BACS HW4

109062710

March 26, 2021

Question 1

Given the critical DOI score that Google uses to detect malicious apps (-3.7), what is the probability that a randomly chosen app from Google's app store will turn off the Verify security feature? (report a precise decimal fraction, not a percentage)

this is an answer

Assuming there were ~2.2 million apps when the article was written, what number of apps on the Play Store did Google expect would maliciously turn off the Verify feature once installed?

this is an answer

Question 2

A. The Null distribution of t-values:

Before we dive in the questions down below, let's explore to see how the dataset looks like.

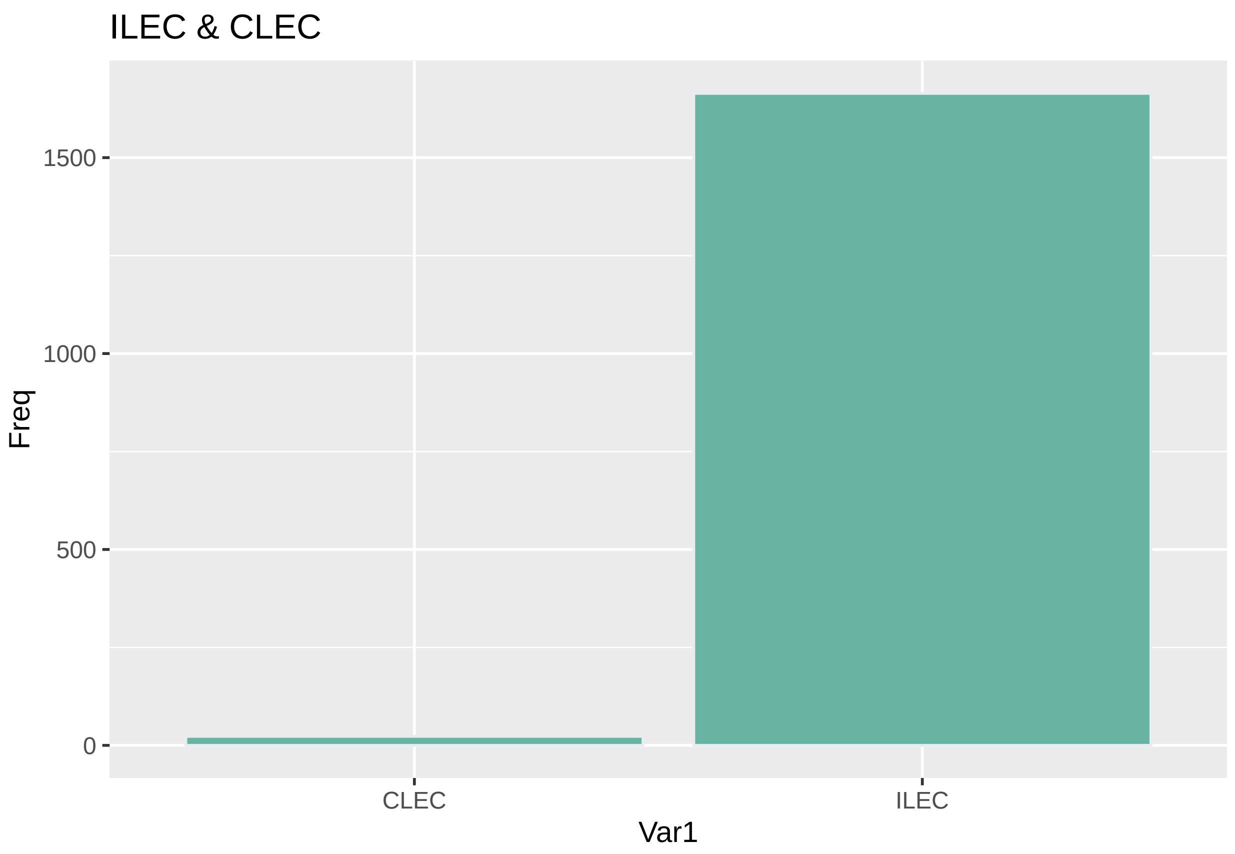
```
##
## CLEC ILEC
##   23 1664
```

```
## max_time -> 191.6
## min_time -> 0.
```

There are two targets in the data set:

- Competitive Local Exchange Carrier (CLEC) is a company that is characterized by providing local telephone service. It is those companies that have rented space from the Incumbent Local Exchange Carrier.
- Incumbent Local Exchange Carrier (ILEC) is companies that are known to provide an alternative service to the ILEC within its territory.

