Intro to Data Science - Lab 4

Week 4 - Sampling

```
# Enter your name here: Hendi Kushta
```

Please include nice comments.

Instructions:

Run the necessary code on your own instance of R-Studio.

Attribution statement: (choose only one and delete the rest)

```
# 1. I did this lab assignment by myself, with help from the book and the professor.
```

A key focus of this week is how to make inferences about populations based on samples. The essential logic lies in comparing a single instance of a statistic such as a sample mean to a distribution of such values. The comparison can lead to one of two conclusions the sample statistic is either extreme or not extreme. But what are the thresholds for making this kind of judgment call (i.e., whether a value is extreme or not)? This activity explores that question. **The problem is this:** You receive a sample containing the ages of 30 students. You are wondering whether this sample is a group of **undergraduates** (mean age = 20 years) or **graduates** (mean age = 25 years). To answer this question, you must compare the mean of the sample you receive to a distribution of means from the population. The following fragment of R code begins the solution:

```
set.seed(2) # make sure that we get the same results for randomization
sampleSize <- 30
studentPop <- rnorm(20000,mean=20,sd=3)
undergrads <- sample(studentPop,size=sampleSize,replace=TRUE)
grads <- rnorm(sampleSize,mean=25,sd=3)
if (runif(1)>0.5) { testSample <- grads } else { testSample <- undergrads }
mean(testSample)
```

After you run this code, the variable ** testSample ** will contain either a sample of undergrads or a sample of grads. The line before last flips a coin by generating one value from a **uniform distribution** (by default the distribution covers 0 to 1) and comparing it to 0.5. The question you must answer with additional code is: Which is it, grad or undergrad? Here are the steps that will help you finish the job:

1. Copy the code above and annotate it with line-by-line commentary. In other words, you must explain what
   each of the seven lines of code above actually do! You will have to lookup the meaning of some commands.

```
set.seed(2) # make sure that we get the same results for randomization
sampleSize <- 30 # assign sample size to 30.
studentPop <- rnorm(20000,mean=20,sd=3) # Generating 20000 random numbers with mean 20
# and deviation of 3 and storing it in studentPop
undergrads <- sample(studentPop,size=sampleSize,replace=TRUE) # generating sample of size
# 30 from studentPop and storing it in undergrads
grads <- rnorm(sampleSize,mean=25,sd=3) # Generating 30 random
# numbers with mean around 25 and deviation of 3 and storing it in grads
if (runif(1)>0.5) { testSample <- grads } else { testSample <- undergrads } # runif
# function used to generate random deviates
```

```
# contains a sample either from grads or undergrads
mean(testSample) # find the mean of testSample
```

```
## [1] 24.89729
```

2. Generate 10 samples from the ** undergrads ** dataset.

```
sample(undergrads, size = 10, replace = TRUE)
```

```
##  [1] 14.13022 25.57921 25.57921 25.92212 13.99717 19.87687 21.48056 24.82276
##  [9] 25.57921 18.73168
```

3. Generate 10 new samples and take the mean of that sample

```
newSample <- sample(undergrads, size = 10, replace = TRUE)
newSample
```

```
##  [1] 21.05056 25.92212 15.74496 16.13685 23.48922 14.13022 19.70899 20.07569
##  [9] 16.13685 20.07569
```

```
mean(newSample)
```

```
## [1] 19.24712
```

4. Repeat this process 3 times (i.e., generate a sample and take the mean 3 times, using the replicate function).

```
anotherMean <- replicate(3, mean(sample(undergrads, size = 10, replace = TRUE)))
anotherMean
```

```
## [1] 19.94657 18.68297 18.99601
```

