# Statistical Analysis in Sociology

*Aaron Gullickson*

*2019-03-15*

# Contents

**Preface** **5**

**1 Understanding Data** **7**
What Does Data Look Like? . . . 7
What Can We Do With Data? . . . 9

**2 The Distribution of a Variable** **13**
What is a Distribution? . . . 13
Measuring the Center of a Distribution . . . 19
Percentiles and the Five Number Summary . . . 24
Measuring the Spread of a Distribution . . . 26

**3 Measuring Association** **31**
The Two-Way Table . . . 31
Odds ratio (Advanced) . . . 35
Mean Differences . . . 37
Scatterplot and Correlation Coefficient . . . 40

**4 Statistical Inference** **51**
The Problem of Statistical Inference . . . 51
The Concept of the Sampling Distribution . . . 53
Confidence Intervals . . . 60
Hypothesis Tests . . . 67

**5 Building Models** **77**
The OLS Regression Line . . . 77
The Power of Controlling for Other Variables . . . 92
Including Categorical Variables as Predictors . . . 99
Interaction Terms . . . 107

**A Using Scripts** **117**
Getting Started with Scripts . . . 117

**B Object Types** **123**
Atomic Modes . . . 123
Vectors and Matrices . . . 124
Factors . . . 127
Logical Values and Boolean Statements . . . 128
Missing Values . . . 129
Lists . . . 130
Data Frames . . . 132

**C Pretty Pictures** **137**
Base Plot . . . . . 139
ggplot . . . . . 157

**D Reading and Writing Data** **169**
Data Formats . . . . . 169
plain text files . . . . . 169
Data in binary format . . . . . 174
Saving data . . . . . 175

# Preface

This PDF combines all of the module sections on Canvas for Sociology 312 into a single document.

# Chapter 1

# Understanding Data

## What Does Data Look Like?

The data that we look at typically take the format of a "spreadsheet" with rows and columns. The table below shows some characteristics of four randomly drawn passengers from the Titanic, in this type of spreadsheet format.

| survival | sex | age | agegroup | pclass | fare | family |
|---|---|---:|---|---|---:|---:|
| Survived | Female | 24.0000 | Adult | First | 69.3000 | 0 |
| Died | Male | 24.0000 | Adult | Third | 7.7958 | 0 |
| Survived | Male | 0.9167 | Child | First | 151.5500 | 3 |
| Died | Male | 60.0000 | Adult | First | 26.5500 | 0 |

Clearly, we can see variation in who survived and died, the passenger classes they were housed in, gender, and age. We also have a measure of the fare they paid for the trip (in English pounds) and the number of family members traveling with them. To understand how to think about data, we need to understand the concepts of an **observation** and a **variable** and the distinction between them.

### The observations

The **observations** are what you have on the rows of your dataset. In the Titanic example, the observations are individual passengers on board the Titanic, but observations can take many different forms.

We use the term unit of analysis to distinguish what kind of observation you have in your dataset. If you are interviewing individual people and recording their responses, then the unit of analysis is individual people. If you are collecting cross-national data by country, then the unit of analysis would be a country. If you are analyzing data on the "best colleges in the US" then the unit of analysis is a university/college. The most common unit of analysis that we will see in this course is an individual person, but several of our datasets involve other units of analysis and it is important to keep in mind that an observation can be many different kinds of things.

### The variables

The **variables** are what you have on the columns of your dataset. Variables measure specific attributes of your observations. If you conduct a survey of individual people and ask them for their age, gender, and education, then these three attributes would be recorded as variables in your dataset. We refer to them as "variables" because they can take different values across the observations. If you were to conduct a survey of individual people and ask your respondents if they are human, then you probably wouldn't have a proper variable because everyone would likely respond "yes" and there would be no variation (although we can't necessarily rule out jedis.

There are two major types of variables. Some variables measure quantities of something and thus can be represented by a number. We refer to these as **quantitative variables**. Other variables indicate a category to which the observation belongs. We refer to these as **categorical variables**.

### Quantitative variables

Quantitative variables measure quantities of something. A person's height, a worker's hourly wage, the number of children that a woman has given birth to, a country's gross domestic product, a US State's poverty rate, and the percent of a university's student body that are women are all examples of quantitative variables. They can all be represented by a number which indicates how much of the thing the observation has.

There are two important sub-types of quantitative variables. **Discrete variables** can logically only take certain values within a range, while **continuous variables** can logically take any value within a range. The most common example of a discrete variable is a count variable. The number of children that a woman has given birth to is an example of a count variable. This number can only take the value of whole numbers (integers) such as 0, 1, 2, 3, and so on. It makes no sense if a respondent says they have given birth to 2.5 children. Count variables are discrete variables because only whole numbers are logical responses.

A person's height is an example of a continuous variable. It is true that we typically measure height only down to a certain level of precision, typically inches in the United States. We might think that if we were to measure a person's height in inches, it would only take whole number values and therefore it is discrete. But limitations in measurement don't define whether a variable is continuous or discrete. Rather the distinction is whether the value could be logically measured to any degree of accuracy. We often measure height out to half inches and we could imagine that if we have a precise enough measurement instrument, we could measure a person's height out to any decimal level that we desired. So, it is perfectly sensible for someone to say they were 69.825467 inches tall, even though we might think they are being a bit tedious.

Note that in both the height and the number of children examples, there are logical limits to the values. You can't have negative children or height. There are no exact upper limits to the values that either can take, but we would likely think we have a data coding error if we saw a report of a 20 foot person or a woman who gave birth to 50 children. This is what I mean by the statement "within a range" above. Both discrete and continuous variables can be limited in the range of values that they can take. What distinguishes them from each other is what values they can logically take within that limited range.

### Categorical variables

Categorical variables are not represented by numerical quantities but rather by a set of mutually exclusive categories to which observations can belong. The gender, race, political party affiliation, and highest educational degree of a person, the public/private status of a university, and the passenger class of a passenger on the Titanic are all examples of categorical variables.

There are also two sub-types of categorical variables. **Ordinal variables** are categorical variables where the categories have an explicit ordered structure, while **nominal variables** are categorical variables where the categories are unordered. Highest educational degree is an example of an ordinal variables because it is ordered such that Graduate Degree > BA Degree > AA Degree > High School Diploma > No degree. Passenger class is also an ordinal variables that starts in Third class (or steerage - Think Leonardo DiCaprio) and ends in First class (think Kate Winslet), with a Second class in between.

Race, gender, and political party affiliation are all examples of nominal variables. The categories here have no ordering to them. While some people might have their own political party preferences, these sort of normative evaluations of categories are irrelevant. For the same reason, even the variable of survival on the Titanic is a nominal variable. We don't judge the value of life and death.

---

# What Can We Do With Data?

We now know what data looks like, but what do social scientists do with data? in the first part of this course, we will learn three fundamental data analysis tasks: analysis of the distribution of a single variable, measuring association, and statistical inference. In the final part of the course, we will build on these fundamentals to learn how to build more complex statistical models.

## How is a variable distributed?

Sometimes, we just want to understand what a single variable "looks like." We may simply be interested in its "average" or we may want to know something else, like how spread out the values of the variable are. In these cases, we calculate **univariate** ("one variable) statistics on the **distribution** of a variable. Typically, univariate statistics aren't as interesting to social scientists as the measures of association discussed below, but even then its often a good idea to look at univariate statistics to understand all of the variables in your research. In some cases, the calculation of a univariate statistics is the important question at hand. For example, when poll researchers try to figure out who is going to win an election, they are very much interested in the univariate distribution of support for each candidate, which gives the proportions of likely voters who intend to vote for each candidate. Here are some other questions we could ask in our data:

- How much variability is there in the amount of money that movies make?
- What percent of passengers survived the Titanic disaster?
- What is the average age of voters in the United States?

## Measuring association

Social scientists are often most interested in the relationships, or **association**, between two or more variables. These associations allow us to test hypotheses about causal relationships between underlying social constructs. For example, we might be interested in whether divorce affected children's well-being. In this case, we would want to look at the relationship between the categorical variable indicating whether a child's parents were divorced and some measure of their well-being, such as feelings of stress, academic performance, etc. There are different ways of measuring association depending on the types of variables involved.. Here are some questions about association we could ask in our data:

- Did the probability of surviving the Titanic depend on passenger class? (categorical and categorical)
- Do the earnings of movies vary by genre? (quantitative and categorical)
- Is income inequality in a state related to its crime rate? (quantitative and quantitative)

In the first part of the course, we will learn how basic measures of association between two variables, depending on what kind of variables we are using. In the final part of the course, we will return to this topic when we learn how to build more complex statistical models discussed below.

## Making statistical inferences

If I told you that in my sample of twenty people, brown-eyed individuals make $5 more than all other eye colors combined, would you believe me? You probably shouldn't, because in a sample of twenty people, even when drawn without systematic bias, weird results like this are not unlikely just by random chance. If I told you I observed this phenomenon on a well-drawn sample of 20,000 individuals, you would probably be more likely to believe me.

The underlying concept here is called **statistical inference**. We often want to draw conclusions about the larger populations from which our samples are drawn. Statistical inference is the technique of quantifying how uncertain we are about whether our data are similar to the population or not. When you hear press reports on political polls with the term "margin of error," they are referring to statistical inference.

Many introductory statistics course focus most of their attention on statistical inference, partly because it is more abstract and complex. However, statistical inference is always secondary to the basic descriptive measures of univariate, bivariate, and multivariate statistics. Therefore, I spend considerably less time on this topic than in most statistics courses, so that we can focus on the more important stuff.

## Building Models

Although our basic measures of association are useful, the most common tool in social science analysis is a **statistical model** in which the user can specify the relationships between variables by some kind of mathematical functions. In the final module of the course, we will learn how to build basic versions of these models that allow us to look at the relationships between multiple quantitative and categorical variables. This section will build on our prior work in all of the previous modules. We will specifically focus on two uses of statistical models.

First, statistical models will allow us to "control" for other variables when we look at the association between any two variables. Controlling for other variables is important because they may be confounded with the relationship we want to measure. For example, we may be interested in the relationship between marital status (e.g. never married, married, widowed, divorced) and sexual frequency in our GSS data. However, these different groups vary significantly in their age. Never married individuals are much younger than all of the other groups and widowed individuals much older. Given the fact that sexual frequency tends to decline with age (something we will show later in this term), it seems problematic to just compare the average sexual frequency across these groups. This is because age **confounds** the relationship between marital status and sexual frequency. Statistical models will gives us tools to account for this problem and to get a better estimate of the relationship between marital status and sexual frequency, net of this confounder.

Second, statistical models will allow us to account for how the relationship between two variables might differ depending on the context of a third variable. This is what we call an **interaction**. For example, lets say we were interested in the relationship between the number of sports played and a student's popularity (measured by friend nominations) in our Add Health data. Because of gender norms, we might expect that this relationship is different for boys and girls. We can use statistical models to empirically examine whether this is true. This kind of contextualization is an important component of sociological practice.

## Observational Data, Experimental Thinking

Much of the data that we use in sociology is **observational** rather than **experimental**. In an experimental design, the researcher randomly selects subjects to receive some sort of treatment and then observes how this treatment affects some outcome. Thus, the research engages in systematic manipulation to observe a response. In observational data, the researcher does not directly manipulate the environment but rather just observes and records the social setting as it is.

Experimental data can be more powerful than observational data because the random assignment of a treatment through researcher manipulation strengthens claims of causation. If there is a relationship between treatment and response it can only come through a causal relationship or random chance. In observational data, the relationship between any two variables can be a result of a causal relationship, random chance,spuriousness. Spuriousness occurs when other variables produce the relationship between two other variables rather than them directly causing each other. The example above about marital status and sexual frequency is a simple example. If we note that widows have less sex than other people, we may be tempted to think that something about being widowed reduces someone's sexual drive or their interactions with others. However, the more obvious explanation is that widows tend to be quite a bit older than other marital status groups and older people have less sex. Age is generating a spurious relationship between widowhood and sexual frequency. This kind of spuriousness is the reason for the frequent claim that "correlation is not causation."

There are two different philosophical approaches to the statistical analysis of observational data where spuriousness can be a problem. The first approach approaches it as **pseudo-experimental** data. The goal of this approach is to try to find ways to mimic the experimental design approach with observational data. At a basic level this can include "controlling" for other variables (which we will learn) and can extend to a variety of techniques of causal modeling that are intending to use some feature of the data to recover causation (which we will not learn).

The second approach treats statistical analysis as a way to describe observed data in a formal, systematic, and replicable way. The goal is to establish to what extent the data are consistent with competing theories that seek to understand the outcome in question, rather than to mimic the experimental approach. Although quantitative and qualitative approaches are often seen as philosophically different approaches, this approach to observational data shares many features with more purely qualitative approaches to data analysis. This is the approach that I take in this course.

# Chapter 2

# The Distribution of a Variable

## What is a Distribution?

We will hear the term **distribution** a lot during this class. When we think about the distribution of a variable, we are referring to how the different values of the variables are distributed across observations. This distribution gives us a sense of how much variation there is and what the most common values are for a given variable. Intuitively, we think about distributions in our personal life any time we think about how we "measure" up relative to everyone else on something (income, number of Facebook friends, GRE scores, etc.). We want to know where we "fall" in the distribution of these variables.

A distribution can be represented graphically. However, the technique we use to graph the distribution will depend on whether we have a categorical or a quantitative variable. For categorical variables, we will use a **barplot**, while for quantitative variables, we will use a **histogram**.

For quantitative variables, we can also calculate summary statistics that will give us information about the distribution, such as its center and spread. We will save those calculations for later modules. For this module, we will focus on graphical techniques.

### Looking at the distribution of a categorical variable

#### Calculating the frequency

In order to graph the distribution of a categorical variable, we first need to calculate its frequency. The frequency is just the number of observations that fall into a given category of a categorical variable. We could, for example, count up the number of passengers who were in the various passenger classes on the Titanic. Doing so would give us the following:

| Passenger class | Frequency |
|---|---:|
| First | 323 |
| Second | 277 |
| Third | 709 |
| **Total** | **1309** |

There were 323 first class passengers, 277 second class passengers, and 709 third class passengers. Adding those numbers up gives us 1,309 total passengers. *R* will calculate these numbers for us easily using the `table` command:

```
table(titanic$pclass)
```

```
##
##  First Second  Third
##    323    277    709
```

### Frequency, proportion, and percent

The frequency we just calculated is sometimes called the "absolute" frequency because it just provides the raw number of observations. It is typically more useful to represent this frequency in terms of proportions. Proportions go from 0-1 and give us the share of observations that fall into each category. We can calculate the proportion by simply dividing our frequency by the total number of observations:

| Passenger class | Frequency | Proportion |
|---|---:|---:|
| First | 323 | 323/1309=0.247 |
| Second | 277 | 277/1309=0.212 |
| Third | 709 | 709/1309=0.542 |
| **Total** | **1309** | |

*R* provides a nice shorthand function titled `prop.table` to conduct this operation:

```
prop.table(table(titanic$pclass))
```

```
##
##     First    Second     Third
## 0.2467532 0.2116119 0.5416348
```

Note that I had to "wrap" the `prop.table` command around the `table` command here because the `prop.table` command expects a vector of numbers that it will turn into proportions.

We often convert proportions to percents which most people are more familiar. To convert a proportion to a percent, just multiply by 100:

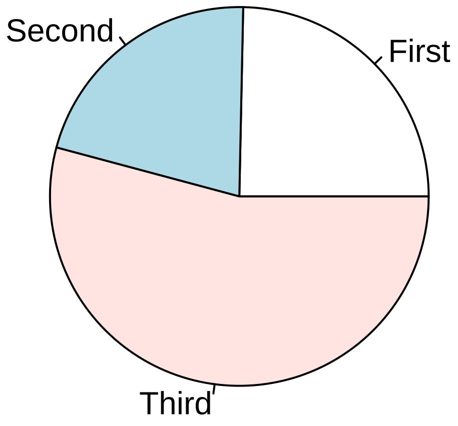
| Passenger class | Frequency | Proportion | Percent |
|---|---:|---:|---:|
| First | 323 | 323/1309=0.247 | 0.247*100=24.7% |
| Second | 277 | 277/1309=0.212 | 0.212*100=21.2% |
| Third | 709 | 709/1309=0.542 | 0.542*100=54.2% |
| **Total** | **1309** | | |

24.7% of passengers were first class, 21.2% of passengers were second class, and 54.2% of passengers were third class. So, just over half of passengers were third class and the remaining passengers were fairly evenly split between first and second class.

### Barplots and piecharts

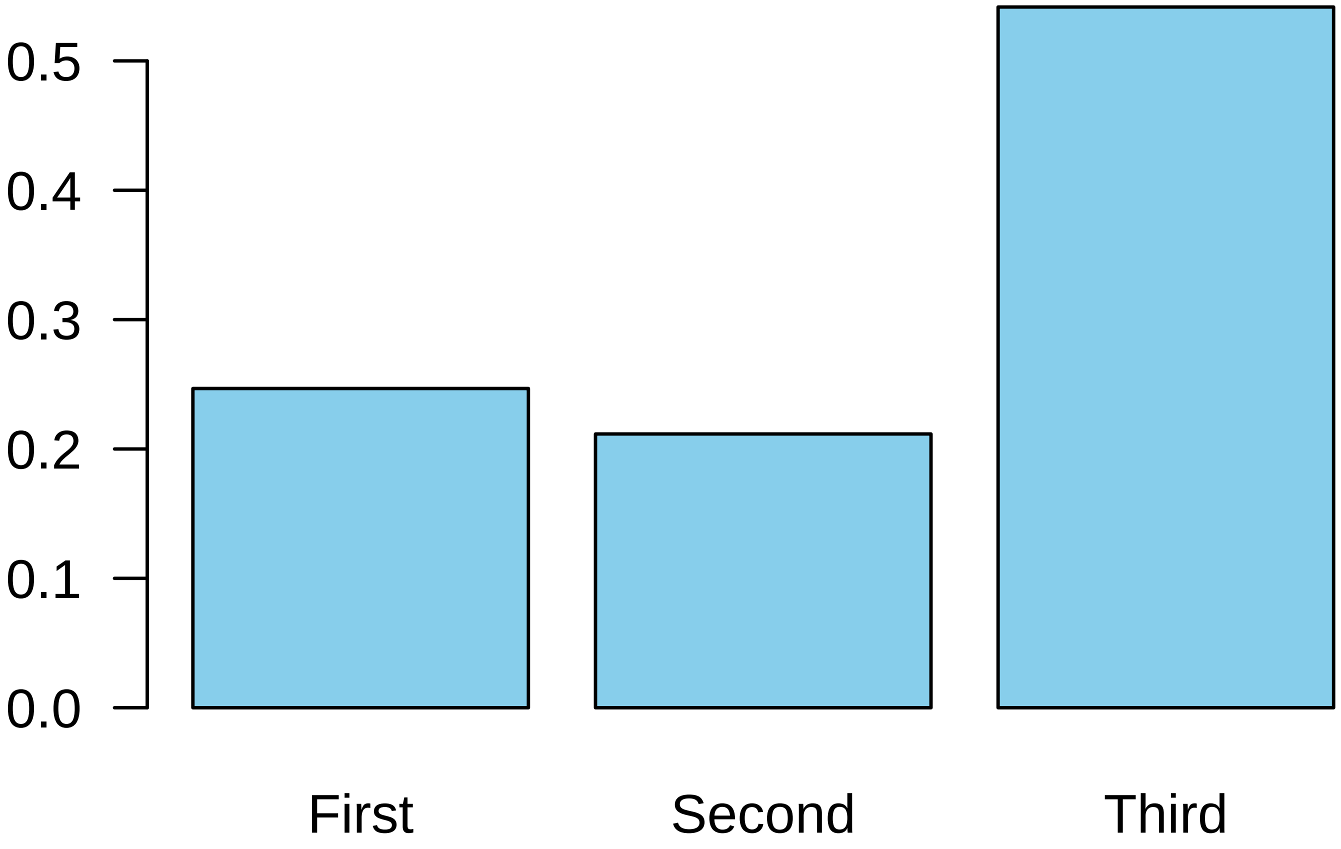
Now that we have proportions/percents, we can use these values to construct a graphical display of the distribution. The **piechart** is a very common technique here. I am sure you have all seen examples of piecharts. In a piechart, you show each category's percentage as a slice of a pie. Here is the R code and output of a piechart for the passenger class variable:

```r
p <- prop.table(table(titanic$pclass))
pie(p)
```



Although piecharts are popular, they are actually not a very good tool. In order to judge the relative size of the slices, your eye has to make judgments in two dimensions and it can often be difficult to decide which slice is bigger. In this case, for example, the relative size of first and second class are quite close. A better way to display the distribution is by using a **barplot** in which vertical (or horizontal) bars give the proportion or percent:

```r
barplot(p, col="skyblue", las=1)
```



Now your eye only has to work in one dimension to identify the relative size of each share. In this case, we can clearly see that the first class bar is higher than the second class bar. Barplots are almost always better at displaying distributions than piecharts.

## Looking at the distribution of a quantitative variable

Barplots won't work for quantitative variables because quantitative variables don't have categories. However, we can do something quite similar with the **histogram**.

One way to think about the histogram is that we are imposing a set of categories on a quantitative variable by breaking our quantitative variable into a set of equally wide intervals that we call bins. The most important decision with a quantitative variable is how wide to make these bins. As an example, lets take the age variable from our politics dataset. I could break this variable into five-year intervals that go from 0-5, 5-10, 10-15, 15-20, and so on. Alternatively, I could use 10-year intervals from 0-10, 10-20, 20-30, and so on. I could also use any other interval I like such as 1-year, 3-year, and so on. Lets use 10-year intervals for this example. I then just have to count up the number of observations that fall into each 10 year interval.

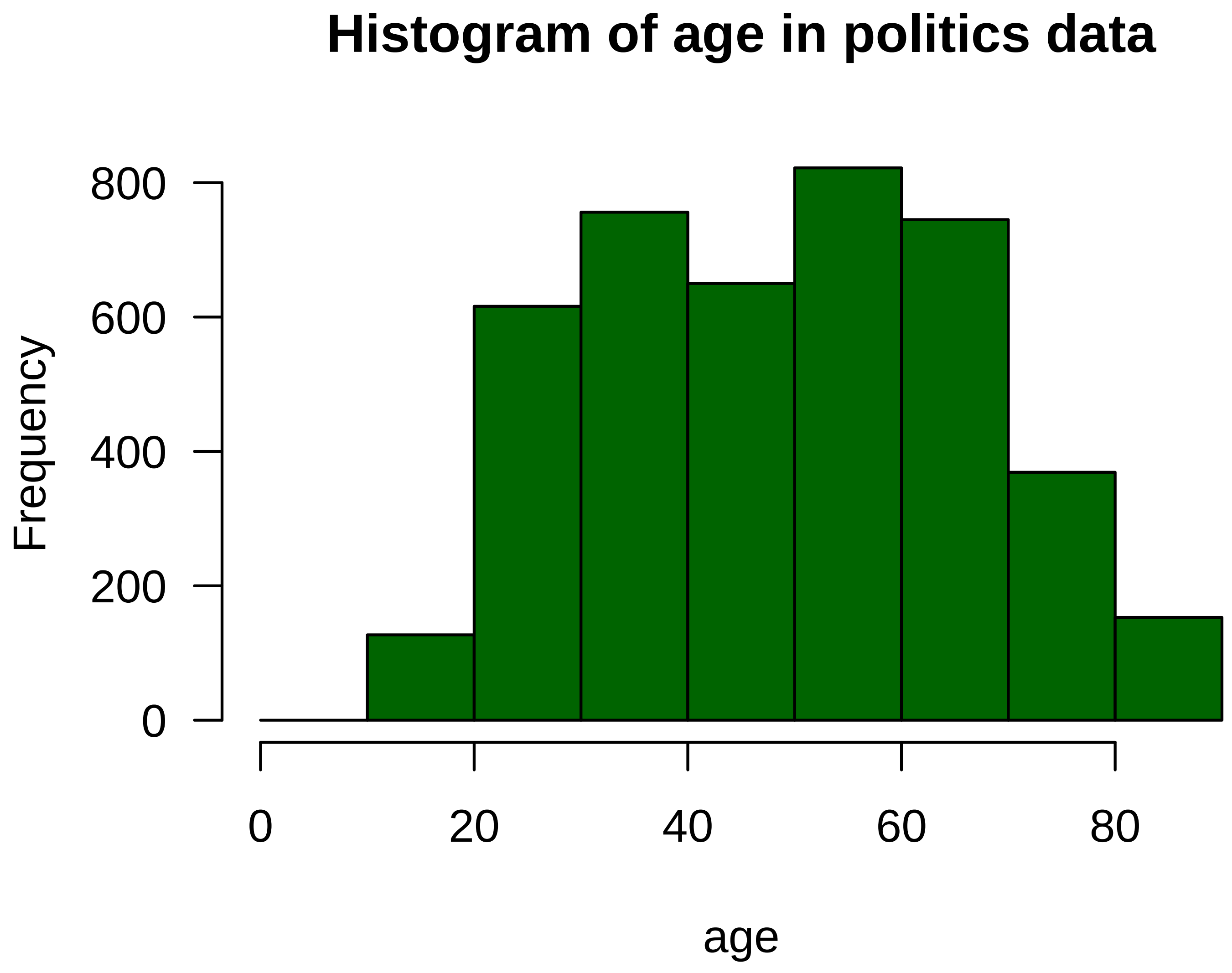
| Age | Frequency |
|---|---:|
| 0-10 | 0 |
| 10-20 | 127 |
| 20-30 | 616 |
| 30-40 | 756 |
| 40-50 | 650 |
| 50-60 | 822 |
| 60-70 | 745 |
| 70-80 | 369 |
| 80-90 | 153 |

Because the survey was only administered to adults, I have zero individuals from 0-10 and only a few in the 10-20 age range. Note that I have to make a decision on how to handle "edge" cases that are on the border between two bins. For example, does someone who is age 20 go in the 10-20 bin or the 20-30 bin? In this case, I have assigned all of these cases to the higher bin (e.g. 20-30 in the case of 20-year olds) because this is the default behavior in *R*.

Now that we have the frequencies for each bin, we can plot the histogram. The histogram looks much like a barplot except for two important differences. First, on the x-axis we have a numeric scale for the quantitative variable rather than categories. Second, we don't put any space between our bars.

Here is some *R* code and output for creating a histogram. First, we define our intervals using the `seq` command to define our bins and then we use the `hist` command to make the plot. Notice, I am adding a few other options to the `hist` command in order to produce some nice labeling and colors.

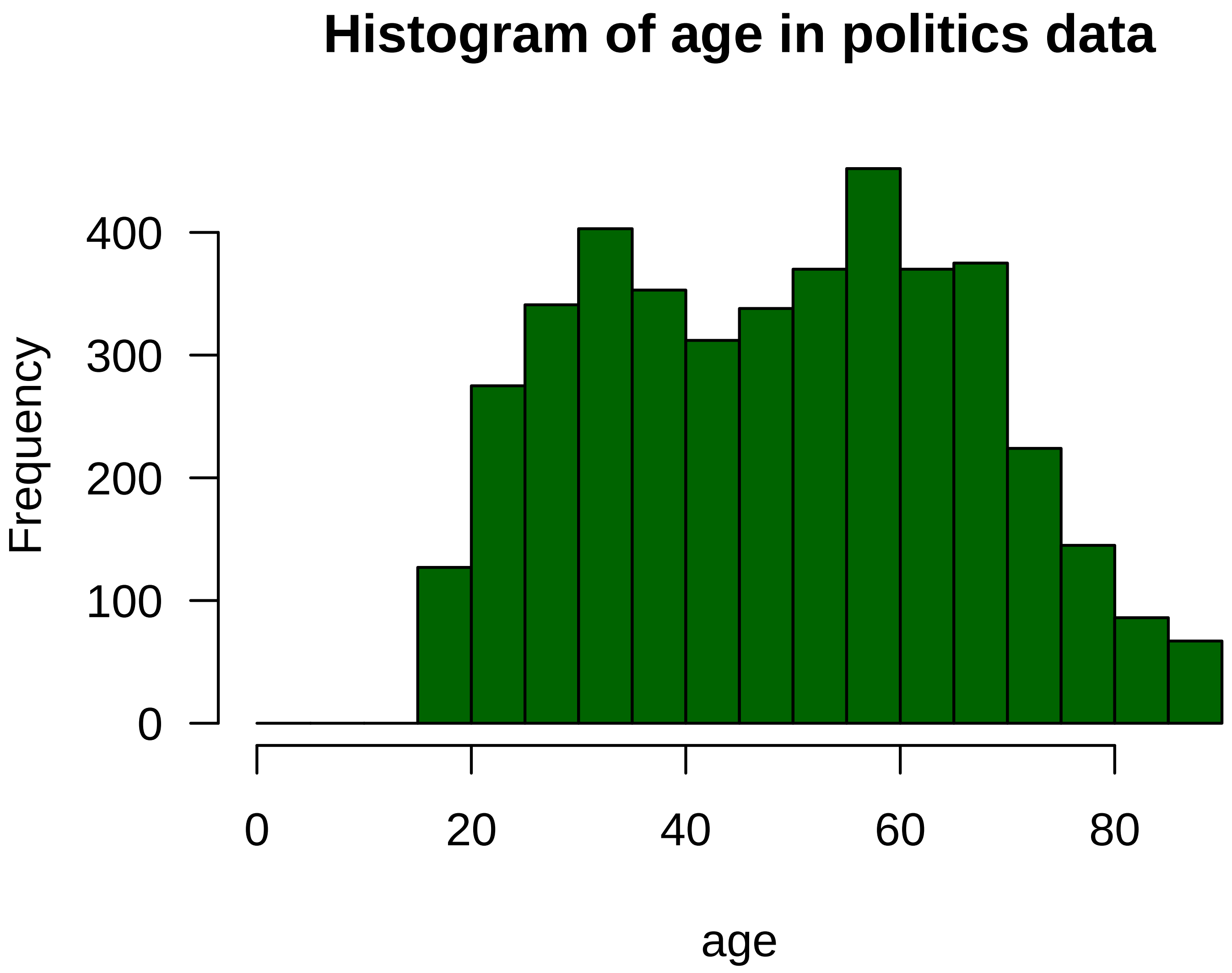
```r
bins <- seq(from=0, to=90, by=10)
hist(politics$age, breaks=bins, col="darkgreen", xlab="age",
     main="Histogram of age in politics data", las=1)
```



I can see the peak in the distribution in the 50's (the baby boomers) with a long fat tail to the right and steeper drop off (due to smaller cohorts and higher mortality) on the left.

What would this histogram look like if I had used 5-year bin widths instead of 10-year bins? Lets try it:

```r
bins <- seq(from=0, to=90, by=5)
hist(politics$age, breaks=bins, col="darkgreen", xlab="age",
     main="Histogram of age in politics data", las=1)
```

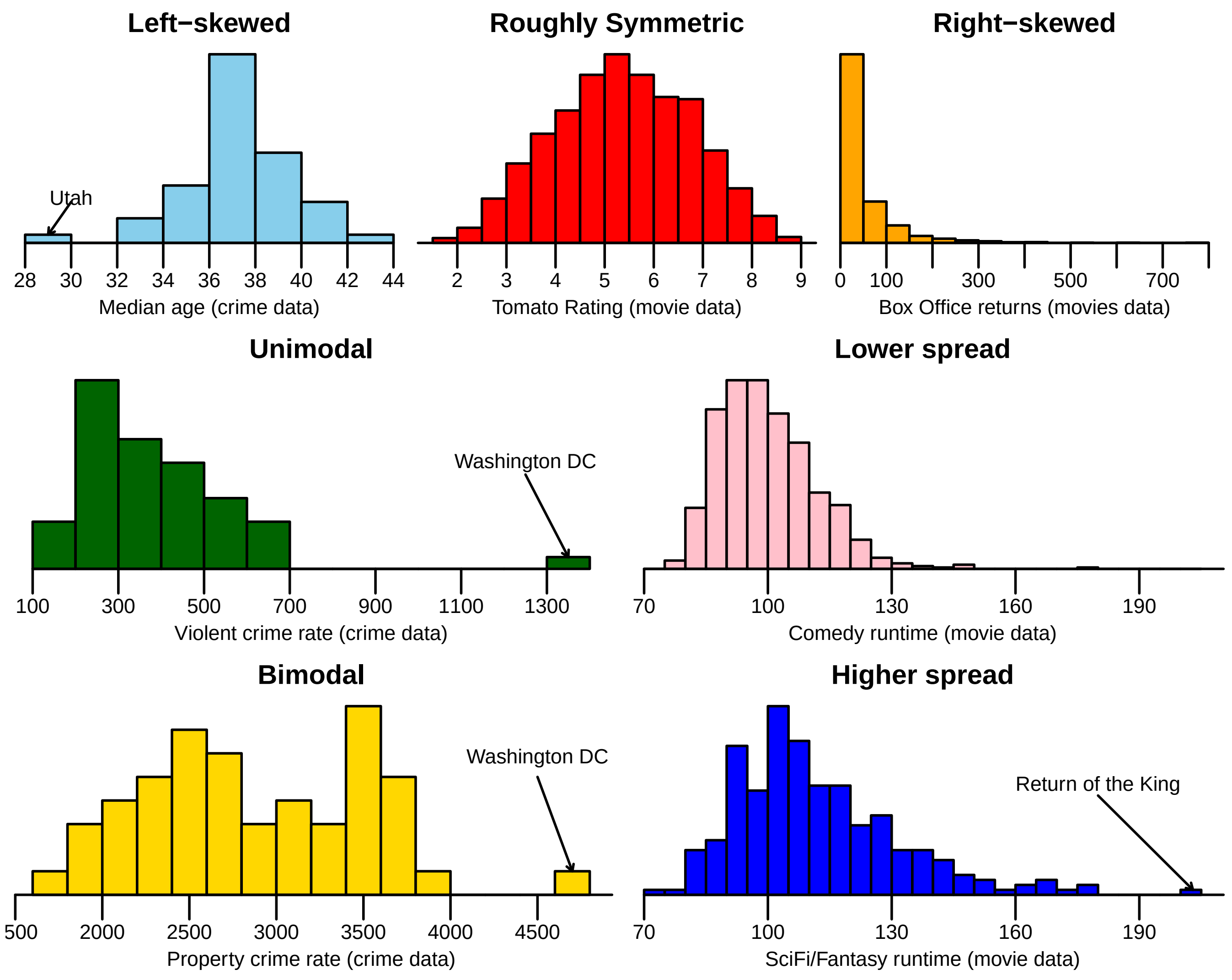


I get more or less the same overall impression but a more fine-grained view. Sometimes adjusting bin width can reveal or hide important trends and sometimes it can just make it more difficult to read. As an exercise, try adjusting the bin width in this example histogram for movie runtime from our movie dataset.

There are four general things to be on the lookout for when examining a histogram.

1. **Shape**. Is the shape symmetric or is one of the tails longer than the other? When the long tail is on the right, we refer to this distribution as **right skewed**. When the long tail is on the left, we refer to the distribution as **left skewed**.
2. **Center**. Where is the center of the distribution? This center is often the same as the peak in the data. Does the histogram have a single peak or multiple peaks? When a distribution has one peak, we call it **unimodal**. When it has two peaks, we call it **bimodal** and so on.
3. **Spread**. How spread out are the values around the center? Do they cluster tightly around the center or are they widely spread out? The answer to this question is generally a relative one, which is to say you can only say how spread out a variable is in relation to the spread of some other variable.
4. **Outliers**. Are there extreme values which fall outside the range of the rest of the data? We want to pay attention to these values because they may have a strong influence on the statistics we calculate to summarize a distribution. They might also help identify data input errors.

The figure below shows some examples of each of these concepts from the various datasets we are using in class.



## Measuring the Center of a Distribution

When we look at a distribution, we often can get an intuitive sense of where its "center" is. But what do we really mean by the term "center?" The notion of center often allows us to think about the value we expect a typical or "average" observation to have, but there are multiple ways of defining this center. In statistics, three different measures of center are used: the **mean**, **median**, and **mode**.

### The mean

The mean is the measure most frequently referred to as the "average" although that term could apply to the median and mode as well. The mean is the **balancing point** of a distribution. Imagine trying to balance a distribution on your finger like a basketball. Where along the distribution would you place your finger to achieve this balance? This point is the mean. It is equivalent to the concept of "center of mass" from physics.

The calculation of the mean is straightforward. First, sum up all the values of your variable across all observations. Then, divide by the number of observations. As an example, lets take a sub-sample of our movie data. I am going to select all the romances (not including rom-coms which are coded as comedies) produced in 2013. There were nine "pure" romances in 2013. I want to know their mean Tomato Meter rating.

| Title | Tomato Meter |
|---|---:|
| Kill Your Darlings | 77 |
| I'm in Love with a Church Girl | 6 |
| Safe Haven | 12 |
| The Face of Love | 43 |
| Love and Honor | 13 |
| Before Midnight | 98 |
| Drinking Buddies | 83 |
| Ain't Them Bodies Saints | 79 |
| The Great Gatsby | 49 |
| **Sum** | **460** |

In the very last row, I show the sum of the Tomato Meter rating which simply sums up the Tomato Meter rating of each movie. To calculate the mean, I simply divide this sum by the number of movies which is 9.

$$\bar{x} = \frac{460}{9} = 51.11$$

The mean Tomato Meter rating for romantic movies in 2013 was 51.11. Notice the funny $x$ with a bar over it ("x bar"). Mathematically, we often represent variables with lower-case roman letters like $x$ or $y$. Putting the bar above the letter is the way to mathematically signify the mean of that variable.

Since we are discussing math symbols, lets talk about creating a mathematical formula for what we just did. In order to do that, I need to introduce a variety of mathematical symbols, but don't get frightened. We are just formalizing the method of calculating the mean that I just demonstrated above.

First, as I said we represent a given variable by a lower-case roman letter, such as $x$. If we want to specify a particular observation of $x$, we use a subscript number. So $x_1$ is the value of the first observation of $x$ in our data and $x_{25}$ is the value of the 25th observation of $x$ in our data. We use the letter n to signify the number of observations so $x_n$ is always the last observation of $x$ in our data.

Now we need some way to indicate "sum up all the values of $x$." This is given by the summation sign which looks as follows:

$$\sum_{i=1}^{n} x_i$$

In English, this just means "sum up all the values of $x$, starting at $x_1$ and ending at $x_n$." That gives us our sum, which we just need to divide by the number of observations, $n$.

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

In *R*, the mean is very straightforward to calculate. Lets calculate the mean of the Tomato Meter rating for all movies in our dataset:

```r
sum(movies$TomatoMeter)/nrow(movies)
```

```
## [1] 47.77595
```

The `sum` command calculates the sum and the `nrow` command calculates the number of rows of our dataset, which is equivalent to the number of observations. Or we could go even simpler and just use the `mean` command:

```r
mean(movies$TomatoMeter)
```

```
## [1] 47.77595
```

We can see that the mean Tomato Meter of romantic movies in 2013 (51.1) were slightly above the mean for all the movies in our dataset (47.8), but not by much.

## The median

The median is almost as widely used as the mean as a measure of center, for reasons I will discuss below. The median is the **midpoint** of the distribution. It is the point at which 50% of observations have lower values and 50% of the observations have higher values.

In order to calculate the median, we need to first order our observations from lowest to highest. Lets do that with the romantic movie data above.

| Title | Tomato Meter | Order |
|---|---:|---:|
| I'm in Love with a Church Girl | 6 | 1 |
| Safe Haven | 12 | 2 |
| Love and Honor | 13 | 3 |
| The Face of Love | 43 | 4 |
| **The Great Gatsby** | **49** | **5** |
| Kill Your Darlings | 77 | 6 |
| Ain't Them Bodies Saints | 79 | 7 |
| Drinking Buddies | 83 | 8 |
| Before Midnight | 98 | 9 |

To find the median, we have to find the observation right in the middle. When we have an odd number of observations, finding the exact middle is easy. In this case, it is given by the 5th observation because it has four observations lower than it and four observations higher than it. So the median Tomato Meter rating for romantic movies in 2013 was 49.

When you have an even number of observations, then finding the median is slightly more tricky because you have no single observation that is exactly in the middle. In this case, you find the two observations that are closest to the middle and calculate their mean (sum them up and divide by two). For example, if we had ten observations, we would take the mean of the 5th and 6th observations to calculate the median.

The `median` command in *R* will calculate the median for you.

```r
median(movies$TomatoMeter)
```

```
## [1] 47
```

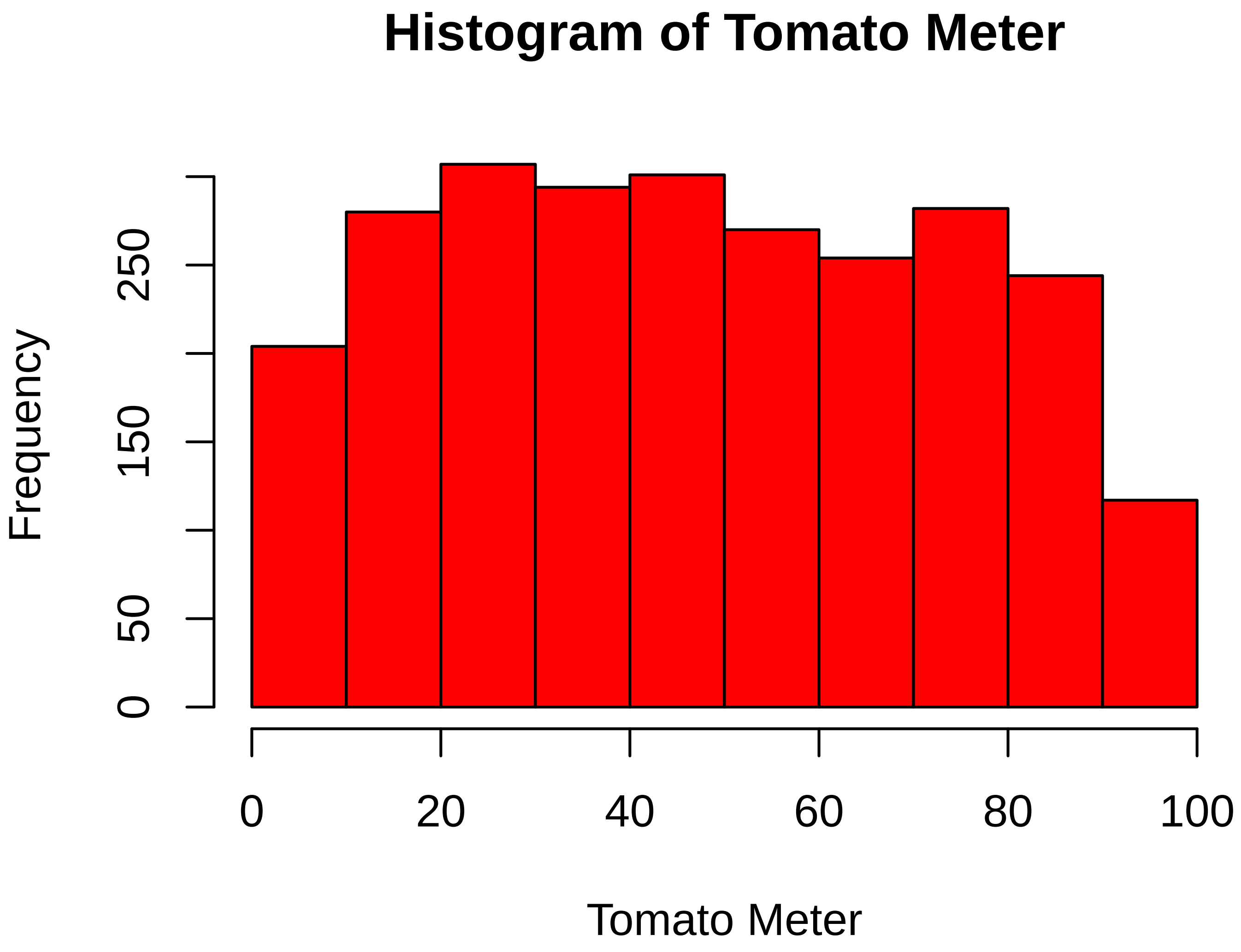
In this particular case, the mean (47.8) and median (47) of the Tomato Meter produced very similar measures of center, but this isn't always the case as I will demonstrate below.

## The mode

The mode is the least used of the three measures of center. In fact, it is so infrequently used that R does not even have a built-in function to calculate it. The mode is the **high point** or **peak** of the distribution. When you look at a distribution graphically, the mode is what your eye is drawn to as a measure of center.

Calculating the mode however is much trickier. Simply speaking, the mode is the most common value in the data. However, when data are recorded down to very precise decimal levels, this can be a misleading number. Furthermore, there may be multiple "peaks" in the data and so speaking of a single mode can be misleading.

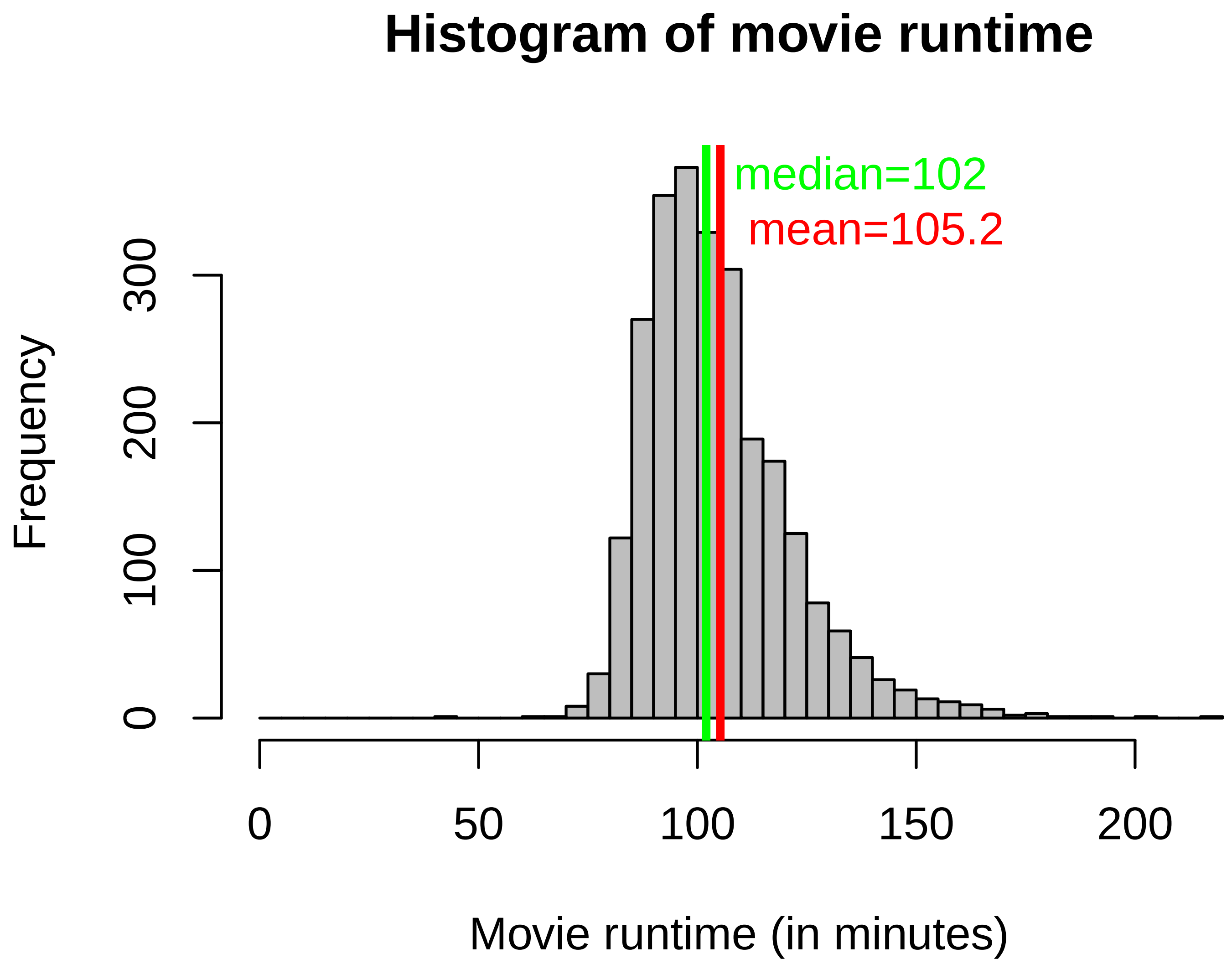
In this case, we have a tie for the most common value. 41 movies had Tomato Meter ratings of 29, 33, and 51, respectively. If we look at the histogram, we can see that the idea of a "peak" in this data is a little misleading anyway because the distribution looks pretty "flat" across most of the values.



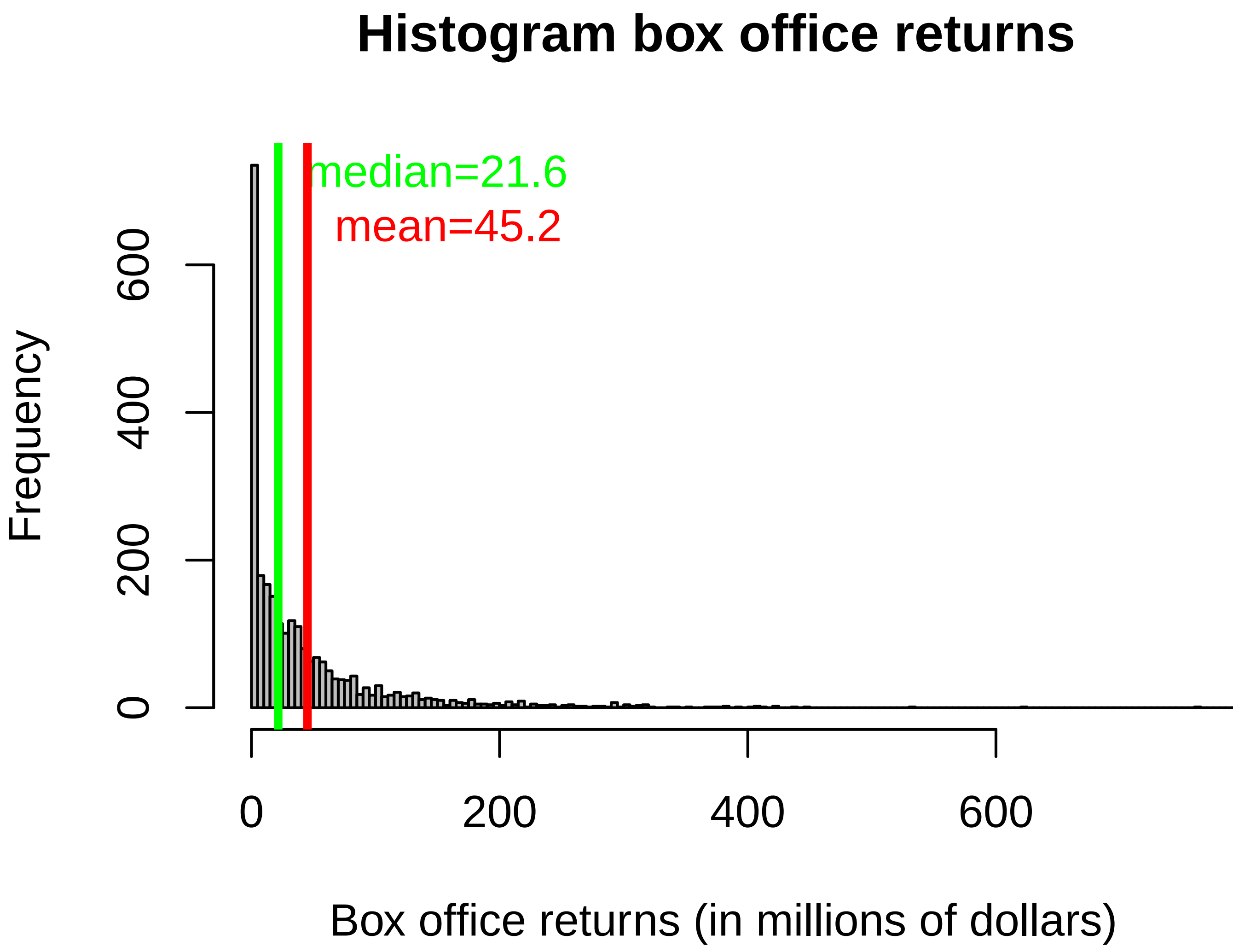
### Comparing the mean and median

Why would the mean and median give different results? Remember that the mean defines the **balancing point** and the median defines the **midpoint**. If you have a perfectly symmetric distribution, then these two points are the same because you would balance the distribution at the midpoint. However, when the distribution is skewed in one direction or another, the mean and the median will be different. In order to maintain balance, the mean will be pulled in the direction of the skew. When you have heavily skewed distributions, this can lead to dramatically different values for the mean and median.

Lets look at this phenomenon for a couple of cases in our movie dataset. As the histogram above showed, the Tomato Meter variable is fairly symmetric and as a result we end up with a mean (47.8) and median (47) that are pretty close. If we look at movie runtime, we see a distribution that is somewhat more right-skewed.



The skew here is not too dramatic, but it pulls the mean about 3 minutes higher than the median. If we look at the distribution of box office returns to movies, however we see a very heavy right skew. Most movies make moderate amounts of money, and then there are a few star performers that make bucket loads of cash.



As a result of this skew, the mean box office returns are about $45.2 million, while the median box office returns are about $21.6 million. The mean here is more than double the median!

Note that neither estimate is in some fundamental way incorrect. They are both correctly estimating what they were intended to estimate. It is up to us to understand and interpret these numbers correctly and to understand their limitations.

In many cases, we are actually more interested in the median as a measure of "average" experience than the mean, even though we think of the mean as the "average." This is, for example, why you see home prices in an area always reported in terms of medians rather than means. Mean home prices tend to be much higher than median home prices because of the relatively few fabulously expensive homes in a given area.

---

## Percentiles and the Five Number Summary

In this section, we will learn about the concept of **percentiles**. Percentiles will allow us to calculate a **five number summary** of a distribution and introduce a new kind of graph for describing a distribution called the **boxplot**.

## Percentiles

We have already seen one example of a percentile. The median is the 50th percentile of the distribution. It is the point at which 50% of the observations are lower and 50% are higher. We can actually use this same logic to calculate other percentiles. We could calculate the 25th percentile of the distribution by finding the point where 25% of the observations are below and 75% are above. We could even calculate something like the 43rd percentile if we were so inclined.

We calculate percentiles in a fashion similar to the median. First, sort the data from lowest to highest. Then, find the exact observation where X% of the observations fall below to find the Xth percentile. In some cases, there might not be an exact observation that fits this description and so you may have to take the mean across the two closest numbers.

The `quantile` command in R will calculate percentiles for us in this fashion (quantile is a synonym for percentile). In addition to telling the quantile command which variable we want the percentiles of, we need to tell it which percentiles we want. In the command below, I ask for the 27th and 57th percentile of age in our sexual frequency data.

```r
quantile(sex$age, p=c(.27,.57))
```

```
## 27% 57%
##  32  46
```

27% of the sample were younger than 32 years of age and 57% of the sample were younger than 46 years of age.

## The five number summary

We can split our distribution into quarters by calculating the minimum(0th percentile), the 25th percentile, the 50th percentile (the median), the 75th percentile, and the maximum (100th percentile). Collectively, these percentiles are known at the **quartiles** of the distribution (not to be confused with quantile) and are also described as the **five number summary** of the distribution.

We can calculate these quartiles with the `quantile` command. If I don't enter in specific percentiles, the `quantile` command will give me the quartiles by default:

```r
quantile(sex$age)
```

```
##   0%  25%  50%  75% 100%
##   18   31   43   56   89
```

The bottom 25% of respondents are between the ages of 18-31. The next 25% are between the ages of 31-43. The next 25% are between the ages of 43-56. The top 25% are between the ages of 56-89.

We can also use this five number summary to calculate the **interquartile range** (IQR) which is just the difference between the 25th and 75th percentile. This gives us a sense of how spread out observations are. In this data:

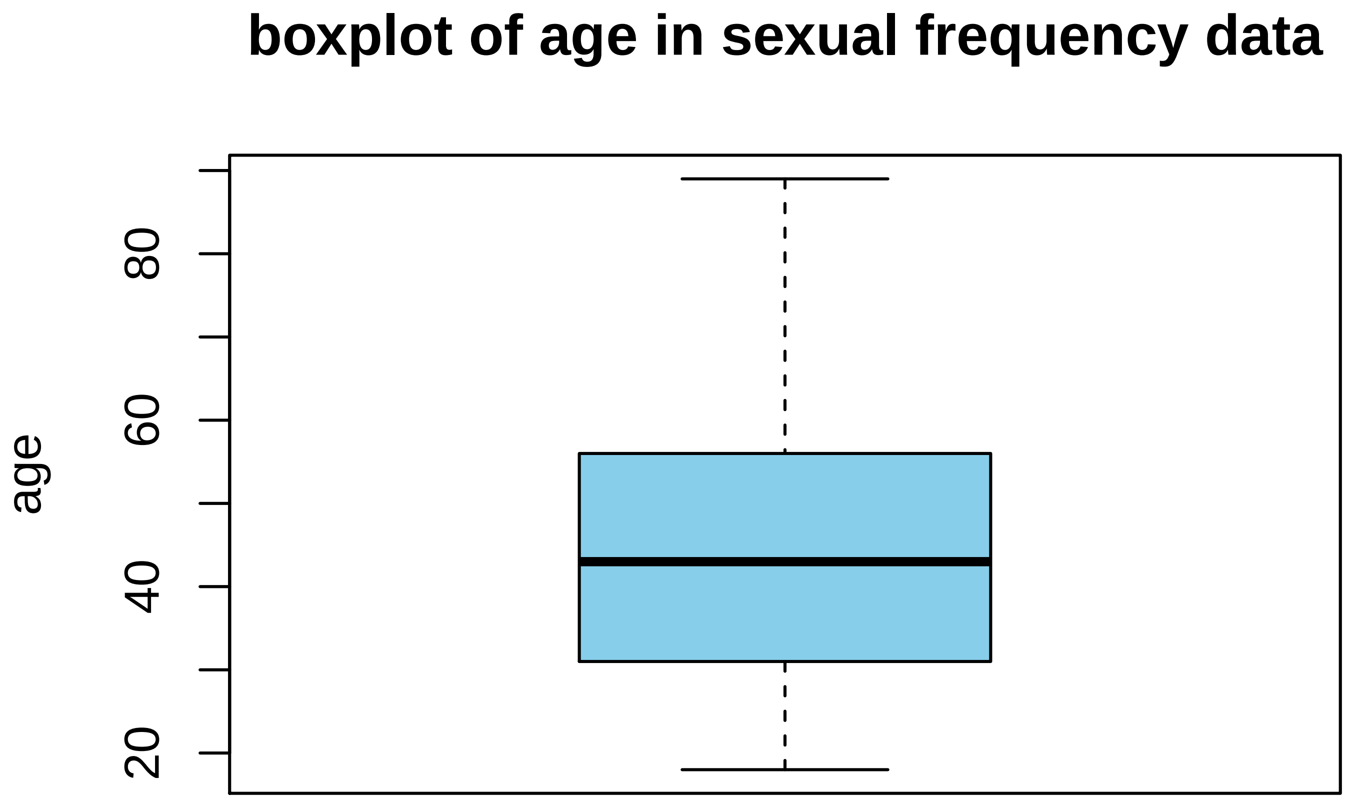
$$IQR = 56 - 31 = 25$$

So, the 25th and 75th percentile of age are separated by 25 years.

## Boxplots

We can also use this five number summary to create another graphical representation of the distribution called the boxplot. Here is an example of a boxplot for the age variable from the sexual frequency data:

```r
boxplot(sex$age,
        col="skyblue", ylab="age",
        main="boxplot of age in sexual frequency data")
```



The "box" in the boxplot is drawn from the 25th to the 75th percentile. The height of this box is equal to the interquartile range. The median is drawn as a thick bar within the box. Finally, "whiskers" are then drawn to the minimum and maximum of the data. Sometimes, the whiskers are drawn to less than the minimum and maximum if these values are very extreme and instead the whiskers are drawn out to 1.5xIQR in length and then individual points are plotted. In this case there were no extreme values, so the whiskers were drawn all the way out to the actual maximum and minimum.

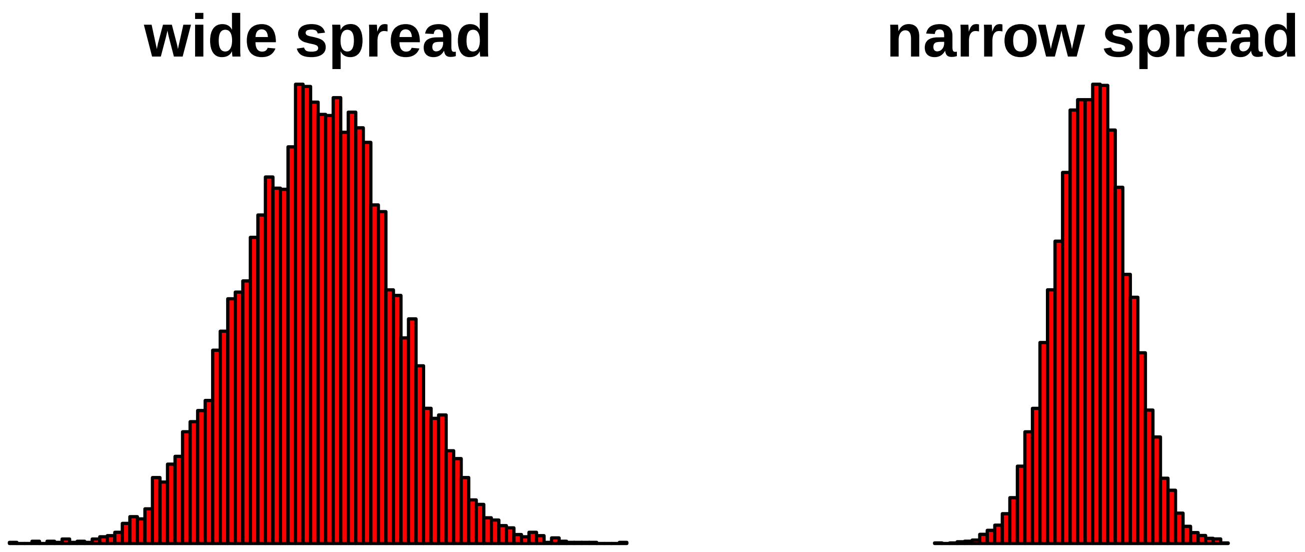
The boxplot is a very rich graph which provides many pieces of information. It shows the center of the distribution as measured by the median. It also gives a sense of the spread of the distribution and extreme values by the height of the box and whiskers. It can also show skewness in the distribution depending on where the median is drawn within the box and the size of the whiskers. If the median is in the center of the box, then that indicates a symmetric distribution. If the median is towards the bottom of the box, then the distribution is right-skewed. If the median is towards the top of the box, then the distribution is left-skewed.

As an exercise, use the slider to see different percentiles on both a histogram and a boxplot.

