Math 145: Introduction to Biostatistics

R Pruim

Fall 2013
# Contents

0 **What Is Statistics?** 5  
0.1 A First Example: The Lady Tasting Tea 6  
0.2 Coins and Cups 8  

1 **Data and Where It Comes From** 11  
1.1 Data 11  
1.2 Samples and Populations 12  
1.3 Types of Statistical Studies 13  
1.4 Data in R 13  
1.5 Important Distinctions 16  

2 **Describing Data** 17  
2.1 Getting Started With RStudio 17  
2.2 Four Things to Know About R 19  
2.3 Data in R 20  
2.4 The Most Important Template 22  
2.5 Summaries of One Variable 22  
2.6 Creating Tables 26  
2.7 Looking at multiple variables at once 28  
2.8 Exporting Plots 32  
2.9 Using R Markdown 33  
2.10 A Few Bells and Whistles 34  
2.11 Graphical Summaries – Important Ideas 40  
2.12 Getting Help in RStudio 41  
2.13 From Excel or Google to R 42  
2.14 Manipulating your data 48  
2.15 Saving Data 50  

3 **Confidence Intervals** 55  
3.1 Sampling Distributions 55  
3.2 Interval Estimates 55  
3.3 The Bootstrap 57  
3.4 More Examples 59
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Hypothesis Tests</td>
<td>67</td>
</tr>
<tr>
<td></td>
<td>4.1 Introduction to Hypothesis Testing</td>
<td>67</td>
</tr>
<tr>
<td></td>
<td>4.2 P-values</td>
<td>67</td>
</tr>
<tr>
<td></td>
<td>4.3 Significance</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>4.4 Creating Randomization Distributions</td>
<td>68</td>
</tr>
<tr>
<td>5</td>
<td>Approximating with a Distribution</td>
<td>79</td>
</tr>
<tr>
<td></td>
<td>5.1 The Family of Normal Distributions</td>
<td>79</td>
</tr>
<tr>
<td>6</td>
<td>Inference with Normal Distributions</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>6.1 Inference for One Proportion</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>6.2 Inference for One Mean</td>
<td>93</td>
</tr>
</tbody>
</table>
This is a course primarily about statistics, but what exactly is statistics? In other words, what is this course about? Here are some definitions of statistics from other people:

- a collection of procedures and principles for gaining information in order to make decisions when faced with uncertainty (J. Utts [Utt05]),
- a way of taming uncertainty, of turning raw data into arguments that can resolve profound questions (T. Amabile [fMA89]),
- the science of drawing conclusions from data with the aid of the mathematics of probability (S. Garfunkel [fMA86]),
- the explanation of variation in the context of what remains unexplained (D. Kaplan [Kap09]),
- the mathematics of the collection, organization, and interpretation of numerical data, especially the analysis of a population’s characteristics by inference from sampling (American Heritage Dictionary [AmH82]).

Here’s a simpler definition:

Statistics is the science of answering questions with data.

This definition gets at two important elements of the longer definitions above:

Data – the raw material

Data are the raw material for doing statistics. We will learn more about different types of data, how to collect data, and how to summarize data as we go along.

1 As we will see, the words statistic and statistics get used in more than one way. More on that later.
What Is Statistics?

Information – the goal

The goal of doing statistics is to gain some information or to make a decision – that is, to answer some question. Statistics is useful because it helps us answer questions like the following:

- Which of two treatment plans leads to the best clinical outcomes?
- Are men or women more successful at quitting smoking? And does it matter which smoking cessation program they use?
- Is my cereal company complying with regulations about the amount of cereal in its cereal boxes?

In this sense, statistics is a science – a method for obtaining new knowledge. Our simple definition is light on describing the context in which this takes place. So let’s add two more important aspects of statistics.

Uncertainty – the context

The tricky thing about statistics is the uncertainty involved. If we measure one box of cereal, how do we know that all the others are similarly filled? If every box of cereal were identical and every measurement perfectly exact, then one measurement would suffice. But the boxes may differ from one another, and even if we measure the same box multiple times, we may get different answers to the question How much cereal is in the box?

So we need to answer questions like How many boxes should we measure? and How many times should we measure each box? Even so, there is no answer to these questions that will give us absolute certainty. So we need to answer questions like How sure do we need to be?

Probability – the tool

In order to answer a question like How sure do we need to be?, we need some way of measuring our level of certainty. This is where mathematics enters into statistics. Probability is the area of mathematics that deals with reasoning about uncertainty.

0.1 A First Example: The Lady Tasting Tea

There is a famous story about a lady who claimed that tea with milk tasted different depending on whether the milk was added to the tea or the tea added to the milk. The story is famous because of the setting in which she made this claim. She was attending a party in Cambridge, England, in the 1920s. Also in attendance were a number of university dons and their wives. The scientists in attendance scoffed at the woman and her claim. What, after all, could be the difference?

All the scientists but one, that is. Rather than simply dismiss the woman’s claim, he proposed that they decide how one should test the claim. The tenor of the conversation changed at this suggestion, and the scientists began to discuss how the claim should be tested. Within a few minutes cups of tea with milk had been prepared and presented to the woman for tasting.

Let’s take this simple example as a prototype for a statistical study. What steps are involved?

1. Determine the question of interest.

\[^2\] The opening pages of each chapter of our book include many more questions.
What Is Statistics?

Just what is it we want to know? It may take some effort to make a vague idea precise. The precise questions may not exactly correspond to our vague questions, and the very exercise of stating the question precisely may modify our question. Sometimes we cannot come up with any way to answer the question we really want to answer, so we have to live with some other question that is not exactly what we wanted but is something we can study and will (we hope) give us some information about our original question.

In our example this question seems fairly easy to state: Can the lady tell the difference between the two tea preparations? But we need to refine this question. For example, are we asking if she always correctly identifies cups of tea or merely if she does better than we could do ourselves (by guessing)?

2. Determine the population.

Just who or what do we want to know about? Are we only interested in this one woman or women in general or only women who claim to be able to distinguish tea preparations?

3. Select measurements.

We are going to need some data. We get our data by making some measurements. These might be physical measurements with some device (like a ruler or a scale). But there are other sorts of measurements too, like the answer to a question on a form. Sometimes it is tricky to figure out just what to measure. (How do we measure happiness or intelligence, for example?) Just how we do our measuring will have important consequences for the subsequent statistical analysis. The recorded values of these measurements are called variables (because the values vary from one individual to another).

In our example, a measurement may consist of recording for a given cup of tea whether the woman’s claim is correct or incorrect.

4. Determine the sample.

Usually we cannot measure every individual in our population; we have to select some to measure. But how many and which ones? These are important questions that must be answered. Generally speaking, bigger is better, but it is also more expensive. Moreover, no size is large enough if the sample is selected inappropriately.

Suppose we gave the lady one cup of tea. If she correctly identifies the mixing procedure, will we be convinced of her claim? She might just be guessing; so we should probably have her taste more than one cup. Will we be convinced if she correctly identifies 5 cups? 10 cups? 50 cups?

What if she makes a mistake? If we present her with 10 cups and she correctly identifies 9 of the 10, what will we conclude? A success rate of 90% is, it seems, much better than just guessing, and anyone can make a mistake now and then. But what if she correctly identifies 8 out of 10? 80 out of 100?

And how should we prepare the cups? Should we make 5 each way? Does it matter if we tell the woman that there are 5 prepared each way? Should we flip a coin to decide even if that means we might end up with 3 prepared one way and 7 the other way? Do any of these differences matter?

5. Make and record the measurements.

Once we have the design figured out, we have to do the legwork of data collection. This can be a time-consuming and tedious process. In the case of the lady tasting tea, the scientists decided to present her with ten cups of tea which were quickly prepared. A study of public opinion may require many thousands of phone calls or personal interviews. In a laboratory setting, each measurement might be the result of a carefully performed laboratory experiment.

6. Organize the data.

Once the data have been collected, it is often necessary or useful to organize them. Data are typically stored in spreadsheets or in other formats that are convenient for processing with statistical packages. Very large data sets are often stored in databases.

Part of the organization of the data may involve producing graphical and numerical summaries of the data. These summaries may give us initial insights into our questions or help us detect errors that may have occurred to this point.
7. Draw conclusions from data.

Once the data have been collected, organized, and analyzed, we need to reach a conclusion. Do we believe the woman’s claim? Or do we think she is merely guessing? How sure are we that this conclusion is correct?

Eventually we will learn a number of important and frequently used methods for drawing inferences from data. More importantly, we will learn the basic framework used for such procedures so that it should become easier and easier to learn new procedures as we become familiar with the framework.

8. Produce a report.

Typically the results of a statistical study are reported in some manner. This may be as a refereed article in an academic journal, as an internal report to a company, or as a solution to a problem on a homework assignment. These reports may themselves be further distilled into press releases, newspaper articles, advertisements, and the like. The mark of a good report is that it provides the essential information about each of the steps of the study.

As we go along, we will learn some of the standard terminology and procedures that you are likely to see in basic statistical reports and will gain a framework for learning more.

At this point, you may be wondering who the innovative scientist was and what the results of the experiment were. The scientist was R. A. Fisher, who first described this situation as a pedagogical example in his 1925 book on statistical methodology [Fis25]. Fisher developed statistical methods that are among the most important and widely used methods to this day, and most of his applications were biological.

0.2 Coins and Cups

You might also be curious about how the experiment came out. How many cups of tea were prepared? How many did the woman correctly identify? What was the conclusion?

Fisher never says. In his book he is interested in the method, not the particular results. But let’s suppose we decide to test the lady with ten cups of tea. We’ll flip a coin to decide which way to prepare the cups. If we flip a head, we will pour the milk in first; if tails, we put the tea in first. Then we present the ten cups to the lady and have her state which ones she thinks were prepared each way.

It is easy to give her a score (9 out of 10, or 7 out of 10, or whatever it happens to be). It is trickier to figure out what to do with her score. Even if she is just guessing and has no idea, she could get lucky and get quite a few correct – maybe even all 10. But how likely is that?

Let’s try an experiment. I’ll flip 10 coins. You guess which are heads and which are tails, and we’ll see how you do.

Comparing with your classmates, we will undoubtedly see that some of you did better and others worse.

Now let’s suppose the lady gets 9 out of 10 correct. That’s not perfect, but it is better than we would expect for someone who was just guessing. On the other hand, it is not impossible to get 9 out of 10 just by guessing. So here is Fisher’s great idea: Let’s figure out how hard it is to get 9 out of 10 by guessing. If it’s not so hard to do, then perhaps that’s just what happened, so we won’t be too impressed with the lady’s tea tasting ability. On the other hand, if it is really unusual to get 9 out of 10 correct by guessing, then we will have some evidence that she must be able to tell something.

But how do we figure out how unusual it is to get 9 out of 10 just by guessing? We’ll learn another method later, but for now, let’s just flip a bunch of coins and keep track. If the lady is just guessing, she might as well be flipping a coin.
So here’s the plan. We’ll flip 10 coins. We’ll call the heads correct guesses and the tails incorrect guesses. Then we’ll flip 10 more coins, and 10 more, and 10 more, and … . That would get pretty tedious. Fortunately, computers are good at tedious things, so we’ll let the computer do the flipping for us using a tool in the mosaic package. This package is already installed in our RStudio server. If you are running your own installation of R you can install mosaic using the following command:

```
install.packages("mosaic")
```

The `rflip()` function can flip one coin

```r
require(mosaic)
rflip()
```

Flipping 1 coin [ Prob(Heads) = 0.5 ] ...

H

Result: 1 heads.

or a number of coins

```r
rflip(10)
```

Flipping 10 coins [ Prob(Heads) = 0.5 ] ...

H H H H T T T T T

Result: 5 heads.

and show us the results.

Typing `rflip(10)` a bunch of times is almost as tedious as flipping all those coins. But it is not too hard to tell R to `do()` this a bunch of times.

```r
do(2) * rflip(10)
```

<table>
<thead>
<tr>
<th>n</th>
<th>heads</th>
<th>tails</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>4</td>
</tr>
</tbody>
</table>

Let’s get R to `do()` it for us 10,000 times and make a table of the results.

```r
results <- do(10000) * rflip(10)
table(results$heads)
```

<table>
<thead>
<tr>
<th>heads</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>102</td>
<td>467</td>
<td>1203</td>
<td>2048</td>
<td>2470</td>
<td>2035</td>
<td>1140</td>
<td>415</td>
<td>108</td>
<td>7</td>
</tr>
</tbody>
</table>

Math 145 : Fall 2013 : Pruim

Last Modified: November 5, 2013
What Is Statistics?

We could also use `tally()` for this.

You might be surprised to see that the number of correct guesses is exactly 5 (half of the 10 tries) only 25% of the time. But most of the results are quite close to 5 correct. 67% of the results are 4, 5, or 6, for example. And 1% of the results are between 3 and 7 (inclusive). But getting 8 correct is a bit unusual, and getting 9 or 10 correct is even more unusual.

So what do we conclude? It is possible that the lady could get 9 or 10 correct just by guessing, but it is not very likely (it only happened in about 1.2% of our simulations). So one of two things must be true:

- The lady got unusually “lucky”, or
- The lady is not just guessing.

Although Fisher did not say how the experiment came out, others have reported that the lady correctly identified all 10 cups! [Sal01]

This same reasoning can be applied to answer a wide range of questions that have a similar form. For example, the question of whether dogs can smell cancer could be answered essentially the same way (although it would be a bit more involved than preparing tea and presenting cups to the Lady).
1.1 Data

Imagine data as a 2-dimensional structure (like a spreadsheet).

- Rows correspond to observational units (people, animals, plants, or other objects we are collecting data about).
- Columns correspond to variables (measurements collected on each observational unit).
- At the intersection of a row and a column is the value of the variable for a particular observational unit.

Observational units go by many names, depending on the kind of thing being studied. Popular names include subjects, individuals, and cases. Whatever you call them, it is important that you always understand what your observational units are.

Variable terminology

categorical variable a variable that places observational units into one of two or more categories (examples: color, sex, case/control status, species, etc.)

These can be further sub-divided into ordinal and nominal variables. If the categories have a natural and meaningful order, we will call them ordered or ordinal variables. Otherwise, they are nominal variables.

quantitative variable a variable that records measurements along some scale (examples: weight, height, age, temperature) or counts something (examples: number of siblings, number of colonies of bacteria, etc.)

