STAT 215: Lab 3

Sampling Distributions and Confidence Intervals

Last Revised September 22, 2017

Note: This lab is adapted from the Sampling Distributions lab accompanying the OpenIntro statistics textbooks.

The goals of this lab are the following:

- to take advantage of the computer's ability to do fast, repeated simulations to get a feel for what factors affect the distribution of a sample statistic, such as a proportion, or a sample mean
- to explore a simulation-based method of computing standard errors of a statistic like a sample mean
- to use that estimated standard error to create confidence intervals about a parameter, like a population mean

What to Turn In

You should turn in two files, an `.Rmd` source file and a `.pdf` output file. Your writeup should contain your responses/solutions to the exercises labeled "Homework". Note that there are two different sections of "Homework" problems.

The suggested Markdown workflow from last week's lab is reproduced on the next page for reference

R Markdown Workflow

Once you have done this a few times you will develop your own habits, but these are good habits to develop the first few times you use Markdown.

1. From the RStudio File menu, select New File > R Markdown.... Now you have several options. From the list on the lefthand side, select From Template, and then on the right, pick mosaic fancy to get a template that includes a bunch of examples showing you how to include various kinds of content. As you create your own documents you will want to delete much of the extra stuff in the middle (starting with the line `## Using RMarkdown` and ending before `* File creation date`), but reading this template is a good way to learn about how Markdown works.

2. The first thing you should do after creating a new Markdown document is to save the file. Do File > Save As... and give your document a name.

3. Before editing anything, press the Knit HTML button in the document toolbar (or type Ctrl-Shift-k / Cmd-Shift-k). If you are on the server or if you happen to have LaTeX installed on your computer, you could choose Knit PDF instead. If on the server you may get a message about a blocked pop-up. Allow it to show, and you will get a new window with a mix of code, text, and plots. If you prefer, you can change the setting that puts the output in a new window and have it show up in the lower right pane of RStudio instead. You will find this option under Tools > Global Options... > R Markdown in a dropdown menu that says "Show output preview in..."

4. The first thing you should edit is the Title and Author fields at the top of the document. Give the document a title such as "STAT 113: Lab 3", and put your name inside the quotes on the Author line. Re-Knit to see your change reflected in the document.

5. When you make a change, you can test that your code works as expected by first sending that chunk's code to the console by clicking the little green triangle (looks like a "play" button) in the upper right corner of the chunk. This will execute all the code in that chunk in the console. If this chunk depends on earlier ones, you may first need to click the button to the left that send all code in previous chunks to the console. If it works as expected, "Knit" again to update your output.

Notice that there is also a reference guide with lots more info on customizing formats, etc., under Help Markdown Quick Reference that will pop up in the Help tab.

Note You should *not* include code in your Markdown document that you would not want to be run over and over again, or that produces output that you would not put in a writeup. This includes `install.package()` lines, lines that access documentation (e.g., `?EmployedACS`), or generate dialogue boxes.

1 Sampling Distributions

We consider real estate data from the city of Ames, Iowa. The details of every real estate transaction in Ames is recorded by the City Assessor's office. Our particular focus for this lab will be all residential home sales in Ames between 2006 and 2010. This collection represents our population of interest. In this lab we would like to learn about these home sales by taking smaller samples from the full population. Let's load the mosaic package and the data.

```
library("mosaic")
Ames <- read.file("http://colinreimerdawson.com/data/ames.csv")
```

We will focus on the variables `Price`, which records the sale price in dollars of each home in the dataset, and `Area`, which contains the total above-ground living area in square feet.

Exercise 1 Plot and compute summary statistics for the `Area` variable. How would you describe the distribution?

In this lab we have access to the entire population, but this is rarely the case in real life. Gathering information on an entire population is often extremely costly or impossible. Because of this, we often take a sample of the population and use that to understand the properties of the population.

Suppose we were interested in estimating the mean living area in Ames. We might survey a random sample of, say, 50 homes in the city and collect various information about each home. The full dataset contains 2930 homes, so we will have data on less than 3% of the population, but it will turn out that we can make some decent estimates about the population, provided our sample is random.