---

## Measuring the Spread of a Distribution

The second most important measure of a distribution is its spread. Spread indicates how far individual values tend to fall from the center of the distribution. As the figure below shows, two distributions can have the same center and general shape (in this case, a bell curve) but have very different spreads.



## Range and interquartile range

One of the simplest measures of spread is to calculate the **range**. The range is the distance between the highest and lowest value. Lets take a look at the range in the fare paid (in British pounds) for tickets on the Titanic. The `summary` command, will give us the information we need:

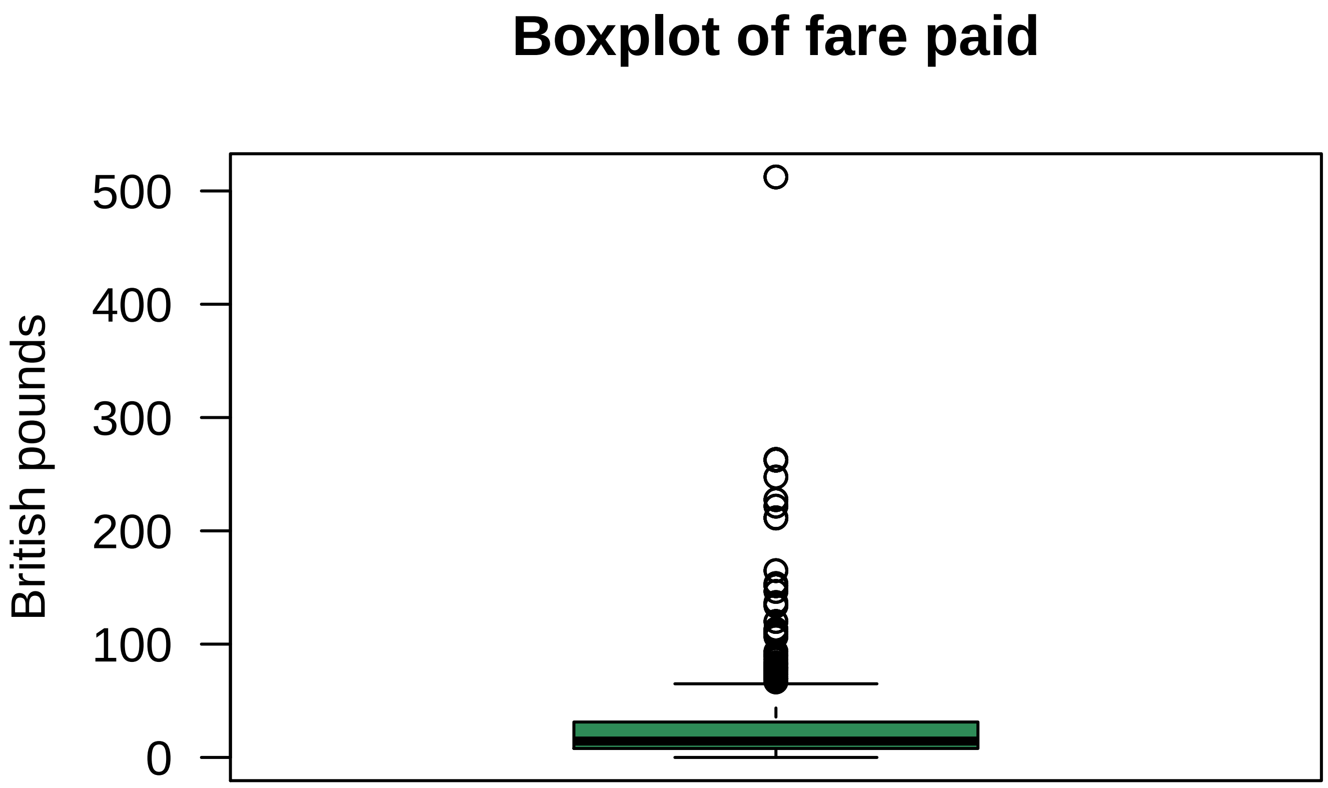
```
summary(titanic$fare)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   7.896  14.454  33.276  31.275 512.329
```

Note that at least one person made it on the Titanic for free. The highest fare paid was 512.3 pounds. So the range is easy to calculate 512.3 - 0 = 512.3. The difference between the highest and lowest paying passenger was about 512 pounds.

This example also reveals the shortcoming of the range as a measure of spread. If there are any outliers in the data, they are going to show up in the range and so the range may give you a misleading idea of how spread out the values are. Note that the 75th percentile here is only 31.28 pounds, which would suggest that the 512.3 maximum is a pretty high outlier. We can check this by graphing a boxplot:

```
boxplot(titanic$fare, col="seagreen",
        las=1, ylab="British pounds",
        main="Boxplot of fare paid")
```



The maximum value is such an outlier that the rest of the boxplot has to be "scrunched" in order to fit it all into the graph. Clearly this is not a good indicator of spread. However, we have already seen a better measure of spread using a similar idea: the **interquartile range** or **IQR**. The IQR is just the range between the 25th and 75th percentile. We already have these numbers from the output above, so the IQR = 31.28-7.90=23.38. So, the difference in fare between the 25th and 75th percentile (the middle 50% of the data) was 23.4 pounds. That result suggests a much smaller spread of fares.

You can also use the `IQR` command in *R* to directly calculate the IQR.

```
IQR(titanic$fare)
```

```
## [1] 23.3792
```

### Variance and standard deviation

The most common measure of spread is the **variance** and its derivative measurement, the **standard deviation**. Its so common in fact, that most people simply refer to the concept of "spread" as "variance." The variance can be defined as the "average squared distance to the mean." Of course, "squared distance" is a bit hard to think about, so we more commonly take the square root of the variance to get the standard deviation which gives us the "average distance to the mean." Imagine if you were to randomly pick one observation from your dataset and guess how far it would be from the mean. Your best guess would be the standard deviation.

The calculation for the variance and standard deviation is a bit intimidating but we will break it down into steps to show it is not that hard. At the same time, I will show you to calculate the parts in R using the fare variable from the Titanic data.

The overall formula for the variance (which is represented as $s^2$) is:

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$$

That looks tough, but lets break it down. The first step is this:

$$(x_i - \bar{x})$$

You take each value of your variable $x$ and subtract the mean from it. This can be done in *R* easily:

```
diffx <- titanic$fare-mean(titanic$fare)
```

This measure gives us a description of how far each observation is from the mean which is already kind of a measure of spread, but we can't do much with it yet because some differences are positive (higher than the mean) and some are negative (lower than the mean). In fact, if we take the mean of these differences, it will be zero by definition because this is what it means for the mean to be the balancing point of the distribution.

```
round(mean(diffx),5)
```

```
## [1] 0
```

The next step is:

$$(x_i - \bar{x})^2$$

We just need to square the differences. This will get rid of our negative/positive problem, because the squared values will all be positive.

```
diffx.squared <- diffx^2
```

The next step is:

$$\sum_{i=1}^{n}(x_i - \bar{x})^2$$

We need to sum up all of our values. This value is sometimes called the **sum of squared X** or **SSX** for short. It is already pretty close to a measure of variance already.

```
ssx <- sum(diffx.squared)
```

The more distance there is from the mean on average, the larger this value will be. However, it also gets larger when we have more values because we are just taking a sum. To get a number that is comparable across different number of observations, we need to do the final step:

$$s^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$$

We are going to divide our SSX value by the number of observations minus one. The "minus one" thing is a bit tricky and I don't want to get into the details of why we do it here. When n is large, this will have little effect and basically you are taking an average of the squared distance from the mean.

```
variance <- ssx/(length(diffx)-1)
variance
```

```
## [1] 2677.398
```

So the average squared distance from the mean fare is 2677.39 pounds squared. Of course, this isn't a very interpretable number, so its probably better to square root it and get the standard deviation:

```
sqrt(variance)
```

```
## [1] 51.74358
```

So, the average distance from the mean fare is 51.74 pounds.

Note that I could have used the power of *R* to do this entire calculation in one line:

```
sqrt(sum((titanic$fare-mean(titanic$fare))^2)/(length(titanic$fare)-1))
```

```
## [1] 51.74358
```

Alternatively, I could have just used the `sd` command to have *R* do all the heavy lifting:

```
sd(titanic$fare)
```

```
## [1] 51.74358
```

# Chapter 3

# Measuring Association

---

## The Two-Way Table

The **two-way table** (also known as a **cross-tabulation** or **crosstab**) gives the joint distribution of two categorical variables. Lets use our politics dataset to construct a two-way table of belief in anthropogenic climate change by political party:

```
tab <- table(politics$party, politics$globalwarm)
tab
```

```
##
##                 No  Yes
##   Democrat     230 1235
##   Republican   577  664
##   Independent  326 1055
##   Other         47  104
```

The two-way table gives us the **joint distribution** of the two variables, which is the number of respondents who fell into both categories. For example, we can see that 234 democrats did not believe in anthropogenic climate change while 1230 did.

From this table, we can also calculate the **marginal distribution** of each of the variables, which are just the distributions of each of the variables separately. We can do that by adding up across the rows and down the columns:

Table 3.1: Table continues below

| | Deniers | Believers |
|---|---|---|
| **Democrat** | 230 | 1235 |
| **Republican** | 577 | 664 |
| **Independent** | 326 | 1055 |
| **Other** | 47 | 104 |
| **Total** | *230+577+326+47=1180* | *1235+664+1055+104=3058* |

| | Total |
|---|---|
| **Democrat** | *230+1235=1465* |

| | Total |
|---|---|
| **Republican** | *577+664=1241* |
| **Independent** | *326+1055=1381* |
| **Other** | *47+104=151* |
| **Total** | *4238* |

The marginal distribution of party affiliation is given by the Total column on the right and the marginal distribution of climate change belief is given by the Total row at the bottom. Looking at the column marginal, we can see that there were a total of 1464 Democrats, 1239 Republicans, and so on. Looking at the row marginal, we can see that there were 1181 anthropogenic climate change deniers and 3057 anthropogenic climate change believers. The final number (4238) in the far right corner is the total number of respondents altogether. You can get this number by summing up the column marginals (1181+3057) or row marginals (1464+1239+1383+152).

The `margin.table` command in *R* will also calculate marginals for us. I can use the `margin.table` command on the table I created and saved above as *tab* to calculate the same marginals as above. Note that you need to indicate which marginal you want by a number, where 1=row and 2=column, as the second option to `margin.table`:

```
margin.table(tab,1)
```

```
##
##    Democrat  Republican Independent       Other
##        1465        1241        1381         151
```

```
margin.table(tab,2)
```

```
##
##   No  Yes
## 1180 3058
```

The two-way table provides us with evidence about the association between two categorical variables. To understand what the association looks like, we will learn how to calculate **conditional distributions**.

## Conditional distributions

To this point, we have learned about the **joint** and **marginal** distributions in a two-way table. In order to look at the relationship between two categorical variables, we need to understand a third kind of distribution: the **conditional distribution**. The conditional distribution is the distribution of one variable conditional on being in a certain category of the other variable. In a two-way table, there are always two ways to calculate a conditional distribution. In our case, we could look at the distribution of climate change belief conditional on party affiliation, or we could look at the distribution of party affiliation conditional on climate change belief. Both of these distributions really give us the same information about the association, but sometimes one way is more intuitive to understand. In this case, I am going to start with the former case and calculate the distribution of climate change belief conditional on party affiliation.

This conditional distribution is basically given by the rows of our two-way table, which give the number of individuals of a given party who fall into each belief category. For example, the distribution of denial/belief among Democrats is 429 and 1932, while among Republicans, this distribution is 708 and 681. However, these two rows are not directly comparable as they are because Republicans are a much smaller group than Democrats. Thus, even if the shares were very different between the two groups, the absolute numbers for Republicans would probably be smaller for both categories. In order to make these rows comparable, we need the proportion of each party that falls into each belief category. In order to do that, we need to divide our rows through by the marginal distribution of party affiliation, like so:

| | Deniers | Believers | Total |
|---|---|---|---|
| **Democrat** | 230/1465 | 1235/1465 | 1465 |
| **Republican** | 577/1241 | 664/1241 | 1241 |
| **Independent** | 326/1381 | 1055/1381 | 1381 |
| **Other** | 47/151 | 104/151 | 151 |

Note that each row gets divided by its row marginal. If we do the math here, we will come out with the following proportions:

| | Deniers | Believers | Total |
|---|---|---|---|
| **Democrat** | 0.157 | 0.843 | 1 |
| **Republican** | 0.4649 | 0.5351 | 1 |
| **Independent** | 0.2361 | 0.7639 | 1 |
| **Other** | 0.3113 | 0.6887 | 1 |

Note that the proportions should add up to 1 within each row because we are basically calculating the share of each row that belongs to each column category. **To understand these conditional distributions, you need to look at the numbers within each row.** For example, the first row tells us that 16% of Democrats are deniers and 84% of Democrats are believers. The second row tells us that 46% of Republicans are deniers and 54% of Republicans are believers.

We can tell if there is an association between the row and column variable if these conditional distributions are different across rows. In this case, they are clearly very different. About 84% of Democrats are believers while only about half (54%) of Republicans are believers. About 76% of Independents are believers, while about 68% of members of other parties are believers.

We can use the `prop.table` command we saw in the last module to estimate these conditional distributions. In order to do that we feed in the crosstab we calculated with tab and one additional argument that indicates which dimension (row or column ) we want to condition across. Like `margin.table` a value of `1` will condition on rows (rows sum to 1) and a value of `2` will condition on columns (columns sum to 1). If we condition on rows here, we will get the same table as above.

```r
prop.table(tab,1)
```

```
##
##                      No       Yes
##   Democrat    0.1569966 0.8430034
##   Republican  0.4649476 0.5350524
##   Independent 0.2360608 0.7639392
##   Other       0.3112583 0.6887417
```

Its important to remember which way you did the conditional distribution and get the interpretation correct. If you are not sure, just note which way the proportions add up to one - this is the direction you should be looking (i.e. within row or column). In this case, I am looking at the distribution of variables within rows, so the proportions refer to the proportion of respondents from a given political party who hold a given belief. But, I could have done my conditional distribution the other way:

```r
prop.table(tab,2)
```

```
##
##                       No        Yes
##   Democrat    0.19491525 0.40385873
##   Republican  0.48898305 0.21713538
##   Independent 0.27627119 0.34499673
##   Other       0.03983051 0.03400916
```

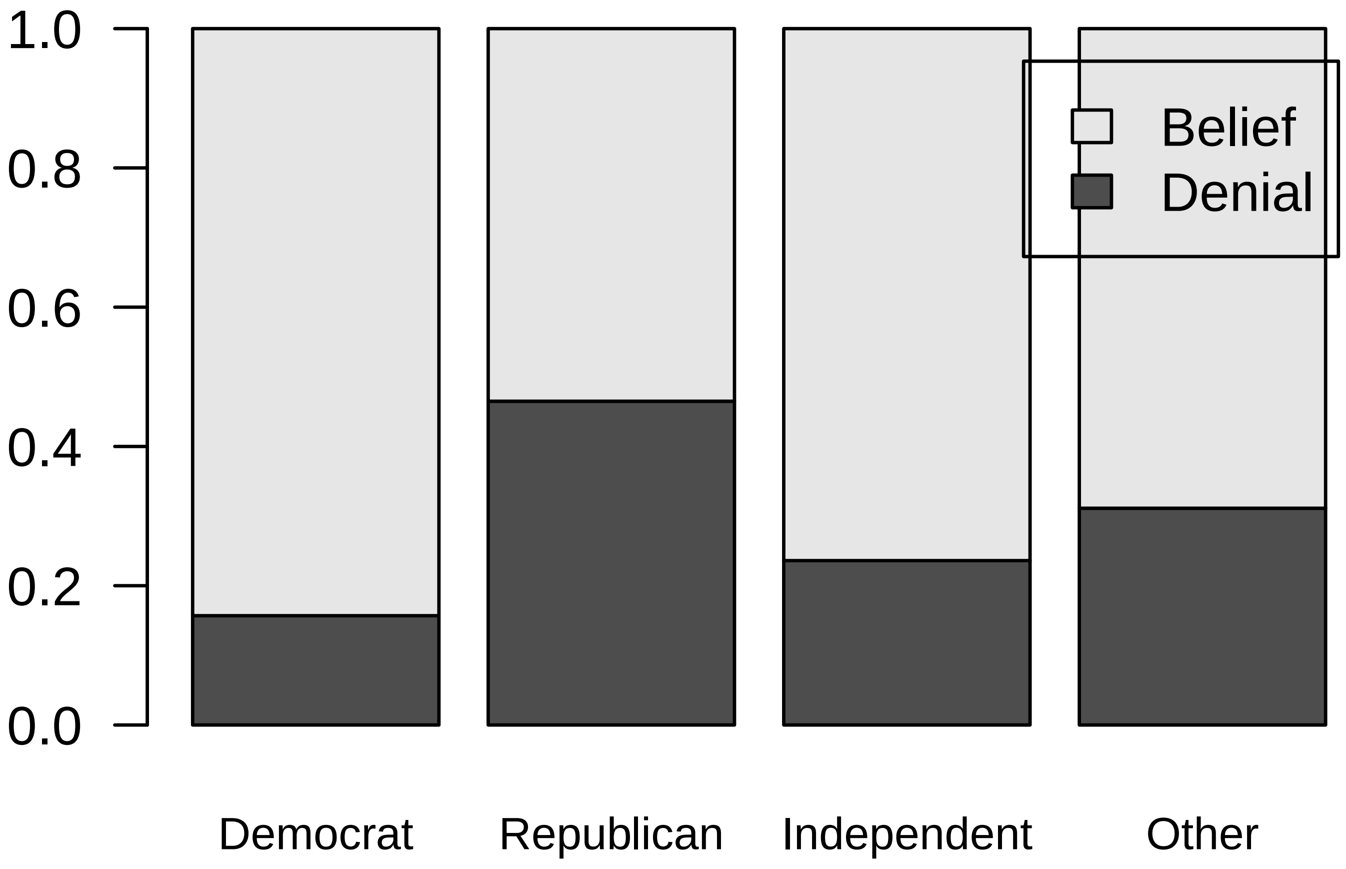
Note that this table looks deceptively similar to the table above. But look again. The numbers now don't add up to one within each row. They do however add up to one within each column. In order to read this table properly, we have to understand that it is giving us the distribution within each column: the proportion of respondents who have a given belief who belong to a given political party. So we can see in the first number that 19.8% of deniers are Democrats, 48.2% are Republicans, 27.9% are Independents, and 4.1% belong to other parties. This distribution is very different from the party affiliation distribution of believers in the second column which tells us that there is an association. However, the large party cleavages on the issue are not as immediately obvious here as they were with the previous conditional distribution. Always think carefully about which conditional distribution is more sensible to interpret and always make sure that you are interpreting them in the correct way.

It is also possible to graph the conditional distribution as a set of barplots. First, lets save the output of our prop.table into a new object.

```
distBeliefByParty <- prop.table(tab,1)
```

We can then use the **barplot** command to graph these distributions. However, there is one "gotcha" that we need to be aware of when running this command. When looking at conditional distributions across rows, barplot will misinterpret our data because it expects it to be oriented so things sum to one down the columns. We can however easily fix this with the **t** command ("t"" for "transpose"") which rotates our results:

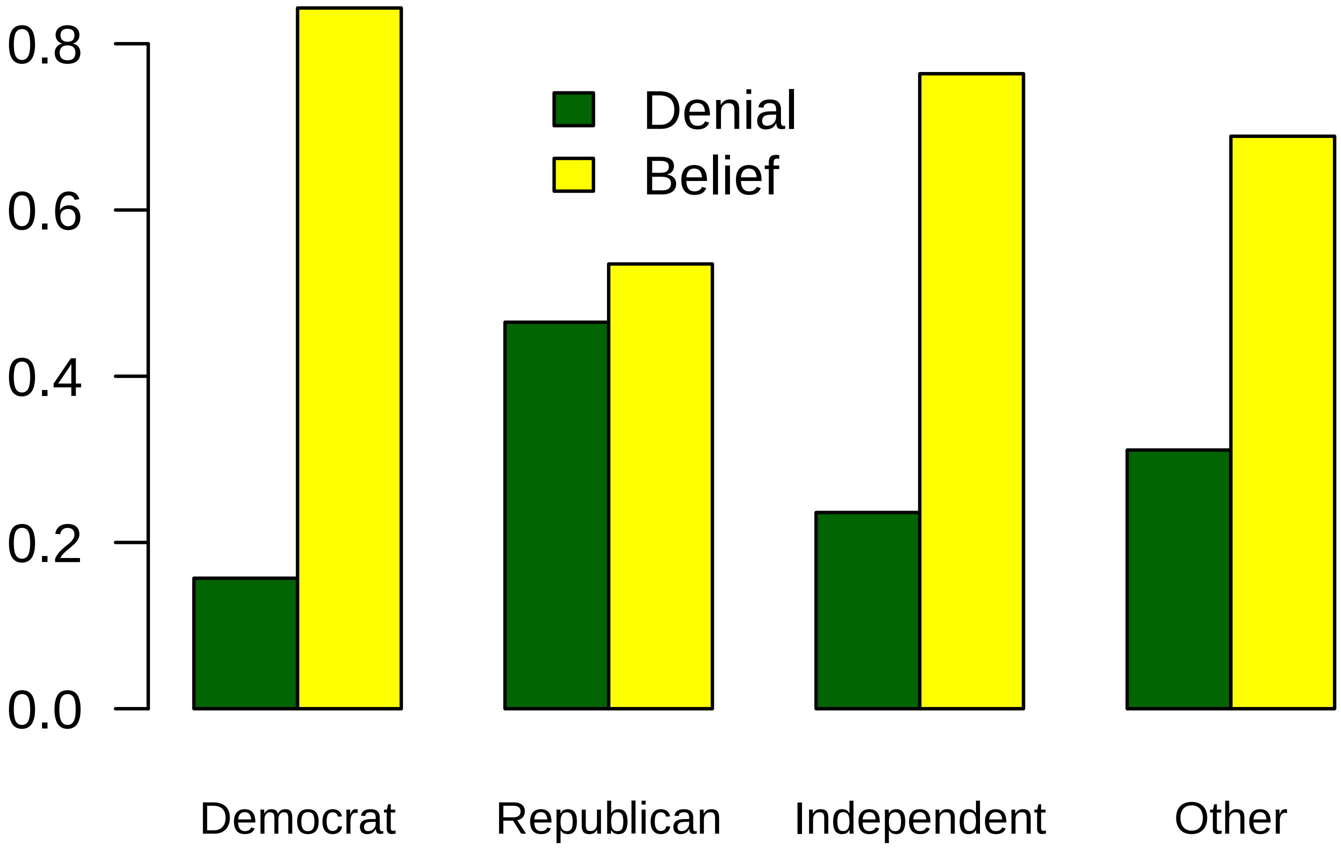
```
barplot(t(distBeliefByParty), legend.text=c("Denial", "Belief"), las=1, cex.names=0.8)
```



Note that when we use barplot like this, it gives us each party in one bar and then shades the bar to show the share of each response. I have used the legend.text option here to indicate which shading belongs to which category.

Another option is to graph the bars for each category next to each other and grouped by party. You can do this with the `beside` option which you set to TRUE:

```
barplot(t(distBeliefByParty), beside=TRUE, las=1, col=c("darkgreen","yellow"), cex.names=0.8)
legend(4, 0.8, legend=c("Denial","Belief"), fill=c("darkgreen","yellow"), bty="n")
```



In this case, I have also added custom colors for the two categories and designed my own legend that I can place better than the automated one.

---

## Odds ratio (Advanced)

We can also use the **odds ratio** to measure the association between two categorical variables. The odds ratio is not a term that is common in everyday speech but it is a critical concept in all kinds of scientific research.

Lets take the different distributions of climate change belief for Democrats and Republicans. About 84% of Democrats were believers, but only 54% of Republicans were believers. How can we talk about how different these two numbers are from one another? We could subtract one from the other or we could take the ratio by dividing one by the other. However, both of these approaches have a major problem. Because the percents (and proportions) have minimum and maximum values of 0 and 100, as you approach those boundaries the differences necessarily have to shrink because one group is hitting its upper or lower limit. This makes it difficult to compare percentage or proportional differences across different groups because the overall average proportion across groups will affect the differences in proportion.

Odds ratios are a way to get around this problem. To understand odds ratios, you first have to understand what odds are. All probabilities (or proportions) can be converted to a corresponding odds. If you have a $p$ probability of success on a task, then your odds $O$ of success are given by:

$$O = \frac{p}{1-p}$$

The odds are basically the ratio of the probability of success to the probability of failure. This tells you how many successes you expect to get for every one failure. Lets say a baseball player gets a hit 25% of the time that the player comes up to bat (an average of 250). The odds are then given by:

$$O = \frac{0.25}{1-0.25} = \frac{0.25}{0.75} = 0.33333$$

The hitter will get on average 0.33 hits for every one out. Alternatively, you could say that the hitter will get one hit for every three outs.

Re-calculating probabilities in this way is useful because unlike the probability, the odds has no upper limit. As the probability of success approaches one, the odds will just get larger and larger.

We can use this same logic to construct the odds that a Democratic and Republican respondent, respectively, will be climate change believers. For the Democrat, the probability is 0.84, so the odds are:

$$O = \frac{0.84}{1-0.84} = \frac{0.84}{0.16} = 5.250$$

Among Democrats, there are 5.25 believers for every one denier. Among Republicans, the probability is 0.541, so the odds are:

$$O = \frac{0.541}{1-0.541} = \frac{0.541}{0.459} = 1.179$$

Among Republicans, there are 1.17 believers for every one denier. This number is close to "even" odds of 1, which happen when the probability is 50%.

The final step here is to compare those two odds. We do this by taking their **ratio**, which means we divide one number by the other:

$$\frac{5.25}{1.179} = 4.45$$

This is our odds ratio. How do we interpret it? This odds ratio tells us how much more or less likely climate change belief is among Democrats relative to Republicans. In this case, I would say that "the odds of belief in anthropogenic climate change are 4.45 times higher among Democrats than Republicans." Note the "times" here. This 4.45 is a multiplicative factor because we are taking a ratio of the two numbers.

You can calculate odds ratios from conditional distributions just as I have done above, but there is also a short cut technique called the **cross-product**. Lets look at the two-way table of party affiliation but this time just for Democrats and Republicans. For reasons I will explain below, I am going to reverse the ordering of the columns so that believers come first.

| | Believer | Denier |
|---|---:|---:|
| **Democrat** | **1230** | *234* |
| **Republican** | *670* | **569** |

The two bolded numbers are called the **diagonal** and the two italicized numbers are the **reverse diagonal**. The cross-product is calculated by multiplying the two numbers in the diagonal by each other and multiplying the two numbers in the reverse diagonal together and then dividing the former product by the latter:

$$\frac{1230 * 569}{670 * 234} = 4.46$$

I get the exact same odds ratio as above (except for rounding error), without having to calculate the proportions and odds themselves. This is a useful shortcut for calculating odds ratios. There is one thing to keep in mind, however. The odds ratio that you calculate is always the odds of the first row being in the first column relative to those odds for the second row. Its easy to show how this would be different if I had kept the original ordering of believers and deniers:

| | Denier | Believer |
|---|---|---|
| **Democrat** | **234** | *1230* |
| **Republican** | *569* | **670** |

$$\frac{234 * 670}{569 * 1230} = 0.22$$

I get a very different odds ratio, but that is because I am calculating something different. I am now calculating the odds ratio of being a denier rather than a believer. So I would say that the "the odds of denial of anthropogenic climate change among Democrats are only 22% of the odds for Republicans." In other words, the odds of being a denier are much lower among Democrats.

Note, however, that the information here is the same because the 0.22 here is exactly equal to 1/4.46. In other words, the odds ratio of denial is just the inverted mirror image of the odds ratio of belief. Its just important that you remember that when you calculate the cross-product, you are always calculating the odds ratio of being in the category of the first column, whatever category that may be.

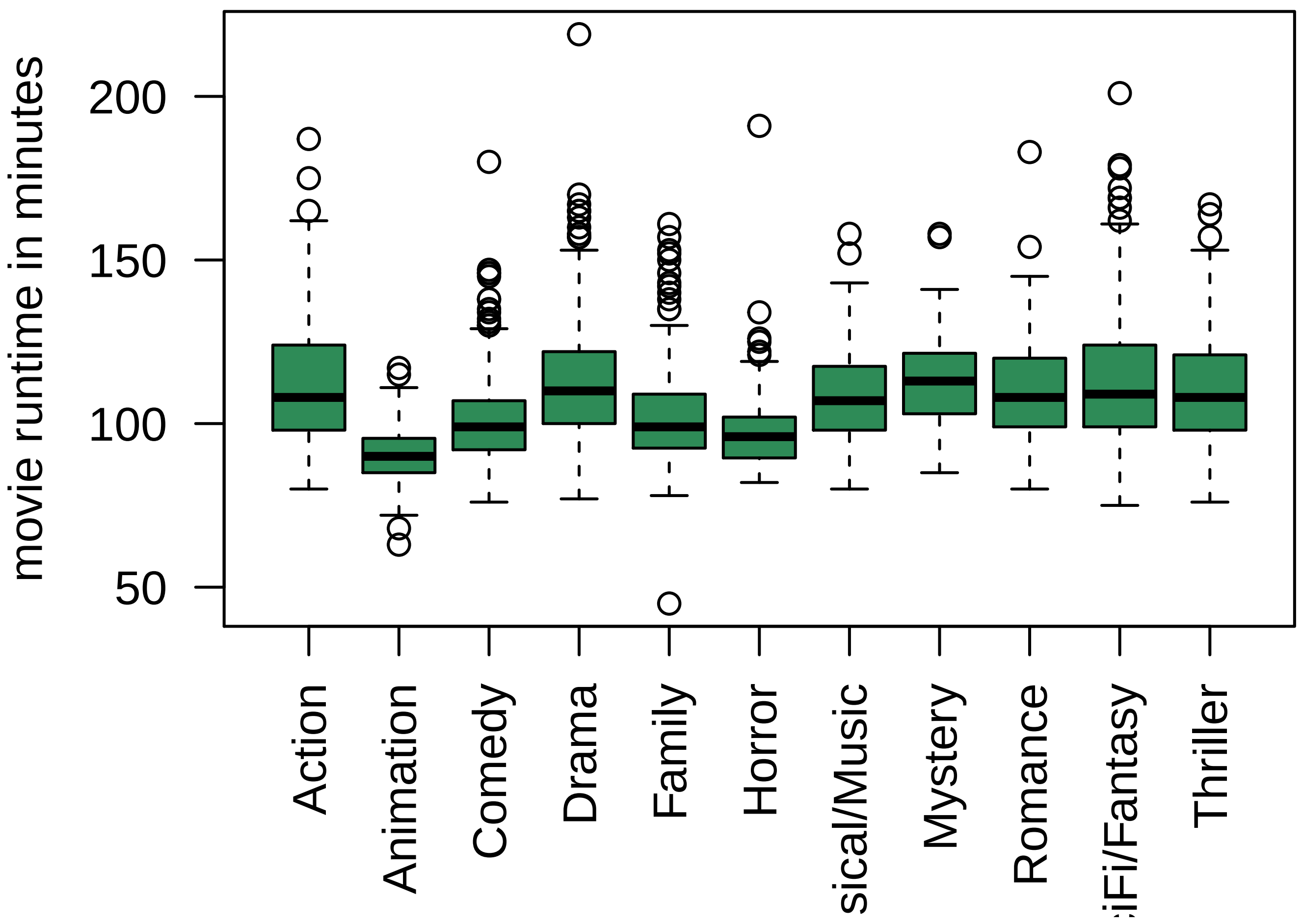
## Mean Differences

Measuring association between a quantitative and categorical variable is fairly straightforward. We want to look for differences in the distribution of the quantitative variable at different categories of the categorical variables. For example, if we were interested in the gender wage gap, we would want to compare the distribution of wages for women to the distribution of wages for men. There are two ways we can do this. First, we can graphically examine the distributions using the techniques we have already developed, particularly the boxplot. Second, we can compare summary measures like the mean across categories.

### Graphically examining differences in distributions

We could compare entire histograms of the quantitative variable across different categories of the categorical variable, but this is often too much information. A cleaner method is to use **comparative boxplots**. Comparative boxplots construct boxplots of the quantitative variable across all categories of the categorical variable and plot them next to each other for easier comparison. Here is an example looking at differences in movie runtime across different movie genres.

```r
boxplot(Runtime~Genre, data=movies, col="seagreen", ylab="movie runtime in minutes", las=2)
```



This command introduces a new syntax. The tilde ("~") in the syntax above allows us to relate one variable to another. In this context, it tells the boxplot command to plot separate boxplots of runtime by the categories of genre.

Comparative boxplots are useful because they allow us to easily compare differences in both the center and spread of distributions at the same time. The thick bars give us an indication of the median movie runtime for each genre. Here we can clearly see that animated and horror movies have the lowest median runtimes and that mysteries have the longest median runtimes. We can also see that while there is significant variation, some genres like romance, scifi/fantasy, and thriller all have similar median runtimes.

However, the boxplot also reveals additional information on the spread of runtimes for each genre. This is easiest to see by comparing the height of the boxes (the interquartile range) across genres. In particular, we can see that scifi/fantasy, action, and dramas have the largest variation in movie runtime, while horror and animated movies have very small variation. Clearly, there is more consensus about the appropriate length of horror and animated movies, than there is about action, drama, and scifi/fantasy movies.

## Comparing differences in the mean

We can also establish a relationship by looking at differences in summary measures. Implicitly, we are already doing this in the comparative boxplot when we look at the median bars across categories. However, in practice it is more common to compare the mean of the quantitative variable across different categories of the categorical variable. In *R*, you can get the mean of one variable at different levels of a categorical variable using the `tapply` command like so:

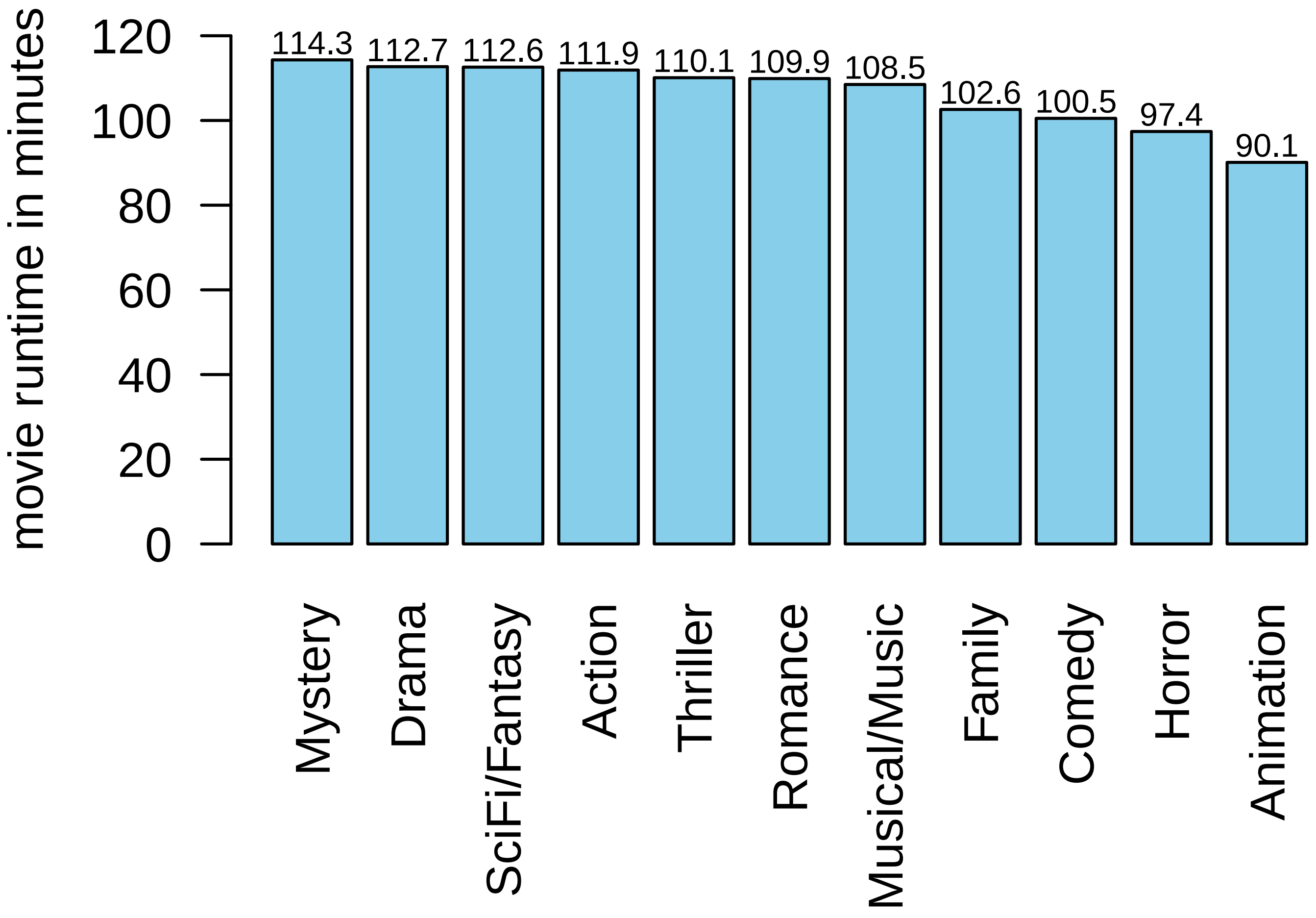
```
tapply(movies$Runtime, movies$Genre, mean)
```

```
##          Action     Animation        Comedy         Drama        Family
##       111.91304      90.06475     100.45711     112.65060     102.60000
##          Horror Musical/Music       Mystery       Romance SciFi/Fantasy
##        97.39545     108.46602     114.25641     109.89855     112.56202
##        Thriller
##       110.13260
```

The `tapply` command takes three options. The first option is the quantitative variable for which we want means. The second option is the categorical variable. The third option is the method we want to run on the quantitative variable, in this case the mean. The output is the mean movie runtime by genre.

We can also display these results graphically using barplots. To make this prettier, I am first going to sort the output from largest to smallest mean runtime:

```
mruntime <- tapply(movies$Runtime, movies$Genre, mean)
mruntime <- sort(round(mruntime,1), decreasing=TRUE)
par(mar=c(7,4,4,2),las=2)
b <- barplot(mruntime, col="skyblue", ylab="movie runtime in minutes", ylim=c(0,125))
text(b[,1],mruntime+4,label=paste(mruntime), cex=0.7)
```



Now we can see very clearly how mean movie runtimes compare across different genres. We can also easily calculate the mean difference in movie length across any two genres. For example, lets say we are interested in the difference between action movies (111.9 minutes, on average) and horror movies (97.4 minutes, on average).

$$111.9 - 97.4 = 14.5$$

Action movies are, on average, 15 minutes longer than horror movies.

---

## Scatterplot and Correlation Coefficient

The techniques for looking at the association between two quantitative variables are more developed than the other two cases, so we will spend more time on this topic. Additionally, the major approach here of ordinary least squares regression turns out to be a very flexible, extendable method that we will build on later in the term.

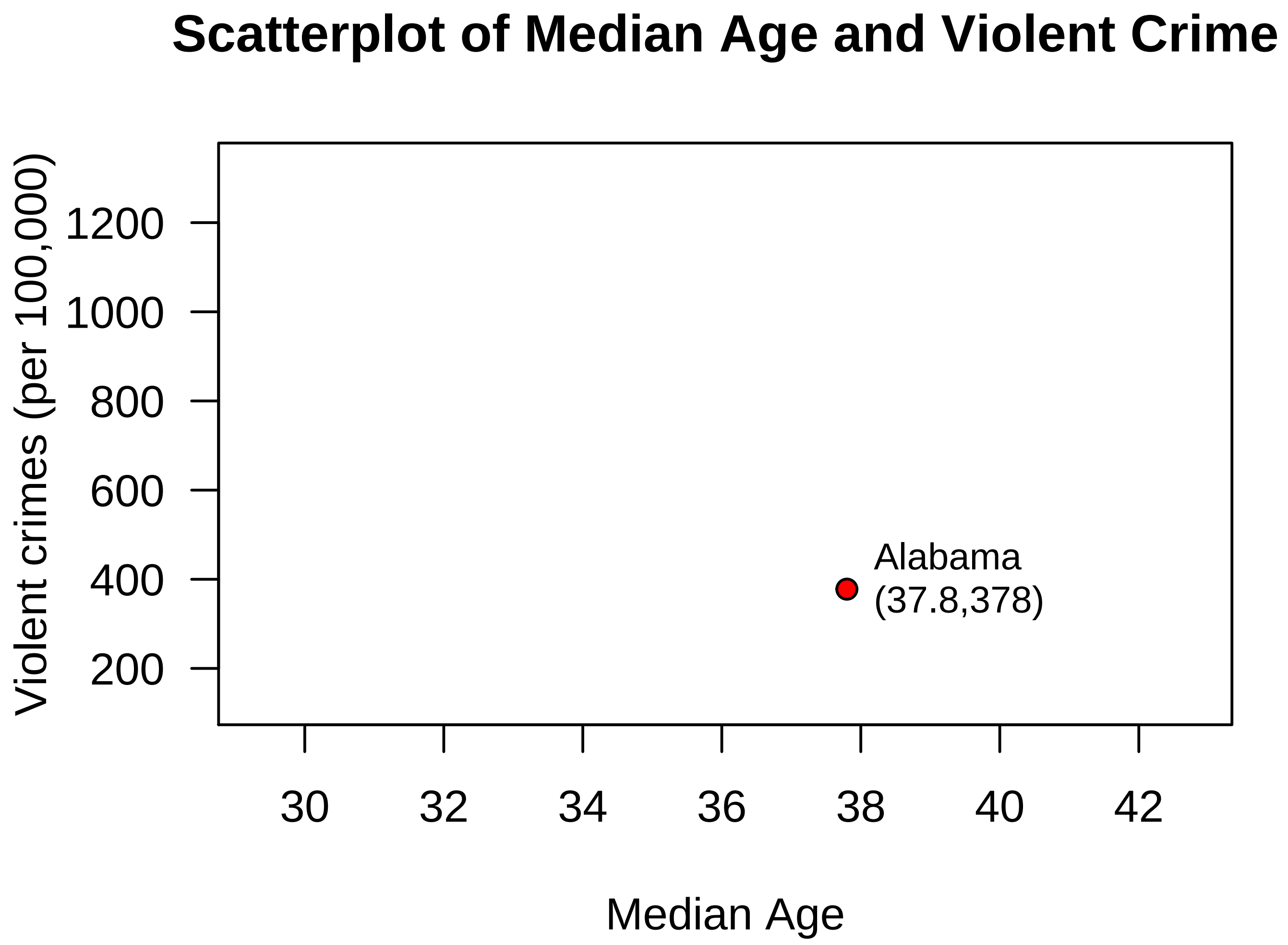
When examining the association between two quantitative variables, we usually distinguish the two variables by referring to one variable as the **dependent variable** and the other variable as the **independent variable**. The dependent variable is the variable whose outcome we are interested in predicting. The independent variable is the variable that we treat as the predictor of the dependent variable. For example, lets say we were interested in the relationship between income inequality and life expectancy. We are interested in predicting life expectancy by income inequality, so the dependent variable is life expectancy and the independent variable is income inequality.

The language of dependent vs. independent variable is causal, but its important to remember that we are only measuring the association. That association is the same regardless of which variable we set as the dependent and which we set as the independent. Thus, the selection of the dependent and independent variable is more about which way it more intuitively makes sense to interpret our results.

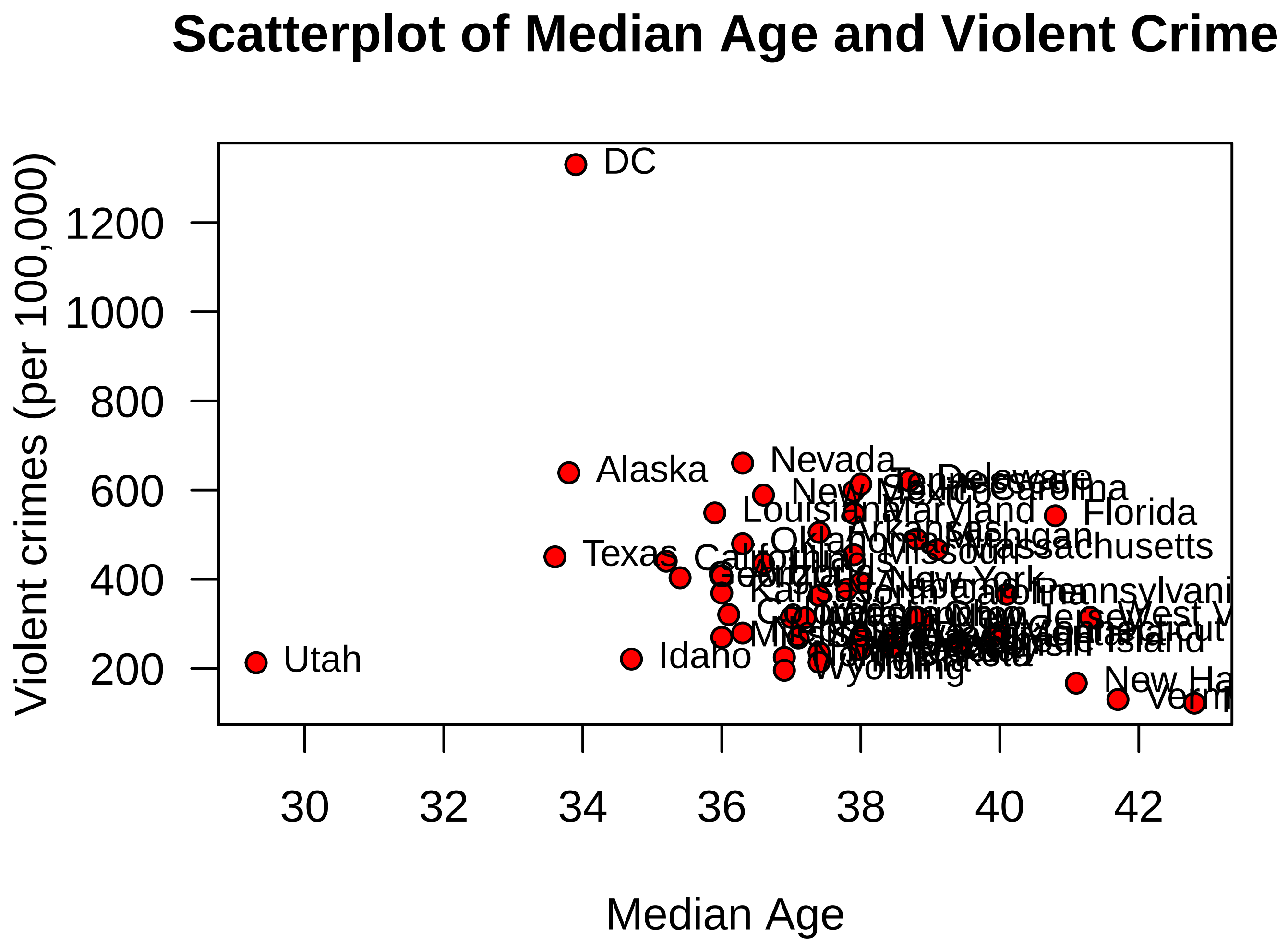
### The scatterplot

We can examine the relationship between two quantitative variables by constructing a **scatterplot**. A scatterplot is a two-dimensional graph. We put the independent variable on the x-axis and the dependent variable on the y-axis. For this reason, we often refer generically to the independent variable as x and the dependent variable generically as y.

To construct the scatterplot, we plot each observation as a point, based on the value of its independent and dependent variable. For example, lets say we are interested in the relationship between the median age of the state population and violent crime in our crime data. Our first observation, Alabama, has a median age of 37.8 and a violent crime rate of 378 crimes per 100,000. We can plot this point on our graph as follows:



If I repeat that process for all of my observations, I will get a scatterplot that looks like:



It is not necessary to show the names of states here, but I have done so here in order to identify individual observations.

What are we looking for when we look at a scatterplot? There are four important questions we can ask of the scatterplot. First, what is the **direction** of the relationship. We refer to a relationship as **positive** if both variables move in the same direction. if y tends to be higher when x is higher and y tends to be lower when x is lower, then we have a positive relationship. On the other hand, if the variables move in opposite directions, then we have a **negative** relationship. If y tends to be lower when x is higher and y tends to be higher when x is lower, then we have a negative relationship. In the case above, it seems like we have a generally negative relationship. States with higher median age tend to have lower violent crime rates.

Second, is the relationship **linear**? I don't mean here that the points fall exactly on a straight line (which is part of the next question) but rather does the general shape of the points appear to have any "curve" to it. If it has a curve to it, then the relationship would be non-linear. This issue will become important later, because our two primary measures of association are based on the assumption of a linear relationship. In this case, there is no evidence that the relationship is non-linear.

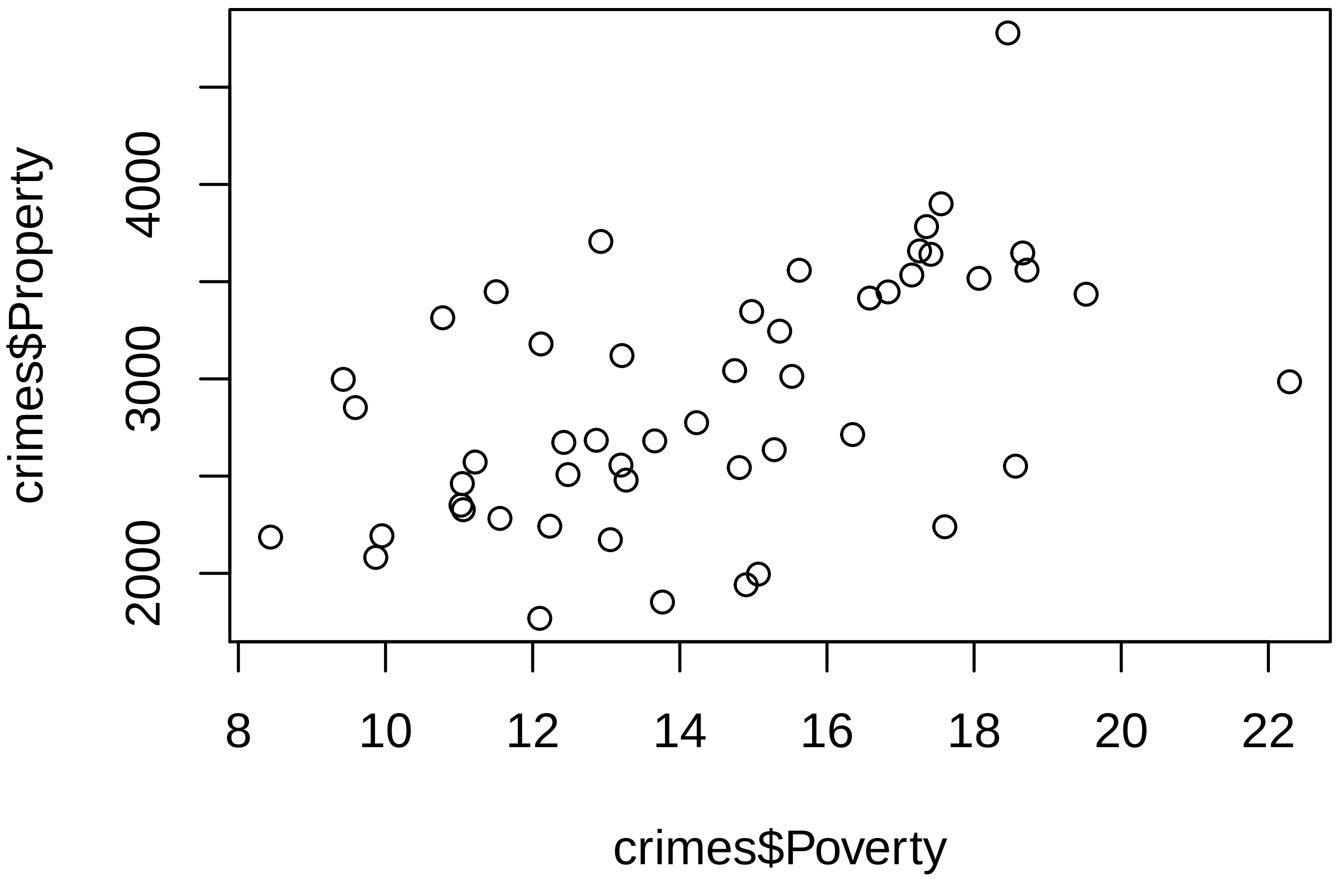
Third, what is the **strength** of the relationship. If all the points fall exactly on a straight line, then we have a very strong relationship. On the other hand, if the points form a broad elliptical cloud, then we have a weak relationship. In practice, in the social sciences, we never expect our data to conform very closely to a straight line. Judging the strength of a relationship often takes practice. I would say the relationship above is of moderate strength.

Fourth, are there **outliers**? We are particularly concerned about outliers that go against the general trend of the data, because these may exert a strong influence on our later measurements of association. In this case, there are two clear outliers, Washington DC and Utah. Washington DC is an outlier because it has an extremely high level of violent crime relative to the rest of the data. Its median age tends to be on the younger side, so its placement is not inconsistent with the general trend. Utah is an outlier that goes directly against the general trend because it has one of the lowest violent crime rates and the youngest populations. This is, of course, driven by Utah's heavily Mormon population, who both have high rates of fertility (leading to a young population) and whose church communities are able to exert a remarkable degree of social control over these young populations.

#### Constructing scatterplots in *R*

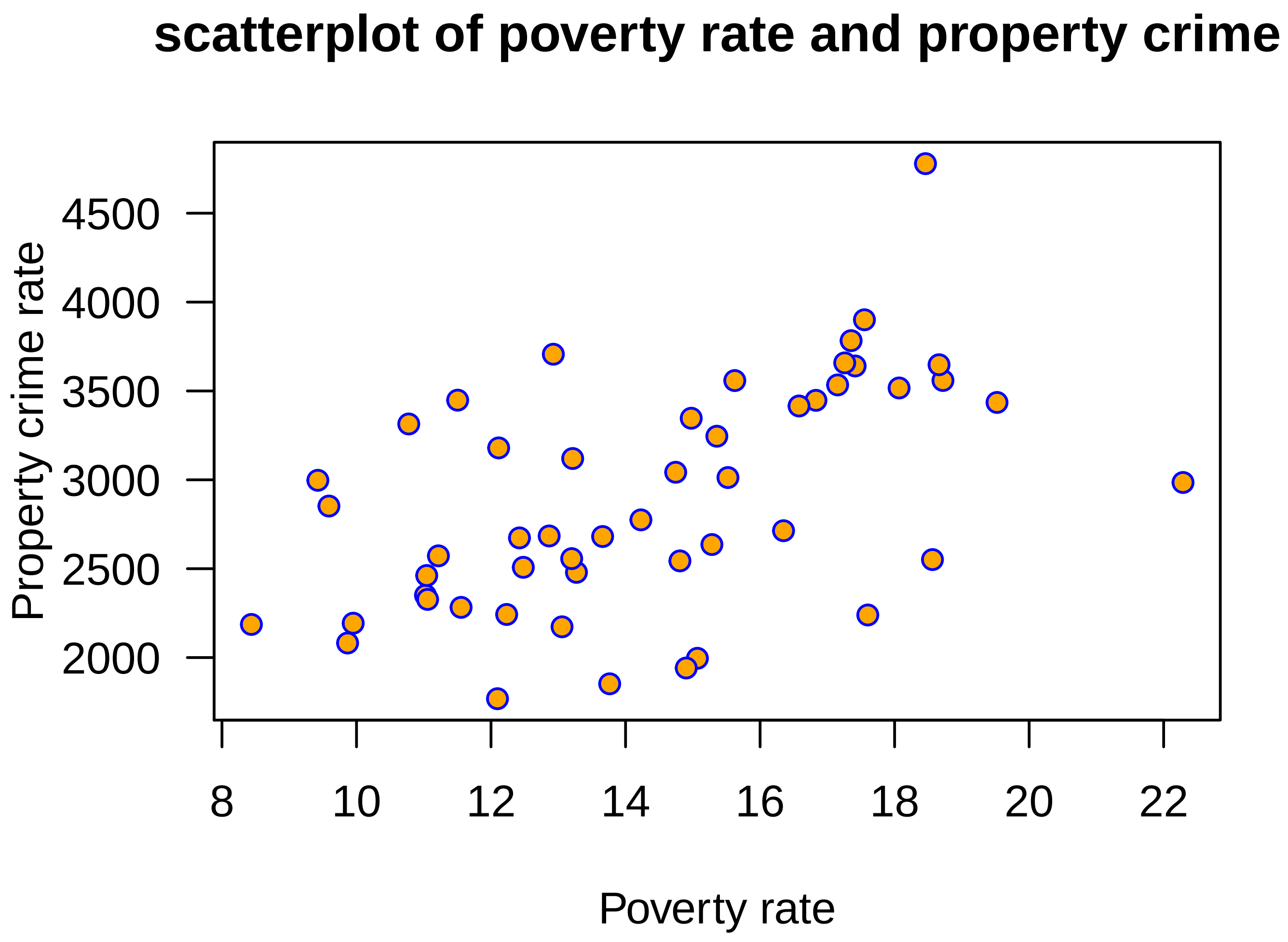
You can construct scatterplots in *R* with the `plot` command. At a very basic level, you can just feed in the independent and dependent variables:

```
plot(crimes$Poverty, crimes$Property)
```



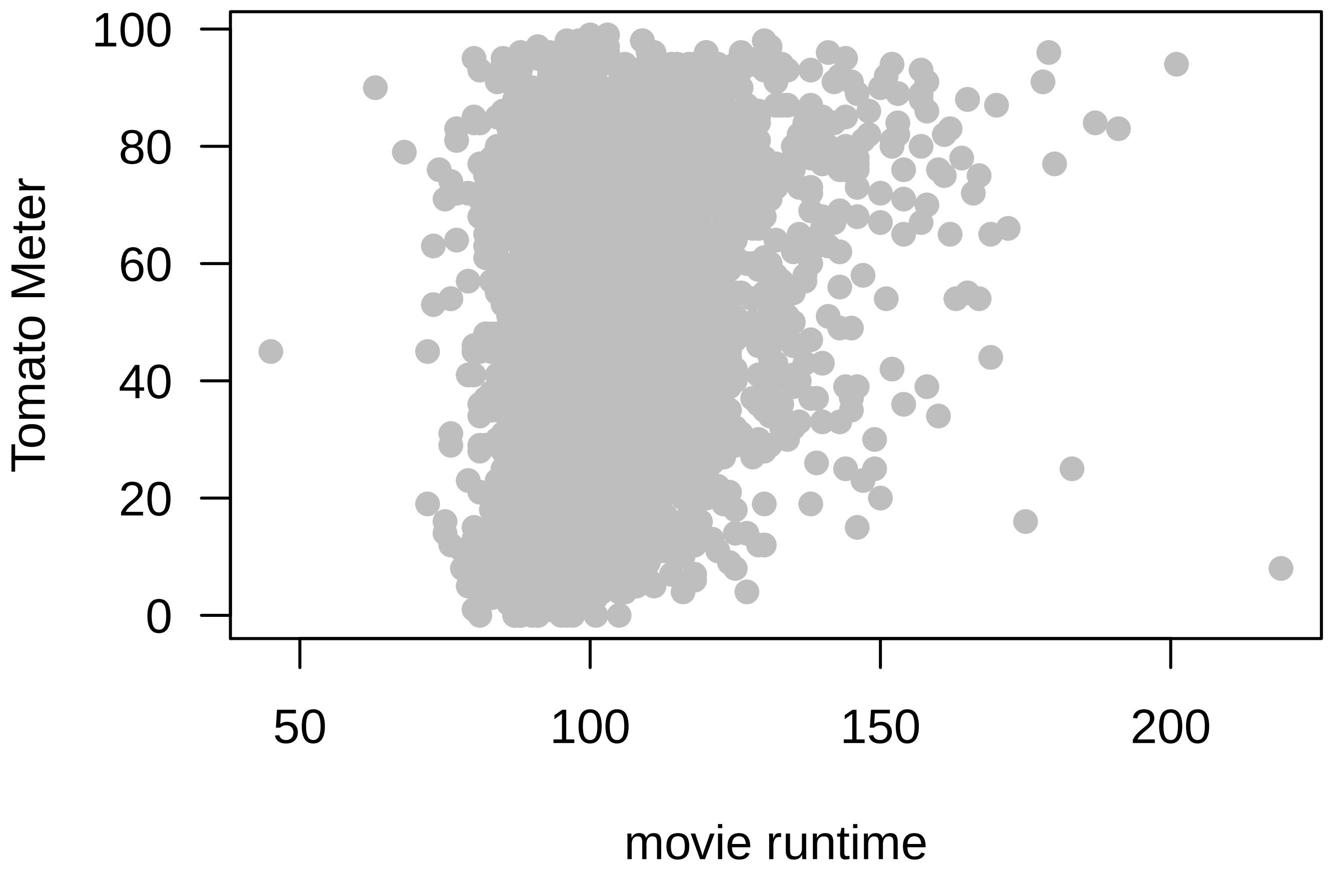
We can improve on this command with our usual bells and whistles to label x and y axes, titles, etc. We can also use the `pch` option to define different dots for our points. If you pull up the help command `?points`, it will show a list of codes for all the different points you can use. I like to use `pch=21` because it will draw circles where you can color in both the center and the border. For example:

```
plot(crimes$Poverty, crimes$Property,
     xlab="Poverty rate", ylab="Property crime rate",
     main="scatterplot of poverty rate and property crime",
     las=1, pch=21, bg="orange", col="blue")
```



Sometimes with large datasets, scatterplots can be difficult to read because of the problem of **overplotting**. This happens when many data points overlap, so that its difficult to see how many points are showing. For example:

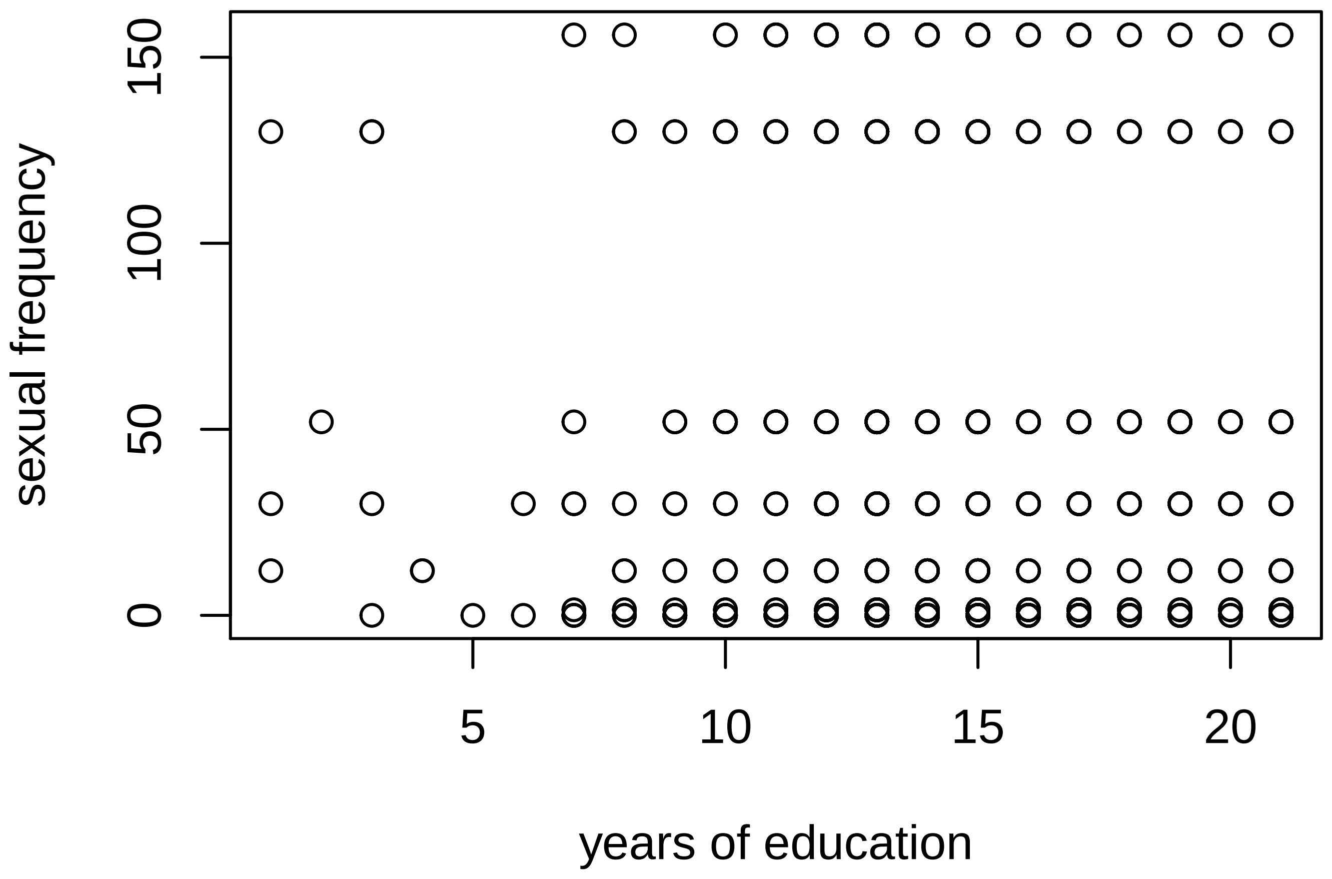
```
plot(movies$Runtime, movies$TomatoMeter, pch=21, bg="grey", col="grey", las=1,
     xlab="movie runtime", ylab="Tomato Meter")
```



Because so many movies are in that 90-120 minute range it is difficult to distinguish them and thus a little tricky to summarize the relationship. There are more advanced ways we can address this issue, but these are beyond our introductory class.

