Chapter 4 Lab A: Statistical Inference and the Sampling Distribution

Sampling from Millbrae, California

In this lab, we'll investigate the ways in which the estimates that we make based on a random sample of data can inform us about what the population might look like. We're interested in formulating a *sampling distribution* of our estimate in order to get a sense of how good of an estimate it might be.

The Data

The dataset that we'll be considering comes from the town of Millbrae, California, near San Francisco. The U.S Census Bureau has recorded information on all 20,718 residents of Millbrae, including age and household income (in thousands of dollars). All residents of Millbrae represent our statistical population. In this lab we would like to learn as much as we can about the residents by taking smaller samples from the full population. Let's load the data.

```
download.file("http://www.openintro.org/stat/data/millbrae.RData",
    destfile = "millbrae.RData")
load("millbrae.RData")
```

We see that two vectors are loaded into the workspace: `age`, which contains the ages of all 20,718 residents of Millbrae and `income`, which contains the incomes for all 500 households to which those residents belong. For now, we'll focus on `age`. Let's look at the distribution of ages in Millbrae by calculating some summary statistics and making a histogram.

```
summary(ages)
hist(ages)
```

Exercise 1 How would you describe this population distribution?

The Unknown Sampling 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 even impossible. Because of this, we often take a smaller sample survey of the population and use that to make educated guesses about the properties of the population.

If we were interested in estimating the mean age in Millbrae based on a sample, we can use the following command to survey the population.

```
samp1 <- sample(ages,75)
```

This command allows us to create a new vector called `samp1` that is a simple random sample of size 75 from the population vector `ages`. At a conceptual level, you can imagine randomly choosing 75 names from the Millbrae phonebook, calling them up, and recording their ages. You would be correct in objecting that the phonebook probably doesn't contain all of the residents and that there will almost certainly be people that don't pick up the phone or refuse to give their age. These are issues that can make gathering data very difficult and are a strong incentive to collect a high quality sample.

> **Exercise 2** How would you describe the distribution of this sample? How does it compare to the distribution of the population?

If we're interested in estimating the average age of all the residents in Millbrae, our best guess is going to be the sample mean from this simple random sample.

```
mean(samp1)
```

Depending which 75 residents you selected, your estimate could be a bit above or a bit below the true population mean of 42.29. But in general, the sample mean turns out to be a pretty good estimate of the average age, and we were able to get it by sampling less than 1% of the population.

> **Exercise 3** Take a second sample, also of size 75, and call it `samp2`. How does the mean of `samp2` compare with the mean of `samp1`? If we took a third sample of size 150, intuitively would you expect the sample mean to be a better or worse 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. This is what is captured by the *sampling distribution*. In this lab, because we have access to the population, we can build up the sampling distribution for the sample mean by repeating the above steps 5000 times.

```
sample.means <- rep(0, 5000)

for(i in 1:5000){
   samp <- sample(ages, 75)
   sample.means[i] <- mean(samp)
   }

hist(sample.means, freq = FALSE)
```

Here we rely on the computational ability of R to quickly take 5000 samples of size 75 from the population, compute each of those sample means, and store them in a vector called `sample.means`. Note that since there are multiple lines in the code that are part of the for loop, it would make your life much easier to first write this code in an R script and then run it from there.

Exercise 4 How many elements are there in `sample.means`? How would you describe this sampling distribution? On what value is it centered? Would you expect the distribution to change if we instead collected 50,000 sample means?

Interlude: The For Loop

Let's take a break from the statistics for a moment to let that last block of code sink in. You have just run your first for loop - a cornerstone of computer programming. The idea behind the for loop is *iteration*: it allows you to execute code as many times as you want without having to type out every iteration. In the case above, we wanted to iterate the two lines of code inside the curly braces that take a random sample of size 75 from `ages` then save the mean of that sample into the `sample.means` vector. Without the for loop, this would be painful:

```
sample.means <- rep(0, 5000)

samp <- sample(ages, 75)
sample.means[1] <- mean(samp)

samp <- sample(ages, 75)
sample.means[2] <- mean(samp)

samp <- sample(ages, 75)
sample.means[3] <- mean(samp)

samp <- sample(ages, 75)
sample.means[4] <- mean(samp)
```

and so on...

and so on...

and so on.

With the for loop, these thousands of lines of code are compressed into 4 lines.

```
sample.means <- rep(0, 5000)

for(i in 1:5000){
   samp <- sample(ages, 75)
   sample.means[i] <- mean(samp)
   print(i)
   }
```

Let's consider this code line by line to figure out what it does. In the first line we *initialized a vector*. In this case, we created a vector of 5000 zeros called `sample.means`. This vector will serve as empty box waiting to be filled by whatever output we produce in the *for loop*.