We can simulate sampling from the population using the following R command.

```r
## Note that if you do not have mosaic loaded,
## you may get different behavior, as base R
## has a sample() function with slightly different
## functionality.
Sample1 <- sample(Ames, size = 50)
```

Exercise 2 Describe the distribution of this random sample. How does it compare to the distribution of the population? Among other things, how does the mean of the sample compare to the mean of the population?

Depending on which 50 homes you selected, your estimate could be a bit above or a bit below the true population mean. In general, though, the sample mean turns out to be a pretty good estimate of the average living area, and we were able to get it by sampling less than 3% of the population.

Exercise 3 Take a second sample, also of size 50, and call it `Sample2`. How does the mean of `Sample2` compare with the mean of `Sample1`? Suppose we took two more samples, one of size 100 and one of size 1000. Which would you think would provide a more accurate estimate of the population mean?

Not surprisingly, every time we take another random sample, we get a different sample mean. It's useful to get a sense of just how much variability we should expect when estimating the population mean this way. The distribution of sample means, called the **sampling distribution**, can help us understand this variability. In the context of a hypothesis test, if we construct a hypothetical sampling distribution assuming the null hypothesis is true, this can give us a sense of how unlikely it would be to get a particular sample mean (say), under this assumption, and therefore to assess the evidence *against* that assumption.

In this lab, because we have access to the population, we can build up the sampling distribution for the sample mean directly by repeating the above steps many times.

Computing a simple sample mean is easy: just get the sample, and compute the mean:

```
mean(~Area, data = Sample1)
```

Note that we could take the sample and compute the mean in one step, as we have done in previous labs with `filter()`:

```
## These are two alternatives to do the same thing: nesting and chaining
mean(~Area, data = sample(Ames, size = 50))  # nesting
sample(Ames, size = 50) %>% mean(~Area, data = .) # chaining
## Note that since each time we run the sample() command we get
## a new sample, the resulting mean will change each time too.
## In contrast, with the first method above (i.e., method (B))
## we have saved our same sample to use over again.  This distinction
## matters when we want to simulate repeated
## sampling: we do not want to use the same sample every time
```

With the speed of modern computers, it is easy to simulate sampling many, many times from a population. For example, we can simulate drawing 5000 samples each of size 50 from the population and looking at how the sample mean varies across these 5000 samples:

```
Sample.means.50 <- do(5000) * sample(Ames, size = 50) %>%
                                mean(~Area, data = .)
```

R and `mosaic` note: The `do()` function in `mosaic` allows you to repeat some code a specified number of times, and store the result of each iteration in a variable. Note that each time we iterate, we get a different random sample. The resulting object, `Sample.means.50`, is a data frame where each case represents a sample of size 50, and the variable recorded is the mean of the `Area` variable for that sample. You can use `head()` to examine the first few means (and take note of the variable name: it can differ depending on whether you used chaining or nesting. It is probably either called `mean` or `result`.

Markdown tip to reduce Knitting time When doing computationally intensive simulation like resampling, it takes a little while to run the sampling code. To avoid having to do this every single time you Knit your document, you can add the option `cache = TRUE` to the code chunk that does the resampling (this works like `include = FALSE` and goes in the same place between the curly braces at the top of the code chunk). This will store the results of the chunk in a "cache" that can simply be read in the next time you Knit. Note though that if any code in the chunk changes, it will be re-run. So it is a good idea to put code that does resampling in a chunk by itself and cache only that chunk, so you are less likely to need to modify it.

Exercise 4 Create a histogram or dot plot of the sample means. You can adjust the number of bins using the `nint=` argument. How many "cases" are there in `Sample.means.50`? Describe the sampling distribution, and be sure to specifically note its center. Would you expect the distribution to change if we instead collected 50,000 sample means?