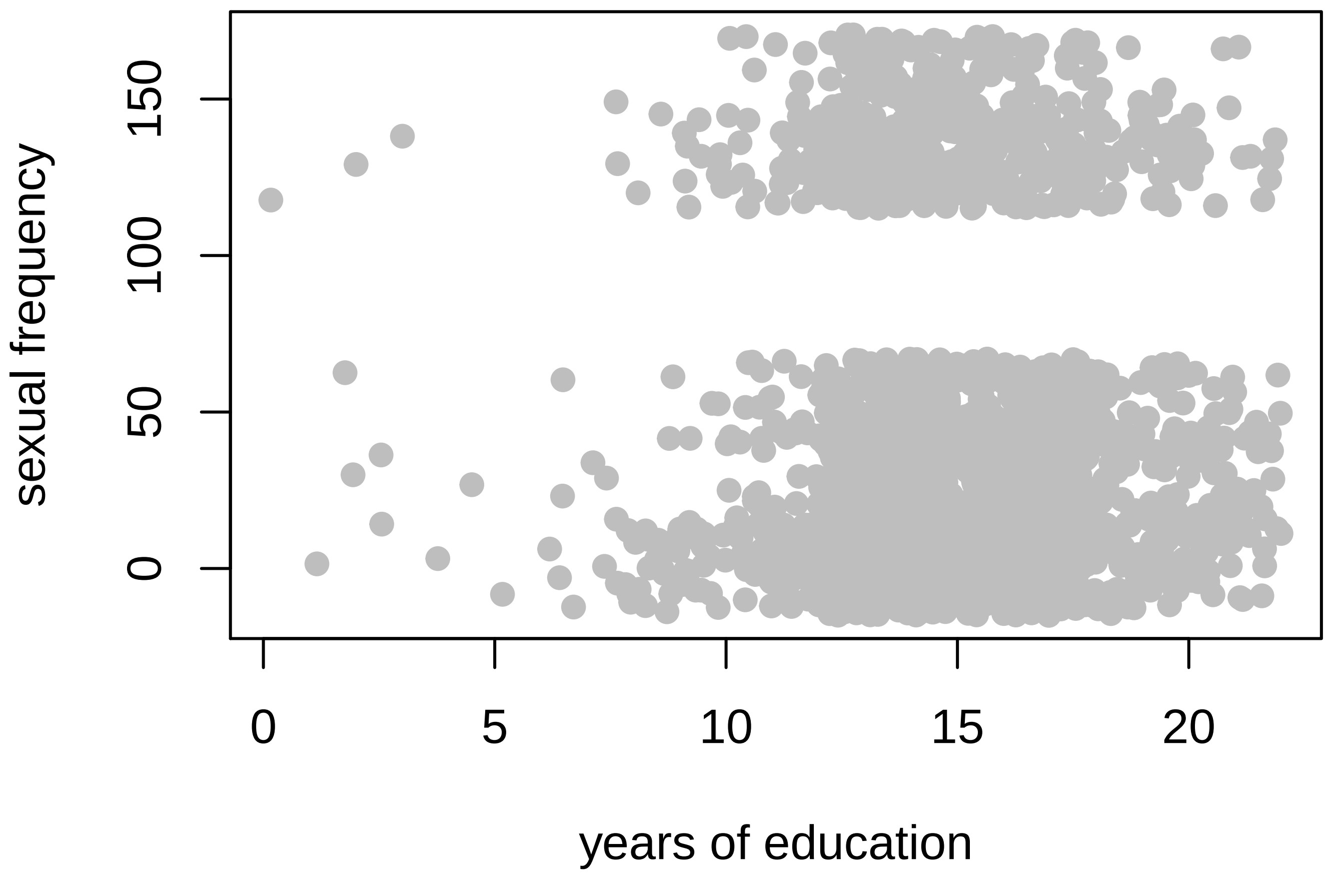
Overplotting can also be a problem with discrete variables because these variables can only take on certain values which will then exactly overlap with one another. For example:

```
plot(sex$educ, sex$sexf,
     xlab="years of education", ylab="sexual frequency")
```



Both of these variables are discrete and thus only take certain values, so all we see on the graph is more or less every possible point based on the possible values of education and sexual frequency. Multiple points are plotted directly on top of each other. We can fix this issue more easily by applying the `jitter` command which just adds a random amount to each observation so they don't exactly overlap. You often have to experiment with `jitter` to find just how much randomness (the second option) you have to apply.

```r
plot(jitter(sex$educ,5), jitter(sex$sexf,50),
     xlab="years of education", ylab="sexual frequency",
     pch=21, bg="grey", col="grey")
```



In this case, because of the large number of observations we still have a lot of overplotting.

## The correlation coefficient

We can measure the association between two quantitative variables with the correlation coefficient, $r$. The formula for the correlation coefficient is:

$$r = \frac{1}{n-1}\sum_{i=1}^{n}(\frac{x_i - \bar{x}}{s_x} * \frac{y_i - \bar{y}}{s_y})$$

That looks complicated, but lets break it down step by step. We will use the association between median age and violent crimes as our example.

The first step is to subtract the means from each of our x and y variables. This will give us the distance above or below the mean for each variable.

```
diffx <- crimes$MedianAge-mean(crimes$MedianAge)
diffy <- crimes$Violent-mean(crimes$Violent)
```

The second step is to divide these differences from the mean of x and y by the standard deviation of x and y, respectively.

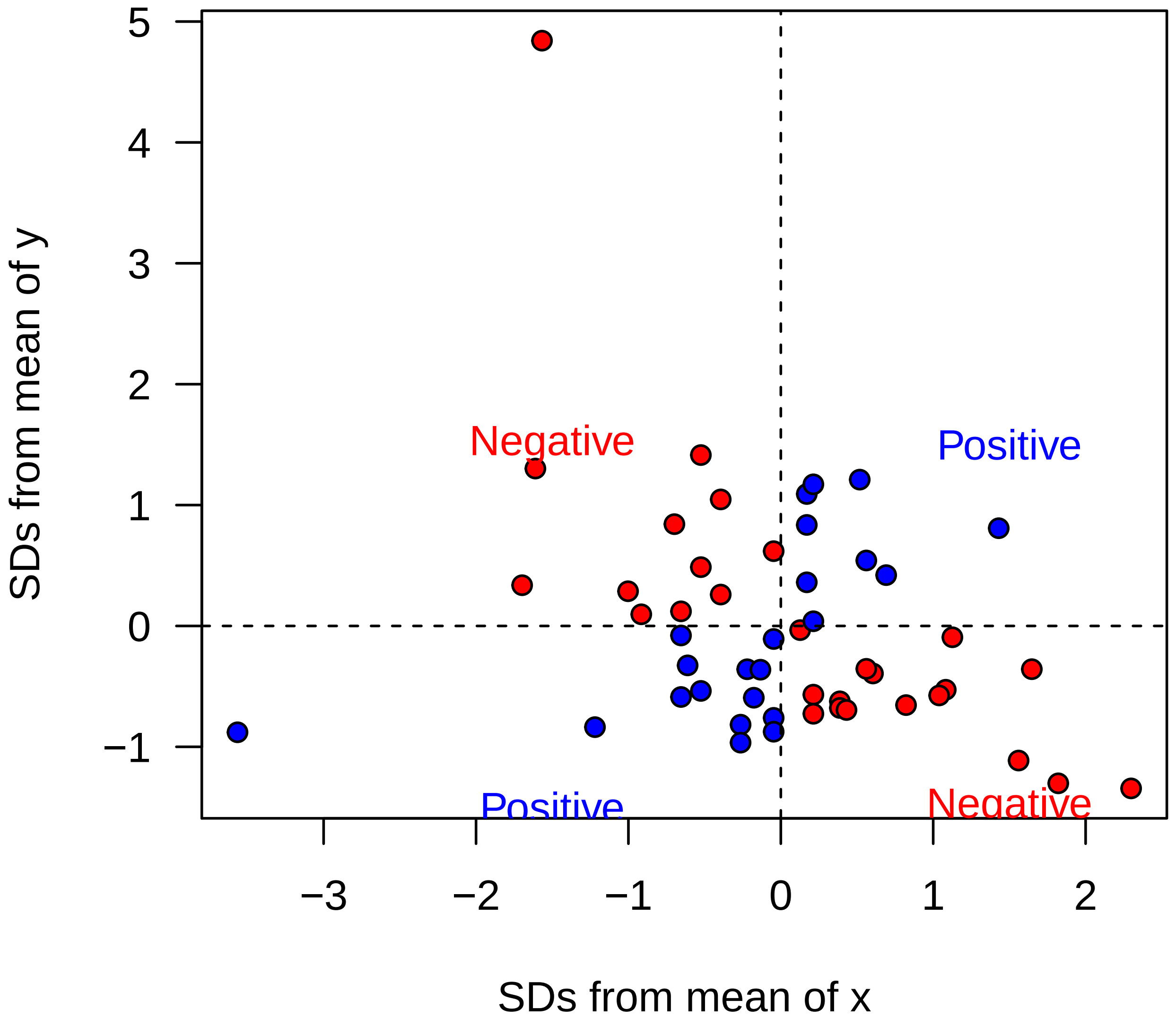
```
diffx.sd <- diffx/sd(crimes$MedianAge)
diffy.sd <- diffy/sd(crimes$Violent)
```

Now each of your x and y values have been converted from their original form into the **number of standard deviations above or below the mean**. This is often called **standarization**. By doing this, we have put both variables on the same scale and have removed whatever original units they were measured in (in our case, years of age and crimes per 100,000).

The third step is to to multiply each converted value of x by each converted value of y.

```
product <- diffx.sd*diffy.sd
```

Why do we do this? First consider this scatterplot of our standardized x and y:



Points shown in blue have either both positive or both negative x and y values. When you take the product of these two numbers, you will get a positive product. This is evidence of a positive relationship. Points shown in red have one positive and one negative x and y value. When you take the product of these two numbers, you will get a negative product. This is evidence of a negative relationship.

The final step is to add up all this evidence of a positive and negative relationship and divide by the number of observations (minus one).

```
sum(product)/(length(product)-1)
```

```
## [1] -0.3015232
```

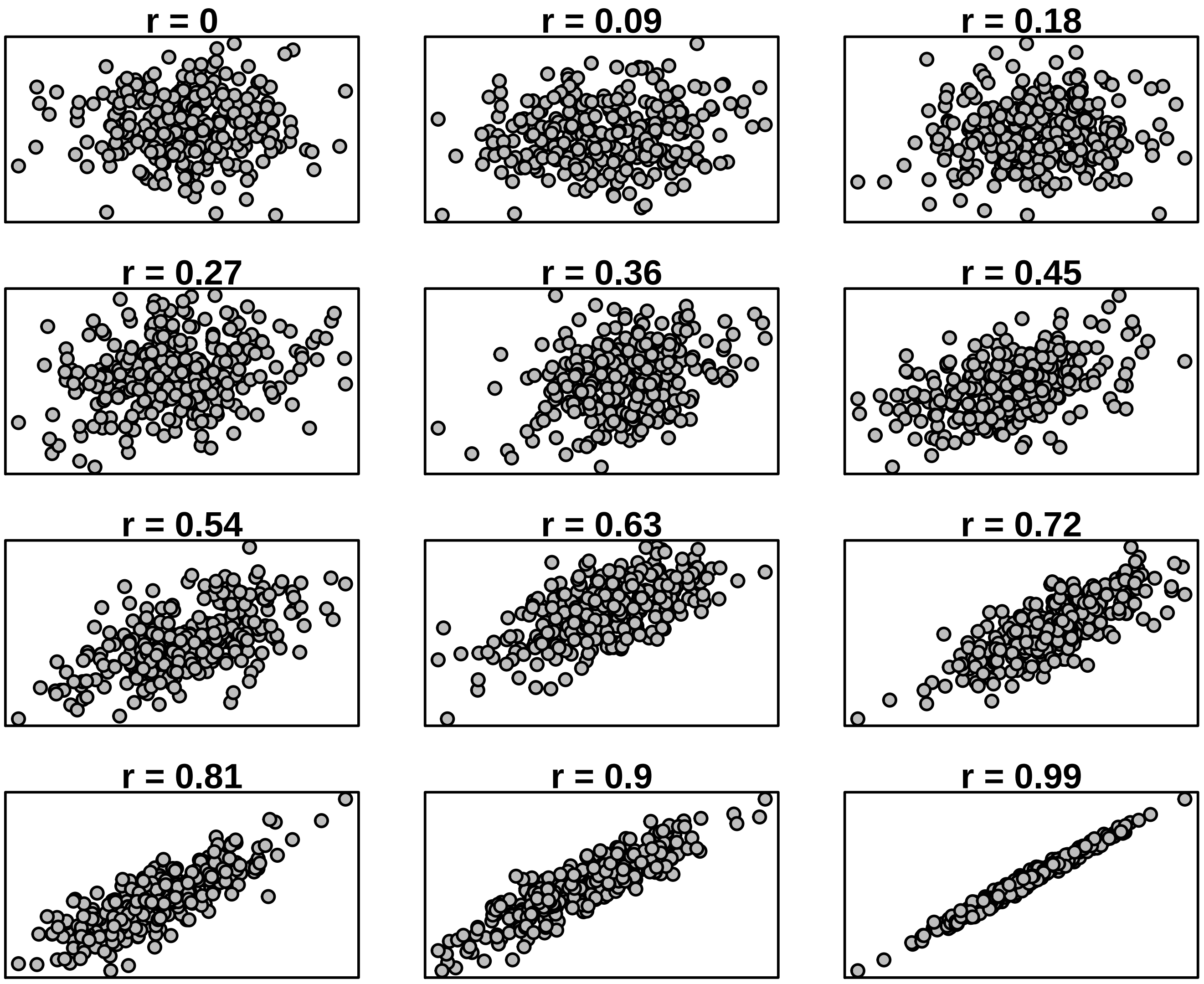
This final value is our correlation coefficient. We could have also calculated it by using the `cor` command:

```r
cor(crimes$MedianAge, crimes$Violent)
```

```
## [1] -0.3015232
```

How do we interpret this correlation coefficient? It turns out the correlation coefficient $r$ has some really nice properties. First, the **sign** of $r$ indicates the direction of the relationship. If $r$ is positive, the association is positive. If $r$ is negative, the association is negative. if $r$ is zero, there is no association.

Second, $r$ has a **maximum** value of 1 and a **minimum** value of -1. These cases will only happen if the points line up exactly on a straight line, which never happens with social science data. However, it gives us some benchmark to measure the strength of our relationship. Here are some simulated scatterplots with different $r$ in order to help you get a sense of the strength of association for different values of $r$.



Third, $r$ is a **unitless** measure of association. It can be compared across different variables and different datasets in order to make a comparison of the strength of association. For example, the correlation coefficient between unemployment and violent crimes is 0.45. Thus, violent crimes are more strongly correlated with unemployment than with median age (0.44>0.30). The association between median age and property crimes is -0.36, so median age is more strongly related to property crimes than violent crimes (0.36>0.30).

There are some important cautions when using the correlation coefficient. First, the correlation coefficient will only give a proper measure of association when the underlying relationship is **linear**. if there is a non-linear (curved) relationship, then $r$ will not correctly estimate the association. Second, the correlation coefficient can be affected by **outliers**. We will explore this issue of outliers and influential points more in later sections.

# Chapter 4

# Statistical Inference

## The Problem of Statistical Inference

So far, we have only been looking at measurements from our actual datasets. We examined both **univariate statistics** like the mean, median, and standard deviation, as well as **measures of association** like the mean difference, correlation coefficient and OLS regression line slope. We can use this measures to draw conclusions about our data.

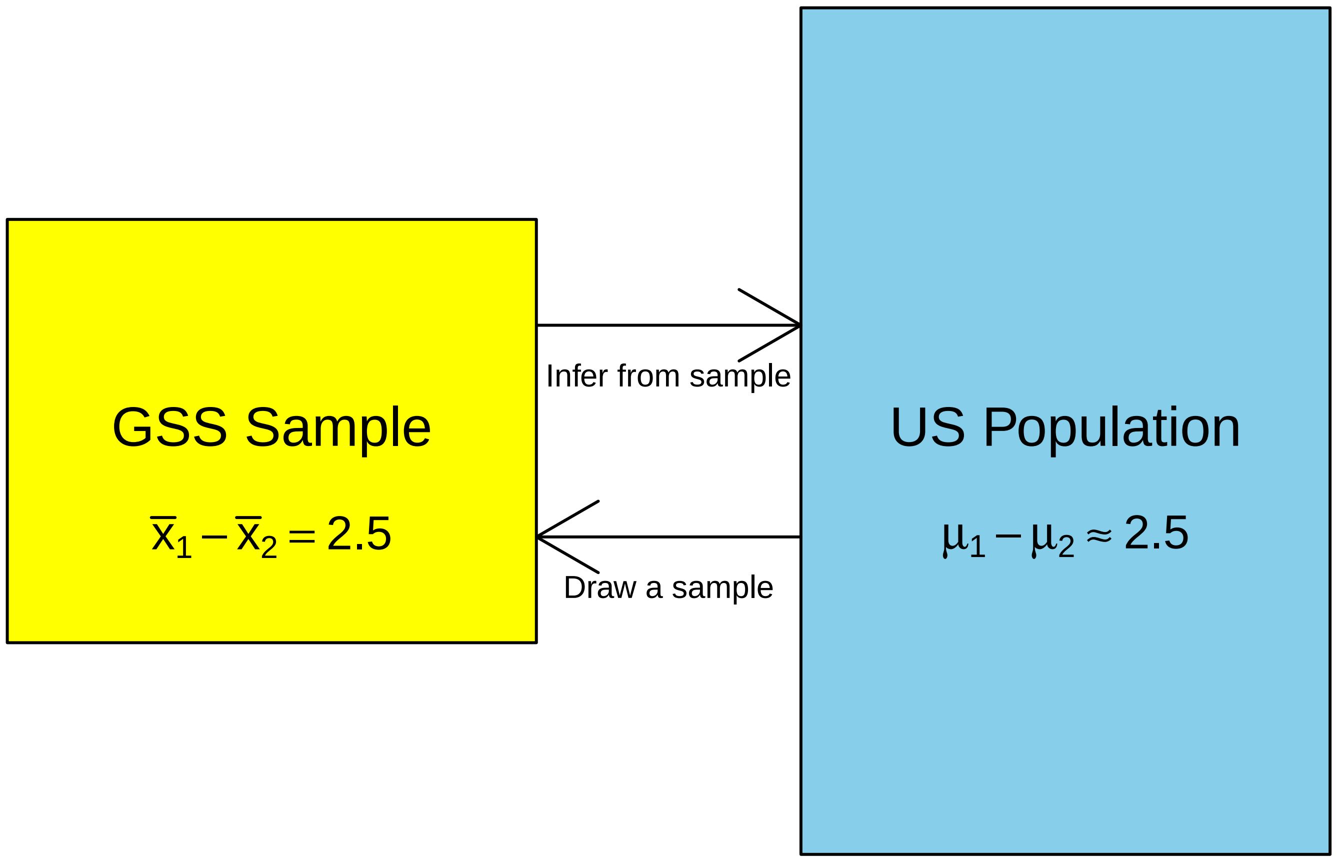
In many cases, the dataset that we are working is only a **sample** from some larger **population**. Importantly, we don't just want to know something about the sample, but rather we want to know something about the population from which the sample was drawn. For example, when polling organizations like Gallup conduct political polls of 500 people, they are not drawing conclusions about just those 500 people, but rather about the whole population from which those 500 people were sampled.

To take another example from our General Social Survey (GSS) data on sexual frequency. We can calculate the mean sexual frequency by marital status:

```r
tapply(sex$sexf, sex$marital, mean)
```

```
##       Married       Widowed      Divorced     Separated Never married 
##     56.094106      9.222628     41.388720     55.652778     53.617704
```

Married respondents had sex 2.5 (56.1-53.6) times more per year than never married individuals. We don't want to draw this conclusion just for our sample. Rather, we want to know what the relationship is between marital status and sexual frequency in the US population as a whole. In other words, we want to **infer** from our sample to the population. The figure below shows this process graphically.



The large blue rectangle is the population that we want to know about. Within this population, there is some value that we want to know. In this case, that value is the mean difference in sexual frequency between married and never married individuals. We refer to this unknown value in the population as a **parameter**. You will also notice that there are some funny-looking greek letters in that box. We always use greek symbols to represent values in the population. In this case, $\mu_1$ is the population mean of sexual frequency for married individuals and $\mu_2$ is the population mean of sexual frequency for never married individuals. Thus, $\mu_1 - \mu_2$ is the population mean difference in sexual frequency between married and never married individuals.

We typically don't have data on the entire population, which is why we need to draw a sample in the first place. Therefore, these population parameters are unknown. To estimate what they are, we draw a sample as shown by the smaller yellow square. In this sample, we can calculate the sample mean difference in sexual frequency between married and never married individuals, $\bar{x}_1 - \bar{x}_2$. We refer to a measurement in the sample as a **statistic**. We represent these statistics with roman letters to distinguish them from the corresponding value in the population.

**The statistic is always an estimate of the parameter**. In this case, $\bar{x}_1 - \bar{x}_2$ is an estimate of $\mu_1 - \mu_2$. We can infer from the sample to the population and conclude that our best guess as to the true mean difference in the population is the value we got in the sample.

The sample mean difference may be our best guess as to the true value in the population, but how confident are we in that guess? Intuitively, if I only had a sample of 10 people I would be much less confident than if I had a sample of 10,000 people. **Statistical inference** is the technique of *quantifying* our uncertainty in the estimate. If you have ever read the results of a political poll, you will be familiar with the term "margin of error." This is a measure of statistical inference.

Why might our sample produce inaccurate results? There are two sources of **bias** that could result in our sample statistic being different from the true population parameter. The first form of bias is **systematic bias**. Systematic bias occurs when something about the procedure for generating the sample produces a systematic difference between the sample and the population. Sometimes, systematic bias results from the way the sample is drawn. For example, if I sample my friends and colleagues on their voting behavior, I will likely introduce very large systematic bias in my estimate of who will win an election because my friends and colleagues are more likely than the general population to hold similar views to my own. Systematic bias can also result from the way questions are worded, the characteristics of interviewers, the time of day interviews are conducted, etc. Systematic bias can often be minimized in well-designed and executed scientific surveys. Statistical inference cannot do anything to account for systematic bias.

The second form of bias is **random bias**. Random bias occurs when the sample statistic is different from the population parameter, just by random chance due to the actual sample that was drawn. In other words, even if there is no systematic bias in my survey design, I can get a bad estimate simply due to the bad luck of drawing a really unusual sample. Imagine that I am interested in estimating mean wealth in the United States and I happen to draw Bill Gates in my sample. I am probably going to dramatically overestimate mean wealth. Random bias affects every sample, regardless of how well-designed and executed.

In practice, the sample statistic is extremely unlikely to be *exactly* equal to the population parameter, so some degree of random bias is always present in every sample. However, this random bias will become less important as the sample size increases. In the previous example, Bill Gates is going to bias my results much more if I draw a sample of 10 people, than if I draw a sample of 100,000 people. Our goal with statistical inference is to more precisely quantify how bad that random bias *could* be in our sample.

Notice the word "could" in the previous sentence. The tricky part about statistical inference is that while we know that random bias could be causing our sample statistic to be very different from the population parameter, we never know for sure whether random bias had a big effect or a small effect in our particular sample, because we don't have the population parameter with which we could compare it. Keep this issue in mind in the next sections, as it plays a key role in how we understand our procedures of statistical inference.

It is also important to keep in mind that statistical inference only works when you are actually drawing a sample from a larger population that you want to draw conclusions about. In some cases, our data either constitute a unique event, as in the Titanic case, that cannot be properly considered a sample of something larger or the data actually constitute the entire population of interest, as is the case in our dataset on movies. Although you will occasionally still see people use inferential measures on such data, it is technically inappropriate because there is no larger population to make inferences about.

---

## The Concept of the Sampling Distribution

Lets say that you want to know the mean years of education of US adults. You implement a well-designed representative survey that samples 100 respondents from the USA. You ask people the simple question "how many years of education do you have?" You then calculate the mean years of education in your sample.

This simple example involves three different kinds of distributions. Understanding the difference between these three different distributions is the key to unlocking how statistical inference works.

1. **The Population Distribution**. This is the distribution of years of education in the entire US population. The mean of this distribution is given by $\mu$ and is a population parameter. If we had data on the entire population we could show the distribution and calculate $\mu$. However, because we don't have data on the full population, the population distribution and $\mu$ are unknown. This distribution is also *static* - it doesn't fluctuate.

2. **The Sample Distribution**. This is the distribution of years of education in the sample of 100 respondents that I have. The mean of this distribution is $\bar{x}$ and is a sample statistic. Since we collected this data, this distribution and $\bar{x}$ are known. We can calculate $\bar{x}$ and we can show the distribution of years of education in the sample (with a histogram or boxplot, for example). The sample distribution is an approximation of the population distribution, but because of random bias, it may be somewhat

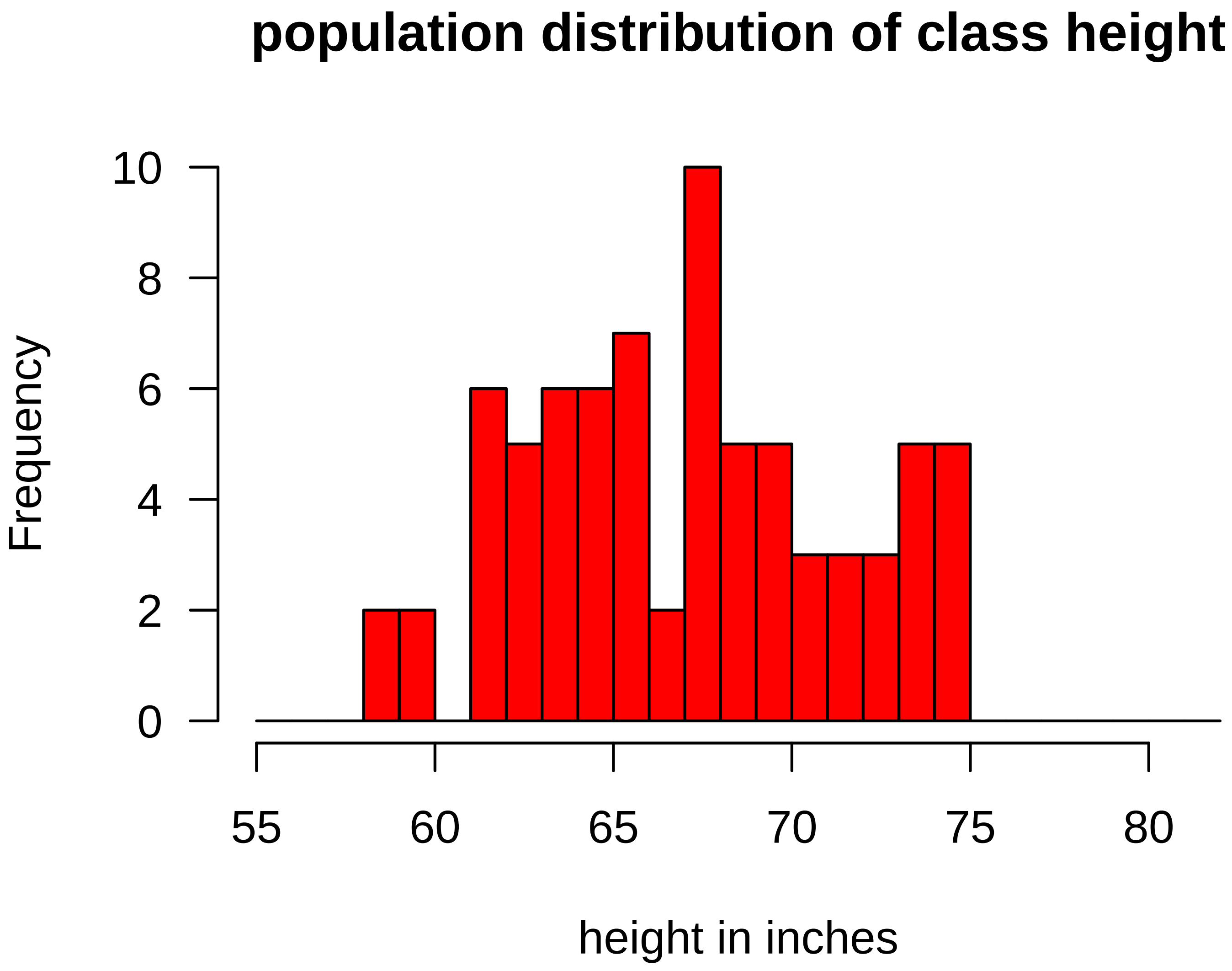
different. Also, because of this random bias, the distribution is not static - if we were to draw another sample the two sample distributions would almost certainly not be identical.

3. **The Sampling Distribution**. Imagine all the possible samples of size 100 that I could have drawn from the US population. Its a tremendously large number. If I had all those samples, I could calculate the sample mean of years of education for each sample. Then, I would have the mean years of education in every possible sample of size 100 that I could have drawn from the population. The sampling distribution is the distribution of all of these possible sample means. More generally, the sampling distribution is the distribution of the desired sample statistic in all possible samples of size $n$.

The sampling distribution is much more abstract than the other two distributions, but is key to understanding statistical inference. When we draw a sample and calculate a sample statistic from this sample, we are in effect reaching into the sampling distribution and pulling out a value. Therefore, the sampling distribution give us information about how variable our sample statistic might be as a result of randomness in the sampling.

### Example: class height

Lets treat our class of 75 students as a population that I would like to know something about. In this case, I would like to know the mean height of the class. In most cases, the population distribution is unknown but in this case, I know the height of all 75 students because of the surveys you all took at the beginning of the term. Here is the population distribution of height in our class:

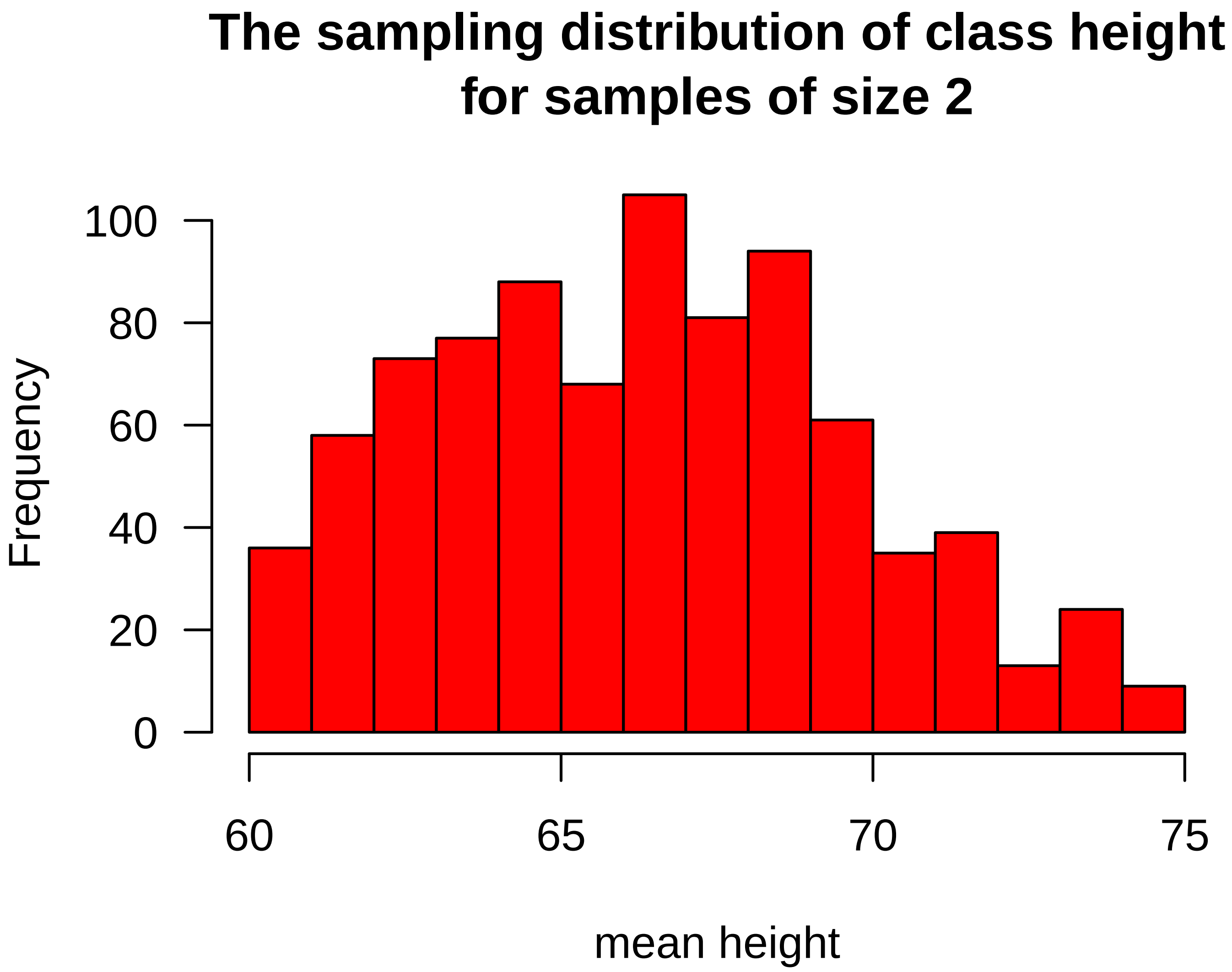


The population distribution of height is bimodal which is typical, because we are mixing the heights of men and women. The population mean of height ($\mu$) is 67.469 inches.

Lets say I lost the results of the student survey and I wanted to know the mean height of the class. I could take a random sample of two students in order to calculate the mean height. Lets say I drew a sample of two people who were 68 and 74 inches respectively in height. I would estimate a sample mean of 71 inches which in this case would be too high. Lets say I took another sample of two students and ended up with a mean height of 66 which would be too low. Lets say I repeat this procedure until I had sampled all possible combinations of two students out of the twenty in the class.

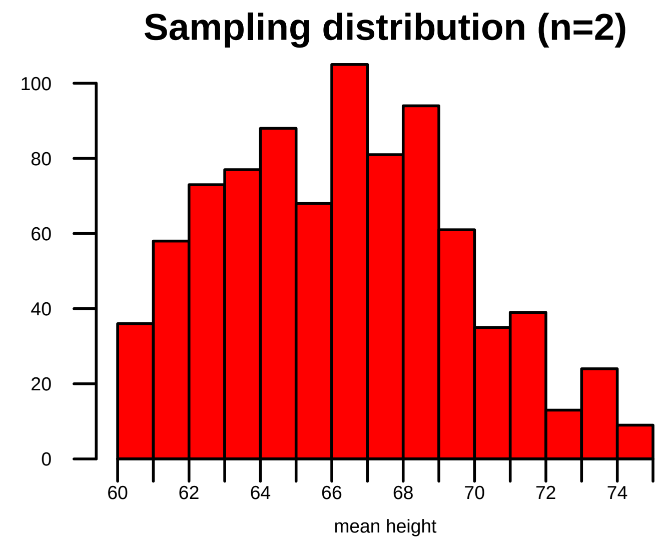
How many samples would this be? On the first draw from the population of 75 students there are 75 possible results. On the second draw, there are 74 possible results, because I won't re-sample the student I selected the first time. This gives me 75*74=5550 possible combinations of 75 students in two draws. However, half of these draws are just mirror images of the other draws where I swap the first and second draw. Since I don't care about the order, I actually have 5550/2=2775 possible samples.

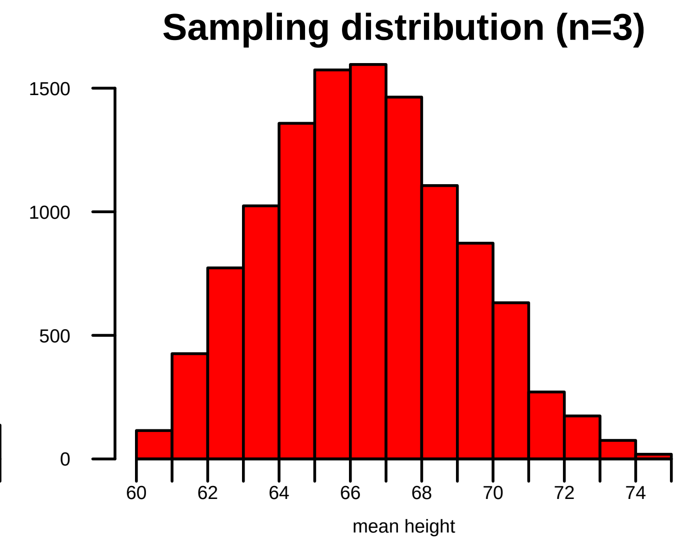
I have used a computer routine to actually calculate the sample means in all 2775 of those possible samples. The distribution of these sample means then gives us the sampling distribution of mean height for samples of size 2. Here is a histogram of that distribution.

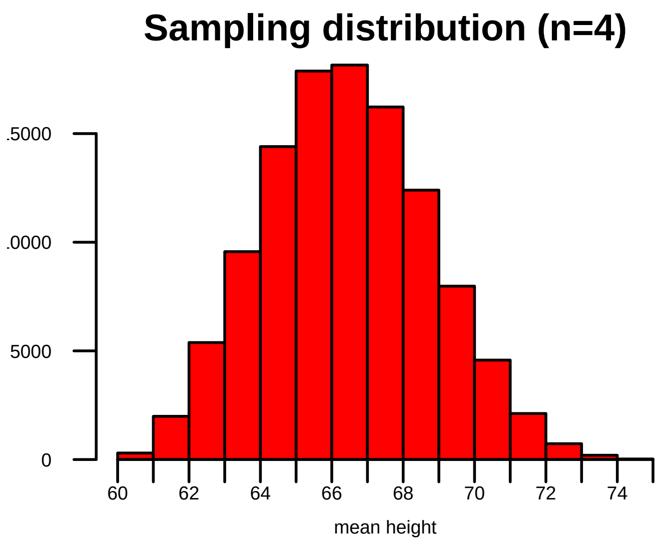


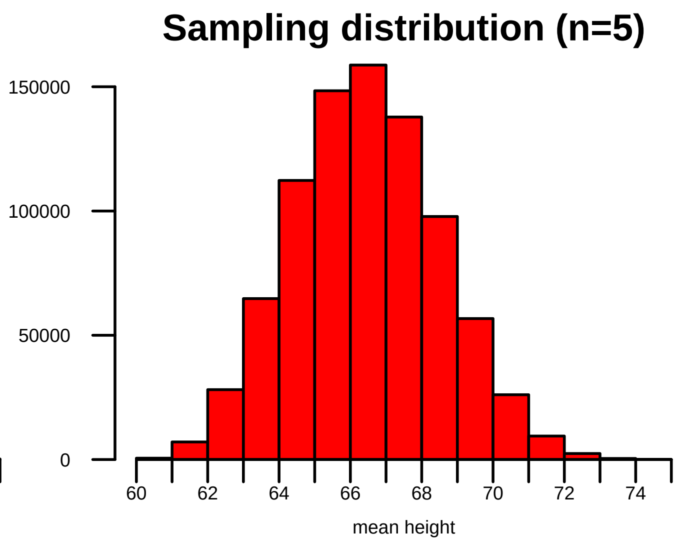
When I randomly draw one sample of size 2 and calculate its mean, I am effectively reaching into this distribution and pulling out one of these values. Note that many of the means are clustered around the true population mean of 67.5 inches, but in a few cases I can get extreme overestimates or extreme underestimates.

What if I were to increase the sample size? I have used the same computer routine to calculate the sampling distribution for samples of size 3, 4, and 5. Here are the results:









There are three thing to note here. First, each sampling distribution seems to have most of their points clustered (where the peak is) around the true population mean of 67.5. In fact, the mean of these distributions is exactly the true population parameter, as I will show below. Second, the spread of the distributions is shrinking as the sample size increases. You can see that the when the sample size increases, the tails of the distribution are "pulled in" and more of the sample means are closer to the center. This indicates that we are less likely to draw a sample mean that is extremely different from the population mean in larger samples. Third, the shape of the distribution at larger sample sizes is becoming more symmetric and "bell-shaped."

Lets take a look at the mean and standard deviation of these sampling distributions:

| Distribution | Mean | Standard Deviation |
|---|---:|---:|
| Population Distribution | 67.469 | 4.344 |
| Sampling Distribution (n=2) | 66.517 | 3.363 |
| Sampling Distribution (n=3) | 66.517 | 2.71 |
| Sampling Distribution (n=4) | 66.517 | 2.316 |

| Distribution | Mean | Standard Deviation |
|---|---|---|
| Sampling Distribution (n=5) | 66.517 | 2.044 |

Note that the mean of each sampling distribution is identical to the true population mean. This is not a coincidence. The mean of the sampling distribution of sample means is always itself equal to the population mean. In statistical terminology, this is the definition of an **unbiased statistic**. Given that we are trying to estimate the true population mean, it is reassuring that the "average" sample mean we should get is the true population mean.

Also note that the standard deviation of the sampling distributions gets smaller with increasing sample size. This is the mathematically way of seeing the shrinking of the spread that we observed graphically. In larger sample sizes, we are less likely to draw a sample mean far away from the true population mean.

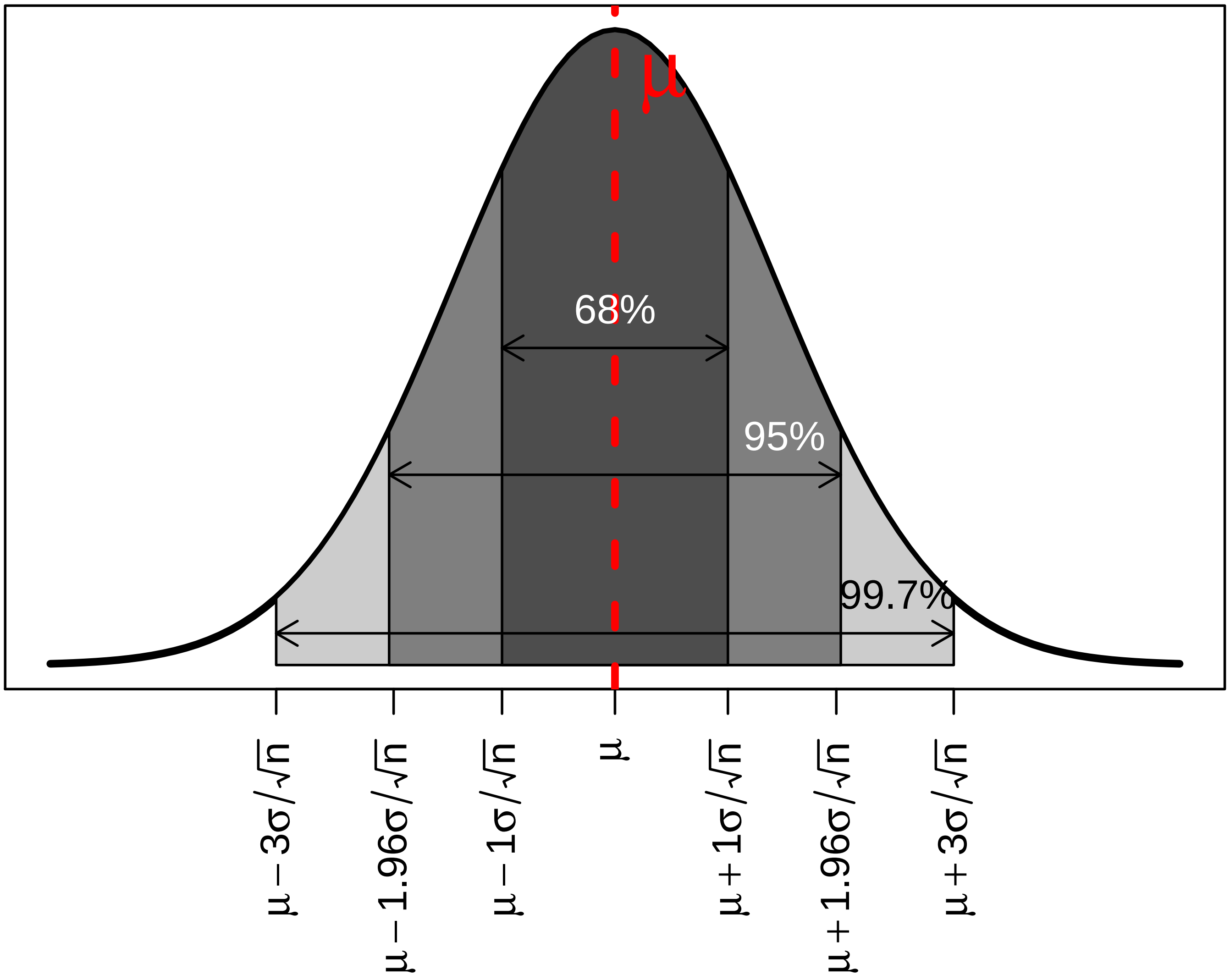
## Central limit theorem and the normal distribution

The patterns we are seeing here are well known to statisticians. In fact, they are patterns that are predicted by the most important theorem in statistics, the **central limit theorem**. We won't delve into the technical details of this theorem. We can generally interpret the central limit theorem to say:

> As the sample size increases, the sampling distribution of a sample mean becomes a **normal distribution**. This normal distribution will be centered on the true population mean $\mu$ and with a standard deviation equal to $\sigma/\sqrt{n}$, where $\sigma$ is the population standard deviation.

What is this "normal" distribution? The name is somewhat misleading because there is nothing particularly normal about the normal distribution. Most real-world distributions don't look normal, but the normal distribution is central to statistical inference because of the central limit theorem. The normal distribution is a bell-shaped, unimodal, symmetric distribution. It has two characteristics that define its exact shape. The mean of the normal distribution define where its center is and the standard deviation of the normal distribution defines its spread.

Lets look at the normal sampling distribution of the mean to become familiar with it.



The distribution is symmetric and bell shaped. The center of the distribution is shown by the red dotted line. This center will always be at the true population mean, $\mu$. The area of the normal distribution also has a regularity that is sometimes referred to as the "68%,95%,99.7%" rule. Regardless of the actual variable, 68% of all the sample means will be within 68% of the true population mean, 95% of all the sample means will be within 95% of the true population mean, and 99.7% of all the sample means will be within three standard deviations of the mean. This regularity will become very helpful later on for making statistical inferences.

**The standard error**

It is easy to get confused by the number of standard deviations being thrown around in this section. There are three standard deviations we need to keep track of to properly understand what is going on here. Each of these standard deviations is associated with one of the types of distributions I introduced at the beginning of this section.

1. $\sigma$: the **population standard deviation**. In our example, this would be the standard deviation of height for all 75 students which is 4.3441429. Typically, like other values in the population, this number is unknown.
2. $s$: the **sample standard deviation**. In our example, this would be the standard deviation of height from a particular sample of a given size from the class. This number can be calculated for the sample that you have, using the techniques we learned earlier in the class.

3. $\sigma/\sqrt{n}$: The **standard deviation of the sampling distribution of the sample mean**. We divide the population standard deviation $\sigma$ by the square root of the sample size. In general, we refer to the standard deviation of the sampling distribution as the **standard error**, for short. So remember that when I refer to the "standard error" I am using shorthand for the "standard deviation of the sampling distribution."

### Other sample statistics

In the example here, I have focused on the sample mean as the sample statistic of interest. However, the logic of the central limit theorem applies to several other important sample statistics of interest to us. In particular, the sampling distributions of:

1. means
2. proportions
3. mean differences
4. proportion differences
5. correlation coefficients

all become normal as the sample size increases. Thus, this normal distribution becomes critically important in making statistical inferences.

Note that the standard error formula $\sigma/\sqrt{n}$ only applies to the sampling distribution of sample means. Other sample statistics have different formulas for their standard errors, which I will introduce in the next section.

## What can we do with the sampling distribution?

Now that we know the sampling distribution of the sample mean should be normally distributed, what can we do with that information? The sampling distribution gives us information about how we would expect the sample means to be distributed. This seems like it should be helpful in figuring out whether we got a value close to the center or not. However, there is a catch. We don't know either $\mu$, the center of this distribution or $\sigma$ which we need to calculate its standard deviation. Thus, we know theoretically what it should look like but we have no concrete numbers to determine its actual shape.

We can fix the problem with not knowing $\sigma$ fairly easily. We don't know $\sigma$ but we do have an estimate of it in $s$, the sample standard deviation. In practice, we us this value to calculate an estimated standard error of $s/\sqrt{n}$. However, this substitution has consequences. Because we are using a sample statistic subject to random bias to estimate the standard error, this creates greater uncertainty in our estimation. I will come back to this issue in the next section.

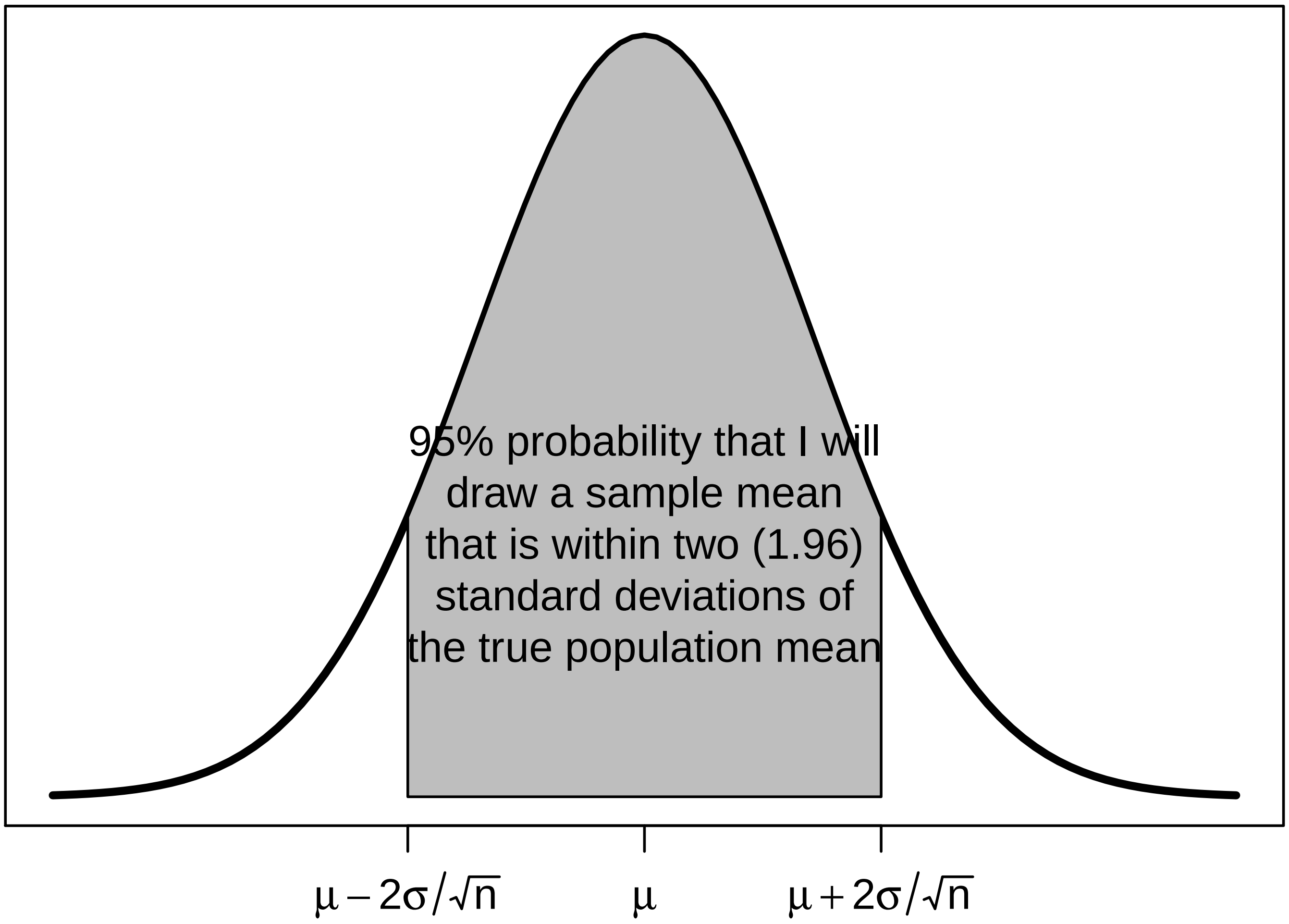
We still have the more fundamental problem that we don't know where the center of the sampling distribution should be. In order to make statistical inferences, we are going to employ two different methods that make use of what we do know about the sampling distribution:

1. **Confidence intervals**: Provide a range of values within which you feel confident that the true population mean resides.
2. **Hypothesis tests**: Play a game of make believe. If the true population mean was a given value, what is the probability that I would get the sample mean value that I actually did?

I will discuss these two different methods in the next two sections.

## Confidence Intervals

Remember from the previous section that 95% of all the possible sample means will be within 1.96 standard errors of the true population mean $\mu$. Therefore:



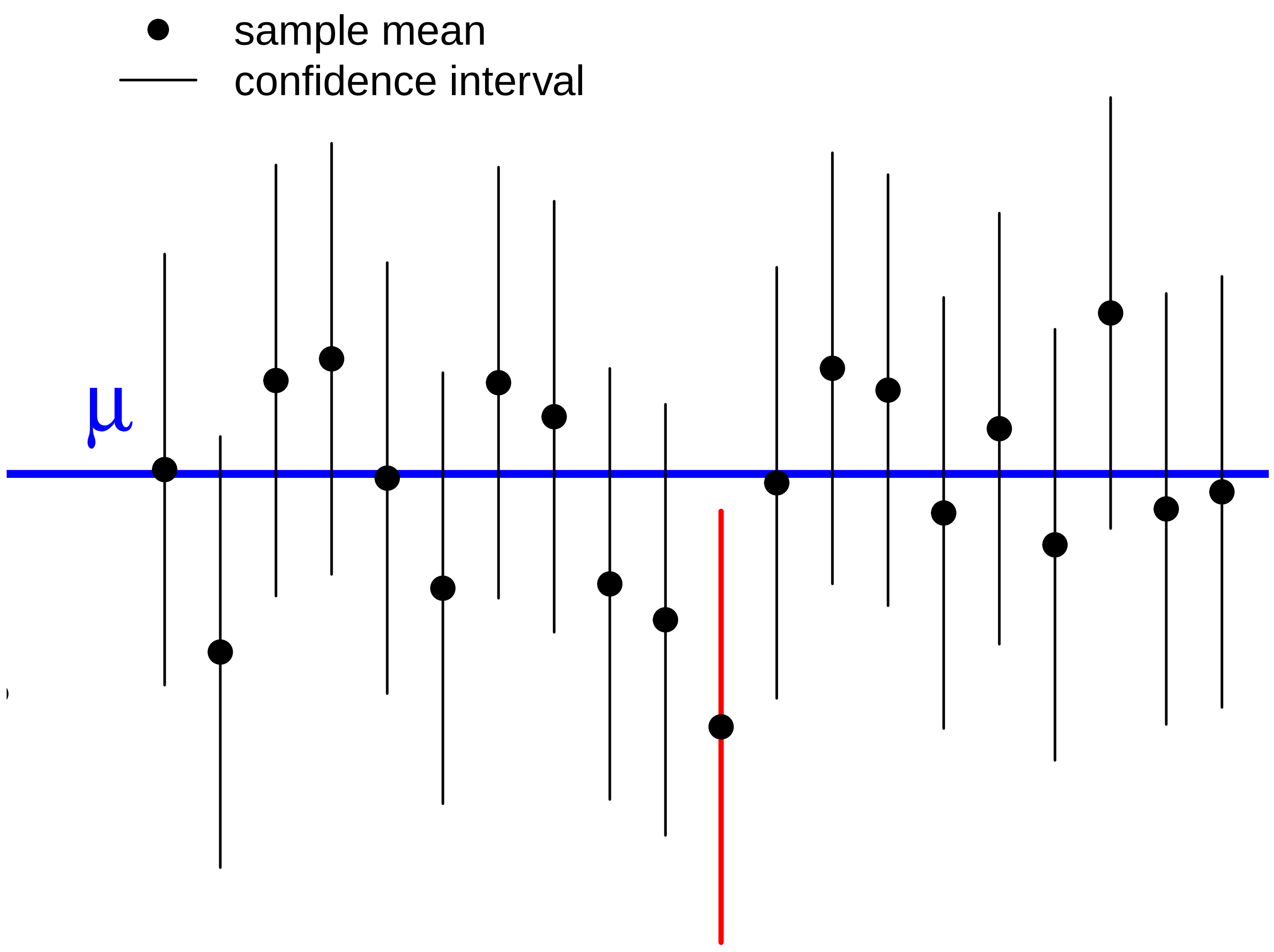
There is a 95% probability that we will draw a sample mean within 1.96 standard errors of the true population mean.

Lets say I were to construct the following interval for every possible sample:

$$\bar{x} \pm 1.96(\sigma/\sqrt{n})$$

t follows from the probability statement above that for 95% of all samples, this interval would contain the true population mean, $\mu$.

To see how this works graphically, imagine constructing this interval for twenty different samples from the same population.



The blue line gives the true population mean. The dots represent the sample means for each of the twenty samples. You can see that these sample means fluctuate around the true population mean due to random sampling error. The lines give the interval outlined above. In 19 out of the 20 cases, this interval contains the true population mean (as you can see by the fact that the interval crosses the blue line). The one sample where this is not true is shown in red. On average, 95% of samples will contain the true population mean in the interval, so 5% or 1 in 20 will not.

We refer to this interval as the **95% confidence interval**. Of course, in practice, we only construct one interval on the sample that we have. We use this interval to give a range of values that we feel "confident" will contain the true population mean.

## What do we mean by "confident?"

The term "confident" is a little ambiguous. Given my statements above, it might be tempting to interpret the 95% confidence interval to mean that there is a 95% probability that the true population mean is within the interval. This interpretation seems intuitive and straightforward, but that interpretation is incorrect according to the classic approach to inference. The problem here is subtle, but from the classical viewpoint, probability is an **objective** phenomenon that relates to the outcomes of future processes over the long run. From this viewpoint, we cannot express our **subjective** uncertainties about numbers in terms of probabilities.The population mean is a single static number. This leads us to a sort of yoda-like statement: The population mean is either in your interval or it is not. There is no probability.

This is why we use a more neutral term like "confidence." If we want to be long-winded about it, we might say that we are 95% confident because "in 95% of all the possible samples I could have drawn, the true population mean would be in the interval. I don't know if I have one of those 95% or the unlucky 5%, but nonetheless, there it is."

If this all seems a bit confusing, you are perfectly normal. As I said, this is the classic view of probability. Intuitively, people often think of uncertainty in probabilistic terms (e.g. what are the odds your team will win the game?). Many contemporary statisticians would in fact agree that it is perfectly okay to express subjective uncertainty as a probability. But, I still need to let you know that from the classic approach, interpreting your confidence interval as a probability statement is a no-no.

## Calculating the confidence interval for the sample mean

Okay, lets try calculating a confidence interval. Lets try this out for age in the politics dataset. The formula is:

$$\bar{x} \pm 1.96(\sigma/\sqrt{n})$$

Oh wait, we can't do it! We don't know the value of the population standard deviation $\sigma$. As I explained in the last section, we are going to have to do a little "fudging" here. Instead of $\sigma$, we can use our sample standard deviation $s$. However, doing so will have consequences. Here is our new formula:

$$\bar{x} \pm t * (s\sqrt{n})$$

As you can see, I have replaced the 1.96 with some number $t$, referred to as the **t-statistic**. Basically to adjust for the greater uncertainty in using a sample statistic in my calculation of the standard error, I need to increase the number here slightly from 1.96. How much I increase it will depend on the **degrees of freedom** which are given by the sample size minus one $(n-1)$. To figure out the correct t-statistic, I can use the `qt` command in *R*.

```
qt(0.975, nrow(politics)-1)
```

```
## [1] 1.960524
```

The first command to `qt` is the confidence you want. This is a little bit tricky because for a 95% confidence interval, we actually want to input 0.975. This is because we are basically asking for only the upper tail of that normal distribution shown at the beginning of this section. This area contains only 2.5% of the area outside, with the other 2.5% being in the lower tail. The second number is the degrees of freedom which equals $n-1$. In this case, we have such a large sample, that the t-statistic we need is very close to 1.96.