Quantitative variables can be continuous or discrete. Continuous variables can (in principle) take on any real-number value in some range. Values of discrete variables are limited to some list and “in-between values” are not possible. Counts are a good example of discrete variables.

response variable a variable we are trying to predict or explain

explanatory variable a variable used to predict or explain a response variable
Distributions

The distribution of a variable answers two questions:
- What values can the variable have?
- With what frequency does each value occur?
  The frequency may be described in terms of counts, proportions (often called relative frequency), or densities (more on densities later).

A distribution may be described using a table (listing values and frequencies) or a graph (e.g., a histogram) or with words that describe general features of the distribution (e.g., symmetric, skewed).

1.2 Samples and Populations

population the collection of animals, plants, objects, etc. that we want to know about

sample the (smaller) set of animals, plants, objects, etc. about which we have data

parameter a number that describes a population or model.

statistic a number that describes a sample.

Much of statistics centers around this question:

What can we learn about a population from a sample?

Estimation

Often we are interested in knowing (approximately) the value of some parameter. A statistic used for this purpose is called an estimate. For example, if you want to know the mean length of the tails of lemurs (that’s a parameter), you might take a sample of lemurs and measure their tails. The mean length of the tails of the lemurs in your sample is a statistic. It is also an estimate, because we use it to estimate the parameter.

Statistical estimation methods attempt to

- reduce bias, and
- increase precision.

bias the systematic tendency of sample estimates to either overestimate or underestimate population parameters; that is, a systematic tendency to be off in a particular direction.

precision the measure of how close estimates are to the thing being estimated (called the estimand).
Sampling

**Sampling** is the process of selecting a sample. Statisticians use **random samples**

- to avoid (or at least reduce) **bias**, and
- so they can quantify **sampling variability** (the amount samples differ from each other), which in turn allows us to quantify precision.

The simplest kind of random sample is called a **simple random sample** (aren’t statisticians clever about naming things?). A simple random sample is equivalent to putting all individuals in the population into a big hat, mixing thoroughly, and selecting some out of the hat to be in the sample. In particular, in a simple random sample, *every individual has an equal chance to be in the sample*, in fact, *every subset of the population of a fixed size has an equal chance to be in the sample.*

Other sampling methods include

- **convenience sampling** using whatever individuals are easy to obtain
  - This is usually a terrible idea. If the convenient members of the population differ from the inconvenient members, then the sample will not be representative of the population.
- **volunteer sampling** using people who volunteer to be in the sample
  - This is usually a terrible idea. Most likely the volunteers will differ in some ways from the non-volunteers, so again the sample will not be representative of the population.
- **systematic sampling** sampling done in some systematic way (every tenth unit, for example).
  - This can sometimes be a reasonable approach.
- **stratified sampling** sampling separately in distinct sub-populations (called **strata**)
  - This is more complicated (and sometimes necessary) but fine as long as the sampling methods in each stratum are good and the analysis takes the sampling method into account.

1.3 Types of Statistical Studies

Statisticians use the word experiment to mean something very specific. In an **experiment**, the researcher determines the values of one or more (explanatory) variables, typically by random assignment. If there is no such assignment by the researcher, the study is an **observational study**.

1.4 Data in R

Data sets in R are usually stored as **data frames** in a rectangular arrangement with rows corresponding to observational unites and columns corresponding to variables. A number of data sets are built into R and its packages. Let’s take a look at **CricketChirps**, a small data set that comes with the Lock5 text.

```r
require(Lock5Data) # Tell R to use the package for our text book
```

If we type the name of the data set, R will display it for us.
This data set has 7 observational units (what are they?) and 2 variables (what are they?) For larger data sets, it is more useful to look at some sort of summary or subset of the data. Here are the first few rows of the iris data set.

```r
head(iris)
```

```
Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1 5.1 3.5 1.4 0.2 setosa
2 4.9 3.0 1.4 0.2 setosa
3 4.7 3.2 1.3 0.2 setosa
4 4.6 3.1 1.5 0.2 setosa
5 5.0 3.6 1.4 0.2 setosa
6 5.4 3.9 1.7 0.4 setosa
```

Here are some summaries of the data set:

```r
str(iris)  # structure of the data set
```

```
'data.frame': 150 obs. of  5 variables:
$ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
$ Sepal.Width : num 3.5 3.2 3.1 3.6 3.9 3.4 3.4 3.9 2.9 3.1 ...
$ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
$ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
$ Species     : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1 ...
```

```r
summary(iris)  # summary of each variable
```

```

<table>
<thead>
<tr>
<th></th>
<th>Min.</th>
<th>1st Qu.</th>
<th>Median</th>
<th>Mean</th>
<th>3rd Qu.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sepal.Length</td>
<td>4.30</td>
<td>5.10</td>
<td>5.80</td>
<td>5.84</td>
<td>6.40</td>
<td>7.90</td>
</tr>
<tr>
<td>Sepal.Width</td>
<td>2.00</td>
<td>2.80</td>
<td>3.00</td>
<td>3.06</td>
<td>3.30</td>
<td>3.30</td>
</tr>
<tr>
<td>Petal.Length</td>
<td>1.00</td>
<td>1.60</td>
<td>4.35</td>
<td>3.76</td>
<td>5.10</td>
<td>6.40</td>
</tr>
<tr>
<td>Petal.Width</td>
<td>0.10</td>
<td>0.30</td>
<td>1.30</td>
<td>1.20</td>
<td>1.80</td>
<td>2.50</td>
</tr>
<tr>
<td>Species</td>
<td>setosa :50</td>
<td>versicolor:50</td>
<td>virginica :50</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```r
nrow(iris)  # how many rows?
```

```
[1] 150
```
Many of the datasets in R have useful help files that describe the data and explain how they were collected or give references to the original studies. You can access this information for the iris data set by typing

\texttt{\texttt{?iris}}

We’ll learn how to make more customized summaries (numerical and graphical) soon. For now, it is only important to observe how the organization of data in R reflects the observational units and variables in the data set.

This is important if you want to construct your own data set (in Excel or a google spreadsheeet, for example) that you will later import into R. You want to be sure that the structure of your spread sheet uses rows and columns in this same way, and that you don’t put any extra stuff into the spread sheet. It is a good idea to include an extra row at the top which names the variables.

Going back to the CricketChirps data, here’s how it might look if we collect that same data in Excel:

![Excel spreadsheet image]

We’ll learn how to get the data from Excel into R soon.
1.5 Important Distinctions

When learning the vocabulary in this chapter, it is useful to focus on the distinctions being made:

- cases vs. variables
- categorical vs. quantitative
  - (nominal vs. ordinal)
  - (discrete vs. continuous)
- experiment vs. observational study
- population vs. sample
- parameter vs. statistic
  - biased vs. unbiased
In this chapter we discuss graphical and numerical summaries of data. These notes focus primarily on how to get R to do the work for you. The textbook includes more information about why and when to use various kinds of summaries.

2.1 Getting Started With RStudio

You can access the Calvin RStudio server via links from our course web site (http://www.calvin.edu/~rpruim/courses/m145/f13/) or from http://dahl.calvin.edu.

2.1.1 Logging in and changing your password

You will be prompted to log in. Your login and password are both your Calvin userid. Once you are logged in, you will see something like Figure 2.1.

You should change your password. Here’s how.

1. From the Tools menu, select Shell
2. Type yppasswd
3. You will be prompted for your old password, then your new password twice.
4. If you give a sufficiently strong new password (at least six letters, at least one capital, etc.) you will receive notice that your password has been reset. If there was a problem, you will see a message about it and can try again.
5. Once you have reset your password, click on Close to close the shell and get back to RStudio.

2.1.2 Using R as a calculator

Notice that RStudio divides its world into four panels. Several of the panels are further subdivided into multiple tabs. The console panel is where we type commands that R will execute.

R can be used as a calculator. Try typing the following commands in the console panel.
You can save values to named variables for later reuse

```r
product = 15.3 * 23.4  # save result
product
# show the result

[1] 358
```

```r
product <- 15.3 * 23.4  # <- is assignment operator, same as =
product

[1] 358
```

```r
15.3 * 23.4 -> newproduct  # -> assigns to the right
newproduct
```

5 + 3

[1] 8

15.3 * 23.4

[1] 358

sqrt(16)

[1] 4

Figure 2.1: Welcome to RStudio.
The semi-colon can be used to place multiple commands on one line. One frequent use of this is to save and print a value all in one go:

```
15.3 * 23.4 -> product; product  # save result and show it
```

[1] 358

### 2.1.3 Loading packages

R is divided up into packages. A few of these are loaded every time you run R, but most have to be selected. This way you only have as much of R as you need.

In the Packages tab, check the boxes next to the following packages to load them:

- mosaic (a package from Project MOSAIC, should autoload on the server)
- Lock5Data (a package for our text book)

### 2.2 Four Things to Know About R

Computers are great for doing complicated computations quickly, but you have to speak to them on their terms. Here are few things that will help you communicate with R.

1. R is case-sensitive
   
   If you mis-capitalize something in R it won’t do what you want.

2. Functions in R use the following syntax:
functionname(argument1, argument2, ...)

- The arguments are always surrounded by (round) parentheses and separated by commas.
  Some functions (like `data()`) have no required arguments, but you still need the parentheses.
- If you type a function name without the parentheses, you will see the code for that function – which probably isn’t what you want at this point.

3. TAB completion and arrows can improve typing speed and accuracy.
   If you begin a command and hit the TAB key, R will show you a list of possible ways to complete the command. If you hit TAB after the opening parenthesis of a function, it will show you the list of arguments it expects. The up and down arrows can be used to retrieve past commands.

4. If you get into some sort of mess typing (usually indicated by extra ’+’ signs along the left edge), you can hit the escape key to get back to a clean prompt.

2.3 Data in R

2.3.1 Data in Packages

Most often, data sets in R are stored in a structure called a data frame. There are a number of data sets built into R and many more that come in various add on packages. The `Lock5Data` package, for example, contains all the data sets from our text book. In the book, data set names are printed in bold text.

You can see a list of them using

```r
data(package = "Lock5Data")
```

You can find a longer list of all data sets available in any loaded package using

```r
data()
```

2.3.2 The HELPrct data set

The `HELPrct` data frame from the `mosaic` package contains data from the Health Evaluation and Linkage to Primary Care randomized clinical trial. You can find out more about the study and the data in this data frame by typing

```r
?HELPrct
```

Among other things, this will tell us something about the subjects in this study:

Eligible subjects were adults, who spoke Spanish or English, reported alcohol, heroin or cocaine as their first or second drug of choice, resided in proximity to the primary care clinic to which they would be referred or were homeless. Patients with established primary care relationships they planned to continue, significant dementia, specific plans to leave the Boston area that would prevent research participation, failure to provide contact information for tracking purposes, or pregnancy were excluded.

Subjects were interviewed at baseline during their detoxification stay and follow-up interviews were undertaken every 6 months for 2 years.
It is often handy to look at the first few rows of a data frame. It will show you the names of the variables and the kind of data in them:

```
head(HELPrc)
```

```
   age anysubstatus anysub cesd d1 daysanysub dayslink drugrisk e2b female sex g1b
1 37       1    yes  49     3      177     225       0   NA   0  male yes
2 37       1    yes  30    22       2       NA       0   NA   0  male yes
3 26       1    yes  39     0       3       365      20   NA   0  male no
4 39       1    yes  15     2      189     343       0    1   1  female no
5 32       1    yes  39    12       2       57       0    1   0  male no
6 47       1    yes  6    31      365       NA      0    1   1  female no
```

```
 homeles i1 i2 id indtot linkstatus link mcs pcs pss_fr racegrp satreat sexrisk
1 housed 13 26 1 39       1    yes 25.112 58.41       0 black no 4
2 homeless 56 62 2 43      NA <NA> 26.670 36.04       1 white no 7
3 housed 0  0  3 41       0    no  6.763 74.81      13 black no 2
4 housed 5  5  4 28       0    no 43.968 61.93      11 white yes 4
5 homeless 10 13 5 38      1    yes 21.676 37.35      10 black no 6
6 housed 4  4  6 29       0    no 55.509 46.48       5 black no 5
```

That's plenty of variables to get us started with exploration of data.

### 2.3.3 Using your own data

We will postpone for now a discussion about getting your own data into RStudio, but any data you can get into a reasonable format (like csv) can be imported into RStudio pretty easily.
2.4 The Most Important Template

Most of what we will do in this chapter makes use of a single R template:

\[
\text{goal}(x \sim y, \text{data} = \text{mydata})
\]

It is useful if we name the slots in this template:

\[
\text{goal}(y \sim x, \text{data} = \text{mydata})
\]

Actually, there are some variations on this template:

### Simpler version
\[
\text{goal}(-x, \text{data} = \text{mydata})
\]

### Fancier version:
\[
\text{goal}(y \sim x | z, \text{data} = \text{mydata})
\]

### Unified version:
\[
\text{goal}(\text{formula, data} = \text{mydata})
\]

To use the template, you just need to know what goes in each slot. This can be determined by asking yourself two questions:

1. What do you want R to do?
   - this determines what function to use (goal).
2. What must R know to do that?
   - this determines the inputs to the function
   - for describing data, must must identify which data frame and which variable(s).

2.5 Summaries of One Variable

A distribution is described by what values occur and with what frequency. That is, the distribution answers two questions:

- What values?
- How often?

Statisticians have devised a number of graphs to help us see distributions visually.

The general syntax for making a graph of one variable in a data frame is

\[
\text{plotname}(-\text{variable, data} = \text{dataName})
\]

In other words, there are three pieces of information we must provide to R in order to get the plot we want:

- The kind of plot (\text{histogram()}, \text{bargraph()}, \text{densityplot()}, \text{bwplot}(), etc.)
- The name of the variable
- The name of the data frame this variable is a part of.
2.5.1 Histograms (and density plots) for quantitative variables

Histograms are a way of displaying the distribution of a quantitative variable. Here are a couple examples:

```r
histogram(~BirthRate, data = AllCountries)
histogram(~age, data = HELPrct)
```

We can control the (approximate) number of bins using the `nint` argument, which may be abbreviated as `n`. The number of bins (and to a lesser extent the positions of the bins) can make a histogram look quite different.

```r
histogram(~age, data = HELPrct, n = 8)
histogram(~age, data = HELPrct, n = 15)
histogram(~age, data = HELPrct, n = 30)
```

We can also describe the bins in terms of center and width instead of in terms of the number of bins. This is especially nice for count or other integer data.

```r
histogram(~age, data = HELPrct, width = 10)
histogram(~age, data = HELPrct, width = 5)
histogram(~age, data = HELPrct, width = 2)
```
Sometimes a **frequency polygon** provides a more useful view. The only thing that changes is `histogram()` becomes `freqpolygon()`.

```r
freqpolygon(~age, data = HELPrct, width = 5)
```

What is a frequency polygon? The picture below shows how it is related to a histogram. The frequency polygon is just a dot-to-dot drawing through the centers of the tops of the bars of the histogram.

