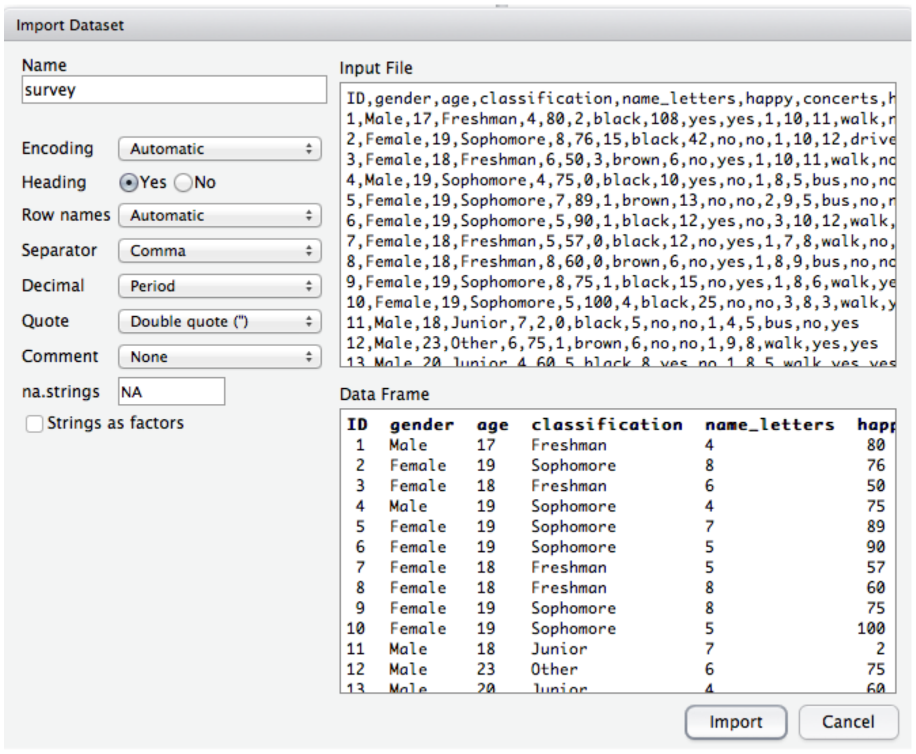
Foundations of Data Analysis

The University of Texas at Austin

R Tutorials: Week 7

Sampling Distributions

In this R tutorial, we're going to use R to simulate drawing many random samples from a population and then using the means of those samples to create a sampling distribution. So for this demonstration, I'm going to be using the student survey data set, which I can import. And we'll just shorten the name to be called "survey".



Name: survey

Encoding: Automatic
Heading: [✓ Yes] No
Row names: Automatic
Separator: Comma
Decimal: Period
Quote: Double quote (")
Comment: None
na.strings: NA
Strings as factors

Input File

```
ID,gender,age,classification,name_letters,happy,concerts,h
1,Male,17,Freshman,4,80,2,black,108,yes,yes,1,10,11,walk,r
2,Female,19,Sophomore,8,76,15,black,42,no,no,1,10,12,drive
3,Female,18,Freshman,6,50,3,brown,6,no,yes,1,10,11,walk,nc
4,Male,19,Sophomore,4,75,0,black,10,yes,no,1,8,5,bus,no,nc
5,Female,19,Sophomore,7,89,1,brown,13,no,no,2,9,5,bus,no,r
6,Female,19,Sophomore,5,90,1,black,12,yes,no,3,10,12,walk,
7,Female,18,Freshman,5,57,0,black,12,no,yes,1,7,8,walk,no,
8,Female,18,Freshman,8,60,0,brown,6,no,yes,1,8,9,bus,no,nc
9,Female,19,Sophomore,8,75,1,black,15,no,yes,1,8,6,walk,ye
10,Female,19,Sophomore,5,100,4,black,25,no,no,3,8,3,walk,y
11,Male,18,Junior,7,2,0,black,5,no,no,1,4,5,bus,no,yes
12,Male,23,Other,6,75,1,brown,6,no,no,1,9,8,walk,yes,yes
13,Male,20,Junior,4,60,5,black,8,yes,no,1,8,5,walk,yes,yes
```

Data Frame

```
ID  gender  age  classification  name_letters  happ
 1  Male    17   Freshman        4               80
 2  Female  19   Sophomore       8               76
 3  Female  18   Freshman        6               50
 4  Male    19   Sophomore       4               75
 5  Female  19   Sophomore       7               89
 6  Female  19   Sophomore       5               90
 7  Female  18   Freshman        5               57
 8  Female  18   Freshman        8               60
 9  Female  19   Sophomore       8               75
10  Female  19   Sophomore       5              100
11  Male    18   Junior          7                2
12  Male    23   Other           6               75
13  Male    20   Junior          4               60
```

Figure 1: Importing Data in RStudio.

