R Homework #5: Having Fun With Statistics

Luke Clement

In this assignment, you will get to stretch your statistics, functions, and functional programming muscles! Below there a variety of questions that will give you experience in making and using functions to investigate statistical concepts. These tasks will be your first baby steps into simulation, which is an essential component of the modeling and data science process.

Question 1: The Bootstrap

The bootstrap is a set of statistical methods that uses re-sampling to build up statistical theory "by the bootstraps". Traditional statistical inference that you learn in an introductory statistics course relies upon a theoretical sampling distribution, which describes the behavior of a statistic using probability. For the sample mean $\bar{x}$ and the sample proportion $\hat{p}$, the central limit theorem provides us a strong statistical result to rely on. But what about the sample standard deviation or the sample median? They don't have the theory to back up assuming that the sampling distribution follows a normal distribution. That's where the bootstrap comes in! The bootstrap basically builds up its own version of a sampling distribution, treating the original sample as the population. The bootstrap is really quite simple! Here are the steps:

1. Take the original sample, which has a sample size of n, and treat it like the population.
2. Take a sample of size n, with replacement, from the new population (original sample)
3. Calculate the statistic for that sample
4. Repeat steps 2 and 3 for a large number of times (10000 or more).
5. Create a histogram of the statistics.

The resulting histogram is called the bootstrap distribution. We can use that distribution to create a confidence interval or conduct a hypothesis test.

Question Starts Here:

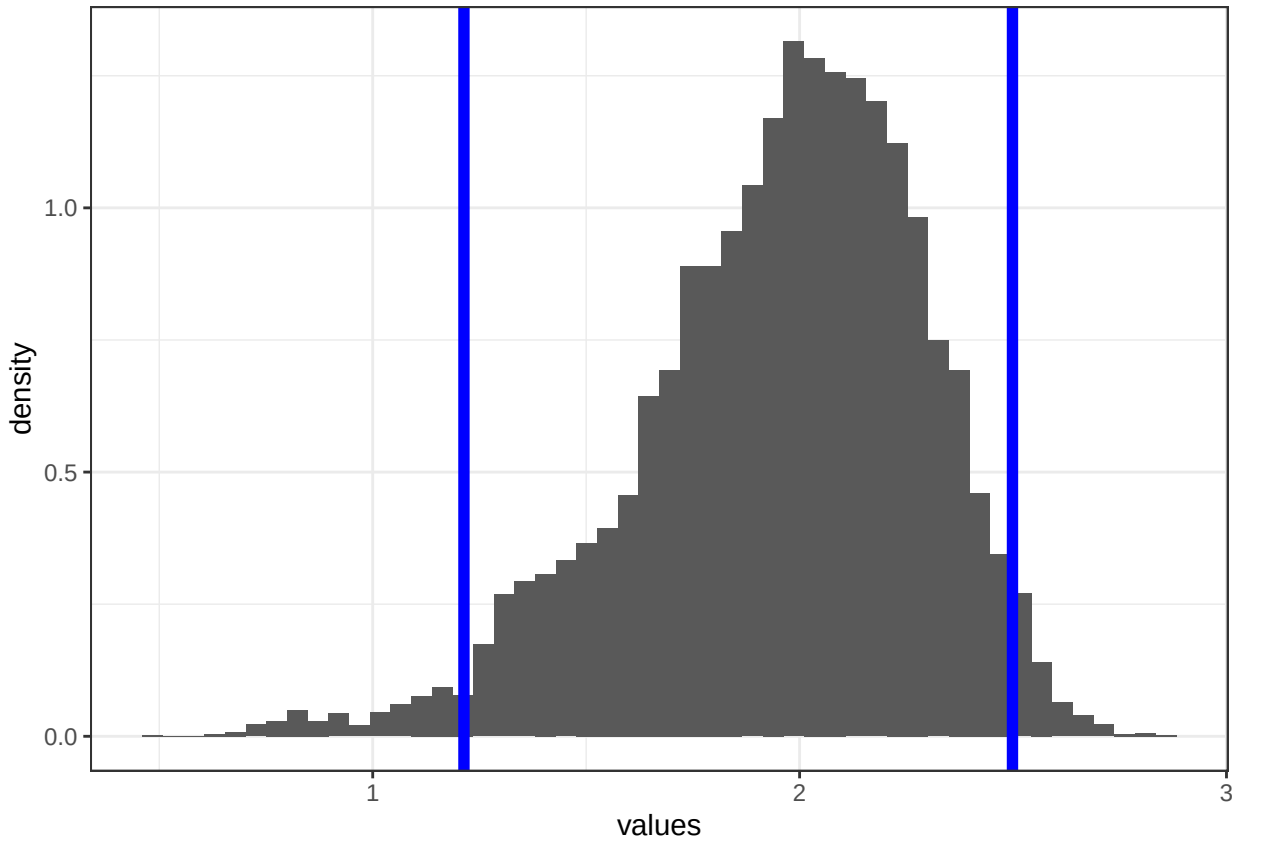
For this question, we are going to create a 95% bootstrap confidence interval for the sample standard deviation!!