In smaller samples, using the t-statistic rather than 1.96 can make a bigger difference. Its not a proper sample, but lets take the case of the crime data. Here there are only 51 observations, so the t-statistic is:

```
qt(0.975, 51-1)
```

```
## [1] 2.008559
```

The difference from 1.96 is a little more noticeable.

Lets return to the politics data. We now have all the information we need to calculate the 95% confidence interval:

```
xbar <- mean(politics$age)
sdx <- sd(politics$age)
n <- nrow(politics)
se <- sdx/sqrt(n)
```

```
t <- qt(0.975,n-1)
xbar+t*se
```

```
## [1] 50.03165
```

```
xbar-t*se
```

```
## [1] 48.97307
```

We are 95% confident that the mean age among all US adults is between 49.01 and 49.87 years of age. As you can see, the large sample of nearly 6,000 respondents produces a very tight confidence interval.

## Calculating the confidence interval for other sample statistics

As noted in the previous section, the sampling distribution of other sample statistics such as proportions, mean differences, and regression slopes is also normally distributed in large enough samples. This means that we can use the same approach to construct confidence intervals for other sample statistics. The general form of the confidence interval is:

$$(samplestatistic) \pm t * (standarderror)$$

In order to do this for any of the above sample statistics, we only need to know how to calculate that sample statistic's standard error and the degrees of freedom used to look up the t-statistic for that sample statistic. Here is a useful cheat sheet of those formulas:

| Type | SE | df for $t$ |
|---|---|---|
| Mean | $s/\sqrt{n}$ | $n-1$ |
| Proportion | $\sqrt{\frac{\hat{p}*(1-\hat{p})}{n}}$ | $n-1$ |
| Mean Difference | $\sqrt{\frac{s_1^2}{n_1}+\frac{s_2^2}{n_2}}$ | $\min(n_1-1,n_2-1)$ |
| Proportion Difference | $\sqrt{\frac{\hat{p}_1*(1-\hat{p}_1)}{n_1}+\frac{\hat{p}_2*(1-\hat{p}_2)}{n_2}}$ | $\min(n_1-1,n_2-1)$ |
| Correlation Coefficient | $\sqrt{\frac{1-r^2}{n-2}}$ | $n-2$ |

I know some of that math might look intimidating but I will go through an example of each case below to show you how it works for each case.

## Example with proportions

As an example, lets use the proportion of respondents who do not believe in anthropogenic climate change. In our politics sample, we get:

```
table(politics$globalwarm)
```

```
##
##   No  Yes
## 1180 3058
```

```
n <- 1181+3057
p_hat <- 1181/n
n
```

```
## [1] 4238
```

```
p_hat
```

```
## [1] 0.2786692
```

About 27.9% of the respondents in our sample are climate change deniers. What can we conclude about the proportion in the US population? First, lets figure out the t-statistic. We use the same $n - 1$ for degrees of freedom:

```
t_stat <- qt(0.975, n-1)
t_stat
```

```
## [1] 1.960524
```

Our sample is large enough that we are basically using 1.96. Now we need to calculate the standard error. The formula from above is:

$$\sqrt{\frac{\hat{p} * (1 - \hat{p})}{n}}$$

The term *hatp* is the standard way to represent the sample proportion, which in this case is 0.279. So, our formula is:

$$\sqrt{\frac{0.279 * (1 - 0.279)}{4238}}$$

We can calculate this in *R*:

```
se <- sqrt(p_hat*(1-p_hat)/n)
se
```

```
## [1] 0.006887018
```

We now have all the pieces to construct the confidence interval:

```
p_hat+t_stat*se
```

```
## [1] 0.2921713
```

```
p_hat-t_stat*se
```

```
## [1] 0.265167
```

We are 95% confident that the true percentage of climate change deniers in the the US population is between 26.5% and 29.2%.

## Example with mean differences

using our Add health data, what is the mean difference in popularity (number of friend nominations) between frequent smokers and those who do not smoke frequently?

```
tab <- tapply(addhealth$indegree, addhealth$smoker, mean)
tab
```

```
## Non-smoker     Smoker
##   4.509636   4.782148
```

```
mean_diff <- 4.782148 - 4.509636
mean_diff
```

```
## [1] 0.272512
```

In our sample data, frequent smokers had 0.273 more friends on average than those who did not smoke frequently. What is the confidence interval for that value in the population? We start by calculating the t-statitic for this confidence interval. We use the size of the smaller group minus one for the degrees of freedom.

```
table(addhealth$smoker)
```

```
##
## Non-smoker     Smoker
##       3736        661
```

```
n1 <- 3736
n2 <- 661
t_stat <- qt(.975, n2-1)
t_stat
```

```
## [1] 1.963565
```

The value is pretty close to 1.96 but a little bigger. Now we need to calculate the standard error. The formula is:

$$\sqrt{\frac{s_1^2}{n_1}+\frac{s_2^2}{n_2}}$$

We already have $n_1$ and $n_2$, so we just need to get the standard deviation of friend nominations for the two groups to get $s_1$ and $s_2$. We can do this with another tapply command, but changing from `mean` to `sd` in the third argument.

```
tapply(addhealth$indegree, addhealth$smoker, sd)
```

```
## Non-smoker     Smoker
##   3.658650   3.869584
```

```
s1 <- 3.65865
s2 <- 3.869584
se <- sqrt(s1^2/n1+s2^2/n2)
se
```

```
## [1] 0.1619752
```

Now we have all the pieces to put together the confidence interval:

```
mean_diff - t_stat*se
```

```
## [1] -0.04553685
```

```
mean_diff + t_stat*se
```

```
## [1] 0.5905609
```

We are 95% confident that in the population of US adolescents, those who smoke frequently have between 0.05 fewer to 0.59 more friend nominations, on average, than those who do not smoke frequently. Note that because our confidence interval contains both negative and positive value, we cannot be confident about whether smoking is truly associated with having more or less friends. The direction of the relationship between the two variables is uncertain.

## Example with proportion differences

Lets continue to use the Add Health data. Do we observe a gender difference in smoking behavior in our sample?

```
prop.table(table(addhealth$smoker, addhealth$sex), 2)
```

```
##
##                 Female      Male
##   Non-smoker 0.8543563 0.8444976
##   Smoker     0.1456437 0.1555024
```

```
p_hat_f <- 0.1456
p_hat_m <- 0.1555
p_hat_diff <- p_hat_m-p_hat_f
p_hat_diff
```

```
## [1] 0.0099
```

In our sample, about 14.6% of girls were frequent smokers and about 15.6% of boys were frequent smokers. The percentage of boys who smoke is about 1% higher than the percentage of girls. Do we think this moderate difference in the sample is true in the population?

Lets start again by calculating the appropriate t-statistic for our confidence interval. We use the same procedure as for mean differences above, choosing the size of the smaller group for the degrees of freedom. However, its important to note that our groups are now boys and girls, not smokers and non-smokers.

```
table(addhealth$sex)
```

```
##
##  Female    Male
##    2307    2090
```

```
n_f <- 2307
n_m <- 2090
t_stat <- qt(.975, n_m-1)
t_stat
```

```
## [1] 1.9611
```

Now we can calculate the standard error. The formula is:

$$\sqrt{\frac{\hat{p}_1 * (1-\hat{p}_1)}{n_1} + \frac{\hat{p}_2 * (1-\hat{p}_2)}{n_2}}$$

That looks like a lot, but we just have to focus on pluggin in the right values in R:

```
se <- sqrt((p_hat_f*(1-p_hat_f)/n_f)+(p_hat_m*(1-p_hat_m)/n_m))
se
```

```
## [1] 0.01080535
```

Now we have all the parts to calculate the confidence interval:

```
p_hat_diff - t_stat*se
```

```
## [1] -0.01129037
```

```
p_hat_diff + t_stat*se
```

```
## [1] 0.03109037
```

We are 95% confident that in the population of US adolescents, between 1.1% fewer to 3.1% more boys smoke frequently than girls. As above, because our confidence interval includes both negative and positive values, we are not very confident at all about whether boys or girls smoke more frequently.

### Example with correlation coefficient

Lets stick with the Add Health data. What is the correlation between GPA and the number of friend nominations that a student receives?

```
r <- cor(addhealth$pseudoGPA, addhealth$indegree)
r
```

```
## [1] 0.1685929
```

In our sample, there is a moderately positive correlation between a student's GPA and the number of friend nominations that a student receives. What is our confidence interval for the population?

For the t-statistic, we use $n - 2$ for the degrees of freedom:

```
n <- nrow(addhealth)
t_stat <- qt(.975, n-2)
```

For the standard error, the formula is:

$$\sqrt{\frac{1-r^2}{n-2}}$$

This is straightforward to calculate in R:

```
se <- sqrt((1-r^2)/(n-2))
se
```

```
## [1] 0.01486822
```

Now we have all the parts we need to calculate the confidence interval:

```
r - t_stat*se
```

```
## [1] 0.1394437
```

```
r + t_stat*se
```

```
## [1] 0.1977421
```

We are 95% confident that the true correlation coefficient between GPA and friend nominations in the population of US adolescents is between 0.139 and 0.198. While there is some difference in the strength of that relationship, we are pretty confident that the correlation is moderately positive.

---

## Hypothesis Tests

In social scientific practice, hypothesis testing is far more common than confidence intervals as a technique of statistical inference. Both techniques are fundamentally derived from the sampling distribution and produce similar results, but the methodology and interpretation of results is very different.

In hypothesis testing, we play a game of make believe. Remember that the fundamental issue we are trying to work around is that we don't know the value of the true population parameter and thus we don't know where the center is for the sampling distribution of the sample statistic. In hypothesis testing, we work around this issue by boldly asserting what we think the true population parameter. We then test whether the data that we got are reasonably consistent with that assertion.

## Example: Coke winners

Lets take a fairly straightforward example. Coca-Cola used to run promotions where a certain percentage of bottle caps were claimed to win you another free coke. In one such promotion, when I was in graduate school, Coca-Cola ran a promotion where they claimed that 1 in 12 bottles were winners. If this is true, then 8.3% (1/12=0.083) of all the coke bottles in every grocery store and mini mart should be winners.

Being a grad student who needed to stay up late writing a dissertation fueled by caffeine and "sugar," I use to drink quite a few Cokes. After only receiving a few winners after numerous attempts, I began to get suspicious of the claim. I started collecting bottle caps to see if I could statistically find evidence of fraudulent behavior.

For the sake of this exercise, lets say I collected 100 coke bottle caps (I never got this high in practice, but its a nice round number) and that I only got five winners. My winning percentage is 5% which is lower than Coke's claim of 8.3%.

The critical question is whether it is likely or unlikely that I would get a winning percentage this different from the claim in a sample of 100 bottle caps. That is what a hypothesis test is all about. We are asking whether the data that we got are likely under some assumption about the true parameter. If they are unlikely, then we reject that assumption. if they are not unlikely, then we do not reject the assumption.

We call that assumption the **null hypothesis**, $H_0$. The null hypothesis is a statement about what we think the true value of the parameter is. The null hypothesis is our "working assumption" until we can be proven to be wrong. In this case, the parameter of interest is the true proportion of winners among the population of all Coke bottles in the US. Coke claims that this proportion is 0.083, so this is my null hypothesis. In mathematical terms, we write:

$$H_0 : \rho = 0.083$$

I use the greek letter $\rho$ as a symbol for the population proportion. I will use $\hat{p}$ to represent the sample proportion in my sample, which is 0.05.

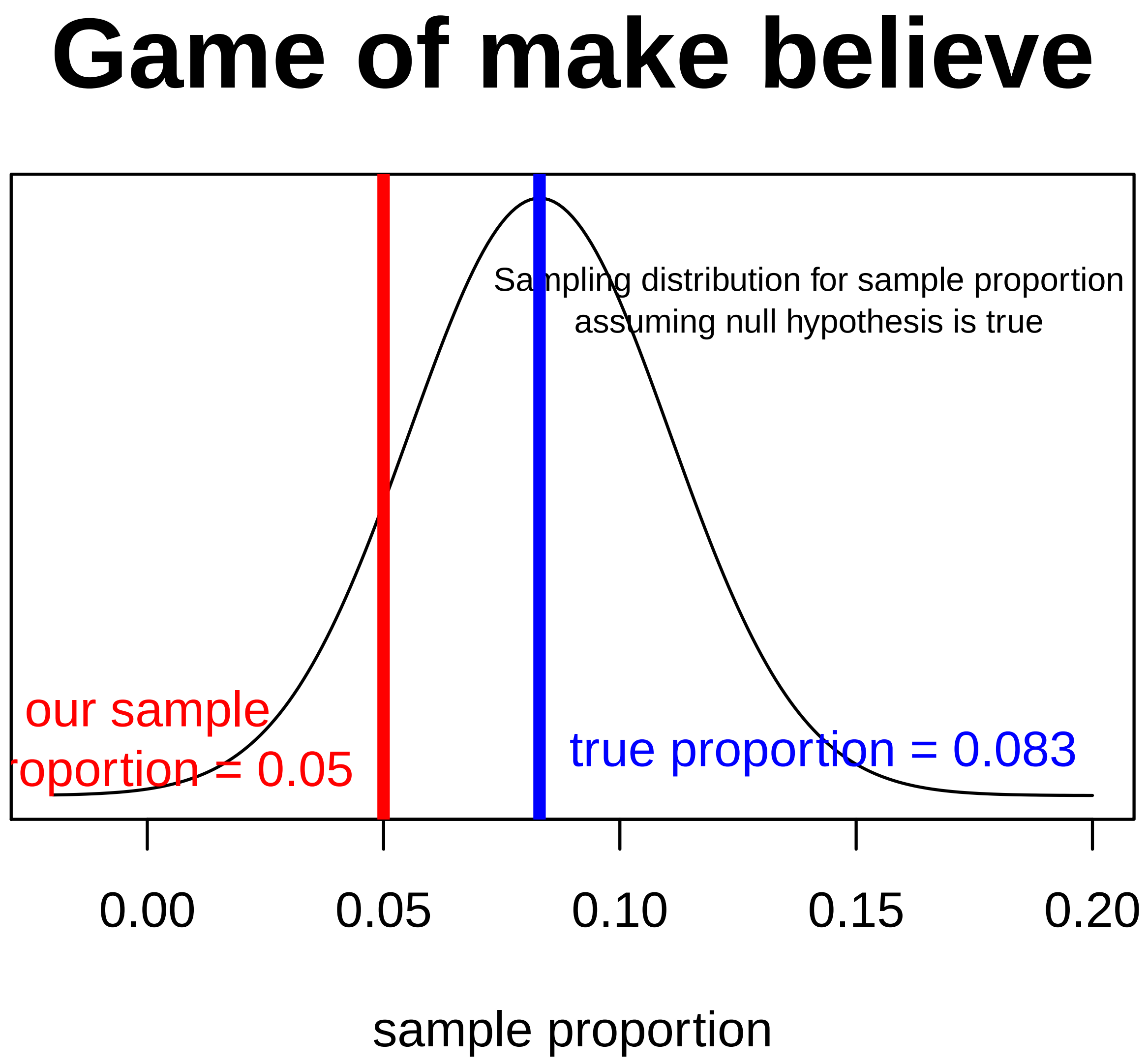
Some standard statistical textbooks will also claim that there is an "alternative hypothesis." That alternative hypothesis is specified as "anything but the null hypothesis." In my opinion, this is incorrect because vague statements about "anything else" do not constitute an actual hypothesis about the data. We are testing only whether the data are consistent with the null hypothesis. No other hypothesis is relevant.

We got a sample proportion of 0.05 on a sample of 100. **Assuming the null hypothesis is true**, what would the sampling distribution look like from which I pulled my 0.05? Note the part in bold above. We are now playing our game of make believe.

We know that on a sample of 100, the sample proportion should be normally distributed. It should also be centered on the true population proportion. Because we are assuming the null hypothesis is true, it should be centered on the value of 0.083. The standard error of this sampling distribution is given by:

$$\sqrt{\frac{0.083 * (1 - 0.083)}{100}} = 0.028$$

Therefore, we should have a sampling distribution that looks like:



The blue line shows the true population proportion assumed by the null hypothesis. The red line shows my actual sample proportion. The **key question of hypothesis testing is whether the observed data (or more extreme data) are reasonably likely under the assumption of the null hypothesis**. Practically speaking, I want to know how far my sample proportion is from the true proportion and whether this distance is far enough to consider it unlikely.

To calculate how far away I am on some standard scale, I divide the distance by the standard error of the sampling distribution to calculate how many standard errors my sample proportion is below the population parameter (assuming the null hypothesis is true).

$$\frac{0.05 - 0.083}{0.028} = \frac{-0.033}{0.028} = -1.18$$

My sample proportion is 1.18 standard errors below the center of the sampling distribution. Is this an unlikely distance? To figure this out, we need to calculate the area in the lower tail of the sampling distribution past my red line. This number will tell us the proportion of all sample proportions that would be 0.05 or lower, assuming the null hypothesis is true. This standardized measure of how far is sometimes called the **test statistic** for a given hypothesis test.

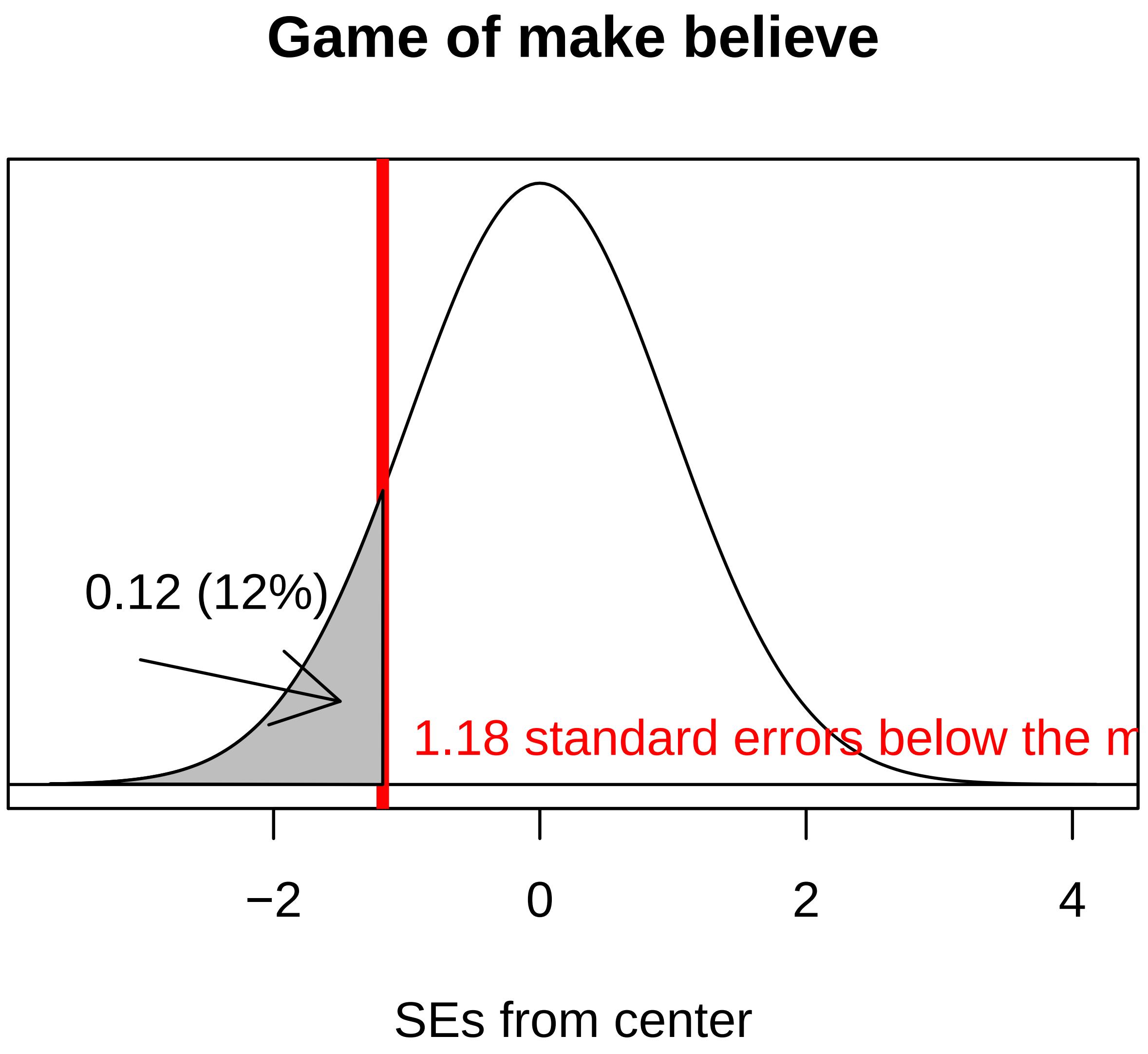
Calculating this area is not a trivial exercise, but *R* provides a straightforward command called `pt` which is somewhat similar to the `qt` command above. We just need to feed in how many standard errors our estimate is away from the center (-1.18) and the degrees of freedom. These degrees of freedom are identical to the ones used in confidence intervals (in this case, $n - 1$, so 99).

```
pt(-1.18, 99)
```

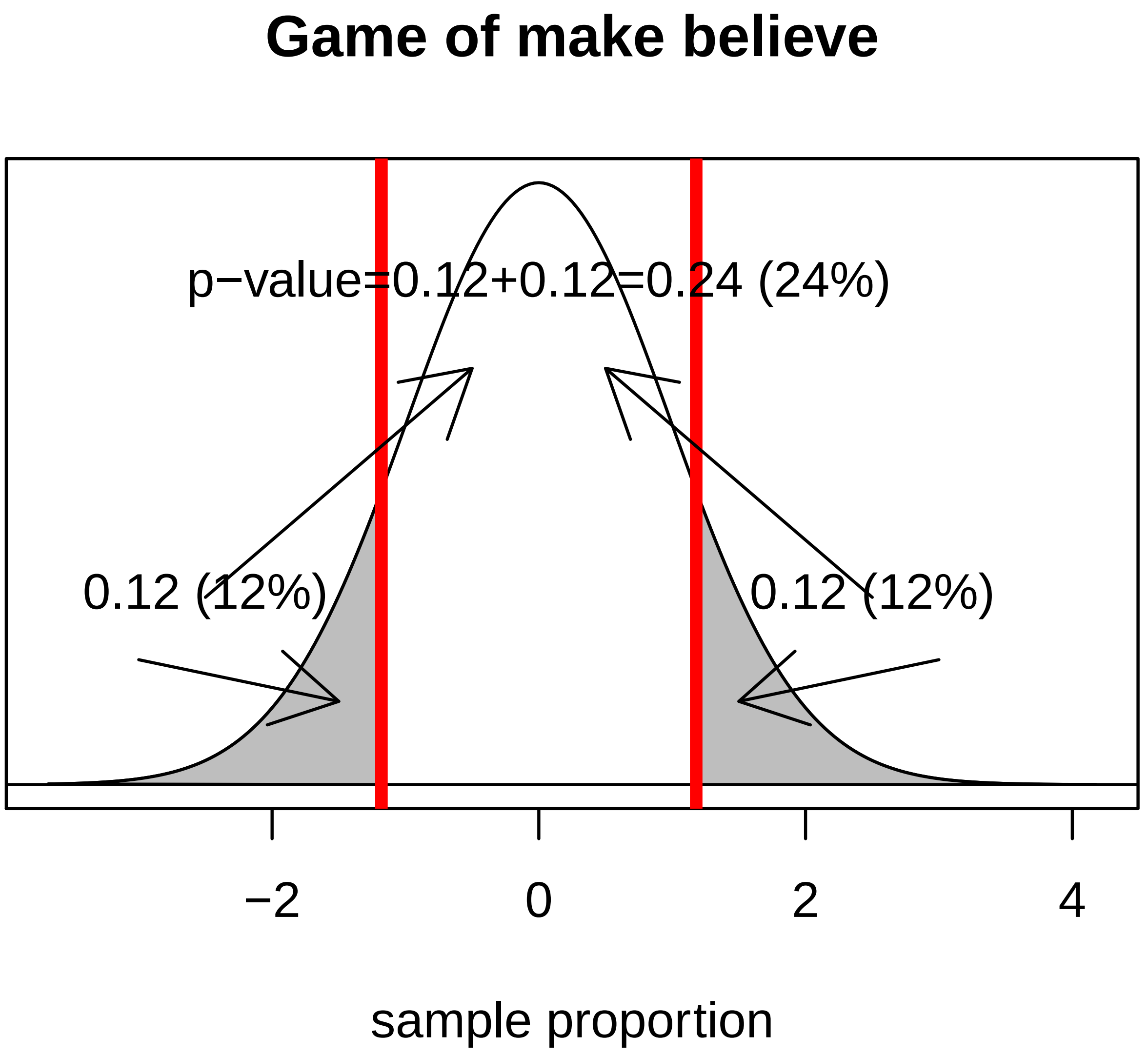
```
## [1] 0.1204139
```

There is one catch with this command. It always gives you the area in the lower tail, so if your sample statistic is above the center, you should still put in a negative value in the first command. We will see an example of this below.

Our output indicates that 12% of all samples would produce a sample proportion of 0.05 or less when the true population proportion is 0.083. Graphically it looks like this:



The grey area is the area in the lower tail. It would seem that we are almost ready to conclude our hypothesis test. However, there is a catch and its a tricky one. Remember that I was interested in the probability of getting a sample proportion this far or farther from the true population proportion. This is not the same as getting a sample proportion this low or lower. I need to consider the possibility that I would have been equally suspicious if I had got a sample proportion much higher than 8.3%. In mathematically terms, that means I need to take the area in the upper tail as well, where I am .033 above the true population proportion. This is called a **two-tailed test**. Luckily, because the normal distribution is symmetric, this area will be identical to the area in the lower tail and so I can just double this percent.



Assuming the null hypothesis is true, there is a 24% chance of getting a sample proportion as far from the true population mean or farther, just by random chance. We call this probability the **p-value**. The p-value is the ultimate goal of the hypothesis test. All hypothesis tests produce a p-value and it is the p-value that we will use to make a decision about our test.

What should that decision be? We have only two choices. If the p-value is low enough, then it is unlikely that we would have gotten this data or data more extreme, assuming the null hypothesis is true. Therefore, we **reject the null hypothesis**. If the p-value is not low enough, then it is reasonable that we would have gotten this data or data more extreme, assuming the null hypothesis is true. Therefore, we **fail to reject the null hypothesis**. Note that we NEVER accept or prove the null hypothesis. It is already our working assumption, so the best we can do for it is to fail to reject it and thus continue to use it as our working assumption.

How low does the p-value need to be in order to reject it? There is no right answer here, because this is a subjective question. However, there is a generally agreed upon practice in the social sciences that we reject the null hypothesis when the p-value is at or below 0.05 (5%). Note that while there is general consensus around this number, it is an arbitrary cutpoint. The practical difference between a p-value of 0.049 and 0.051 is negligible, but under this arbitrary standard, we would make different decisions in each case. I would rather that you just learn to think about what the p-value represents and reach your own decision.

No reasonable scientist, however, would reject the null hypothesis with a p-value of 24% as we have in our

Coke case. Nearly 1 in 4 samples of size 100 would produce a sample proportion this different from the assumed true proportion of 8.3% just by random chance. I therefore do not have sufficient evidence to reject the null hypothesis that Coke is telling the truth. Note that I have not proved that Coke is telling the truth. I have only failed to produce evidence that they are lying.

## The general procedure of hypothesis testing

The general procedure of hypothesis testing is as follows:

1. State a **null hypothesis**. This null hypothesis is a claim about the true value of an unknown parameter.
2. Calculate a **test statistic** that tells you how far your sample statistic is from the center of the sampling distribution, *assuming the null hypothesis is true.* For our purposes, this test statistic will always be the number of standard errors above or below the true population parameter, assuming the null hypothesis is true.
3. Calculate the **p-value** for the test statistic. The p-value is the probability of getting a sample statistic this far or farther (in absolute value) from the true population parameter, *assuming the null hypothesis is true.*
4. If the p-value is below some threshold (typically 0.05), **reject the null hypothesis**. Otherwise, **fail to reject the null hypothesis**.

### Interpreting p-values correctly

P-values are widely misunderstood in practice. Studies have been done of practicing researchers across different disciplines where these researchers were asked to interpret a p-value from a multiple choice question and the majority get it wrong. Therefore, don't feel bad if you are having trouble understanding a p-value. You are in good company! Nonetheless, proper interpretation of a p-value is critically important for our understanding of what a hypothesis test does.

The reason many people get the interpretation of p-values wrong is that they want the p-value to express the probability of a hypothesis being correct or incorrect. People routinely misinterpret the p-value as a statement about the probability of the null hypothesis being correct. **The p-value is NOT a statement about the probability of a hypothesis being correct or incorrect.** For the same reason that we cannot call a confidence interval a probability statement, the classical approach dictates that we cannot characterize our subjective uncertainty about whether hypotheses are true or not by a probability statement. The hypothesis is either correct or it is not. There is no probability.

Correctly interpreted, **the p-value is a probability statement about the data, not about the hypothesis**. Specifically, we are asking what the probability is of observing data this extreme or more extreme, assuming the null hypothesis is true. We are not making a probability statement about hypotheses. Rather we are assuming a hypothesis and then asking about the probability of the data. This difference may seem subtle, but it is in fact quite substantial in interpretation.

The reason why everyone (including you and me) struggles with this is that our brains want it to be the other way around. Ultimately by rejecting or failing to reject we are making statements about whether we believe the hypothesis or not, but we are not doing that directly by a probability statement about the hypothesis but rather a probability statement about the likelihood of the data given the hypothesis.

## Hypothesis tests of relationships

The hypothesis tests that we care the most about in the sciences are hypothesis tests about relationships between variables. We want to know whether the association we are observing in the sample is true in the population. In all of these cases, our null hypothesis is that there is no association, and we want to know whether the association we observe in the sample is strong enough to reject this null hypothesis of no association. We can do hypothesis tests of this nature for both mean differences and regression slopes.

**Example: mean differences**

Lets look at differences in mean income (measured in \$1000) by religion in the politics dataset.

```
tapply(politics$income, politics$relig, mean)
```

```
##    Mainline Protestant Evangelical Protestant               Catholic
##               81.83439               58.32606               77.53498
##                 Jewish          Non-religious                  Other
##              120.92958               88.62963               60.75311
```

I want to look at the difference between Roman Catholics and "Other Religions." The mean difference here is:

$$63.35337 - 61.74611 = 1.607$$

Roman Catholics make \$1,607 more than members of other religions, in my sample.

Let me set up a hypothesis test where the null hypothesis is that Roman Catholics and members of other religions have the same income, or in other words, the mean difference in income is zero:

$$H_0 : \mu_c - \mu_o = 0$$

Where $\mu_c$ is the population mean income of Catholics and $\mu_o$ is the population mean income of members of other religions. In order to figure out how far my sample mean difference of 0.548 is from 0, I need to find the standard error of the mean difference. The formula for this number is:

$$\sqrt{\frac{s_c^2}{n_c} + \frac{s_o^2}{n_o}}$$

I can calculate this in *R*:

```
tapply(politics$income, politics$relig, sd)
```

```
##    Mainline Protestant Evangelical Protestant               Catholic
##               62.90763               51.23396               66.59281
##                 Jewish          Non-religious                  Other
##               89.84166               71.46392               56.37886
```

```
table(politics$relig)
```

```
##
##    Mainline Protestant Evangelical Protestant               Catholic
##                    785                    917                   1015
##                 Jewish          Non-religious                  Other
##                     71                    567                    883
```

```
sqrt(55.49944^2/1393+52.78854^2/257)
```

```
## [1] 3.613047
```

```
1.607/3.613
```

```
## [1] 0.4447827
```

The t-statistic of 0.44 here is not very large. I am only 0.44 standard errors above 0 on the sampling distribution, assuming the null hypothesis is true. Lets go ahead and calculate the p-value for this t-statistic. Remember that I need to put in the negative version of this number to the `pt` command. I also need to use the smaller of the two sample sizes for my degrees of freedom:

```
2*pt(-0.44, 257-1)
```

```
## [1] 0.6603084
```

In a sample of this size, there is an 66% chance of observing a mean income difference of $1,607 or more between Catholics and members of other religions, just by sampling error, assuming that there is no difference in income in the population. Therefore, I **fail to reject the null hypothesis** that Catholics and members of other religions make the same income.

### Example of proportion differences

Lets look at the difference in smoking behavior between white and black students in the Add Health data. Our null hypothesis is:

$$H_0 : \rho_w - \rho_c = 0$$

In simple terms, our null hypothesis is that the same proportion of white and black adolescents smoke frequently. Lets look at the actual numbers from our sample:

```
prop.table(table(addhealth$smoker, addhealth$race),2)
```

```
##
##                    White  Black/African American     Latino
##   Non-smoker 0.79673872              0.94851658 0.89250000
##   Smoker     0.20326128              0.05148342 0.10750000
##
##              Asian/Pacific Islander      Other
##   Non-smoker             0.90062112 0.92592593
##   Smoker                 0.09937888 0.07407407
##
##              American Indian/Native American
##   Non-smoker                      0.80769231
##   Smoker                          0.19230769
```

```
p_w <- 0.203
p_b <- 0.051
p_diff <- p_w-p_b
p_diff
```

```
## [1] 0.152
```

About 20.3% of white students smoked frequently, compared to only 5.1% of black students. The difference in proportion is a large 15.2% in the sample. This would seem to contradict our null hypothesis. However, we need to confirm that a difference this large in a sample of our size is unlikely to happen by random chance. To do that we need to calculate the standard error, just as we learned to do it for proportion differences in the confidence interval section:

```
table(addhealth$race)
```

```
##
##                           White          Black/African American
##                            2637                            1146
##                          Latino          Asian/Pacific Islander
##                             400                             161
##                           Other American Indian/Native American
##                              27                              26
```

```r
n_w <- 2637
n_b <- 1146
se <- sqrt((p_w*(1-p_w)/n_w)+(p_b*(1-p_b)/n_b))
se
```

```
## [1] 0.01017778
```

Now many standard errors is our observed difference in proportion from zero?

```r
t_stat <- p_diff/se
t_stat
```

```
## [1] 14.9345
```

Wow, thats a lot. We can be pretty confident already without the final step of the p-value, but lets calculate it anyway. Remember ot always take the negative version of the t-statistic you calculated:

```r
2*pt(-14.9345, n_b-1)
```

```
## [1] 3.26654e-46
```

The p-value is astronomically small. In a sample of this size, the probability of observing a difference in the proportion frequent smokers between whites and blacks of 15.2% or larger if there is no difference in the population is less than 0.0000001%. Therefore, I **reject the null hypothesis** and conclude that white students are more likely to be frequent smokers than are black students.

### Example of correlation coefficient

Lets look at the correlation between the parental income of a student and the number of friend nominations they recieve. Our null hypothesis will be that there is no relationship between parental income and student popularity in the population of US adolescents. Lets look at the data in our sample:

```r
r <- cor(addhealth$parentinc, addhealth$indegree)
r
```

```
## [1] 0.1266054
```

In the sample we observe a moderately positive correlation between a student's parental income and the number of friend nominations they receive. How confident are we that we wouldn't observe such a large correlation coefficient in our sample by random chance if the null hypothesis is true?

First, we need to calculate the standard error:

```r
n <- nrow(addhealth)
se <- sqrt((1-r^2)/(n-2))
se
```

```
## [1] 0.01496276
```

How many standard errors are we away from the assumption of zero correlation?

```r
r/se
```

```
## [1] 8.461366
```

What is the probablity of being that far away from zero for a sample of this size?

```r
2*pt(-8.461366, n-2)
```

```
## [1] 3.554616e-17
```

The probability is very small. In a sample of this size, the probability is less than 0.00000001% of observing a correlation coefficient between parental income and friend nominations received of an absolute magnitude of 0.127 or higher when the true correlation is zero in the population. Therefore, we **reject the null hypothesis** and conclude that there is a positive correlation between parental income and populatity among US adolescents.

### Statistical Significance

When a researcher is able to reject the null hypothesis of "no association," the result is said to be **statistically significant**. This is a somewhat unfortunate phrase that is sometimes loosely interpreted to indicate that the result is "important" in some vague scientific sense.

In practice, it is important to distinguish between substantive and statistical significance. In very large datasets, standard errors will be very small, and thus it is possible to observe associations that are very small in substantive size that are nonetheless statistically significant. On the flip side, in small datasets, standard errors will often be large, and thus it is possible to observe associations that are very large in substantive size but not statistically significant.

It is important to remember that "statistical significance" is a reference to statistical inference and not a direct measure of the actual magnitude of an association. I prefer the term "statistically distinguishable" to "statistically significant" because it more clearly indicates what is going on. In the previous example, we found that the income differences in our sample between Catholics and members of other religions are not statistically distinguishable from zero. We also found that the negative association in our sample between age and sexual frequency was statistically distinguishable from zero. Establishing whether an association is worthwhile in its substantive effect is a totally different exercise from establishing whether it is statistically distinguishable from zero.

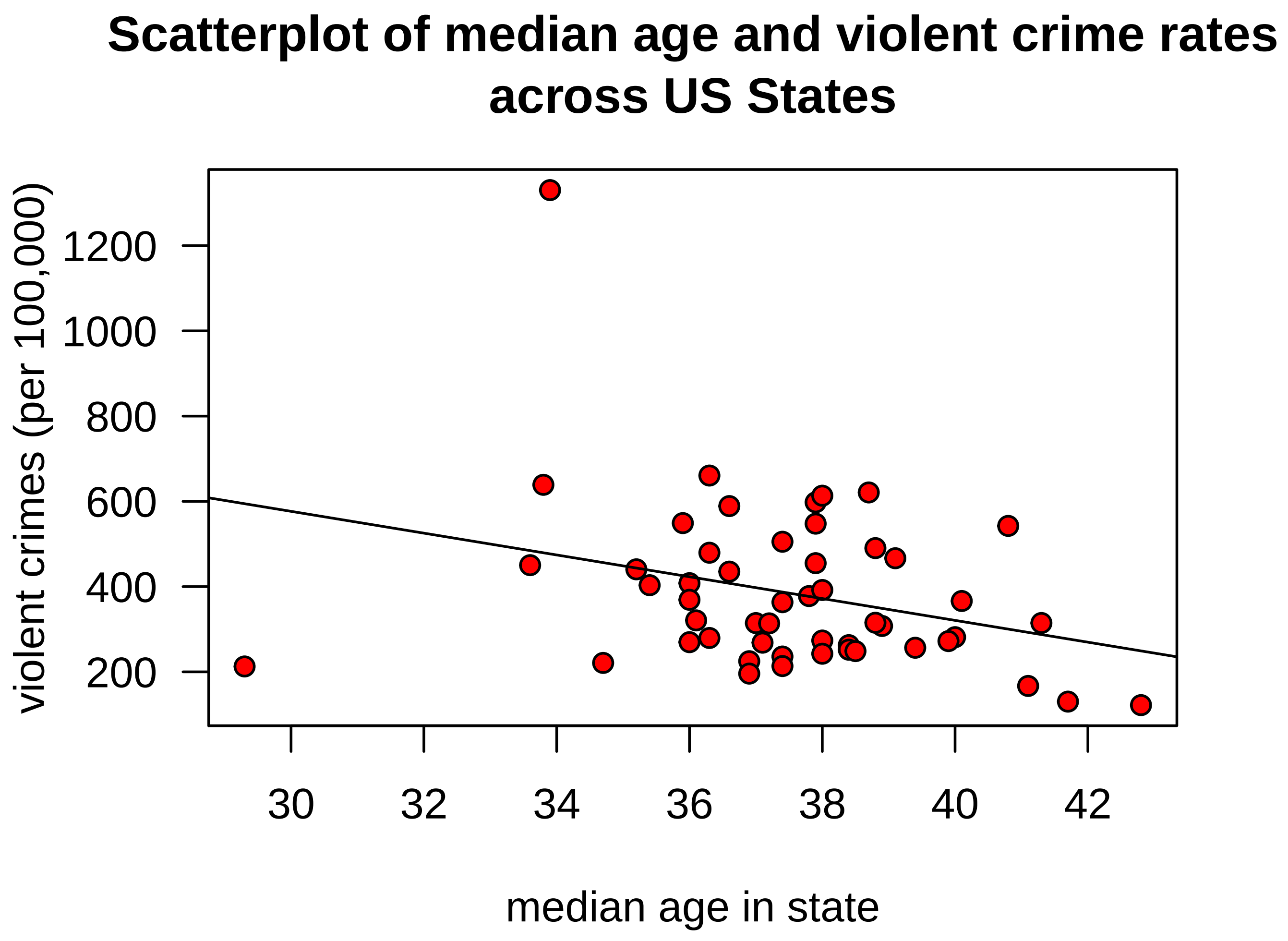
It is also important to remember that a statistically insignificant finding is not evidence of no relationship because we never accept the null hypothesis. We have just failed to find sufficient evidence of a relationship. No evidence of an association is not evidence of no association.

Chapter 5

# Building Models

## The OLS Regression Line

Lets take another look at the scatterplot we constructed in the previous section that showed the relationship between median age and violent crime rates:



Notice that I now have a line plotted through those points. When you were trying to determine the direction of the relationship many of you were probably imagining a line going through the points already. Of course, if we just tried to "eyeball" the best line, we would get many different results. The line I have graphed above, however, is the best fitting line, at least according to some reasonable criteria. It is the best-fitting line because it minimizes the total distance from all of the points collectively to the line. This line is called the **ordinary least squares regression line** ( or OLS regression line, for short) and it is the basis for the most used statistical technique in the social sciences. This fairly simply concept of fitting the best line to a set of points on a scatterplot is the workhorse of social science statistics.

## The Formula for a Line

Remember the basic formula for a line in two-dimensional space? In algebra, you probably learned something like this:

$$y = a + bx$$

The two numbers that relate $x$ to $y$ are $a$ and $b$. The number $a$ gives the **y-intercept**. This is the value of $y$ when $x$ is zero. The number $b$ gives the **slope** of the line, sometimes referred to as the "rise over the run." The slope indicates the change in $y$ for a one-unit increase in $x$.

The OLS regression line above also has a slope and a y-intercept. But we use a slightly different syntax to describe this line than the equation above. The equation for an OLS regression line is:

$$\hat{y}_i = b_0 + b_1 x_i$$

On the right-hand side, we have a linear equation (or function) into which we feed a particular value of $x$ ($x_i$). On the left-hand side, we get not the actual value of $y$ for the $i$th observation, but rather a **predicted value** of $y$. The little symbol above the $y$ is called a "hat" and it indicates the "predicted value of $y$." We use this terminology to distinguish the actual value of $y$ ($y_i$) from the value predicted by the OLS regression line ($\hat{y}_i$).

The y-intercept is given by the symbol $b_0$. The y-intercept tells us the predicted value of $y$ when $x$ is zero. The slope is given by the symbol $b_1$. The slope tells us the predicted change in $y$ for a one-unit increase in $x$. In practice, the slope is the more important number because it tells us about the association between $x$ and $y$. Unlike the correlation coefficient, this measure of association is not unitless. We get an estimate of how much we expect $y$ to change in terms of its units for a one-unit increase in $x$.

In the case above, the slope is -25.6 and the y-intercept is 1343.9. We could therefore write the equation like so:

$$\hat{crimerate}_i = 1343.9 - 25.6(medianage_i)$$

We would interpret our numbers as follows. The model predicts that a one-year increase in age within a state is associated with 25.6 fewer violent crimes per 100,000 population, on average. The model predicts that in a state where the median age is zero, the violent crime rate will be 1343.9 crimes per 100,000 population, on average.

There is a lot to digest in these interpretations and I want to return to them in detail, but first I want to address a more basic question. How did I know that these are the right numbers for the best-fitting line?

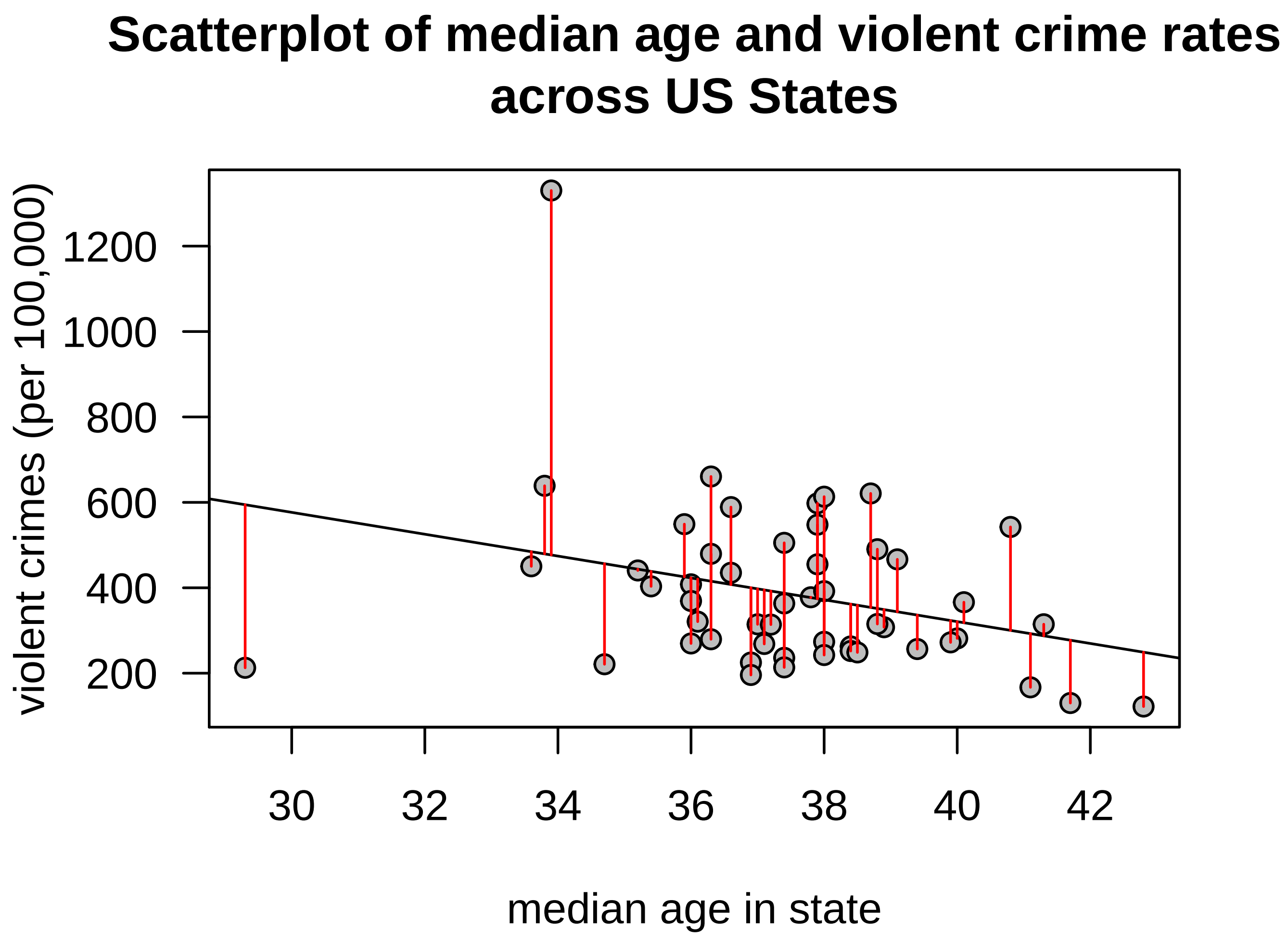
## Calculating the Best-Fitting Line

The slope and intercept of the OLS regression line are determined based on addressing one simple criteria: minimize the distance between the actual points and the line. More formally, we choose the slope and intercept that produce the **minimum sum of squared residuals (SSR)**.

A **residual** is the vertical distance between an actual value of y for an observation and its predicted value:

$$residual_i = y_i - \hat{y}_i$$

These residuals are also sometimes called **error terms**, because the larger they are in absolute value, the worse is our prediction. Take a look at the same scatterplot as above but this time with the residuals drawn in red:



Unless the points all fall along an exact straight line, there is no way for me to eliminate these residuals altogether, but some lines will produce higher residuals than others. What I am aiming to do is minimize the sum of squared residuals which is given by:

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

I square each residual and then sum them up. By squaring, I eliminate the problem of some residuals being negative and some positive.

To see how this all works try this exercise where you can experiment with trying to find the best-fitting line to minimize the sum of squared residuals for another scatterplot.

Fortunately, we don't have to figure out the best slope and intercept by trial and error, as in the exercise above. There are relatively straightforward formulas for calculating the slope and intercept. They are:

$$b_1 = r\frac{s_y}{s_x}$$

$$b_0 = \bar{y} - b_1 * \bar{x}$$

The $r$ here is the correlation coefficient. The slope is really just a re-scaled version of the correlation coefficient. We can calculate this with the example above like so:

```
slope <- cor(crimes$MedianAge, crimes$Violent)*sd(crimes$Violent)/sd(crimes$MedianAge)
slope
```

```
## [1] -25.5795
```

I can then use that slope value to get the y-intercept:

```
mean(crimes$Violent)-slope*mean(crimes$MedianAge)
```

```
## [1] 1343.936
```

## Using the `lm` command to calculate OLS regression lines in *R*

We could just use the given formulas to calculate the slope and intercept in *R*, as I showed above. However, the `lm` command will become particularly useful later in the term when we extend this basic OLS regression line to more advanced techniques.

In order to run the `lm` command, you need to input a formula. The structure of this formula looks like "dependent~independent" where "dependent" and "independent" should be replaced by your specific variables. The tilde (~) sign indicates the relationship. So, if we wanted to use `lm` to calculate the OLS regression line we just looked at above, I would do the following:

```
model1 <- lm(crimes$Violent~crimes$MedianAge)
```

**The dependent variable always goes on the left-hand side of this equation.**

In this case, I have entered in the variable names using the `data$variable` syntax, but `lm` also offers you a more streamlined way of specifying variables, by including a `data` option separately so that you only have to put the variable names in the formula, like so:

```
model1 <- lm(Violent~MedianAge, data=crimes)
```

Because I have specified "crimes" as the dataset, *R* knows that the variables "Violent" and "MedianAge" refer to variables within this dataset. The result will be the same as the previous command, but this approach makes it easier to read the formula itself.

I have saved the output of the `lm` command into a new object that I have called "model1". You can call this object whatever you like. This is out first real example of the "object-oriented" nature of *R*. I can apply a variety of functions to this object in order to extract information about the relationship. If I want to get the most information, I can run a `summary` on this model.

```
summary(model1)
```

```
##
## Call:
## lm(formula = Violent ~ MedianAge, data = crimes)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -381.76 -118.02  -36.25   99.28  853.41
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1343.94     434.21   3.095  0.00325 **
## MedianAge     -25.58      11.56  -2.214  0.03154 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 188.1 on 49 degrees of freedom
## Multiple R-squared:  0.09092,    Adjusted R-squared:  0.07236
## F-statistic:   4.9 on 1 and 49 DF,  p-value: 0.03154
```

There is a lot information here and we actually don't know what most of it means yet. All we want is the intercept and slope. These numbers are given by the two numbers in the "Estimate" column of the "Coefficients" section. The intercept is 1343.94 and the slope is -25.58.

We could also run the `coef` command which will give us just the slope and intercept of the model.

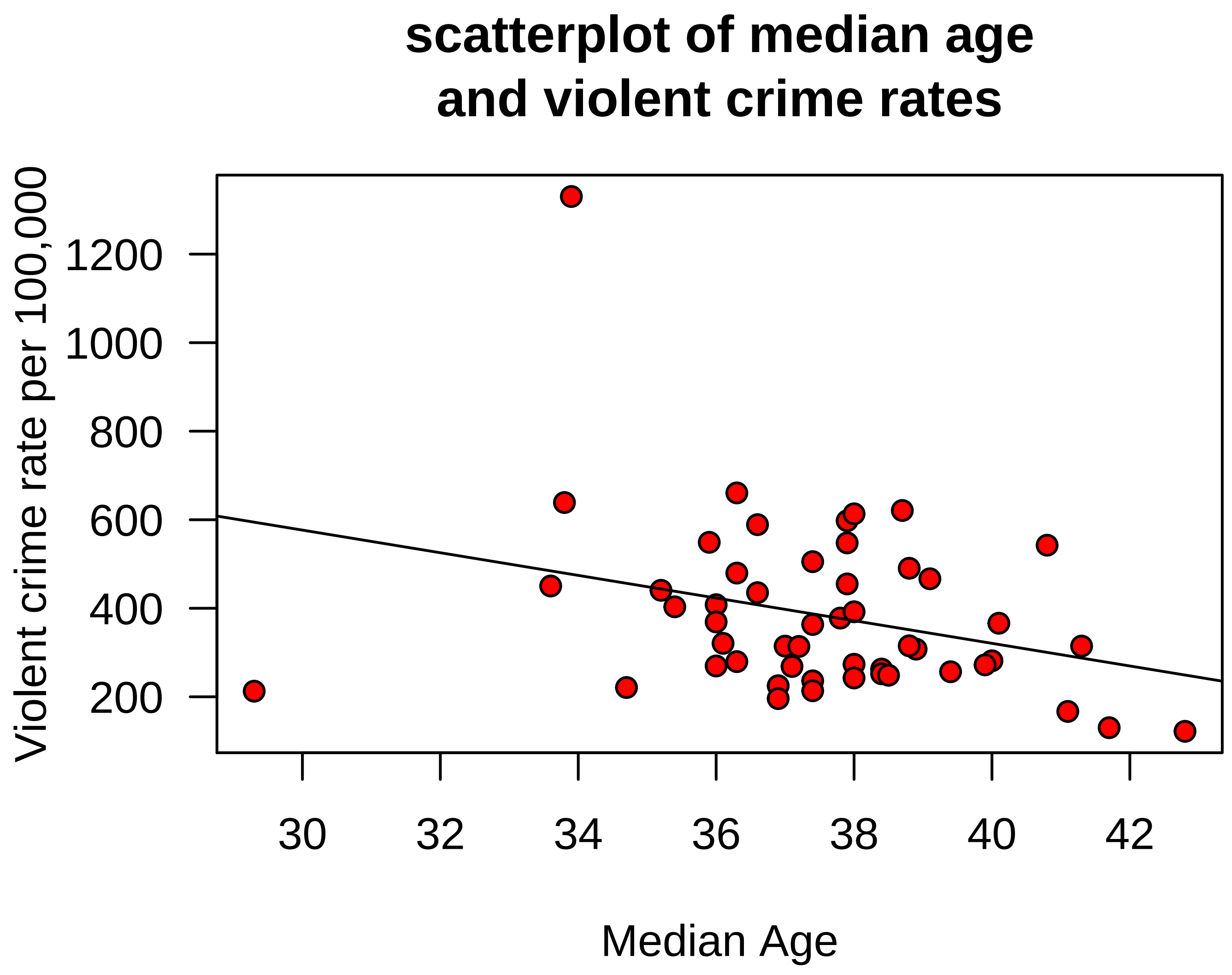
```
coef(model1)
```

```
## (Intercept)   MedianAge
##   1343.9360    -25.5795
```

This result is much more compact and will do for our purposes at the moment.

Finally, we can also plot the line to a scatterplot as shown in the very first figure of this module by using the `abline` command on our `model1` object.

```
plot(crimes$MedianAge, crimes$Violent,
     pch=21, bg="red",
     xlab="Median Age", ylab="Violent crime rate per 100,000",
     main="scatterplot of median age\nand violent crime rates", las=1)
abline(model1)
```



## The OLS regression line as a model

You will note that I saved the output of my `lm` command above as model. The `lm` command itself stands for "linear model." What do I mean by this term "model?" When we talk about "models" in statistics, we are talking about modeling the relationship between two or more variables in a formal mathematical way. In the case of the OLS regression line, we are predicting the dependent variable as a **linear function** of the independent variable.

Just as the general term model is used to describe something that is not realistic but rather an idealized representation, the same is true of our statistical models. We certainly don't believe that our linear function provides a correct interpretation of the exact relationship between our two variables. Instead we are trying to abstract from the details and fuzziness of the relationship to get a "big picture" of what the relationship looks like.

However, we always have to consider that our model is not a very good representation of the relationship. The most obvious potential problem is if the relationship is non-linear and yet we fit the relationship by a linear model, but there can be other problems as well. I will discuss these more below and the next few sections of this module will give us techniques for building better models. However, we first need to focus on how to interpret the results we just got.

## Interpeting Slopes and Intercepts

Learning to properly interpret slopes and intercepts (especially slopes) is the number one most important thing you will learn all term, because of how common the use of OLS regression is in social science statistics. You simply cannot pass the class unless you can interpret these numbers. So take the time to be careful in interpretation here.

### Interpreting Slopes

In abstract terms, the slope is always the predicted change in $y$ for a one unit increase in $x$. However, this abstract definition will simply not do when you are dealing with specific cases. You need to think about the units of $x$ and $y$ and interpret the slope in concrete terms. There are also a few other caveats to consider.

Take the interpretation I used above for the -25.6 slope of median age as a predictor of violent crime rates. My interpretation was:

> The **model predicts** that a **one year increase in age** within a state **is associated** with **25.6 fewer violent crimes per 100,000 population**, **on average**.

There are multiple things going on in this sentence that need to be addressed. First, lets address the phrase "model predicts." The idea of a model is something we will explore more later, but for now I will say that when we fit a line to a set of points to predict $x$ by $y$, we are applying a model to the data. In this case, we are applying a model that relates $y$ to $x$ by a simple linear function. All of our conclusions are dependent on this being a good model. Prefacing your interpretation with "the model predicts. . . " highlights this point.

Second, a "one year increase in age" indicates the meaning of a one unit increase in $x$. Never literally say a "one unit increase in $x$." Think about the units of $x$ and describe the change in $x$ in these terms.

Third, I use "is associated with" to indicate the relationship. This phrase is intentionally passive. We want to avoid causal language when we describe the relationship. Saying something like "when $x$ increases by one $y$ goes up by $b_1$" may sound more intuitive, but it also implies causation. The use of "is associated with" here indicates that the two variables are related without implicitly implying that one causes the other. Using causal language is the most common mistake in describing the slope.

Fourth, "25.6 fewer violent crimes per 100,000 population" is the expected change in $y$. Again, you always have to consider the unit scale of your variables. In this case, $y$ is measured as the number of crimes per 100,000 population, so a decrease of 25.6 means 25.6 fewer violent crimes per 100,000 population.

Fifth, I append the term "on average" to the end of my interpretation. This is because we know that our points don't fall on a straight line and so we don't expect a deterministic relationship between median age and violent crime. Rather, we think that if we were to take a group of states that had one year higher median age than another group of states, the average difference between the groups would be -25.6.

Lets try a couple of other examples to see how this works. I will use the lm command in R to calculate the slopes and intercepts, which I explain in the section below. First, lets look at the association between age and sexual frequency (I will explain the code I use here later in this section).

```r
coef(lm(sexf~educ, data=sex))
```

```
## (Intercept)        educ
##  49.7295901   0.0266939
```

The slope here is 0.03. Education is measured in years and sexual frequency is measured as the number of sexual encounters per year. So, the interpretation of the slope should be:

> The model predicts that a one year increase in education is associated with 0.03 more sexual encounters per year, on average.

There is a tiny positive effect here, but in real terms the relationship is basically zero. It would take you about 100 years more education to get laid 3 more times. Just think of the student loan debt.

Now, lets take the relationship between movie runtimes and tomato meter ratings:

```
coef(lm(TomatoMeter~Runtime, data=movies))
```

```
## (Intercept)     Runtime
##   5.1074601   0.4054953
```

The slope is 0.41. Runtime is measured in minutes. The tomato meter is the percent of reviews that were judged to be positive.

> The model predicts that a one minute increase in movie runtime length is associated with a 0.38 percentage point increase in the movie's Tomato Meter rating, on average.

Longer movies tend to have higher ratings. We may rightfully question the assumption of linearity for this relationship however. It seems likely that if a movie can become too long, so its possible the relationship here may be non-linear. We will explore ways of modeling that potential non-linearity later in the term.

### Interpreting Intercepts

Intercepts give the predicted value of $y$ when $x$ is zero. Again you should never interpret an intercept in these abstract terms but rather in concrete terms based on the unit scale of the variables involved. What does it mean to be zero on the $x$ variable?

In our example of the relationship of median age to violent crime rates, the intercept was 1343.9. Our independent variable is median age and the dependent variable is violent crime rates, so:

> The model predicts that in a state where the median age is zero, the violent crime rate would be 1343.9 crimes per 100,000 population, on average.

Note that I use the same "model predicts" and "on average" prefix and suffix for the intercept as I used for the slope. Beyond that I am just stating the predicted value of $y$ (crime rates) when $x$ is zero in the concrete terms of those variables.

Is it realistic to have a state with a median age of zero? No, its not. You will never observe a US state with a median age of zero. This is a common situation that often confuses students. In cases when zero falls outside the range of the independent variable, the intercept is not a particular useful number because it does not tell us about a realistic situation. The intercept's only "job" is to give a number that allows the line to go through the points on the scatterplot at the right level. You can see this in the interactive exercise above if you select the right slope of 148 and then vary the intercept.

In general making predictions for values of $x$ that fall outside the range of $x$ in the observed data is problematic. This is ofen leads to intercepts which don't make a lot of sense. This problem with zero being outside the range of data is also evident in the other two examples of slopes from the previous section. When looking at the relationship between education and sexual frequency, no respondents are actually at zero years of education and no movies are at zero minutes of runtime.

In truth, to fit the line correctly, we only need the slope and one point along the line. It is convenient to choose the point where $x = 0$ but there is no reason why we could not choose a different point. It is actually quite easy to calculate a different predicted value along the line by **re-centering** the independent variable.

