MTH 324, Lab #5, Fall 2022 Name ____

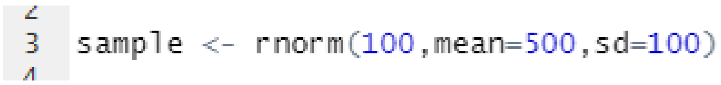
Instructions: Follow along with the tutorial portion of the lab. Replicate the code examples in R on your own, along with the demonstration. Then use those examples as a model to answer the questions/perform the tasks that follow. Copy and paste the results of your code to answer questions where directed. Submit your response file and the code used (both for the tutorial and part two). Your code file and your lab response file should each include your name inside.

Simulations

We will build on our random sampling from the last lab our r(dist) functions can be used to generate random samples. If we calculate statistics on those samples, we can then create a graph of our sampling distributions. To do this for many samples, your coding skills will come in handy as we'll need to write a for-loop to collect our sample statistics.

Let's start with the normal distribution. SAT tests are designed to have a mean of 500 and a standard deviation of 100 for individual tests (such as just the math portion).

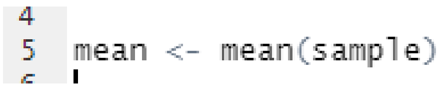
Let's look at the elements of our loop first.



```r
sample <- rnorm(100,mean=500,sd=100)
```

First, we generate a random sample using rnorm() since test scores are normally distributed. Save that to a vector.

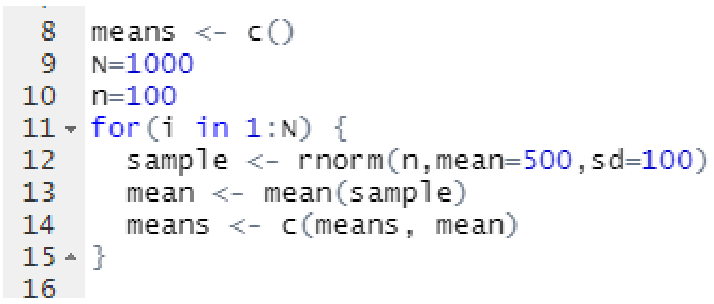
Then we find the sample mean and save that to a vector.



```r
mean <- mean(sample)
```

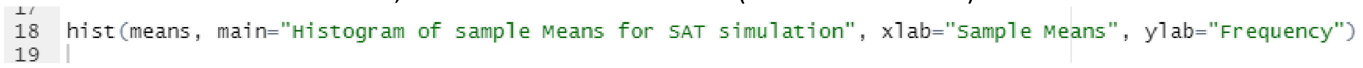
For the loop, we'll need to collect these means in a new vector.

Here is the full loop.

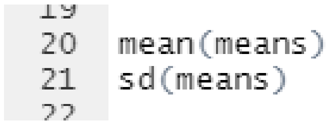


```r
means <- c()
N=1000
n=100
for(i in 1:N) {
  sample <- rnorm(n,mean=500,sd=100)
  mean <- mean(sample)
  means <- c(means, mean)
}
```

We initialize the vector of means (where we'll collect our statistics) as an empty vector. I've declared the sample size and number of samples as variables to make them easy to change. The last line is something that R allows us to do to concatenate vectors, or add additional elements. At the end of the loop, the vector means will contain our 1000 sample means. We'll plot this as a histogram now. We can also calculate the mean of the means, and the standard deviation (our standard error).



```r
hist(means, main="Histogram of sample Means for SAT simulation", xlab="Sample Means", ylab="Frequency")
```