The following steps are what you need to complete in order to create the proper interval. A general hint is that you should try to think about how to do this once, and then generalize.

1. Create 10000 samples, of size n, from the original population `samp` using `rerun()` and `sample()`. Remember to sample with replacement, `replace = TRUE` inside `sample()`, and make sure each sample is of the same size as the population, `n = 25`.
2. Find the standard deviation of each dataset using the `map_dbl()` function and the `sd()` function inside `map_dbl()`.
3. The bounds of a 95% bootstrap confidence interval are found by taking the 2.5th percentile and the 97.5th percentile of the standard deviations found in step 2. To find percentiles, you use the `quantile()` function.
4. To finish the question off, plot a histogram of the standard deviations to see the bootstrap distribution.

```
samp <- rexp(20, .3)
samples <- rerun(10000,sample(samp,length(samp), replace = TRUE))
samples <- map_dbl(samples, sd)

confid95 <- quantile(samples,probs = c(0.025,.975))
```

```
tb <- tibble(values = samples)

ggplot(tb,aes(x = values,..density..)) + geom_histogram(bins = 50) +
  geom_vline(xintercept=confid95[1],lwd=2,colour="blue") +
  geom_vline(xintercept=confid95[2],lwd=2,colour="blue")
```




```
confid95
```

```
##     2.5%    97.5%
## 1.213677 2.498688
```

Question 2: A Coverage Simulation

In the last question, I asked you to make a 95% confidence interval. What does the 95% part of the confidence interval mean? This is one of the most challenging parts of a confidence interval for people in an intro stat class. Remember that the goal of a confidence interval is to estimate the true value of a characteristic of the population, called a parameter. The interpretation that makes the most sense for us is the following: "If I were to take many samples of the same size from the population, and compute a confidence interval for each sample using the same method, then 95% of the intervals should contain the true value of the parameter". This is called the coverage interpretation of a confidence interval. Coverage for a confidence interval is defined as the proportion of times the intervals contained the truth. Did you ever wonder if the interval actually lives up to the theory? In other words, does the method we use actually create intervals that contain the truth 95% of the time? In theory, yes they do! In practice, sometimes not so much...

The traditional confidence interval used to estimate the population proportion has notoriously bad coverage

behavior! As a reminder, to estimate a population proportion p, we use the interval

$$\hat{p} \pm z_{\alpha/2} \times \sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$$

where $\hat{p}$ is the sample proportion, n is the sample size, and $z_{\alpha/2}$ is a critical value of the standard normal distribution based on the confidence level. The sample proportion $\hat{p}$ is called the point estimate and $\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$ is called the standard error.

Question Starts Here:

In order to evaluate the coverage behavior of an interval in general, the following steps need to be taken:

1. Fix a sample size n and some true value for the parameter.
2. Simulate many samples from the population with the same n and parameter value.
3. For each sample, construct the confidence interval.
4. Check whether or not the interval contains the true parameter value.
5. Find the proportion of intervals that contained the truth.

I want you to do this for the confidence interval for p described above! To do so, complete the following steps:

1. Create a function called `z_int()` that takes in the data, which will be in a series of 0's and 1's (corresponding to no and yes), and a specific confidence level called `conf`. You'll want to output a vector of length 2 with the endpoints of the interval. I have started the function below and calculated the critical value because it is the most challenging part.

2. The data for this type of problem is a series of successes and failures. We will encode 0 as a failure and 1 as a success. The way that we can simulate a series of successes and failures that have a common probability of success p is with a binomial distribution. To be specific, if I wanted simulate a sample of 25 successes and failures with the probability of success being 0.25, I would write `rbinom(n = 25, 1, p = 0.25)`. I want you to simulate 10000 samples of size $n = 25$ from a binomial distribution with probability of success $p = 0.25$. You'll want to use the `rerun()` function that you've used in previous homeworks. The result should be a list of vectors that contain a bunch of 0's and 1's.

3. Compute the confidence interval for each of the samples. This will entail mapping the interval function to each of the samples using the `map` function.

4. This is the tricky part! We need to check whether or not the intervals actually contain the true value of the parameter. Again, we will want to use purrr's functionals to do this! There are a few different ways to complete this part! You can create a separate function that checks if the true parameter is in the interval or you can use purrr's anonymous functions with the `map_lgl()` function. For instance, if I wanted to check if `a <- 0.25` is inside the interval `x <- c(.22, .36)`, I would run the code `x[1] <= a && a <= x[2]`.

5. Lastly, you need to calculate the proportion of intervals that contain the true value of the parameter. After completing step 4, you should have a list or vector of true's and false's. To find the proportion of true's, you just need to take the mean of the logical vectors.