To re-center the independent variable $x$, we just need to to subtract some constant value $a$ from all the values of $x$, like so:

$$x^* = x - a$$

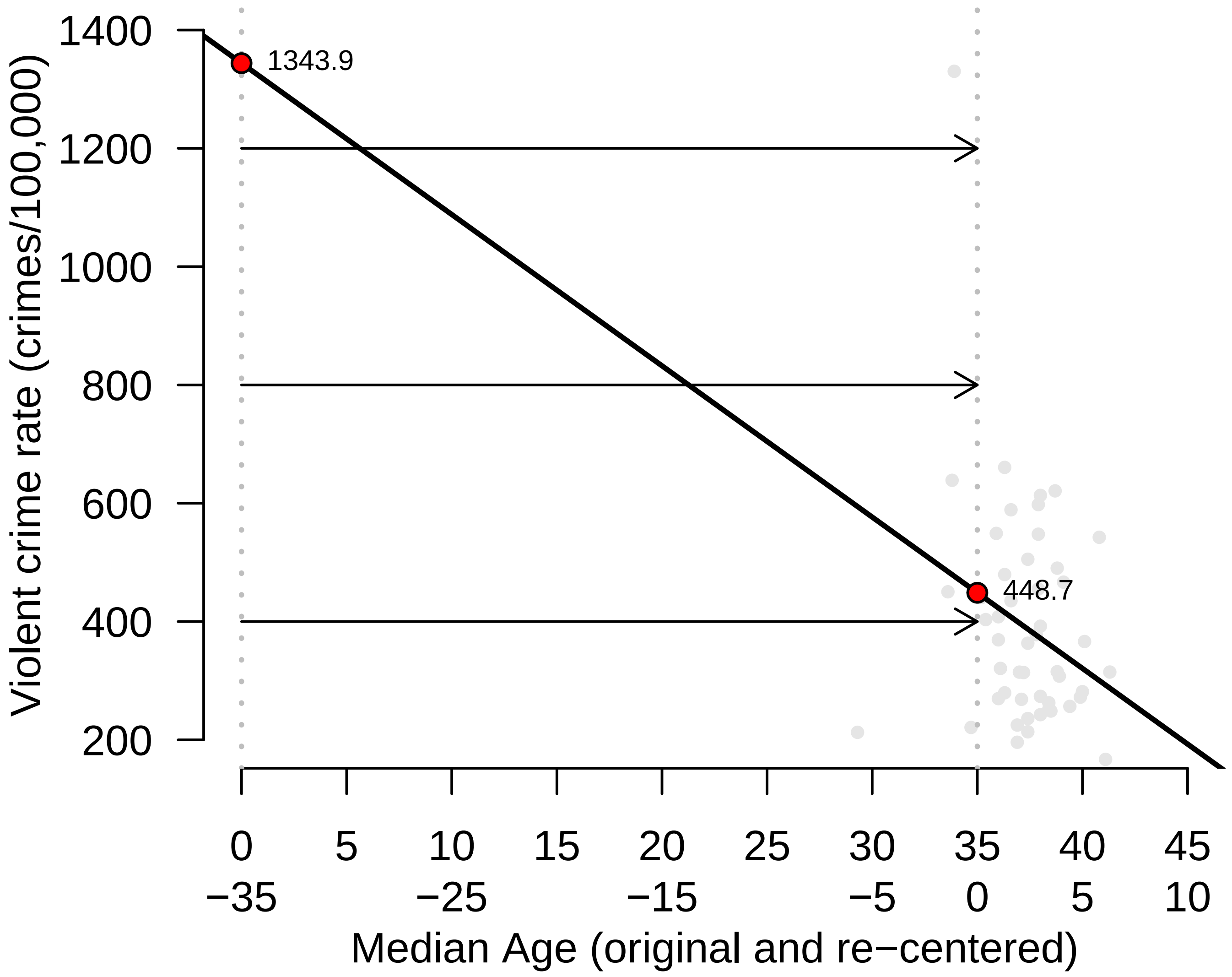
The zero value on our new variable $x^*$ will indicates that we are at the value of $a$ on the original variable $x$. If we then use $x^*$ in the OLS regression line rather than $x$, the intercept will give us the predicted value of $y$ when $x$ is equal to $a$.

Lets try this out on the model predicting violent crimes by median age. We will create a new variable where we subtract 35 from the median age variable and use that in the regression model.

```r
crimes$MedianAge.ctr <- crimes$MedianAge-35
coef(lm(Violent~MedianAge.ctr, data=crimes))
```

```
##   (Intercept) MedianAge.ctr
##      448.6533      -25.5795
```

The intercept now gives me the predicted violent crime rate in a state with a median age of 35. In effect, I have moved my y-intercept from zero to thirty-five as is shown in the figure below.



Its also possible to re-center an independent variable in the `lm` command without creating a whole new variable. If you surround the re-centering in the `I()` function within the formula, R will interpret the result of whatever is inside the `I()` function as a new variable. Here is an example based on the previous example:

```r
coef(lm(Violent~I(MedianAge-35), data=crimes))
```

```
##       (Intercept) I(MedianAge - 35)
##          448.6533          -25.5795
```

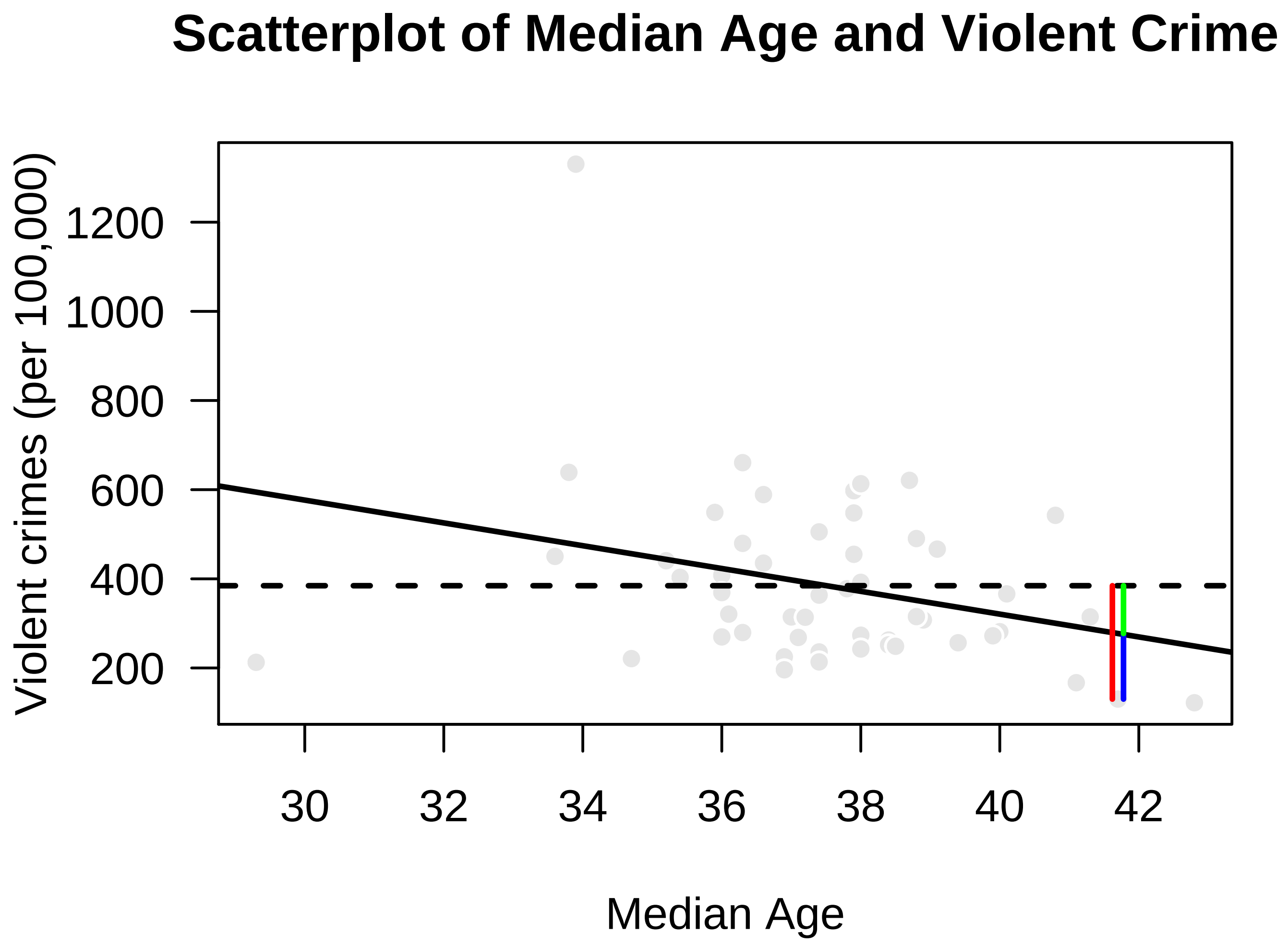
### How good is $x$ as a predictor of $y$?

If I selected a random observation from the dataset and asked you to predict the value of $y$ for this observation, what value would you guess? Your best guess would be to guess the mean of y because this is the case where your average error would be smallest. This error is defined by the distance between the mean of y and the selected value, $y_i - \bar{y}$.

Now, lets say instead of making you guess randomly I first told you the value of another variable $x$ and gave you the slope and intercept predicting $y$ from $x$. What is your best guess now? You should guess the predicted value of $\hat{y}_i$ from the regression line because now you have some additional information. There is no way that having this information could make your guess worse than just guessing the mean. The question is how much better do you do than guessing the mean. Answering this question will give us some idea of how good $x$ is as a predictor of $y$.

We can do this by separating, or *partitioning* the total possible error in our first case when we guessed the mean, into the part accounted for by $x$ and the part that is unaccounted for by $x$.

I can demonstrate this partitioning for one observation in our crime data (the state of Vermont) with this scatterplot:



The distance in red is the total distance between the observed property violent rate in the state of Vermont and the mean violent crime rate across all states (given by the dotted line). If I were instead to use the regression line predicting the violent crime rate by median age, I would predict a lower violent crime rate than average for Vermont because of its relatively high median age, but I would still predict a value that is too high. This red line is then partitioned into th green line which is the improvement in my estimate and the blue line which is the error that remains in my prediction from the regression line. If I could then repeat this process for all of the states, I could calculate the percentage of the total red lines that the green lines cover. This would give me an estimate of how much I reduce the error in my prediction by using the regression line rather than the mean to predict Vermont's violent crime rate.

In practice, we actually need to square those vertical distances because some are negative and some are positive and then we can sum them up over all the observations. So we get the following formulas:

- Total variation: $SSY = \sum_{i=1}^{n}(y_i - \bar{y})^2$
- Explained by model: $SSM = \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2$
- Unexplained by model: $SSR = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

The proportion of the variation in $y$ that is explainable or accountable by variation in $x$ is given by $SSM/SSY$.

This looks like a kind of nasty calculation, but it turns out there is a much simpler way to calculate this proportion. If we just take our correlation coefficient $r$ and square it. We will get this proportion. This measure is often called "r squared" and can be interpreted as the proportion of the variation in $y$ that is explainable or accountable by variation in $x$.

In the example above, we can calculate R squared:

```
cor(crimes$MedianAge, crimes$Violent)^2
```

```
## [1] 0.09091622
```

About 9% of the variation in violent crime rates across states can be accounted for by variation in the median age across states.

## Inference for OLS Regression models

When working with sample data, our usual issues of statistical inference apply to regression models. In this case, our primary concern is the estimate of the regression slope because the slope measures the relationship between $x$ and $y$. We can think of an underlying OLS regression model in the population:

$$\hat{y}_i = \beta_0 + \beta_1 x_i$$

We use greek "beta" values because we are describing unobserved parameters in the population. The null hypothesis in this case would be that the slope is zero indicating no relationship between $x$ and $y$:

$$H_0 : \beta_1 = 0$$

In our sample, we have a sample slope $b_1$ that is an estimate of $\beta_1$. We can apply the same logic of hypothesis testing and ask whether our $b_1$ is different enough from zero to reject the null hypothesis. We just need to find the standard error for this sample slope and the degrees of freedom to use for the test and we can do this manually.

However, I have good news for you. You don't have to do any of this by hand because the `lm` function does it for you automatically. Lets look at the full output of the model predicting violent crime rates from median age again using the `summary` command:

```
model <- lm(Violent~MedianAge, data=crimes)
summary(model)
```

```
##
## Call:
## lm(formula = Violent ~ MedianAge, data = crimes)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -381.76 -118.02  -36.25   99.28  853.41
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1343.94     434.21   3.095  0.00325 **
## MedianAge     -25.58      11.56  -2.214  0.03154 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 188.1 on 49 degrees of freedom
## Multiple R-squared:  0.09092,    Adjusted R-squared:  0.07236
## F-statistic:   4.9 on 1 and 49 DF,  p-value: 0.03154
```

The “Coefficients” table in the middle gives us all the information we need. The first column gives us the sample slope of -25.58. The second column gives us the standard error for this slope of 11.56. The third column gives us the t-statistic derived by dividing the first column by the second colum. The final column gives us the p-value for the hypothesis test. In this case, there is about a 3.2% chance of getting a sample slope this large on a sample of 51 cases if the true value in the population is zero. Of course, in this case its nonsensical because we don’t have a sample, but the numbers here will be valuable in cases with real sample data.

## Regression Line Cautions

OLS regression models can be very useful for understanding relationships, but they do have some important limitations that you should be aware of when you are doing statistical analysis.

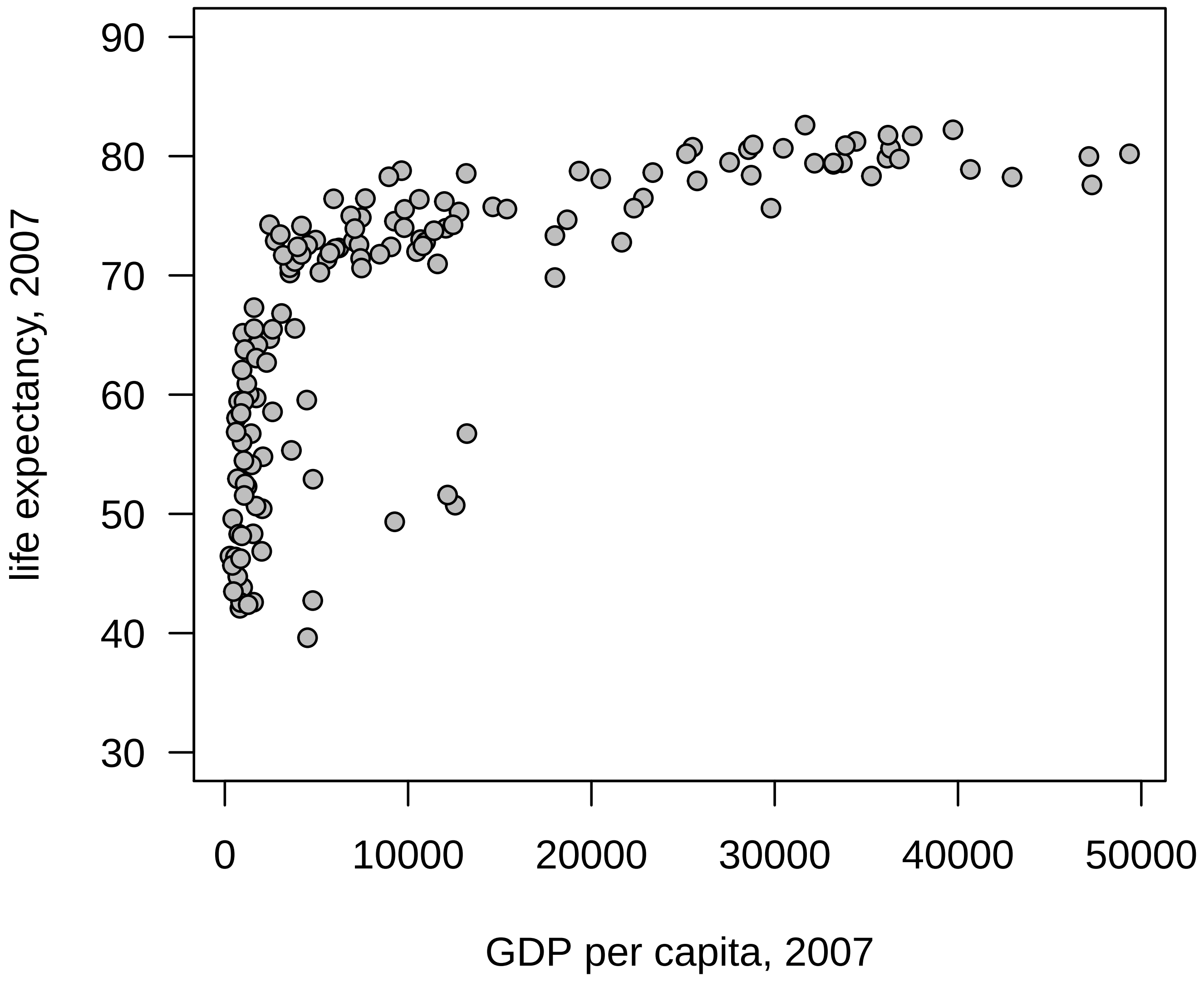
There are three major limitations/cautions to be aware of when using OLS regression:

1. OLS regression only works for linear relationships.
2. Outliers can sometimes exert heavy influence on estimates of the relationship
3. Don’t extrapolate beyond the scope of the data.

### Linearity

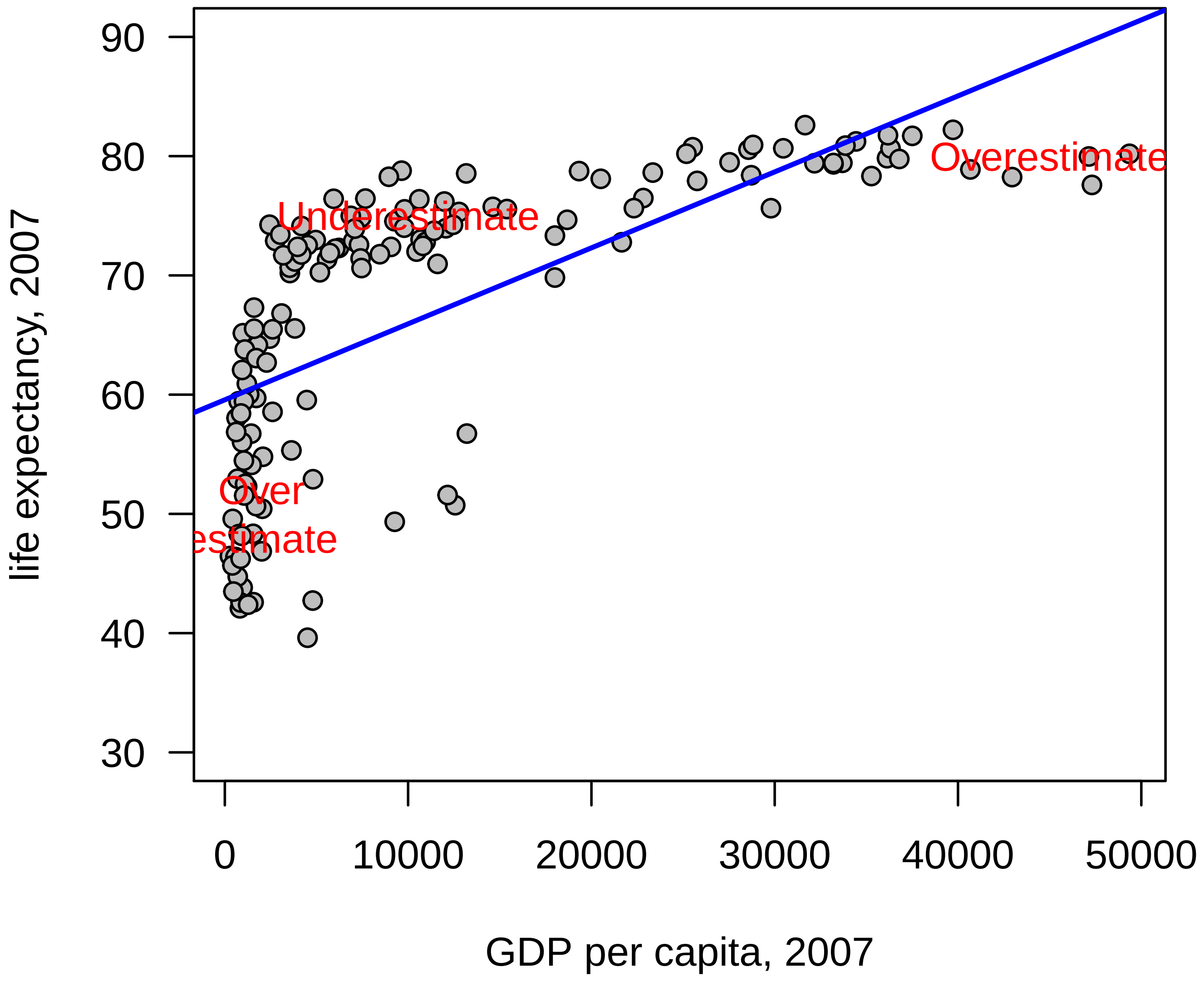
By definition, an OLS regression line is a straight line. If the underlying relationship between x and y is non-linear, then the OLS regression line will do a poor job of measuring that relationship.

One common case of non-linearity is the case of diminishing returns in which the slope gets weaker at higher values of x. That is the case with this data:



The data here plot the GDP per capita of countries in 2007 to the life expectancy of countries in 2007 The relationship is clearly a strongly positive one, but also one of diminishing returns where the positive effect seems to plateau at higher levels of national income and life expectancy. This makes sense because the same absolute increase in national income at low levels of life expectancy can be used to reduce the incidence of well-understood infectious and parasitic diseases, whereas the same absolute increase in national income at high levels of life expectancy must try to reduce the risk of less understood and treatable diseases like cancer. You get more bang for your buck when life expectancy is low.

If we try to fit a line to this data, we will get this:



Clearly a straight line is a poor fit. We systematically overestimate life expectancy at low and high GDP and underestimate life expectancy in the middle.

Its possible, in some circumstances, to correct for this problem of non-linearity but we will not explore those options in this class. For now, its just important to be aware of the problem and if you see clear non-linearity then you should question the use of an OLS regression line.

### Outliers and Influential Points

An outlier is an **influential point** if removing this observation from the dataset substantially changes the slope of the OLS regression line. You can try this out in this exercise on the crime data by clicking on points to remove them. Note how much the line changes when you remove a point. (You can click on a point a second time to add it back again).

For the case of median age, Utah and DC both have fairly strong influences on the shape of the line. Removing DC makes the relationship weaker, while removing Utah makes the relationship stronger. Outliers will tend to have the strongest influence when their placement is inconsistent with the general pattern. In this case, Utah is very inconsistent with the overall negative effect because it has both low median age and low crime rates.

Lets say that you have identified an influential point. What then? In truth there is only so much you can do.

You cannot remove a valid data point just because it is an influential point. There are two cases where it would be legitimate to exclude the point. First, if you have reason to believe that the observation is an outlier because of a data error, then it would be acceptable to remove it. Second, if you have a strong argument that the observation does not belong with the rest of the cases, because it is logically different, then it might be OK to remove it.

In our case, there is no legitimate reason to remove Utah, but there probably is a legitimate reason to remove DC. Washington DC is really a city and the rest of our observations are states that contain a mix of urban and rural population. Because crime rates are higher in urban areas, DC's crime rates look very exaggerated compared to states. Because of this "apples and oranges" problem, it is probably better to remove DC. If our unit of analysis was cities, on the other hand, then DC should remain.

In large datasets (1000+ observations), its unusual that a single point or even a small cluster of points will exert much influence on the shape of the line. The concern about influential points is mostly a concern in small datasets like the crime dataset.

#### Thou Doth Extrapolate Too Much

Its dangerous enough to assume that a linear relationship holds for your data (see the first point in this module). Its doubly dangerous to assume that this linear relationship holds beyond the scope of your data. Lets take the relationship between sexual frequency and age. We saw in the previous module that the slope here is -1.3 and the intercept is 108. The intercept itself is outside the scope of the data because we only have data on the population 18 years and older. It would be problematic to make predictions about the sexual frequency of 12 year olds, let alone zero-year olds.

Another trivial example would be to look at the growth rate of children 5-15 years of age by correlating age with height. It would be acceptable to use this model to predict the height of a 14 year old, but not a 40 year old. We expect that this growth will eventually end sometime outside the range of our data when individuals reach their final adult height. If we extrapolated the data, we would predict that 40 year olds would be very tall.

---

## The Power of Controlling for Other Variables

In the previous module, I showed that the OLS regression line predicting sexual frequency by years of education was 0.03. So in my dataset, there is a very small positive association between sexual frequency and years of education.

Its possible that this is a causal effect. We could even spin stories about why we think such a positive association (a very small one) might exist. Maybe more educated people appear sexier to the opposite sex. Maybe more educated people take better care of themselves and thus are healthier and more able to have sex. Maybe more educated people are just more sexually liberated.

Before we get carried away however its important to consider whether our results might be **spurious**. Its possible that the positive association between years of education and sexual frequency is driven by a third variable that we haven't accounted for. This is a common problem in research using observational data. Association does not necessarily mean causation because of the potential for other variables to account for our observed association (and because of the possibility of reverse causation). We refer to such variables as **lurking** or **confounding** variables.

In this case, the potentially confounding variable that we need to consider is age. Lets look at the association between age and each of our other variables (sexual frequency and education).

```r
cor(sex$sexf,sex$age)
```

```
## [1] -0.3974668
```

```r
cor(sex$educ,sex$age)
```

```
## [1] -0.06018569
```

Age is negatively correlated with sexual frequency. We have observed this relationship before and it is not terribly surprising. Older people have less sex, on average. The negative correlation between age and years of education is perhaps a little more surprising. Older people have less education than younger people, on average. This may seem surprising to you because as you get older you have more opportunity to complete more education. However, you have to remember that the data we have are a snapshot in time. We are not tracking individuals over time as they age, but rather looking at differences between older and younger people at a single point in time. This kind of data is often called a **cross-sectional** dataset. Because we are looking at a single point in time, the age differences really reflect differences in **birth cohorts** or what people often loosely call "generations." Remember that this dataset is from 2004. The difference between a 20 year old and a 60 year old is that the 20 year old was born in 1984 and the 60 year old was born in 1944.

Because we are comparing birth cohorts, the differences in educational attainment reflect history more than life cycle. Older cohorts were less educated than younger birth cohorts. On average, you will be more educated than your parents and your parents were more educated than your grandparents. Thus, the correlation between age and education is negative.

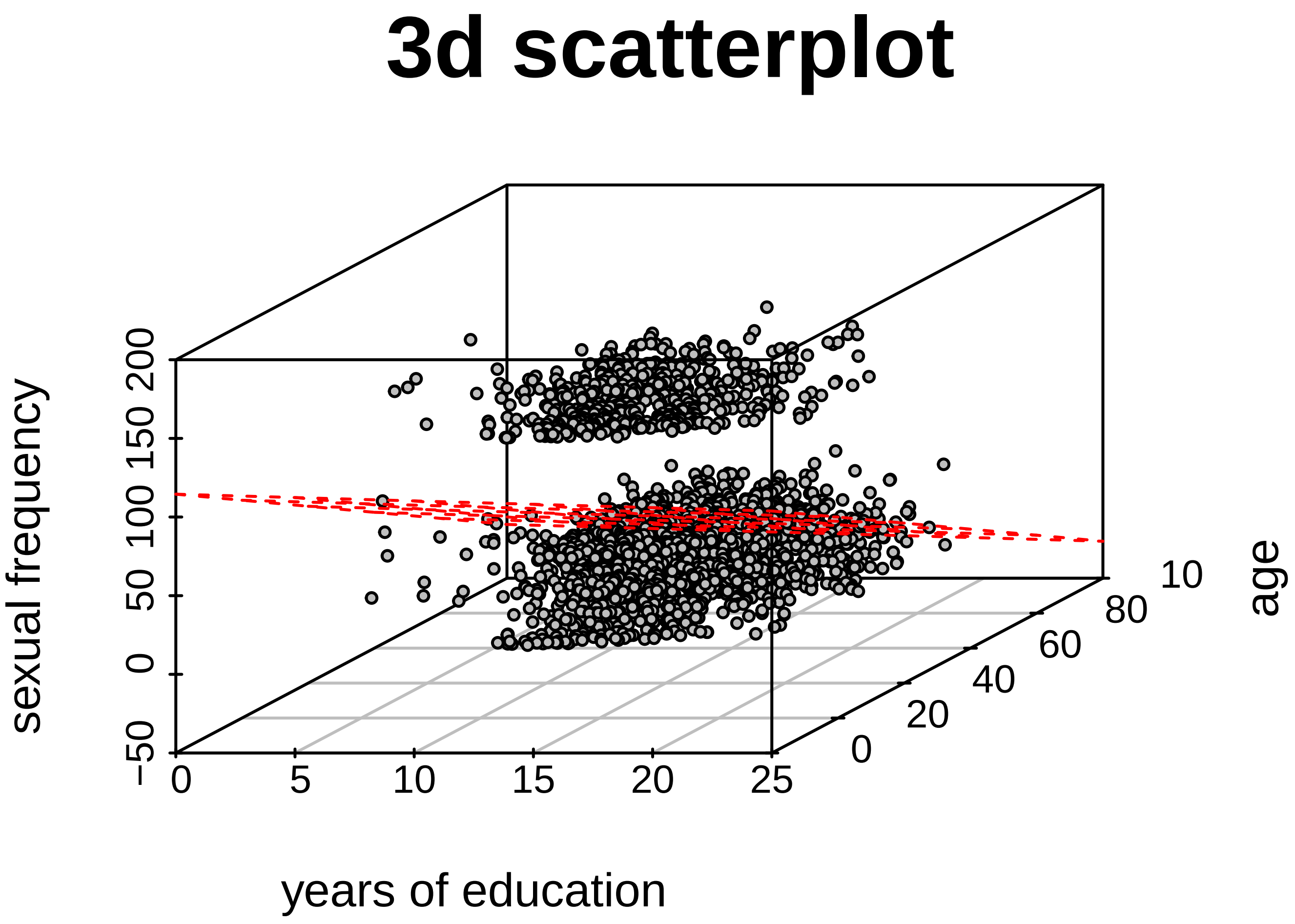
These two negative correlations suggest a **spurious** reason why we might observe a positive association between sexual frequency and education. Younger people have more education and younger people have more sex. Thus, when we look at the relationship between sexual frequency and education, we see a positive association but that positive association is indirectly driven by youth and the association of youth with both education and sex.

How can we examine whether this potential spurious explanation is accurate? It turns out that we can add more than one independent variable to an OLS regression model at the same time. The mathematical structure of such a model would be:

$$\hat{frequency}_i = b_0 + b_1(education_i) + b_2(age_i)$$

We now have two different "slopes", $b_1$ and $b_2$. These two slopes give the association of education and age, respectively, on sexual frequency, while **controlling for the other independent variable**. We now have what is called a **multivariate** OLS regression model.

This "controlling" concept is a key point that I will return to below, but first I want to try to think graphically about what this model is doing. In the case of bivariate regression, we thought of fitting a line to a scatterplot in two-dimensional space. We are doing something similar here, but since we now have three variables, our scatterplot is in three dimensions.



The dependent variable is shown on the vertical (z) axis and the two independent variables are shown on the "width" and "depth" axes (x and y). The red dashes show a flat plane across the data. The OLS regression model equation above defines this plane, rather than a single line. So, rather than fitting a straight line through the data, I am fitting a plane. Note however that if I could rotate the 3d scatterplot to hide the age "depth" dimension and in that case it would look like a two-dimensional scatterplot and the edge of the plane would look like a line that describes the relationship between education and sexual frequency. Similarly, I could rotate it the other way to look at the relationship between age and sexual frequency.

How do I know what are the best values for b0, b1, and b2 that define my plane? The logic is the same as for bivariate OLS regression: I choose values that minimize the sum of squared residuals (SSR):

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

SSR is a measure of how far the predicted values of the dependent variable are from the actual values, so we want the intercept and slopes that minimizes this error. Unlike the bivariate case, however, there is no simple formula that I can give you for the slope and intercept, without some knowledge of matrix algebra. However, R can calculate the correct numbers for you easily. I am not concerned with your technical ability to calculate these numbers by hand, but I do want you to understand why those are the "best" numbers. **They are the best numbers because they minimize the sum of the squared residuals for the model**.

We can calculate this model in *R* just by adding another variable to our model in the `lm` command:

```
model <- lm(sexf~educ+I(age-18), data=sex)
coef(model)
```

```
## (Intercept)        educ I(age - 18)
##   91.062080   -0.427747   -1.303385
```

Note that as I did in the previous module, I am re-centering age on 18 years so that I have reasonable value for the interpretation of the intercept. In equation form, our model will look like:

$$\hat{frequency_i} = 91.06 - 0.43(education_i) - 1.30(age_i - 18)$$

## Interpreting results in a multivariate OLS regression models

How do we interpret the results?

- **Intercept**: The model predicts that **18-year old individuals** at **with no education** will have 91.06 sexual encounters per year, on average.
- **Education Slope**: The model predicts that, **holding age constant**, an additional year of education is associated with 0.43 fewer sexual encounters per year, on average.
- **Age Slope**: The model predicts that, **holding education constant**, an additional year of age is associated with 1.3 fewer sexual encounters per year, on average.

The intercept is now the predicted value **when all independent variables are zero**. My interpretation of the slopes is almost identical to the bivariate case, except for one very important addition. I am now estimating the effect of each independent variable on the dependent variable while **holding constant all other independent variables**. You could also say "controlling for all other independent variables."

What does it mean to "hold other variables constant?" It means that when we look at the effect of one independent variable, we are looking at how the predicted value of the dependent variable changes while keeping all the other variables the same. For instance, the education effect above is the effect of a one year increase in education *among individuals of the same age*. Because we are looking at the effect of education among individuals of the same age, age should no longer have a confounding effect on our estimate of the effect of education. Thus holding constant/controlling for other variables helps to remove the potential spurious effect of those variables as confounders.

Note how the effect of education on sexual frequency changed once I included age as a control variable. Before controls, I estimated a slightly positive slope (0.03) but now I am estimating a substantial negative slope (-0.43). So my understanding of the relationship between education and sexual frequency is completely reversed. *When you compare individuals of the same age*, more educated individuals have less sex, on average, than less educated individuals.

### Crime example

Lets build a regression model where we predict the property crime rate in a state by the percent of adults in the state without a high school diploma and the median age of the state's residents.

```r
summary(lm(Property~PctLessHS+MedianAge, data=crimes))
```

```
##
## Call:
## lm(formula = Property ~ PctLessHS + MedianAge, data = crimes)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1080.38  -376.78    12.19   346.82  1600.56
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5136.66    1411.21   3.640 0.000666 ***
## PctLessHS      69.47      24.79   2.803 0.007286 **
## MedianAge     -83.31      35.30  -2.360 0.022368 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 567.2 on 48 degrees of freedom
## Multiple R-squared:  0.2495, Adjusted R-squared:  0.2182
## F-statistic: 7.977 on 2 and 48 DF,  p-value: 0.001021
```

Note that I am giving you the full output of summary now, but we can find the slopes and intercept by looking at the Estimate column of the "Coefficients" table. "Coefficients" is another term for slopes and intercepts because that it the technical term for these values in the regression model equation.

The model is:

$$cr\hat{i}me_i = 5137 + 69(pctlesshs_i) - 83(medianage_i)$$

The model predicts that, comparing two states with the same median age of residents, a one percent increase in the percent of the state with less than a high school diploma is associated with an increase of 69 property crimes per 100,000, on average. The model predicts that, comparing two states with the same percentage of adults without a high school diploma, a one year increase in the median age of a state's residents is associated with a decrease of 83 property crimes per 100,000, on average.

Note that we also get the $R^2$ value from the summary command. In multivariate models, the $R^2$ value always tells you what proportion of the variation in the dependent variable is accountable for by variation in all of the independent variables combined. In this case $R^2$ is 0.2495. About 25% of the variation in property crime rates across states is accountable for by variation in the percent of adults without a high school diploma and the median age of residents across states.

## Including more than two independent variables

If we can include two independent variables in a regression model, why stop there? Why not include three or four or more? The number of independent variables you can include is only limited by the sample size (you can never have more independent variables than the sample size minus one), although in practice we generally stop well short of this limit for pragmatic reasons.

Lets take the model above predicting property crime rates by percent of adults with less than a high school diploma and the median age of residents. Lets add the poverty rate as another predictor:

```r
summary(lm(Property~PctLessHS+MedianAge+Poverty, data=crimes))
```

```
##
## Call:
## lm(formula = Property ~ PctLessHS + MedianAge + Poverty, data = crimes)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1174.43  -236.69   -30.96   286.41  1218.77
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4240.38487 1383.49276   3.065   0.0036 **
## PctLessHS      0.04504   36.07642   0.001   0.9990
## MedianAge    -73.27098   33.68882  -2.175   0.0347 *
## Poverty       97.67933   38.52145   2.536   0.0146 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 537.6 on 47 degrees of freedom
## Multiple R-squared:  0.3398, Adjusted R-squared:  0.2976
## F-statistic: 8.063 on 3 and 47 DF,  p-value: 0.0001947
```

The model predicts:

- A one percent increase in the percent of adults in a state without a high school diploma is associated with 0.05 more property crimes per 100,000, on average, **holding constant the median age of residents and the poverty rate in a state**. This result is about as close to zero as you will find.

- A one year increase in the median age of a state's residents is associated with 73 fewer property crimes per 100,000, on average, **holding constant the percent of adults without a high school diploma and the poverty rate in a state**.
- A one percent increase in a state's poverty rate is associated with 98 more crimes per 100,000, on average, **holding constant the percent of adults without a high school diploma and the median age of residents in a state**.
- 34% of the variation in property crime rates across states can be accounted for by variation in the percent of adults without a high school diploma, residents' median age, and the poverty rates across states.

When I interpret the models now, I am holding constant the other two variables when I estimate the effect of each. Note that controlling for the poverty rate has a huge effect on the education variable whose effect goes from a substantial positive effect to basically zero effect. What does this tell us? Poverty rates and high school dropout rates are positively correlated and so when you don't control for poverty rates, it looks like the high school dropout rate predicts crime because states with high high school dropout rates have high poverty rates and high poverty rates predict property crime rates. Once you control for the poverty rate, you see that it is economic deprivation not educational deprivation that is driving the crime rate.

In general, the form of the multivariate regression model is:

$$\hat{y}_i = b_0 + b_1x_{i1} + b_2x_{i2} + b_3x_{i3} + \ldots + b_px_{ip}$$

The intercept is given by $b_0$. This is the predicted value of $y$ when all of the independent variables are zero. The remaining $b$'s give the slopes for all of the variables up through the $p$th variable. Each of these gives the predicted change in $y$ for a unit increase in that independent variable, **holding all other independent variables constant**.

## How to read a table of regression results

In academic journal articles and books, the results of OLS regression models are represented in a fairly standard way. In order to understand how to read these articles, you need to understand this presentation style. Its not immediately intuitive for everyone. The table below shows the typical style. In this table, I am reporting three regression models with the property crime rates as the dependent variable and three different independent variables.

OLS regression models predicting violent crime rates for US states

(1)

(2)

(3)

Percent Less than HS

78.83***

69.47***

0.05

(25.58)

(24.79)

(36.08)

Median Age

-83.31**

-73.27**

(35.30)

(33.69)

Poverty Rate

97.68**

(38.52)

Constant

1,892.85***

5,136.66***

4,240.38***

(335.33)

(1,411.21)

(1,383.49)

Observations

51

51

51

R2

0.16

0.25

0.34

Note:

*p<0.1;* ***p<0.05;*** p<0.01

Standard errors in parenthesis

When reading this table and others like it, keep the following issues in mind:

1. The first question you should ask is "what is the dependent variable?" This is the outcome that we are trying to predict. Typically, the dependent variable will be listed in the title of the table. In this case, the title tells you that the dependent variable is property crime rates and the unit of analysis is US states.
2. The independent variables are listed on the rows of the table. In this case, I have independent variables of percent less than HS, median age, and the poverty rate. As I will explain below, just because an independent variable is listed here does not mean that it is actually included in all models.
3. The term "constant" is a synonym for "intercept."
4. Models are listed in each column of the table. If numbers are listed for the row of a particular independent variable then that variable is included in that particular model. In this case, I have three different models. The first model only has numbers listed for Percent less than HS, so that is the only independent variable in the first model. The second model has numbers listed for Percent less than HS and Median Age, so both of these variables are included in the model. The third model includes all three variables in the model. Remember that in each case the dependent variable is the property crime rate.
5. Within each cell with numbers listed there is a lot going on. We are primarily interested in the main number listed at the top. This number is the slope (or intercept in the case of the "Constant" row). The number in parenthesis is the standard error for each slope in the model. You could use this standard error and the slope estimate above it to calculate t-statistics and p-values exactly. However, the asterisks give you an easy visual short cut to determine the rough size of the p-value. These asterisks indicate if the p-value is below a certain level, as shown in the notes at the bottom. The cut-offs of 0.05, 0.01, and 0.001 used here are pretty standard for the discipline. So an asterisks generally means that the result is "statistically significant." However, its important to keep in mind as noted above that these cut-offs are ultimately arbitrary and should never be confused with the substantive size of the effect itself.
6. At the bottom, you typically get a number of summary measures of the model. The only two we care about are the number of observations and the $R^2$ of the model.

---

## Including Categorical Variables as Predictors

To this point, we only know how to include quantitative variables into OLS regression models. However, it turns out you can use a fairly easy trick to include categorical variables as independent variables in OLS regression models. By including categorical variables as independent variables, we expand considerably the range of things that we can do with OLS regression models. The most difficult part of this trick is correctly interpreting your results.

### Indicator variables

As an example, I am going to look at the relationship between religious affiliation and sexual frequency. To keep our example simple I am going to **dichotomize** the religious affiliation variable, which means I am going to collapse it into two categories, rather than the six categories in the dataset. I will use a simply dichotomy of "Not Religious/Religious." In R, I can create this variable like so:

```
sex$norelig <- sex$relig=="None"
```

This is technically a **boolean** variable, which means it takes a TRUE or FALSE value. For our purposes, TRUE is a non-religious person.

We already know how to look at the relationship between sexual frequency and this dichotomized religious affiliation variable. We can look at the mean differences in sexual frequency across our two categories:

```
tapply(sex$sexf, sex$norelig, mean)
```

```
##    FALSE     TRUE
## 48.33671 59.84862
```

```
59.84862-48.33671
```

```
## [1] 11.51191
```

The non-religious have sex 11.5 more times per year than the religious, on average. Hallelujah?

We can represent this same mean difference in a regression model framework by using an indicator variable. An indicator variable is a variable that only takes a value of zero or one. It takes a value of one when the observation is in the indicated category and a zero otherwise. Mathematically, we would say:

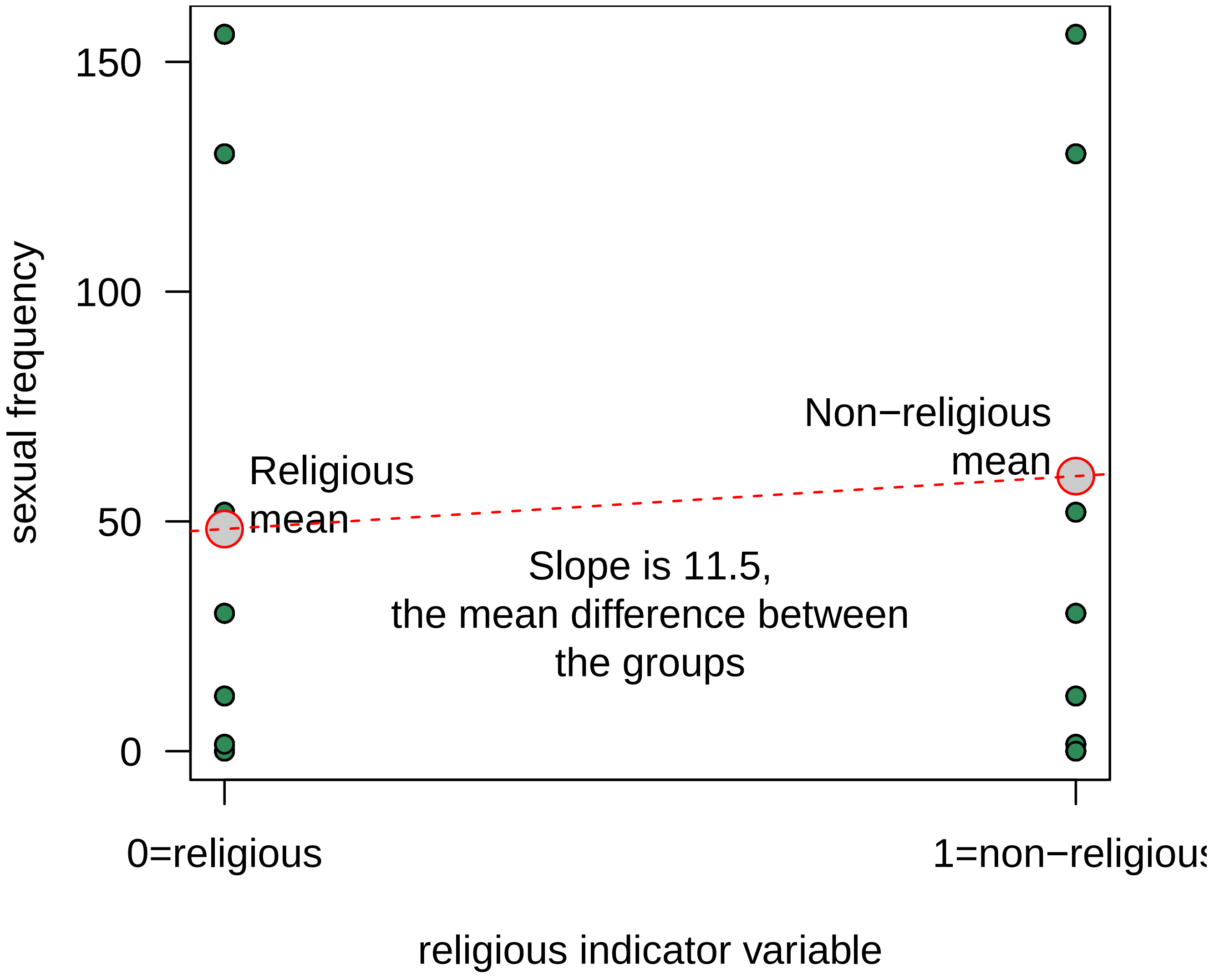
$$nonrelig_i = \begin{cases} 1 & \text{if non-religious} \\ 0 & \text{otherwise} \end{cases}$$

The **indicated category** is the category which gets a one on the indicator variable. In this case the indicated category is non-religious. The **reference category** is the category that gets no indicator variable. In this case, that is just the religious group. Later on, we will see that this can become slightly more complicated. You can think of the indicator variable as an on/off switch where 1 indicates that it is "on" (i.e. the observation belongs to the indicated category) and 0 indicates that it is "off" (i.e. the observation does not belong to the indicated category).

What would happen if we put this indicator variable into a regression model predicting sexual frequency like so:

$$\hat{frequency}_i = b_0 + b_1(nonrelig_i)$$

How would we interpret the slope and intercept for such a model? It might help to look at a scatterplot of this relationship.



Notice that all of the points align vertically either at the 0 or 1 on the x-axis. This is because the indicator variable can only take these two values. In addition, there is a lot of overplotting of these points right on top of one another so it is hard to see the trend. To simplify things I have plotted the means of the two groups in grey dots and the OLS regression line for the scatterplot in red. In order to be the best-fitting line, this OLS regression line must connect those two dots that represent the mean of each group.

What will the slope of this line be? If we go up "one unit" on the non-religious indicator variable we have gone from a religious person to a non-religious person and the change in predicted sexual frequency is equal to the mean difference of 11.5 between the groups. The intercept is given by the value at zero which is just given by the mean sexual frequency among the religious of 48.3. So, the OLS regression line should look like:

$$\hat{frequency_i} = 48.3 + 11.5(nonrelig_i)$$

I can calculate these same numbers in *R* with the `lm` command:

```
coef(lm(sexf~norelig, data=sex))
```

```
## (Intercept) noreligTRUE
##    48.33671    11.51191
```

The numbers are the same. More important than the numbers, however, is the interpretation of the numbers.

The intercept is the mean of the dependent variable for the reference category. The slope is the mean difference between the reference category and the indicated category. In this case, I would say:

- Religious individuals have sex 48.3 times per year, on average.
- Non-religious individuals have sex 11.5 times more per year than non-religious individuals, on average.

Note that I can derive the sexual frequency of the non-religious from these two numbers by taking the value for the non-religious and adding the mean difference to find out that non-religious individuals have sex 59.8 times per year, on average.

### Reversing the indicator variable

What if I switched my indicator variable so that the religious were indicated and the non-religious were the reference category?

$$relig_i = \begin{cases} 1 & \text{if religious} \\ 0 & \text{otherwise} \end{cases}$$

Lets try it out in $R$ and see (the `!=` below is computer lingo for "not equal to"):

```r
sex$religious <- sex$relig!="None"
coef(lm(sexf~religious, data=sex))
```

```
##   (Intercept) religiousTRUE 
##      59.84862     -11.51191
```

Lets compare the two models:

$$\hat{frequency_i} = 48.3 + 11.5(nonrelig_i)$$

$$\hat{frequency_i} = 59.8 - 11.5(relig_i)$$

Both models give me the exact same information, but from the perspective of a different reference group. The first model tells me the mean sexual frequency of the religious (48.3) and how much *more* sex the non-religious have on average (11.5). The second model tells me the mean sexual frequency of the non-religious (59.8) and how much *less* sex the religious have (-11.5). I can easily derive one model from the other, without actually having to calculate it in R. Therefore, which category you set as the reference category is really a matter of taste, rather than one of consequence. The results are the same either way.

## Categorical variables with more than two categories

What if I have a categorical variable that has more than two categories? Lets expand the religious variable that I dichotomized back to its original scale. There are six different categories: Fundamentalist Protestant, Mainline Protestant, Catholic, Jewish, Other, and None:

```r
summary(sex$relig)
```

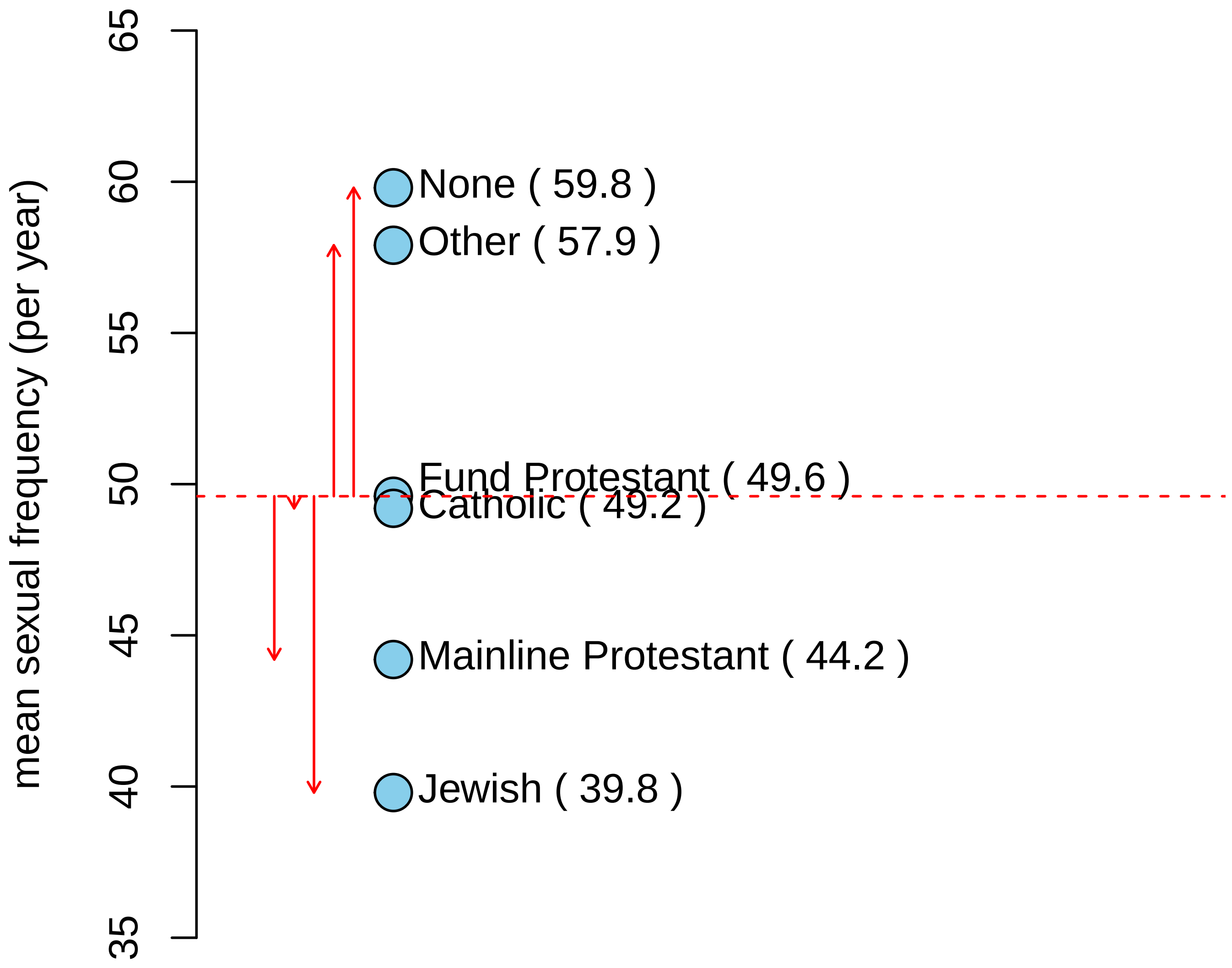
```
##     Fund Protestant Mainline Protestant            Catholic 
##                 556                 529                 507 
##              Jewish               Other                None 
##                  39                 145                 327
```

Lets look at the mean sexual frequency for each of these groups.

```r
round(tapply(sex$sexf, sex$relig, mean),1)
```

```
##     Fund Protestant Mainline Protestant            Catholic
##                49.6                44.2                49.2
##              Jewish               Other                None
##                39.8                57.9                59.8
```

We could plot up these means on a number line to get a visual display of the differences:



Nones and others clearly have much higher mean sexual frequency than the remaining religious groups and Jews have much lower mean sexual frequency. The three Christian groups cluster in the middle, although mainline protestants have a lower mean sexual frequency than the other two.

This plot also shows the mean differences between the groups, with fundamentalist Protestants set as the reference category. The vertical distances from the dotted red line (the mean of fundamentalist Protestants) give the mean differences between each religious group and fundamentalist Protestants. So we can see that "Nones" have sex 10.2 more times per year than fundamentalist Protestants, on average, and mainline Protestants have sex 5.4 fewer times per year, on average, than fundamentalist Protestants.

We can use the same logic of indicator variables we developed above to represent the mean differences between groups observed here in a regression model framework. However, because we now have six categories, we will need five indicator variables. You always need **one less indicator variable than the number of**

**categories**. The category which doesn't get an indicator variable is your reference category. As per the graph above, I will make Fundamentalist Protestants my reference category. Therefore, I need one indicator variable for each of the other five categories:

$$main_i = \begin{cases} 1 & \text{if main} \\ 0 & \text{otherwise} \end{cases}$$

$$catholic_i = \begin{cases} 1 & \text{if catholic} \\ 0 & \text{otherwise} \end{cases}$$

$$jewish_i = \begin{cases} 1 & \text{if jewish} \\ 0 & \text{otherwise} \end{cases}$$

$$other_i = \begin{cases} 1 & \text{if other religion} \\ 0 & \text{otherwise} \end{cases}$$

$$none_i = \begin{cases} 1 & \text{if no religion} \\ 0 & \text{otherwise} \end{cases}$$

Now lets put these variables into an OLS regression model:

$$\hat{frequency_i} = b_0 + b_1(main_i) + b_2(catholic_i) + b_3(jewish_i) + b_4(other_i) + b_5(none_i)$$

We can figure out how all this works by getting the predicted value for the member of a specific group. That respondent should get a 1 for the variable where they are a member and a zero on all other variables. For example, a fundamentalist protestant should get a zero on all of these variables:

$$\hat{frequency_i} = b_0 + b_1(0) + b_2(0) + b_3(0) + b_4(0) + b_5(0) = b_0$$

So, the intercept is the predicted value for fundamentalist Protestants. Similarly we could calculate the predicted value for mainline Protestants:

$$\hat{frequency_i} = b_0 + b_1(1) + b_2(0) + b_3(0) + b_4(0) + b_5(0) = b_0 + b_1$$

The difference between the two is $b_1$, so this "slope" gives the mean difference between mainline and fundamentalist Protestants. We could do the same thing for Catholics:

$$\hat{frequency_i} = b_0 + b_1(0) + b_2(1) + b_3(0) + b_4(0) + b_5(0) = b_0 + b_2$$

The mean difference between Catholics and fundamentalist Protestants is given by $b_2$.

In general, each of the "slopes" is the mean difference between the indicated category and the **reference category**. In this case, the reference category is fundamentalist Protestants so each of the slopes gives the mean difference between that religious category and fundamentalist Protestant, just like the graph above.

R is fairly intelligent about handling all of these indicator variables and you don't actually have to create these five different variables. If you put a categorical variable into your regression formula, R will know to treat it as a set of indicator categories. The only catch is that R will already have a default category set as the reference. It just so happens that in our GSS data, fundamentalist Protestants are already set as the reference. So I can run this model by:

```
model <- lm(sexf~relig, data=sex)
round(coef(model),2)
```

```
##              (Intercept) religMainline Protestant            religCatholic
##                    49.60                    -5.44                    -0.36
##              religJewish               religOther                religNone
##                    -9.84                     8.30                    10.25
```

You can tell which category is the reference by which category is left out here. Note how the coefficients (given by the estimates column) match the mean differences I calculated above in the graph. We are simply reproducing these mean differences in a regression model framework.

## Categorical and quantitative variables combined in a single model

If all we are doing is reproducing mean differences between categories, what good is this method? After all, we already know how to do that. The major advantage of putting these mean differences into a regression model framework is that we can **control for other potentially confounding variables**.

These sexual frequency differences by religious affiliation are a prime example. Lets take a look at the age differences between religious affiliations:

```
round(tapply(sex$age, sex$relig, mean),1)
```

```
##     Fund Protestant Mainline Protestant            Catholic
##                45.4                47.3                44.9
##              Jewish               Other                None
##                53.7                38.6                39.5
```

Notice how closely these age differences mirror the differences in sexual frequency. Others and nones are the youngest, while Jews are the oldest. Among Christians, mainline Protestants are older than fundamentalist Protestants and Catholics. We also know from prior work that age has a negative effect on sexual frequency. This should make us suspicious that some (or all) of the observed differences in sexual frequency between religious groups simply reflect age differences between those groups.

We can easily address this issue by simply including age as a control variable in our model:

```
model <- lm(sexf~relig+age, data=sex)
round(coef(model),2)
```

```
##              (Intercept) religMainline Protestant            religCatholic
##                   107.90                    -3.04                    -0.96
##              religJewish               religOther                religNone
##                     0.88                    -0.43                     2.69
##                      age
##                    -1.28
```

We now interpret those slopes as the mean difference in sexual frequency between fundamentalist Protestants and the indicated category, **among individuals of the same age**. So for example, we would interpret the 2.69 on "None" as:

> The model predicts that, among individuals of the same age, those with no religious preference have sex 2.69 more times per year than fundamentalist protestants, on average.

We would also interpret the age effect controlling for religious affiliation like so:

> The model predicts, that holding religious affiliation constant, a one year increase in age is associated with 1.28 fewer instances of sex per year, on average.

The table below helps to highlight the change in the effects once age is controlled.

OLS regression models predicting sexual frequency

| | (1) | (2) |
|---|---|---|
| Mainline Protestant | -5.44* | -3.04 |
| | (3.24) | (2.99) |
| Catholic | -0.36 | -0.96 |
| | (3.28) | (3.02) |
| Jewish | -9.84 | 0.88 |
| | (8.84) | (8.17) |
| Other | 8.30* | -0.43 |
| | (4.98) | (4.61) |
| None | 10.25*** | 2.69 |
| | (3.72) | (3.45) |
| Age | | -1.28*** |
| | | (0.07) |
| Constant | 49.60*** | 107.90*** |
| | (2.26) | |

(3.68)

Observations

2,103

2,103

R2

0.01

0.16

Note:

*p<0.1;* ***p<0.05;*** p<0.01

SE's in parenthesis. Reference category is fund. Protestant

All of the coefficients (except for Catholic, which was tiny anyway) have declined substantially in size. The mean differences from fundamentalist Protestants for both Jews and other religions have basically disappeared and the "None" effect has been severely reduced. In other words, almost all or all of the observed differences in sexual frequency by religious affiliation were indirectly a product of underlying age differences between religious affiliations. If you were just about ready to convert to a different religion to get laid more often (or less depending on your preferences), you may want to hold off for the moment.

---

## Interaction Terms

By definition, a linear model is an **additive** model. As you increase or decrease the value of one independent variable you increase or decrease the predicted value of the dependent variable by a set amount, **regardless of the other values of the independent variable**. This is an assumption built into the linear model by its additive form, and it may misrepresent some relationships where independent variables **interact** with one another to produce more complicated effects. In particular, in this section, we want to know whether the effect (i.e. the slope) of one independent variable varies by the value of another independent variable.

### The nature of additive models

As an example for this section, I am going to look at the relationship between movie genre, runtime, and tomato meter ratings. To simplify things, I am going to only look at these relationships for two genres: action and comedy. I can limit my movies dataset to these two genres with the following command:

```
movies.short <- subset(movies, Genre=="Comedy" | movies$Genre=="Action")
```

Now lets look at a simple model where genre and runtime both predict Tomato Meter ratings.

```
round(coef(lm(TomatoMeter~Genre+Runtime, data=movies.short)),2)
```

```
## (Intercept) GenreComedy     Runtime
##       -1.75        4.43        0.41
```

Genre is a categorical variable and action movies are set as the reference category. In equation form, the model looks like:

$$\hat{meter}_i = -1.75 + 4.43(comedy_i) + 0.31(runtime_i)$$

I can interpret my slopes as follows:

- The model predicts that when comparing movies of the same runtime, comedies have Tomato Meter ratings 4.43 percentage points higher than action movies, on average.
- The model predicts that, holding constant movie genre, a one minute increase in movie runtime is associated with a 0.31 percentage point increase in the Tomato Meter rating, on average.

This is an additive model. If we move from an action movie to a comedy of the same runtime, our predicted Tomato Meter rating goes up by 4.43, **regardless of the actual value of runtime**. If we increase movie runtime by one minute while keeping genre the same, our predicted Tomato Meter rating goes up by 0.41, **regardless of whether that genre is action or comedy**.