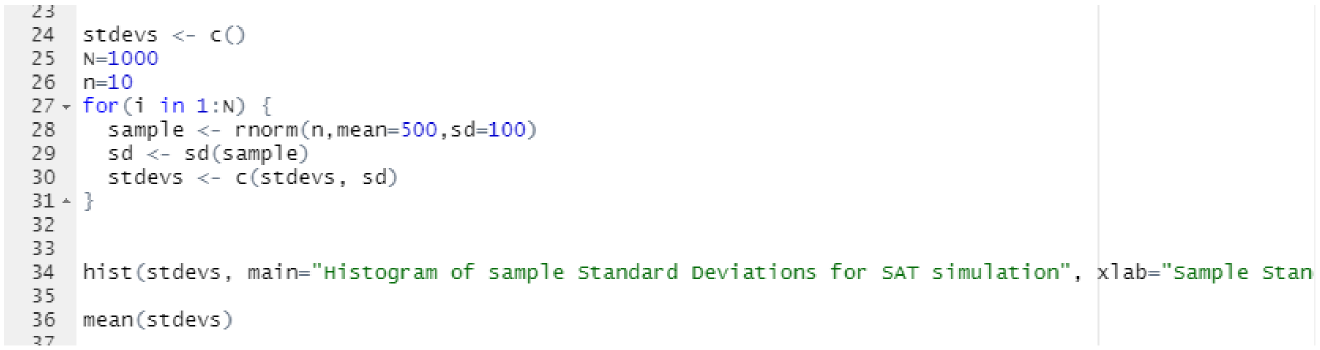


```r
mean(means)
sd(means)
```

Our mean of the means in the example I got was 500.3671, and the standard error was 9.91039. If you run the code yourself, since I didn't set the random seed, you should get similar but non-identical values. The mean of the means should be centered around the population mean (which we set to 500), and the standard error is $\frac{\sigma}{\sqrt{n}} = \frac{100}{\sqrt{100}} = 10$. And this is very close to what we got.

We can change the sample size from 100 to some larger or smaller value to see how that changes our standard error. We could take more samples, maybe 10,000, and get more precise estimates of both the mean and standard error.

Standard deviation statistics don't follow a normal distribution, but instead follows a χ^2 distribution (however, depending on the degrees of freedom, it may not appear very skewed, and if the sample size is large, the Central Limit Theorem will take over). We can adjust our loop to measure and plot those.

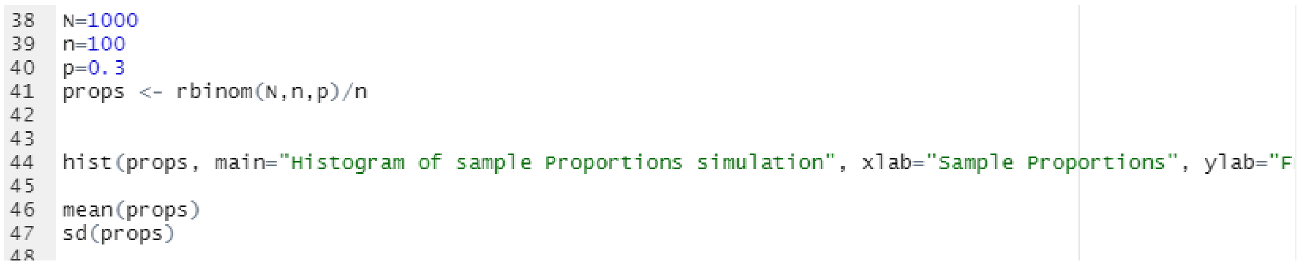


```r
stdevs <- c()
N=1000
n=10
for(i in 1:N) {
  sample <- rnorm(n,mean=500,sd=100)
  sd <- sd(sample)
  stdevs <- c(stdevs, sd)
}


hist(stdevs, main="Histogram of sample Standard Deviations for SAT simulation", xlab="Sample Stan

mean(stdevs)
```

The plotting labels are somewhat cut off, but use appropriate labels.

The mean of the standard deviations does a good job of approximating our population standard deviation. For my data, this came out to be 99.71429.

To do a binomial, we don't need the loop since the model is already counting successes.

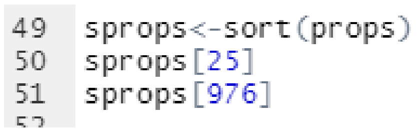


```r
N=1000
n=100
p=0.3
props <- rbinom(N,n,p)/n


hist(props, main="Histogram of sample Proportions simulation", xlab="Sample Proportions", ylab="F

mean(props)
sd(props)
```

The mean of the sampling distribution for a proportion should be the value of p from the population. My sample produced 0.30084. The standard error, recall, is approximately $SE = \sqrt{\frac{p(1-p)}{n}} = \sqrt{\frac{0.3(0.7)}{100}} \approx 0 \approx .045825$ …. My data produced 0.04603057.

What if I wanted to do a confidence interval based on the simulated sampling distribution?

