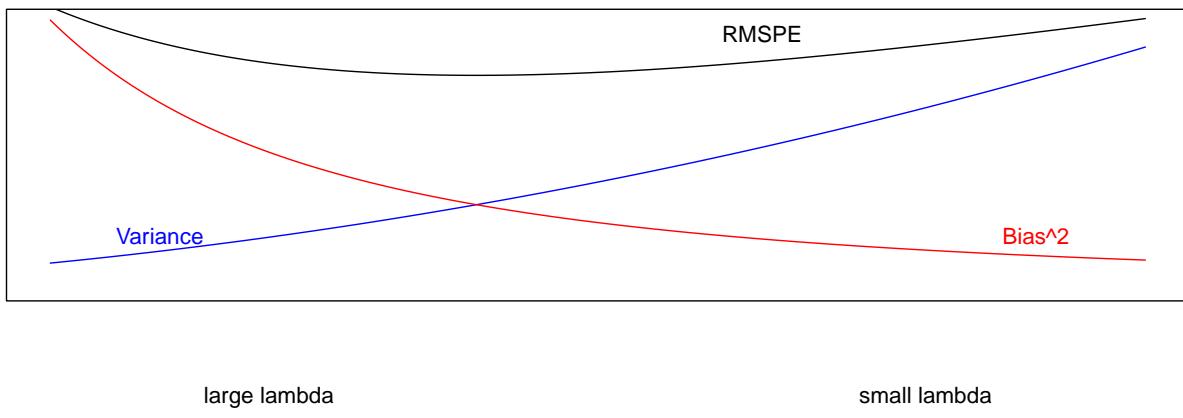
Larger values of b_j typically help the model better fit the training data, thereby making the first term smaller, but also make the second term larger. The idea is the find optimal values of b_0, b_1, \dots, b_p that are large enough to allow the model to fit the data well, thus keeping the first term (SSR) small, while also keeping the penalty term small as well.

7.3.3 Choosing 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.

$$\begin{aligned} Q &= \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^p b_j^2 \\ &= \sum_{i=1}^n (y_i - (b_0 + b_1 x_{i1} + b_2 x_{i2} + \dots + b_p x_{ip}))^2 + \lambda \sum_{j=1}^p b_j^2 \end{aligned}$$

- 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



When using ridge regression, 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.

```
Train_sc <- Train_Data %>% mutate_if(is.numeric, scale)
```

7.3.4 Ridge Regression on Housing Dataset

We'll use the `caret` package to perform cross validation in order to find the optimal value of λ . To use ridge regression, we specify `method = "glmnet"`, and `tuneGrid=expand.grid(alpha=0, lambda=l_vals)`. Note the `alpha` value can be changed to use other types of penalized regression sometimes used in predictive modeling, such as lasso or elastic net.

```
control = trainControl("repeatedcv", number = 10, repeats=10)
l_vals = 10^seq(-3, 3, length = 100) # test values between 1/1000 and 1000

set.seed(11162020)
Housing_ridge <- train(SalePrice ~ .,
                        data = Train_sc, method = "glmnet", trControl=control ,
                        tuneGrid=expand.grid(alpha=0, lambda=l_vals))
```

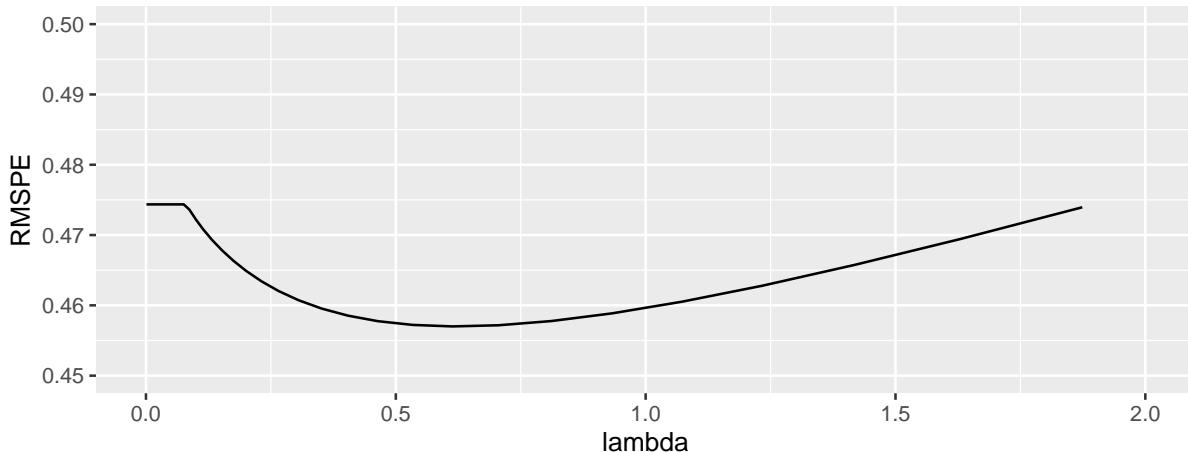
Value of λ minimizing RMSPE:

```
Housing_ridge$bestTune$lambda
```

```
[1] 0.6135907
```

We examine RMSPE on the withheld data as a function of λ .

Ridge Regression Cross Validation Results



Using $\lambda = 0.6135907$, obtain the following set of ridge regression coefficients. Notice how the ridge coefficients are typically closer to 0 than the ordinary least squares coefficients, indicating a less complex model.

```

M_OLS_sc <- lm(data=Train_sc, SalePrice ~ .)
OLS_coef <- M_OLS_sc$coefficients
Ridge_coef <- coef(Housing_ridge$finalModel, Housing_ridge$bestTune$lambda) [,1]
df <- data.frame(OLS_coef[2:10], Ridge_coef[2:10])
names(df) <-c("OLS Coeff", "Ridge Coeff")
df

```

	OLS Coeff	Ridge Coeff
`Overall Qual`	0.121728754	0.10435284
`Year Built`	0.187102422	0.03451303
`Mas Vnr Area`	0.080212607	0.06202880
`Central Air`Y	-0.046191694	0.04289126
`Gr Liv Area`	0.237623291	0.00000000
`Lot Frontage`	-0.004290945	0.07967743
`1st Flr SF`	0.069910650	0.01020597
`Bedroom AbvGr`	-0.022457937	0.07194208
`TotRms AbvGrd`	0.017574153	0.01342224

Predictions and residuals for the first six houses in the training data, using ordinary least squares and ridge regression, are shown below.

```

library(glmnet)
MAT <- model.matrix(SalePrice~., data=Train_sc)
ridge_mod <- glmnet(x=MAT, y=Train_sc$SalePrice, alpha = 0, lambda=Housing_ridge$bestTune$lambda)

```

```

y <- Train_sc$SalePrice
Pred_OLS <- predict(M_OLS_sc)
Pred_Ridge <- predict(ridge_mod, newx=MAT)
OLS_Resid <- y - Pred_OLS
Ridge_Resid <- y - Pred_Ridge
Resdf <- data.frame(y, Pred_OLS, Pred_Ridge, OLS_Resid, Ridge_Resid)
names(Resdf) <- c("y", "OLS Pred", "Ridge Pred", "OLS Resid", "Ridge Resid")
kable(head(Resdf))

```

	y	OLS Pred	Ridge Pred	OLS Resid	Ridge Resid
859	-0.6210832	-0.4637429	-0.4651589	-0.1573403	-0.1559243
1850	0.6800520	1.1897467	1.0528536	-0.5096947	-0.3728016
1301	-0.4545873	-0.4527781	-0.4958630	-0.0018092	0.0412758
981	-0.6408161	-0.6626212	-0.7711186	0.0218051	0.1303025
2694	-0.7937457	-0.8679455	-0.7543093	0.0741997	-0.0394365

	y	OLS Pred	Ridge Pred	OLS Resid	Ridge Resid
2209	-0.7906625	-0.6955254	-0.6449779	-0.0951370	-0.1456845

7.3.5 Ridge vs OLS

In OLS, we choose b_0, b_1, \dots, b_p are chosen in a way that minimizes

$$\sum *i = \ln(y_i - \hat{y}_i)^2 = \sum i = \ln(y_i - (b_0 + b_1x_{i1} + b_2x_{i2} + \dots + b_px_{ip}))^2$$

OLS: $\sum_{i=1}^n (y_i - \hat{y}_i)^2$

```
sum((y-Pred_OLS)^2)
```

[1] 56.94383

Ridge: $\sum_{i=1}^n (y_i - \hat{y}_i)^2$

```
sum((y-Pred_Ridge)^2)
```

[1] 127.1331

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.

In ridge regression, b_0, b_1, \dots, b_p are chosen in a way that minimizes

$$\begin{aligned} Q &= \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^p b_j^2 \\ &= \sum_{i=1}^n (y_i - (b_0 + b_1x_{i1} + b_2x_{i2} + \dots + b_px_{ip}))^2 + \lambda \sum_{j=1}^p b_j^2 \end{aligned}$$

OLS: $\sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^p b_j^2$

```
sum((y-Pred_OLS)^2) + 0.6136*sum(coef(M_OLS_sc)[-1]^2)
```

[1] 373.1205

$$\text{Ridge: } \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^p b_j^2$$

```
sum((y-Pred_Ridge)^2) + 0.6136*sum((Ridge_coef)[-1]^2)
```

[1] 130.3375

We see that the ridge coefficients achieve a lower value of Q than the OLS ones.

7.3.6 Lasso and Elastic Net

Two other techniques that are similar to ridge regression are lasso and elastic net. Both also aim to avoid overfitting by shrinking regression coefficients toward 0 in a manner similar to ridge regression.

Lasso regression is very similar to ridge regression. Coefficients b_0, b_1, \dots, b_p are chosen in a way that minimizes

$$\begin{aligned} & \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^p |b_j| \\ &= \sum_{i=1}^n (y_i - (b_0 + b_1 x_{i1} + b_2 x_{i2} + \dots + b_p x_{ip}))^2 + \lambda \sum_{j=1}^p |b_j| \end{aligned}$$

Regression with an elastic net uses both ridge and lasso penalty terms and determines the values of b_0, b_1, \dots, b_p by minimizing

$$\begin{aligned} & \sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^p |b_j| \\ &= \sum_{i=1}^n (y_i - (b_0 + b_1 x_{i1} + b_2 x_{i2} + \dots + b_p x_{ip}))^2 + \lambda_1 \sum_{j=1}^p b_j^2 + \lambda_2 \sum_{j=1}^p |b_j| \end{aligned}$$

7.4 Decision Trees

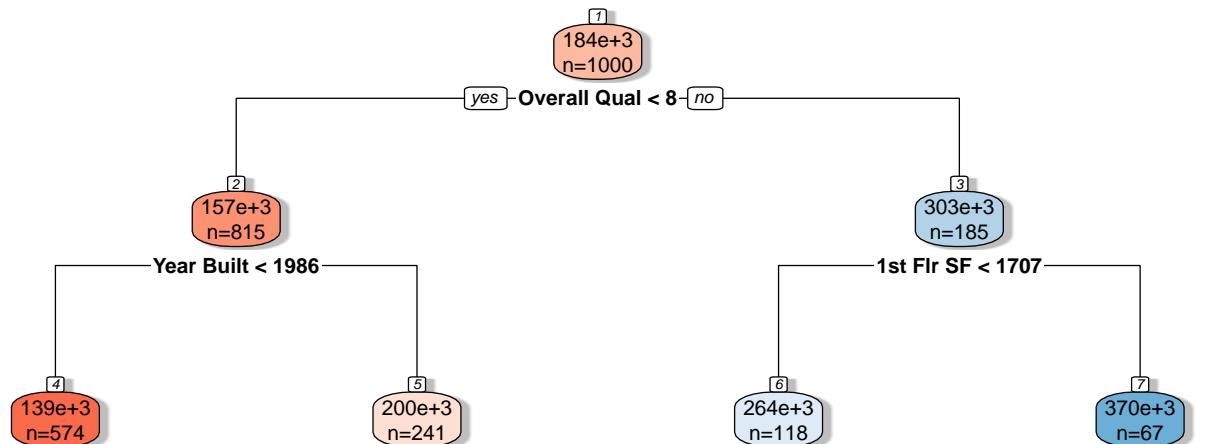
7.4.1 Basics of Decision Trees

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.

Decision Tree Example:

We fit a decision tree to the Ames Housing dataset, using the `rpart` function in a package by the same name.

```
library(rpart)
library(rpart.plot)
tree <- rpart(SalePrice~., data=Train_Data, cp=0.04)
rpart.plot(tree, box.palette="RdBu", shadow.col="gray", nn=TRUE, cex=1, extra=1)
```



We see that the houses are first split based on whether or not their overall quality rating was less than 8. Each of the resulting nodes are then split again, using information from other explanatory variables. Each split partitions the data further, so that houses in the same node can be thought of as being similar to one another.

- The predicted price of a House with overall quality 7, and was built in 1995 is \$200,000.
- The predicted price of a House overall quality 8 and 1,750 sq. ft. on the first floor is \$370,000.

7.4.2 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:

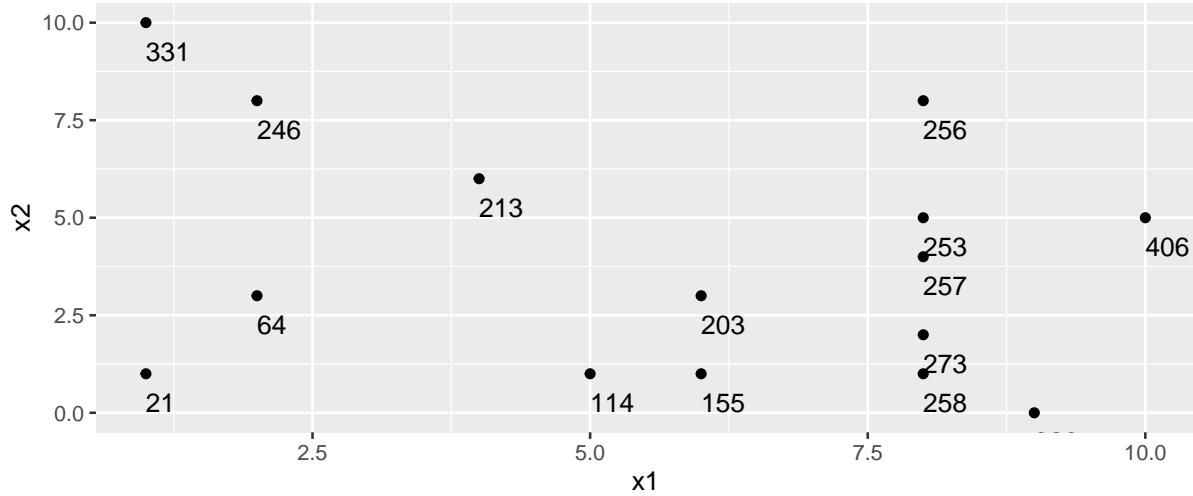
$$\sum_{i=1}^{n_L} (y_i - \bar{y}_L)^2 + \sum_{i=1}^{n_R} (y_i - \bar{y}_R)^2$$

We choose the split that minimizes this quantity.