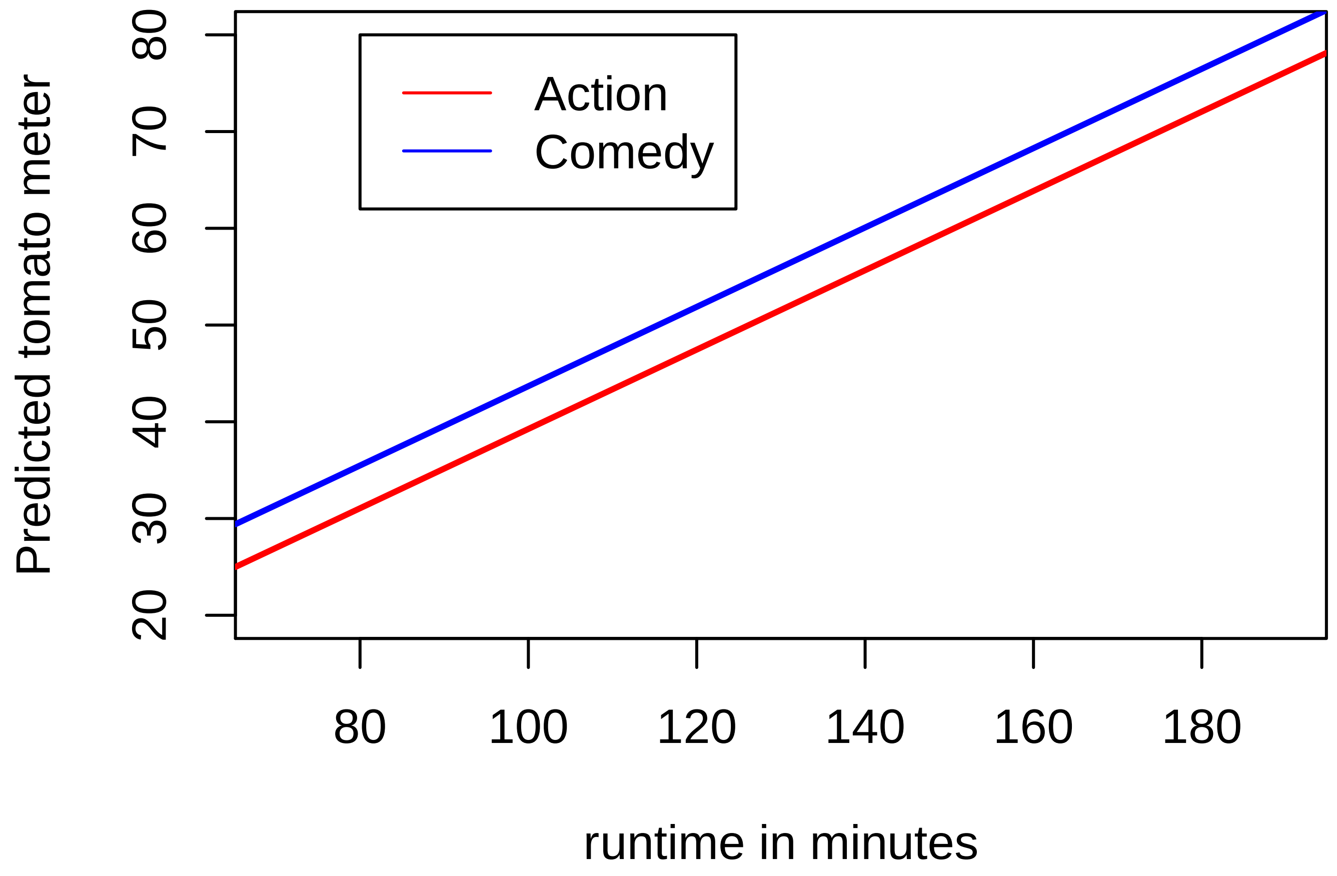
It may help to graphically visualize the nature of this additive relationship. We can do this by plotting lines showing the relationship between runtime and Tomato Meter ratings separately for our two different genres of action and comedy. The line for action movies is given by:

$$\hat{meter_i} = -1.75 + 4.43(0) + 0.41(runtime_i) = -1.75 + 0.41(runtime_i)$$

The line for comedy movies is given by:

$$\hat{meter_i} = -1.75 + 4.43(1) + 0.41(runtime_i) = 2.68 + 0.41(runtime_i)$$

Each line has an intercept and a slope. Notice that the intercepts are different but the slopes are the same. That means we have two parallel lines at different levels. You can see this easily by graphing the lines out:



The parallel lines are an assumption of the OLS regression model structure we have used. There are two consequences of this assumption. First, At every single level of runtime, the predicted Tomato Meter difference between comedy and action movies is exactly 4.62. This can be seen on the graph by the consistent gap between the blue and red line. Second, the effect of runtime on the Tomato Meter rating is assumed to be the same for action and comedy movies. This can be seen on the graph by the fact that both lines have the exact same slope.

Although these may seem like two different issues, they are really the same issue from different perspectives. If we were to allow the slopes of the blue and red line to be different, then the gap between them would not be static. The questions is how can we allow the slopes of the two lines to be different. This is where the concept of the **interaction term** comes in.

## The interaction term

An interaction term is a variable that is constructed from two other variables by multiplying those two variables together. In our case, we can easily construct an interaction term as follows:

```
movies.short$comedy <- movies.short$Genre=="Comedy"
movies.short$interaction <- movies.short$Runtime*movies.short$comedy
```

In this case, I had to create a real indicator variable for comedy before I could multiply them, but then I just multiply this indicator variable by movie runtime. Now lets add this interaction term to the model:

```
model <- lm(TomatoMeter~Runtime+comedy+interaction, data=movies.short)
round(coef(model), 2)
```

```
## (Intercept)     Runtime  comedyTRUE interaction
##      -14.45        0.52       24.36       -0.19
```

We now have an additional "slope" for the interaction term. Lets write this model out in equation form to try to figure out what is going on here.

$$\hat{meter_i} = -14.45 + 24.36(comedy_i) + 0.52(runtime_i) - 0.19(runtime_i * comedy_i)$$

Remember that the interaction term is just a literal multiplication of the two other variables. To figure out how this all works, lets once again separate this into two lines predicting Tomato Meter by runtime, for comedies and action movies separately.

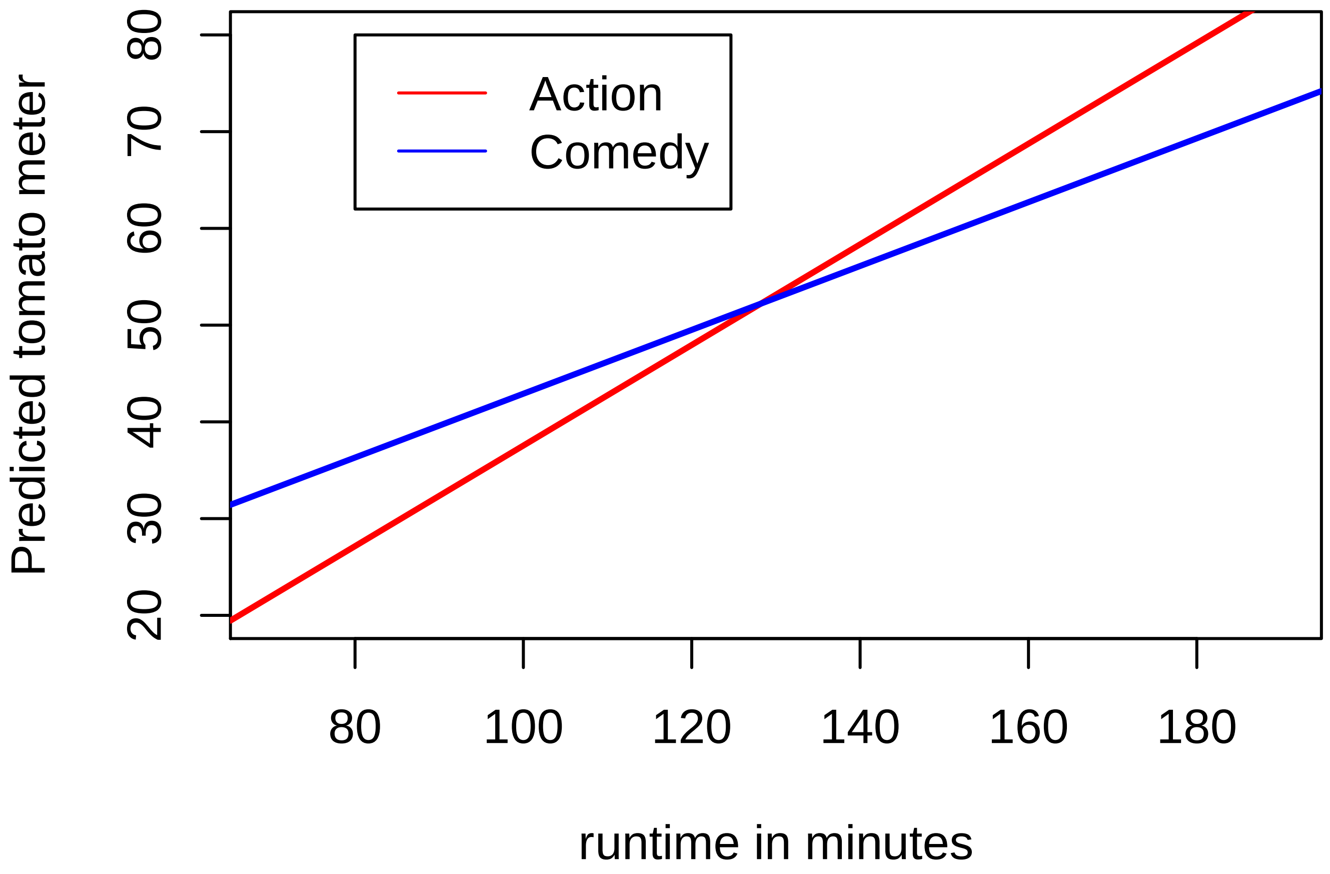
For action movies, the equation is:

$$\hat{meter_i} = -14.45 + 24.36(0) + 0.52(runtime_i) - 0.19(runtime_i * 0) = -14.45 + 0.52(runtime_i)$$

For comedy movies, the equation is:

$$\hat{meter_i} = -14.45+24.36(1)+0.52(runtime_i)-0.19(runtime_i*1) = (-14.45+24.36)+(0.52-0.19)(runtime_i) = 9.91+0.33(run$$

We now have two lines with different intercept and **different slopes**. The interaction term has allowed the effect of runtime on the Tomato Meter to vary by type of genre. In this case, the interaction term tells us how much smaller the slope is for comedy movies than for action movies. We can also just plot the lines to see how it looks:



The pattern here is fairly clear. Short comedies get better ratings than short action movies, while long comedies get worse ratings than long action movies. Put another way, comedies get less "return" in terms of their ratings when increasing their length than do action movies. This can be seen by the much steeper slope for action movies.

### Interpreting interaction terms

Interpreting interaction terms can be tricky, because the inclusion of an interaction term **also changes the meaning of other slopes in the model**. The slopes for the two variables that make up the interaction term are called the **main effects**. In our example, those two variables are runtime and the comedy indicator variable and the main effects of these variables are 0.52 and 24.36, respectively. The most important rule to remember is that when an interaction term is in a model, the main effects are only the expected effects **when the other variable involved in the interaction is zero**. This is because the interaction implies that the effects of the two variables are not constant but rather change depending on the value of the other variable in the interaction term. Therefore, we can only interpret effects at a particular value of the other variable. So I would interpret these main effects as follows:

- The model predicts that **among action movies**, a one minute increase in movie runtime is associated with a 0.52 point increase in the Tomato Meter rating, on average.
- The model predicts that **among movies with zero minutes of runtime** (outside the scope of data of course), comedies are predicted to have Tomato Meter ratings 24.36 points higher than action movies, on average.

Notice that I did not have to say I was controlling for the other variable. I am doing more than controlling when I include an interaction term. I am conditioning the effect of one variable on the value of another. That is why I instead use the phrase "among observations that are zero on the other variable." Note that you could also include other non-interacted variables in this model as well, like maturity rating, in which case you would also need to indicate that you controlled for those variables.

Interpreting interaction terms themselves can also be tricky because they are the difference in the effect of on variable depending on the value of another. One approach is to interpret this difference in effect directly. In this case, we would say:

- The model predicts that the predicted increase in Tomato Meter ratings for a one minute increase in movie runtime is 0.19 points smaller for comedy movies than for action movies, on average.

You have to be careful with this type of interpretation. In this case, both slopes were still positive so I can talk about how the effect was smaller. However, in some cases, the slopes may end up in different directions entirely which would require a somewhat different interpretation. Another approach is to actually calculate the slope for the indicated category (comedies) and interpret it directly:

- The model predicts that **among comedy movies**, a one minute increase in movie runtime is associated with a 0.33 increase in the Tomato Meter rating, on average (which is lower than for action movies).

In short, you have to be careful and thoughtful when thinking about how to interpret interaction terms.

## Interaction terms in *R*

In the example above, I created the interaction term manually, but I didn't actually need to do this. *R* has a shortcut method for calculating interaction terms:

```
model <- lm(TomatoMeter~Runtime*Genre, data=movies.short)
round(coef(model),2)
```

```
##         (Intercept)             Runtime         GenreComedy
##              -14.45                0.52               24.36
## Runtime:GenreComedy
##               -0.19
```

The results are exactly the same as before. To include an interaction term between two variables I just have to connect them with a `*` rather than a `+` in the `lm` formula. By default, *R* will include each variable separately as well as their interaction.

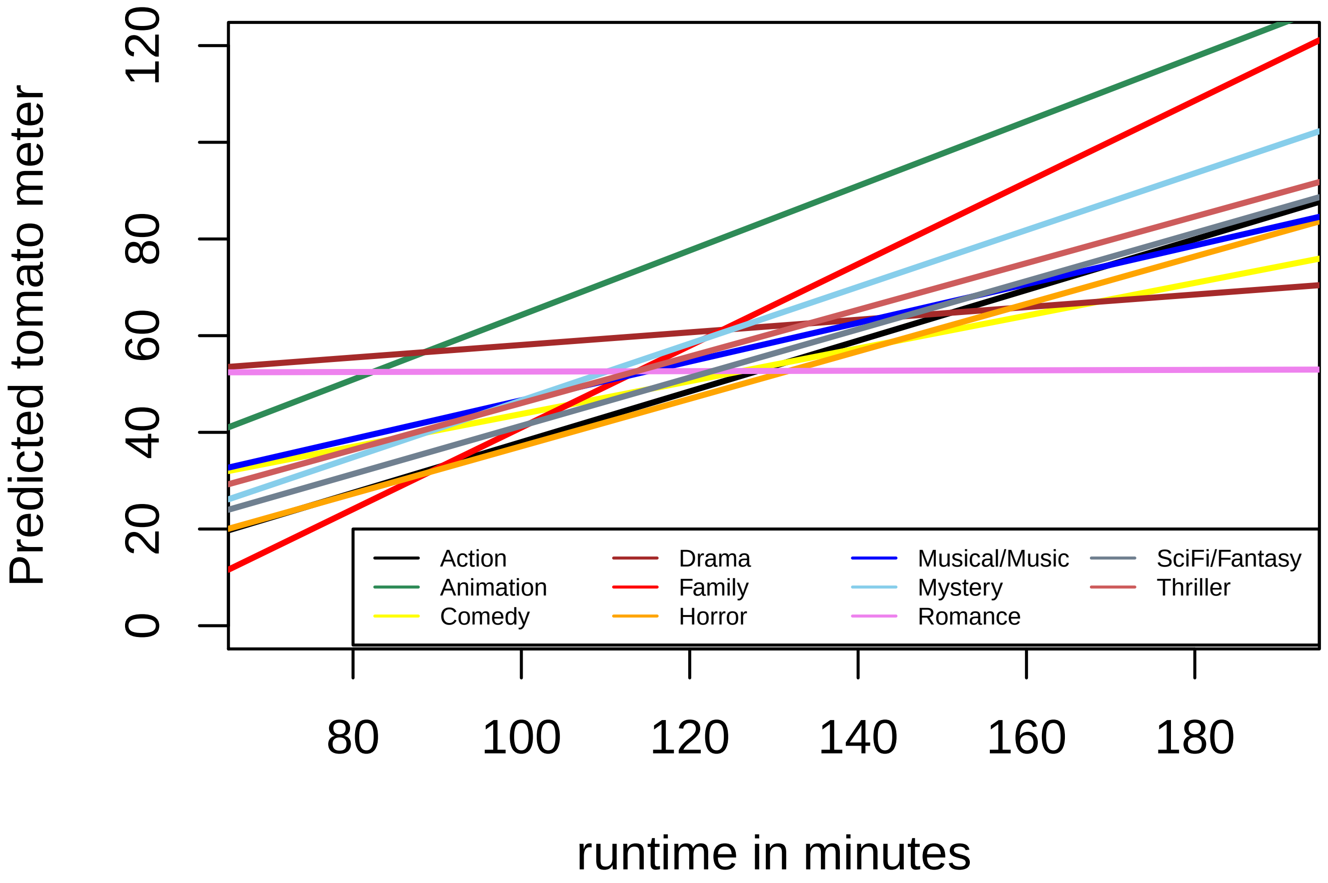
## Interaction terms with multiple categories

In the above example, I only compared comedy and action movies in order to keep the comparison simple, but it is possible to run the same analysis on the full movie dataset to see how runtime varies across all genres.

```
model <- lm(TomatoMeter~Runtime*Genre, data=movies)
round(coef(model),2)
```

```
##                (Intercept)                    Runtime
##                     -14.45                       0.52
##             GenreAnimation                GenreComedy
##                      11.96                      24.36
##                 GenreDrama                GenreFamily
##                      59.50                     -29.06
##                GenreHorror        GenreMusical/Music
##                       2.51                      21.07
##               GenreMystery               GenreRomance
##                       2.30                      66.59
##         GenreSciFi/Fantasy              GenreThriller
##                       5.92                      12.26
##     Runtime:GenreAnimation        Runtime:GenreComedy
##                       0.14                      -0.19
##         Runtime:GenreDrama        Runtime:GenreFamily
##                      -0.39                       0.32
##        Runtime:GenreHorror Runtime:GenreMusical/Music
##                      -0.03                      -0.12
##       Runtime:GenreMystery       Runtime:GenreRomance
##                       0.06                      -0.52
## Runtime:GenreSciFi/Fantasy      Runtime:GenreThriller
##                      -0.03                      -0.04
```

Thats a lot of numbers! There is a slope for each genre except action (10 in all) and an interaction between runtime and each genre except action (another 10 in all). What we are estimating here are 11 different lines (on for each genre) for the relationship between runtime and Tomato Meter rating. Because action movies are the reference, the main effect of runtime is the slope for action movies (0.52). The interaction terms show us how much larger or smaller the effect of runtime is for each given genre. So the effect is 0.39 smaller for dramas for a total effect of 0.13 (0.52-0.39). It is 0.32 larger for family movies for a total effect of 0.84 (0.52+0.32), and so forth. Similarly, the intercept is the intercept only for action movies. To get the intercept for other genres, we take the intercept value itself and add the main effect of genre. So for dramas the intercept is -14.45+59.5=45.05 and for family movies it is -14.45-29.06=-43.51. If we put all these slopes and intercepts together, we will get 11 lines like so:



There is a lot going on here, but we can detect some interesting patterns. Almost all of the lines are positive indicating that longer movies tend to generally get better ratings. This is not true of Romances however, where there is a slight negative relationship between movie runtime and Tomato Meter ratings. Dramas also have a fairly flat slope and a high intercept, so they tend to outperform most other short movies but don't fare as well compared to other genres when they are longer. The steepest slope is for family movies, which apparently are horrible when short (think "Beethoven 6: Beethoven saves Christmas, again" or something), but do much better when longer (although I can't think of many examples here). SciFi/Fantasy movies have the highest ratings at every single runtime, although they don't get quite as much return from runtime as family movies.

## Interaction terms with two categorical variables

The examples so far have involved interacting a quantitative variable with a categorical variable which gives you a different line for each category of your categorical variable. However, we can also create an interaction term between two categorical variables.

As an example, lets look at differences in income in the politics dataset by race and education. To simplify things, I am going to dichotimize race into white/non-white and education into less than college/college or more, as follows:

```r
politics$nwhite <- politics$race!="White"
politics$college <- as.numeric(politics$educ)>3
```

Lets look at the mean income across these combination of categories:

```r
tapply(politics$income, politics[,c("nwhite","college")], mean)
```

```
##        college
## nwhite     FALSE     TRUE
##   FALSE 59.44806 102.9907
##   TRUE  43.81107  97.2459
```

White college graduates make $103K, on average, while non-white college graduates make $96K, on average. Whites without a college degree make $59K, on average, while non-whites without a college degree make $43K, on average. If we put this in a table, I can show that there are four different ways to make comparisons between these numbers.

| | No degree | College degree | Difference |
|---|---|---|---|
| **White** | 59.4 | 103 | 43.6 |
| **Non-white** | 43.8 | 97.2 | 53.4 |
| **Difference** | -15.6 | -5.8 | 9.8 |

If we look at the two differences along the far-right column, we are seeing the "returns" in terms of income for a college degree separately for whites and non-whites. The return for whites is $43.4K and the return for non-whites is higher at $52K. if we look at the differences along the bottom row, we are seeing the racial inequality in income separately for those with no degree and those with a college degree. Among those with no college degree, non-whites make $15.9K less than whites. Among those with a college degree, non-whites make $7.3K less than whites. The racial gap in income gets smaller at the higher level of education.

Now lets look at the difference in the differences. For the racial gap in income this is given by -7.3-(-15.9)=8.6. For the returns to a college degree this is given by 52-43.4=-8.6. The difference in the differences is the same! This is because we are looking at the same relationship in two different ways. If non-whites get a better return to college than whites, then the racial gap in income must get smaller among the college-educated. Similarly, if the racial gap in income gets smaller at the college level, it tells us that non-whites must get a better return on their college education.

This 8.6 number is basically an interaction term. We can interpret the number as the difference in returns to income from a college degree between whites and non-whites. Alternatively, we can interpret the number as the difference in the racial income gap between those with no degree and those with a college degree. Either way, we have the same information, with the same finding: greater educational attainment reduces racial inequality because minorities get greater return on their college degrees.

Lets try modeling this relationship with an OLS regression model. First lets try a model without interaction terms:

```r
model <- lm(income~nwhite+college, data=politics)
coef(model)
```

```
## (Intercept)  nwhiteTRUE collegeTRUE 
##    58.37914   -12.31446    46.06442
```

Lets put this into an equation framework:

$$\hat{income_i} = 58.5 - 13.0(nwhite_i) + 45.6(college_i)$$

We can use this equation to fill in the predicted valued of the same table we calculated by hand above:

| | No degree | College degree | Difference |
|---|---:|---:|---:|
| **White** | 58.5 | 58.5+45.6=104.1 | 45.6 |
| **Non-white** | 58.5-13.0=45.5 | 58.5-13+45.6=91.1 | 45.6 |
| **Difference** | -13 | -13 | 0 |

The predicted values do not match the exact values above. More importantly, if you look at the differences, you can see that the returns to education are assumed to be identical for whites and non-whites (45.6) and the racial gap is assumed to be the same for those with no degree and those with a college degree (-13). This is the limitation of the additive model. We assume that the effects of race and college completion are not affected by each other. If we want to determine whether returns to college are different by race, we need to model the interaction term, as follows:

```r
model <- lm(income~nwhite*college, data=politics)
coef(model)
```

```
##            (Intercept)             nwhiteTRUE            collegeTRUE 
##               59.44806              -15.63699               43.54263 
## nwhiteTRUE:collegeTRUE 
##                9.89220
```

In equation form:

$$\hat{income_i} = 59.4 - 15.9(nwhite_i) + 43.4(college_i) + 8.6(nwhite_i * college_i)$$

Lets use this model to get predicted values in our table:

| | No degree | College degree | Difference |
|---|---:|---:|---:|
| **White** | 59.4 | 59.4+43.4=102.8 | 43.4 |
| **Non-white** | 59.4-15.9=43.5 | 59.4-15.9+43.4+8.6=95.5 | 52 |
| **Difference** | -15.9 | -7.3 | 8.6 |

Our model now fits the data exactly and the differences are allowed to vary by the other category, so that we can see the differences in returns to college by race and the differences in the racial gap by education level. The interaction term itself of 8.6 is basically the same to what we calculated by hand.

If we were to interpret the intercept and slopes from the model above, we would say:

- Whites with no college degree had a mean income of $59,400.
- Among those with no college degree, non-whites earn $15,900 less than whites, on average.
- Among whites, those with a college degree have incomes $43,400 higher on average than those without a college degree.
- The returns to income from a college degree are $8,600 larger for non-whites than they are for whites, on average.

# Appendix A

# Using Scripts

One of the primary advantages of using statistical software like *R* or *Stata* is the ability to perform your analysis in a script. A script is a text file of commands that can be run to reproduce a data cleaning exercise, an analysis, a simulation, etc. It is a good practice to **always** do your statistical work using scripts. The advantage of scripting is that you will have an easily reproducible record of exactly what you did. Lets say that I did a thorough and time-consuming statistical analysis in Excel or (god forbid) SPSS using pull-down menus and buttons and I found The Best Result Ever. I quickly put together a paper and submit it to a flagship journal. My R&R comes back several months later and the reviewers say that my results look great, but they want to see if the results hold up if I limit my sample in some way. Now I have to (a) try to remember exactly what I did months ago, and (b) re-run that entire time-consuming analysis from scratch. That is horribly inefficient and is likely to lead to inconsistencies between the two analyses. Imagine that instead of cowboying my analysis, I had written everything into an R or Stata script. Now, I only have to add a line of code at the top that subsets my sample and processes the script. This will take a few minutes at most.

## Getting Started with Scripts

Lets put together a simple script in R. In RStudio, you can go to `File > New File > R Script` to get a new script up and running. This will open a text pane in the upper left corner of RStudio. This is your R script. Lets start by writing a simple "Hello World" script. Type the following command into your script:

```r
cat("Hello World!")
2+2
```

You can run this script a couple of different ways:

- You could just copy and paste both lines of code to the console in the lower left. This is quick and dirty but is typically not the most efficient way to run code from your script.
- You could run your entire script by clicking the "Source" button in the upper right corner of the script pane. Note that with this button, you won't get the output, unless you use the drop-down arrow to choose "Source with Echo".
- You could run a single line of your script where the cursor is located. You can do this by either hitting the "Run" button or by clicking Ctrl+Enter on Windows/OSX or Command+Enter on OSX. When you do execute the code this way, your cursor will automatically move to the next line of code, so you can execute your way through the code simply by clicking Ctrl+Enter repeatedly.
- If you have saved the script to a file, you can also type in the `source` command in the console to source the script.

Lets run this script using the "Source with Echo" button. Here is what the output looks like in my console:

```
> source('~/.active-rstudio-document', echo=TRUE)

> cat("Hello World")
Hello World
> 2+2
[1] 4
```

Now, lets save this script as `helloworld.R` (".R" is the expected suffix for R scripts) and source it from the console:

```
source("resources/helloworld.R", echo=TRUE)
```

```
##
## > cat("Hello World")
## Hello World
## > 2+2
## [1] 4
```

Now lets try a slightly more useful script. In this script, I am going to do the following:

- load in the politics dataset
- re-code the gay marriage support variable into support for gay marriage vs. all else, and the religion variable as evangelical vs. all else
- Create a two-way table (crosstab) of these two new variables.
- Calculate the odds ratio between the two new variables.

I don't expect you to know how all of this code works yet. I just want to show you an example of a script that actually does something interesting.

```
load("example_datasets/politics/politics.RData")
politics$supportgmar <- politics$gaymarriage=="Support gay marriage"
politics$evangelical <- politics$relig=="Evangelical Protestant"
tab <- table(politics$supportgmar, politics$evangelical)
tab
prop.table(tab, 2)
OR <- tab[1,1]*tab[2,2]/(tab[1,2]*tab[2,1])
OR
```

If I run this script, I will get:

```
source("resources/politics.R", echo=TRUE)
```

```
##
## > load("example_datasets/politics/politics.RData")
##
## > politics$supportgmar <- politics$gaymarriage=="Support gay marriage"
##
## > politics$evangelical <- politics$relig=="Evangelical Protestant"
##
## > tab <- table(politics$supportgmar, politics$evangelical)
##
## > tab
##
##         FALSE TRUE
##   FALSE  1103  662
##   TRUE   2218  255
##
## >  prop.table(tab, 2)
```

```
##
##               FALSE      TRUE
##   FALSE 0.3321289 0.7219193
##   TRUE  0.6678711 0.2780807
##
## > OR <- tab[1,1]*tab[2,2]/(tab[1,2]*tab[2,1])
##
## > OR
## [1] 0.1915562
```

About 49% of non-evangelicals supported gay marriage while only 24% of evangelicals supported it. That works out to an odds ratio of 0.33, meaning that the odds of gay marriage support were about a third as high for evangelicals as for non-evangelicals.

Lets say I then wanted to change this script to broaden the definition of support for gay marriage to include civil unions. I could then just change the line of code creating the `supportgmar` variable to:

```
politics$supportgmar <- politics$gaymarriage=="Support gay marriage" | politics$gaymarriage=="Civil uni
```

If I re-run this new script, I will get:

```
source("resources/politics2.R", echo=TRUE)
```

```
##
## > load("example_datasets/politics/politics.RData")
##
## > politics$supportgmar <- politics$gaymarriage=="Support gay marriage" | politics$gaymarriage=="Civil
##
## > politics$evangelical <- politics$relig=="Evangelical Protestant"
##
## > tab <- table(politics$supportgmar, politics$evangelical)
##
## > tab
##
##           FALSE TRUE
##   FALSE     368  405
##   TRUE     2953  512
##
## > prop.table(tab, 2)
##
##               FALSE      TRUE
##   FALSE 0.1108100 0.4416576
##   TRUE  0.8891900 0.5583424
##
## > OR <- tab[1,1]*tab[2,2]/(tab[1,2]*tab[2,1])
##
## > OR
## [1] 0.1575431
```

The percentage of overall support gets higher, of course, but the odds ratio between evangelicals and everyone else stays about the same.

## Not Everything Goes Into Your Script

You don't need to put every command into your script. The script should provide a narrative of your analysis (the part that you want to be easily reproducible), not a log of every single command you ran. Sometimes you may try out some exploratory commands or may just need to get some basic information about a variable. For example, in the script above, I first had to remember what the names were for the categories of the `gaymarriage` variable. To figure this out, I typed:

```
table(politics$gaymarriage)
```

```
##
## No legal recognition          Civil unions Support gay marriage
##                  773                   992                 2473
```

From there I could see that the category I wanted was called “Support gay marriage” and I could use that in my script. However, there was no need to put this command into my script because it wasn’t producing anything that I needed to know or that was necessary for later commands in the script.

## Commenting for Sanity

There is one big thing missing from the scripts listed above: comments! Comments are crucial for good script writing. Comments are lines in your script that will not be processed by R (or Stata). In R, you can create single line comments by using the pound sign (#). Anything after the pound sign will be ignored by R until a new line. You should use these comments to explain what the code is doing. You can also use them to help visually separate the script into sections for easier readability. Comments help you remember what you were doing when you come back to a project you haven’t worked on for weeks or months. They are also useful for other people who might end up reading your code (co-authors, advisers, etc). Increasingly, academics are being asked to do more “open” research where we make our code available. As you write code, its useful to think of it as something that will eventually be seen by other people and thus needs to be well documented.

Here is the script from above, but now with some helpful commenting:

```
#################################################################
# Aaron Gullickson
# Program to analyze the differences in support for gay marriage
# between evangelical christians and all those of other religious
# affiliations
#################################################################

#### DATA ORGANIZATION ####

#load the politics dataset. Note that this command will not work if
#you have the dataset loaded into a different directory.
load("example_datasets/politics/politics.RData")

#dichotomize both the support for gay marriage and the religious variable
politics$supportgmar <- politics$gaymarriage=="Support gay marriage"
politics$evangelical <- politics$relig=="Evangelical Protestant"

#### ANALYSIS ####

#create a crosstab
tab <- table(politics$supportgmar, politics$evangelical)
tab

#Distribution of support conditional on religion
prop.table(tab, 2)

#Calculate the odds ratio of support
```

```
OR <- tab[1,1]*tab[2,2]/(tab[1,2]*tab[2,1])
OR
```

Even if you don't understand what the code here is doing yet, you can at least get a sense of what it is supposed to be doing. Notice that I use multiple pound signs to draw attention to the header and larger sections. For a script this small, the division between DATA ORGANIZATION and ANALYSIS is probably overkill, but for larger scripts, this sectioning can be very useful in helping to easily distinguish different components of the analysis.

One nice feature of RStudio is that if you surround your comments with at least four pound signs on either side, then the outline of your script will show these sections and you can navigate to them. You can also go to `Code > Insert Section` to add a section with a somewhat different look.

### Don't Overcomment

While commenting is necessary for good script writing, *more* commenting is not always better. The key issue is that you need to think of commenting as documentation of your code. If you change the code, you also need to change the documentation. If you change the code, but not the documentation, then you will actually make your code more confusing to other readers. **Only write documentation that you will actively maintain.**

The most common error that novices make is to use comments to describe the results of their code. This is very bad practice, because as you make changes to your data and methods, these results are likely to change and if you don't update your comments, there will be a lack of alignment between your real results and your documentation of the results. Later in the term, we will learn a better way to separate out the reporting of results in a "lab notebook" fashion, from comments in scripts.

# Appendix B

# Object Types

R is an object-oriented programming language. This means that within R, you can save a variety of objects to memory. These objects will show up in the upper right panel in the environment tab in RStudio when you create them. A variety of functions can be applied to objects and in many cases, the same function may produce different results when applied to different kinds of objects. We will work most commonly with the `data.frame` object, but it is useful to know some other basic object types to understand how R works.

Objects can have different types and R will often handle them differently depending on their type. However, type is defined in two different ways. Each object has a `mode` and a `class`. The class of an object is typically what determines how it is handled by other functions. The `mode` is most important in terms of the "atomic" or "basic" modes of R objects. These are the basic-building blocks of everything else.

## Atomic Modes

The three most important atomic modes for our purposes are:

- `numeric`: records a numeric value.
- `character`: records a set of characters. This is often called a "string" in computer science parlance.
- `logical`: records either a TRUE or FALSE value. In computer science parlance, this is called a "boolean."

There is also a fourth type `complex` for things like imaginary numbers, but we won't really need to worry about that. Lets try assigning values to an object of each mode:

```
a <- 3
b <- "bob said"
c <- TRUE
a
```

```
## [1] 3
```

```
mode(a)
```

```
## [1] "numeric"
```

```
class(a)
```

```
## [1] "numeric"
```

```
b
```

```
## [1] "bob said"
```

```
mode(b)
```

```
## [1] "character"
```

```
class(b)
```

```
## [1] "character"
```

```
c
```

```
## [1] TRUE
```

```
mode(c)
```

```
## [1] "logical"
```

```
class(c)
```

```
## [1] "logical"
```

note that for these simple objects the mode is the same as the class. These objects will now show up in my upper right panel. However, programming with these simple objects with one value is not very useful. What we usually really want is an aggregation of many of these values. There are a variety of ways we can do this.

## Vectors and Matrices

### Vectors

If you want to put a bunch of values of the same mode together (don't call it a "list" because that means something else, see below), you can do that with a `vector`. You can create a vector with the concatenation function `c`. Lets do that below:

```
name <- c("Bob","Juan","Maria","Jane","Howie")
age <- c(15,25,19,12,21)
ate_breakfast <- c(TRUE,FALSE,TRUE,TRUE,FALSE)
name
```

```
## [1] "Bob"   "Juan"  "Maria" "Jane"  "Howie"
```

```
age
```

```
## [1] 15 25 19 12 21
```

```
ate_breakfast
```

```
## [1]  TRUE FALSE  TRUE  TRUE FALSE
```

```
mode(name)
```

```
## [1] "character"
```

```
class(name)
```

```
## [1] "character"
```

```
mode(age)
```

```
## [1] "numeric"
```

```
class(age)
```

```
## [1] "numeric"
```

```
mode(ate_breakfast)
```

```
## [1] "logical"
```

```
class(ate_breakfast)
```

```
## [1] "logical"
```

Note that the `mode` and `class` of each vector is given by the `mode` of the values that makes up the vector. As you can see, vectors are basically equivalent to variables for the purpose of data analysis. We can even calculate values like the mean for numeric and logical vectors:

```
mean(age)
```

```
## [1] 18.4
```

```
mean(ate_breakfast)
```

```
## [1] 0.6
```

How can you calculate a mean for a logical vector? R automatically converts logical values to numeric values where `TRUE=1` and `FALSE=0` when it seems like a numeric value is needed. Therefore, the mean is tellling us that 60% of respondents ate breakfast.

You can also force vectors (and some other objects) into a different basic type:

```
as.character(age)
```

```
## [1] "15" "25" "19" "12" "21"
```

```
as.numeric(ate_breakfast)
```

```
## [1] 1 0 1 1 0
```

```
as.numeric(name)
```

```
## Warning: NAs introduced by coercion
```

```
## [1] NA NA NA NA NA
```

Note that this didn't work so well when converting a character string to a number and I ended up with a set of missing values (NA).

## Matrices

A `matrix` is just an extention of a vector, but two dimensional instead of one dimensional. We can use the `matrix` command to turn a vector into a matrix, by speficying the number of rows and columns.

```
x <- matrix(c(4,5,3,9,7,8), 3, 2)
x
```

```
##      [,1] [,2]
## [1,]    4    9
## [2,]    5    7
## [3,]    3    8
```

```
mode(x)
```

```
## [1] "numeric"
```

```
class(x)
```

```
## [1] "matrix"
```

I created a matrix of numeric values with three rows and two columns. Notice that when putting all of these values, R tries to fill up each column before moving on to the next column. Its also worth noting that the mode and class of my matrix are no longer the same. This means that I can specify specific functions that will apply to the class of `matrix`. The mode tells me what kind of values I have within the matrix.

I can also create a matrix by binding together vectors into different rows (`rbind`) or columns (`cbind`).

```
a <- c(4,5,3)
b <- c(9,7,8)
cbind(a,b)
```

```
##      a b
## [1,] 4 9
## [2,] 5 7
## [3,] 3 8
```

```
rbind(a,b)
```

```
##   [,1] [,2] [,3]
## a    4    5    3
## b    9    7    8
```

This might seem like a good way to create a full dataset from the variables I created above, but there is a problem:

```
cbind(name, age, ate_breakfast)
```

```
##      name    age  ate_breakfast
## [1,] "Bob"   "15" "TRUE"
## [2,] "Juan"  "25" "FALSE"
## [3,] "Maria" "19" "TRUE"
## [4,] "Jane"  "12" "TRUE"
## [5,] "Howie" "21" "FALSE"
```

A matrix has to have values of the same atomic mode. In most cases, if there is any character vector in the binding, then everything will get converted to characters. We will see a better way to create a dataset (spoiler: the `data.frame` object) below.

Note there is an extension to the matrix object called the `array` which generalizes it to n-dimensions rather than two. However, we will not make much use of that in this class.

## Indexing Vectors and Matrices

What if I want to know a specific value in my vector or matrix. Lets say I want to know the name of the fourth person in my `name` vector. You can easily get this value by indexing the vector or matrix with square brackets. In my case:

```
name[4]
```

```
## [1] "Jane"
```

Because a vector only has one dimension, I only need one index. In the case of matrices, you will need two numbers, separated by a comma. If you want to get an entire row or column, you can leave one of the indices blank, but you still need the comma.

```
x
```

```
##      [,1] [,2]
## [1,]    4    9
## [2,]    5    7
## [3,]    3    8
```

```
#value in 2nd row, 1st column
x[2,1]
```

```
## [1] 5
```

```
#2nd row
x[2,]
```

```
## [1] 5 7
```

```
#1st column
x[,1]
```

```
## [1] 4 5 3
```

```
#1st and 2nd row
x[c(1,2),]
```

```
##      [,1] [,2]
## [1,]    4    9
## [2,]    5    7
```

## Factors

You will note that we don't yet have a representation for categorical variables. Lets say for example that I wanted to include highest degree received for my respondents from above. I could create this as a character variable:

```
high_degree <- c("Less than HS", "College", "HS Diploma", "HS Diploma", "College")
summary(high_degree)
```

```
##    Length     Class      Mode 
##         5 character character
```

However, this is not very satisfying because there is very little that can be done with categorical variables. As you see the `summary` command failed to produce anything useful.

What I want to do is turn this into a `factor`. A factor in R is the standard object for coding categorical variables. Each value is actually recorded with a mode of "numeric" but the object also contains a set of labels that provide the meaning of each level. Almost all functions in R will know how to handle these factor objects correcly.

To create a factor object, I can just apply the `factor` function to my vector:

```
high_degree_fctr <- factor(high_degree)
levels(high_degree_fctr)
```

```
## [1] "College"      "HS Diploma"   "Less than HS"
```

```
summary(high_degree_fctr)
```

```
##      College   HS Diploma Less than HS 
##            2            2            1
```

```
mode(high_degree_fctr)
```

```
## [1] "numeric"
```

```r
class(high_degree_fctr)
```

```
## [1] "factor"
```

Now the summary command gives me a table of frequencies for each category.

### Re-ordering categories in factors

The only problem with my factor is that this is an ordinal variable and the categories are backwards with "College" first and "Less than HS" last. This is because R sorts alphabetically by default. In order to ensure a specific order to the categories in the factor, I will need to specify the `levels` argument in the `factor` function and explicitly write out the order I want:

```r
high_degree_fctr <- factor(high_degree,
                           levels=c("Less than HS","HS Diploma","College"))
levels(high_degree_fctr)
```

```
## [1] "Less than HS" "HS Diploma"   "College"
```

```r
summary(high_degree_fctr)
```

```
## Less than HS   HS Diploma      College
##            1            2            2
```

Another option is to use the `relevel` function on a factor to just change the very first level of a factor to something else:

```r
levels(high_degree_fctr)
```

```
## [1] "Less than HS" "HS Diploma"   "College"
```

```r
levels(relevel(high_degree_fctr,"HS Diploma"))
```

```
## [1] "HS Diploma"   "Less than HS" "College"
```

## Logical Values and Boolean Statements

One of the most important features of all computer programming languages is the ability to create statements that will either evaluate to a "boolean" value of TRUE or FALSE (a "logical" value in R parlance). These kinds of statements are called boolean statements. The following basic operators will allow you to make boolean statements in R.

| Operator | Meaning |
|---|---|
| == | equal to |
| != | not equal to |
| < | less than |
| > | greater than |
| >= | less than or equal |
| <= | greater than or equal |

Note that the "equal to" syntax is a double-equals. This is because the single equals is used for assignment of values to objects.

As a simple example, lets say that I wanted to identify all respondents from my data above that were 18 years of age or older:

```r
age>=18
```

```
## [1] FALSE  TRUE  TRUE FALSE  TRUE
```

You can use factor variables in boolean statements of equality as well, but you need to use the character string variables. Lets say I want to identify all respondents with a college degree:

```r
high_degree=="College"
```

```
## [1] FALSE  TRUE FALSE FALSE  TRUE
```

A very important feature of boolean statements is the ability to string together multiple boolean statements with a & (AND) or | (OR) in order to make compound statement. Lets say I wanted to identify all respondents who had either a high school diploma or a college degree:

```r
high_degree=="College" | high_degree=="HS Diploma"
```

```
## [1] FALSE  TRUE  TRUE  TRUE  TRUE
```

Lets say I want to find all respondents who are between the ages of [20,25):

```r
age>=20 & age<25
```

```
## [1] FALSE FALSE FALSE FALSE  TRUE
```

You can also use parenthesis to ensure that your compound boolean statements are interpreted in the correct order.

```r
(age>=20 & age<25) & (high_degree=="College" | high_degree=="HS Diploma")
```

```
## [1] FALSE FALSE FALSE FALSE  TRUE
```

Another useful option is the ability to put a `!` sign in front of a boolean variable to indicate "not". Lets say I wanted to find all respondents who had NOT eaten breakfast:

```r
!ate_breakfast
```

```
## [1] FALSE  TRUE FALSE FALSE  TRUE
```

## Missing Values

Missing values are a common feature of most real-world data. They can exist for a variety of reasons, but *item non-response* (i.e. respondent declined to answer a specific question) is one of the most common reasons. In R, missing values are represented with the `NA` value. missing values can exist for any of the modes we have discussed.

Lets insert a missing value into the age vector that we have been using:

```r
age[4] <- NA
age
```

```
## [1] 15 25 19 NA 21
```

Watch what happens now when we try to calculate the mean of age:

```r
mean(age)
```

```
## [1] NA
```

The mean is missing! This is the default behavior for many functions in R. If the values you feed in have missing values, R will return a missing value. R wants to be sure that you explicitly decide how to treat missing values. In Soc 513, we will learn about other ways of dealing with missing values, but our approach this term will be to simply remove observations that have missing values (i.e. *casewise deletion*) and then calculate the appropriate statistics. In many of the functions in R, this can be accomplished by setting the argument `na.rm` to TRUE:

```
mean(age, na.rm=TRUE)
```

```
## [1] 20
```

Another useful function to know is the `is.na` function. If you feed in a vector of values, this function will return a logical vector that evaluates to TRUE if a value is missing.

```
is.na(age)
```

```
## [1] FALSE FALSE FALSE  TRUE FALSE
```

Combining this with the `!` from the previous section, we can easily create a function that tells us which observations have non-missing values:

```
!is.na(age)
```

```
## [1]  TRUE  TRUE  TRUE FALSE  TRUE
```

## Lists

Lists are one of the most flexible types of standard objects. Lists are just collections of different objects and the objects can be of different types and dimensions. You can even put lists into lists and end up with lists of lists.

Lets put our four variables into a list:

```
my_list <- list(name, age, ate_breakfast, high_degree_fctr)
my_list
```

```
## [[1]]
## [1] "Bob"   "Juan"  "Maria" "Jane"  "Howie"
##
## [[2]]
## [1] 15 25 19 NA 21
##
## [[3]]
## [1]  TRUE FALSE  TRUE  TRUE FALSE
##
## [[4]]
## [1] Less than HS College      HS Diploma   HS Diploma   College
## Levels: Less than HS HS Diploma College
```

```
mode(my_list)
```

```
## [1] "list"
```

```
class(my_list)
```

```
## [1] "list"
```

In this case, each item in the list is a vector of the same length but they don't need to be.

## Accessing elements of the list

You will notice a lot of brackets in the list output above. To access an object at a specific index of the list, I need to use double square brackets. Lets say, I wanted to access the third object (ate_breakfast):

```
my_list[[3]]
```

```
## [1]  TRUE FALSE  TRUE  TRUE FALSE
```

If I want to access a specific element of that vector, I can follow up that double bracket indexing with single indexing:

```
my_list[[3]][4]
```

```
## [1] TRUE
```

My fourth respondent did eat breakfast. Good to know.

There is another way to access objects within the list but to do this, I need to provide a name for each object in the list. I can do this within the initial list command by using a `name=value` syntax for each object:

```
my_list <- list(name=name, age=age, ate_breakfast=ate_breakfast, high_degree=high_degree_fctr)
```

Now, I can call up any object by its name with the basic syntax `list_name$object_name`. Lets do that for age:

```
my_list$age
```

```
## [1] 15 25 19 NA 21
```

```
mean(my_list$age, na.rm=TRUE)
```

```
## [1] 20
```

You will notice in RStudio that when you type the "$", it brings up a list of all the names you could want. You can select the one you want and tab to complete. Thanks, RStudio!

## The `lapply` command is awesome

Lets say that I just want to get a summary of each object in my list:

```
summary(my_list)
```

```
##               Length Class  Mode
## name          5      -none- character
## age           5      -none- numeric
## ate_breakfast 5      -none- logical
## high_degree   5      factor numeric
```

Well, that was not super helpful. R is giving me a summary of the list itself rather than a summary of each object in the list. What if I want to apply the same function to every object in the list? This is exactly what the `lapply` command does. Feed in a list and a function (even a custom function) and it will sequentally apply that function to each object in the list:

```
lapply(my_list, summary)
```

```
## $name
##    Length     Class      Mode
##         5 character character
##
## $age
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
##      15      18      20      20      22      25       1 
## 
## $ate_breakfast
##    Mode   FALSE    TRUE 
## logical       2       3 
## 
## $high_degree
## Less than HS   HS Diploma      College 
##            1            2            2
```

Thats much better. It won't be immediately obvious, but `lapply` turns out to be a very powerful and helpful tool. We will learn more about it later this term.

## Data Frames

The list was a nice way to organize my data, but it wasn't ideal because it didn't represent data in the two-dimensional observations-on-the-row and variables-on-the-columns way we expect most datasets to be typical organized. This is where the `data.frame` object comes in. This is the object that we will mostly work directly with in this class and the one you are most likely to work with in real projects.

The `data.frame` object is basically a special form of a list in which each object in the list is required to be a vector of the same length, but not necessarily the same mode and class. The results can be displayed like a matrix and the same kinds of options for indexing that are available for matrices can be used on data.frames.

Lets put the variables into a data.frame:

```r
my_data <- data.frame(name, age, ate_breakfast, high_degree=high_degree_fctr)
my_data
```

```
##    name age ate_breakfast  high_degree
## 1   Bob  15          TRUE Less than HS
## 2  Juan  25         FALSE      College
## 3 Maria  19          TRUE   HS Diploma
## 4  Jane  NA          TRUE   HS Diploma
## 5 Howie  21         FALSE      College
```

Now that looks like a dataset. Note that by default it just used the name of the object as the column name. However, I specifically changed this behavior for high_degree.

We can run a summary on the whole dataset now:

```r
summary(my_data)
```

```
##     name        age     ate_breakfast         high_degree
##  Bob  :1   Min.   :15   Mode :logical   Less than HS:1  
##  Howie:1   1st Qu.:18   FALSE:2         HS Diploma  :2  
##  Jane :1   Median :20   TRUE :3         College     :2  
##  Juan :1   Mean   :20                                   
##  Maria:1   3rd Qu.:22                                   
##            Max.   :25                                   
##            NA's   :1
```

It is now treating the name variable like a factor. By default, R will convert all character variables into factors when including them in a `data.frame`.

I can also access any specific variable with the same "$" syntax as for lists:

```
mean(my_data$age, na.rm=TRUE)
```

```
## [1] 20
```

I can also use the same indexing as for matrices to retrieve particular values:

```
my_data[3,2]
```

```
## [1] 19
```

## Renaming Variables in a `data.frame`

Vectors, matrices, lists, and data.frames can all have names associated with their elements. You have already seen an example of specifying names in the creation of the list and data.frame objects. But you can also change names of elements after creation. We will focus here specifically on the case of data frames, but much of this is generalizable to other objects as well.

the `rownames` and `colnames` command can be used to both retrieve and set the row and column names, respectively for a data frame. The `colnames` command is generally more useful because we typically care more about variable names than observation names.

```
colnames(my_data)
```

```
## [1] "name"          "age"           "ate_breakfast" "high_degree"
```

Lets say I decided that I wanted to have all my variable names capitalized. I can easily change this by just assigning a new character vector to this colnames command:

```
colnames(my_data) <- c("Name","Age","Ate_breakfast","High_degree")
my_data
```

```
##    Name Age Ate_breakfast  High_degree
## 1   Bob  15          TRUE Less than HS
## 2  Juan  25         FALSE      College
## 3 Maria  19          TRUE   HS Diploma
## 4  Jane  NA          TRUE   HS Diploma
## 5 Howie  21         FALSE      College
```

I can also use indexing of the colnames to just change specific variable names. Lets say I decided that "Ate_breakfast" was too long:

```
colnames(my_data)[3] <- "Breakfast"
my_data
```

```
##    Name Age Breakfast  High_degree
## 1   Bob  15      TRUE Less than HS
## 2  Juan  25     FALSE      College
## 3 Maria  19      TRUE   HS Diploma
## 4  Jane  NA      TRUE   HS Diploma
## 5 Howie  21     FALSE      College
```

## Subsetting Data Frames

One of the most common tasks of organizing data is slicing and dicing datat into smaller subsets. There are two cases that are common.

### Subsetting Observations

You may want only a subset of observations by some logical characteristics (e.g. only women, people born after 1975, people who grew up in Lake Geneva WI, member-states of the OECD). One nice feature of R is that you can use the same indexing of rows as I discussed above but instead of specific index numbers, you can provide a boolean statement and only observations that evaluate to TRUE will be kept. Lets say that you want limit the dataset we have been using to only those who completed college:

```
my_data[my_data$High_degree=="College",]
```

```
##    Name Age Breakfast High_degree
## 2  Juan  25     FALSE     College
## 5 Howie  21     FALSE     College
```

One "gotcha" with this approach is missing values in your boolean statement. Lets try to subset the data to only those respondents 18 years and older:

```
my_data[my_data$Age>=18,]
```

```
##     Name Age Breakfast High_degree
## 2   Juan  25     FALSE     College
## 3  Maria  19      TRUE  HS Diploma
## NA  <NA>  NA        NA        <NA>
## 5  Howie  21     FALSE     College
```

Notice we get one full row of NA values. This was because of the one missing value on age. To avoid this, we first need to tell R to only keep cases that do not have missing values on age:

```
my_data[!is.na(my_data$Age) & my_data$Age>=18,]
```

```
##    Name Age Breakfast High_degree
## 2  Juan  25     FALSE     College
## 3 Maria  19      TRUE  HS Diploma
## 5 Howie  21     FALSE     College
```

Now it works. However, we can also use the `subset` command to do the same thing and the `subset` command will be more robust to this missing value problem. The first argument to the `subset` command is the data frame you want to subset and the second is a boolean statement about the variables in that data frame. When this boolean statement evaluates to true for an observation it will be kept in the subset.

```
subset(my_data, Age>=18)
```

```
##    Name Age Breakfast High_degree
## 2  Juan  25     FALSE     College
## 3 Maria  19      TRUE  HS Diploma
## 5 Howie  21     FALSE     College
```

Notice that I did not have to use the full `my_data$Age` specification in my boolean statement, because the subset command knows that this variable must be in the dataset that I fed in. I also did not have to deal with the whole business of missing values. In general, the `subset` command produces cleaner and more readable code than bracketing and indexing.

## Subsetting by variables

Sometimes you want to drop variables. This may be because you have used those variables to construct another variable and you don't need them any longer or maybe your data source just contains a lot more variables than you need. I am a big proponent of dropping irrelevant variables. It keeps your dataset cleaner and easier to read for others.

You can use indexing of columns in your data frame to choose variables that you want to keep. Lets say that I only wanted to keep age and highest degree in the dataset that we have been looking at:

```r
my_data[,c("Age","High_degree")]
```

```
##   Age  High_degree
## 1  15 Less than HS
## 2  25      College
## 3  19   HS Diploma
## 4  NA   HS Diploma
## 5  21      College
```

You can also use the negative sign in front of a number index to remove it rather than keep it.

```r
my_data[,c(-1,-3)]
```

```
##   Age  High_degree
## 1  15 Less than HS
## 2  25      College
## 3  19   HS Diploma
## 4  NA   HS Diploma
## 5  21      College
```

You can also use the `select` argument in the `subset` command to keep only certain variables.

```r
subset(my_data, select=c("Age","High_degree"))
```

```
##   Age  High_degree
## 1  15 Less than HS
## 2  25      College
## 3  19   HS Diploma
## 4  NA   HS Diploma
## 5  21      College
```

You can even use the subset command to simultaneously subset observations and drop variables. Lets say that I wanted to use my `Name` variable as row names for my dataset and I also want to drop any cases that are missing on age:

```r
rownames(my_data) <- my_data$Name
my_data2 <- subset(my_data, !is.na(Age),
                   select=c("Age","Breakfast","High_degree"))
my_data2
```

```
##       Age Breakfast  High_degree
## Bob    15      TRUE Less than HS
## Juan   25     FALSE      College
## Maria  19      TRUE   HS Diploma
## Howie  21     FALSE      College
```

# Appendix C

# Pretty Pictures

One of the best features of R is the ability to make beautiful graphs. R gives the user an incredible amount of control over exactly how your graphs appear. If you know enough, you can virtually make any graph that you can imagine in your head. This flexibility comes at a cost, however. The curve in learning graphics is quite substantial, because of the number of parameters that can be controlled. Nonetheless, if you intend to do quantitative data analysis, then its well worth the effort. In my opinion, figures are almost always a better way to express a result than a table. If the figure is well-designed it will make your results much more intelligible to a broader audience because you have literally helped them to visualize it.

In this lab, I am going to show you two techniques for plotting in R. First, I am going to show you how to plot things in R using the basic plotting functions, sometimes called “base plot.” Base plotting in R gives you full control over pretty much everything you could want to stick on a canvas, but sometimes it gives you too much control and you can end up fiddling around a lot to get things to look just how you want them to. So, I am also going to show you how to plot in R using the `ggplot2` library. ggplot has become so popular as a method of plotting that many people prefer it to base plot. The “gg” in ggplot refers to teh “graphics of grammar.” The basic idea of ggplot is that rather than control everything about your plot, you give R a “grammar” of what data you want to graph, how you want to graph, and how you want to tie aesthetics to the different pieces of data. In future versions of this course, I plan to transition many of the graphs we use to `ggplot` although most slides are still written in base plot.

There are two different ways you should use pictures in your work. The first method is to use figures to **visualize what you data looks like**. This is what we have been learning already in class. Histograms, barplots, boxplots, and scatterplots are all examples of trying to visualize your data. Most work on how to make figures is devoted to the topic of visualizing your data like this. However, there is a second way to use pictures, which I think is actually equally important and under-utilized. You can use figures to **vizualize your findings**. The difference here might not be immediately obvious, partly because we haven’t talked about “models” yet, which is the primary way that results are expressed in quantitative work. Let me give you an example from my own work that may help to communicate what I mean. Here is a figure from an AJS article that shows how to think about the results of a model I ran that tries to predict the number of lynchings in a county by the inequality in occupation between individuals classified as black and mulatto and the inequality between individuals classified as white and mulatto or black (i.e. “African ancestry”).

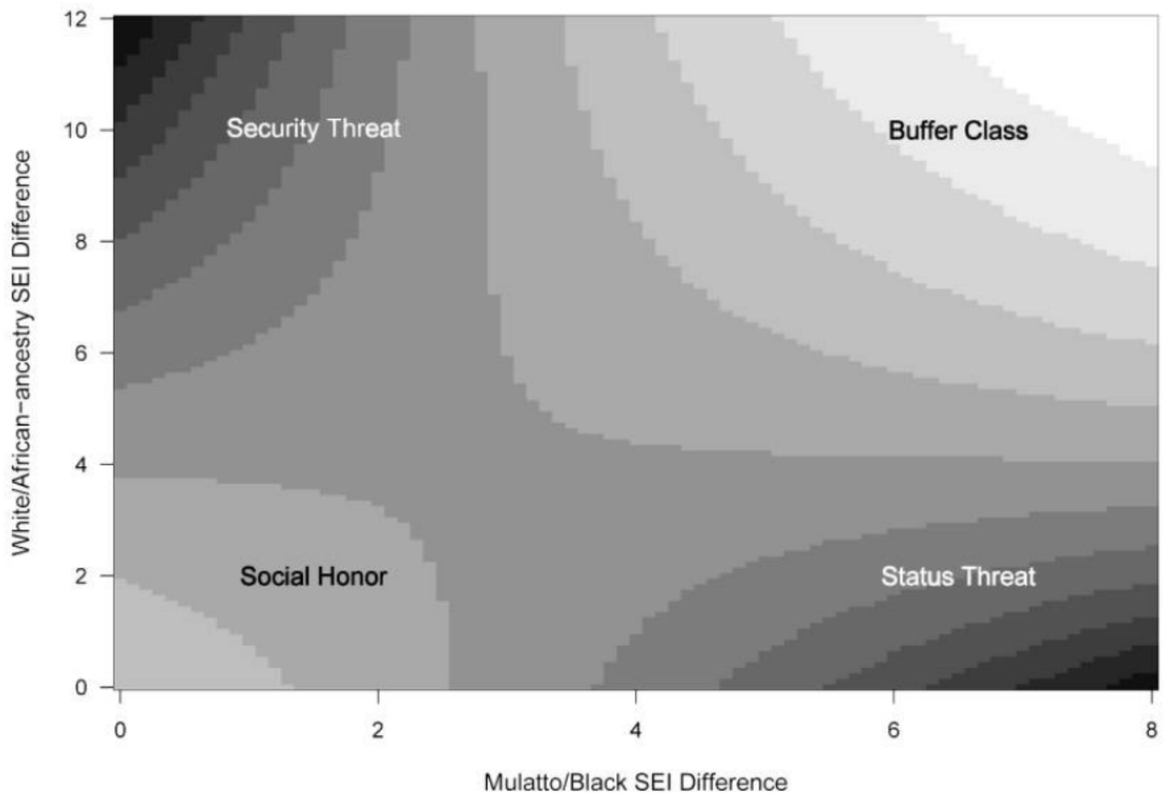


**FIG. 6.—Lynching risk as a function of mulatto/black SEI difference and white/African-ancestry SEI difference. Fitted values are based on model 4 (table 4).**

This is a heat (or topological) map of risk where dark areas indicate greater risk (of lynching). The point was to show that the effect of each variable produces opposite effects depending on the level of the other variable. I am not describing data here, but rather the results of a model of lynching risk that is based on the underlying county-level data.

Since we don't know much about these types of models yet, we will focus for now on using graphs to describe data. The data we will look at come from one of my current projects. In this project, my colleague (Nicole Marwell) and I have data on social service contracts awarded by the City of New York to non-profit organizations working within the city. The data I am using here are created by aggregating all the money allocated to a particlar health area (a bureaucractic neighborhood boundary in NYC) between the years of 2009 and 2010. We combine that data with data on the poverty rate for each health areas and also divide by the population in a health area to get an estimate of the **social service funding per capita.** Here is a peak at what the data look like:

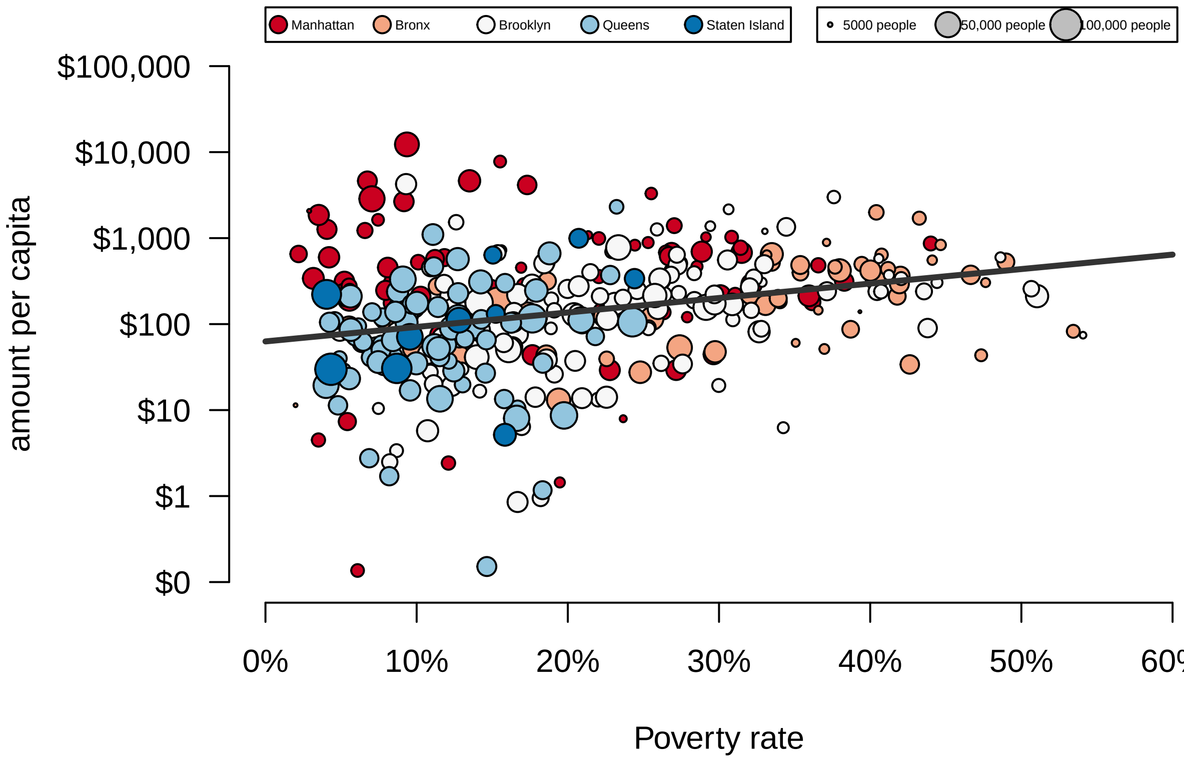
```r
head(nyc)
```

```
##   health_area  amtcapita  poverty unemployed   income   borough     popn
## 1       10110  29.168592 22.77850  11.324410 40287.02 Manhattan 27983.43
## 2       10120 109.451055 22.62417  10.179587 43266.45 Manhattan 20235.00
## 3       10210 216.306532 30.08883  11.616439 29015.87 Manhattan 28688.00
## 4       10221  29.148974 27.16760  17.279031 33987.68 Manhattan 27161.08
## 5       10222   2.422527 12.10421   8.580756 66350.84 Manhattan 13372.73
## 6       10300 506.985063 27.39747  13.606836 32591.55 Manhattan 16035.99
```

We have the numeric code of the health area, the amount of funding per capita, the poverty rate as a percentage of the population, a numeric code for borough (Manhattan, Brooklyn, Bronx, Queens, Staten Island), and the population size. You can download this data in the files section of Canvas.

