# Lab 5 - Sampling Distributions

In this lab, we investigate the ways in which the statistics from a random sample of data can serve as point estimates for population parameters. We're interested in formulating a *sampling distribution* of our estimate to learn about its properties, such as its distribution.

## The data

Every two years, the CDC conducts national surveys in schools to monitor and assess the six largest contributors to youth morbidity and mortality. These contributors include health risks, such as high body mass index, and risky behaviors, such as tobacco and alcohol use, drunk driving, and failure to use seat belts. In 2013, 47 states participated in this school-based survey titled the Youth Risk and Behavior Social Survey, yielding 13,583 respondents and 213 variables. A subset of this data set with no missing data for 16 selected variables is provided in the file yrbss2013.csv. You can access the complete survey and data documentation on the CDC website.

| Variable | description |
|---|---|
| age | Q1: How old are you? |
| gender | Q2: What is your sex? |
| height_m | calculated variable: height in meters |
| weight_kg | calculated variable: weight in kilograms |
| bmi | calculated variable: body mass index=height m/(weight kg)^2 |
| BMIPCT | calculated variable: BMI percentile for age and sex |
| seatbelt | Q9: How often do you wear a seat belt when riding in a car driven by someone else? |
| seatbelt2 | calculated variable: seatbelt never vs otherwise |
| ride_drunkdriver | Q10: During the past 30 days, have you ridden in a car or other vehicle driven by someone who had been drinking alcohol? |
| drive_drunk | Q11: During the past 30 days, how many times did you drive a car or other vehicle when you had been drinking alcohol? |
| drive_text | Q12: During the past 30 days, on how many days did you text or e-mail while driving a car or other vehicle? |
| carried_weapon | Q13: During the past 30 days, did you carry a weapon such as a gun, knife, or club? |
| unsafe_school | Q16: During the past 30 days, did you not go to school because you felt you would be unsafe at school or on your way to or from school? |
| bullied | Q24: During the past 12 months, have you ever been bullied on school property? |
| sad | Q26: During the past 12 months, did you ever feel so sad or hopeless almost every day for two weeks or more in a row that you stopped doing some usual activities? |
| days_smoke | Q33: During the past 30 days, on how many days did you smoke cigarettes? |
| days_drink | Q43: During the past 30 days, on how many days did you have at least one drink of alcohol? |

Let's load the data.

```
setwd("location of working directory")
yrbss <- read.csv("yrbss2013.csv", header = T)
```

# Sampling Distribution for Proportions

We'll consider a categorical variable in our data set, `bullied`, which focuses on whether students were bullied on the school property.

```
bullied <- factor(yrbss$bullied)
```

Note that you should generally use the variables inside the data set using code like: `yrbss$bullied`. However, since we will be using this variable frequently, we have created an object with the observations from the variable.

Let's look at the distribution of students bullied on the school property by calculating a few summary statistics and making a bar graph.

```
table(bullied)
prop.table(table(bullied))

# create bar plot with proportions
barplot(prop.table(table(bullied)), beside = T)
```

## The unknown sampling distribution

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

If we were interested in estimating the proportion of students bullied based on a sample, we could use the following command to survey the population.

```
samp1 <- sample(x = bullied, size = 10)
```

This command collects a simple random sample of size ten from the vector `bullied`, assigned to `samp1`. It is like going into the data set and picking ten random students. Working with these ten students would be considerably more straightforward than working with all 8482 students.

If we're interested in estimating the proportion of students bullied using the sample, our best single guess is the sample proportion.

```
prop.table(table(samp1))
```

Depending on which ten students you selected, your estimated proportion could be a bit above or below the true population proportion. In the true population proportion, `0.81 (81%)` students were not bullied and `0.19 (19%)` students were bullied, and in the sample proportion, `0.9 (90%)` students were not bullied and `0.1 (10%)` students were bullied.

Now let us take a second sample, with size 200, and call it `samp2`.

```
samp2 <- sample(x = bullied, size = 200)
prop.table(table(samp2))
```

By increasing the sample size, we obtain sample proportions (`'0.795' for not bullied and '0.205' for yes bullied`), which are closer in values to the true population proportions (`'0.81' for not bullied` `and '0.19' for yes bullied).` In general, the sample proportions turns out to be a pretty good estimate of the students bullied, and we were able to get it by sampling less than 3% of the population.