Partitioning Example

Consider a dataset with two explanatory variables, x_1 and x_2 , and a response variable y , whose values are shown numerically in the graph.

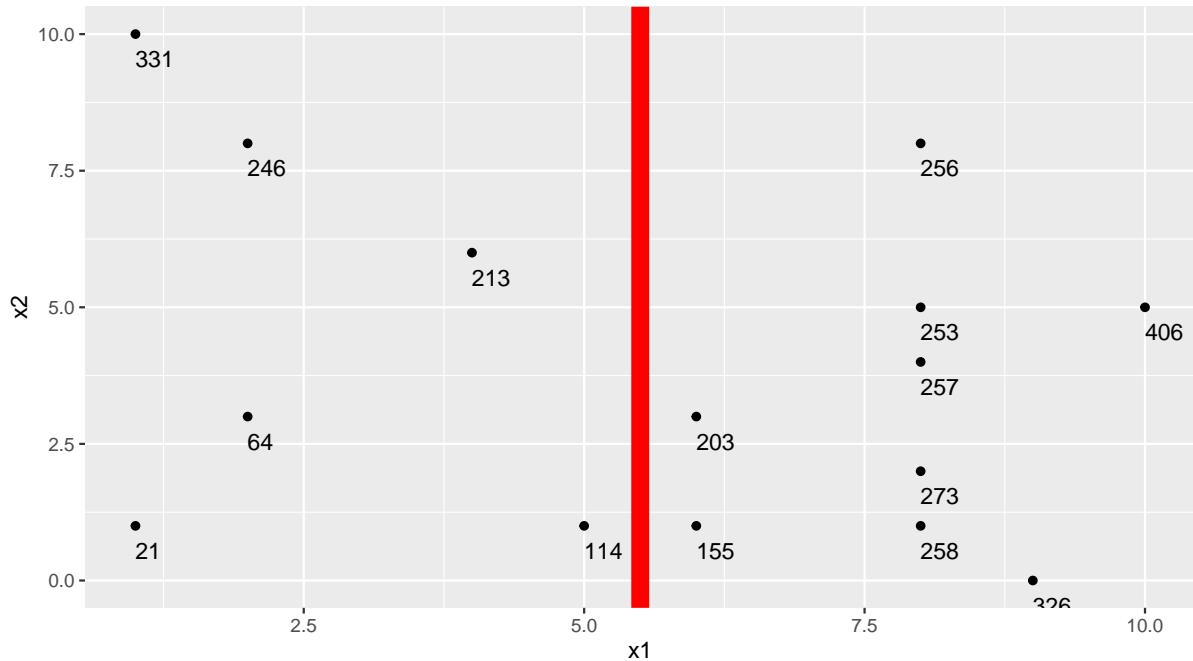
	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	[,11]	[,12]	[,13]	[,14]
x1	8	2	8	1	8	6	2	5	1	8	4	10	9	8
x2	5	3	1	1	4	3	8	1	10	8	6	5	0	2
y	253	64	258	21	257	203	246	114	331	256	213	406	326	273
	[,15]													
x1	6													
x2	1													
y	155													



The goal is to split up the data, using information about x_1 and x_2 in a way that makes the y values grouped together as similar as possible.

1. One Possible Split ($x_1 < 5.5$)

We could split the data into 2 groups depending on whether $x_1 < 5.5$.



We calculate the mean y -value in each resulting node:

- $\bar{y}_L = (331 + 246 + 213 + 21 + 64 + 114)/6 \approx 164.84$

- $\bar{y}_R = (203 + 155 + 256 + 253 + 257 + 273 + 258 + 326 + 406)/9 \approx 265.22$

To measure the amount of deviation in the node, we calculate the sum of the squared difference between each individual value and the overall mean in each node.

$$\begin{aligned} & \sum_{i=1}^{n_L} (y_i - \bar{y}_L)^2 \\ &= (331 - 164.83)^2 + (246 - 164.33)^2 + \dots + (114 - 164.33)^2 \\ &= 69958.83 \end{aligned}$$

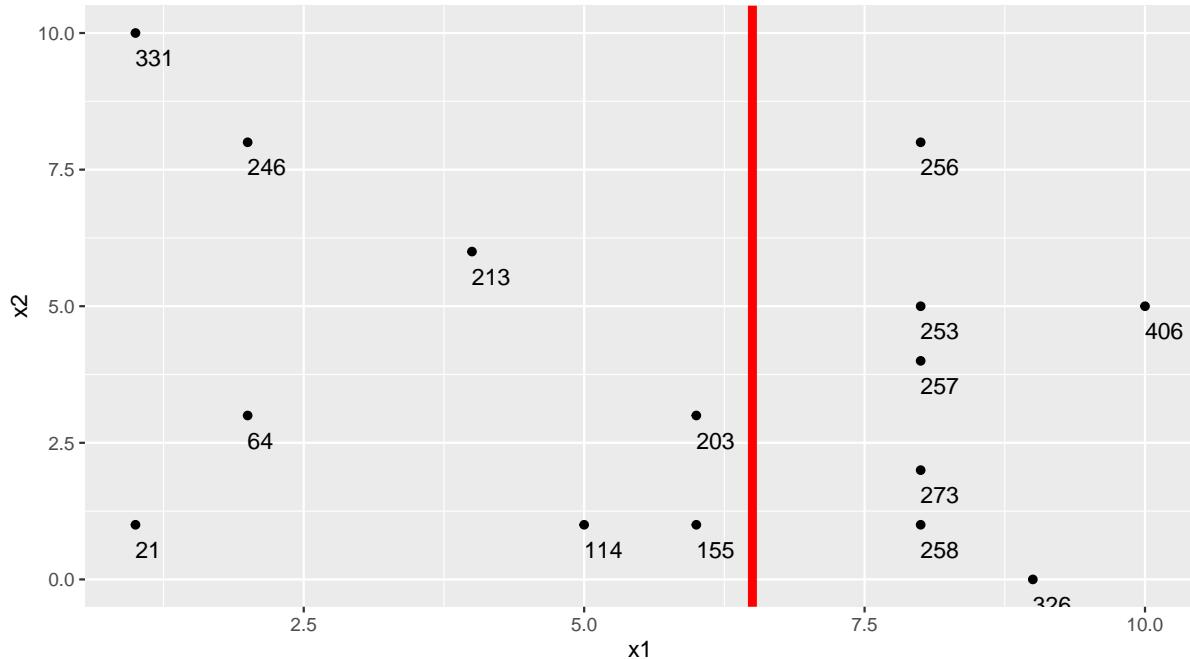
$$\begin{aligned} & \sum_{i=1}^{n_R} (y_i - \bar{y}_R)^2 \\ &= (203 - 265.22)^2 + (155 - 265.22)^2 + \dots + (406 - 265.22)^2 \\ &= 39947.56 \end{aligned}$$

Adding together these two quantities, we obtain an overall measure of the squared deviations between observations in the same node.

- $69958.83 + 39947.56 = 109906.4$

2. Second Possible Split ($x_1 < 6.5$)

We could alternatively split the data into 2 groups depending on whether $x_1 < 6.5$.



Using this split,

- $\bar{y}_L = (331 + 246 + 213 + 21 + 64 + 114 + 203 + 155)/8 \approx 168.375$
- $\bar{y}_R = (256 + 253 + 257 + 273 + 258 + 326 + 406)/7 \approx 289.857$

$$\begin{aligned} & \sum_{i=1}^{n_L} (y_i - \bar{y}_L)^2 \\ &= (331 - 168.375)^2 + (246 - 168.375)^2 + \dots + (203 - 168.375)^2 \\ &= 71411.88 \end{aligned}$$

$$\begin{aligned} & \sum_{i=1}^{n_R} (y_i - \bar{y}_R)^2 \\ &= (203 - 289.857)^2 + (155 - 289.857)^2 + \dots + (406 - 289.857)^2 \\ &= 19678.86 \end{aligned}$$

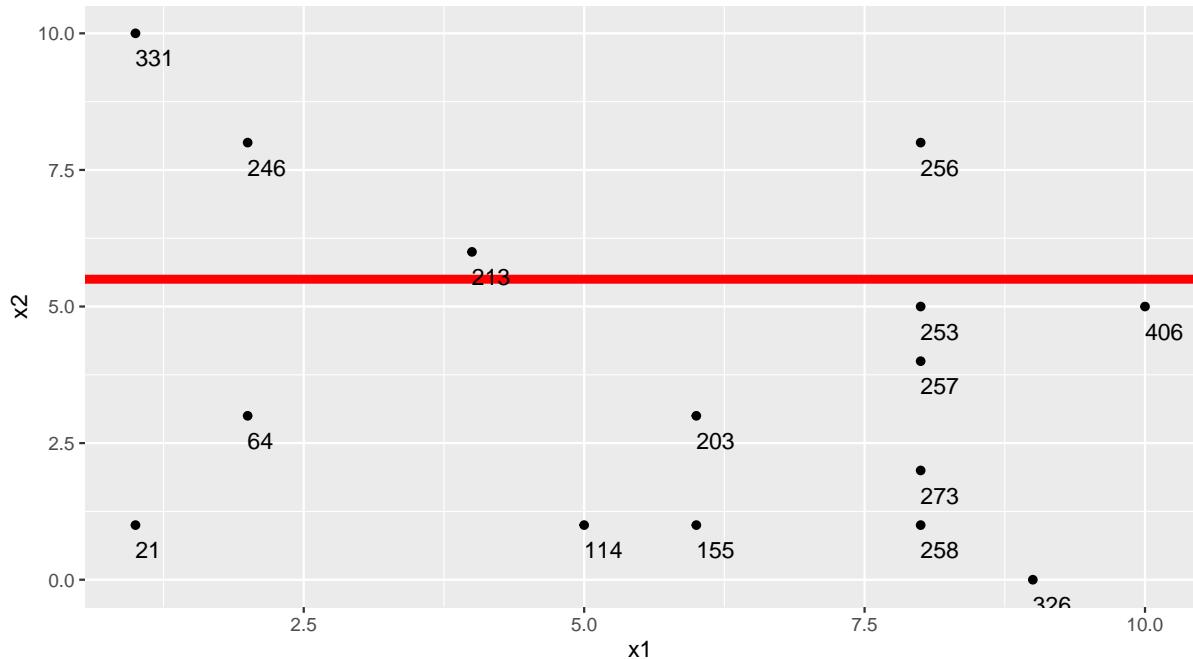
The total squared deviation is:

- $71411.88 + 19678.86 = 91090.74$

The split at $x_1 < 6.5$ is better than $x_1 < 5.5$

3. Third Possible Split ($x_2 < 5.5$)

We could also split the data into 2 groups depending on whether $x_2 < 5.5$.



Using this split,

- $\bar{y}_L = (331 + 246 + 213 + 256)/4 \approx 261.5$
- $\bar{y}_R = (21 + 64 + \dots + 406)/11 \approx 211.82$

$$\begin{aligned} & \sum_{i=1}^{n_L} (y_i - \bar{y}_L)^2 \\ &= (331 - 261.5)^2 + (246 - 261.5)^2 + (213 - 261.5)^2 + (256 - 261.5)^2 \\ &= 7453 \end{aligned}$$

$$\begin{aligned} & \sum_{i=1}^{n_R} (y_i - \bar{y}_R)^2 \\ &= (21 - 211.82)^2 + (64 - 211.82)^2 + \dots + (406 - 211.82)^2 \\ &= 131493.6 \end{aligned}$$

