Lab 4A: Foundations for statistical inference - Sampling distributions

Sampling from Ames, Iowa

In this lab, we will 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 is.

The Data

The data set that we'll be considering comes 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 statistical population. In this lab we would like to learn as much as we can about these home sales by taking smaller samples from the full population. Let's load the data.

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

load("ames.RData")
```

We see that there are quite a few variables in the data set, enough to do a very in-depth analysis. For this lab, though, we'll restrict our attention to just two of the variables: the above ground living area of the house in square feet (`Gr.Liv.Area`) and the sale price (`SalePrice`). To save some effort throughout the lab, let's assign these long vectors names to shorter ones.

```
area <- ames$Gr.Liv.Area

price <- ames$SalePrice
```

Let's look at the distribution of area in our population of home sales by calculating some summary statistics and making a histogram.

```
summary(area)

hist(area)
```

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 living area in Ames based on a sample, we can use the following command to survey the population.

```
samp1 <- sample(area, 50)
```

This command allows us to create a new vector called `samp1` that is a simple random sample of size 50 from the population vector `area`. At a conceptual level, you can imagine going into the full assessors database and pulling up the files on 50 random home sales. Working with these 50 files could be considerably simpler than working with all 2928 home sales.

> **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 living area in homes in Ames, our best guess is going to be the sample mean from this simple random sample.

```
mean(samp1)
```

Depending which 50 homes you selected, your estimate could be a bit above or a bit below the true population mean of 1499.69 square feet. But in general, 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 `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_means50 <- rep(0, 5000)

for (i in 1:5000) {
    samp <- sample(area, 50)
    sample_means50[i] <- mean(samp)
}

hist(sample_means50)
```

Here we rely on the computational ability of R to quickly take 5000 samples of size 50 from the population, calculate each of those sample means, and store them in a vector called `sample_means50`. Note that since the for loop involves several lines of code, 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_means50`? 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 50 from `area` then save the mean of that sample into the `sample_means50` vector. Without the for loop, this would be painful:

```
sample_means50 <- rep(0, 5000)

samp <- sample(area, 50)
sample_means50[1] <- mean(samp)

samp <- sample(area, 50)
sample_means50[2] <- mean(samp)

samp <- sample(area, 50)
sample_means50[3] <- mean(samp)

samp <- sample(area, 50)
sample_means50[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_means50 <- rep(0, 5000)

for (i in 1:5000) {
    samp <- sample(area, 50)
    sample_means50[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_means50`. This vector will serve as an 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 50 from `area`, take its mean, and store it as the i^{th} element of `sample_means50`.

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. This line of code 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 50 from `area` 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?

Sample Size and 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_means50)
```

The sampling distribution that we computed tells us everything that we would hope for 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 age of the the population and the spread of the distribution indicates how much variability is induced by sampling only 50 home sales.

To get a sense of the effect that sample size has on our distribution, let's build up two more sampling distributions: one based on a sample size of 10 and another based on a sample size of 100.

```
sample_means10 <- rep(0, 5000)
sample_means100 <- rep(0, 5000)

for (i in 1:5000) {
    samp <- sample(area, 10)
    sample_means10[i] <- mean(samp)
    samp <- sample(area, 100)
    sample_means100[i] <- mean(samp)
}
```

Here we're able to use a single for loop to build two distributions by adding additional lines inside the curly braces. Don't worry about the fact that `samp` is used for the name of two different objects. In the second command of the for loop, the mean of `samp` is saved to the relevant place in the vector `sample_means10`. With the mean saved, we're now free to overwrite the object `samp` with a new sample, this time of size 100. In general, anytime you create an object using a name that is already in use, the old object will get replaced with the new one.

To see the effect that different sample sizes have on the sampling distribution, plot the three distributions on top of one another.

```
par(mfrow = c(3, 1))
hist(sample_means10, breaks = 20, xlim = range(sample_means10))
```

```
hist(sample_means50, breaks = 20, xlim = range(sample_means10))
hist(sample_means100, breaks = 20, xlim = range(sample_means10))
```

The first command specifies that you'd like to subdivide the plotting area into 3 rows of plots and 1 column[†]. The second argument in the `hist` command allows you to control the number of bins used in constructing the histogram. The third argument ensures that all three histograms will be plotted on the same x-axis.

Exercise 6 As the sample size grew, what happened to the shape of the sampling distribution? What happened to the center? What about the spread?

On Your Own

So far, we have only focused on estimating the mean living area in homes in Ames. Now you'll try to estimate the mean home price.

1. Take a random sample of size 50 from `price`. Using this sample, what is your best point estimate of the population mean?
2. Since you have access to the population, compute the sampling distribution for $\bar{x}_{price}$ 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 home price to be? How does it compare to the true mean home price calculated from the full population?
3. 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 to be the mean sale price of homes in Ames?
4. 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?
5. 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.

[†]You may need to stretch your plotting window to accommodate the extra plots. To return to the default setting of plotting one plot at a time, run the following command:

```
par(mfrow = c(1, 1))
```