IST687 – Sampling: Decision Making Thresholds

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)
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. Annotate the code above with line-by-line commentary. To get full credit on this assignment you must demonstrate a clear understanding of what the six lines of code above actually do! You will have to lookup the meaning of some commands.
2. The next line of code should generate a list of sample means from the population called "studentPop." Very similar code to accomplish this appears right in the middle of Chapter 7. 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.
3. Once you have your list of sample means generated from studentPop, 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.
4. Your code should end with a print() statement that could say either, "Sample mean is extreme," or, "Sample mean is not extreme."
5. Please submit both the output of your runs and the R code.

Learning Goals for this activity:

A. Generate random numbers in a normal distribution and assign a variable name.
B. Understand and use a conditional statement.
C. Reason about percentiles.
D. Reason about distributions of sample means.
E. Use R code to report the results of a test.

Essential Guide for All IST687 Activities (appears at the end of all activity guides)

1. All IST687 activities work on what some people call a "constructivist learning" model. By developing a product on your own, testing it to find flaws, improving it, and comparing your solution to the solutions of other people, you can obtain a deeper understanding of a problem, the tools that might solve that problem, and a range of solutions that those tools may facilitate. The constructivist model only works to the extent that the student/learner has the drive to explore a problem, be frustrated, fail, try again, possibly fail again, and finally push through to a satisfactory level of understanding.
2. Each IST687 activity builds on skills and knowledge developed in the previous activities, so your success across the span of the course depends at each stage on your investment in earlier stages. Take the time to experiment, play, try new things, practice, improve, and learn as much as possible. These investments will pay off later.
3. Using the expertise of others, the Internet, and other sources of information is not only acceptable - it is expected. You must ***always, always, always*** give credit to your sources. For example, if you find a chunk of code from r-bloggers.com that helps you with developing a solution, by all means borrow that chunk of code, but make sure to use a comment in your code to document the source of the borrowed code chunk. The discussion boards in the learning management system have been setup to encourage appropriate sharing of knowledge and wisdom among peers. Feel free to ask a question or pose a solution on these boards.
4. Building on the previous point, when submitting code as your solution to the activity, the comments matter at least as much, if not more than the code itself. A good rule of thumb is that every line of code should have a comment, and every meaningful block of code should be preceded by a comment block that is just about as long as the code itself. As noted above, you can use comments to give proper credit to your sources and you can use comments to identify your submission as your own.