## Base Plot

For this example, we are going to use a scatterplot to look at the association between funding per capita and poverty rates. Here is the ultimate figure that we want to end up with:

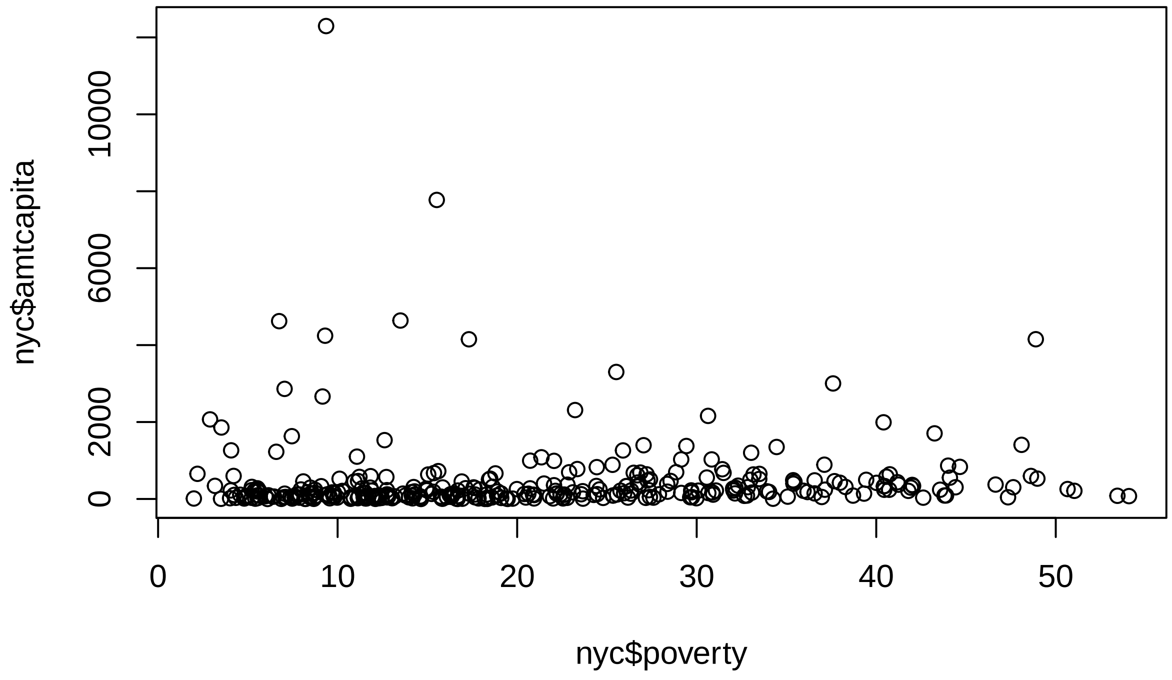


There is a lot going on in this figure. Lets identify a few of the nifty features:

- dots are color-coded by borough
- the scale of the y-axis is *logarithmic* which means that an increase of one unit is actually a multiplicative increase of 10 (sort of like the richter scale).
- The size of dots is scaled to the population size of the health area.
- There is a best-fitting line drawn through the points. It looks positive.
- the y-axis has a dollar-sign and commas to make the numbers easier to read and the x-axis has a % marker to make clear the units.
- there are two different legends drawn on the outside margins of the figure.

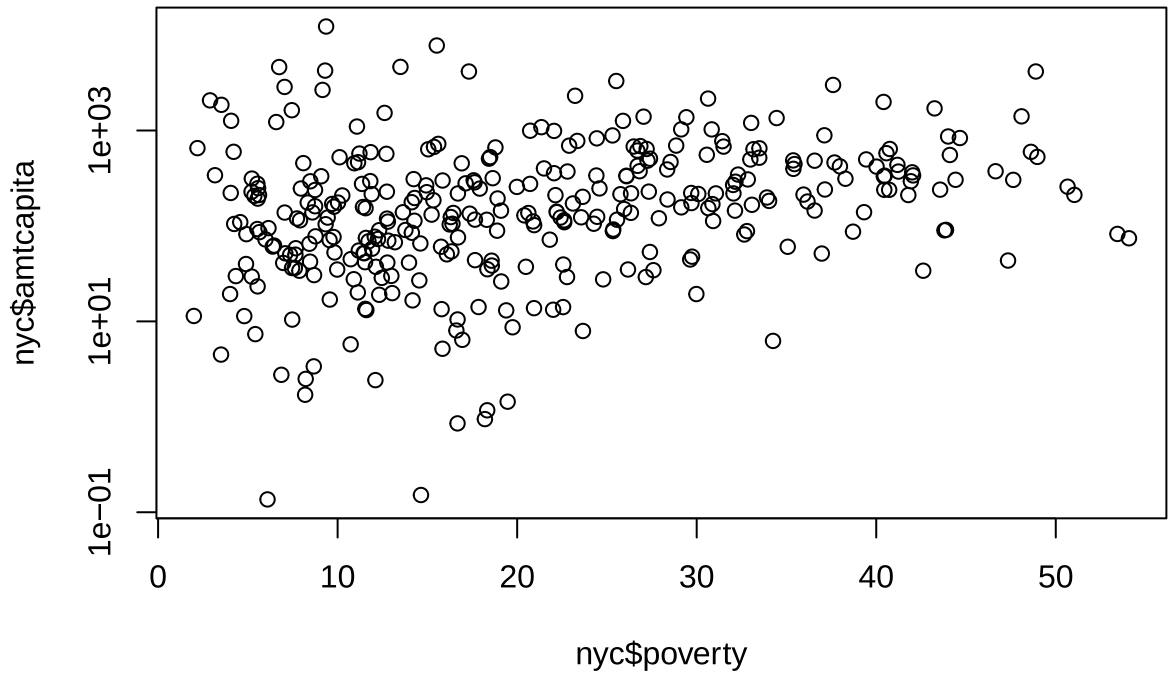
Thats a lot of stuff. Lets start from the basics and build our way up to this final model. Lets start by just running the basic `plot` command for a scatterplot:

```
plot(nyc$poverty, nyc$amtcapita)
```



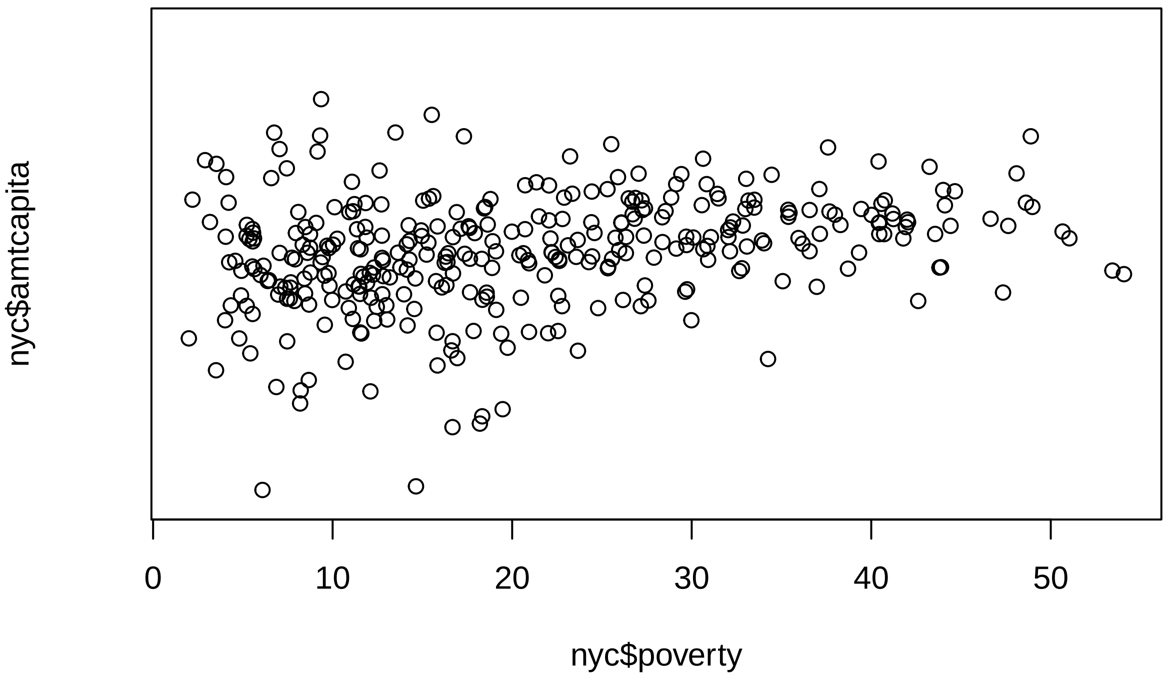
Ok, thats a lot different. Lets deal with the biggest substantive problem. The amount per capita is so heavily right-skewed that it is difficult to see the relationship because almost all of the data points are “squished” down at the bottom of the y-axis. This can be resolved by using whats called a *logarithmic scale* such that each tick mark indicates not an additive unit increase on the y-axis but a multiplicative increase. We can do this easily by specifying “y” as an option to the `log` argument in `plot`:

```
plot(nyc$poverty, nyc$amtcapita,
     log="y")
```



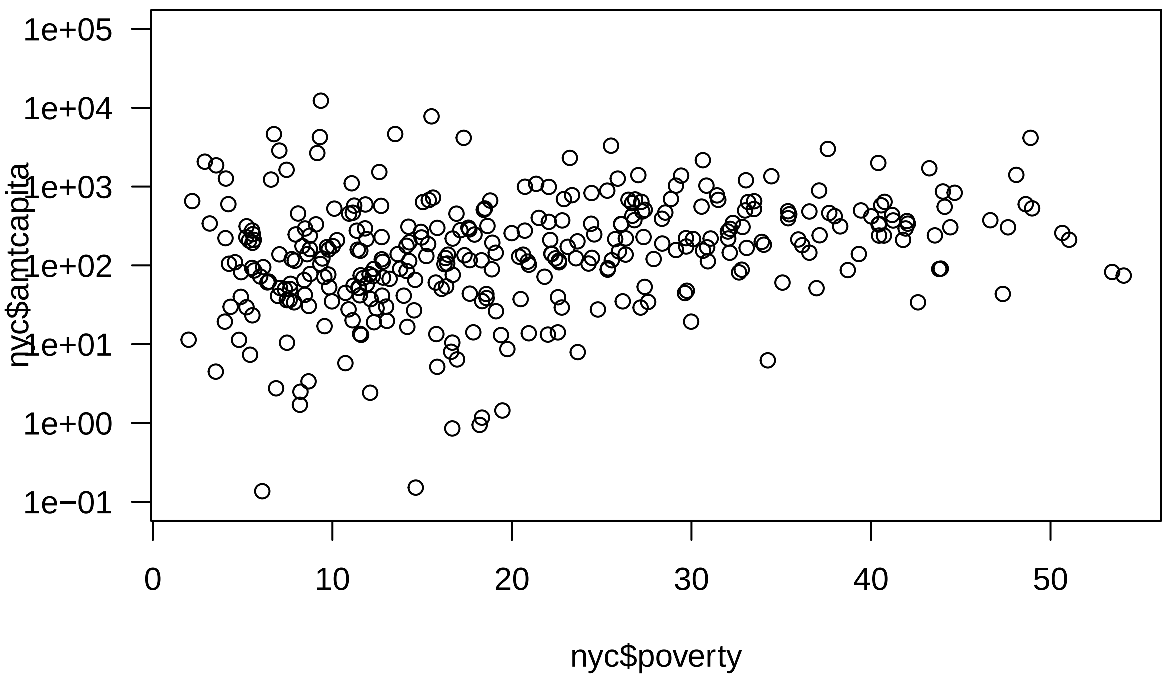
Ok, that looks better. This graphical approach is related to the idea of **tranformations** that we will discuss later in the term. Now we can better see that there is a positive effect. However, the numbers on the y-axis tick-marks look horrible. Instead of using the default tickmarks, I can specify my own axis. A quick check using `summary` reveals that the maximum amount per capita is $12,290. If I am going up by multiples of ten, the next tick mark above this would be 100,000 (from 10,000, to 100,000). So, I am going to do a few things in the next command. First, I am going to use `yaxt="n"` to tell R not to draw the default y-axis. Then I am going to use `ylim=c(0.1,100000)` to specify the upper and lower limits on my y-axis.

```
plot(nyc$poverty, nyc$amtcapita,
     log="y", yaxt="n", ylim=c(0.1,100000))
```



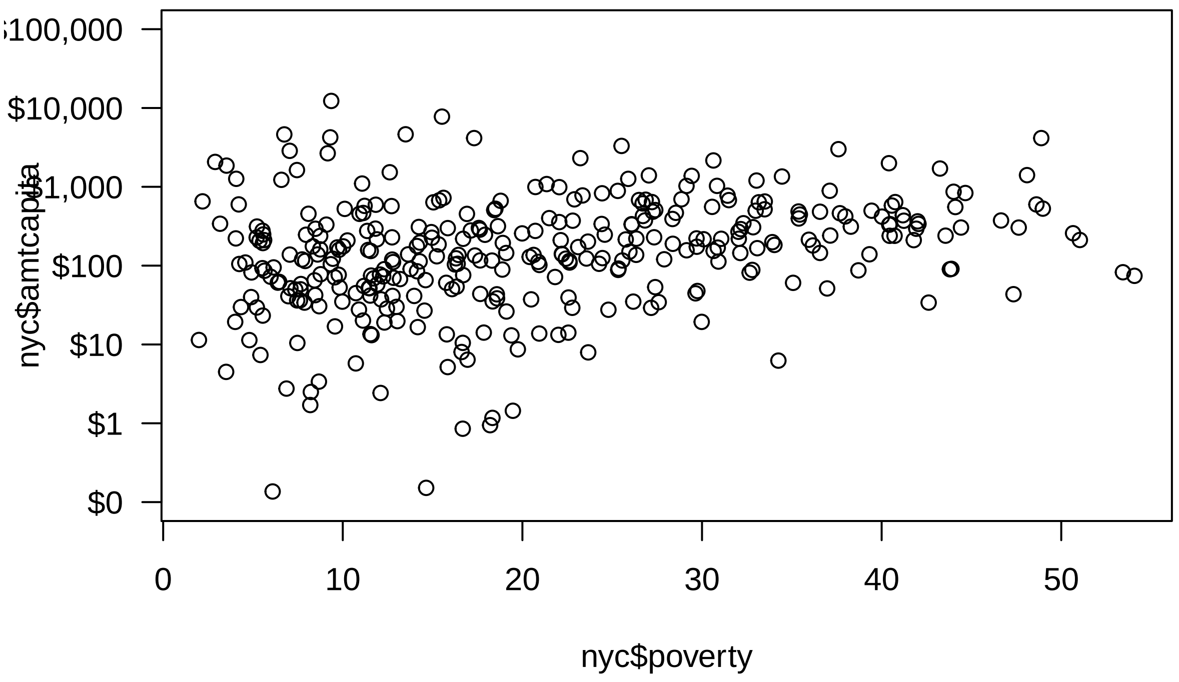
Notice that there is now no y-axis tickmarks. I can define those myself with the `axis` command.

```
plot(nyc$poverty, nyc$amtcapita,
     log="y", yaxt="n", ylim=c(0.1,100000))
axis(2,at=c(0.1,1,10,100,1000,10000,100000), las=1)
```



The `2` argument tells R to draw the axis for the y-axis and the `at` argument allows me to specify where I want the tick marks. The `las` argument specifies that I want my tickmark labels to be perpendicular to the axis, for easier reading. There are still a couple of problems with this axis. First, the label for the y-axis is now overlapping with the tickmark labels. Second, the tickmark labels are in an ugly exponential notation. These can both be fixed. First, I can fix the tickmark labels. There are a variety of functions such as `paste`, `format`, and `formatC` which will allow you to turn numbers into pretty well-formatted character strings. In this case, I am going to use those functions to write out the full number with a comma at the thousands place and put a dollar sign in front:

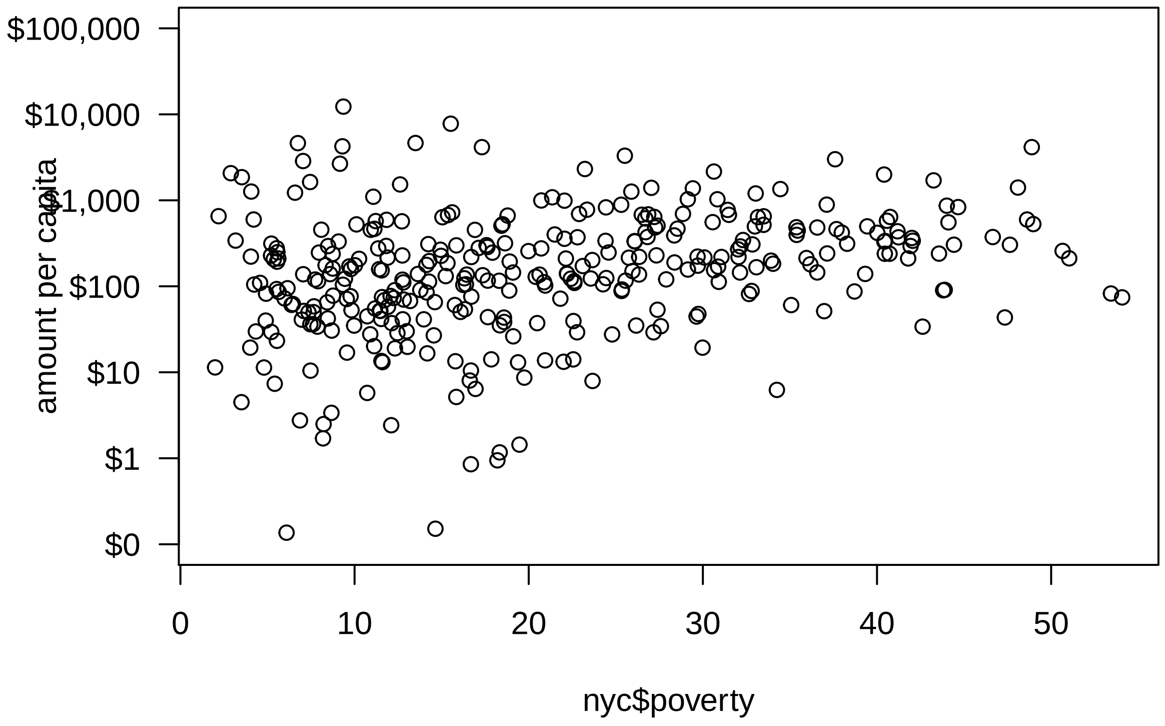
```
plot(nyc$poverty, nyc$amtcapita,
     log="y", yaxt="n", ylim=c(0.1,100000))
axis(2,at=c(0.1,1,10,100,1000,10000,100000), las=2,
     labels=paste("$",formatC(c(0.1,1,10,100,1000,10000,100000),
                                  format="d", big.mark=","), sep=""))
```



I am not going to go over the details of these commands here, but you can see how the tickmark labels are much easier to read now. I still have the problem of the overlapping label for the y-axis and now, the left margin is also not quite large enough for the top label which is being cut-off.

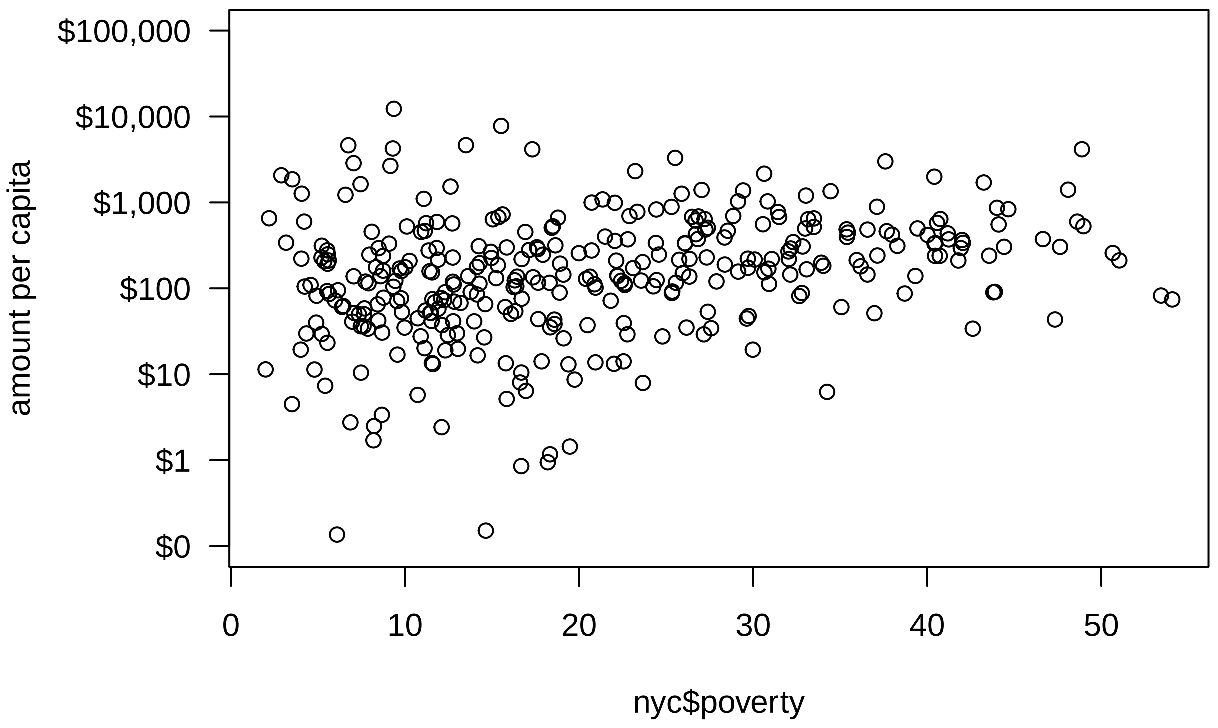
To adjust the margin, let me introduce the `par` command. The `par` command can be called before the plot command to set up the parameters of the plot. If you look at the help file for `par` you will see that there are a huge number of parameters that we can adjust. In our case, we want to adjust the `mar` argument which is a vector of four numbers defining the size of the bottom, left, top, and right margins, respectively. The default for these is `c(5,4,4,2)`. I will increase the left margin slightly and reduce the right margin since we won't use it. In addition, I am going to remove the default y label by specifying `ylab=""`.

```r
par(mar=c(5,6,3,1))
plot(nyc$poverty, nyc$amtcapita,
     log="y", yaxt="n", ylim=c(0.1,100000), ylab="amount per capita")
axis(2,at=c(0.1,1,10,100,1000,10000,100000), las=2,
     labels=paste("$",formatC(c(0.1,1,10,100,1000,10000,100000),
                                  format="d", big.mark=","), sep=""))
```



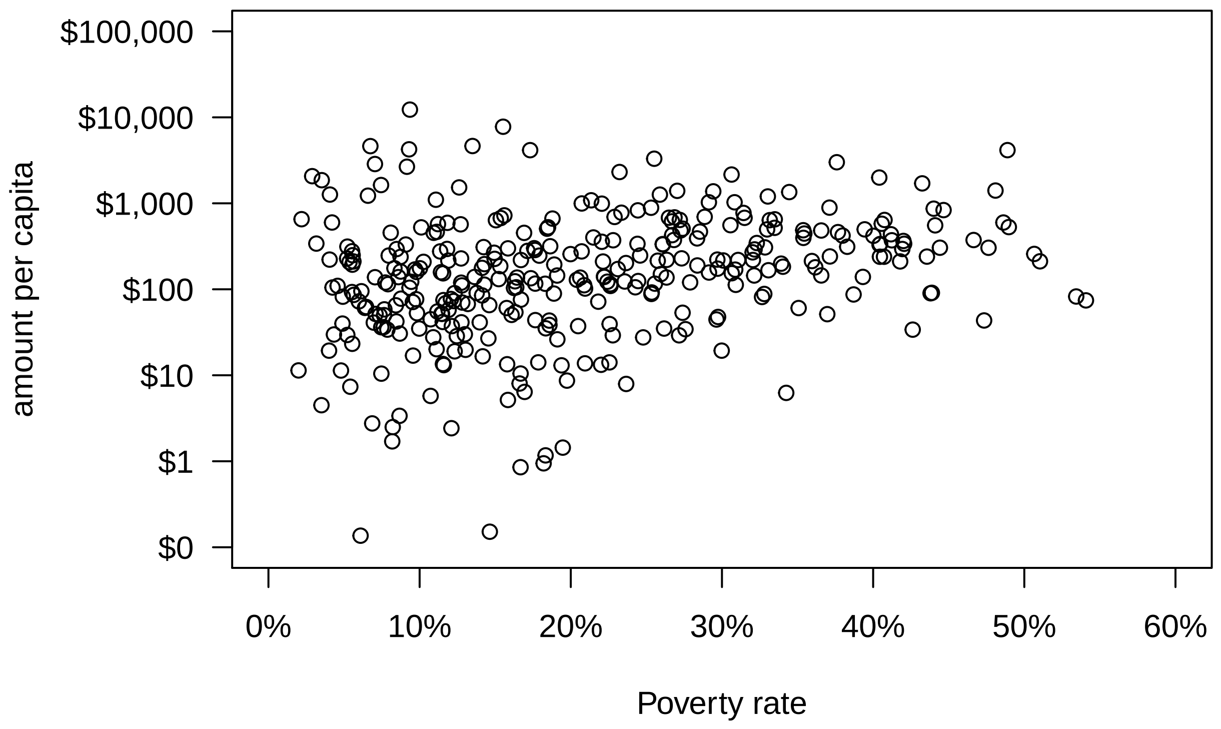
The final step is to define my own y-axis label. I can do this with the `title` command. If you look at the help file for the `title` command, you will see that in addition to telling R which title you want to add, you can specify the number of lines away from the graph for the title, which allows us to control its placement. After experimenting around a bit, I discovered that `line=5` looked nice.

```
par(mar=c(5,6,3,1))
plot(nyc$poverty, nyc$amtcapita,
     log="y", yaxt="n", ylim=c(0.1,100000), ylab="")
axis(2,at=c(0.1,1,10,100,1000,10000,100000), las=2,
     labels=paste("$",formatC(c(0.1,1,10,100,1000,10000,100000),
                             format="d", big.mark=","), sep=""))
title(ylab="amount per capita", line=5)
```



The y-axis now looks pretty good. I was able to override the default y-axis with the `yaxt="n"` and `ylab=""` arguments to `plot` and then use the `axis` and `title` commands to customize tick marks and labels. Now, I will do the same to the x-axis:

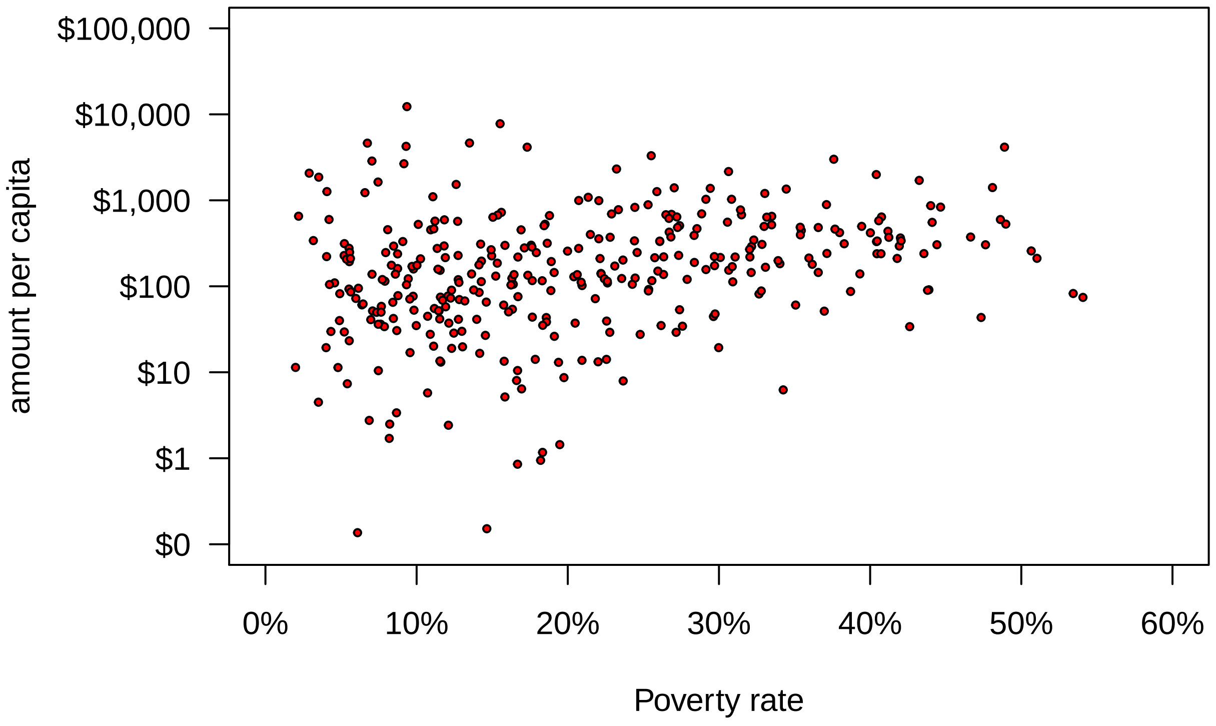
```
par(mar=c(5,6,3,1))
plot(nyc$poverty, nyc$amtcapita,
     log="y", yaxt="n", ylim=c(0.1,100000), ylab="",
     xaxt="n", xlim=c(0,60), xlab="Poverty rate")
axis(2,at=c(0.1,1,10,100,1000,10000,100000), las=2,
     labels=paste("$",formatC(c(0.1,1,10,100,1000,10000,100000),
                              format="d", big.mark=","), sep=""))
title(ylab="amount per capita", line=5)
axis(1,at=seq(from=0,to=60,by=10),labels=paste(seq(from=0,to=60,by=10),"%",sep=""))
```



In this case, I didn't have to define a separate title, because the default x-axis label fits fine. I also don't need the `las` command because the default has the correct alignment.

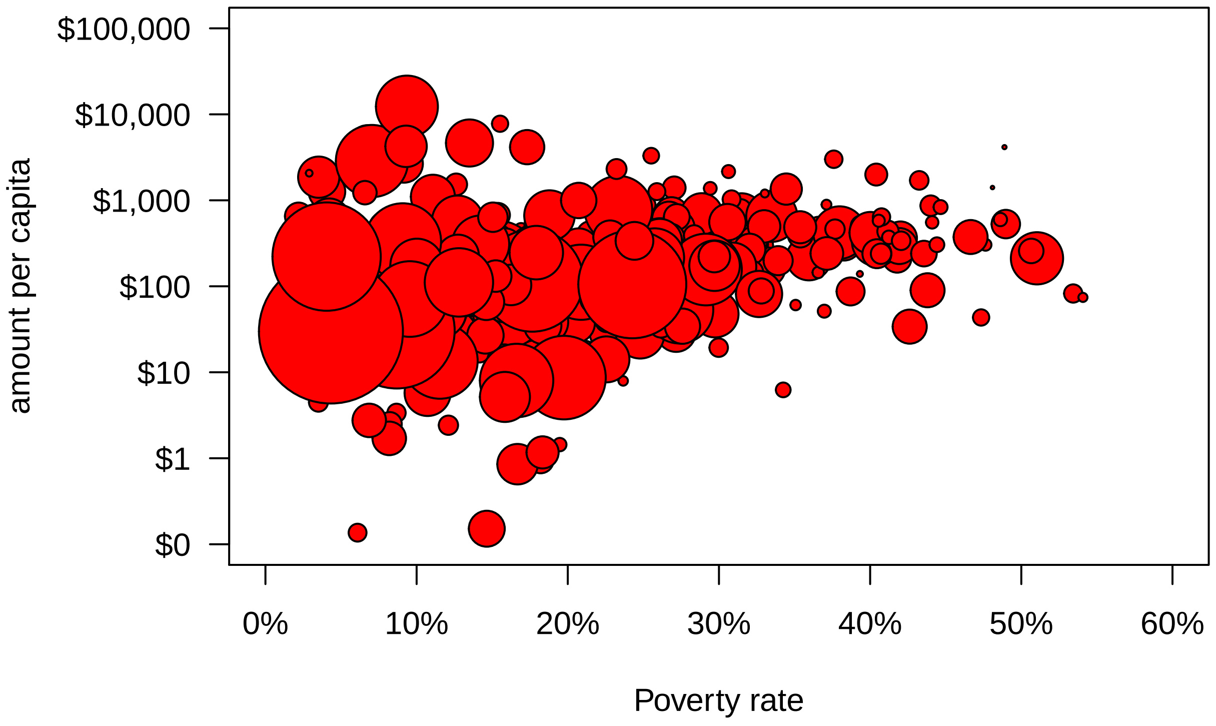
Now that I have my axes well-labeled. I can focus on the actual plot. The default "dots" for scatterplots in R are prety ugly, but there are lots of options for better dots. You can specify the shape and style of the dot with the `pch` argument in plot. If you use `?points`, the help file will give you a list of the numeric codes that correspond to different kinds of dots. I usually use `pch=21` because it will give me a circle that has a separate border and fill color. The border color can be specified by the `col` argument an the fill color can be specified by the `bg` argument. I can also use the `cex` option to define the size of the dots (relative to the default of 1). I will use this now to create circles of half the average size with a red fill and a black border.

```
model <- lm(I(log(nyc$amtcapita))~nyc$poverty)
par(mar=c(5,6,3,1))
plot(nyc$poverty, nyc$amtcapita,
     log="y", yaxt="n", ylim=c(0.1,100000), ylab="",
     xaxt="n", xlim=c(0,60), xlab="Poverty rate",
     pch=21, bg="red", col="black", cex=0.5)
axis(2,at=c(0.1,1,10,100,1000,10000,100000), las=2,
     labels=paste("$",formatC(c(0.1,1,10,100,1000,10000,100000),
                             format="d", big.mark=","), sep=""))
title(ylab="amount per capita", line=5)
axis(1,at=seq(from=0,to=60,by=10),labels=paste(seq(from=0,to=60,by=10),"%",sep=""))
```



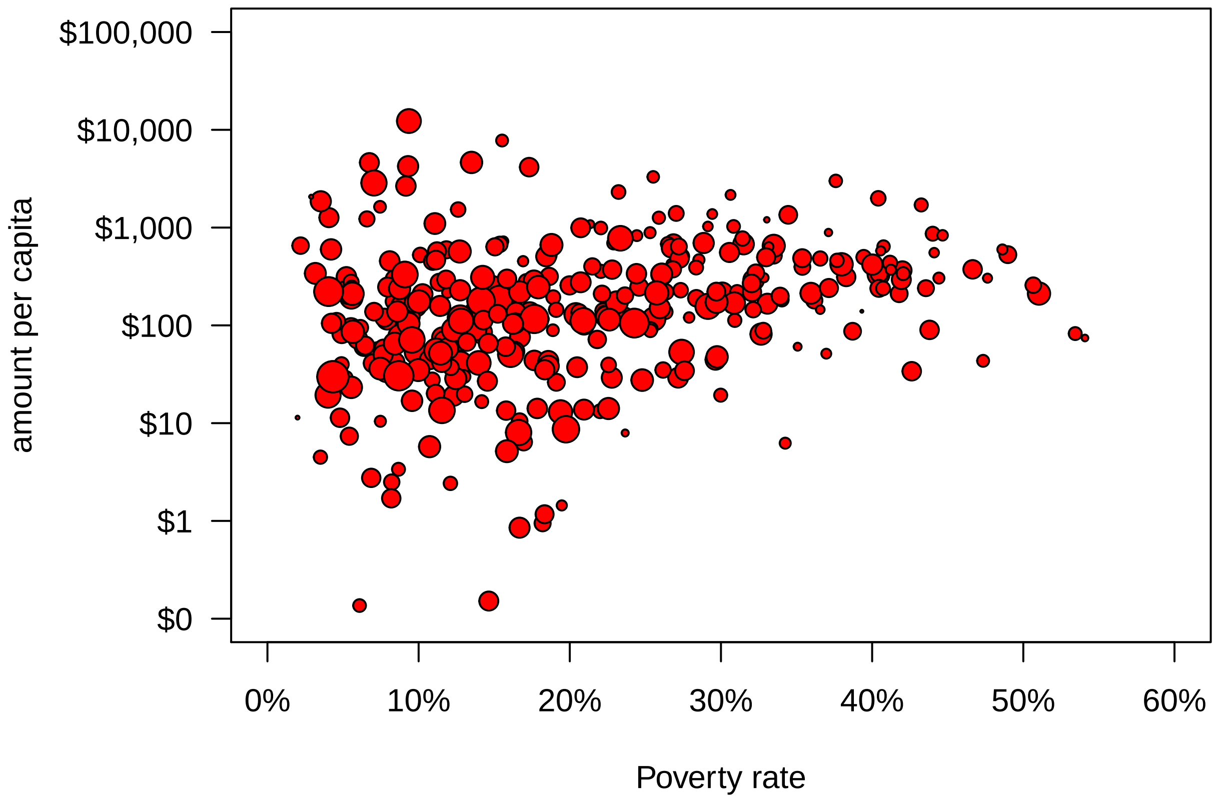
This is now pretty good, but the final touch here is to vary the size of the dots by the size of the health area and the fill color by the borough of the health area. R is very flexible about these types of arguments. If I give a vector of numbers for the size or a vector of colornames for the fill color, R will assume that those colors correspond to the individual dots and will allow for variation in the size and color. For example, lets just feed in population size divided by 10,000 to `cex`.

```
par(mar=c(5,6,3,1))
plot(nyc$poverty, nyc$amtcapita,
     log="y", yaxt="n", ylim=c(0.1,100000), ylab="",
     xaxt="n", xlim=c(0,60), xlab="Poverty rate",
     pch=21, bg="red", col="black", cex=nyc$popn/10000)
axis(2,at=c(0.1,1,10,100,1000,10000,100000), las=2,
     labels=paste("$",formatC(c(0.1,1,10,100,1000,10000,100000),
                             format="d", big.mark=","), sep=""))
title(ylab="amount per capita", line=5)
axis(1,at=seq(from=0,to=60,by=10),labels=paste(seq(from=0,to=60,by=10),"%",sep=""))
```



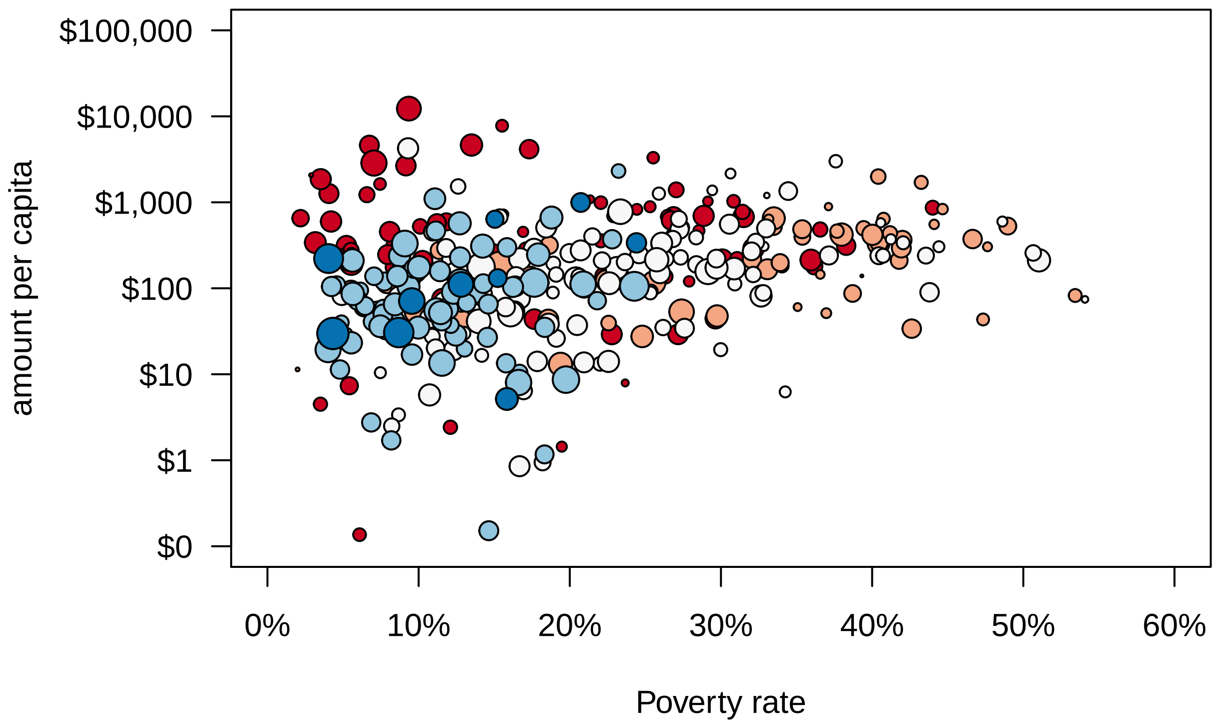
Well, that sort of worked but some of those dots are way too big and some are way too small. The variation in size between health areas is so large that any kind of linear scaling of the population size is going to result in this problem. There are a variety of potential solutions to this, but our earlier use of logarithmic scales suggests an easy one. Logging the population values will allow for differences in size but at a diminishing scale difference. After experimentation, I decided that dividing population size by 3000 and logging with a base of 5 produced good size variation.

```r
par(mar=c(5,6,1,1))
plot(nyc$poverty, nyc$amtcapita,
     log="y", yaxt="n", ylim=c(0.1,100000), ylab="",
     xaxt="n", xlim=c(0,60), xlab="Poverty rate",
     pch=21, bg="red", col="black", cex=log(nyc$popn/3000,5))
axis(2,at=c(0.1,1,10,100,1000,10000,100000), las=2,
     labels=paste("$",formatC(c(0.1,1,10,100,1000,10000,100000),
                              format="d", big.mark=","), sep=""))
title(ylab="amount per capita", line=5)
axis(1,at=seq(from=0,to=60,by=10),labels=paste(seq(from=0,to=60,by=10),"%",sep=""))
```



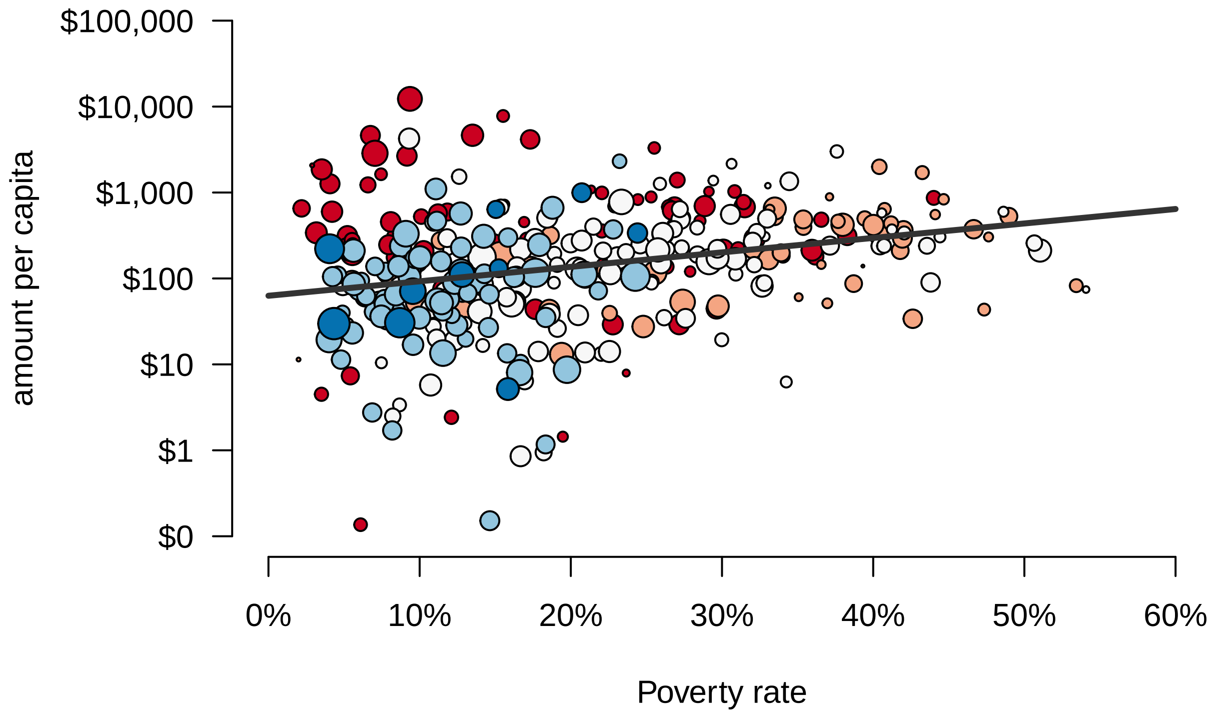
The final step is to color-code the dots. I could put in a vector of any five colors. However, it is important to think about accessibility for color-blind individuals and whether a color combination will show up well in print. There are many online resources for this sort of thing. I like ColorBrewer. Here I have specified five classes with a diverging scheme that are colorblind and print friendly. I wiould have selected qualitative scale but there are no color-blind options in that category for five classes. ColorBrewer gives me some options with hexadecimal color codes which I can feed into R. I first create a vector of the five color names and then I use the borough index to assign them in my plot command.

```
color_choices <- c("#ca0020","#f4a582","#f7f7f7","#92c5de","#0571b0")
par(mar=c(5,6,3,1))
plot(nyc$poverty, nyc$amtcapita,
     log="y", yaxt="n", ylim=c(0.1,100000), ylab="",
     xaxt="n", xlim=c(0,60), xlab="Poverty rate",
     pch=21, bg=color_choices[nyc$borough], col="black", cex=log(nyc$popn/3000,5))
axis(2,at=c(0.1,1,10,100,1000,10000,100000), las=2,
     labels=paste("$",formatC(c(0.1,1,10,100,1000,10000,100000),
                              format="d", big.mark=","), sep=""))
title(ylab="amount per capita", line=5)
axis(1,at=seq(from=0,to=60,by=10),labels=paste(seq(from=0,to=60,by=10),"%",sep=""))
```



We are almost there but I also want to add a best-fitting line. Normally, I could do this with the `abline` command as discussed in the Canvas section on the OLS regression line. However, that won't work in this case because of the the logarithmic scale on the y-axis. Instead, I can create a sequence of poverty rate values and then based on a model, I can calculate the predicted amount per capita (note that you don't know how to do this yet, so just hang tight). I can then feed those x and y values into a `lines` command to draw a line on my plot.

```r
model <- lm(I(log(nyc$amtcapita))~nyc$poverty)
x <- 0:60
y <- exp(model$coef[1])*exp(model$coef[2])^x
par(mar=c(5,6,3,1))
plot(nyc$poverty, nyc$amtcapita,
     log="y", yaxt="n", ylim=c(0.1,100000), ylab="",
     xaxt="n", xlim=c(0,60), xlab="Poverty rate",
     pch=21, bg=color_choices[nyc$borough], col="black", cex=log(nyc$popn/3000,5),
     bty="n")
axis(2,at=c(0.1,1,10,100,1000,10000,100000), las=2,
     labels=paste("$",formatC(c(0.1,1,10,100,1000,10000,100000),
                              format="d", big.mark=","), sep=""))
title(ylab="amount per capita", line=5)
axis(1,at=seq(from=0,to=60,by=10),labels=paste(seq(from=0,to=60,by=10),"%",sep=""))
lines(x,y, lwd=3, col="grey20")
```

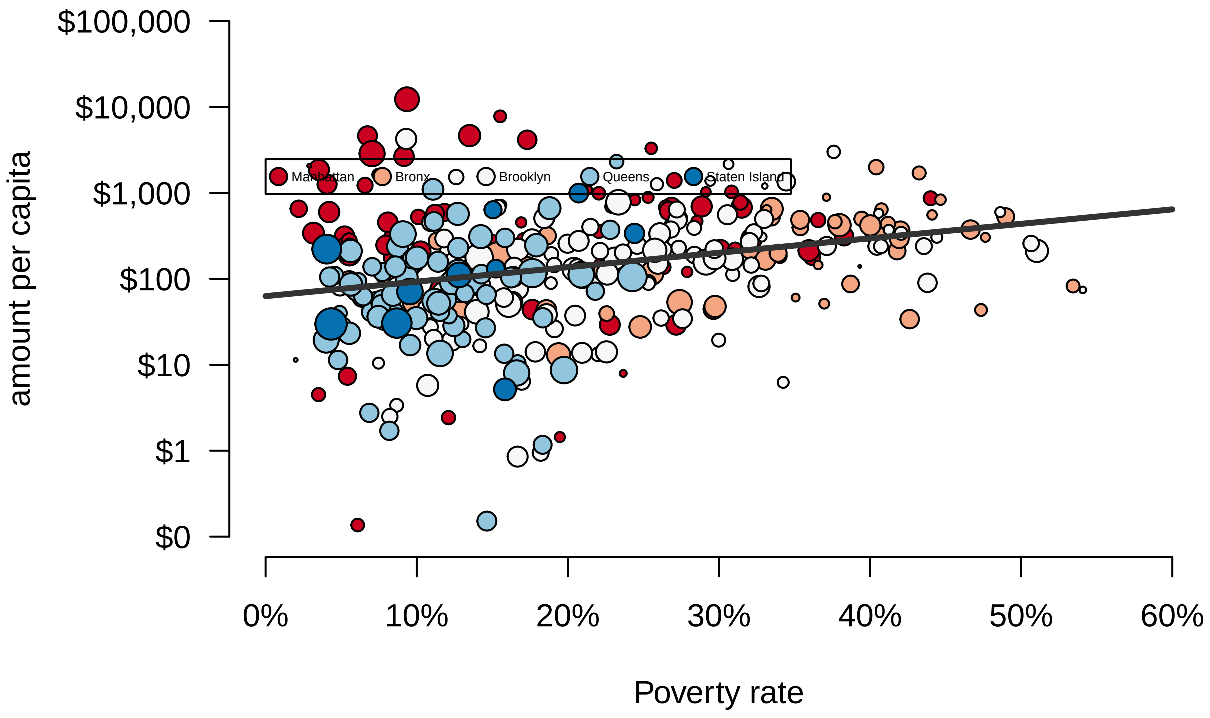


I don't want you to worry to much about the model part. The important feature I want to highlight here is that `lines` is one of a number of commands that include `points`, `text`, and `mtext` that you can use to later add other stuff to a plot that you have already made. In this case, I have added a straight line. The `lwd` argument defines the width of the line and the `col` argument defines the color of the line.

I snuck in one other argument here that made a noticeable change. The argument `bty` defines how the border is drawn around the overall plot area. By setting this to "n", I removed the border altogether, which I think gives it a cleaner look.

The last step is to add some legends. Legends can be tricky. The first thing you have to figure out is where to place the legend. In my case, I would rather have the legend in the margin than in the main plot area, but R won't do this by default. In order to do that, I need to specify an `xpd=TRUE` argument in the `par` command to allow writing output to the margins and not just the main plot area. In the `legend` command itself, I need to make a label for each component of the legend and then I need to specify how each component is identified. Lets start with the legend for boroughs.

```
model <- lm(I(log(nyc$amtcapita))~nyc$poverty)
x <- 0:60
y <- exp(model$coef[1])*exp(model$coef[2])^x
par(mar=c(5,6,3,1), xpd=TRUE)
plot(nyc$poverty, nyc$amtcapita,
     log="y", yaxt="n", ylim=c(0.1,100000), ylab="",
     xaxt="n", xlim=c(0,60), xlab="Poverty rate",
     pch=21, bg=color_choices[nyc$borough], col="black", cex=log(nyc$popn/3000,5),
     bty="n")
axis(2,at=c(0.1,1,10,100,1000,10000,100000), las=2,
     labels=paste("$",formatC(c(0.1,1,10,100,1000,10000,100000),
                                         format="d", big.mark=","), sep=""))
title(ylab="amount per capita", line=5)
axis(1,at=seq(from=0,to=60,by=10),labels=paste(seq(from=0,to=60,by=10),"%",sep=""))
lines(x,y, lwd=3, col="grey20")
lg <- legend(0, 100000,
             legend=c("Manhattan","Bronx","Brooklyn","Queens","Staten Island"),
             pch=21, col="black", pt.bg=color_choices, ncol=5,
             cex=0.45, pt.cex=1.2, yjust=5)
```



The first two arguments to `legend` give the x and y placement. The `legend` argument to `legend` (I know, its weird) gives the labels for the legend components. The `pch` argument tells the legend that I am using points and what their shape is. The `col` argument gives the border color for these points and the `pt.bg` argument gives their fill color. The `ncol` tells the legend to use five separate columns rather than a vertical alignment all in one column. The `cex` and `pt.cex` arguments indicate the size of the overall legend and the size of the dots, respectively. The `yjust` argument allows me to fudge the placement to get it just right.

Notice, that I saved the output of legend to an object that I called `lg`. This is a very useful feature of plots. I want to draw my next legend for health area size next to this first legend, but I have no idea exactly how big the first legend will be, so its hard to know at what value of x to start it. I could guess a number here, but that might also change if I rescale the figure manually. However, if I look at the `lg` object, the information I am looking for is returned there:

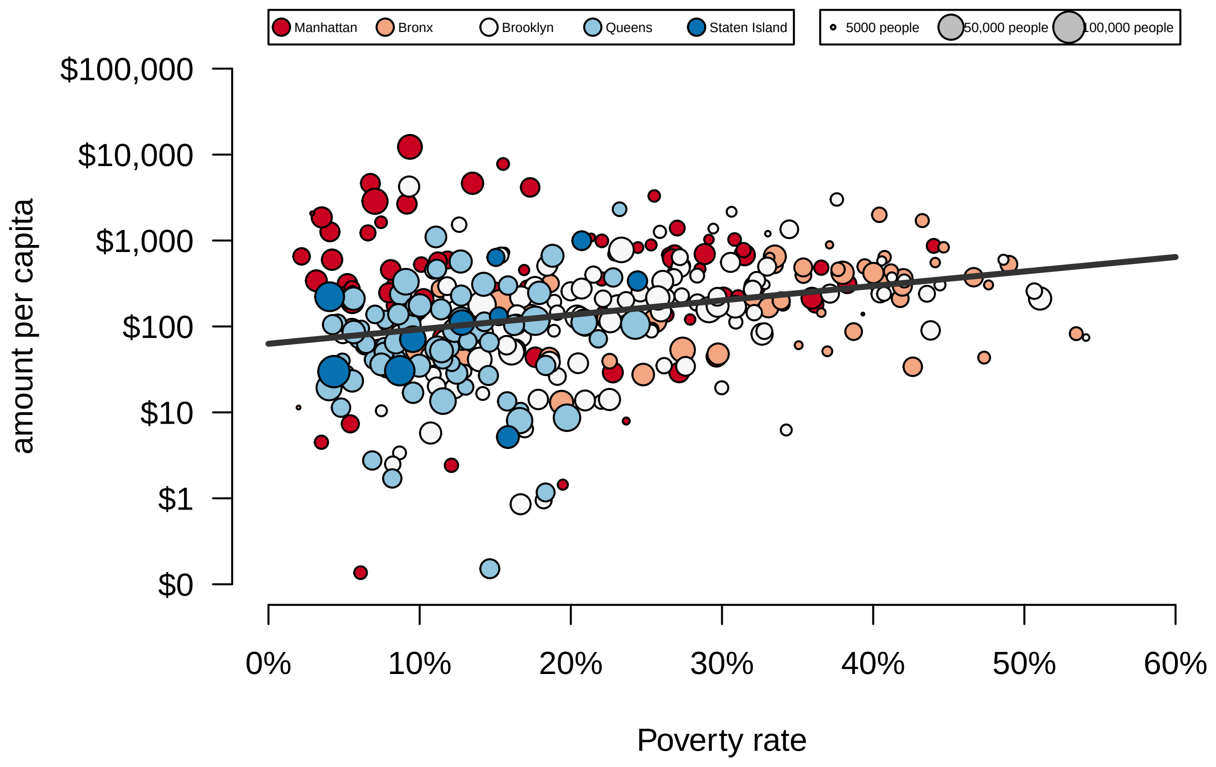
```
lg
```

```
## $rect
## $rect$w
## [1] 34.75676
##
## $rect$h
## [1] 0.4022069
##
## $rect$left
## [1] 0
##
## $rect$top
## [1] 3.391172
##
```

```
##
## $text
## $text$x
## [1]  1.715294  8.580882 15.446471 22.312059 29.177647
##
## $text$y
## [1] 3.190069 3.190069 3.190069 3.190069 3.190069
```

In this specific case, I am looking for `lg$rect$w` which gives the width of the first legend. I can use that to set up the placement of my second legend:

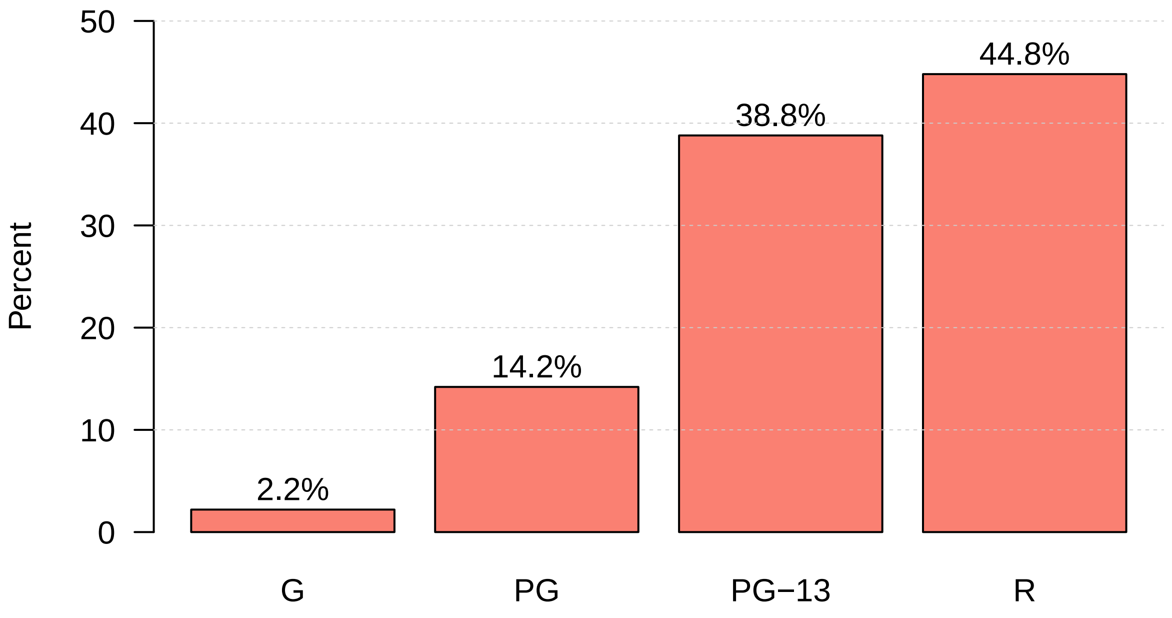
```
model <- lm(I(log(nyc$amtcapita))~nyc$poverty)
x <- 0:60
y <- exp(model$coef[1])*exp(model$coef[2])^x
par(mar=c(5,6,3,1), xpd=TRUE)
plot(nyc$poverty, nyc$amtcapita,
     log="y", yaxt="n", ylim=c(0.1,100000), ylab="",
     xaxt="n", xlim=c(0,60), xlab="Poverty rate",
     pch=21, bg=color_choices[nyc$borough], col="black", cex=log(nyc$popn/3000,5),
     bty="n")
axis(2,at=c(0.1,1,10,100,1000,10000,100000), las=2,
     labels=paste("$",formatC(c(0.1,1,10,100,1000,10000,100000),
                              format="d", big.mark=","), sep=""))
title(ylab="amount per capita", line=5)
axis(1,at=seq(from=0,to=60,by=10),labels=paste(seq(from=0,to=60,by=10),"%",sep=""))
lines(x,y, lwd=3, col="grey20")
lg <- legend(0, 100000,
             legend=c("Manhattan","Bronx","Brooklyn","Queens","Staten Island"),
             pch=21, col="black", pt.bg=color_choices, ncol=5,
             cex=0.45, pt.cex=1.2, yjust=-0.7)
legend(lg$rect$w*1.05, 100000,
       legend=c("5000 people","50,000 people","100,000 people"),
       pch=21, col="black",pt.bg="grey",ncol=3,
       pt.cex=log(c(5000,50000,100000)/3000, 5),
       yjust=-0.7, cex=0.45)
```



And there is the final product. Keep in mind that some of the final touches here are fairly complex. I am not expecting you to be able to produce graphs of this complexity tomorrow. The goal was to show you the richness and depth of graphing in R and to give you some reference points for beginning to build your own beautiful graphs.