Alternatively, we can import the dataset with the "**read.csv()**" function:

```r
survey <- read.csv("~/Desktop/SDSFoundations/StudentSurvey.csv")
```

So this is a survey we gave every single undergraduate in our Introductory Statistics course. And in this survey, we asked them various questions, some demographic questions, including their age, which is the variable we're going to look at now. And because we asked this of all of our undergraduates, we can think of this as the population of all undergraduates in our introductory statistics course.

So we're going to do is take this age variable, and then we're going to draw random samples from it and see what the distributions of those sample means looks like. So first, let's look at the distribution of our age variable. Let's calculate the mean and also, the standard deviation.

```r
mean(survey$age)
```

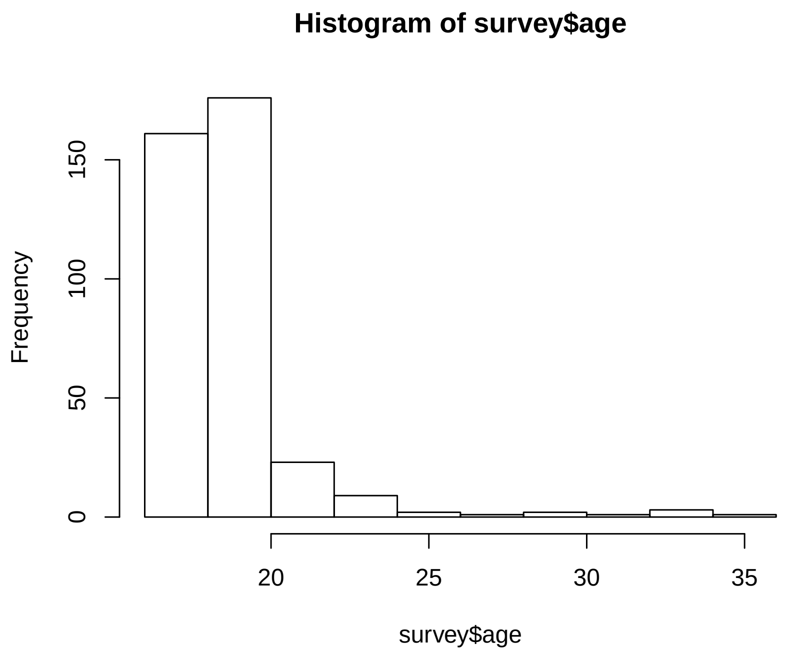
```
## [1] 19.25594
```

```r
sd(survey$age)
```

```
## [1] 2.269055
```

So we can see that the mean age of students in our class is about 19.3 years with a standard deviation of about 2.3 years. And if we look at the distribution of age, we can see a very positively skewed shape.

```r
hist(survey$age)
```



So most of the students are around 18 or 19 years old, but there are a few older students in the class. So this population distribution is clearly not normally distributed. Now we can randomly draw a sample from this vector age by using the sample function in R. All we have to give it is a vector of numbers, and then, after comma, tell it how big of a sample we actually want to draw with the Size Equals To option. So let's say we want to draw 30 students from our population at random and grab their ages. If we run this, we will simply be outputted a list of 30 randomly-selected ages.

```r
sample(survey$age, 30)
```

```
##  [1] 18 23 19 19 20 18 22 20 20 20 21 19 19 18 18 18 18 20 18 19 20 18 21
## [24] 18 18 18 20 18 20 18
```

Now keep in mind, if you run the sample function on your computer, you're likely going to get a different set of numbers than I just did. And that's because R randomly generates them every time. So if you rerun sample over and over again, you're going to get a new randomly-generated sample each time.