R also provides a “smooth” version called a density plot; just change the function name from `histogram()` to `densityplot()`.

```r
densityplot(~BirthRate, data = AllCountries)
densityplot(~age, data = HELPrct)
```
2.5.2 The shape of a distribution

If we make a histogram (or any of these other plots) of our data, we can describe the overall shape of the distribution. Keep in mind that the shape of a particular histogram may depend on the choice of bins. Choosing too many or too few bins can hide the true shape of the distribution. (When in doubt, make more than one histogram.)

Here are some words we use to describe shapes of distributions.

**symmetric** The left and right sides are mirror images of each other.

**skewed** The distribution stretches out farther in one direction than in the other. (We say the distribution is skewed toward the long tail.)

**uniform** The heights of all the bars are (roughly) the same. (So the data are equally likely to be anywhere within some range.)

**unimodal** There is one major “bump” where there is a lot of data.

**bimodal** There are two “bumps”.

**outlier** An observation that does not fit the overall pattern of the rest of the data.

We’ll learn about another graph used for quantitative variables (a boxplot, `bwplot()` in R) soon.

2.5.3 Numerical Summaries (i.e., statistics)

Recall that a statistics is a number computed from data. Numerical summaries are computed using the same template as graphical summaries. Here are some examples.

```r
mean(~age, data = HELPrct)
[1] 35.65
median(~age, data = HELPrct)
[1] 35
max(~age, data = HELPrct)
[1] 60
min(~age, data = HELPrct)
[1] 19
sd(~age, data = HELPrct) # standard deviation
[1] 7.71
```
26 Describing Data

```r
var(~age, data = HELPrct)  # variance
[1] 59.45

iqr(~age, data = HELPrct)  # inter-quartile range
[1] 10

favstats(~age, data = HELPrct)  # some favorite statistics

  min     Q1  median     Q3    max mean     sd    n missing
 19 30 35 40 60 35.65 7.71 453      0
```

2.5.4 Bar graphs for categorical variables

Bar graphs are a way of displaying the distribution of a categorical variable.

```r
bargraph(~substance, data = HELPrct)
bargraph(~substance, data = HELPrct, horizontal = TRUE)
```

![Bar graphs for categorical variables](chart)

Statisticians rarely use pie charts because they are harder to read.

2.6 Creating Tables

2.6.1 Tabulating a categorical variable

The `tally()` function tabulates categorical variables. This syntax is just like the syntax for plotting, we just replace `bargraph()` with `tally()`:

```r
tally(~substance, data = HELPrct)
```

<table>
<thead>
<tr>
<th>alcohol</th>
<th>cocaine</th>
<th>heroin</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>177</td>
<td>152</td>
<td>124</td>
<td>453</td>
</tr>
</tbody>
</table>

Last Modified: November 5, 2013
2.6.2 Tabulating a quantitative variable

Although `tally()` works with quantitative variables as well as categorical variables, this is only useful when there are not too many different values for the variable.

```
tally(~age, data = HELPct)
```

<p>| | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>13</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>20</td>
<td>28</td>
</tr>
<tr>
<td>34</td>
<td>35</td>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
<td>40</td>
<td>41</td>
<td>42</td>
<td>43</td>
<td>44</td>
<td>45</td>
</tr>
<tr>
<td>18</td>
<td>25</td>
<td>23</td>
<td>20</td>
<td>18</td>
<td>27</td>
<td>10</td>
<td>20</td>
<td>10</td>
<td>13</td>
<td>7</td>
<td>13</td>
</tr>
<tr>
<td>49</td>
<td>50</td>
<td>51</td>
<td>52</td>
<td>53</td>
<td>54</td>
<td>55</td>
<td>56</td>
<td>57</td>
<td>58</td>
<td>59</td>
<td>60</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>8</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>453</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Tabulating in bins (optional)

It is more convenient to group them into bins. We just have to tell R what the bins are. For example, suppose we wanted to group the 20s, 30s, 40s, etc. together.

```
binnedAge <- cut(HELPct$age, breaks = c(10, 20, 30, 40, 50, 60, 70))
tally(~binnedAge)  # no data frame given because it's not in a data frame
```

<table>
<thead>
<tr>
<th>(10,20]</th>
<th>(20,30]</th>
<th>(30,40]</th>
<th>(40,50]</th>
<th>(50,60]</th>
<th>(60,70]</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>113</td>
<td>224</td>
<td>97</td>
<td>16</td>
<td>0</td>
<td>453</td>
</tr>
</tbody>
</table>

That's not quite what we wanted: 30 is in with the 20s, for example. Here's how we fix that.

```
binnedAge <- cut(HELPct$age, breaks = c(10, 20, 30, 40, 50, 60, 70), right = FALSE)
tally(~binnedAge)  # no data frame given because it's not in a data frame
```

<table>
<thead>
<tr>
<th>[10,20)</th>
<th>[20,30)</th>
<th>[30,40)</th>
<th>[40,50)</th>
<th>[50,60)</th>
<th>[60,70)</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>97</td>
<td>232</td>
<td>105</td>
<td>17</td>
<td>1</td>
<td>453</td>
</tr>
</tbody>
</table>

We won't use this very often, since typically seeing this information in a histogram is more useful.

Labeling a histogram

The `histogram()` function offers you the option of adding the counts to the graph.

```
histogram(~age, data = HELPct, label = TRUE, type = "count", width = 10, center = 5, ylim = c(0, 300), right = FALSE)
```
2.7 Looking at multiple variables at once

2.7.1 Conditional plots

The formula for a `lattice` plot can be extended to create multiple panels (sometimes called `facets`) based on a “condition”, often given by another variable. The general syntax for this becomes

```
plotname(~variable | condition, data = dataName)
```

For example, we might like to see how the ages of men and women compare in the HELP study, or whether the distribution of weights of male mosquitoes is different from the distribution for females.

```
histogram(~age | sex, HELPrct, width = 5)
histogram(~BirthRate | Developed, data = AllCountries, width = 5)
```

We can do the same thing for bar graphs.

```
bargraph(~substance | sex, data = HELPrct)
```
### 2.7.2 Grouping

Another way to look at multiple groups simultaneously is by using the `groups` argument. What groups does depends a bit on the type of graph, but it will put the information in one panel rather than multiple panels. Using `groups` with `histogram()` doesn't work so well because it is difficult to overlay histograms. Density plots work better for this.

Here are some examples. We use `auto.key=TRUE` to build a simple legend so we can tell which groups are which.

```r
bargraph(~substance, groups = sex, data = HELPrct, auto.key = TRUE)
densityplot(~age, groups = sex, data = HELPrct, auto.key = TRUE)
```

We can even combine grouping and conditioning in the same plot.

```r
densityplot(~age | sex, groups = substance, data = HELPrct, auto.key = TRUE)
```

---

1. The `mosaic` function `histogram()` does do something meaningful with `groups` in some situations.
This plot shows that for each substance, the age distributions of men and women are quite similar, but that the distributions differ from substance to substance.

### 2.7.3 Scatterplots

The most common way to look at two quantitative variables is with a scatter plot. The `lattice` function for this is `xyplot()`, and the basic syntax is

```r
xyplot(yvar ~ xvar, data = dataName)
```

Notice that now we have something on both sides of the `~` since we need to tell R about two variables.

```r
xyplot(mcs ~ age, data = HELPrct)
```
Grouping and conditioning work just as before. With large data set, it can be helpful to make the dots semi-transparent so it is easier to see where there are overlaps. This is done with `alpha`. We can also make the dots smaller (or larger) using `cex`.

```r
xyplot(mcs ~ age | sex, groups = substance, data = HELPrct, alpha = 0.6, cex = 0.5, auto.key = TRUE)
```

### 2.7.4 Cross-tables: Tabulating two or more variables

`tally()` can also compute cross tables for two (or more) variables. Here are several ways to get a table of sex and substance in the `HELPrct` data.

```r
tally(~sex + substance, data = HELPrct)
```

<table>
<thead>
<tr>
<th></th>
<th>alcohol</th>
<th>cocaine</th>
<th>heroin</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>female</td>
<td>36</td>
<td>41</td>
<td>30</td>
<td>107</td>
</tr>
<tr>
<td>male</td>
<td>141</td>
<td>111</td>
<td>94</td>
<td>346</td>
</tr>
<tr>
<td>Total</td>
<td>177</td>
<td>152</td>
<td>124</td>
<td>453</td>
</tr>
</tbody>
</table>

```r
tally(sex ~ substance, data = HELPrct)
```
Describing Data

Notice that (by default) some of these use counts and some use proportions. If you don’t like the defaults, or if you don’t want the row and column totals (called marginal totals), we can change the defaults by adding a bit more instruction.

```
tally(~sex + substance, data = HELPrct)
```

```r
table

<table>
<thead>
<tr>
<th></th>
<th>substance</th>
</tr>
</thead>
<tbody>
<tr>
<td>sex</td>
<td>alcohol</td>
</tr>
<tr>
<td></td>
<td>female</td>
</tr>
<tr>
<td></td>
<td>male</td>
</tr>
<tr>
<td></td>
<td>Total</td>
</tr>
</tbody>
</table>
```

We can arrange the table differently by converting it to a data frame.

```
as.data.frame(tally(~sex + substance, data = HELPrct))
```

```
table

<table>
<thead>
<tr>
<th></th>
<th>substance</th>
</tr>
</thead>
<tbody>
<tr>
<td>sex</td>
<td>alcohol</td>
</tr>
<tr>
<td></td>
<td>female</td>
</tr>
<tr>
<td></td>
<td>male</td>
</tr>
<tr>
<td></td>
<td>Total</td>
</tr>
</tbody>
</table>
```

2.8 Exporting Plots

You can save plots to files or copy them to the clipboard using the Export menu in the Plots tab. It is quite simple to copy the plots to the clipboard and then paste them into a Word document, for example. You can even adjust the height and width of the plot first to get it the shape you want.
R code and output can be copied and pasted as well. It’s best to use a fixed width font (like Courier) for R code so that things align properly.

RStudio also provides a way (actually multiple ways) to create documents that include text, R code, R output, and graphics all in one document so you don’t have to do any copying and pasting. This is a much better workflow since it avoids copy-and-paste which is error prone and makes it easy to regenerate an entire report should the data change (because you get more of it or correct an error, for example).

2.9 Using R Markdown

Although you can export plots from RStudio for use in other applications, there is another way of preparing documents that has many advantages. RStudio provides several ways to create documents that include text, R code, R output, graphics, even mathematical notation all in one document. The simplest of these is R Markdown.

To create a new R Markdown document, go to “File”, “New”, then “R Markdown”:

![R Studio File Menu]

When you do this, a file editing pane will open with a template inserted. If you click on “Knit HTML”, RStudio will turn this into an HTML file and display it for you. Give it a try. You will be asked to name your file if you haven’t already done so. If you are using the RStudio server in a browser, then your file will live on the server (“in the cloud”) rather than on your computer.

If you look at the template file you will see that the file has two kinds of sections. Some of this file is just normal text (with some extra symbols to make things bold, add in headings, etc.) You can get a list of all of these mark up options by selecting the “Markdown Quick Reference” in the question mark menu.
The second type of section is an R code chunk. These are colored differently to make them easier to see. You can insert a new code chunk by selecting “Insert Chunk” from the “Chunks” menu:

(You can also type `r` to begin and `r` to end the code chunk if you would rather type.) You can put any R code in these code chunks and the results (text output or graphics) as well as the R code will be displayed in your HTML file.

There are options to do things like (a) run R code without displayng it, (b) run R code without displaying the output, (c) controling size of plots, etc., etc. But for starting out, this is really all you need to know.

**R Markdown files must be self-contained**

R Markdown files do not have access to things you have done in your console. (This is good, else your document would change based on things not in the file.) This means that you must explicitly load data, and require packages *in the R Markdown file* in order to use them. In this class, this means that most of your R Markdown files will have a chunk near the beginning that includes

```r
require(mosaic)  # load the mosaic package
require(Lock5Data)  # get data sets from the book
```

For your first assignment, I’ll provide you a template to get you going.

**2.9.1 Printing your document**

The preview window has an icon that looks like an arrow pointing at a window. If you click on that the document will open in a regular browser window. From there you can use your browser’s print features to print the document.

**2.10 A Few Bells and Whistles**

There are lots of arguments that control how these plots look. Here are just a few examples, some of which we have already seen.
2.10.1 auto.key

auto.key=TRUE turns on a simple legend for the grouping variable. (There are ways to have more control, if you need it.)

\[
xyplot(Sepal.Length \sim Sepal.Width, groups = \text{Species}, \text{data} = \text{iris}, \text{auto.key} = \text{TRUE})
\]

2.10.2 alpha, cex

Sometimes it is nice to have elements of a plot be partly transparent. When such elements overlap, they get darker, showing us where data are “piling up.” Setting the alpha argument to a value between 0 and 1 controls the degree of transparency: 1 is completely opaque, 0 is invisible. The cex argument controls “character expansion” and can be used to make the plotting “characters” larger or smaller by specifying the scaling ratio.

Here is another example using data on 150 iris plants of three species.

\[
xyplot(Sepal.Length \sim Sepal.Width, groups = \text{Species}, \text{data} = \text{iris}, \text{auto.key} = \text{list(columns = 3)}, \text{alpha} = 0.5, \text{cex} = 1.3)
\]
**xyplot**(`Sepal.Length ~ Sepal.Width, groups = Species, data = iris, main = "Some Iris Data", sub = "(R. A. Fisher analyzed this data in 1936)", xlab = "sepal width (cm)", ylab = "sepal length (cm)\)

alpha = 0.5, auto.key = list(columns = 3))

Some Iris Data

(R. A. Fisher analyzed this data in 1936)

**layout**

`layout` can be used to control the arrangement of panels in a multi-panel plot. The format is

\[
\text{layout} = \text{c}(\text{cols, rows})
\]

where `cols` is the number of columns and `rows` is the number of rows. (Columns first because that is the x-coordinate of the plot.)

`lty`, `lwd`, `pch`, `col`

These can be used to change the line type, line width, plot character, and color. To specify multiples (one for each group), use the `c()` function (see below).