The `plot` function is one of the most basic functions for creating plots and once you get the basics down you can create a wide variety of two-dimensional plots. However, there are a variety of other functions that will draw more specific plots. We have already seen examples of `pie`, `barplot`, `hist`, and `boxplot`. Most of the options for customization that are available for `plot` are also available for these other functions. For example, in the code here I use the `text` function to plot the actual percentage values at the top of my bars for a barplot of movie maturity rating and to create lines for the y-axis at 10% intervals.

```
percent <- round(100*table(movies$Rating)/sum(table(movies$Rating)),1)
b <- barplot(percent, las=1, ylab="Percent", ylim=c(0,50), col="salmon")
text(b[,1], percent+2, paste(percent, "%", sep=""))
abline(h=seq(from=10,to=50,by=10), lwd=0.5, col="grey80", lty=2)
```



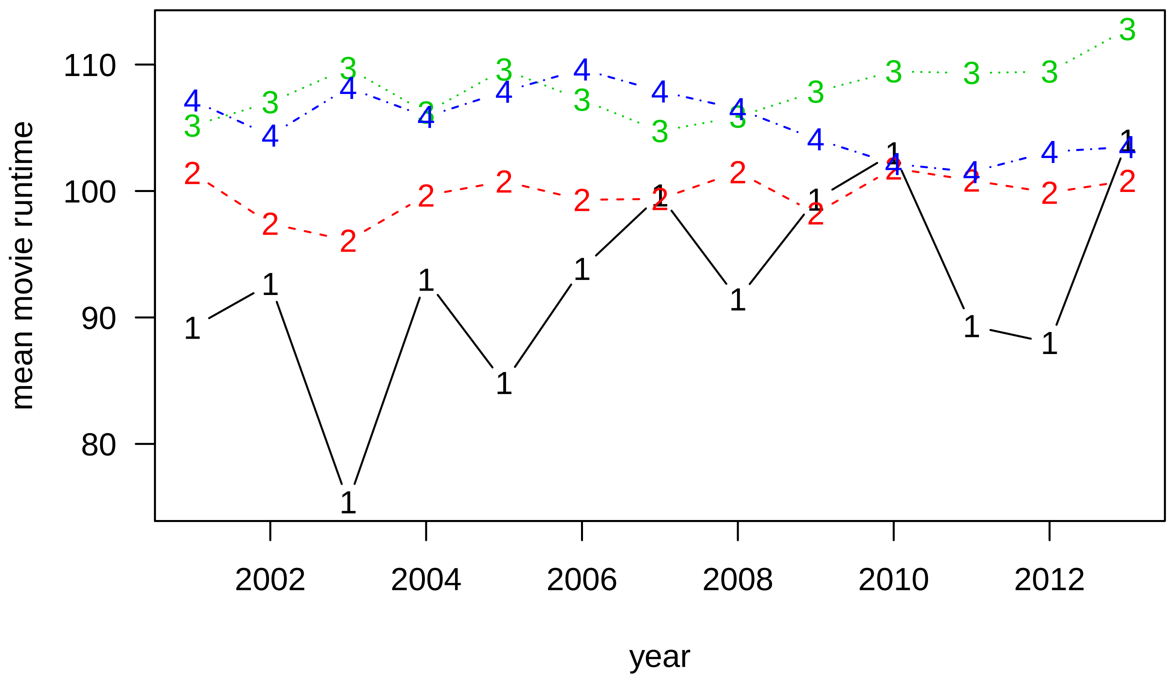
Another useful command that we haven't learned yet is `matplot` which is short for matrix plotting. The `matplot` function will plot the values of one dimension of the matrix across the indices of the other dimension. This allows you to plot, for example, trend lines separately by different categories. Lets try it out by plotting the time trend in movie runtime separately by maturity rating. The first step in doing this is to calculate the mean of movie runtime by year and maturity rating using the `tapply` command.

```
tab <- tapply(movies$Runtime, movies[,c("Year","Rating")], mean)
tab
```

```
##       Rating
## Year           G        PG    PG-13        R
##   2001  89.20000 101.43750 105.1667 107.1744
##   2002  92.66667  97.42308 107.0139 104.3974
##   2003  75.40000  96.08333 109.7143 108.1692
##   2004  93.00000  99.65714 106.2027 105.8769
##   2005  84.83333 100.76471 109.6024 107.8659
##   2006  93.85714  99.30233 107.2118 109.6282
##   2007  99.66667  99.37931 104.7324 107.8900
##   2008  91.44444 101.50000 105.8750 106.4935
##   2009  99.33333  98.24242 107.8481 104.1071
##   2010 103.00000 101.79412 109.4559 102.1495
##   2011  89.33333 100.84000 109.3297 101.5098
##   2012  88.00000  99.87500 109.4400 103.1043
##   2013 104.00000 100.81250 112.7971 103.4904
```

Now, I can feed this matrix into `matplot` to see the trend across time. In this case, I am going to leave off NC-17 and Unrated because the small number of movies here makes these measures very noisy.

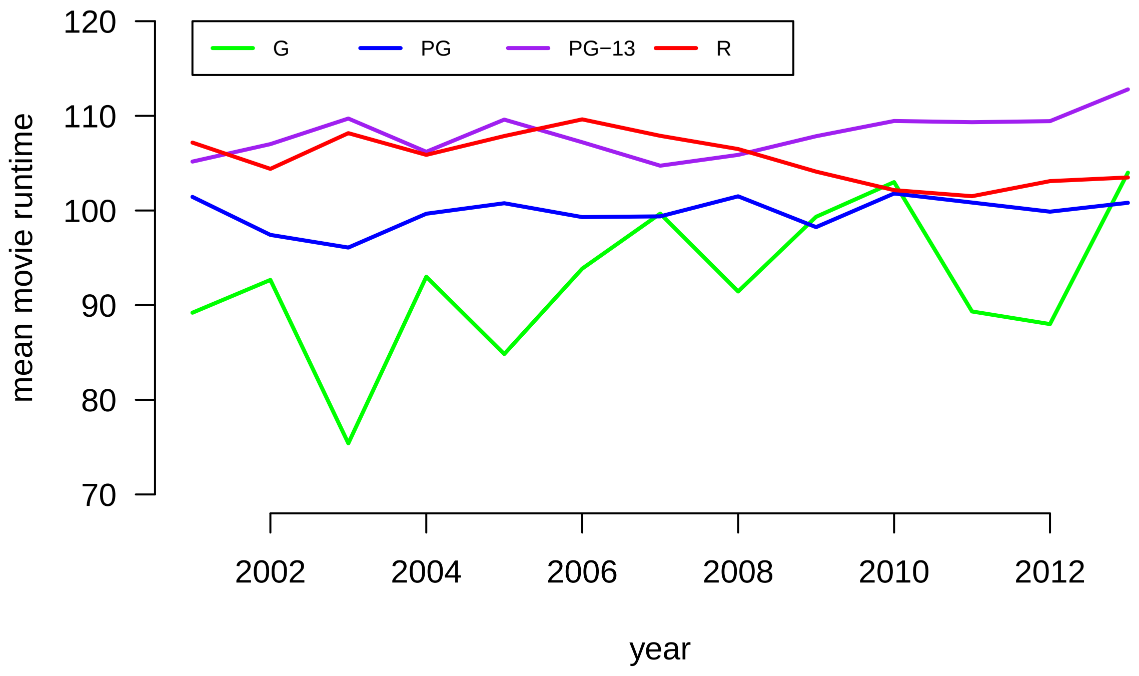
```
matplot(2001:2013, tab[,1:4], type="b", xlab="year", ylab="mean movie runtime", las=1)
```



One interesting trend is that PG-13 movies have become longer than R movies, mostly because the runtime of R movies has gotten progressively smaller since 2006. G movies are also getting slight longer over time, but its highly variable from year to year.

I could have also created this plot with the basic `plot` command and some `lines` commands, like so:

```r
plot(-1,-1, xlab="year", ylab="mean movie runtime", las=1,
     xlim=c(2001,2013), ylim=c(70,120), bty="n")
lines(2001:2013, tab[,1], lwd=2, col="green")
lines(2001:2013, tab[,2], lwd=2, col="blue")
lines(2001:2013, tab[,3], lwd=2, col="purple")
lines(2001:2013, tab[,4], lwd=2, col="red")
legend(2001, 120, legend=c("G","PG","PG-13","R"), lty=1, lwd=2,
       col=c("green","blue","purple","red"), ncol=4, cex=0.7)
```



The plot command here basically creates an empty canvas because I give a single point coordinate (-1,-1) that is outside the range of my xlim and ylim values. I can then use the `lines` command to write specific lines onto this blank canvas.

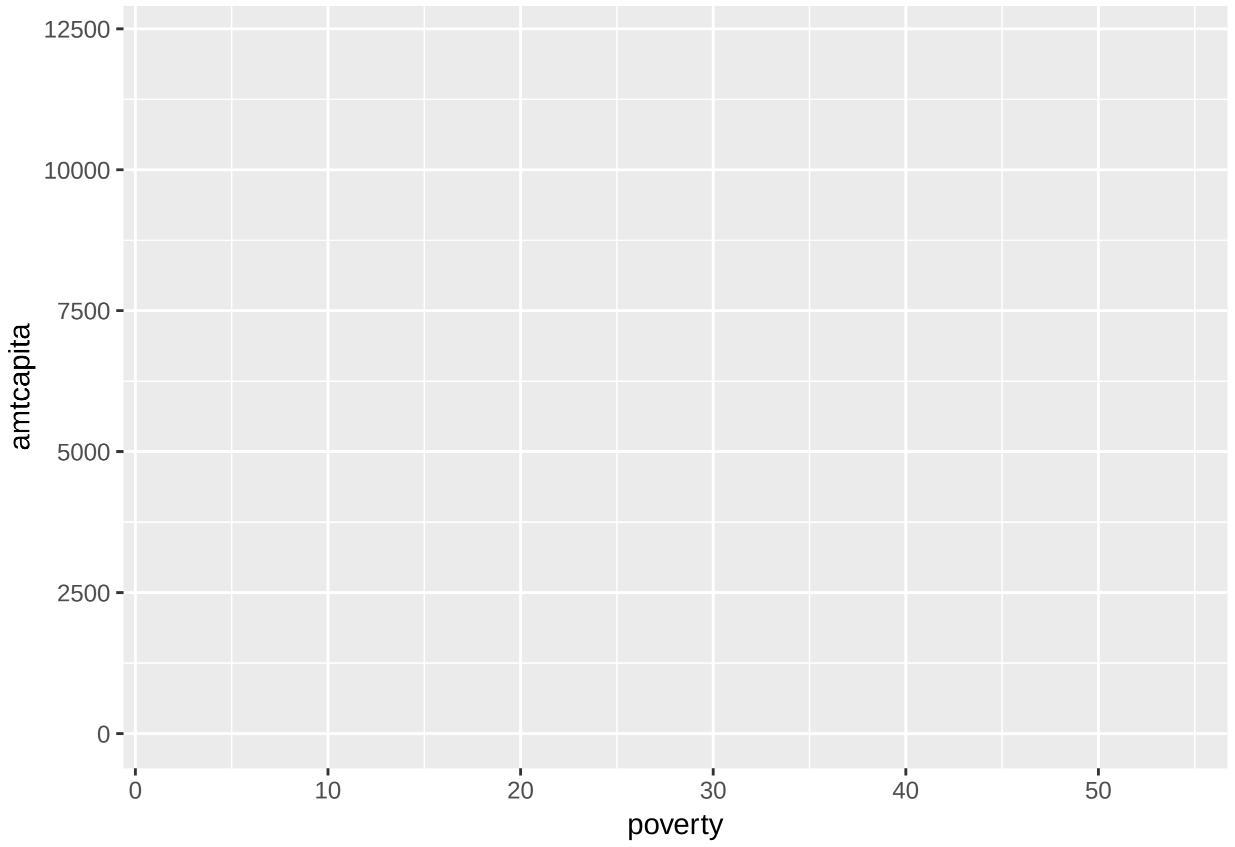
## ggplot

Now, I want to show you how to plot that same scatterplot of non-profit funding in NYC using ggplot. First you will need to install ggplot and load the library:

```
install.packages("ggplot2", repos =  "http://cran.us.r-project.org")
install.packages("scales", repos =  "http://cran.us.r-project.org")
library(ggplot2)
library(scales)
```

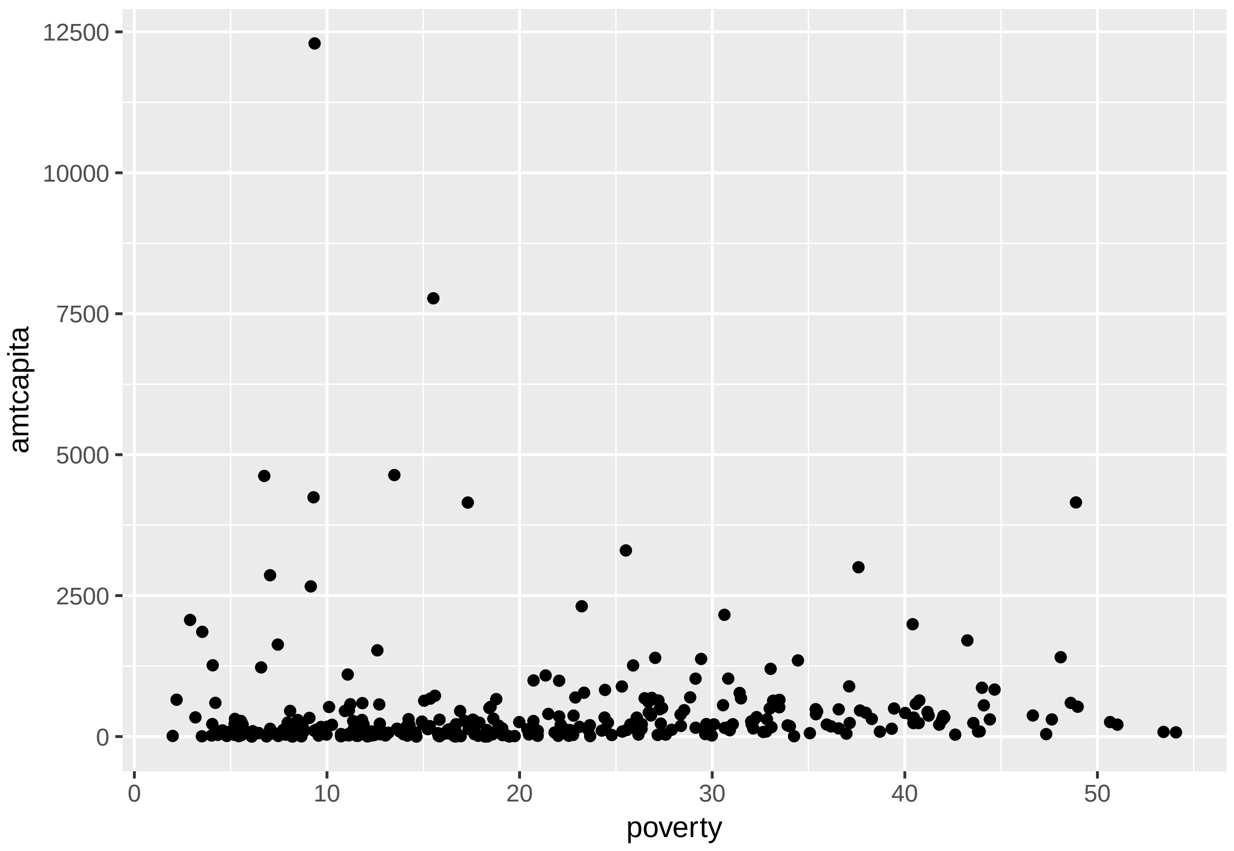
Ggplot builds up a graph from layers. The first and most essential component is the function `ggplot` where I indicate the data I am using and the aesthetics that I want to be carried through to all of the other layers:

```
ggplot(nyc, aes(x=poverty, y=amtcapita))
```



This command does not actually plot anything yet. It just sets up the basic structure of my plot by identifying the dataset and that I will use `poverty` as my `x` variable and `amtcapita` as my `y` variable. I then can add layers to this basic command using the “+” sign. For example, if I wanted to create a scatterplot by plotting points:

```r
ggplot(nyc, aes(x=poverty, y=amtcapita))+
  geom_point()
```

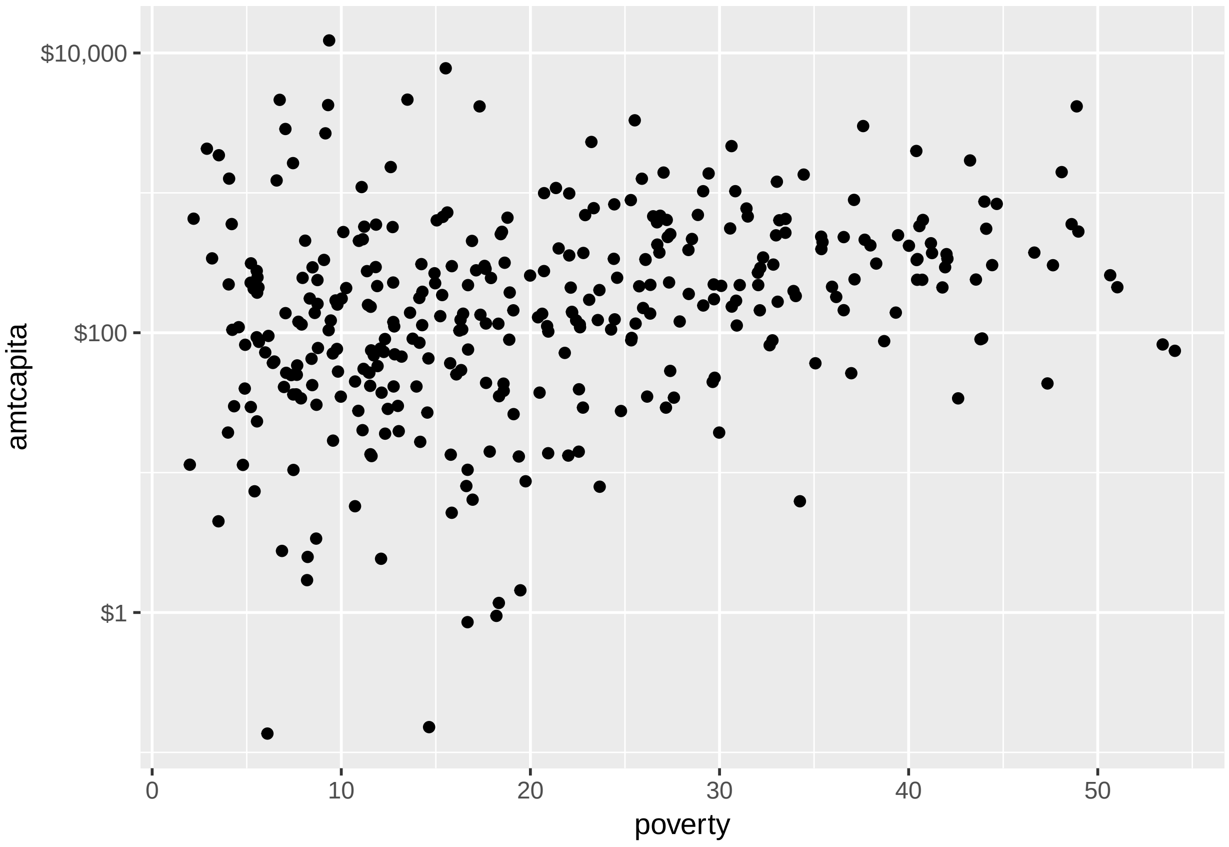


Now, I have a basic (and very ugly) scatterplot that is similar to what I started with in base plot. I can now add a variety of layers to that start to make a better graph. There are three types of layers I can add:

1. `geoms` - these are a variety of geometric patterns such as points, bars, lines, etc.
2. The `coords` that define the coordinate system used to plot the values. We typically won't futz around with this much because everything is drawn on a Cartesian coordinate system, but it can be useful for maps and some other things.
3. `scales` - these indicate how I want the scales of my various aesthetics to work. This can include the scaling of my x and y variables, but also things like color gradiations.
4. `labels` - I can identify labels and themes to use.

For example, let me use the `scale_y_log10()` function to re-scale my y-axis to a logarithmic basis:

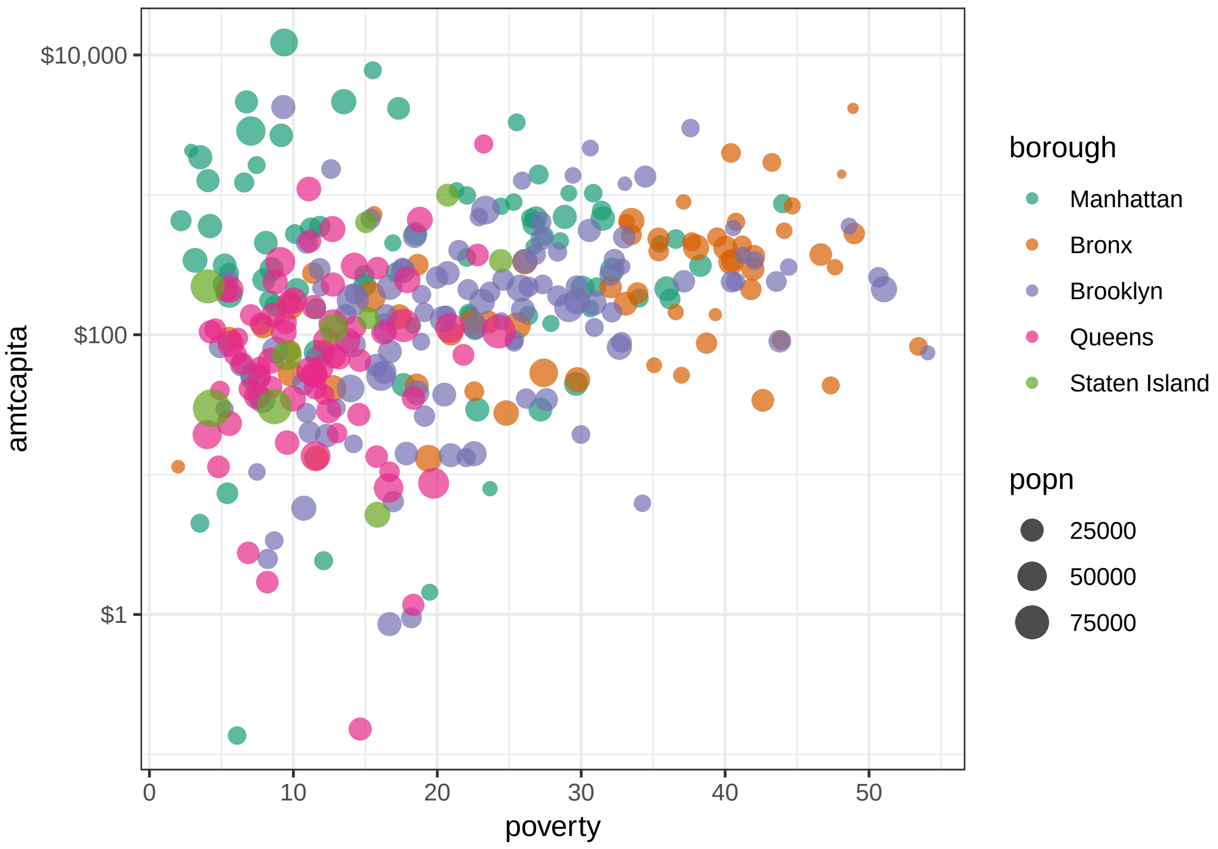
```r
ggplot(nyc, aes(x=poverty, y=amtcapita))+
  geom_point()+
  scale_y_log10(labels=dollar)
```



That looks better. Notice that I also gave the `scale_y_log10` command an argument of `labels`. This argument identifies the specific labels I want to use for the tick mark. In this case, I am supplying a function from the `scales` library that turns raw numbers into formatted dollar amounts.

From here I can add a variety of layers and aesthetics to enrich my graph. Let me first add aesthetics for color and size. I will also tranform borough into a proper factor variable so it displays more nicelyin the legend.

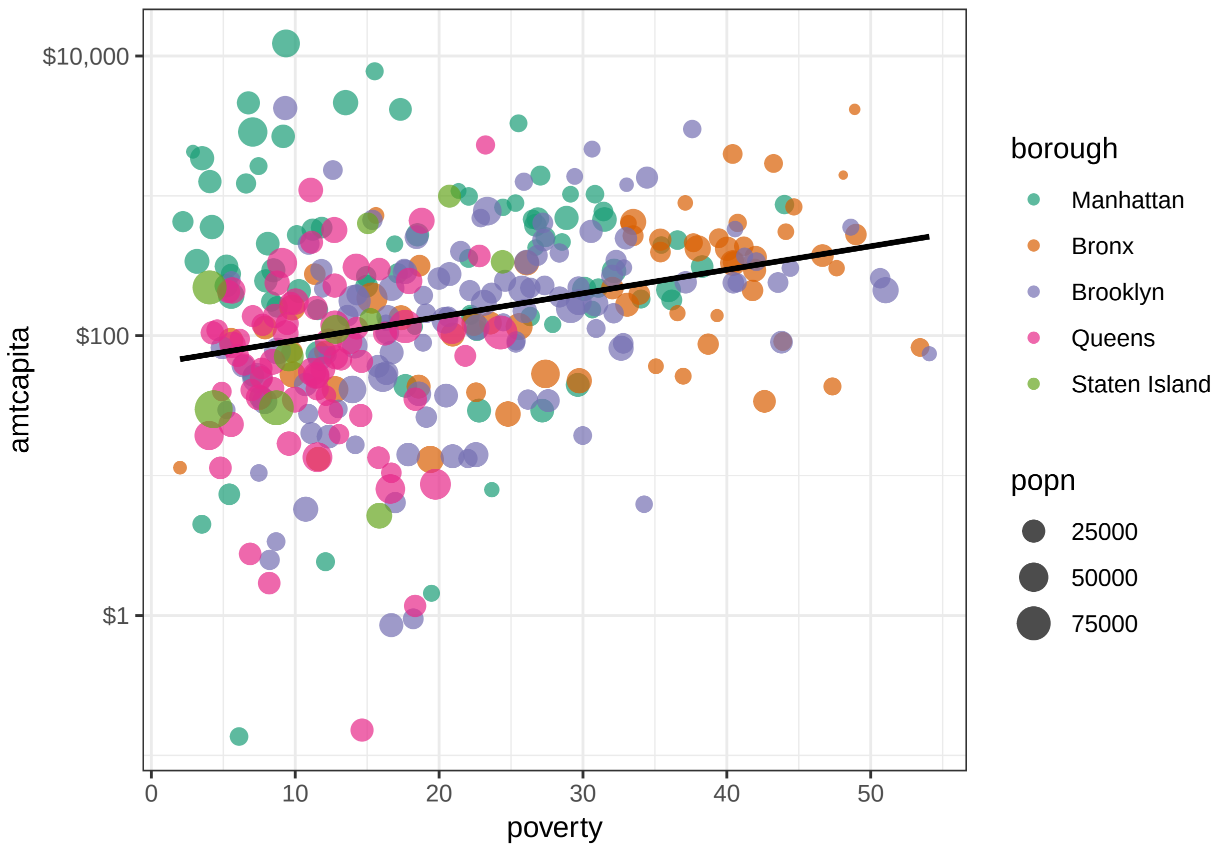
```r
ggplot(nyc, aes(x=poverty, y=amtcapita, size=popn, color=borough))+
  geom_point(alpha=0.7)+
  scale_y_log10(labels=dollar)+
  scale_color_brewer(palette="Dark2")+
  theme_bw()
```



By just adding in the borough and pop size as aesthetics, the graph was quickly adjusted. I didn't have to fiddle around with exact sizing of the dots. Ggplot handled those details. Note that I also added a scale for the color using one of ggplot's pre-defined palettes. I also used the argument of `alpha=0.7` to add some transparency to points, which helps me deal with issues of overplotting. Finally, I used `theme_bw()` at the bottom to change to a black and white theme.

I also want to add a line for the best-fitting OLS regressin line. The `geom_smooth` function will allow me to do this, although I will have to specify the method:

```
ggplot(nyc, aes(x=poverty, y=amtcapita))+
  geom_point(alpha=0.7, aes(color=borough, size=popn))+
  geom_smooth(method="lm", color="black", se=FALSE)+
  scale_y_log10(labels=dollar)+
  scale_color_brewer(palette="Dark2")+
  theme_bw()
```



I made a couple of important changes here that are quite subtle but important. First, I moves the aesthetics for color and size out of the `ggplot` command and put them into the `geom_point`. This is because I don't want those aesthetics to apply to all geoms. I only want them to apply to `geom_point`. If I had left them in they would have affected the `geom_smooth` and we would have had five separate lines for each borough.

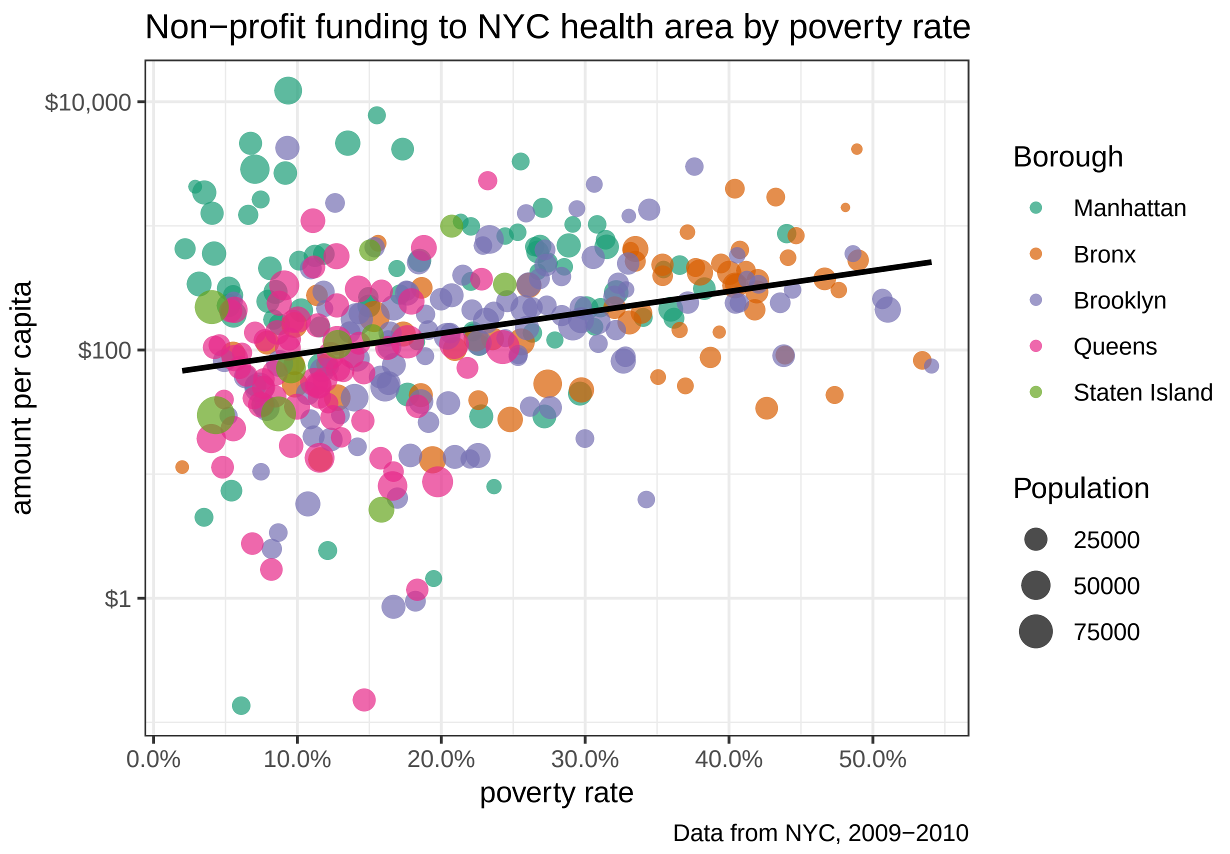
I also added `color="black"` to the `geom_smooth` command. Note that this is not part of an aesthetic call (e.g. `aes()`). It is not considered an aesthetic because we are just asking for the line to be a single color. Try surrounding that command in an `aes` and see what happens.

I also used the `se=FALSE`. If I don't do this then the line above will be surrounded by a confidence band, which may be good or bad. For our purposes, I did not want to clutter the graph.

We are now pretty close to being complete, but I still need to label all of my axes and provide a title. I also want better labeling for the two legends. This can all be done with the `labs` command which we append to the entire plot:

```
rm(percent)
```

```
ggplot(nyc, aes(x=poverty/100, y=amtcapita))+
  geom_point(alpha=0.7, aes(color=borough, size=popn))+
  scale_x_continuous(label=percent)+
  geom_smooth(method="lm", color="black", se=FALSE)+
  scale_y_log10(labels=dollar)+
  scale_color_brewer(palette="Dark2")+
  theme_bw()+
  theme(legend.position="right")+
  labs(x="poverty rate",
       y="amount per capita",
       title="Non-profit funding to NYC health area by poverty rate",
       caption="Data from NYC, 2009-2010",
       color="Borough",
       size="Population")
```

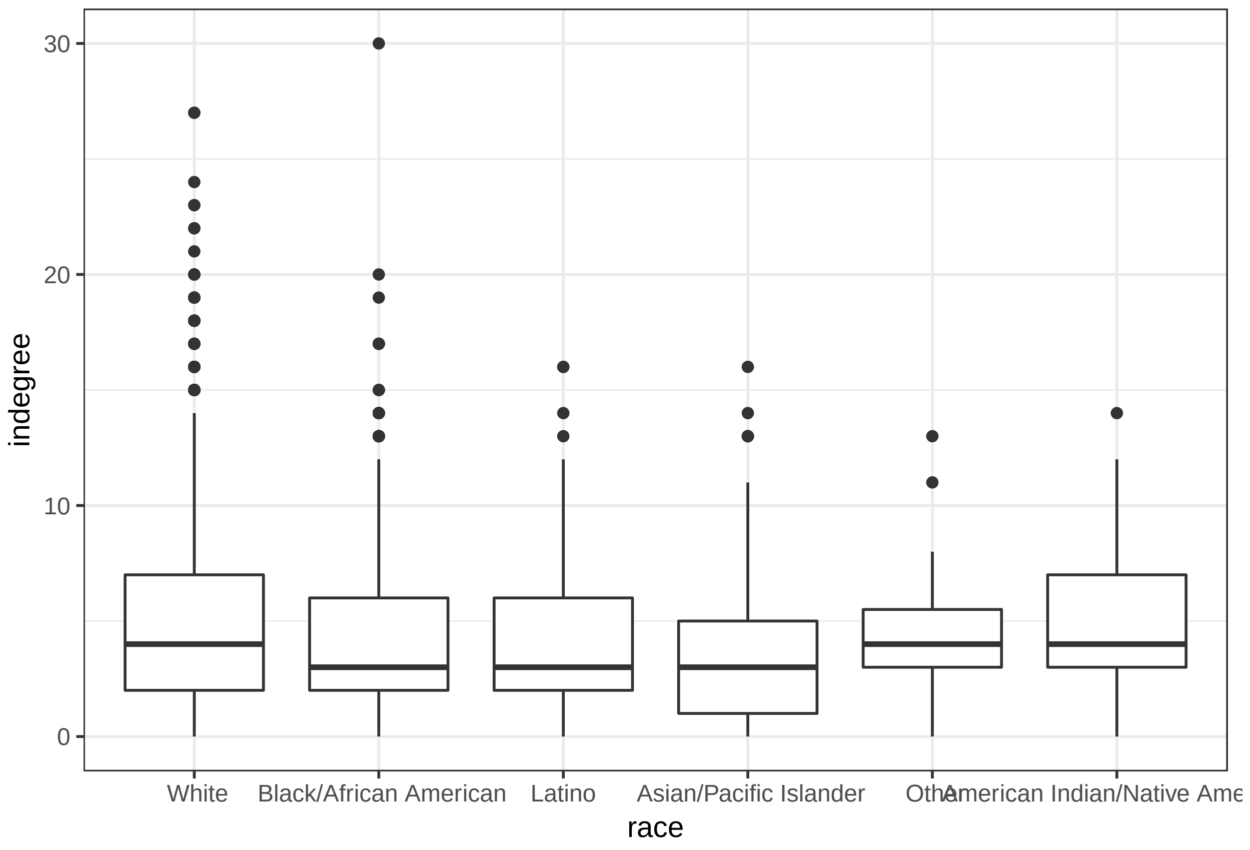


I did a couple of other things here as well. I added a `scale_x_continuous` so I could label the x tick mark labels as percents. I also added another `theme` command that would allow me to change the placement of the legends. I want to keep it on the right, but could have chosen "left", "right", "top", or "bottom."

We now have a very nice looking graph. Ggplot can be a little overwhelming at first, but it has quite a few advantages over base plot. It is designed so that we have to fidget around less with things like the size of our labels, the exact placement of our legends, and the margins of our table. All of that just works internally, and we can focus on the "grammar of graphics", i.e. the logic structure of what we are trying to say with our graph.

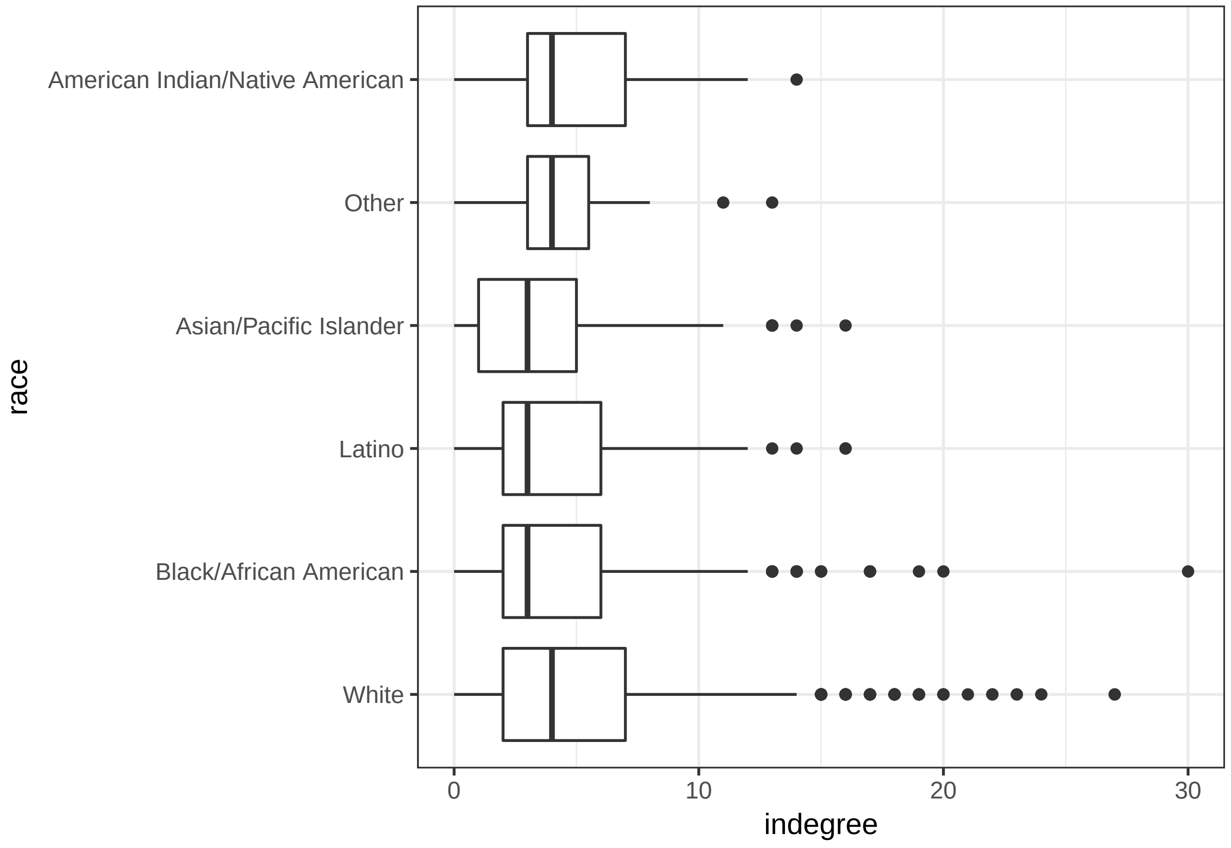
Lets do one more example to show how flexible ggplot is. Lets look at the distribution of popularity by race in the Add Health data. Because we have one categorical and one quantitative variable, we want comparative boxplots. Here is our basic set up:

```
load("example_datasets/add_health/addhealth.RData")
ggplot(addhealth, aes(x=race, y=indegree))+
  geom_boxplot()+
  theme_bw()
```



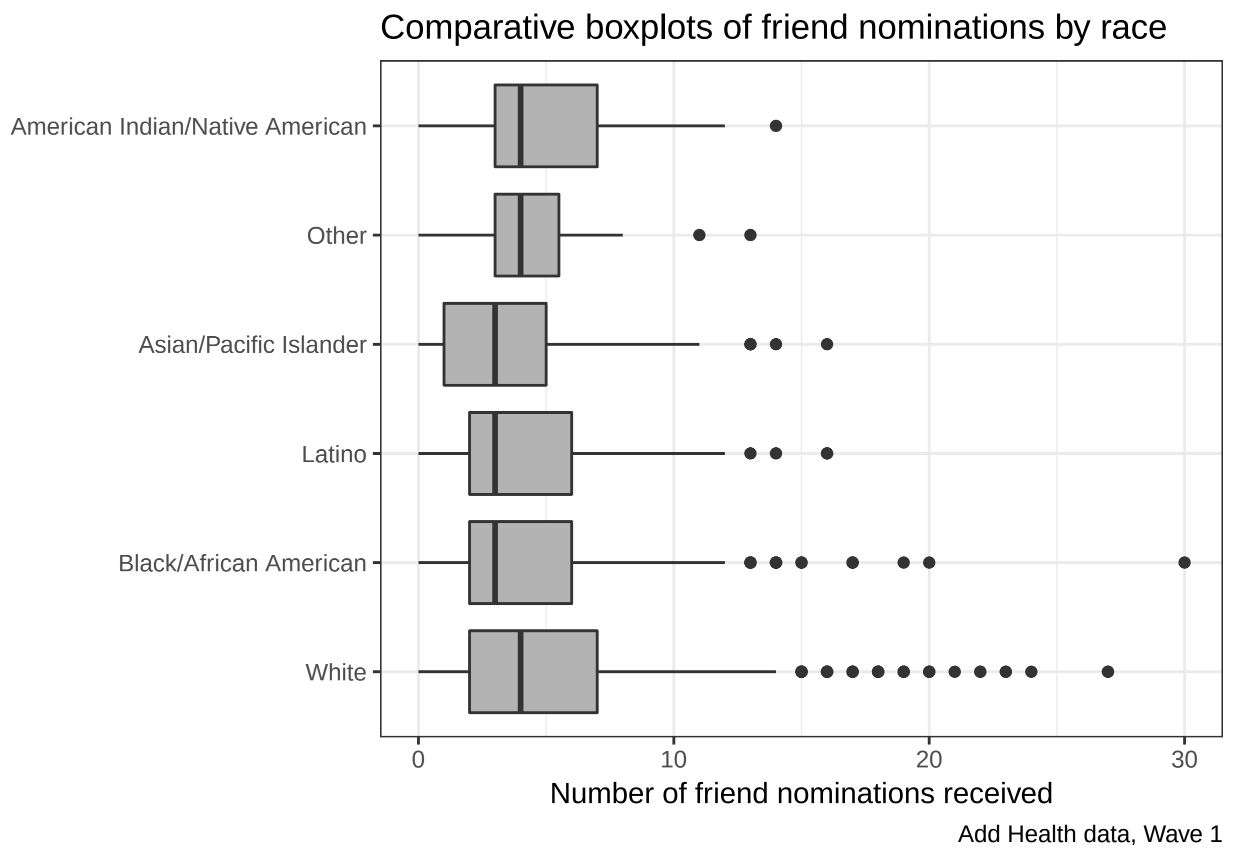
That works pretty well, However, its often better to display these boxplots horizontally so that we don't have to worry about category labels overlapping. We can do that with ggplot with the `coord_flip` command (an example of a coordinate layer):

```r
ggplot(addhealth, aes(x=race, y=indegree))+
  geom_boxplot()+
  coord_flip()+
  theme_bw()
```



This is already pretty good. Notice that I don't have to worry about specifying margins to make sure my category labels fit. Ggplot does that for me. I just need to apply labels and maybe a bit of tint to my boxplots.

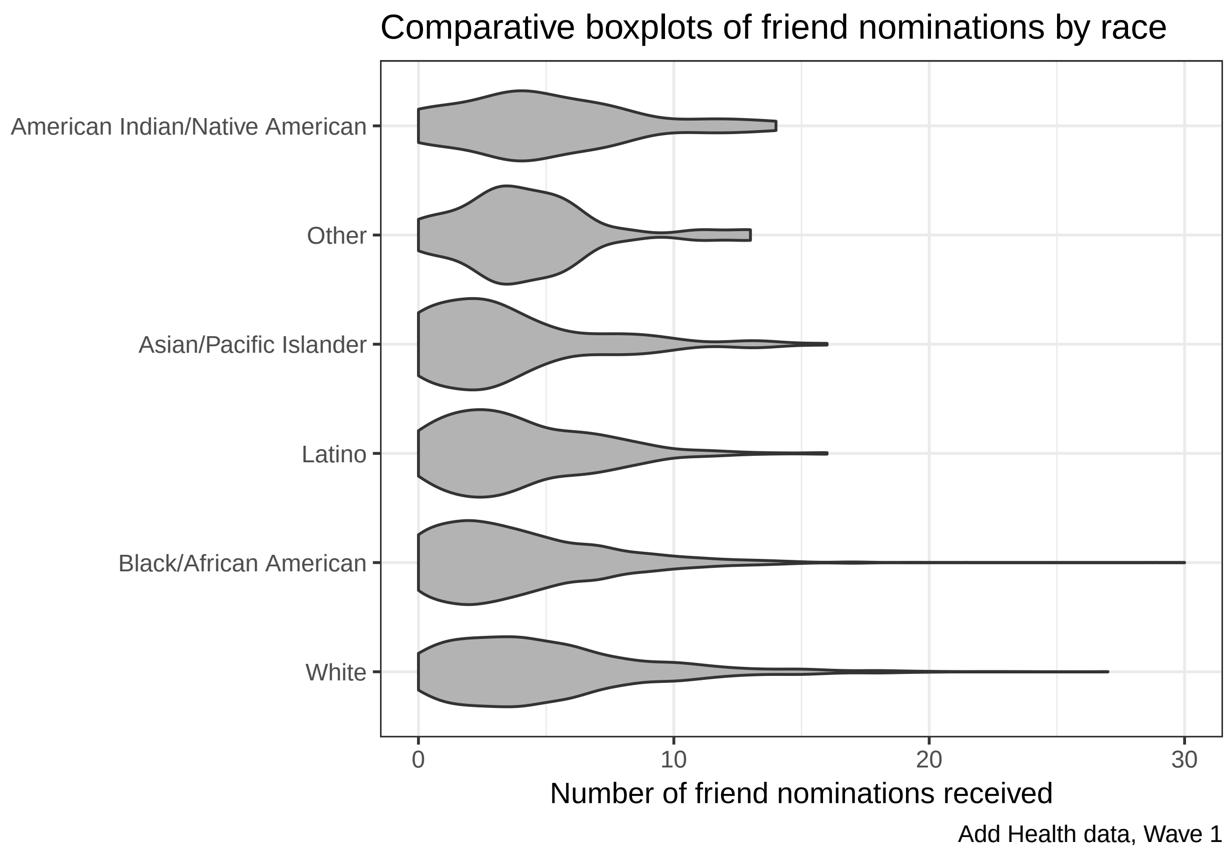
```
ggplot(addhealth, aes(x=race, y=indegree))+
  geom_boxplot(fill="grey70")+
  coord_flip()+
  theme_bw()+
  labs(x=NULL,
       y="Number of friend nominations received",
       title="Comparative boxplots of friend nominations by race",
       caption="Add Health data, Wave 1")
```



Note that the labels for x and y refer to the logic decision of which value is x and y as defined in the aesthetics not the actual placement, which was reversed due to the `coord_flip`. Note also that I used `NULL` for the x label because the category labels and title are self-explanatory.

There are also some more advanced geoms that do something similar to a boxplot. A popular one is the `geom_violin` which plots a mirror image of the density distribution. With ggplot, its as simple as swapping out my boxplot with the violin:

```r
ggplot(addhealth, aes(x=race, y=indegree))+
  geom_violin(fill="grey70")+
  coord_flip()+
  theme_bw()+
  labs(x=NULL,
       y="Number of friend nominations received",
       title="Comparative boxplots of friend nominations by race",
       caption="Add Health data, Wave 1")
```



I encourage you to explore the online ggplot documentation. I would also highly recommend Kieran Healy's new book, *Data Visualization* which uses ggplot extensively if you want to learn more about using ggplot most effectively.

Appendix D

# Reading and Writing Data

Students often get hung up at the start of a quantitative research project with the simple task of getting their data loaded into the statistical software package they are using. This is frequently a problem because data are distributed in inconsistent and often confusing ways by the agencies that release them. Knowing how to work with raw data in multiple formats is an important skill in being able to quickly get up and running with the more important parts of your analysis.

## Data Formats

Data typically come to us in one of two general formats: (1) plain text, or (2) binary. R also has tools for accessing data that is loaded into some kind of database format (e.g. SQL, MS Access) and R also has tools for "scraping" your own data from online sources. However, the vast majority of data comes in either text or binary format and so that is what we will focus on in this class.

Base R comes with several helpful methods for reading in plain text data such as `read.csv` and `read.fwf`. For a long time, binary data from a variety of sources could be read in using the `foreign` library. However, some more recent packages that are part of the tidyverse have been developed that provide much more easy and efficient ways to read in a variety of plain text and binary formats. I will use those libraries for all of the examples here. These libraries are:

- readr for reading in plain-text rectangular data.
- haven for reading in binary data files from SAS, Stata, and SPSS.
- readxl for reading in Excel files.

All three of these packages read data into an object called a tibble. Tibbles are the tidyverse upgrade to the venerable `data.frame`. In general, they operate just like data.frames with some nice additional features, so you shouldn't really need to worry about it. In a pinch, a tibble can be recast as a data.frame with the `as.data.frame` command:

```r
df <- as.data.frame(df)
```

## plain text files

Plain text files (also known as ASCII files), in contrast to binary files, are easily readable across any platform without specialized (and often proprietary) software. When you write a document in WordPad in Windows or TextEdit in OSX, you are writing a plain text file. When you write a document in Microsoft Word, you are writing a binary file. Plain text files are easily transportable across a variety of different program formats, usually take up less memory, and are better for tracking changes in version control systems. If you want to share data with others, plain text is the best format to use because it is accessible regardless of statistical software or computer platform.

While plain text has the advantage of accessibility and portability, it has the disadvantage of lacking any ability to add meta-characteristics to your data. Lets say that you coded an ordinal variable for highest degree earned in your data using a factor in R. When you output this data as plain-text, the variable names will simply show up as character strings with no information about proper ordering. When re-loaded into R or another statistical software package, the ordering of the variables that you so carefully specified will not be respected and they will simply be ordered in some default manner (e.g. alphabetically in R). When organizations release their data as plain text, they typically code all variables as numeric and provide a codebook that can be used to convert things like categorical variables and missing value codes. This is a good practice for data distribution, but not terribly helpful when we want to save the analytical data that we worked so hard to organize for our own project.

Data in plain text files usually comes in one of two formats: comma-separated values (CSV) files or fixed-width files. In both cases, one line of text corresponds to an observation, or a row of data. The difference between the two formats is how to distinguish the values for different variables within a line of text (i.e. the columns).

## Working with CSV files

In a comma-separated values format, the columns in the data are separated by commas. Here is an example with a very small CSV data file named "data.csv".

```
name,location,race,gender,yrsed
"Bobbi","Eugene,OR","Black","Female",12
"Jim","Seattle,WA","White","Male",11
"Mary","Oakland,CA","White","Female",16
```

There are a couple of things to note here about the format. First, the top line is the "header" line that gives the names of the variables. Its important for your statistical program to know whether the data file has a header line or not in order to properly process the file. Second, notice that character strings representing category labels are surrounded by quotes. This is good practice because character strings may sometimes include commas within them, as the location variable does in this case, and the program will treat that as a delimiter if not surrounded by quotation marks.

We can read this data into R with the `read_csv` command in the `readr` library:

```r
library(readr)
```

```
##
## Attaching package: 'readr'

## The following object is masked from 'package:scales':
##
##     col_factor
```

```r
mydata <- read_csv("resources/data.csv")
```

```
## Parsed with column specification:
## cols(
##   name = col_character(),
##   location = col_character(),
##   race = col_character(),
##   gender = col_character(),
##   yrsed = col_integer()
## )
```

```
mydata
```

```
## # A tibble: 3 x 5
##   name  location   race  gender yrsed
##   <chr> <chr>      <chr> <chr>  <int>
## 1 Bobbi Eugene,OR  Black Female    12
## 2 Jim   Seattle,WA White Male      11
## 3 Mary  Oakland,CA White Female    16
```

Notice that The `read_csv` command tried to guess what type of variable each column should be. It correctly guessed that `yrsed` was a quantitative variable. You can also give more explicity information about column types with ther `col_types` command. The `read_csv` command also assumed that the first line was a header row. If that were not true, I could change that with the `col_names=FALSE` argument.

The use of the comma as the delimiter between columns is pretty standard today, but you will occasionally find other delimiters used. The next most common delimiter is the tab (“"̂ in R speak). The `readr` package provides a handy function called `read_tsv` but we can also use the `read_delim` function to do the same thing and this will also show how easy it is to”roll your own" read function. Here is the same data as above, but this time separated by tabs and named “data_tab.txt”:

```
name    location    race    gender  yrsed
"Bobbi" "Eugene,OR" "Black" "Female"    12
"Jim"   "Seattle,WA"    "White" "Male"  11
"Mary"  "Oakland,CA"    "White" "Female"    16
```

I can read it in using the with the `read_delim` function:

```
mydata <- read_delim("resources/data_tab.txt", delim="\t")
```

```
## Parsed with column specification:
## cols(
##   name = col_character(),
##   location = col_character(),
##   race = col_character(),
##   gender = col_character(),
##   yrsed = col_integer()
## )
```

```
mydata
```

```
## # A tibble: 3 x 5
##   name  location   race  gender yrsed
##   <chr> <chr>      <chr> <chr>  <int>
## 1 Bobbi Eugene,OR  Black Female    12
## 2 Jim   Seattle,WA White Male      11
## 3 Mary  Oakland,CA White Female    16
```

All I had to do was specify what character served as the delimiter. In actuality, `read_csv` and `read_tsv` are just convenienct functions taht both call `read_delim`. If you access the help file for `read_delim`, you will see that there are many different arguments for dealing with specific problems that might arise in your dataset. Lets say for example that I had a data file saved as “data_messy.csv” that looked like this:

```
*Some data that I collected
*I cant remember when
name of person,location of person,racial category,gender of person,years of education
name,location,race,gender,yrsed
"Bobbi","Eugene,OR","Black","Female",12
"Jim","Seattle,WA","na","Male",11
```

```
"Mary","Oakland,CA","White","Female",16
```

There are several complications here. First, there are a couple of comment lines at the top where the comment symbol is "*" that I don't want to get processed. I could handle this with either the `skip` argument to skip a certain number of rows before reading the data or the `comment` argument to define what lines to skip by what character they start with. Second, there is a line above the proper headers with a description of each variable. I could use the `skip` option here again to skip this line. Finally, the lower-case "na" won't be recognized by default as a missing value by R, but the option `na` will allow me to specify additional character strings like "na" that should be interpreted as missing values. All together, I use the command:

```
mydata <- read_delim("resources/data_messy.csv", delim=",",
                     comment="*", skip=1, na="na")
```

```
## Parsed with column specification:
## cols(
##   name = col_character(),
##   location = col_character(),
##   race = col_character(),
##   gender = col_character(),
##   yrsed = col_integer()
## )
```

```
mydata
```

```
## # A tibble: 3 x 5
##   name  location   race  gender yrsed
##   <chr> <chr>      <chr> <chr>  <int>
## 1 Bobbi Eugene,OR  Black Female    12
## 2 Jim   Seattle,WA <NA>  Male      11
## 3 Mary  Oakland,CA White Female    16
```

Note that because I used the `comment` argument I only specify `skip=1` .

## Working with fixed-width text files

The second form that data in text format can take is "fixed-width" format where the specific length of each variable in terms of the number of characters is specified. For example, here is the same dataset in fixed-width format, saved as "data_fw.txt":

```
BobbiEugene,OR BlackFemale12
Jim  Seattle,WAWhiteMale  11
Mary Oakland,CAWhiteFemale16
```

Notice that the actual starting location of each variable is the same within each row. If you count the characters up, you will see that the first variable has a width of 5 characters, the second variable has a width of 10 characters, and so on. Note also that this file does not contain headers which is pretty typical of fixed width files. We can use the `read_fwf` command by feeding in this data, by explicitly specifying the starting and endind positions of each variable:

```
mydata <- read_fwf("resources/data_fw.txt",
                   col_positions = fwf_positions(start=c(1, 6,16,21,27),
                                                 end  =c(5,15,20,26,28),
                                                 col_names=c("name","location","race","gender","yrsed"))
```

```
## Parsed with column specification:
## cols(
##   name = col_character(),
##   location = col_character(),
##   race = col_character(),
##   gender = col_character(),
##   yrsed = col_integer()
## )
```

```
mydata
```

```
## # A tibble: 3 x 5
##   name  location   race  gender yrsed
##   <chr> <chr>      <chr> <chr>  <int>
## 1 Bobbi Eugene,OR  Black Female    12
## 2 Jim   Seattle,WA White Male      11
## 3 Mary  Oakland,CA White Female    16
```

I have to do a little more work than I did with CSV files. First I need to define a `col_positions` argument and feed in the results from the function `fwf_positions`. This `fwf_positions` function needs to have a vector of starting and ending positions for each variable. I also use the `col_names` argument in this function to assign names to each of my variables.

Now I want to show you an example from my own work that demonstrates how flexible these methods are. The data here are fixed-width data created from an IPUMS extract of the 1980 US Census. The data come zipped up as a "G-zip" file with suffix "gz" which is a standard method of compressing data files. You can download the file here. You can also download the codebook that IPUMS provides which shows where the start and ending positions are for each variable.

One of the nice features of all of the tidyverse input packages is that they can read in data directly from zipped files without having to unzip them first. This saves you room on your computer and in your git repository (GitHub has a limit of 100MB for files). I am going to use that feature here as well as a couple of other nifty features. Here is my input command:

```
census1980 <- read_fwf("resources/usa_00074.dat.gz",
                       col_positions = fwf_positions(start = c(1, 7,15,25,33,37,47,48,51,52,53,
                                                               56,60,63,71,75,83,86,90,
                                                               93,94,97,100,103,107,109),
                                                     end   = c(4,14,24,27,36,46,47,50,51,
                                                               52,54,58,62,65,74,76,85,89,92,
                                                               93,96,99,102,106,108,111),
                                                     col_names = c("year","hhid","hhwt","metarea",
                                                                   "pernum","perwt","sex",
                                                                   "age","marst","marrno",
                                                                   "agemarr","raced","hispand",
                                                                   "bpl","yrimmig","language",
                                                                   "educd","pernum_sp","age_sp",
                                                                   "marrno_sp","raced_sp","hispand_sp",
                                                                   "bpl_sp","yrimmig_sp","language_sp",
                                                                   "educd_sp")),
                       col_types = cols(.default = "i"), #ensure that all variables are read in as inte
                       progress = FALSE)
census1980
```

```
## # A tibble: 5,621,191 x 26
##     year  hhid  hhwt metarea pernum perwt   sex   age marst marrno agemarr
##    <int> <int> <int>   <int>  <int> <int> <int> <int> <int>  <int>   <int>
##  1  1980     1  2000      38      1  2000     2    45     1      1      19
##  2  1980     1  2000      38      2  2000     1    50     1      1      25
##  3  1980     1  2000      38      3  2000     1    23     6      0       0
##  4  1980     2  2000      38      1  2000     2    58     1      2      22
##  5  1980     2  2000      38      2  2000     1    59     1      2      27
##  6  1980     3  2000      38      1  2000     1    32     1      1      21
##  7  1980     3  2000      38      2  2000     2    29     1      1      18
##  8  1980     4  2000      38      1  2000     1    57     1      2      20
##  9  1980     4  2000      38      2  2000     2    47     1      2      17
## 10  1980     4  2000      38      3  2000     1    24     6      0       0
## # ... with 5,621,181 more rows, and 15 more variables: raced <int>,
## #   hispand <int>, bpl <int>, yrimmig <int>, language <int>, educd <int>,
## #   pernum_sp <int>, age_sp <int>, marrno_sp <int>, raced_sp <int>,
## #   hispand_sp <int>, bpl_sp <int>, yrimmig_sp <int>, language_sp <int>,
## #   educd_sp <int>
```

If you look closely, you will see that I am skipping some index positions in my data because I don't want to bother with some variables. I am also using the `col_types` argument to force `read_fwf` to read in all variables as numeric integers because this is how all the data come from IPUMS. Finally, I set `progress=FALSE` for the output here, but I can ghange that to the `progress=TRUE` option to get a progresss bar as my data is read in that lets me know how long I can expect to wait.

## Data in binary format

As a result of initiatives to make science more open, data is increasingly becoming available in simple text format, which improves its portability and accessibility. However, there are still many cases where data is available in a binary format that is readable only by a specific statistical software program. For example, the quick download page for the General Social Survey provides the comprehensive GSS data files but only in Stata and SPSS formats. Thats fine if you have purchased that software, but you are out of luck if you have not. This approach is Bad For Science, but you will still run into it quite a bit.

Luckily, R can usually still read those files. For aeons upon aeons, the preferred packages for this was `foreign`, but the tidyverse has changed the game. The `haven` package can easily read in data from Stata, SAS, and SPSS. The `readxl` package can easily read in data from an Excel file.

To test these libraries out, I have loaded the data exampleI have been using into Stata and saved it as a binary stata dataset (*.dta). I can use the `read_dta` function in `haven` to read in this data.

```r
library(haven)
mydata <- read_dta("resources/data.dta")
mydata
```

```
## # A tibble: 3 x 5
##   name  location   gender yrsed race
##   <chr> <chr>      <chr>  <dbl> <dbl+lbl>
## 1 Bobbi Eugene,OR  Female    12 1
## 2 Jim   Seattle,WA Male      11 2
## 3 Mary  Oakland,CA Female    16 2
```

Everything looks good except that because I didn't encode location and gender in Stata as categorical variables, those variables show up in R as character strings rather than factor variables. If I wanted to turn my either variable into a proper factor (categorical) variable, I can use the `factor` command:

```r
mydata$gender <- factor(mydata$gender)
summary(mydata$gender)
```

```
## Female   Male
##      2      1
```

## Saving data

R has its own binary format for keeping track of data. You can save any object or set of objects to your filesystem with the `save` command. This will save a file in a binary *.RData format. These objects can then be loaded back into R with the `load` command:

```
save(mydata, file="resources/data.RData")
load("resources/data.RData")
mydata
```

```
## # A tibble: 3 x 5
##   name  location   gender yrsed race
##   <chr> <chr>      <fct>  <dbl> <dbl+lbl>
## 1 Bobbi Eugene,OR  Female    12 1
## 2 Jim   Seattle,WA Male      11 2
## 3 Mary  Oakland,CA Female    16 2
```

I recommend saving your own analytical data as RData files because these files are light-weight and contain all of the added meta-information that you have created for categorical variables and such. However, for sharing data more widely I recommend that you provide data in plain text format. The `readr` has a variety of `write_*` functions including `write_csv`, `write_delim`, and `write_tsv` for outputing data as delimited plain text. For example, I could write my data to a csv with:

```
write_csv(mydata, path="resources/mydata_R.csv")
```

```
name,location,gender,yrsed,race
Bobbi,"Eugene,OR",Female,12,1
Jim,"Seattle,WA",Male,11,2
Mary,"Oakland,CA",Female,16,2
```

The `haven` package also has several `write_*` functions for outputing data in other binary formats. Here are the most common:

- `write_dta` for a Stata data file
- `write_sav` for a SPSS data file
- `write_xpt` for a SAS transport file and `write_sas` for a SAS data file