You should notice two things about the distribution of sample means: it is bell-shaped, and it is centered at the population mean. The sampling distribution that we computed tells us much about estimating the average living area in homes in Ames. Because the sample mean is an unbiased estimator, the sampling distribution is centered at the true average living area of the the population, and the spread of the distribution indicates how much variability is induced by sampling only 50 home sales.

Having simluated the actual sampling distribution, we can calculate its standard error, which is just the standard deviation of the values.

```
true.standard.error <- sd(~result, data = Sample.means.50)
```

Exercise 5 To get a sense of the effect that sample size has on our distribution, build up three sampling distributions: one based samples of size 10, one based on samples of size 50, and another based on samples of size 100. How does the standard error change?

To see the effect that different sample sizes have on the sampling distribution, let's plot the three distributions on top of one another. To do this, let's combine your three data frames into one by stacking them vertically, and then add a column representing the sample size associated with each sample.

```
## The rbind() function stacks data frames on top of each other
## (it assumes they have the same set of variables)
Sampling.Dists <- rbind(Sample.means.10, Sample.means.50, Sample.means.100)
Sampling.Dists <- mutate(Sampling.Dists,
                         Sample.size = factor(rep(c(10,50,100), each = 5000)))
```

Now let's construct overlaid density curves for the sample mean variable, grouped by sample size.

```
densityPlot(result ~ Sample.size, data = Sampling.Dists)
```

Exercise 6 When the sample size is larger, what happens to the center? What about the spread?

2 Homework

So far, we have only focused on living area in homes in Ames. Now let's look at house prices.

1. Take a random sample of size 50. Using this sample, what is your best point estimate of the population mean for the `Price` variable?

2. Since you have access to the population, simulate the sampling distribution for $\bar{x}_{price}$ by taking 5000 samples from the population of size 50 and computing 5000 sample means. Plot the data, then describe the shape of this sampling distribution. Based on this sampling distribution, what would you guess the mean home price of the population to be? Finally, calculate and report the population mean.

3. Change your sample size from 50 to 150, then compute the sampling distribution using the same method as above. Describe the shape of this sampling distribution, and compare it to the sampling distribution for a sample size of 50. Based on this sampling distribution, what would you guess to be the mean sale price of homes in Ames?

4. Of the sampling distributions from 2 and 3, which has a smaller spread? If we're concerned with making estimates that are more often close to the true value, would we prefer a distribution with a large or small spread?

3 Bootstrap Resampling

In reality of course we do not have access to the population, so we cannot get the true standard error by repeatedly sampling: somehow we have to do everything with a single sample.

Let's take a sample of size 60 from the population and compute the mean Area in the sample.

```
Sample60 <- sample(Ames, size = 60)
sample.mean <-  mean(~Area, data = Sample60)
```

We could rely on probability theory to get an analytic estimate of the standard error from the sample. As you may know, the standard error of a sample mean is $\sqrt{n}$

times smaller than the standard deviation of the variable in the population (where n is the sample size):

$$SE_{\bar{X}} = \frac{\sigma_X}{\sqrt{n}}$$

We can estimate σ_X, the population standard deviation, with s_X, the sample standard deviation, and use this to get an estimated standard error.

However, this approach is specific to the mean, and only works if the population distribution is approximately Normal or the sample is large.

A more general approach is to use the sample as an estimate of the entire population distribution (think of the sample histogram as an estimate of the population histogram) and then simulate repeated sampling from this estimated population. Now not only are we using the sample standard deviation, but we are also using information about the shape of the distribution (skew, "kurtosis", or fatness of the tails) to inform our estimate of the standard error.

To simulate repeatedly sampling from a population with the same histogram as the sample, we want to act as though there are many copies of each sample case — that is, the distribution is the same but the number of cases is much larger. We can achieve this effect by drawing samples *with replacement* from the sample itself. That is, each time we draw a case for our simulated sample, we keep that case in the pool so that it can be drawn again. This is as if there are many many copies of each case so that we never "use up" a value by drawing it.