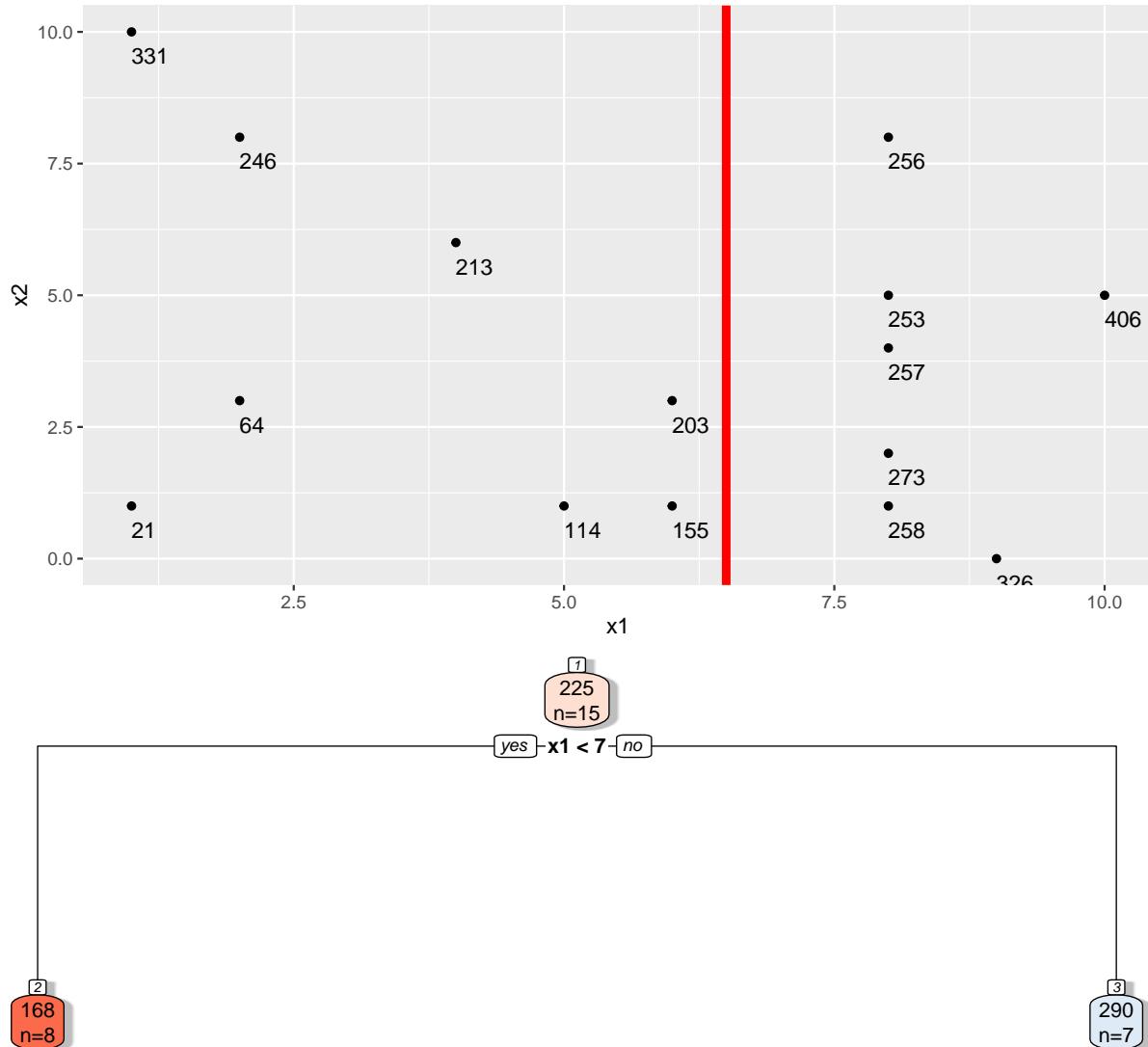
The sum of squared deviations is:

- $7453 + 131493.6 = 138946.6$

Comparison of Splits

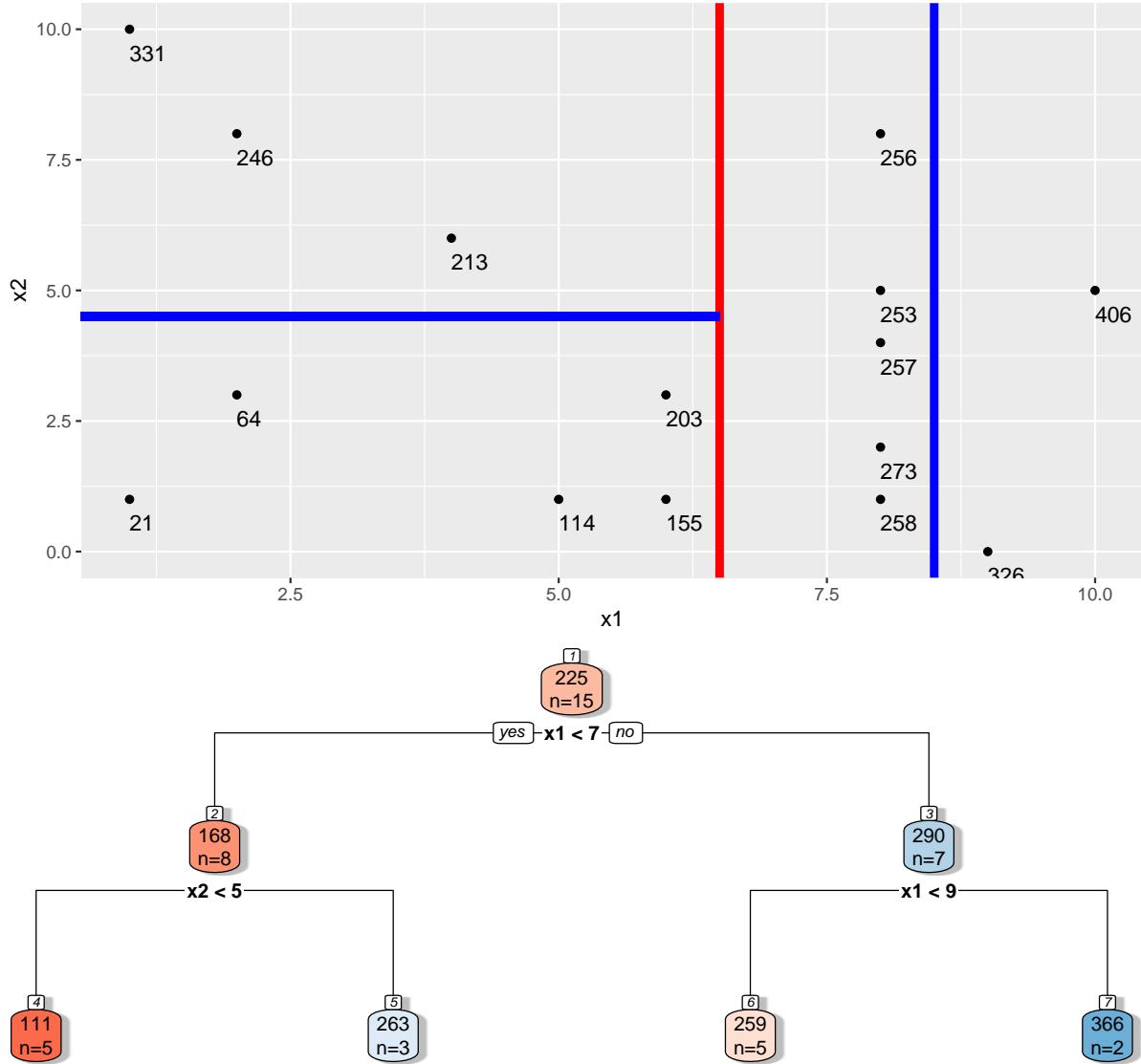
- Of the three split's we've calculated, $\sum_{i=1}^{n_L} (y_i - \bar{y}_L)^2 + \sum_{i=1}^{n_R} (y_i - \bar{y}_R)^2$ is minimized using $x_1 < 6.5$.
- In fact, if we calculate all possible splits over x_1 and x_2 , $\sum_{i=1}^{n_L} (y_i - \bar{y}_L)^2 + \sum_{i=1}^{n_R} (y_i - \bar{y}_R)^2$ is minimized by splitting on $x_1 < 6.5$

Thus, we perform the first split in the tree, using $x_1 < 6.5$.



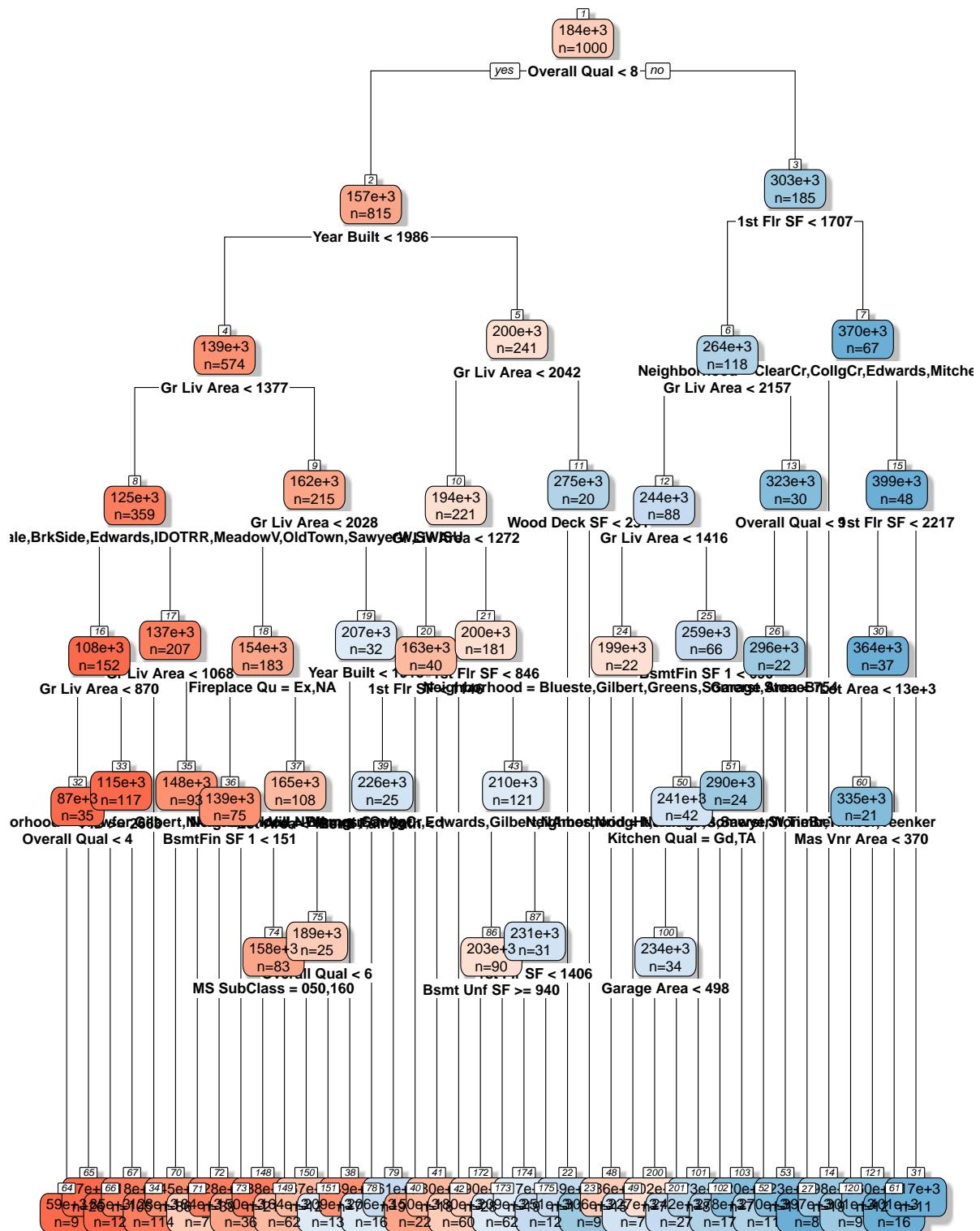
7.4.3 Next Splits

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 $x_1 < 8.5$.



7.4.4 Recursive Partitioning

Splitting continues until nodes reach a certain predetermined minimal size, or until change improvement in model fit drops below a predetermined value



7.4.5 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, since they allow splits even when model fit only improves by a little.

7.4.6 Cross-Validation on Housing Data

We'll use `caret` to determine the optimal value of the `cp` parameter. We use `method="rpart"` to grow decision trees.

```
cp_vals = 10^seq(-8, 1, length = 100) # test values between 1/10^8 and 1
colnames(Train_sc) <- make.names(colnames(Train_sc))

set.seed(11162020)
Housing_Tree <- train(data=Train_sc, SalePrice ~ ., method="rpart", trControl=control,
                        tuneGrid=expand.grid(cp=cp_vals))
```

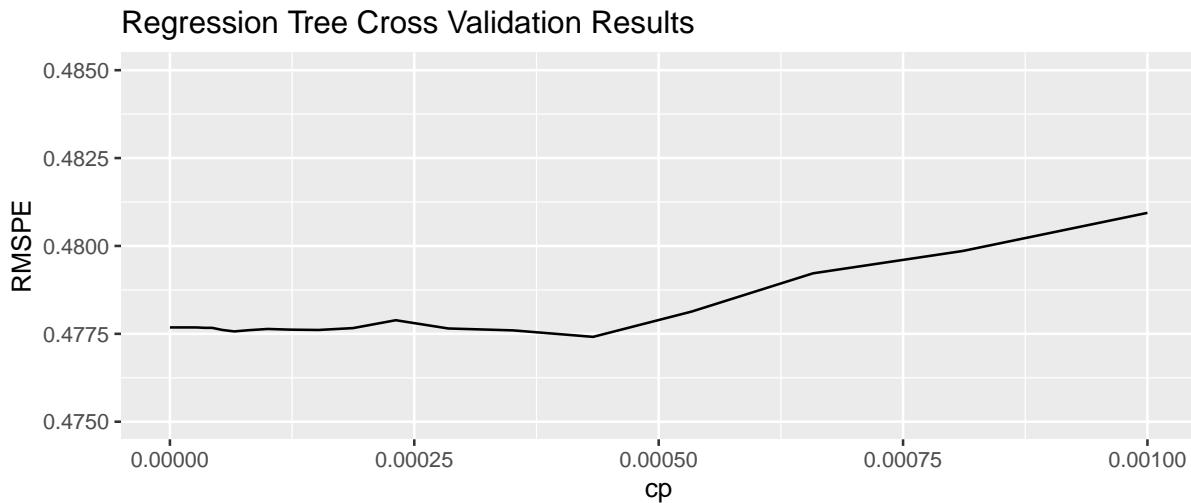
The optimal value of `cp` is:

```
Housing_Tree$bestTune
```

```
      cp
52 0.0004328761
```

We plot RMSPE on the holdout data as a function of `cp`.

```
cp <- Housing_Tree$results$cp
RMSPE <- Housing_Tree$results$RMSE
ggplot(data=data.frame(cp, RMSPE), aes(x=cp, y=RMSPE))+geom_line() + xlim(c(0,0.001)) + ylim
  ggttitle("Regression Tree Cross Validation Results")
```



7.4.7 Comparing OLS, Lasso, Ridge, and Tree

```
set.seed(11162020)
Housing_OLS <- train(data=Train_sc, SalePrice ~ ., method="lm", trControl=control)
set.seed(11162020)
Housing_lasso <- train(SalePrice ~., data = Train_sc, method = "glmnet", trControl=control,
                      tuneGrid=expand.grid(alpha=1, lambda=l_vals))
```

RMSPE on the standardized version of the response variable is displayed below for ordinary least squares, ridge regression, lasso regression, and a decision tree.

```
min(Housing_OLS$results$RMSE)
```

```
[1] 0.5634392
```

```
min(Housing_ridge$results$RMSE)
```

```
[1] 0.4570054
```

```
min(Housing_lasso$results$RMSE)
```

```
[1] 0.4730672
```

```
min(Housing_Tree$results$RMSE)
```

```
[1] 0.477414
```

In this situation, the tree outperforms OLS, but does not do as well as lasso or ridge. 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.