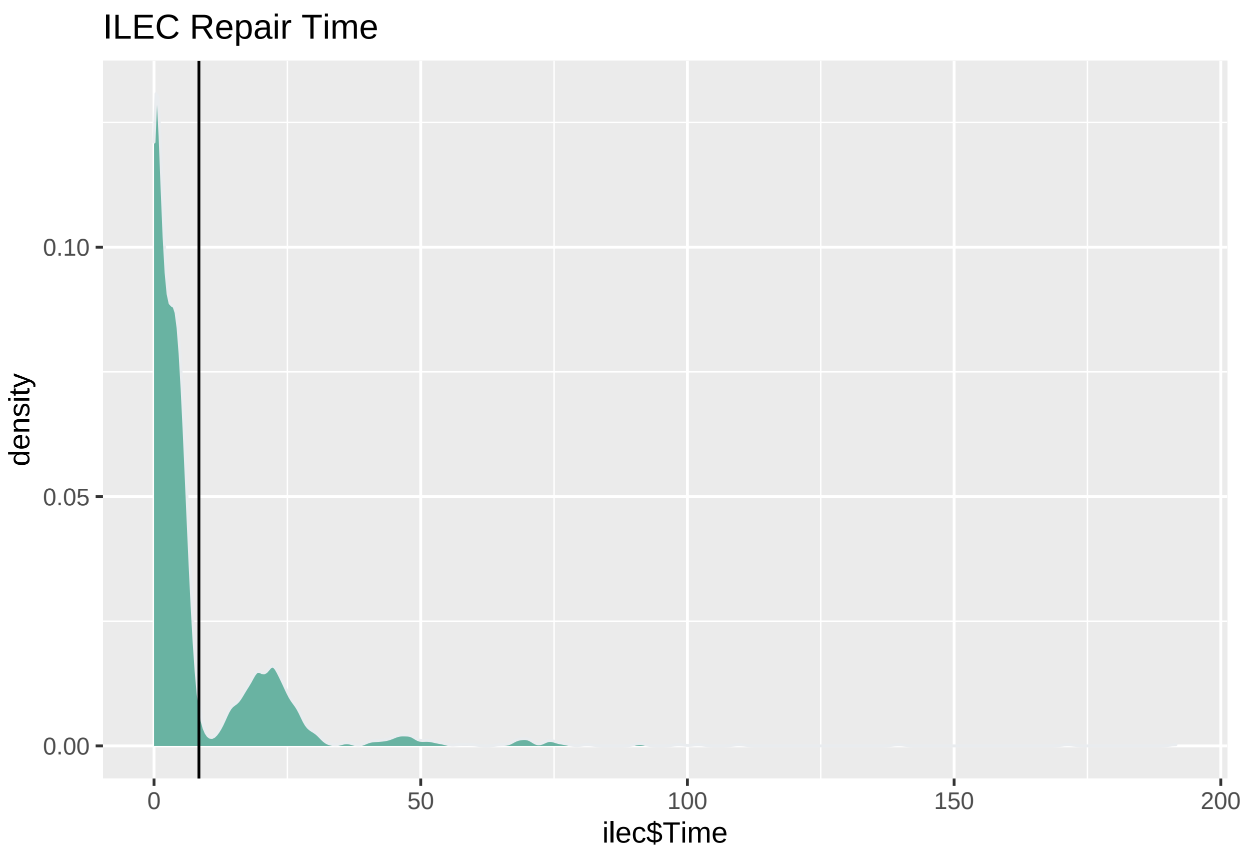
```
counts = as.data.frame(table(verizon["Group"]))
ggplot(data=counts, aes(x=Var1, y=Freq)) +
  geom_bar(
    stat="identity",
    fill="#69b3a2",
    color="#e9ecef"
  ) +
  ggtitle("ILEC & CLEC")
```



1. Visualize the distribution of Verizon's repair times, marking the mean with a vertical line

In this part, I am going to split the data set into two parts, ILEC and CLEC.