To do sampling with replacement in R we can use the `resmaple()` function. Let's simulate drawing 5000 samples of size 60 with replacement from our original sample of size 60, and computing means for each one. This is called **bootstrap resampling** (based on the metaphor of "pulling ourselves up by our bootstraps", since we are seemingly circularly using one sample to assess the behavior of many samples)

```
### Use cache = TRUE
Bootstrap.means <- do(5000) *
    resample(Sample60, size = 60) %>% mean(~Area, data = .)
```

Exercise 3 Plot the means from the bootstrap sample. Where are they centered? Is this different than where the true sampling distribution is centered?

How does the variability compare?

4 Confidence Intervals

If we knew the true sampling distribution of the sample statistic (e.g., the sample mean), then the standard error would be the standard deviation of that distribution, and we could compute a 95% margin of error which is twice the standard error. When we only have one sample, we can estimate the standard error using the standard deviation of the distribution of bootstrap sample statistics (means, in this case).

```
estimated.standard.error <- sd(~result, data = Bootstrap.means)
```

Since the sample mean is within 2 SE of the population mean 95% of the time, we estimate that the population mean is within 2 SE of the sample mean:

```
CI.lower <- sample.mean - 2 * estimated.standard.error
CI.upper <- sample.mean + 2 * estimated.standard.error
CI.lower; CI.upper
```

Exercise 4 Write a sentence about the confidence interval you get, describing how to interpret it. What does it mean that the interval has a 95% confidence level?

Exercise 5 Determine whether, in this case, the true population mean does in fact fall in your interval. Everyone in the class will have a different intervals, since you each generated different samples. Draw your interval on the whiteboard so everyone can see how they vary. How many captured the true population mean? If you were to repeat this exercise many many times, what fraction of the time do you expect your interval to "miss" the true value? (Or, equivalently if many many students did the exercise, what proportion of them would "miss"?)

Notice that we are either underestimating or overestimating the true standard error by some amount, and we don't know which or by how much, so our margin of error could be too large or too small. But that's okay; we only require that our confidence interval capture the population parameter 95% of the time. As long as things average out, our interval-generating procedure will be valid.

Let's simulate drawing many samples and generating a confidence interval for each one. We could use bootstrap intervals, but this would take a long time, so we'll fall back on the analytic procedure of estimating the standard error as

$$\hat{SE}_{\bar{X}} = \frac{s_X}{\sqrt{n}}$$

```
## use cache = TRUE for this chunk if using Markdown
Sample.stats <- do(100) *
    sample(Ames, size = 60) %>% favstats(~Area, data = .)
```

Look at the data frame created — there should be one column for each of the sample statistics computed by `favstats()`, and one row for each of the 100 samples drawn.

Let's add two new statistics: the lower and upper boundaries of a 95% confidence interval.

```
Sample.stats <-
    mutate(Sample.stats,
           CI.lower = mean - 2 * sd / sqrt(n),
           CI.upper = mean + 2 * sd / sqrt(n))
```

(Notice that this time we are overwriting the original data frame by using the same name again.)

The `plot_ci()` function plots a set of confidence intervals, along with the true population mean, highlighting those that miss.

Call the function on your collection of sample statistics, specifying the true population mean in the `mu=` argument. Note that this function assumes that you have defined the new variables `CI.lower` and `CI.upper`, by those exact names, in your collection of sample statistics.

```
### The function is defined in a script on my website
source("http://colindawson.net/stat215/code/plot_ci.R")
plot_ci(Sample.stats, mu = mean(~Area, data = Ames))
```

Exercise 6 How many of the 100 intervals in your simulation missed? Is this what you expected?

5 Homework

5. Construct a bootstrap 95% confidence interval for the mean sale price in Ames using a sample of size 60. Construct two more intervals based on samples of size 30 and size 120, respectively. How do the intervals differ? Does the result make sense? Explain.
6. Construct a 95% confidence interval for the correlation between `Area` and `Price` the same way. Do you notice anything different about the bootstrap distribution of correlations, compared to the distribution of means?