**densityplot**(`~age, data = HELPrct, groups = sex, lty = 1, lwd = c(2, 4))

**histogram**(`~age, data = HELPrct, col = "green"`)
# There are 25 numbered plot symbols; `pch`=plot character

```r
xyplot( mcs ~ age, data=HELPct, groups=sex,
       pch=c(1,2), col=c('brown', 'darkgreen'), cex=.75 )
```

Note: If you change this this way, they will not match what is generated in the legend using `auto.key=TRUE`. So it can be better to set these things in a different way if you are using `groups`. See below.

You can a list of the hundreds of available color names using

```r
colors()
```

### 2.10.3 `trellis.par.set()`

Default settings for lattice graphics are set using `trellis.par.set()`. Don’t like the default font sizes? You can change to a 7 point (base) font using

```r
trellis.par.set(fontsize = list(text = 7))  # base size for text is 7 point
```

Nearly every feature of a lattice plot can be controlled: fonts, colors, symbols, line thicknesses, colors, etc. Rather than describe them all here, we’ll mention only that groups of these settings can be collected into a theme. `show.settings()` will show you what the theme looks like.

```r
trellis.par.set(theme = col.whitebg())  # a theme in the lattice package
show.settings()
```
trellis.par.set(theme = col.mosaic())  # a theme in the mosaic package
show.settings()

trellis.par.set(theme = col.mosaic(bw = TRUE))  # black and white version
show.settings()
trellis.par.set(theme = col.mosaic())  # back to the mosaic theme

trellis.par.set(fontsize = list(text = 9))  # and back to a 9 point font

Want to save your settings?

# save current settings
mySettings <- trellis.par.get()
# switch to mosaic defaults
trellis.par.set(theme = col.mosaic())
# switch back to my saved settings
trellis.par.set(mySettings)

2.11 Graphical Summaries – Important Ideas

2.11.1 Patterns and Deviations from Patterns

The goal of a statistical plot is to help us see

- potential patterns in the data, and
- deviations from those patterns.

2.11.2 Different Plots for Different Kinds of Variables

Graphical summaries can help us see the distribution of a variable or the relationships between two (or more) variables. The type of plot used will depend on the kinds of variables involved. There is a nice summary of these on page 48. You can use demo() to see how to get R to make the plots in this section.

Later, when we do statistical analysis, we will see that the analysis we use will also depend on the kinds of variables involved, so this is an important idea.

2.11.3 Side-by-side Plots and Overlays Can Reveal Importance of Additional Factors

The lattice graphics plots make it particularly easy to generate plots that divide the data into groups and either produce a panel for each group (using |) or display each group in a different way (different colors or symbols, using the groups argument). These plots can reveal the possible influence of additional variables – sometimes called covariates.

2.11.4 Area = (relative) frequency

Many plots are based on the key idea that our eyes are good at comparing areas. Plots that use area (e.g., histograms, mosaic plots, bar charts, pie charts) should always obey this principle

\[ \text{Area} = \text{(relative) frequency} \]

Plots that violate this principle can be deceptive and distort the true nature of the data.
2.12 Getting Help in RStudio

2.12.1 The RStudio help system

There are several ways to get RStudio to help you when you forget something. Most objects in packages have help files that you can access by typing something like:

```r
?bargraph
?histogram
?HELPrc
```

You can search the help system using

```r
help.search("Grand Rapids") # Does R know anything about Grand Rapids?
```

This can be useful if you don’t know the name of the function or data set you are looking for.

2.12.2 Tab completion

As you type the name of a function in RStudio, you can hit the tab key and it will show you a list of all the ways you could complete that name, and after you type the opening parenthesis, if you hit the tab key, you will get a list of all the arguments and (sometimes) some helpful hints about what they are.

2.12.3 History

If you know you have done something before, but can’t remember how, you can search your history. The history tab shows a list of recently executed commands. There is also a search bar to help you find things from longer ago.

2.12.4 Error messages

When things go wrong, R tries to help you out by providing an error message. If you can’t make sense of the message, you can try copying and pasting your command and the error message and sending to me in an email. One common error message is illustrated below.

```r
fred <- 23
frd
```

```
Error: object ‘frd’ not found
```

The object `frd` is not found because it was mistyped. It should have been `fred`. If you see an “object not found” message, check your typing and check to make sure that the necessary packages have been loaded.
2.13 From Excel or Google to R

So far we have been using data that lives in R packages. This has allowed us to focus on things like how to make plots and create numerical summaries without worrying too much about the data themselves. But if you are going to do any of your own statistical analyses, then you will need to import your own data into R and have some tools for manipulating the data once it is there.

Excel or Google spreadsheets are reasonable tools for entering (small) data sets by hand and doing basic data tidying (organizing) and cleaning (correcting errors). This section describes how to get data from a spreadsheet into R.

2.13.1 While you are still in the spreadsheet

If you are creating your own data in a spreadsheet with the intent of bringing into R (or some other statistical package) for analysis, it is important that you design your spreadsheet appropriately. For most data sets this will mean

1. The first row should contain variables names.
   These should be names that will work well in R. This usually means they will be relatively short and avoid spaces and punctuation.

2. Each additional row corresponds to a case/observational unit.

3. Each column corresponds to a variable

4. There is nothing else in the spreadsheet.
   Do not include notes to yourself, plots, numerical summaries, etc. These things can be kept in a separate worksheet, another file, your lab notebook, just not in the worksheet you are going to export.

2.13.2 Exporting to csv

The comma separated values (csv) format has become a standard way of transferring data between programs. Both Google and Excel can export to this format, and R can import from this format. Once your data are ready to go, export them to csv. Give the file a good name, and remember where you have put it.

2.13.3 Uploading the data (RStudio server only)

To get the data from your computer onto the server, you need to upload the data. (You can skip this step if you are working with a local copy of RStudio.) Uploading transfers a copy of your data from your computer onto the server (the “cloud”). This is like uploading pictures to Facebook so you can later use them in posts or as a cover photo or tag your friends or whatever else once the photo is on Facebook.

To upload the data, go to the Files tab and click on Upload:
A window will pop up prompting you to browse to the file's location on your computer. Choose the file and it will upload to the server. You should see it appear in your file menu.

### 2.13.4 Importing the data into R

Now that the file is on the server, you can import it into R. This takes place in the **Environment** tab. Once there, choose **Import Dataset** and then **From Text File**.

The instructions are pretty clear from there, but here are some things to watch for:

- The default name for the data set is taken from the file name. If you used a very long file name, you will probably want to shorten this down. (But don’t call it Data or something too generic either.) If the data are from the asters you have been tagging, perhaps call it Asters. If you are working with multiple data sets that deal with asters, add a bit more detail, perhaps Asters01 or some such thing.

- Be sure to select to use your first line as variable names (Heading = Yes).
The data set should now be ready for use in R.

2.13.5 A shortcut for Google Spreadsheets

You can avoid all the uploading step if you use a Google spreadsheet and import directly from Google. To do this, you must first **publish** your Google spreadsheet, and then copy the csv URL from Google. Here's how.

1. In the file menu select **Publish**
2. In the publish menu, select **Start Publishing**

3. Now choose the **CSV** file format.

4. Once you have done that, you can copy the URL:

5. In R, use the `fetchGoogle()` function to load the data into R:

   ```r
   # The URL will be really long and ugly...
   Asters3 <- fetchGoogle("https://docs.google.com/spreadsheet/pub?key=0ApQwsmr3d8V2cmprN01YNnNqMEkxbH
   
   Don't forget the quotation marks! (They won't be part of what you copy from Google.)"
2.13.6 Using R commands to read a data file

Even if you primarily use the RStudio interface to import data, it is good to know about the command line methods since these are required to import data into scripts, RMarkdown, and Rnw files. CSV files (and a few other types of files as well) can be read with

```r
someData <- read.file("file.csv")
```

This can be used to read data directly from a URL as well. For example, here is some data from the US Census Bureau:

```r
dim(Population)
```

```r
[1] 57 46
```

```r
head(Population, 4)
```

<table>
<thead>
<tr>
<th>Sumlev</th>
<th>Region</th>
<th>Division</th>
<th>State</th>
<th>Name</th>
<th>CENSUS2010POP</th>
<th>ESTIMATESBASE2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>United States</td>
<td>308745538</td>
<td>308745708</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>1</td>
<td>0</td>
<td>Northeast Region</td>
<td>55317240</td>
<td>55317245</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>2</td>
<td>0</td>
<td>Midwest Region</td>
<td>66927001</td>
<td>66927489</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>3</td>
<td>0</td>
<td>South Region</td>
<td>114555744</td>
<td>114557147</td>
</tr>
</tbody>
</table>

```r
```

```r
1 30932625       311587816 313914040 578717 2261591 2326224
2 55376926       55597646 55761091 59681 220720 163445
3 66972135       67145089 67316297 44646 172954 171208
4 114853800      116022230 117257221 296653 1168430 1234991

```r
```

```r
1 987836       3977039 3953593 598716 2490976 2513173 389120
2 160353       644224 631961 108678 465661 466970 51675
3 210660       839312 825776 141184 580842 580718 69476
4 373379       1503557 1510568 235600 956673 970844 137779

```r
INTERNATIONALINC2011 INTERNATIONALINC2012 INTERNATIONALMIG2010 INTERNATIONALMIG2011
```

```r
1 1486063       1440420          189597          775528
2 178563        164991           48282           211281
3 258470        245058           24794           100624
4 546884        539724           71591           282102

```r
INTERNATIONALMIG2012 DOMESTICMIG2010 DOMESTICMIG2011 DOMESTICMIG2012 NETMIG2010
```

```r
1 885804        0             0              0        189597
2 221546       -38396         -161531        -220968     9886
3 11790       -49082         -184696        -185118    -24288
4 337769       86302          325546         353879    157893

```r
```

```r
1 775528       885804          0             0          0       12.81      12.64
2 49750        578             -1880         -7593       -2124     11.61      11.35
3 -84072      -73328           -542          -1444       -522     12.52      12.28
4 607648       696483          981           13898       3619      13.02      12.95

```r
```

```r
1 8.024       8.036            4.787          4.606        2.498
2 8.392       8.387            3.218          2.963        3.808
3 8.662       8.638            3.854          3.645        1.501
4 8.287       8.323            4.737          4.627        2.444
```
Many web sites provide data in csv format. Here some examples:

- [http://www.census.gov/](http://www.census.gov/) (Census Bureau data)
- [http://www.ncdc.noaa.gov/data-access](http://www.ncdc.noaa.gov/data-access) (NOAA Weather and climate data)
- [http://www.gapminder.org/data/](http://www.gapminder.org/data/) (Gapminder data)
- [http://introcs.cs.princeton.edu/java/data/](http://introcs.cs.princeton.edu/java/data/) has a number of data sets, some in csv format, collected from other places on the internet.
- [http://www.exploredata.net/Downloads](http://www.exploredata.net/Downloads) has data from WHO, a genome expression study, and a microbiome study.

But be aware that some of these files might need to be cleaned up a bit before they are usable for statistics. Also, some internet files are very large and may take a while to download. Many sites will give an indication of the size of the data set so you know what you are in for. The better sites will include links to a codebook (a description of all the variables, units used, how and when the data were collected, and any other information relevant to interpreting the data). Such a document is available for the population data loaded above. You can find it at [http://www.census.gov/popest/data/national/totals/2012/files/NST-EST2012-alldata.pdf](http://www.census.gov/popest/data/national/totals/2012/files/NST-EST2012-alldata.pdf)

There are similar functions for reading various other sorts of data. There is even a `read.xls()` function in the `gdata` package that can read directly from Excel spreadsheets without having to first export them to csv format. There are also utilities for converting to and from native data formats of other statistical programs (like SAS, SPSS, etc.). But since these typically all know how to read and write csv files, learning a workflow that goes through CSV is a broadly applicable skill.

### 2.13.7 Missing Data

The `na.strings` argument can be used to specify codes for missing values. The following can be useful, for example:

```r
someData <- read.file('file.csv',
na.strings=c('NA','','.','-','na'))
```

because SAS uses a period (.) to code missing data, and some csv exporters use ‘-‘. By default R reads these as string data, which forces the entire variable to be of character type instead of numeric.

By default, R will recode character data as a factor. If you prefer to leave such variables in character format, you can use

```r
somData <- read.file('file.csv',
na.strings=c('NA','','.','-','na'),
stringsAsFactors=FALSE)
```

---

2 Even finer control can be obtained by manually setting the class (type) used for each column in the file. In addition, this speeds up
2.14 Manipulating your data

2.14.1 Subsets

You may want to work with only a subset of your data. This can be done with the subset command. Here are some examples.

1. Select only the males from the HELPrct data set.

```r
tally(~sex, data = HELPrct)
```

<table>
<thead>
<tr>
<th>female</th>
<th>male</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>107</td>
<td>346</td>
<td>453</td>
</tr>
</tbody>
</table>

```r
HELPmales <- subset(HELPrct, sex == "male")  # notice the double =
dim(HELPmales)
```

```
[1] 346 27
```

2. Select only the subjects over 50:

```r
HELPold <- subset(HELPrct, age > 50)
```

3. Only the states (and Puerto Rico and District of Columbia) data from the US Census data set loaded above.

```r
tally(~Sumlev, data = Population)
```

<table>
<thead>
<tr>
<th>40</th>
<th>10</th>
<th>20</th>
<th>40 Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>52</td>
<td>57</td>
</tr>
</tbody>
</table>

```r
States <- subset(Population, Sumlev == 40)
dim(States)
```

```
[1] 52 46
```

The `subset()` function can use any condition that evaluates to TRUE or FALSE for each row (case) in the data set.

2.14.2 Creating new variables

We can add a new variable to data set using `transform()`:

```r
someData <- read.file('file.csv', na.strings=c('NA','','.'),'-','na'), colClasses=c('integer','character'))
```
Describing Data

head(iris,3)

<table>
<thead>
<tr>
<th>Sepal.Length</th>
<th>Sepal.Width</th>
<th>Petal.Length</th>
<th>Petal.Width</th>
<th>Species</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>3.5</td>
<td>1.4</td>
<td>0.2</td>
<td>setosa</td>
</tr>
<tr>
<td>4.9</td>
<td>3.0</td>
<td>1.4</td>
<td>0.2</td>
<td>setosa</td>
</tr>
<tr>
<td>4.7</td>
<td>3.2</td>
<td>1.3</td>
<td>0.2</td>
<td>setosa</td>
</tr>
</tbody>
</table>

iris2 <- transform(iris,
  Sepal.Ratio = Sepal.Length / Sepal.Width,
  Petal.Ratio = Petal.Length / Petal.Width)

head(iris2,3)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>3.5</td>
<td>1.4</td>
<td>0.2</td>
<td>setosa</td>
<td>1.457</td>
<td>7.0</td>
</tr>
<tr>
<td>4.9</td>
<td>3.0</td>
<td>1.4</td>
<td>0.2</td>
<td>setosa</td>
<td>1.633</td>
<td>7.0</td>
</tr>
<tr>
<td>4.7</td>
<td>3.2</td>
<td>1.3</td>
<td>0.2</td>
<td>setosa</td>
<td>1.469</td>
<td>6.5</td>
</tr>
</tbody>
</table>

States <- transform(States,
  Pop.Increase = 100 * (POPESTIMATE2012 - POPESTIMATE2010)/POPESTIMATE2010)

histogram(~ Pop.Increase, data=States, width=0.5,
  main="% Population increase (2010 to 2012)"
)

% Population increase (2010 to 2012)

Generally, it is a good idea to keep raw data (like Sepal.Length and Sepal.Width in your data file, but let R do the computation of derived variables for you. Among other advantages, if you ever fix an error in a Sepal.Length measurement, you don’t have to worry about remembering to also recompute the ratio. Furthermore, your R code documents how the derived value was computed.