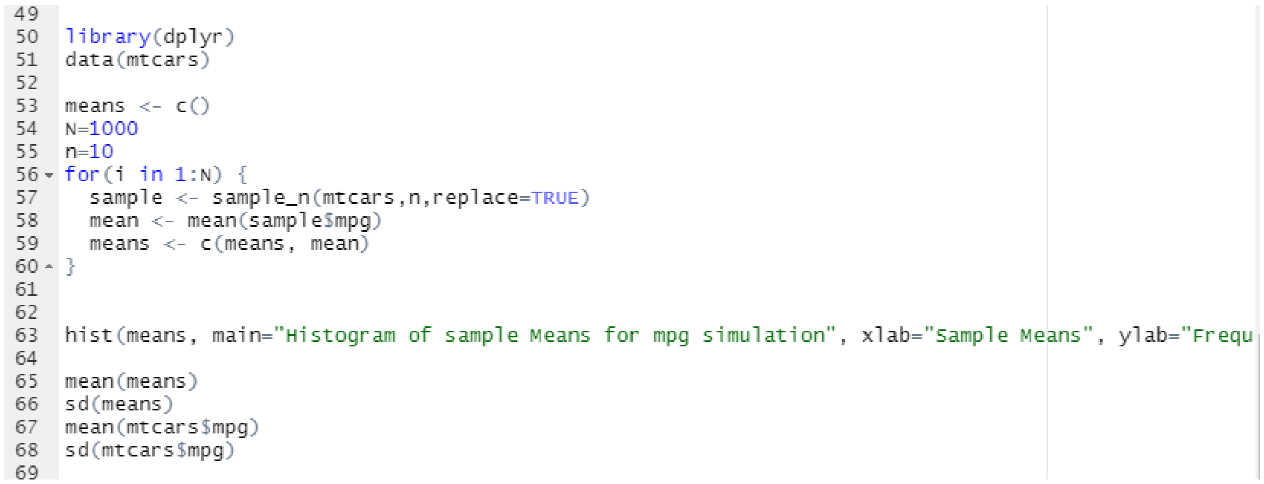
First, sort your data. Then, if we want a 95% confidence interval, we want to leave out 2.5% on each end. That corresponds to 25 values from 1000 observations. Get the 25th value, and the 976th value (the end of the bottom 25, and the bottom of the top 25).



```
sprops<-sort(props)
sprops[25]
sprops[976]
```

Use those values as the ends of your confidence interval. For my data that's (0.21, and 0.39). While proportions have a formula we can construct a confidence interval with, some statistics like the median doesn't. We can use simulations to estimate what those would have to be.

We can use a similar trick to simulate sampling from an existing dataset (we'll do more with this later on in the course when we look at bootstrapping).



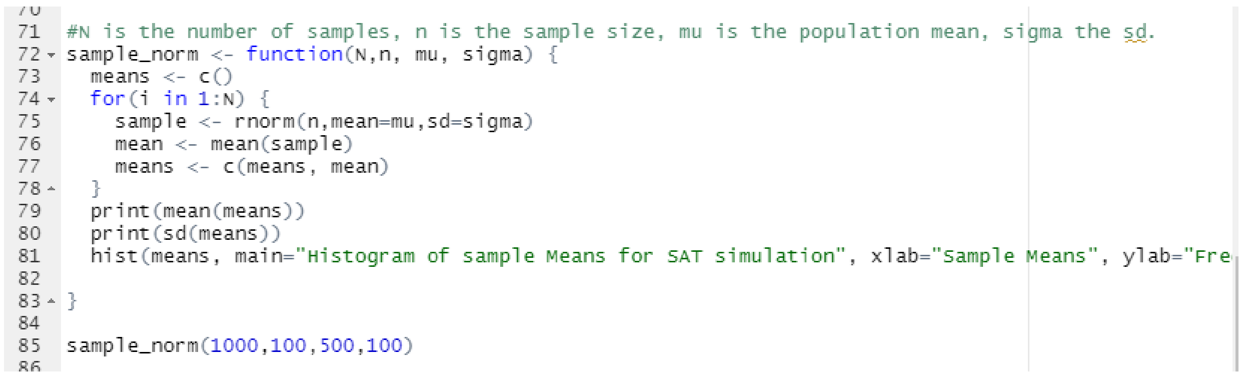
```

library(dplyr)
data(mtcars)

means <- c()
N=1000
n=10
for(i in 1:N) {
  sample <- sample_n(mtcars,n,replace=TRUE)
  mean <- mean(sample$mpg)
  means <- c(means, mean)
}


hist(means, main="Histogram of sample Means for mpg simulation", xlab="Sample Means", ylab="Frequ

mean(means)
sd(means)
mean(mtcars$mpg)
sd(mtcars$mpg)

```

You'll need to install the dplyr library before loading it.