*(Do it yourself)* Take a third sample, also of size 200, and call it `samp3`. How do the proportions of `samp3` compare with the proportions of `samp2`? 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 proportions?

Not surprisingly, every time we take another random sample, we get different sample proportions. It's helpful to get a sense of just how much variability we should expect when estimating the population proportions this way. The distribution of sample proportions, called the *sampling distribution*, can help us understand this variability. In this lab, we can build up the sampling distribution for the sample proportions by repeating the above steps many times because we have access to the population.

Here we use R to take 500 samples of size ten from the population, calculate the proportions of each sample, and store each result in a vector called `sample_prop10`. In the next section, we'll review how these codes work.

```
sample_prop10 <- matrix(rep(NA, 500), nrow= 500, ncol = 2)

for(i in 1:500){
  samp <- sample(bullied, 10)
  sample_prop10[i,] <- prop.table(table(samp))
  }

barplot(colMeans(sample_prop10), names.arg = c('no', 'yes'), ylim = c(0,1))
```

*(Do it yourself)*: How many elements are there in sample_prop10? Would you expect the distribution to change if we collected 50,000 sample proportions instead?

## The for loop

Some of you may 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 ten from `bullied` then save the proportions of that sample into the `sample_prop10` matrix. Without the `for` loop, this would be painful:

```
sample_prop10 <- matrix(rep(NA, 500), nrow= 500, ncol = 2)

samp <- sample(bullied, 10)
sample_prop10[1] <- prop.table(table(samp))

samp <- sample(bullied, 10)
sample_prop10[2] <- prop.table(table(samp))

samp <- sample(bullied, 10)
sample_prop10[3] <- prop.table(table(samp))

samp <- sample(bullied, 10)
sample_prop10[4] <- prop.table(table(samp))
```

and so on...

With the for loop, these hundreds of lines of code are compressed into a handful of lines. We've added one extra line to the code below, which prints the variable `i` during each iteration of the `for` loop. Run this code.

```r
sample_prop10 <- matrix(rep(NA, 500), nrow= 500, ncol = 2)
#Creates an empty matrix of 500 rows and 2 columns

for(i in 1:500){
  print(paste0('i = ', i))
  # Prints the i-th value

  samp <- sample(bullied, 10)
  # Creates a vector with 10 samples from the "bullied" vector

  sample_prop10[i,] <- prop.table(table(samp))
  # Adds the proportions of samp to the sample_prop10 matrix.
  # The comma after 'i' enters proportion values into the 'i'-th row of the matrix
  }

barplot(colMeans(sample_prop10), names.arg = c('no', 'yes'), ylim = c(0,1))
# Creates a bar plot of the averaged proportion values.
```

Let's consider this code line by line to figure out what it does.

1. `sample_prop10 <- matrix(rep(NA, 500), nrow= 500, ncol = 2)` initializes an empty matrix of 500 rows with 2 columns. This matrix stores values generated within the `for` loop.
2. `for(i in 1:500){` calls the `for` loop itself. The syntax can be loosely read as, "for every element `i` from 1 to 500, run the following lines of code". You can think of `i` as the counter that keeps track of which loop you're on. Therefore, more precisely, the loop will run once when `i = 1`, then once when `i = 2`, and so on up to `i = 500`.The body of the `for` loop is the part inside the curly braces.
3. `print(paste0('i = ', i))` prints every i-th element in the for loop so `i = 1`, then once when `i = 2`, and so on up to `i = 500`.
4. `samp <- sample(bullied, 10)` uses the sample function to draw a sample of size ten from the `bullied` variable. Then it assigns this sample to a variable samp, which is then a vector with ten values.
5. `sample_prop10[i,] <- prop.table(table(samp))` computes the proportions of samp and saves the value as i-th row element of the matrix sample_prop10.
6. `}` indicates the end of the code to be repeated 500 times.
7. `barplot(colMeans(sample_prop10), names.arg = c('no', 'yes'))` produces a barplot of the averaged proportion values of sample_prop10.

The `for` loop allows us not just to run the code 500 times but to neatly package the results, element by element, into the empty matrix that we initialized at the outset.

*(Do it yourself)* To make sure you understand what you've done in this loop, try running a smaller version. Initialize a matrix of 100 zeros called `sample_prop_small`. Run a loop that takes a sample size 20 from `bullied` and stores the sample proportions in `sample_prop_small,` but only iterate from 1 to 100. Print the output to your screen. How many elements are there in this object called `sample_prop_small`? What does each element represent?