7.4.8 Random Forest

A popular extension of a decision tree is a random forest. A random forest consists of many (often ~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.

7.5 Regression Splines

7.5.1 Regression Splines

We've seen that we can use polynomial regression to capture nonlinear trends in data.

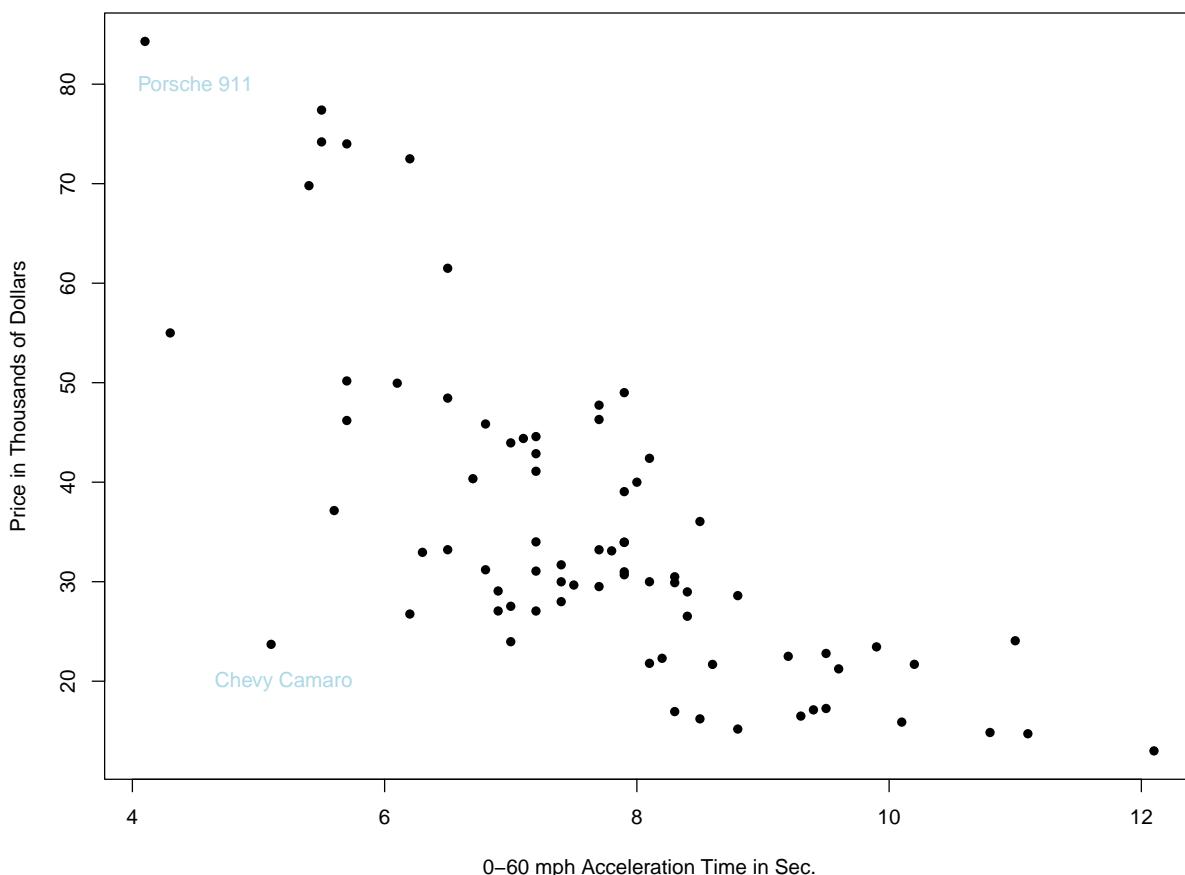
- A **regression spline** is a piecewise function of polynomials.

Here we'll keep things simple by focusing on a spline with a single explanatory variable. Splines can also be used for multivariate data.

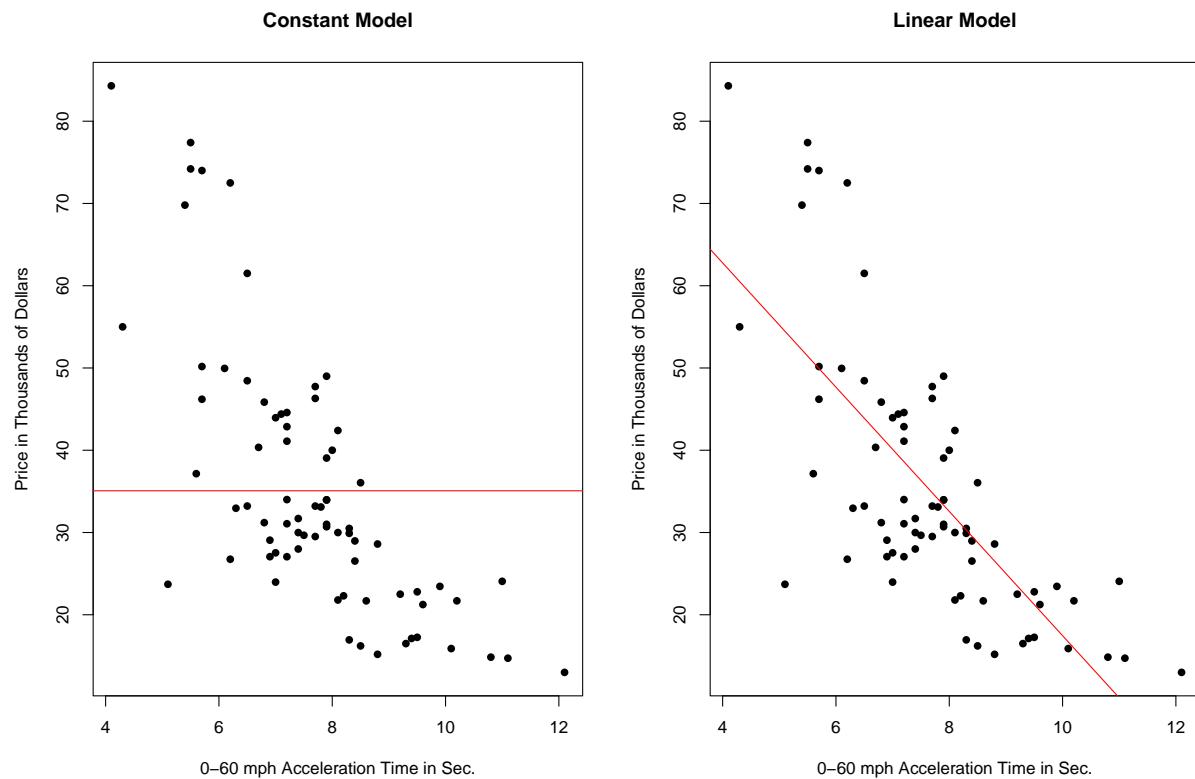
We'll examine the use of splines on the car price prediction dataset.

We divide the data into a set of 75 cars, which we'll use to train the model, and 35 cars, on which we'll make and evaluate predictions.

The 75 cars in the training set are shown below.

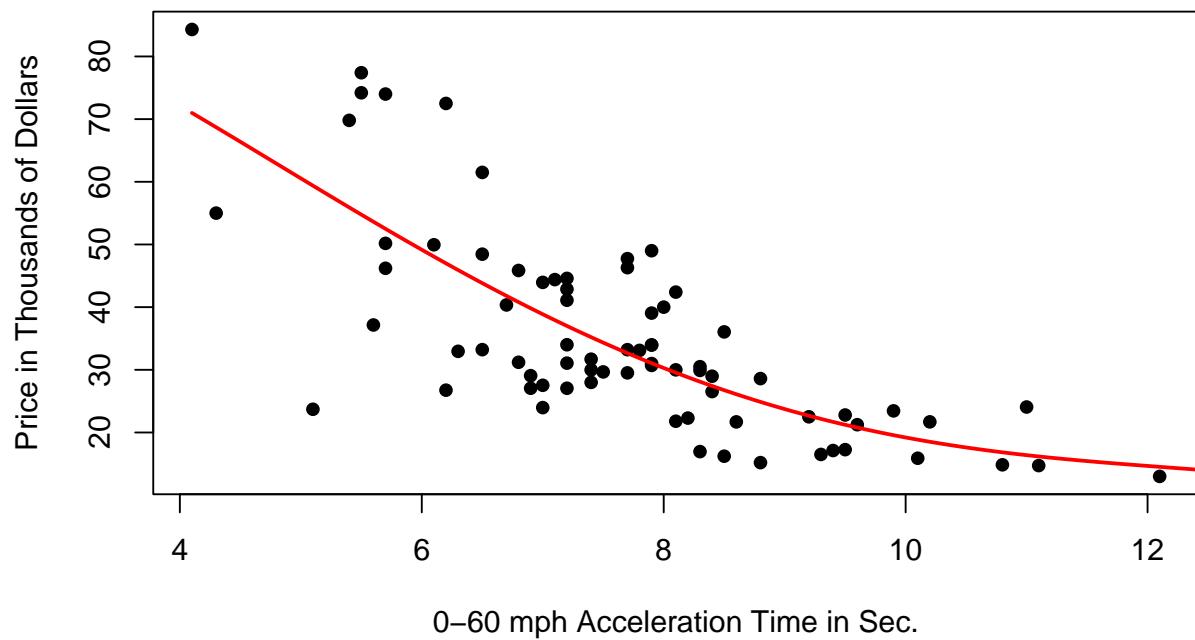


7.5.2 Two Models with High Bias



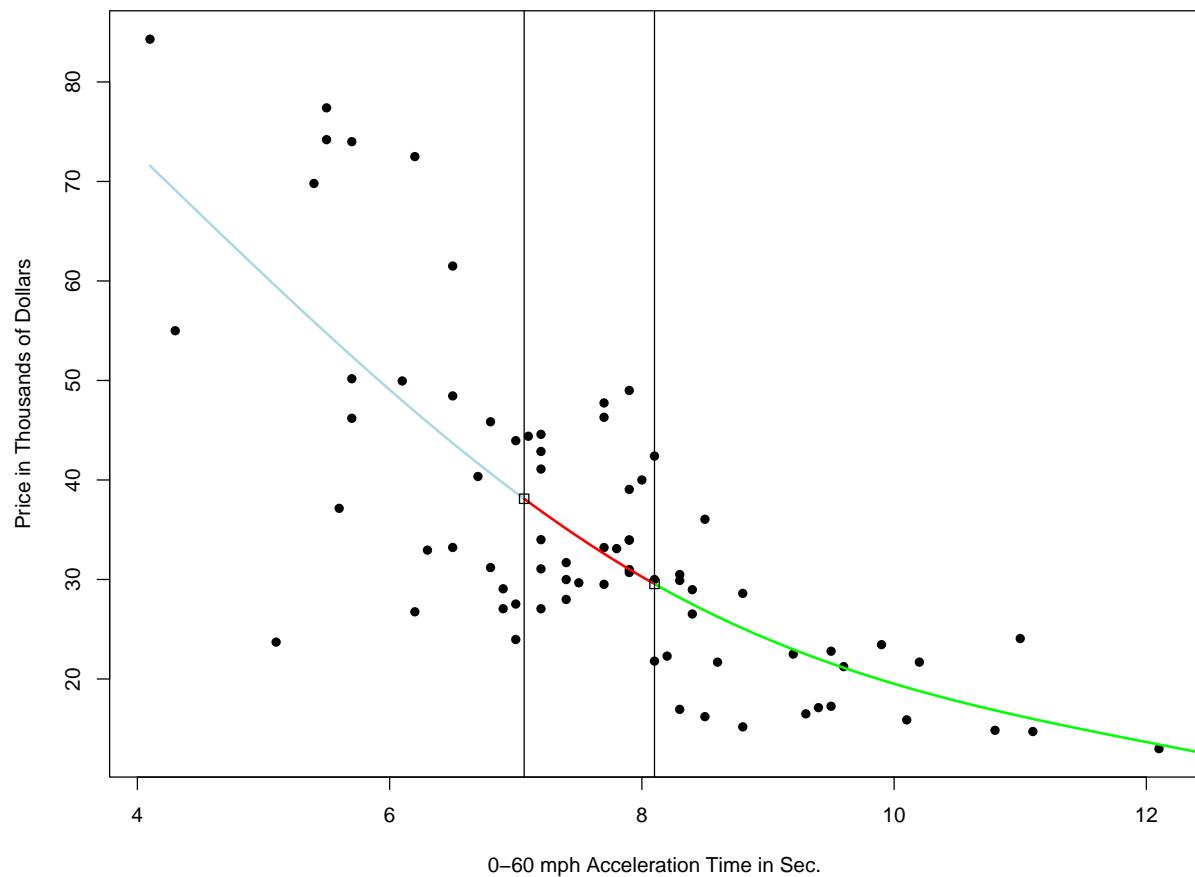
The constant and linear models have high bias, as they are not complex enough to capture the apparent curvature in the relationship between price and acceleration time.

A cubic model, on the other hand might better capture the trend.