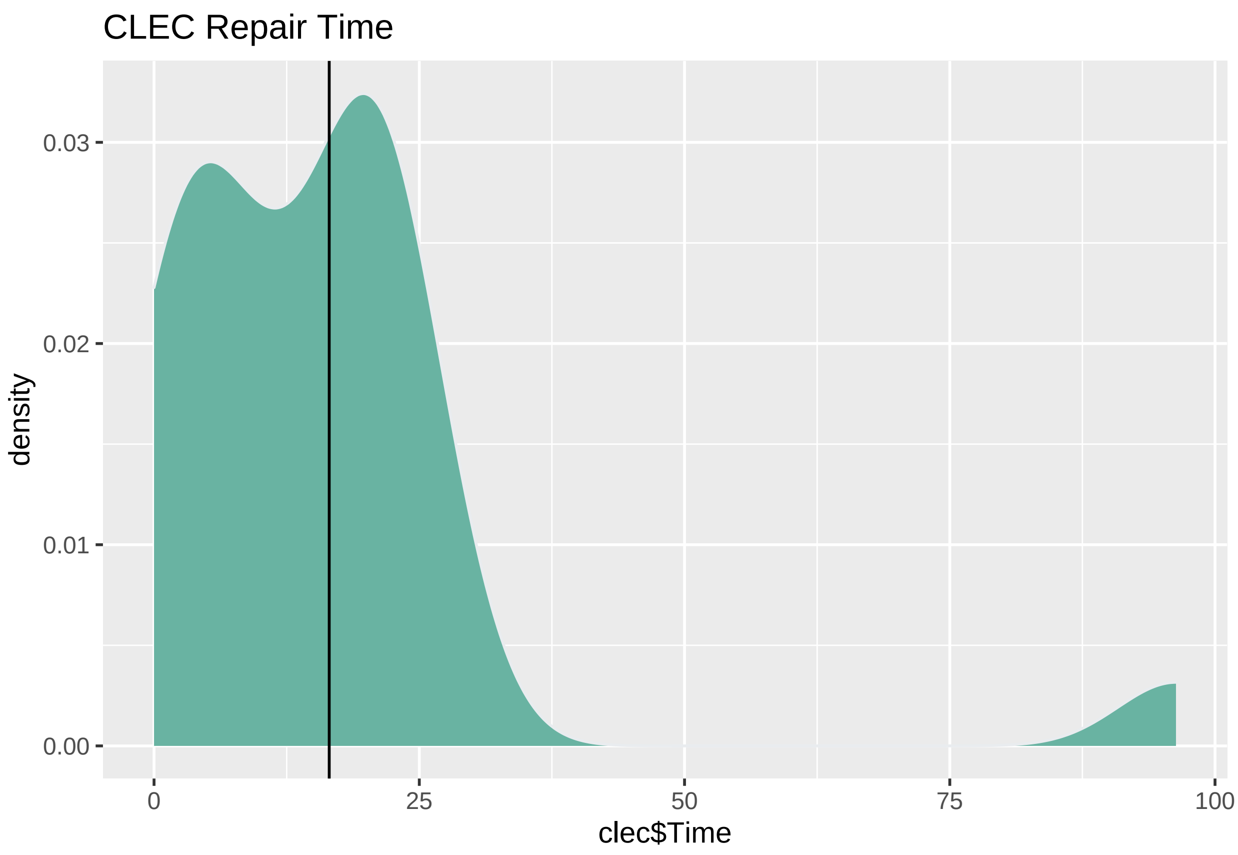
```
ilec <- as.data.frame(verizon[verizon$Group == "ILEC", ])

mean <- mean(ilec$Time)

ggplot(mapping = aes(ilec$Time)) +
  geom_density(
    fill="#69b3a2",
    color="#e9ecef"
  ) +
  ggtitle("ILEC Repair Time") +
  coord_cartesian(xlim = c(0, max(ilec$Time))) +
  geom_vline(xintercept = mean)
```



ILEC data set information:

```
##      Number of observation in ILEC -> 1664
##      ILEC minimum repair time -> 0
##      ILEC average repair time -> 8.41161057692308
##      ILEC maximum repair time -> 191.6
```

```
clec <- as.data.frame(verizon[verizon$Group == "CLEC", ])

mean <- mean(clec$Time)

ggplot(mapping = aes(clec$Time)) +
  geom_density(
    fill="#69b3a2",
    color="#e9ecef"
  ) +
  ggtitle("CLEC Repair Time") +
  coord_cartesian(xlim = c(0, max(clec$Time))) +
  geom_vline(xintercept = mean)
```



CLEC data set information:

```
##      Number of observation in CLEC -> 23
##      CLEC minimum repair time -> 0
##      CLEC average repair time -> 16.5091304347826
##      CLEC maximum repair time -> 96.32
```