2.14.3 Dropping Variables

Less frequently, you might want to remove a variable from a data frame. We can use subset() for this as well:

iris3 <- subset(iris2, select = -Sepal.Ratio)  # the minus sign means drop
head(iris3, 3)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>3.5</td>
<td>1.4</td>
<td>0.2</td>
<td>setosa</td>
<td>7.0</td>
</tr>
</tbody>
</table>
2.15 Saving Data

`write.csv()` can be used to save data from R into csv formatted files. This can be useful for exporting to some other program.

```r
write.csv(iris3, "iris3.csv")
```

Data can also be saved in native R format. Saving data sets (and other R objects) using `save()` has some advantages over other file formats:

- Complete information about the objects is saved, including attributes.
- Data saved this way takes less space and loads much more quickly.
- Multiple objects can be saved to and loaded from a single file.

The downside is that these files are only readable in R.

```r
save(iris3, file = "iris3.rda")  # the traditional file extension is rda for R native data.
load("iris3.rda")  # loads previously saved data
```

For more on importing and exporting data, especially from other formats, see the R Data Import/Export manual available on [CRAN](https://cran.r-project.org/doc/manuals/R-data.html).

### 2.15.1 Merging datasets

The `fusion1` data frame in the `fastR` package contains genotype information for a SNP (single nucleotide polymorphism) in the gene *TCF7L2*. The `pheno` data frame contains phenotypes (including type 2 diabetes case/control status) for an intersecting set of individuals. We can merge these together to explore the association between genotypes and phenotypes using `merge()`.

```r
require(fastR)
head(fusion1, 3)
```

<table>
<thead>
<tr>
<th>id</th>
<th>marker</th>
<th>markerID</th>
<th>allele1</th>
<th>allele2</th>
<th>genotype</th>
<th>Adose</th>
<th>Cdose</th>
<th>Gdose</th>
<th>Tdose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9735</td>
<td>RS12255372</td>
<td>1</td>
<td>3</td>
<td>GG</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>10158</td>
<td>RS12255372</td>
<td>1</td>
<td>3</td>
<td>GG</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>9380</td>
<td>RS12255372</td>
<td>1</td>
<td>3</td>
<td>GT</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

```r
head(pheno, 3)
```

<table>
<thead>
<tr>
<th>id</th>
<th>t2d</th>
<th>bmi</th>
<th>sex</th>
<th>age</th>
<th>smoker</th>
<th>chol</th>
<th>waist</th>
<th>weight</th>
<th>height</th>
<th>whr</th>
<th>sbp</th>
<th>dbp</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1002</td>
<td>case</td>
<td>32.86</td>
<td>F</td>
<td>70.76</td>
<td>4.57</td>
<td>112.0</td>
<td>85.6</td>
<td>161.4</td>
<td>0.9868</td>
<td>135</td>
<td>77</td>
</tr>
<tr>
<td>2</td>
<td>1009</td>
<td>case</td>
<td>27.39</td>
<td>F</td>
<td>53.92</td>
<td>7.32</td>
<td>93.5</td>
<td>77.4</td>
<td>168.1</td>
<td>0.9397</td>
<td>158</td>
<td>88</td>
</tr>
<tr>
<td>3</td>
<td>1012</td>
<td>control</td>
<td>30.47</td>
<td>M</td>
<td>53.86</td>
<td>5.02</td>
<td>104.0</td>
<td>94.6</td>
<td>176.2</td>
<td>0.9327</td>
<td>143</td>
<td>89</td>
</tr>
</tbody>
</table>
Describing Data

```r
# merge fusion1 and pheno keeping only id's that are in both
fusion1m <- merge(fusion1, pheno, by.x = "id", by.y = "id", all.x = FALSE, all.y = FALSE)
head(fusion1m, 3)
```

<table>
<thead>
<tr>
<th>id</th>
<th>marker</th>
<th>markerID</th>
<th>allele1</th>
<th>allele2</th>
<th>genotype</th>
<th>Adose</th>
<th>Cdose</th>
<th>Gdose</th>
<th>Tdose</th>
<th>t2d</th>
<th>bmi</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1002</td>
<td>RS12255372</td>
<td>1</td>
<td>3</td>
<td>GG</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>case</td>
<td>32.86</td>
</tr>
<tr>
<td>2</td>
<td>1009</td>
<td>RS12255372</td>
<td>1</td>
<td>3</td>
<td>GG</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>case</td>
<td>27.39</td>
</tr>
<tr>
<td>3</td>
<td>1012</td>
<td>RS12255372</td>
<td>1</td>
<td>3</td>
<td>GG</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>control</td>
<td>30.47</td>
</tr>
</tbody>
</table>

In this case, since the values are the same for each data frame, we could collapse `by.x` and `by.y` to `by` and collapse `all.x` and `all.y` to `all`. The first of these specifies which column(s) to use to identify matching cases. The second indicates whether cases in one data frame that do not appear in the other should be kept (`TRUE`) or dropped (filling in `NA` as needed) or dropped from the merged data frame.

Now we are ready to begin our analysis.

```r
tally(~t2d + genotype, fusion1m)
```

<table>
<thead>
<tr>
<th>genotype</th>
<th>t2d</th>
<th>GG</th>
<th>GT</th>
<th>TT</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>case</td>
<td>737</td>
<td>375</td>
<td>48</td>
<td>1160</td>
<td></td>
</tr>
<tr>
<td>control</td>
<td>835</td>
<td>309</td>
<td>27</td>
<td>1171</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>1572</td>
<td>684</td>
<td>75</td>
<td>2331</td>
<td></td>
</tr>
</tbody>
</table>
Exercises

For problems 1–7, include both the plots and the code you used to make them as well as any required discussion. Once you get the plots figured out, feel free to use some of the bells and whistles to make the plots even better.

1. Use R’s help system to find out what the \texttt{i1} and \texttt{i2} variables are in the \texttt{HELPct} data frame. Make histograms for each variable and comment on what you find out. How would you describe the shape of these distributions? Do you see any outliers (observations that don’t seem to fit the pattern of the rest of the data)?

2. Compare the distributions of \texttt{i1} and \texttt{i2} among men and women.

3. Compare the distributions of \texttt{i1} and \texttt{i2} among the three \texttt{substance} groups.

4. Where do the data in the \texttt{CPS85} data frame (in the \texttt{mosaic} package) come from? What are the observational units? How many are there?

5. Choose a quantitative variable that interests you in the \texttt{CPS85} data set. Make an appropriate plot and comment on what you see.

6. Choose a categorical variable that interests you in the \texttt{CPS85} data set. Make an appropriate plot and comment on what you see.

7. Create a plot that displays two or more variables from the \texttt{CPS85} data. At least one should be quantitative and at least one should be categorical. Comment on what you can learn from your plot.

8. The \texttt{fusion2} data set in the \texttt{fastR} package contains genotypes for another SNP. Merge \texttt{fusion1}, \texttt{fusion2}, and \texttt{pheno} into a single data frame.

Note that \texttt{fusion1} and \texttt{fusion2} have the same columns.

\begin{verbatim}
head(fusion1, 2)

   id    marker markerID allele1 allele2 genotype Adose Cdose Gdose Tdose
  1  9735  RS12255372      1      3      3     GG   0   0   2   0
  2 10158  RS12255372      1      3      3     GG   0   0   2   0

head(fusion2, 2)

   id   marker markerID allele1 allele2 genotype Adose Cdose Gdose Tdose
  1  9735  RS7903146      2      2      2     CC   0   2   0   0
  2 10158  RS7903146      2      2      2     CC   0   2   0   0
\end{verbatim}
You may want to use the `suffixes` argument to `merge()` or rename the variables after you are done merging to make the resulting data frame easier to navigate.

Tidy up your data frame by dropping any columns that are redundant or that you just don’t want to have in your final data frame.
3.1 Sampling Distributions

The key idea in this chapter is the notion of a sampling distribution. Do not confuse it with the population (what we would like to know about) or the sample (what we actually have data about). If we could repeatedly sample from a population, and if we computed a statistic from each sample, the distribution of those statistics would be the sampling distribution. Sampling distributions tell us how things vary from sample to sample and are the key to interpreting data.

The standard deviation of a sampling distribution is called the standard error, denoted $SE$.

The standard error is our primary way of measuring how much variability there is from sample statistic to sample statistic, and therefore how precise our estimates are.

3.2 Interval Estimates

An interval estimate gives a range of plausible values for a population parameter.

This is better than a single number (also called a point estimate) because it gives some indication of the precision of the estimate.

3.2.1 Margin of Error

One way to express an interval estimate is with a point estimate and a margin of error.
We can convert margin of error into an interval by adding and subtracting the margin of error to/from the statistic. So for the Gallup Poll, the interval would be

\[ 0.50 \pm 0.02 \text{ which is the same as } (0.48, 0.52) \]

So Gallup is claiming that any election result in which Obama received between 48% and 52% of the vote would be plausible.

The actual election results? 50.4% voted for Obama. Notice that 0.504 is inside the confidence interval \(0.50 \pm 0.02\).

A confidence interval for a parameter is an interval computed from sample data by a method that will capture the parameter for a specified proportion of all samples.

1. The probability of correctly containing the parameter is called the coverage rate or confidence level.
2. So 95% of 95% confidence intervals contain the parameter being estimated.
3. The margins of error in the tables above were designed to produce 95% confidence intervals.

### 3.2.2 Using the standard error to compute a margin of error

In many (but not all) situations, the sampling distribution is

- unimodal,
- symmetric, and
- bell-shaped

(The technical phrase is “approximately normal”.) In these situations, a 95% confidence interval can be estimated with

\[ \text{statistic} \pm 2SE \]

**Example.** A large sample of American adults the mean body mass index was 27.655. The standard error was 0.009. The 95% confidence interval is therefore

\[ 27.655 \pm 0.018 \]
The important question remains: If we don’t have access to the entire population, and we only get to look at one sample, how do we tell the shape of the sampling distribution and estimate its standard error? Stay tuned.

3.3 The Bootstrap

Here’s the clever idea: We don’t have the population, but we have a sample. Probably the sample it similar to the population in many ways. So let’s sample from our sample. We’ll call it resampling (also called bootstrapping). We want samples the same size as our original sample, so we will need to sample with replacement. This means that we may pick some members of the population more than once and others not at all. We’ll do this many times, however, so each member of our sample will get its fair share.

In class we did two examples by physical manipulation.

• We estimated the proportion of blue milk jug lids by drawing milk jug lids out of paper sacks (sample size was \( n = 10 \))
• We estimated the mean commute time in Atlanta by placing commute times on cards and shuffling them (again the sample size was \( n = 10 \))

Let’s see how this works in an example with a larger data set. It would be really tedious to do that kind mixing and shuffling physically, but the computer can easily do all of the resampling (and also the computation of statistics) for us.

3.3.1 Method 1: statistic \( \pm 2 \) SE

The example below uses data from 500 Atlanta commuters.

```
x.bar <- mean(~Time, data = CommuteAtlanta)
x.bar  # mean commute time in original sample
[1] 29.11

mean(~Time, data = resample(CommuteAtlanta))  # mean commute time in one resample
[1] 29.23

mean(~Time, data = resample(CommuteAtlanta))  # mean commute time in another resample
[1] 29.11

# Now we'll do it 1000 times
Bootstrap <- do(1000) * mean(~Time, data = resample(CommuteAtlanta))

We should check that that our bootstrap distribution has an appropriate shape:

histogram(~result, data = Bootstrap, n = 30)
dotPlot(~result, data = Bootstrap, n = 50)
```
Since the shape of the bootstrap distribution looks good, we can estimate the standard error and use it to compute a 95% confidence interval.

```r
# Estimate the standard error
SE <- sd(result, data=Bootstrap); SE  # standard error
[1] 0.9043

MoE <- 2 * SE; MoE  # margin of error for 95% CI
[1] 1.809

x.bar - MoE  # lower limit of 95% CI
[1] 27.3

x.bar + MoE  # upper limit of 95% CI
[1] 30.92
```

The same steps used in this example, get used in a wide variety of confidence interval situations.

1. Compute the statistic from the original sample.
2. Create a bootstrap distribution by resampling from the sample.
   (a) same size samples as the original sample
   (b) with replacement
   (c) compute the statistic for each sample

   The distribution of these statistics is the bootstrap distribution
3. Estimate the standard error SE by computing the standard deviation of the bootstrap distribution.
4. 95% CI is

\[
\text{statistic} \pm 2SE
\]
3.3.2 Another way to compute a confidence interval

Another way to create a 95% confidence interval is to use the middle 95% of the bootstrap distribution. The \texttt{cdata()} function can compute this for us as follows:

\begin{verbatim}
cdata(0.95, result, data = Bootstrap)
\end{verbatim}

\begin{center}
\begin{tabular}{ll}
\textbf{low} & \textbf{hi} \\
95\% & 27.26 & 30.83
\end{tabular}
\end{center}

This is not exactly the same as the interval above, but it is pretty close.

One advantage of this method is that it is easy to change the confidence level. To make a 90% confidence interval, we use the middle 90% of the bootstrap distribution instead.

\begin{verbatim}
cdata(0.9, result, data = Bootstrap)
\end{verbatim}

\begin{center}
\begin{tabular}{ll}
\textbf{low} & \textbf{hi} \\
90\% & 27.59 & 30.53
\end{tabular}
\end{center}

Notice that this interval is narrower. This will always be the case. Higher levels of confidence lead to wider confidence intervals.

(Method 1 can also be adjusted for other confidence levels as well – the number 2 needs to be replaced by an appropriate alternative.)