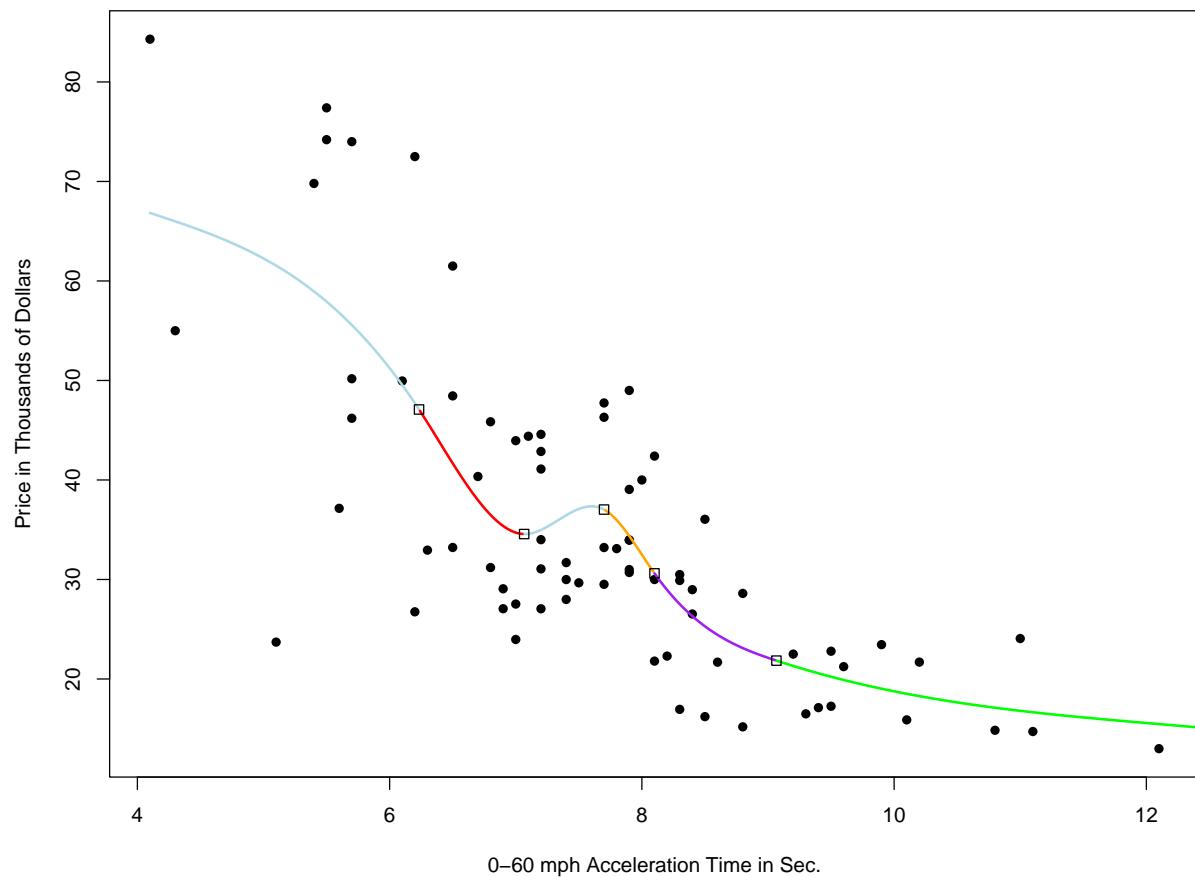
7.5.3 Cubic Splines

It's possible that the behavior of the response variable might differ in different regions of the x-axis. A cubic spline allows us to fit different models in different regions of the x-axis.

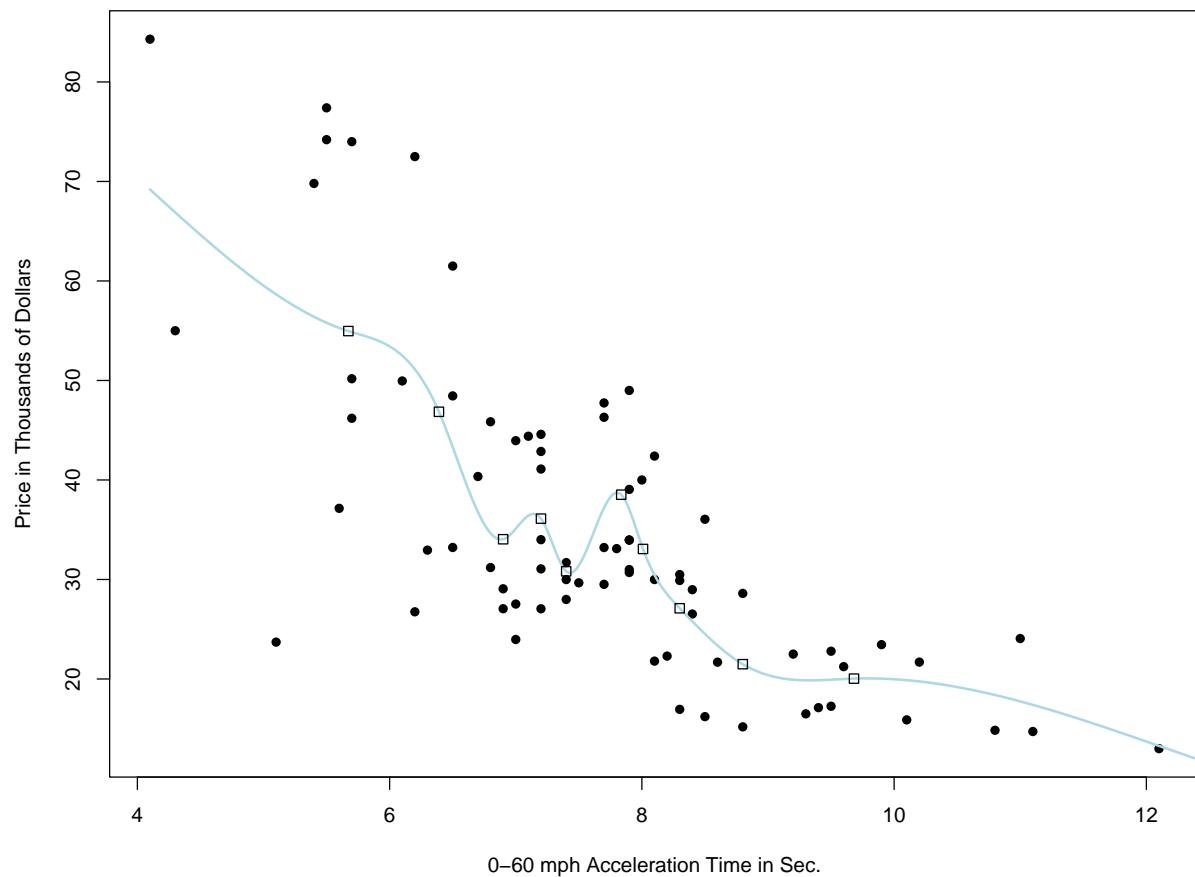


The region boundaries are called **knots**

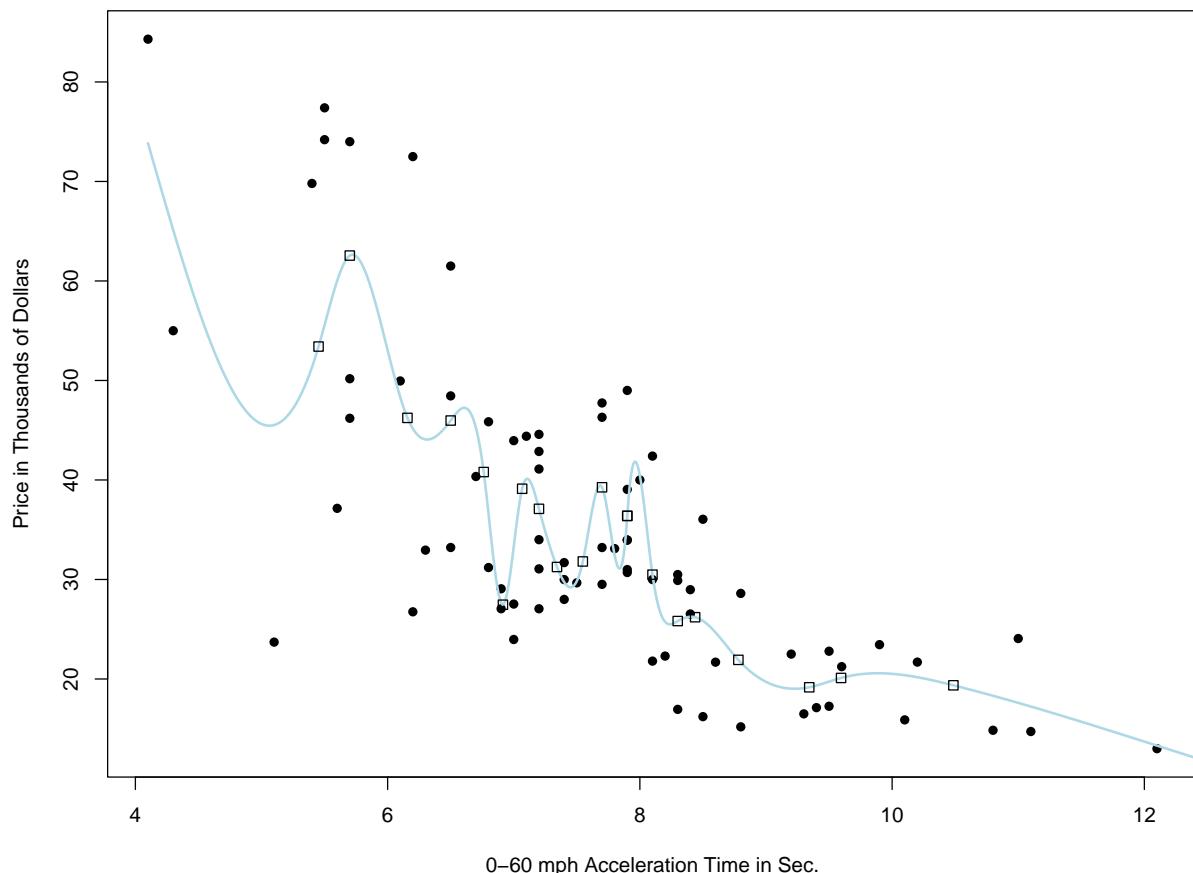
Cubic Spline with 5 Knots



Cubic Spline with 10 Knots



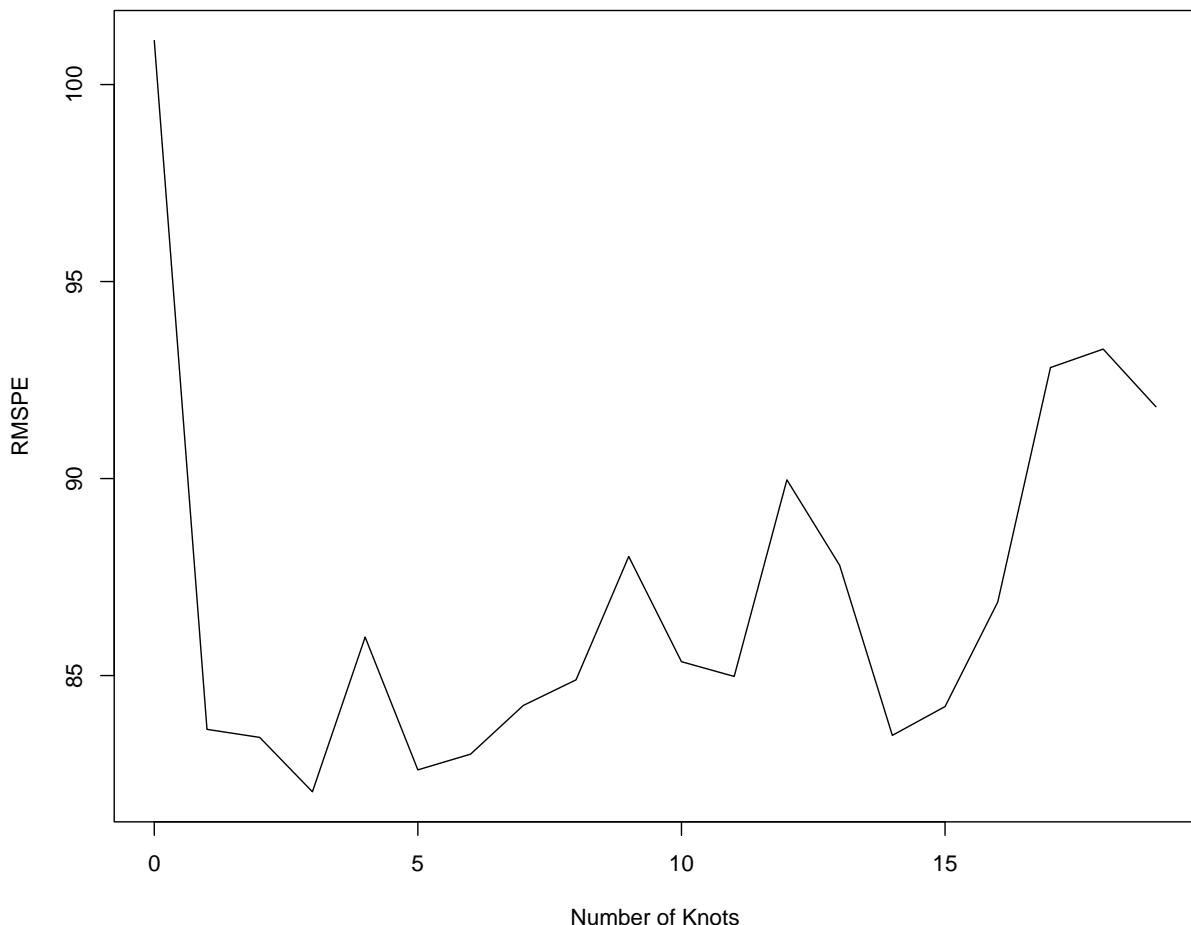
Cubic Spline with 20 Knots



Notice that as the number of knots increases, the model becomes more and more complex. We would not expect the relationship between price and acceleration time to look like it does in these more complicated pictures. It is likely that as the number of knots gets big, the model overfits the training data.

7.5.4 Predicting Test Data

Shown below is a plot of RMSPE when predictions are made on the new test data.



We see that RMSPE is minimized using the model with three knots.

7.5.5 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.

7.6 Summary and Comparison

In the previous sections, we've applied various predictive modeling techniques to predict house prices in Ames, IA. In each section, we've focused on an individual predictive technique (OLS, ridge/lasso regression, trees, splines), but in practice, we often test out these techniques together to find which is likely to perform best on a set of data. Here, we'll go through the steps to test out and evaluate these techniques on the Ames Housing dataset.