So in order to get a sampling distribution of sample means for this variable, we're going to have to draw a lot of these samples over and over again. And luckily, R can do that for us really quickly with something called a For Loop. Don't worry about having to create this For Loop on your own. You will be given the code in any assignment. But the first thing we want to start out doing is to create an empty vector to store all of our sample means. So I'm just going to create a vector called *myxbar*.

```
myxbar <- rep(NA, 1000)
```

And it's going to be an empty vector, basically containing a bunch of N/As, and it will have a length of 1,000. So we're going to basically draw 1,000 samples and store their means into this vector that's now empty called *myxbar*. Now, to randomly sample 1,000 times, we're going to use the **for()** loop. So I'm gonna say for I N, 1 through 1,000, do whatever's inside these brackets.

```
for (i in 1:1000) {

}
```

Basically, where R is going to do is start with 1, and wherever it finds an I, it's going to substitute 1 for that I. It's going to run through everything in the brackets, and then it's going to go up and do the same thing but now it's gonna be on 2. And then 3. And then 4, all the way up to 1,000. So I want to do this 1,000 times. I want to draw a sample– I'll just call it *mysamp* – that is a random sample from our age variable of size 30.

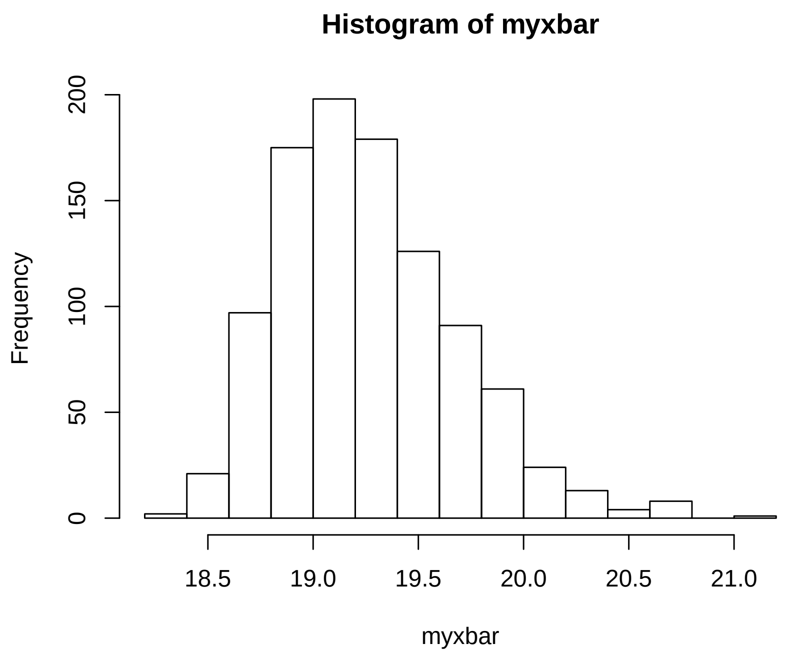
```
for (i in 1:1000) {
    mysamp <- sample(survey$age, size = 30)
}
```

Now, once I have that random sample, I want to calculate its mean and store it in this *myxbar* vector. So *myxbar*, on the Ith index, is going to hold the mean of *mysamp*. So in this first iteration, the first spot in *myxbar* is going to hold my first sample. Then, as it goes to the second iteration, it's going to go on to the second index and so on and so forth.

```
for (i in 1:1000) {
    mysamp <- sample(survey$age, size = 30)
    myxbar[i] <- mean(mysamp)
}
```

So let's go ahead and run this command. And then look and see what *myxbar* now holds. So if we make a histogram of *myxbar*, we're going to see the sampling distribution of means that we just ran.

```
hist(myxbar)
```



So we can see from the plot that, although this isn't exactly normally distributed, it's a whole lot more normal than our population was. And that is one property of the central limit theorem. We can also see that the means of all these samples aren't spread out very much. In our original population, we saw the spread go from about 18 to 35. Here, the sample means are much more constricted. They all fall between roughly 18.5 and 20.5 years, and that's the second property of the central limit theorem. And we can see from what we calculated before, the true mean of our population is 19.2 years, which would fall somewhat close to the center of the sampling distribution. So just to check that, let's find the mean of *myxbar*.

```
mean(myxbar)
```

```
## [1] 19.27913
```

And we see it's really, really close to the true population mean. They're both 19.25. Now we would expect, from the central limit theorem, the standard deviation of our sampling distribution, 0.387, to be close to the population standard deviation divided by the square root of our sample size, which in this case was 30.

```
sd(myxbar)
```

```
## [1] 0.4197469
```

```
sd(survey$age)/sqrt(30)
```

```
## [1] 0.4142708
```

And we can see that these aren't exactly the same but they're fairly close. Also, if you wanted to change the sample size of these random samples that you're pulling, you could keep the For Look the same and only change this argument here. If you want to draw samples of size, say 10, you could just change that one argument.

```r
for (i in 1:1000) {
    mysamp <- sample(survey$age, size = 10)
    myxbar[i] <- mean(mysamp)
}
```