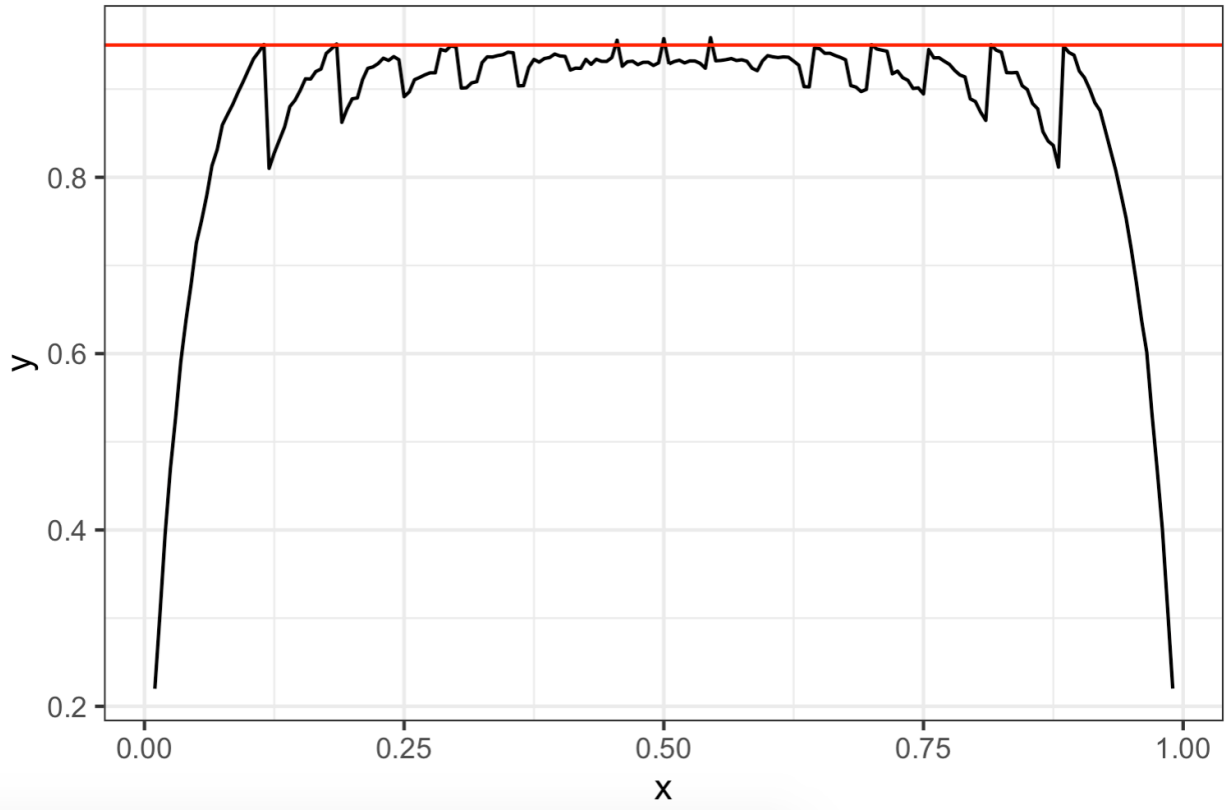
At the end, you should have one number between 0 and 1. That number is the actual coverage of the interval at that point. Theoretically, it should be near 0.95, but it will probably closer to .90!

```
z_int <- function(data, conf = 0.95) {
  crit_val <- qnorm(conf + (1 - conf)/2)
  mean_d <- mean(data)

  margin <- crit_val*sqrt(mean_d*(1-mean_d)/25)
  low <- mean_d - margin
  high <- mean_d + margin
```

```
  return(c(low,high))
}

  check1 <- (function(x) x[1] <= .25 && .25 <= x[2])

  bino_dist <- rerun(10000,rbinom(n = 25, 1, p = 0.25)) # binomial distribution
  bino_dist <- map(bino_dist, z_int)
  bino_dist2 <- map_lgl(bino_dist, check1)

  mean(bino_dist2)
```

```
## [1] 0.8987
```

Extra Credit: We did the simulation for fixed n and fixed p, but what happens to the behavior when you vary p? To answer this, I want you to apply the simulation you just completed above to a fine mesh of probabilities (you can keep the number of simulations and the sample size the same). I've placed the fine mesh in the code chunk below! After finding the coverage probability for each p, I want you to plot the results as a line, as well as a horizontal line at the nominal 95% coverage point. Your graph should be close to the image below:




```
z_int <- function(data, conf = 0.95) {
  crit_val <- qnorm(conf + (1 - conf)/2)
  mean_d <- mean(data)

  margin <- crit_val*sqrt(mean_d*(1-mean_d)/25)
  low <- mean_d - margin
  high <- mean_d + margin
```

```
  return(c(low,high))
}

values <- seq(from = .01, to = .99, by = .005)

checked_values  <- function(values){

  check2 <- (function(x) x[1] <= values && values <= x[2])

  b_d <- rerun(10000,rbinom(n = 25, 1, p = values)) # binomial distribution
  b_d <- map(b_d, z_int)
  b_d2 <- map_lgl(b_d, check2)

  return(mean(b_d2))
}

p_changes <- map(values, checked_values)
p_changes <- as.numeric(p_changes)

df <- tibble(x = values, y = p_changes)

ggplot(df,aes(x, y)) + geom_line(lwd=0.7)  +
  geom_hline(yintercept=.95,lwd=1,colour="red")
```