There are no new statistical ideas presented in this section, just a synthesis of the preceding material. We leave out splines, since we did not discuss using splines in a multivariate setting, but we compare OLS, ridge and decision trees.

We use a subset of variables for illustrative purposes.

```
set.seed(10302021)
samp <- sample(1:nrow(ames_raw), 1000)
Ames_Houses <- ames_raw[samp, ]
```

```
New_Houses <- ames_raw <- ames_raw[-samp, ]
New_Houses <- New_Houses[1:5, ]
```

We'll begin by doing some data preparation.

We standardize all explanatory variables in the training and new data. We do not standardize the response variable, price, so we can interpret predicted values more easily.

```
Houses_Combined <- rbind(Ames_Houses, New_Houses)
Houses_sc <- Houses_Combined %>% mutate_if(is.numeric, scale)
Houses_sc$SalePrice <- as.numeric(Houses_Combined$SalePrice)
Houses_sc_Train <- Houses_sc[1:1000, ]
Houses_sc_New <- Houses_sc[1001:1005, ]
```

The `Houses_sc_Train` dataset contains standardized values for the 1000 houses in the training data. The first six rows are shown below.

```
head(Houses_sc_Train)
```

	Overall.Qual	Year.Built	Central.Air	Gr.Liv.Area	X1st.Flr.SF
2771	-0.07619084	0.79212207		Y -0.003620472	-0.9340706
2909	-0.77229808	0.18658251		Y -0.392399656	-1.3891530
2368	-0.07619084	0.01837707		Y -0.938684253	-1.6993209
2604	-2.16451257	-1.89916487		Y -0.571836202	-1.1908489

669	-0.07619084	-0.14982836		Y	-0.918746859	-0.3111924
1427	2.01213088	1.22945620		Y	0.528707949	1.5345608
	Bedroom.AbvGr	TotRms.AbvGrd	Lot.Area	Lot.Shape	Land.Contour	
2771	0.1632018	-0.2910546	0.28570669	IR1	Lvl	
2909	0.1632018	-0.9160758	-0.88703104	Reg	Lvl	
2368	0.1632018	-0.2910546	-0.98002927	Reg	Lvl	
2604	0.1632018	-0.2910546	0.04235132	Reg	Lvl	
669	0.1632018	-0.2910546	0.17314883	IR1	Lvl	
1427	0.1632018	0.9589877	0.20650820	Reg	Lvl	
	Overall.Cнд	Exter.Qual	Heating.QC	Paved.Drive	SalePrice	
2771	-0.4680319	Gd	Ex	Y	187000	
2909	0.4279149	TA	TA	Y	104500	
2368	1.3238618	TA	Ex	Y	116000	
2604	-1.3639788	TA	TA	Y	105000	
669	-1.3639788	Fa	Gd	Y	163000	
1427	-0.4680319	Ex	Ex	Y	395039	

The `Houses_sc_New` displays standardized values for the new houses that we're trying to predict.

```
head(Houses_sc_New)
```

	Overall.Qual	Year.Built	Central.Air	Gr.Liv.Area	X1st.Flr.SF	Bedroom.AbvGr	
2	-0.77229808	-0.3516749		Y	-1.2058453	-0.6772922	-1.0608118
4	0.61991640	-0.1161873		Y	1.2145543	2.4091326	0.1632018
6	-0.07619084	0.8930453		Y	0.2057222	-0.6010214	0.1632018
7	1.31602364	0.9939686		Y	-0.3246125	0.4464308	-1.0608118
8	1.31602364	0.6911988		Y	-0.4402494	0.2989739	-1.0608118
	TotRms.AbvGrd	Lot.Area	Lot.Shape	Land.Contour	Overall.Cнд	Exter.Qual	
2	-0.9160758	0.1877886	Reg	Lvl	0.4279149	TA	
4	0.9589877	0.1323496	Reg	Lvl	-0.4680319	Gd	
6	0.3339665	-0.0094877	IR1	Lvl	0.4279149	TA	
7	-0.2910546	-0.6164362	Reg	Lvl	-0.4680319	Gd	
8	-0.9160758	-0.6062364	IR1	HLS	-0.4680319	Gd	
	Heating.QC	Paved.Drive	SalePrice				
2	TA	Y	105000				
4	Ex	Y	244000				
6	Ex	Y	195500				
7	Ex	Y	213500				
8	Ex	Y	191500				

Since the `glmnet` command requires training data to be entered as a matrix, we create versions of the datasets in matrix form.

```
Houses_sc$SalePrice[is.na(Houses$SalePrice)] <- 0 #can't take NA's when fitting model matrix
Houses_sc_Combined_MAT <- model.matrix(SalePrice~., data=rbind(Houses_sc))
Houses_sc_Train_MAT <- Houses_sc_Combined_MAT[1:1000, ]
Houses_sc_New_MAT <- Houses_sc_Combined_MAT[1001:1005, ]
```

7.6.1 Modeling with OLS

We first fit an ordinary least squares regression model to the data.

```
Housing_OLS <- lm(data=Houses_sc_Train, SalePrice~ .)
coef(Housing_OLS)
```

	Overall.Qual	Year.Built	Central.AirY	Gr.Liv.Area
238908.0614	23368.4152	14036.2549	-4497.1153	29640.4606
X1st.Flr.SF	Bedroom.AbvGr	TotRms.AbvGrd	Lot.Area	Lot.ShapeIR2
8320.1472	-5011.0485	1554.7785	7566.9377	1570.1676
Lot.ShapeIR3	Lot.ShapeReg	Land.ContourHLS	Land.ContourLow	Land.ContourLvl
19082.7508	-4566.5111	44704.8906	22406.5959	19096.4163
Overall.Cond	Exter.QualFa	Exter.QualGd	Exter.QualTA	Heating.QCFA
7965.6704	-68750.2773	-62800.5804	-74028.3841	-3972.4036
Heating.QCGd	Heating.QCPo	Heating.QCTA	Paved.DriveP	Paved.DriveY
-4478.6876	-23000.4394	-5272.7136	-1901.9235	709.1532

7.6.2 Ridge Regression with Housing Data

Now, we'll use ridge regression to predict insurance costs.

We use cross validation to determine the optimal value of lambda. We perform 10 repeats of 10-fold cross-validation. We test 100 lambda-values ranging from 10^{-5} to 10^5 .

```
control = trainControl("repeatedcv", number = 10, repeats=10)
l_vals = 10^seq(-5, 5, length = 100)

set.seed(2022)
Housing_ridge <- train( SalePrice ~ ., data = Houses_sc_Train, method = "glmnet", trControl=
```



```
Housing_ridge$bestTune$lambda
```

```
[1] 6135.907
```

We fit a model to the full training dataset using the optimal value of *lambda* .

```
ridge_mod <- glmnet(x=Houses_sc_Train_MAT, y=Houses_sc_Train$SalePrice, alpha = 0, lambda=Hou  
coef(ridge_mod)
```

```
26 x 1 sparse Matrix of class "dgCMatrix"  
           s0  
(Intercept) 206079.5442  
(Intercept) .  
Overall.Qual 24716.6502  
Year.Built   12909.8224  
Central.AirY -1693.4832  
Gr.Liv.Area  24137.8185  
X1st.Flr.SF  10707.5001  
Bedroom.AbvGr -5034.2266  
TotRms.AbvGrd 5394.5170  
Lot.Area     7086.7613  
Lot.ShapeIR2 3322.3720  
Lot.ShapeIR3 18987.3176  
Lot.ShapeReg -5345.7478  
Land.ContourHLS 41239.7682  
Land.ContourLow 15011.2269  
Land.ContourLvl 12784.0351  
Overall.Cond  6560.4987  
Exter.QualFa -29042.5581  
Exter.QualGd -24942.6445  
Exter.QualTA -35102.6069  
Heating.QCFa -8118.6371  
Heating.QCGd -6380.1279  
Heating.QCPo -19693.6611  
Heating.QCTA -7645.0855  
Paved.DriveP -613.3798  
Paved.DriveY 1324.2704
```

The regression coefficients are displayed together with the OLS coefficients in a data.frame.

```
Ridge_coef <- as.vector(ridge_mod$beta)[-1] #leave off intercept using [-1]  
OLS_coef <- coef(Housing_OLS)[-1]  
data.frame(OLS_coef, Ridge_coef)
```

```
OLS_coef  Ridge_coef
```

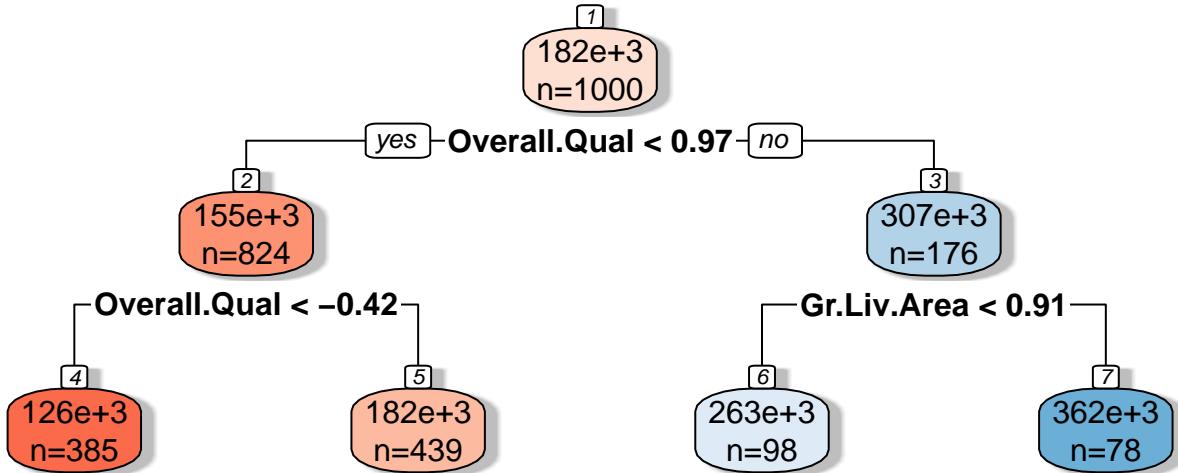
Overall.Qual	23368.4152	24716.6502
Year.Built	14036.2549	12909.8224
Central.AirY	-4497.1153	-1693.4832
Gr.Liv.Area	29640.4606	24137.8185
X1st.Flr.SF	8320.1472	10707.5001
Bedroom.AbvGr	-5011.0485	-5034.2266
TotRms.AbvGrd	1554.7785	5394.5170
Lot.Area	7566.9377	7086.7613
Lot.ShapeIR2	1570.1676	3322.3720
Lot.ShapeIR3	19082.7508	18987.3176
Lot.ShapeReg	-4566.5111	-5345.7478
Land.ContourHLS	44704.8906	41239.7682
Land.ContourLow	22406.5959	15011.2269
Land.ContourLvl	19096.4163	12784.0351
Overall.Cond	7965.6704	6560.4987
Exter.QualFa	-68750.2773	-29042.5581
Exter.QualGd	-62800.5804	-24942.6445
Exter.QualTA	-74028.3841	-35102.6069
Heating.QCFA	-3972.4036	-8118.6371
Heating.QCGd	-4478.6876	-6380.1279
Heating.QCPo	-23000.4394	-19693.6611
Heating.QCTA	-5272.7136	-7645.0855
Paved.DriveP	-1901.9235	-613.3798
Paved.DriveY	709.1532	1324.2704

7.6.3 Decision Tree

Now, we'll predict house prices using using a decision tree.

First, we grow and display a small decision tree, by setting the `cp` parameter equal to 0.05.

```
tree <- rpart(SalePrice~., data=Houses_sc_Train, cp=0.05)
rpart.plot(tree, box.palette="RdBu", shadow.col="gray", nn=TRUE, cex=1, extra=1)
```



Now we use cross-validation to determine the optimal value of the `cp` parameter. We use 10 repeats of 10-fold cross-validation. We test 1000 `cp`-values ranging from 10^{-5} to 10^5 .

```

cp_vals = 10^seq(-5, 5, length = 100)
colnames(Houses_sc_Train) <- make.names(colnames(Houses_sc_Train))

set.seed(2022)
Housing_Tree <- train(data=Houses_sc_Train, SalePrice ~ ., method="rpart", trControl=control)
Housing_Tree$bestTune

```

```

cp
4 0.00002009233

```

We grow a full tree using the optimal `cp` value.

```
Housing_Best_Tree <- rpart(SalePrice~., data=Houses_sc_Train, cp=Housing_Tree$bestTune)
```

7.6.4 Comparing Performance

We use cross-validation to compare the performance of the linear model, ridge regression model, and decision tree.

```

set.seed(2022)
Housing_OLS <- train(data=Houses_sc_Train, SalePrice ~ ., method="lm", trControl=control)

```

```
min(Housing_OLS$results$RMSE)
```

```
[1] 35313.3
```

```
min(Housing_ridge$results$RMSE)
```

```
[1] 35747.22
```

```
min(Housing_Tree$results$RMSE)
```

```
[1] 33675.46
```

The tree predictions give slightly lower RMSPE.

7.6.5 Predictions on New Data

We now predict the sale price of the five new houses using each technique.

Ordinary Least-Squares model:

```
OLS_pred <- predict(Housing_OLS, newdata=Houses_sc_New)
head(OLS_pred)
```

```
2           4           6           7           8
114709.4 253695.8 195078.1 222117.6 242493.9
```

Ridge regression model:

We use the `Customers_sc_New_MAT` dataset, since the `glmnet` package requires inputs in matrix form.

```
ridge_pred <- predict(ridge_mod, newx=Houses_sc_New_MAT)
head(ridge_pred)
```

```
s0
2 114950.4
4 259359.8
6 195288.0
7 226841.6
8 249064.8
```

Decision tree:

```
tree_pred <- predict(Housing_Best_Tree, newdata=Houses_sc_New)
head(tree_pred)
```

2	4	6	7	8
132926.5	287045.1	183978.6	207079.4	207079.4

7.7 Assessing a Classifier's Performance

7.7.1 Measuring Prediction Accuracy

Just as we've done for models with quantitative variables, we'll want to compare and assess the performance of models for predicting categorical responses. This might involve comparing logistic regression models with different explanatory variables, or comparing a regression model to another technique such as a decision tree.