The second line calls the for loop itself. The syntax can be loosely read as, "for every element `i` from 1 to 5000, run the following lines of code".

The last component of the loop is whatever is inside the curly braces; the lines of code that we'll be iterating over. Here, on every loop, we take a random sample of size 75 from `ages`, take its mean, and store it as the i^{th} element of `sample.means`.

Notice that in addition to automatically iterating the code 5000 times, the for loop provided an object that is specific to each loop: the `i`. You can think of `i` as the counter that keeps track of which loop you're on. On the first pass, `i` takes the value of 1; on the second pass, it takes the value of 2; and so on, until the 5000^{th} and final loop, where it takes the value of 5000. In order to display that this is really happening we asked R to print `i` at each iteration so that. The line of code where we ask R to print `i`, `print(i)`, is optional, and is only used for displaying what's going on while the for loop is running.

The for loop allows us not just to run the code 5000 times, but to neatly package the results, element by element, back into the empty vector that we initialized at the outset.

> **Exercise 5** To make sure you understand what you've done in this loop, try running a smaller version. Initialize a vector of 100 zeros called `sample.means.small`. Run a loop that takes a sample of size 75 from `ages` and stores the sample mean in `sample.means.small`, but only iterate from 1 to 50. Print the output to your screen (type `sample.means.small` into the console and press enter). How many elements are there in this object called `sample.means.small`? What does each element represent? Do you notice something odd going on here? How would you fix this?

Approximating the Sampling Distribution

Mechanics aside, let's return to the reason we used a for loop: to compute a sampling distribution, specifically, this one.

```
hist(sample.means, freq = FALSE)
```

The sampling distribution that we computed tells us everything that we would hope for about the average age of the residents of Millbrae. Because the sample mean is an unbiased estimator, the sampling distribution is centered at the true average age of the the population and the spread of the distribution indicates how much variability is induced by sampling only 75 of the residents.

We computed the sampling distribution for mean age by drawing 5000 samples from the population and calculating 5000 sample means. This was only possible because we had access to the population. In most cases you don't (if you did, there would be no need to estimate!).

Therefore, you have only your single sample to rely upon ... that, and the Central Limit Theorem.

The Central Limit Theorem states that, under certain conditions, the sample mean follows a normal distribution. This allows us to make the inferential leap from our single sample to the full sampling distribution that describes every possible sample mean you might come across. But we need to look before we leap.

> **Exercise 6** Does `samp1` meet the conditions for the sample mean to be nearly normal, as described in section 4.2.3?

If the conditions are met, then we can find the approximate sampling distribution by plugging in our best estimate for the population mean and standard error: $\bar{x}$ and $s/\sqrt{n}$.

```
xbar <- mean(samp1)
se <- sd(samp1)/sqrt(75)
```

We can add a curve representing this approximation to our existing histogram using the command `lines`. The x-coordinates cover the range of the x in the histogram and the y-coordinates are the values that correspond to this particular normal distribution.

```
hist(sample.means)
lines(x = seq(33,52,.1), y = dnorm(seq(33,52,.1), mean = xbar, sd = se))
```

We can see that the line does a decent job of tracing the histogram that we derived from having access to the population. In this case, our approximation based on the CLT is a good one.

On Your Own

So far we have only focused on estimating the mean age of the residents of Millbrae. Now we'll try to estimate the mean household income.

1. Take a random sample of size 50 from `incomes`. Using this sample, what is your best point estimate of the population mean?
2. Check the conditions for the sampling distribution of $\bar{x}_{income}$ to be nearly normal.
3. Since you have access to the population, compute the sampling distribution for $\bar{x}_{income}$ by taking 5000 samples from the population of size 50 and computing 5000 sample means. Store these means in a vector called `sample.means50`. Describe the shape of this sampling distribution. Based on this sampling distribution, what would you guess the mean income of the residents of Millbrae to be?
4. Change your sample size from 50 to 150, then compute the sampling distribution using the same method as above and store these means in a new vector called `sample.means150`. 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 the mean income of the residents of Millbrae to be?

5. Based on their shape, which sampling distribution would you feel more comfortable approximating by the normal model?

6. Which sampling distribution has a smaller spread? If we're concerned with making estimates that are consistently close to the true value, is having a sampling distribution with a smaller spread more or less desirable?

7. What concepts from the textbook are covered in this lab? What concepts, if any, are not covered in the textbook? Have you seen these concepts elsewhere, e.g. lecture, discussion section, previous labs, or homework problems? Be specific in your answer.