Make a histogram to look at the sampling distribution. We can compare the mean and standard deviation of our sampling distribution against the mean and standard deviation of the data in the dataframe which we are treating as our population.

Our samples produce a mean (of means) to be 20.09745 while the mean of the original data is 20.09062. The standard error is 1.885518. We compare that to the standard error from the central limit theorem. For that, we need the standard deviation of the data, which is 6.026948. Our sample sizes were $n = 10$, so we get $\frac{\sigma}{\sqrt{n}} = \frac{6.026948}{\sqrt{10}} \approx 1.905888$.

We could define a function to do these operations for us.



```
#N is the number of samples, n is the sample size, mu is the population mean, sigma the sd.
sample_norm <- function(N,n, mu, sigma) {
  means <- c()
  for(i in 1:N) {
    sample <- rnorm(n,mean=mu,sd=sigma)
    mean <- mean(sample)
    means <- c(means, mean)
  }
  print(mean(means))
  print(sd(means))
  hist(means, main="Histogram of sample Means for SAT simulation", xlab="Sample Means", ylab="Fre

}

sample_norm(1000,100,500,100)
```

A good function should have the comments inside the function, but I've put them outside so that it's easier to see the structure of the function in R. This example just repeats our first sample, and outputs the mean and standard deviation of our sampling distribution.

The advantage here is that I only have to change the inputs to do another run with new parameters.

Tasks

1. Use the function we created for the normal distribution to look at the sampling distribution of heights of men (mean = 70 inches, standard deviation of 3.5 inches) with sample sizes of $n = 10\,, 50, 100$. Compare the mean of means, and the standard error to the predictions of the central limit theorem. What happens to the sampling distribution as the sample size increases? Plot all three distributions with the same axis limits. Paste the graphs below. Do 10,000 samples in each case.

2. Sample a Poisson distribution with a mean of 3 with a sample size of 5. Plot the distribution of the mean and paste it below. What is the shape of the distribution? Use your distribution to find a 95% confidence interval.

3. For this problem you will need the library "sn" installed for the skew-normal distribution. This distribution has a skewness parameter alpha that makes the distribution right-skewed if positive and left-skewed if negative (and looks like the standard normal distribution if alpha is 0). You can ignore the other parameters. (A reference for the package is linked below.) Draw a distribution (for a single sample, i.e. n=1) for some positive value of alpha. Then sample from that distribution with n=10. Calculate the median of the distribution and plot the sampling distribution of the median. Likewise for the mean of the same data. Repeat this for a negative value of alpha. Paste all 6 graphs below. Use at least 1000 samples in each case. What do you notice about the medians, and their standard deviations/standard errors? How do the mean and standard deviation compare to each other? Which standard error is larger? The mean or the median?

4. Sample from the mtcars data set, with sample sizes of 20. Calculate the range of the sample and plot that distribution. Paste the graph below. Do you notice anything different about the distribution compared to our other examples?

In our next lab session, you'll have time to work on your tutorial.

References:

1. Discovering Statistics Using R. Andy Field, Jeremy Miles, Zoe Field. (2012)
2. https://book.stat420.org/applied_statistics.pdf
3. https://scholarworks.montana.edu/xmlui/handle/1/2999
4. https://www.rstudio.com/resources/cheatsheets/
5. https://www.w3schools.com/r/r_for_loop.asp
6. https://www.geeksforgeeks.org/take-random-samples-from-a-data-frame-in-r-programming-sample_n-function/
7. https://stat.ethz.ch/R-manual/R-devel/library/base/html/sort.html
8. https://www.tutorialspoint.com/r/r_functions.htm
9. https://cran.r-project.org/web/packages/sn/sn.pdf