Just as we did before, we'll divide the data so that we can evaluate predictions on a subset of the data that was not used to fit the model.

We'll divide the credit card dataset into a set of 9,000 observations, on which we'll fit our models and assess predictions on the remaining 1,000.

```
set.seed(08172022)
samp <- sample(1:nrow(Default), 1000)
Default_Test <- Default[samp, ]
Default_Train <- Default[-samp, ]
```

We fit the model with interaction to the training data:

```
LR_Default_M_Int <- glm(data=Default_Train, default ~ balance * student, family = binomial(link = "logit"))
summary(LR_Default_M_Int)
```

```
Call:
glm(formula = default ~ balance * student, family = binomial(link = "logit"),
     data = Default_Train)

Coefficients:
              Estimate Std. Error z value      Pr(>|z|)
(Intercept) -11.2714061  0.5188284 -21.725 <0.000000000000002 ***
balance       0.0060696  0.0003273  18.547 <0.000000000000002 ***
studentYes    0.0924588  0.8606304   0.107      0.914
balance:studentYes -0.0004749  0.0005142  -0.924      0.356
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 2617.1  on 8999  degrees of freedom
```

```
Residual deviance: 1385.5 on 8996 degrees of freedom
AIC: 1393.5
```

Number of Fisher Scoring iterations: 8

We then use the model to estimate the probability of a person defaulting on their credit card payment.

Information about 10 different credit card users, as well as the logistic regression estimate of their probability of default are shown below. The table also shows whether or not the user really defaulted on their payment.

```
LR_Prob <- predict(LR_Default_M_It, newdata=Default_Test, type="response") %>% round(2)
Actual_Default <- factor(ifelse(Default_Test$default==1, "Yes", "No"))
student <- Default_Test$student
balance <- Default_Test$balance
LR_Res_df <- data.frame(student, balance, LR_Prob, Actual_Default)
kable(head(LR_Res_df, 50)%>% arrange(desc(LR_Prob)) %>% head(10))
```

	student	balance	LR_Prob	Actual_Default
2465	Yes	2026.864	0.54	No
1228	No	1682.201	0.26	No
6656	No	1551.028	0.14	No
1185	No	1541.813	0.13	No
9963	Yes	1635.175	0.12	No
6635	No	1434.128	0.07	Yes
9691	No	1391.318	0.06	No
5921	Yes	1513.542	0.06	No
9755	No	1233.619	0.02	No
7569	Yes	1294.286	0.02	No

7.7.2 Decision Tree Classifier

For comparison, let's use a decision tree to predict whether a person will default.

In a binary classification problem, we can treat a default as $y = 1$ and non-default as $y = 0$, and grow the tree as we would in regression.

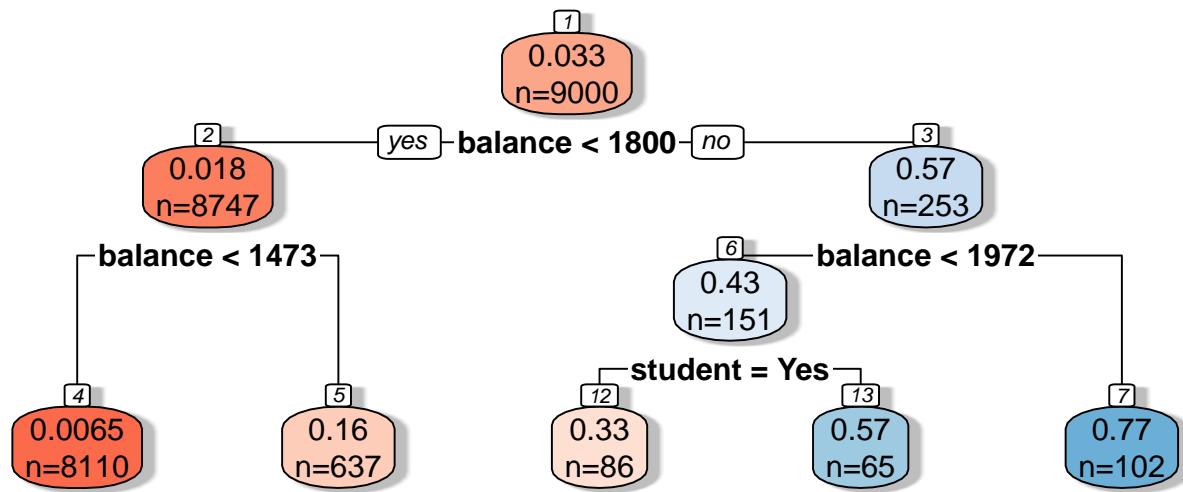
The mean response in a node \bar{Y} , which is equivalent to the proportion of people in the node who defaulted, can be interpreted as the probability of default.

The first few splits of the tree are shown.

```

library(rpart)
library(rpart.plot)
# grow shorter tree for illustration
tree <- rpart(data=Default_Train, default~balance + student, cp=0.005)
rpart.plot(tree, box.palette="RdBu", shadow.col="gray", nn=TRUE, cex=1, extra=1)

```



```

# grow full tree
tree <- rpart(data=Default_Train, default~balance + student)

```

```
Tree_Prob <- predict(tree, newdata = Default_Test) %>% round(2)
```

We add the decision tree probabilities to the table seen previously.

```

LR_Res_df <- data.frame(student, balance, LR_Prob, Tree_Prob, Actual_Default)
kable(head(LR_Res_df, 50)) %>% arrange(desc(Tree_Prob)) %>% head(10)

```

	student	balance	LR_Prob	Tree_Prob	Actual_Default
2465	Yes	2026.864	0.54	0.77	No
1228	No	1682.201	0.26	0.16	No
6656	No	1551.028	0.14	0.16	No
1185	No	1541.813	0.13	0.16	No
9963	Yes	1635.175	0.12	0.16	No
6635	No	1434.128	0.07	0.01	Yes
9691	No	1391.318	0.06	0.01	No
5921	Yes	1513.542	0.06	0.16	No

	student	balance	LR_Prob	Tree_Prob	Actual_Default
9755	No	1233.619	0.02	0.01	No
7569	Yes	1294.286	0.02	0.01	No

We see that the tree estimates that the first person has a 0.77 probability of defaulting on the payment, compared to an estimate of 0.54, given by the logistic regression model. On the other hand, the tree estimates only a 0.16 probability of the second person defaulting, compared to 0.26 for the logistic regression model.

7.7.3 Assessing Classifier Accuracy

We've seen $\text{RMSPE} = \sqrt{\sum_{i=1}^n (\hat{y}_i - y_i)^2}$ used as a measure of predictive accuracy in a regression problem.

Since our outcome is not numeric, this is not a good measure of predictive accuracy in a classification problem. We'll examine some alternatives we can use instead.

Classification Accuracy

One simple approach is calculate the proportion of credit card users classified correctly. If a person has model estimates a predicted probability of default greater than 0.5, the person is predicted to default, while if the probability estimate is less than 0.5, the person is predicted to not default.

The table shows the prediction for each of the 10 users, using both logistic regression and the decision tree.

```
LR_Pred <- factor(ifelse(LR_Prob > 0.5, "Yes", "No"))
Tree_Pred <- factor(ifelse(Tree_Prob > 0.5, "Yes", "No"))
LR_Res_df <- data.frame(student, balance, LR_Prob, Tree_Prob, LR_Pred, Tree_Pred, Actual_Default)
kable(head(LR_Res_df, 50) %>% arrange(desc(LR_Prob)) %>% head(10))
```

	student	balance	LR_Prob	Tree_Prob	LR_Pred	Tree_Pred	Actual_Default
2465	Yes	2026.864	0.54	0.77	Yes	Yes	No
1228	No	1682.201	0.26	0.16	No	No	No
6656	No	1551.028	0.14	0.16	No	No	No
1185	No	1541.813	0.13	0.16	No	No	No
9963	Yes	1635.175	0.12	0.16	No	No	No
6635	No	1434.128	0.07	0.01	No	No	Yes
9691	No	1391.318	0.06	0.01	No	No	No

	student	balance	LR_Prob	Tree_Prob	LR_Pred	Tree_Pred	Actual_Default
5921	Yes	1513.542	0.06	0.16	No	No	No
9755	No	1233.619	0.02	0.01	No	No	No
7569	Yes	1294.286	0.02	0.01	No	No	No

Notice that although the probabilities differ, the logistic regression model and classification tree give the same predictions for these ten cases. Both correctly predict 8 out of the 10 cases, but mistakenly predict the first person to default, when they didn't, and mistakenly predict that the sixth person would not default when they did.

We'll check the classification accuracy for the model and the tree.

```
sum(LR_Pred == Actual_Default)/1000
```

```
[1] 0.972
```

```
sum(Tree_Pred == Actual_Default)/1000
```

```
[1] 0.971
```

We see that the two techniques are each right approximately 97% of the time.

This may not really be as good as it sounds. Can you think of a very simple classification strategy that would achieve a similarly impressive predictive accuracy on these data?

7.7.4 Confusion Matrix

In addition to assessing overall accuracy, it is sometimes helpful to assess how well models are able to predict outcomes in each class. For example, how accurately can a model detect people who do actually default on their payments?

A **confusion matrix** is a two-by-two table displaying the number of cases predicted in each category as columns, and the number of cases actually in each category as rows

	Actually Negative	Actually Positive
Predicted Negative	# True Negative	# False Negative
Predicted Positive	# False Positive	# True Positive

The `confusionMatrix` matrix command in R returns the confusion matrix for all 1,000 test cases.

Let's look at the confusion matrix for all 1,000 test cases. The `data` argument is the predicted outcome, and the `reference` argument is the true outcome. The `positive` argument is the category that we'll classify as a positive.

Logistic Regression Confusion Matrix

```
confusionMatrix(data=LR_Pred, reference=factor(Actual_Default) , positive="Yes")
```

Confusion Matrix and Statistics

		Reference	
		No	Yes
Prediction	No	957	20
	Yes	8	15

Accuracy : 0.972
95% CI : (0.9598, 0.9813)
No Information Rate : 0.965
P-Value [Acc > NIR] : 0.12988

Kappa : 0.5035

McNemar's Test P-Value : 0.03764

Sensitivity : 0.4286
Specificity : 0.9917
Pos Pred Value : 0.6522
Neg Pred Value : 0.9795
Prevalence : 0.0350
Detection Rate : 0.0150
Detection Prevalence : 0.0230
Balanced Accuracy : 0.7101

'Positive' Class : Yes

Out of 965 people who did not default, the logistic regression model correctly predicted 957 of them.

Out of 35 people that did default, the model correctly predicted 15 of them.

Tree Confusion Matrix

```
# data is predicted class  
# reference is actual class  
confusionMatrix( data = Tree_Pred , reference= Actual_Default, "Yes")
```

Confusion Matrix and Statistics

		Reference
Prediction	No	Yes
No	960	24
Yes	5	11

Accuracy : 0.971
95% CI : (0.9586, 0.9805)
No Information Rate : 0.965
P-Value [Acc > NIR] : 0.1724819

Kappa : 0.4186

McNemar's Test P-Value : 0.0008302

Sensitivity : 0.3143
Specificity : 0.9948
Pos Pred Value : 0.6875
Neg Pred Value : 0.9756
Prevalence : 0.0350
Detection Rate : 0.0110
Detection Prevalence : 0.0160
Balanced Accuracy : 0.6546

'Positive' Class : Yes

Out of 965 people who did not default, the logistic regression model correctly predicted 960 of them.

Out of 35 people that did default, the model correctly predicted 11 of them.

Notice that the tree was less likely to predict a person to default in general, returning only 16 positive predictions, compared to 23 for the logistic regression model.

7.7.5 Sensitivity and Specificity

The **sensitivity** of a classifier is the proportion of all positive cases that the model correctly identifies as positive. (i.e. probability model says “positive” given actually is positive.)

$$\text{Sensitivity} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} = \frac{\text{Correctly Predicted Positives}}{\text{Total Number of Actual Positives}}$$

LR Sensitivity

$$\frac{15}{15 + 20} \approx 0.4286$$

Tree Sensitivity

$$\frac{11}{11 + 24} \approx 0.3143$$

The **specificity** of a classifier is the proportion of all negative cases that the model correctly identifies as negative (i.e. probability model says “negative” given truly is negative.)

$$\text{Specificity} = \frac{\text{True Negative}}{\text{True Negative} + \text{False Positive}} = \frac{\text{Correctly Predicted Negatives}}{\text{Total Number of Actual Negatives}}$$

LR Specificity

$$\frac{957}{957 + 8} \approx 0.9917$$

Tree Specificity

$$\frac{960}{960 + 5} \approx 0.9948$$

In a given situation, we should think about the cost of a false negative vs a false positive when determining whether to place more weight on sensitivity or specificity. For example, “is it worse to tell a patient they tested positive for a disease when they really don’t have it, or to not tell them they tested positive when they really do have it?”

7.8 Receiver Operating Characteristic Curve

7.8.1 Separating +'s and -'s

The prediction accuracy, sensitivity, and specificity measures, seen in the previous section are based only on the predicted outcome, without considering the probability estimates themselves. These techniques treat a 0.49 estimated probability of default the same as a 0.01 estimated probability.

We would hope to see more defaults among people with high estimated default probabilities than low ones. To assess this, we can list the people in order from highest to lowest probability estimates and see where the true defaults lie.

For example, consider the following fictional probability estimates produced by two different classifiers (models) for eight credit card users:

Classifier 1

	Classifier1_Probability_Estimate	True_Outcome
1	0.90	Yes
2	0.75	Yes
3	0.60	No
4	0.40	Yes
5	0.30	No
6	0.15	No
7	0.05	No
8	0.01	No

Classifier 2

	Classifier2_Probability_Estimate	True_Outcome
1	0.80	Yes
2	0.70	No
3	0.55	No
4	0.40	Yes
5	0.35	No
6	0.15	No
7	0.10	Yes
8	0.02	No

Classifier 1 is better able to separate the “Yes’s” from “No’s” as the three true “Yes’s” are among the four highest probabilities. Classifier 2 is less able to separate the true “Yes’s” from true “No’s.”