3.4 More Examples

This section contains the R code for some additional examples (with minimal discussion)

3.4.1 Comparing two proportions

\begin{verbatim}
tally( play ~ playVer, data=littleSurvey )
\end{verbatim}

\begin{verbatim}
prop( play ~ playVer, data=littleSurvey )
\end{verbatim}

\begin{verbatim}
stat <- diff( prop( play ~ playVer, data=littleSurvey ) ); stat
\end{verbatim}
```r
no.v2
0.2387

Boot.Play <- do(1000) * diff(prop(play ~ playVer, data=resample(littleSurvey)))
head(Boot.Play)

no.v2
1 0.2130
2 0.3425
3 0.2419
4 0.2416
5 0.3290
6 0.2324

histogram(~ no.v2, data=Boot.Play)

SE <- sd(~ no.v2, data = Boot.Play)
SE

[1] 0.05836

stat - 2 * SE

no.v2
0.1219

stat + 2 * SE

no.v2
0.3554

cdata(0.95, no.v2, data = Boot.Play)

  low  hi
95% 0.1248 0.3538
```

Last Modified: November 5, 2013

Math 145: Fall 2013: Pruim
Again, the two confidence intervals are similar.

Notice importantly that 0 is not in these intervals. That means all the plausible values for the difference between the proportions have the version 2 respondents more likely to \textit{not} go to the play.

### 3.4.2 Comparing two means

We can compare two means in a similar way. Let’s estimate the difference between mean pulse for men and women (based on a sample of students in an introductory statistics course at another institution).

```
mean( Pulse ~ Sex, data=StudentSurvey )
```

<table>
<thead>
<tr>
<th>Female</th>
<th>Male</th>
</tr>
</thead>
<tbody>
<tr>
<td>71.64</td>
<td>67.76</td>
</tr>
</tbody>
</table>

```
stat <- diff(mean( Pulse ~ Sex, data=StudentSurvey )); stat

Male
-3.883
```

```
Boot.Pulse <- do(1000) * diff(mean( Pulse ~ Sex, data=resample(StudentSurvey) ))
head(Boot.Pulse, 3)
```

<table>
<thead>
<tr>
<th>Male</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

```
head(Boot.Pulse, 3)
```

```
stat - 2 * SE
```

```
SE <- sd(~Male, data = Boot.Pulse)
SE

[1] 1.242
```

```
stat - 2 * SE
```
3.4.3 One proportion – from data

What percent of students smoke?

```r
stat <- prop(~Smoke, data = StudentSurvey)
stat

No
0.8812

Boot.Smoke <- do(1000) * prop(~Smoke, data = resample(StudentSurvey))
head(Boot.Smoke, 3)

No
1 0.8646
2 0.8785
3 0.8619

histogram(~No, data = Boot.Smoke)
```

![Histogram of No proportions](image)
Confidence Intervals

\[ \text{SE} \leftarrow \text{sd}(-\text{No, data = Boot.Smoke}) \]
\[ \text{stat} - \text{SE} \]

\begin{align*}
\text{No} & \quad 0.8642 \\
\text{stat} + \text{SE} & \quad 0.8982 \\
\end{align*}

\[ \text{cdata}(0.95, \text{No, data = Boot.Smoke}) \]

\begin{align*}
\text{low} & \quad 0.8481 \\
\text{hi} & \quad 0.9144 \\
\end{align*}

3.4.4 One proportion – from a data summary

Here is a report from a recent USA Today poll:

Let's compute a 95% confidence interval for the proportion of people disapprove of the way Obama is handling health care policy. This time we don't have data, but we can still simulate the bootstrap distribution – this time we will use \text{rflip}().

\[ \text{stat} \leftarrow 0.53 \]
\[ \text{Boot.Poll} \leftarrow \text{do}(1000) * \text{rflip}(1503, 0.53) \]
\[ \text{head}(\text{Boot.Poll}, 3) \]

\begin{tabular}{lll}
\hline
n & heads & tails & prop \\
\hline
1 & 1503 & 817 & 686 & 0.5436 \\
\hline
\end{tabular}
2 1503 795 708 0.5289
3 1503 803 700 0.5343

\texttt{histogram(~prop, data = Boot.Poll)}

\begin{verbatim}
SE <- sd(~prop, data = Boot.Poll)
SE
[1] 0.01308
2 * SE  # margin of error
[1] 0.02615
stat - 2 * SE  # lower end of interval
[1] 0.5038
stat + 2 * SE  # upper end of interval
[1] 0.5562
cdata(0.95, prop, data = Boot.Poll)  # quantile method
  low  hi
95% 0.505 0.5542
\end{verbatim}

These results are consistent with USA Today’s claim that the margin of error is ±3%.

\subsection*{3.4.5 A non-example: Estimating the median from a small sample}

\texttt{median(~Price, data = MustangPrice)}

\begin{verbatim}
[1] 11.9
\end{verbatim}
Boot.Mustang <- do(1000) * median(~Price, data = resample(MustangPrice))
head(Boot.Mustang, 3)

result
1 16.0
2 22.0
3 12.9

histogram(~result, data = Boot.Mustang, n = 50)

This time the histogram does not have the desired shape. There are two problems:

1. The distribution is not symmetric. (It is right skewed.)
2. The distribution has spikes and gaps.
   Since the median must be an element of the sample when the sample size is 25, there are only 25 possible values for the median (and some of these are very unlikely.

Since the bootstrap distribution does not look like a normal distribution (bell-shaped, symmetric), we cannot safely use our methods for creating a confidence interval.
4.1 Introduction to Hypothesis Testing

4.1.1 The 4-step outline

The following 4-step outline is a useful way to organize the ideas of hypothesis testing.

1. State the Null and Alternative Hypotheses
2. Compute the Test Statistic
   - The test statistic is a number that summarizes the evidence
3. Determine the p-value (from the Randomization Distribution)
4. Draw a conclusion

4.2 P-values

4.2.1 Testing one proportion

Randomization distributions are a bit like bootstrap distributions except that instead of resampling from our sample (in an attempt to approximate resampling from the population), we need to sample from a situation in which our null hypothesis is true.

The simplest case for this is a null hypothesis about a single proportion. For example, to test whether a coin is a fair coin, we might flip the coin some number of times and keep track of the heads and tails. Suppose the data are 60 heads and 40 tails in 100 tries.

1. $H_0: p = 0.5; H_a: p \neq 0.5$
2. Test statistic: $\hat{p}$ (the sample proportion)
3. We can simulate a world in which $p = 0.5$ using `rf1ip()`:
Hypothesis Tests

```r
RandomizationDist <- do(5000) * rflip(100, 0.5)  # 100 because n=100
define head(RandomizationDist)

  n heads tails prop
1 100  53  47  0.53
2 100  49  51  0.49
3 100  53  47  0.53
4 100  49  51  0.49
5 100  54  46  0.54
6 100  39  61  0.39

prop(\neg \text{prop} \geq 0.6), \text{data} = \text{RandomizationDist})

  \text{TRUE}
0.029

prop(\neg \text{prop} \leq 0.4), \text{data} = \text{RandomizationDist})

  \text{TRUE}
0.032

# a 2-sided p-value is the sum of the values above
prop(\neg \text{prop} \leq 0.4 \mid \text{prop} \geq 0.6), \text{data} = \text{RandomizationDist})

  \text{TRUE}
0.061

# We can also approximate the p-value by doubling one side:
2 \times \text{prop}(\neg \text{prop} \geq 0.6), \text{data} = \text{RandomizationDist})

  \text{TRUE}
0.058
```

4. This p-value is on the small side, but not extremely small. If we flip a fair coin 100 times, there is a 6.1% chance that we would get 60% or more head or tails.

4.3 Significance

4.4 Creating Randomization Distributions

In order to use these methods to estimate a p-value, we must be able to generate a randomization distribution. In the case of a test with null hypothesis claiming that a proportion has a particular value (e.g., \(H_0: \ p = 0.5\)), this is pretty easy. If the population has proportion 0.50, we can simulate sampling from that proportion by flipping a fair coin. If the proportion is some value other than 0.50, we simply flip a coin that has the appropriate probability of resulting in heads. So the general template for creating such a randomization distribution is

```r
do(1000) * rflip(n, hypothesized_proportion)
```
where \( n \) is the size of the original sample.

In other situations, it can be more challenging to create a randomization distribution because the null hypothesis does not directly specify all of the information needed to simulate samples.

- \( H_0: p_1 = p_2 \)
  This would be simple if we knew the value of \( p_1 \) and \( p_2 \) (we could use \text{rflip()} \) twice, once for each group),

- \( H_0: \mu = \text{some number} \)
  Just knowing the mean does not tell us enough about the distribution. We need to know about its shape. (We might need to know the standard deviation, for example, or whether the distribution is skewed.)

- \( H_0: \mu_1 \neq \mu_2 \text{ some number}. \)
  Now we don’t know the common mean and we don’t know the things mentioned in the previous example either.

So how do we come up with randomization distribution?

The main criteria to consider when creating randomization samples for a statistical test are:

- Be consistent with the null hypothesis.
  If we don’t do this, we won’t be testing our null hypothesis.

- Use the data in the original sample.
  With luck, the original data will shed light on some aspects of the distribution that are not determined by null hypothesis.

- Reflect the way the original data were collected.

### 4.4.1 Testing Whether Two Proportions are Equal

Creating some data

Data 4.7 in the text describes some data that are not in a data frame. This often happens when a data set has only categorical variables because a simple table completely describes the distributions involved. Here’s the table from the book:

<table>
<thead>
<tr>
<th></th>
<th>Relapse</th>
<th>No Relapse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lithium</td>
<td>18</td>
<td>6</td>
</tr>
<tr>
<td>Placebo</td>
<td>20</td>
<td>4</td>
</tr>
</tbody>
</table>

Here’s one way to create the data in R:

```r
Cocaine <- rbind(
   do(18) * data.frame( treatment = "Lithium", response="Relapse"),
   do(6)  * data.frame( treatment = "Lithium", response="No Relapse"),
   do(20) * data.frame( treatment = "Placebo", response="Relapse"),
   do(4)  * data.frame( treatment = "Placebo", response="No Relapse")
)
```

1 The book includes data on an additional treatment group which we are omitting here.
A Randomized Experiment: Lithium vs. Placebo

tally(response ~ treatment, data = Cocaine)

<table>
<thead>
<tr>
<th>Treatment</th>
<th>Lithium</th>
<th>Placebo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relapse</td>
<td>0.7500</td>
<td>0.8333</td>
</tr>
<tr>
<td>No Relapse</td>
<td>0.2500</td>
<td>0.1667</td>
</tr>
<tr>
<td>Total</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

prop(response ~ treatment, data = Cocaine)

Relapse.Lithium  Relapse.Placebo
                0.75000  0.83333

diff(prop(response ~ treatment, data = Cocaine))

Relapse.Placebo
0.08333

Randomization.Coc <- do(1000) * diff(prop(response ~ shuffle(treatment), data = Cocaine))
head(Randomization.Coc)

<table>
<thead>
<tr>
<th>Relapse.Placebo</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0833</td>
</tr>
<tr>
<td>2</td>
<td>0.1667</td>
</tr>
<tr>
<td>3</td>
<td>-0.0833</td>
</tr>
<tr>
<td>4</td>
<td>0.1667</td>
</tr>
<tr>
<td>5</td>
<td>0.0833</td>
</tr>
<tr>
<td>6</td>
<td>0.0833</td>
</tr>
</tbody>
</table>

histogram(~Relapse.Placebo, data = Randomization.Coc, v = c(-0.0833, 0.0833), width = 0.08)

prop(~(Relapse.Placebo > 0.0833), data = Randomization.Coc)  # 1-sided

TRUE
4.4.2 Testing Whether Two Means are Equal

Random assignment

The procedure for comparing two means is very similar

```r
mean(BodyTemp ~ Sex, data = BodyTemp50)

Female  Male
   98.17 98.35

diff(mean(BodyTemp ~ Sex, data = BodyTemp50))

Male
   0.176
```

```r
Randomization.Temp <- do(1000) * diff(mean(BodyTemp ~ shuffle(Sex), data = BodyTemp50))
head(Randomization.Temp, 3)

   Male
 1 -0.424
 2  0.376
 3 -0.456

histogram(~Male, data = Randomization.Temp, v = c(0.176, -0.176))
```
Male Density

1 -0.5 0.0 0.5

prop(\neg (\text{Male} > 0.176), \text{data} = \text{Randomization.Temp}) \quad \# \text{1-sided}

TRUE 
0.216

2 * prop(\neg (\text{Male} > 0.176), \text{data} = \text{Randomization.Temp}) \quad \# \text{2-sided}

TRUE 
0.432

prop(\neg (\text{abs(Male)} > 0.176), \text{data} = \text{Randomization.Temp}) \quad \# \text{2-sided}

TRUE 
0.44

Separate sampling from two groups

But wait; one of our criteria was that we our randomization should mimic the design of the study. In this case, we did not randomly assign the sex of the subjects (for obvious reasons). Is there a randomization method that more accurately reflects what actually happened? It turns out there is. We could

- First pool the 50 subjects’ body temperatures.
  (If men and women aren’t different, then they really form one population with the same distribution rather than two different distributions, so we can safely combine in this way when the null hypothesis is true.)

- Then randomly draw two samples of size 25 with replacement to create simulated groups of women and men.

Here’s a way to do this in R:

\begin{verbatim}
Randomization.Temp2 <- do(1000) * diff(mean(resample(BodyTemp) ~ shuffle(Sex), data = BodyTemp50))
head(Randomization.Temp2, 3)
\end{verbatim}

\begin{verbatim}
   Male 
1 -0.160
\end{verbatim}
Sampling from one population with two variables

If on the other hand our study design was to randomly sample 50 people and record their body temperature and their sex, then another approach could be used:

```r
Randomization.Temp3 <- do(1000) * diff(mean(BodyTemp ~ shuffle(Sex), data = resample(BodyTemp50)))
head(Randomization.Temp3, 3)
```

```
   Male
1 0.1160
2 0.2300
3 0.4906
```

```r
histogram(~Male, data = Randomization.Temp3, v = c(0.176, -0.176))
```
Hypothesis Tests

As we see, the three methods give essentially the same results in this case. But they correspond to three different study designs:

1. Randomized assignment of treatments (shuffle the treatment)
2. Separate sampling from two populations (resample the response, shuffle the treatment)
   Each randomization sample will have the same number of cases in each group.
3. Observational study on two variables (resample entire data set, and shuffle the explanatory variable)
   Different randomization samples will have different numbers of cases in the two groups.