5. Generate a list of sample means from the population called ** undergrads ** How many sample means should you generate? Really, you can create any number that you want hundreds, thousands, whatever but I suggest for ease of inspection that you generate just 100 means. That is a pretty small number, but it makes it easy to think about percentiles
and ranks.

```
UnderGrad_SampleMeans <- replicate(100,mean(sample(undergrads, size=30, replace=TRUE)),
                                   simplify=TRUE)
UnderGrad_SampleMeans
```

```
##   [1] 20.16517 18.48848 20.09595 19.62763 21.12619 19.27297 18.20366 19.59489
##   [9] 19.31100 18.92050 20.87433 19.07601 19.72588 19.58355 19.10306 20.72901
##  [17] 19.88727 20.14191 20.68962 19.17023 19.27992 20.52954 18.80988 20.16121
##  [25] 19.48345 18.77768 19.12045 19.01316 18.48120 20.02377 18.64783 19.27291
##  [33] 19.84487 19.49827 20.01711 19.77535 20.66725 19.88999 19.01036 20.04011
##  [41] 19.38777 18.10251 20.04040 19.04230 19.49116 19.51246 19.44658 20.13167
##  [49] 20.36412 20.05961 19.76280 19.56519 19.43774 19.59471 19.63865 19.06506
##  [57] 20.10158 19.48047 19.25562 19.81261 20.21025 18.72290 19.56629 19.68147
##  [65] 19.88227 20.10363 18.89122 19.65326 18.89982 19.28170 19.41080 17.97810
##  [73] 18.97894 19.02924 19.47843 18.74232 19.35892 20.07010 20.20226 19.28419
##  [81] 19.41253 18.96814 18.88952 19.74187 19.81454 18.52465 20.04321 18.86444
##  [89] 19.75736 20.25337 20.14645 18.77723 19.10154 19.92150 19.29946 18.71680
##  [97] 20.12484 19.68628 19.39574 19.81919
```

6. Once you have your list of sample means generated from **undergrads**, the trick is to compare **mean(testSample)** to that list of sample means and see where it falls. Is it in the middle of the pack? Far out toward one end? Here is one hint that will help you: In chapter 7, the **quantile()** command is

used to generate percentiles based on thresholds of 2.5% and 97.5%. Those are the thresholds we want, and the **quantile()** command will help you create them.

```
compare <- quantile(UnderGrad_SampleMeans, probs=c(0.025,0.975))
compare
```

```
##     2.5%    97.5%
## 18.33549 20.71030
```

7. Your code should have a **print()** statement that should say either, Sample mean is extreme, or, Sample mean is not extreme.

```
if ((compare[1] < mean(testSample)) && (mean(testSample) < compare[2]) )
  {print("Sample mean is not extreme")} else {print("Sample mean is extreme")}
```

```
## [1] "Sample mean is extreme"
```

8. Add a comment stating if you think the **testSample** are undergrad students. Explain why or why not.

```
# It is a low chance that testSample are undergrad students,
# because there is only 2.5% chance that the students can have
# a mean more than 20.6 which is 97.5%.
```

9. Repeat the same analysis to see if the **testSample** are grad students.

```
Grad_SampleMeans <- replicate(100,mean(sample(grads, size=30, replace=TRUE)),
                              simplify=TRUE)
Grad_SampleMeans
```

```
##   [1] 24.03972 24.34677 24.75612 24.71380 26.38457 24.68281 25.30414 24.38346
##   [9] 25.82247 25.08307 25.03880 24.91226 24.77307 25.97735 24.62804 25.49867
##  [17] 24.45882 25.34217 24.45823 24.92486 23.93638 25.89585 24.30644 24.76934
##  [25] 24.69198 25.38711 25.44763 24.66237 24.05573 24.89453 24.78763 24.29726
##  [33] 24.42121 24.76073 25.95555 25.62601 25.10137 23.97804 24.09608 26.14913
##  [41] 24.24392 25.13440 24.92069 24.64552 24.90491 25.46981 24.09442 26.06678
##  [49] 24.79430 25.03386 25.63245 24.09521 25.09949 24.50681 24.87755 24.54981
##  [57] 25.19682 24.84125 25.11304 24.98587 24.48824 25.24386 24.83184 24.78963
##  [65] 25.06779 25.17466 24.98648 24.77333 25.21425 24.52669 24.90504 25.11903
##  [73] 24.20014 24.91389 24.64829 25.06648 24.67390 26.37534 24.84686 24.17902
##  [81] 25.36863 24.29126 24.91120 24.93826 24.65713 25.46094 24.61825 24.67593
##  [89] 24.14110 25.67976 24.65775 25.03179 25.81302 26.01757 24.60725 25.62692
##  [97] 24.60702 24.94220 25.11416 25.19620
```

```
newCompare <- quantile(Grad_SampleMeans, probs=c(0.025,0.975))
newCompare
```

```
##     2.5%    97.5%
## 24.04732 26.11002
```

```
if ((newCompare[1] < mean(testSample)) && (mean(testSample) < newCompare[2]) )
  {print("Sample mean is not extreme")} else {print("Sample mean is extreme")}
```

```
## [1] "Sample mean is not extreme"
```

```
# Since the mean of testSample falls within 95% of the grad sample,
# testSample can be a grad student sample.
```