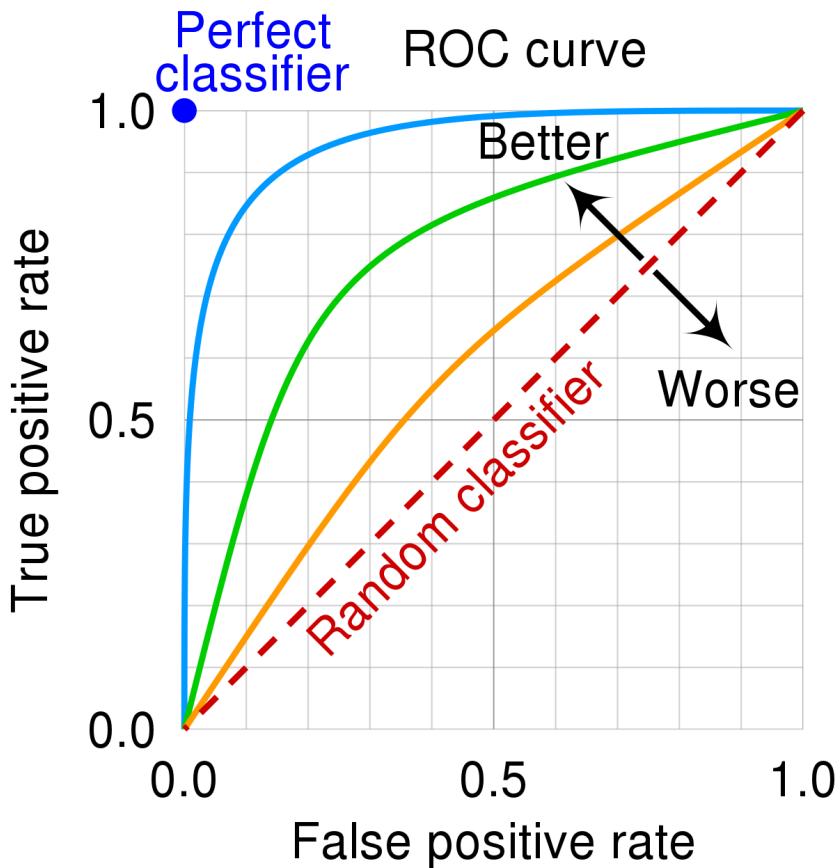
7.8.2 ROC Curve

A receiver operating characteristic (ROC) curve tells us how well a predictor is able to separate positive cases from negative cases.

The blog (Toward Data Science) [<https://towardsdatascience.com/applications-of-different-parts-of-an-roc-curve-b534b1aafb68>] writes

“Receiver Operating Characteristic (ROC) curve is one of the most common graphical tools to diagnose the ability of a binary classifier, independent of the inherent classification algorithm. The ROC analysis has been used in many fields including medicine, radiology, biometrics, natural hazards forecasting, meteorology, model performance assessment, and other areas for many decades and is increasingly used in machine learning and data mining research [1]. If you are a Data Scientist, you might be using it on a daily basis.”

The ROC curve plots the true positive (or hit) rate against the false positive rate (false alarm) rate, as the cutoff for a positive classification varies.



The higher the curve, the better the predictor is able to separate positive cases from negative ones.

Predictions made totally at random would be expected to yield a diagonal ROC curve.

7.8.3 Constructing ROC Curve

1. Order the probabilities from highest to lowest.
2. Assume only the case with the highest probability is predicted as a positive.
3. Calculate the true positive rate (hit rate)

$$\frac{\# \text{ True Positives}}{\# \text{ Actual Positives}}$$

and false positive (false alarm)

$$\frac{\# \text{ False Positives}}{\# \text{ Actual Negatives}}$$

rate.

4. Plot the point

$$\left(\frac{\# \text{ False Positives}}{\# \text{ Actual Negatives}}, \frac{\# \text{ True Positives}}{\# \text{ Actual Positives}} \right)$$

in the coordinate plane.

5. Now assume the cases with the two highest probabilities are predicted as positives, and repeat steps 3-4.
6. Continue, by classifying one more case as positive in each step.

7.8.4 Construct ROC Example

Let's practice constructing an ROC curve for a small set of probability estimates.

```
prob <- c(0.9, 0.8, 0.7, 0.65, 0.45, 0.3, 0.2, 0.15, 0.1, 0.05)
Actual <- c("+", "-", "+", "+", "-", "-", "-", "+", "-")
Hit_Rate <- c("1/4", "1/4", "2/4", "", "", "", "", "", "", "")
FA_Rate <- c("0/6", "1/6", "1/6", "", "", "", "", "", "", "")
kable(data.frame(prob, Actual, Hit_Rate, FA_Rate))
```

prob	Actual	Hit_Rate	FA_Rate
0.90	+	1/4	0/6
0.80	-	1/4	1/6
0.70	+	2/4	1/6
0.65	+		
0.45	-		
0.30	-		
0.20	-		
0.15	-		
0.10	+		
0.05	-		

Finish filling in the table and sketch a graph of the resulting ROC curve.

Question: If the probability estimate of 0.45 were instead 0.5 or 0.55, would this change the ROC curve? Why or why not?

7.8.5 AUC

The area under the ROC curve, (AUC) provides a measure of the model's predictive strength.

While there is no standard for what constitutes a good" AUC, higher is better, and AUC" is useful for comparing models.

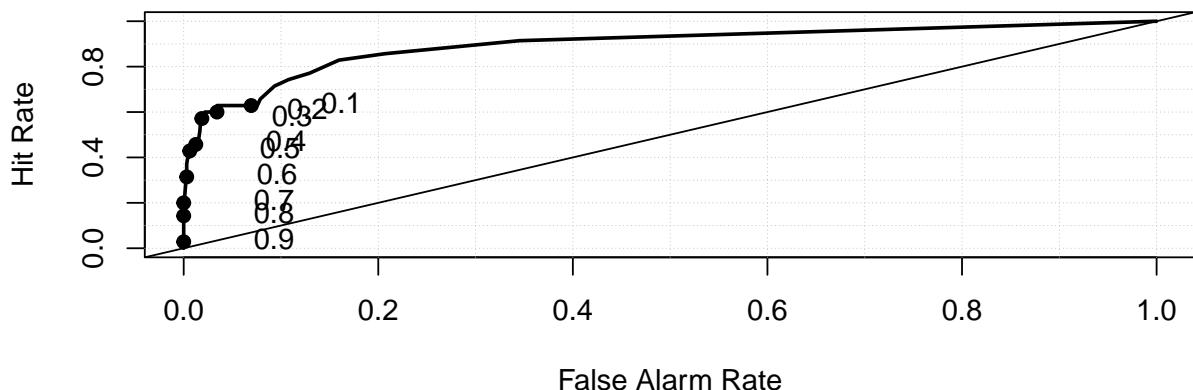
A model that can perfectly separate successes from failures will have an AUC of 1.

A model that assigns probabilities at random is expected to have an AUC of 0.5.

7.8.6 LR and Tree ROC Curves

```
library(pROC)
library(verification)
roc.plot(x=Default_Test$default, pred = LR_Prob)
```

ROC Curve

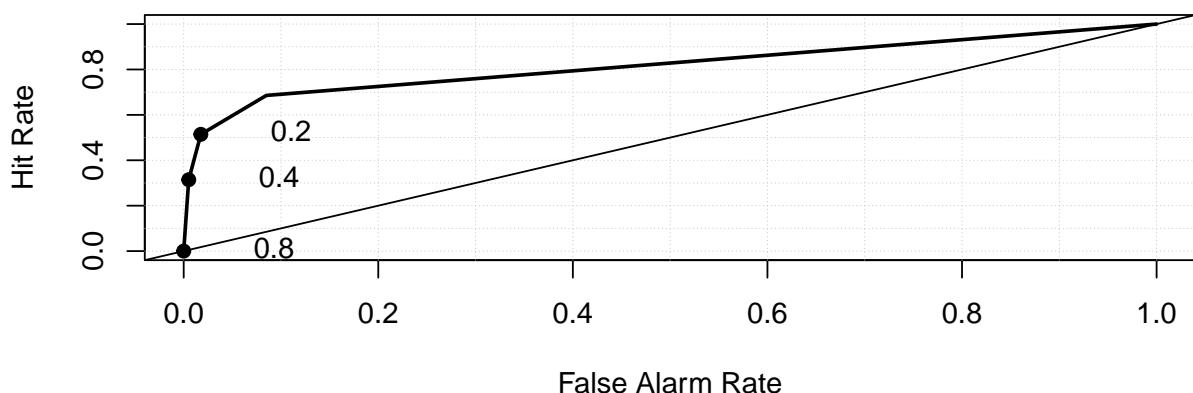


```
auc(response=Default_Test$default, predictor = LR_Prob)
```

Area under the curve: 0.8953

```
roc.plot(x=Default_Test$default, pred = Tree_Prob)
```

ROC Curve

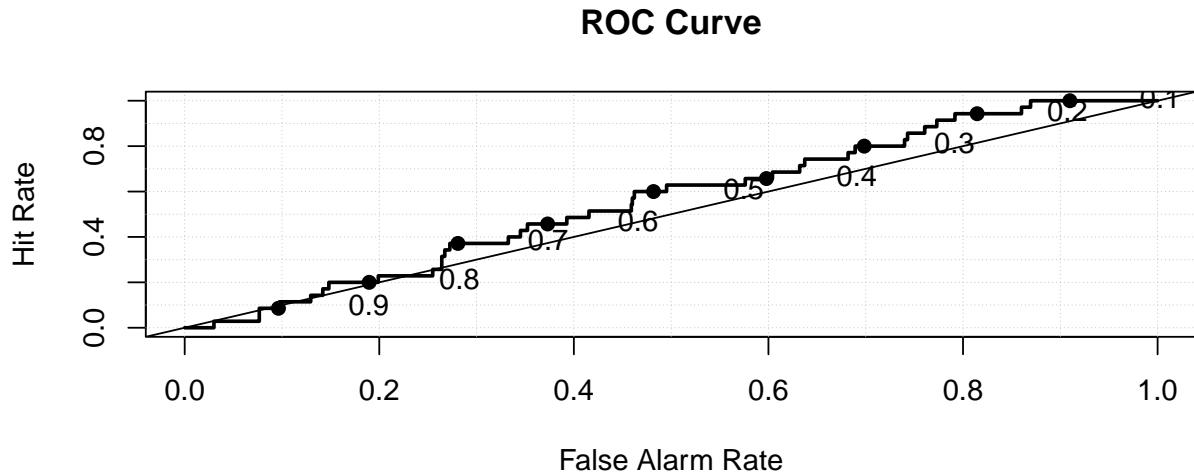


```
auc(response=Default_Test$default, predictor = Tree_Prob)
```

Area under the curve: 0.8176

```
RandProb <- runif(1000, 0, 1)
```

```
roc.plot(x=Default_Test$default, pred = RandProb)
```



```
auc(response=Default_Test$default, predictor = RandProb)
```

Area under the curve: 0.563

Even though a model that assigns predictions randomly, with 97% predicted as negatives will have a high accuracy rate, it will yield a poor ROC curve indicating an inability to separate positive cases from negative ones.

7.9 Ethical Considerations in Predictive Modeling

7.9.1 Assumptions in Predictive Models

Like any other statistical technique, predictive inference (sometimes done through machine learning algorithms) depends on the validity of assumptions.

1. The response variable observed in the data is actually the thing we want to predict
2. Training/Test data representative of population of interest
3. Prediction accuracy is appropriate metric

Below are some examples of real uses of predictive inference in which some of these assumptions were violated, leading to inappropriate and unethical conclusions.

7.9.2 Amazon Hiring Algorithm

In 2014, Amazon began working on an algorithm to predict whether a job applicant would be suitable for hire for software developer positions, based on characteristics of their job application.

response variable: rating of candidate's strength (1-5) explanatory variables: many variables based on information included on the resume (e.g. highest degree, major, GPA, college/university, prior job experiences, internships, frequency of certain words on resume, etc.)

The algorithm was trained using data from past applications, rated by humans, over the past 10 years. It could then be used to predict ratings of future job applicants.

According to [Reuters]](<https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scaps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G>),

"In effect, Amazon's system taught itself that male candidates were preferable. It penalized resumes that included the word "women's," as in "women's chess club captain." And it downgraded graduates of two all-women's colleges, according to people familiar with the matter."

While the algorithm was intended to predict candidate quality, the response variable on the training data actually reflected biases in past hiring decisions, leading the algorithm to do the same.

7.9.3 Facial Recognition

Facial recognition technology is used by law enforcement surveillance, airport passenger screening, and employment and housing decisions. It has, however, been banned for use by police in some cities, including San Francisco and Boston, due to concerns about inequity and privacy.

[Research](#) has shown that although certain facial recognition algorithms achieve over 90% accuracy overall, accuracy rate is lower among subjects who are female, Black, or 18-30 years old.

This is likely due, at least in part, to the algorithms being trained primarily on data from images of people who are not members of these groups.

Although the algorithms might attain strong accuracy overall, it is inappropriate to evaluate them on this basis, without accounting for performance on subgroups in the population.

7.9.4 Comments

The biases and assumptions noted above are not reasons to abandon predictive modeling, but rather flaws to be aware of and work to correct.

Predictive algorithms, are only as good as the data on which they are trained and the societies in which they are developed, and will reflect inherent biases. Thus, they should be used cautiously and with human judgment, just like any other statistical technique.

Beware of statements like:

“The data say this!”

“The algorithm is objective.”

“The numbers don’t lie.”

Any data-driven analysis depends on assumptions, and sound judgment and awareness of context are required when assessing the validity of conclusions drawn.

7.9.5 Modeling 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.
- Be sure that model is predicting what you intend it to
 - Reflective of biases inherent in the data on which it was trained