Similar approaches work for testing whether two proportions are equal as well.

A tell-tale sign that your method isn’t correct

Take a look at the three histograms in this section. Notice how each is centered at 0. (They also have our hoped for bell shape.)

```
prop(~(Male > 0.176), data = Randomization.Temp3)  # 1-sided

TRUE
0.202

2 * prop(~(Male > 0.176), data = Randomization.Temp3)  # 2-sided

TRUE
0.404

prop(~(abs(Male) > 0.176), data = Randomization.Temp3)  # 2-sided

TRUE
0.426
```
This is because we are computing the difference in means assuming the difference in means is 0.

If your randomization distribution is centered at the wrong value, then it isn’t simulating a world in which the null hypothesis is true. This would happen, for example, if we got confused about randomization vs. bootstrapping.

```
Boot.Temp <- do (1000) * diff( mean( BodyTemp ~ Sex, data=resample(BodyTemp50) ) )
head(Boot.Temp,3)
```

```
   Male 1  0.2643
2 -0.1806
3  0.3752
```

```
histogram(~ Male, data=Boot.Temp, v=c(0.176, -0.176),
             main="This is NOT a randomization distribution")
mean(~ Male, data=Boot.Temp)
```

[1] 0.1829

Notice that the distribution is now centered at our test statistic instead of at the value from the null hypothesis.
4.4.3 Testing One mean

Is body temperature really 98.6?

Suppose now that we want to test whether the mean body temperature is really 98.6 degrees F., as we’ve always been told. Our null and alternative hypotheses are

- $H_0: \mu = 98.6$
- $H_a: \mu \neq 98.6$

The sample mean body temperature is

```r
mean(~BodyTemp, data = BodyTemp50)
```

```
[1] 98.26
```

This will be our test statistic. Notice that it differs a bit from 98.6

```r
98.6 - mean(~BodyTemp, data = BodyTemp50)
```

```
[1] 0.34
```

But might this just be random variation? We need a randomization distribution to compare against. But how do we create one? If we resample, the mean will not be 98.6. But we shift the distribution a bit, then we will have the desired mean while preserving the shape of the distribution indicated by our sample.

```r
Randomization.Temp4 <- do(1000) * (mean(~BodyTemp, data = resample(BodyTemp50)) + 0.34)
head(Randomization.Temp4, 3)
```

```
result
1 98.76
2 98.69
3 98.62
```

```r
mean(~result, data = Randomization.Temp4)
```

```
[1] 98.6
```

```r
histogram(~result, data = Randomization.Temp4, v = c(98.26, 98.94))
```
From this we can estimate the p-value:

`prop(abs(result - 98.6) > 0.34, data = Randomization.Temp4)`

TRUE

How do we interpret this (estimated) p-value of 0? Is it impossible to have a sample mean so far from 98.6 if the true population mean is 98.6? No. This merely means that we didn’t see any such cases in our 1000 randomization samples. We might estimate the p-value as \( p < 0.001 \).

To more accurately estimate small p-values, we must use many more randomization samples.

`Randomization.Temp4a <- do(10000) * (mean(~BodyTemp, data = resample(BodyTemp50)) + 0.34)`

`histogram(~result, data = Randomization.Temp4a, v = c(98.26, 98.94))`

`prop(abs(result - 98.6) > 0.34, data = Randomization.Temp4a)`

TRUE

0.001

A different approach

An equivalent way to do the preceding test is based on a different way of expressing our hypotheses.

- \( H_0: \mu - 98.6 = 0 \)
• \( H_a: \mu - 98.6 \neq 0 \)

Now our test statistic is

\[
\text{mean}(\text{BodyTemp, data = BodyTemp50}) - 98.6
\]

\[\text{[1]} -0.34\]

and we create a randomization distribution centered at 0:

\[
\text{Randomization.Temp5} \leftarrow \text{do(1000)} \times (\text{mean}(\text{BodyTemp, data = resample(BodyTemp50)}) - 98.26)
\]

\[
\text{head(Randomization.Temp5, 3)}
\]

\[
\text{result}
\]

1 -0.034
2 0.042
3 0.040

\[
\text{mean(\text{\textminus{}result, data = Randomization.Temp5})}
\]

\[\text{[1]} 0.002082\]

\[
\text{histogram(\text{\textminus{}result, data = Randomization.Temp5, v = c(0.34, -0.34))}
\]

From this we can estimate the p-value:

\[
\text{prop(\text{\textminus{}abs(result)} > 0.34, data = Randomization.Temp5)}
\]

\[
\text{TRUE}
\]

0.001

Often there are multiple ways to express the same hypothesis test.
5.1 The Family of Normal Distributions

Normal Distributions

- are symmetric, unimodal, and bell-shaped
- can have any combination of mean and standard deviation (as long as the standard deviation is positive)
- satisfy the 68–95–99.7 rule:
  - Approximately 68% of any normal distribution lies within 1 standard deviation of the mean.
  - Approximately 95% of any normal distribution lies within 2 standard deviations of the mean.
  - Approximately 99.7% of any normal distribution lies within 3 standard deviations of the mean.

Many naturally occurring distributions are approximately normally distributed.

Normal distributions are also an important part of statistical inference.
Example: SAT scores

SAT scores used to be approximately normally distributed with a mean of 500 and a standard deviation of 100, so

- approximately 68% of scores were between 400 and 600
- approximately 95% of scores were between 300 and 700
- approximately 2.5% of scores were above 700
- approximately 81.5% (34% + 47.5%) of scores are between 400 and 700

The two main functions we need for working with normal distributions are `pnorm()` and `qnorm()`. `pnorm()` computes the proportion of a normal distribution below a specified value:

\[
\text{pnorm}(x, \text{mean}=\mu, \text{sd}=\sigma) = \Pr(X \leq x)
\]

when \( X \sim \text{Norm}(\mu, \sigma) \).

We can obtain arbitrary probabilities using `pnorm()`:

```r
pnorm(650, 500, 100) # proportion who score below 650 on SAT
[1] 0.9332
1 - pnorm(650, 500, 100) # proportion who score above 650 on SAT
[1] 0.06681
pnorm(650, 500, 100) - pnorm(400, 500, 100) # proportion who score between 400 and 650 on SAT
[1] 0.7745
```

`qnorm()` goes the other direction: You provide the quantile (percentile expressed as a decimal) and R gives you the value.

```r
# find 80th percentile in Norm(500, 100)
qnorm(0.8, 500, 100)
```

Last Modified: November 5, 2013
The `xpnorm()` function gives a bit more verbose output and also gives you a picture.

\[ \text{xpnorm}(700, 500, 100) \]

If \( X \sim N(500, 100) \), then

\[
\begin{align*}
P(X \leq 700) &= P(Z \leq 2) = 0.9772 \\
P(X > 700) &= P(Z > 2) = 0.0228
\end{align*}
\]

\[ \text{[1] 0.9772} \]

Z-scores (standardized scores)

Because probabilities in a normal distribution depend only on the number of standard deviations above and below the mean, it is useful to define Z-scores (also called standardized scores) as follows:

\[
Z\text{-score} = \frac{\text{value} - \text{mean}}{\text{standard deviation}}
\]

If we know the population mean and standard deviation, we can plug those in. When we do not, we will use the mean and standard deviation of a random sample as an estimate.

Z-scores provide a second way to compute normal probabilities:

\[
\text{pnorm} ( 650, 500, 100 ) \quad \# \text{proportion of SATs below 650}
\]

\[ \text{[1] 0.9332} \]

\[
z \leftarrow (650 - 500) / 100 \quad \# \text{z-score}
\]

\[
\text{pnorm} (1.5) \quad \# \text{proportion with z-score below 1.5}
\]

\[ \text{[1] 0.9332} \]
When sampling distributions, bootstrap distributions, and randomization distributions are well approximated by normal distributions, and when we have a way of computing the standard error, we can use normal distributions to compute confidence intervals and p-values using the following general templates:

- confidence interval: \( \text{statistic} \pm \text{critical value} \cdot SE \)

- hypothesis testing: \( \text{test statistic} = \frac{\text{statistic} - \text{null parameter}}{SE} \)

### 6.1 Inference for One Proportion

If we are working with one categorical variable, we can compute confidence intervals and p-values using the following standard error formula:

\[ SE = \sqrt{\frac{p(1-p)}{n}} \]

Actually that’s not quite right. We could use this if we knew \( p \) (the population proportion), but we never will. Instead, we will use one of these approximations:

- confidence intervals: \( SE = \sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \)
  
  We will estimate \( p \) using our sample statistic \( \hat{p} \).

- p-values: \( SE = \sqrt{\frac{p_0(1-p_0)}{n}} \)
  
  Since p-values are computed assuming \( p = p_0 \) (the null parameter), we should use \( p_0 \) for computing \( SE \).

The distributions will be close enough to normal for our purposes when \( pn \geq 10 \text{ and } (1-p)n \geq 10 \).

(We’ll check these using \( \hat{p} \) or \( p_0 \).)
6.1.1 Examples

Rock, Paper, Scissors

In a study to see whether people playing rock paper scissors are equally likely to choose rock, paper, or scissors on their first round, 119 people were observed. 66 of them chose rock.

We can use this data to test the hypotheses:

- $H_0: p = 1/3$
- $H_a: p \neq 1/3$

where $p$ is the proportion of people who start with rock.

First we check that the normal approximation is good enough:

```r
p.hat <- 66/119; p.hat
[1] 0.5546

p.hat * 119 # check >= 10
[1] 66

(1 - p.hat) * 119 # check >= 10
[1] 53

SE <- sqrt( 1/3 * 2/3 / 119 ); SE
[1] 0.04321

z <- (p.hat - 1/3) / SE; z
[1] 5.121

pnorm(z) # large side (rounded)
[1] 1

1 - pnorm(z) # small side (less rounding)
[1] 1.521e-07

2 * (1 - pnorm(z)) # p-value = 2 * small side
[1] 3.042e-07
```
We hardly needed to compute a p-value in this case. A test statistic of $z = 5.12$ already tells us that the p-value will be quite small.

We can also compute a 95% confidence interval for the proportion of people who start with rock:

```
p.hat # same as above
[1] 0.5546

1/3 * 119                     # check >= 10
[1] 39.67

2/3 * 119                     # check >= 10
[1] 79.33

SE <- sqrt( p.hat * (1 - p.hat) / 119 ); SE # est. SE is (a little) different now
[1] 0.04556

p.hat - 1.96 * SE            # lower end of CI
[1] 0.4653

p.hat + 1.96 * SE            # upper end of CI
[1] 0.6439
```

If we want a different confidence level, we simply adjust the critical value. Let’s compute a 98% confidence interval.

```
z.star <- qnorm( 0.99 ); z.star # critical value for 98% CI
[1] 2.326

p.hat <- 66/119; p.hat # same as above
[1] 0.5546

SE <- sqrt( p.hat * (1 - p.hat) / 119 ); SE # est. SE is (a little) different now
[1] 0.04556

p.hat - z.star * SE # lower end of CI
[1] 0.4653
```
Pollination

Many plants are capable of self-pollination. To keep this from happening, some flowers come in left- and right-handed versions. In left-handed plants, the pollen gets on the left sides of visiting bees. In right-handed plants, pollen gets on the right sides of bees. As the bees fly from flower to flower, the right-handed plants pollinate the left-handed plants, and vice versa.

One genetic model suggests that there should be a 3:1 ratio of right-handed to left-handed flowers. Scientists observed 6 right-handed and 21 right-handed flowers. How is this data related to the conjecture of a simple Mendelian inheritance model?

We can test $H_0: p = 1/4$ against a two-sided alternative as follows:

```
p.hat <- 6/27; p.hat

[1] 0.2222
```

```
Rand.Flowers <- do (1000) * rflip(27, .25)
histogram(~prop, data=Rand.Flowers, width=1/27)
prop(~ (prop < p.hat), data=Rand.Flowers)  # one-sided p-value

TRUE
0.29

2 * prop(~ (prop < p.hat), data=Rand.Flowers)  # two-sided p-value

TRUE
0.58
```

Notice that we need to be a bit cautious about using the normal distribution in this case since $p_0 n < 10$: 
Let's see how different the results are:

\[
SE \leftarrow \sqrt{\frac{1/4 \times 3/4}{27}}; \ SE
\]

\[
[1] \ 0.08333
\]

\[
z \leftarrow \frac{\text{p.hat} - 1/4}{SE}; \ z
\]

\[
[1] \ -0.3333
\]

\[
pnorm(z) \quad \text{# one-sided p-value}
\]

\[
[1] \ 0.3694
\]

\[
2 \times pnorm(z) \quad \text{# two-sided p-value}
\]

\[
[1] \ 0.7389
\]

The approximation is not very good in this case (0.7389 vs. 0.58), but the conclusion either way would be the same – do not reject the null hypothesis. In this case, that means that the data are consistent with the null hypothesis. But there is not a lot of data, and it may well be consistent with other hypotheses as well.

Helper-Hinderer

We all recognize the difference between naughty and nice, right? What about children less than a year old – do they recognize the difference and show a preference for nice over naughty? In a study reported in the November 2007 issue of *Nature*, researchers investigated whether infants take into account an individual's actions towards others in evaluating that individual as appealing or aversive, perhaps laying the foundation for social interaction. In one component of the study, 10-month-old infants were shown a climber character (a piece of wood with eyes glued onto it) that could not make it up a hill in two tries. Then they were alternately shown two scenarios for the climber's next try, one where the climber was pushed to the top of the hill by another character (“helper”) and one where the climber was pushed back down the hill by another character (“hinderer”). The infant was alternately shown these two scenarios several times. Then the child was presented with both pieces of wood (the helper and the hinderer) and asked to pick one to play with. The researchers found that the 14 of the 16 infants chose the helper over the hinderer.

What can we conclude? More of them chose the helper than the hinderer. But maybe it was just chance. Maybe they just grab one of the toys and it has nothing to do with whether or not they had seen the toys be helpful.

We can test

\[ H_0: p = 1/2 \]

\[ H_a: p \neq 1/2 \]
to see whether 14 out of 16 is unusual enough to lead us to believe that the children are doing something other than guessing.

Once again, the normal approximation method might not be as accurate as we would like because the sample size is a bit small (but closer to our rule of thumb than in the preceding example):

$$16 \times \frac{1}{2} \quad \# \text{a bit too small}$$

[1] 8

So let's do this two ways. Once using the normal approximation and once using the randomization distribution and see how close the results are.

```
p.hat <- 14/16; p.hat
[1] 0.875

SE <- sqrt((1/2 * 1/2) / 16); SE
[1] 0.125

z <- (p.hat - 1/2) / SE; z
[1] 3

pnorm(z)
[1] 0.9987

2 * (1 - pnorm(z)) \quad \# p-value
[1] 0.0027
```

```
Rand.toys <- do(5000) * rflip(16, 0.5)
head(Rand.toys, 3)

   n heads tails  prop
1  16     10     6  0.6250
2  16      6    10  0.3750
3  16     11     5  0.6875

2 * prop(~(prop >= p.hat), data = Rand.toys)

TRUE
0.004

histogram(~prop, data = Rand.toys, width = 1/16)
```
In this case, the normal approximation is not so far off (which shouldn’t surprise us too much since 8 is not that far from 10), but we would expect that randomization p-value to be the more accurate one in this situation. In both cases the p-values are less than 0.05, so we can reject the null hypothesis (at the usual $\alpha = 0.05$ level) and conclude that the kids are not just guessing.

### 6.1.2 Automating Everything

R can automate the entire process for us. Here are some examples.

**Pollination**

```r
binom.test(6, 27, p = 0.25)  # using an exact method

Exact binomial test

data:  x and n
two-sided p-value = 1

prop.test(6, 27, p = 0.25)  # using normal approximation

1-sample proportions test with continuity correction

data:  x and n
X-squared = 0.0123, df = 1, p-value = 0.9115
alternative hypothesis: true p is not equal to 0.25
95 percent confidence interval:
 0.09376 0.42734
sample estimates:
     p
0.2222
```

Math 145: Fall 2013: Pruim

Last Modified: November 5, 2013
The proportion test above uses a fancier version of the normal approximation (employing the so-called continuity correction, which usually makes it more accurate). We can get R to do without this correction, and then the results will match ours above:

```r
prop.test(6, 27, p = 0.5, correct = FALSE)
```

```
1-sample proportions test without continuity correction
data:  x and n
X-squared = 8.333, df = 1, p-value = 0.003892
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.1061 0.4076
sample estimates:
    p
0.2222
```

\[ z^2 \] # should match X-squared in the output above.

[1] 9

---

Helper-Hinderer

```r
binom.test(14, 16, p = 0.5)  # using an exact method
```

```
Exact binomial test
data:  x and n
number of successes = 14, number of trials = 16, p-value = 0.004181
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
 0.6165 0.9845
sample estimates:
probability of success
0.875
```

```r
prop.test(14, 16, p = 0.5)  # using normal approximation
```

```
1-sample proportions test with continuity correction
data:  x and n
X-squared = 7.562, df = 1, p-value = 0.00596
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.6041 0.9780
sample estimates:
    p
0.875
```
The proportion test above uses a fancier version of the normal approximation (employing the so-called continuity correction, which usually makes it more accurate). We can get R to do without this correction, and then the results will match ours above:

```r
prop.test(14, 16, p = 0.5, correct = FALSE)
```

```
1-sample proportions test without continuity correction
data:  x and n
X-squared = 9, df = 1, p-value = 0.0027
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.6398 0.9650
sample estimates:
p 0.875
```

\[z^2 \# \text{should match } X^2 \text{-squared in the output above.}\]

```
[1] 9
```

### 6.1.3 Determining Sample Size

An important part of designing a study is deciding how large the sample needs to be for the intended purposes of the study.

**Q.** You have been asked to conduct a public opinion survey to determine what percentage of the residents of a city are in favor of the mayor’s new deficit reduction efforts. You need to have a margin of error of ±3%. How large must your sample be? (Assume a 95% confidence interval.)

**A.** The margin of error will be \[1.96 \sqrt{\frac{\hat{p}(1-\hat{p})}{n}}\], so our method will be to make a reasonable guess about what \(\hat{p}\) will be, and then determine how large \(n\) must be to make the margin of error small enough.

Since \(SE\) is largest when \(\hat{p} = 0.50\), one safe estimate for \(\hat{p}\) is 0.50, and that is what we will use unless we are quite sure that \(p\) is close to 0 or 1. (In those latter cases, we will make a best guess, erring on the side of being too close to 0.50 to avoid doing the work of getting a sample much larger than we need.)

We can solve 0.03 = \[\sqrt{\frac{0.5 \cdot 0.5}{n}}\] algebraically, or we can play a simple game of “higher and lower” until we get our value of \(n\). We could also graph \[\sqrt{\frac{0.5 \cdot 0.5}{n}}\] and use the graph to estimate \(n\). Here’s the “higher/lower” guessing method.

```
1.96 * sqrt(0.5 * 0.5/400)
```

```
[1] 0.049
```

```
1.96 * sqrt(0.5 * 0.5/800)
```

```
[1] 0.03465
```
Inference with Normal Distributions

\[ 1.96 \times \sqrt{0.5 \times 0.5/1200} \]

[1] 0.02829

\[ 1.96 \times \sqrt{0.5 \times 0.5/1000} \]

[1] 0.03099

\[ 1.96 \times \sqrt{0.5 \times 0.5/1100} \]

[1] 0.02955

\[ 1.96 \times \sqrt{0.5 \times 0.5/1050} \]

[1] 0.03024

So we need a sample size a bit larger than 1050, but not as large as 1100. We can continue this process to get a tighter estimate if we like:

\[ 1.96 \times \sqrt{0.5 \times 0.5/1075} \]

[1] 0.02989

\[ 1.96 \times \sqrt{0.5 \times 0.5/1065} \]

[1] 0.03003

\[ 1.96 \times \sqrt{0.5 \times 0.5/1070} \]

[1] 0.02996

\[ 1.96 \times \sqrt{0.5 \times 0.5/1068} \]

[1] 0.02999

\[ 1.96 \times \sqrt{0.5 \times 0.5/1067} \]

[1] 0.03

We see that a sample of size 1067 is guaranteed to give us a margin of error of at most 3%.

It isn’t really important to get this down to the nearest whole number, however. Our goal is to know roughly what size sample we need (tens? hundreds? thousands? tens of thousands?). Knowing the answer to 2 significant figures is usually sufficient for planning purposes.

Side note: The R function \texttt{uniroot} can automate this guessing for us, but it requires a bit of programming to
use it:

```r
# uniroot finds when a function is 0, so we need to build such a function; this function is
# 0 when the margin of error is 0.03:
f <- makeFun(1.96 * sqrt(0.5 * 0.5/n) - 0.03 ~ n)
# uniroot needs a function and a lower bound and upper bound to search between
uniroot(f, c(1, 50000))$root

[1] 1067
```

Example

Q. How would things change in the previous problem if

1. We wanted a 98% confidence interval instead of a 95% confidence interval?
2. We wanted a 95% confidence interval with a margin of error at most 0.5%?
3. We wanted a 95% confidence interval with a margin of error at most 0.5% and we are pretty sure that \( p < 10\% \)?

A. We’ll use `uniroot()` here. You should use the “higher-lower” method or algebra and compare your results.

```r
f1 <- makeFun(qnorm(0.99) * sqrt(0.5 * 0.5/n) - 0.03 ~ n)
uniroot(f1, c(1, 50000))$root

[1] 1503
```

```r
f2 <- makeFun(qnorm(0.975) * sqrt(0.5 * 0.5/n) - 0.005 ~ n)
uniroot(f2, c(1, 50000))$root

[1] 38415
```

```r
f3 <- makeFun(qnorm(0.99) * sqrt(0.1 * 0.9/n) - 0.005 ~ n)
uniroot(f3, c(1, 50000))$root

[1] 19483
```

### 6.2 Inference for One Mean

If we are working with one quantitative variable, we can compute confidence intervals and p-values using the following standard error formula:

\[
SE = \frac{\sigma}{\sqrt{n}}
\]

Once again, there is a small problem: we won’t know \( \sigma \). So we will estimate \( \sigma \) using our data:

\[
SE \approx \frac{s}{\sqrt{n}}
\]
Unfortunately, the distribution of 
\[ \frac{\bar{x} - \mu}{s/\sqrt{n}} \]
does not have a normal distribution. Instead the distribution is a bit “shorter and fatter” than the normal distribution. The correct distribution is called the t-distribution with \( n - 1 \) degrees of freedom. All t-distributions are symmetric and centered at zero. The smaller the degrees of freedom, the shorter and fatter the t-distribution.

Using t-distributions, our templates for confidence intervals and p-values become the following:

- confidence intervals: \( \bar{x} \pm t \times SE \) where \( SE = \frac{s}{\sqrt{n}} \) and \( df = n - 1 \).
- p-values: \( t = \frac{\bar{x} - \mu_0}{SE} \) where \( SE = \frac{s}{\sqrt{n}} \) and \( df = n - 1 \).

Here is some R code related to t-tests and confidence intervals.

6.2.1 The Long Way

The test statistic for a null hypothesis of \( H_0 : \mu = \mu_0 \) is
\[ t = \frac{\bar{x} - \mu_0}{s/\sqrt{n}} = \frac{\text{estimate} - \text{hypothesis value}}{\text{standard error}} \]

This is easily computed in RStudio:

```r
# some ingredients
x.bar <- mean(~ BodyTemp, data=BodyTemp50); x.bar
[1] 98.26
sd <- sd(~ BodyTemp, data=BodyTemp50); sd
[1] 0.7653
n <- nrow(BodyTemp50); n
[1] 50
se <- sd/sqrt(n); se
[1] 0.1082

# test statistic
t <- (x.bar - 98.6) / se; t
[1] -3.141

# 2-sided p-value
2 * pt(-abs(t), df=49)
[1] 0.002851
```
Similarly, we can compute a 95% confidence interval

```r
t.star <- qt(0.975, df = 49)
t.star

[1] 2.01

# lower limit
x.bar - t.star * se

[1] 98.04

# upper limit
x.bar + t.star * se

[1] 98.48
```

### 6.2.2 Super Short Cuts

Of course, RStudio can do all of the calculations for you if you give it the raw data:

```r
t.test(~BodyTemp, data = BodyTemp50, mu = 98.6)
```

One Sample t-test

data:  data$BodyTemp
t = -3.141, df = 49, p-value = 0.002851
alternative hypothesis: true mean is not equal to 98.6
95 percent confidence interval:
  98.04 98.48
sample estimates:
mean of x
  98.26

You can also zoom in just the information you want:

```r
pval(t.test(~BodyTemp, data = BodyTemp50, mu = 98.6))
```

p.value
0.002851

```r
confint(t.test(~BodyTemp, data = BodyTemp50))
```

<table>
<thead>
<tr>
<th>mean of x</th>
<th>lower</th>
<th>upper</th>
<th>level</th>
</tr>
</thead>
<tbody>
<tr>
<td>98.26</td>
<td>98.04</td>
<td>98.48</td>
<td>0.95</td>
</tr>
</tbody>
</table>
6.2.3 Sample Size Estimation

Suppose we want to estimate the mean height of male Calvin students to within ±1 inch with 95% confidence. That is we want

\[ t \times SE = t \times \frac{s}{\sqrt{n}} = 1 \]

In order to determine our sample size, we will estimate \( t \approx 2 \) (this should be pretty close unless the require sample size is very small.) We also need to come up with some estimate of the standard deviation. In this case, other data has been used to estimate the standard deviation of the heights of 18–25 year-old males to be about 3 in. If we assume that our population has a similar standard deviation, then we want

\[ 2 \times \frac{3}{\sqrt{n}} = 1 \]

We can solve this pretty easily to show that \( n = 6^2 = 36 \). So we need roughly 40 subjects to gain that level of precision. Again, the point is not to determine the exact number but to know roughly what size sample will suffice.

6.2.4 The Central Limit Theorem and When These Approximations are Good Enough

William Gosset proved that using \( SE = \frac{s}{\sqrt{n}} \) and the appropriate \( t \)-distribution works perfectly when the population is exactly normally distributed. But what happens when the population is not normally distributed? An important theorem called the Central Limit Theorem says that as the sample sizes get larger and larger, the shape of the population distribution matters less and less.

The usual rule of thumb given is that

1. When the population is normal, we can use these methods for any sample size.
2. When the population is very nearly normal (unimodal and at most modestly skewed), we can use these methods even for fairly small sample sizes.
3. For most population distributions that occur in practice, sample sizes of \( n \geq 30 \) work quite well. (But see the next item.)
4. Because outliers can substantially affect both the mean \( \bar{x} \) and the standard deviation \( s \), outliers should be investigated. If it is unclear why an unusual observation is in the data, it may not be clear whether it should be removed or not, and the conclusions may differ substantially with and without the outlier.

As with all rules of thumb, the numbers given are guidelines. Things always work better when the sample size is larger and the population distribution is more like a normal distribution. Unfortunately, when sample sizes are small, it is difficult to assess whether the population is normal.\(^1\)

Tim Hesterberg (a statistician at Google) has argued that (1) this rule of thumb doesn’t actually work very well, (2) there is a much better approach to deciding based on looking at the bootstrap distribution, and (3) it can be very difficult to tell how well things are working when sample sizes are very small. For small sample sizes, we really need to be convinced (for reasons other than our data) that the population should be expected to be approximately normally distributed.

---

\(^1\) Gossett offered the following advice: when the sample size is large, the population distribution doesn’t matter, so use the \( t \)-distribution methods. When the sample size is small, you won’t be able to tell that the population isn’t normal, so use the \( t \)-distribution methods. That’s a little too cavalier with small samples. But the method does work remarkably well in many situations – even situations where the sample sizes are modest.


Index

CPS85, 52
CricketChirps, 13, 15
HELPct, 20, 31, 48, 52
Lock5Data, 20
bargraph(), 22
bargraph, 26
bwplot(), 22, 25
c(), 36
cdata(), 59
colors(), 37
cut(), 27
data(), 20
demo(), 40
densityplot(), 22, 24
do(), 9
fastR, 50, 52
fectchGoogle(), 45
freqpolygon(), 24
fusion1, 50, 52
fusion2, 52
gdata, 47
histogram(), 22, 24, 27, 29
iris, 14, 15
lattice, 28, 30, 40
log(), 17
log10(), 17
merge(), 53
mosaic, 9, 20, 29, 52
pheno, 50, 52
pnorm, 80
qnorm(), 80
qnorm, 80
read.xls(), 47
rflip(), 9, 63, 67, 69
save(), 50
show.settings(), 37
sqrt(), 17
subset(), 48, 49
tally(), 10, 26, 27, 31
tally, 26
trellis.par.set(), 37
write.csv(), 50
xpnorm(), 81
xyplot(), 30
data, 9
Fisher, R. A., 6, 8
lady tasting tea, 6
population, 7
randomness, 6
sample, 7