2. Given what PUC wishes to test, how would you write the hypothesis? (not graded)

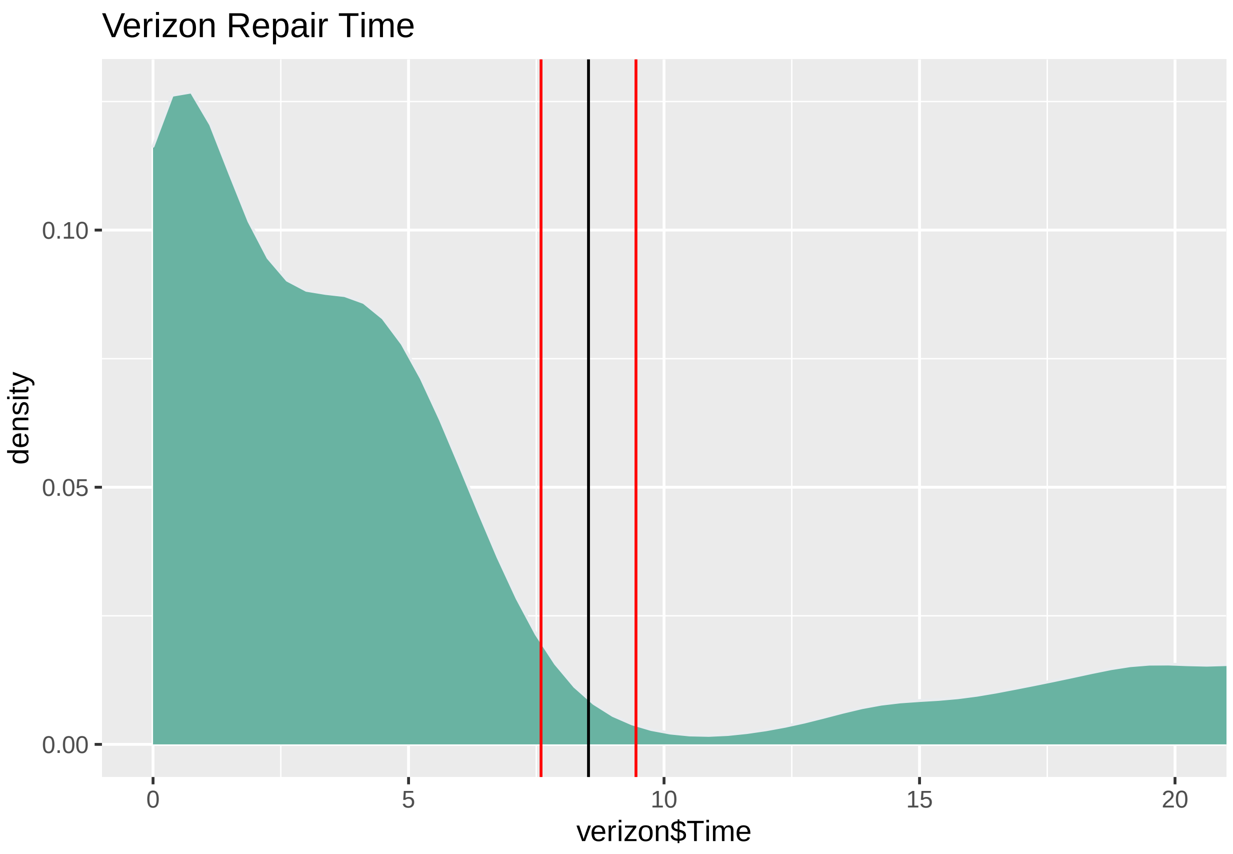
3. Estimate the population mean, and the 99% confidence interval (CI) of this estimate

```
population_mean <- mean(verizon$Time)
population_sd <- sd(verizon$Time)
population_sderr <- population_sd / sqrt(nrow(verizon))

ci99_low <- population_mean - population_sderr * 2.58
ci99_high <- population_mean + population_sderr * 2.58
```

```
glue("
  The population mean is {population_mean}.
  The 99% CI is between {ci99_low} and {ci99_high}.
")
```

```
##   The population mean is 8.52200948429164.
##   The 99% CI is between 7.59307342589326 and 9.45094554269003.
```

```r
mean <- mean(verizon$Time)

ggplot(mapping = aes(verizon$Time)) +
  geom_density(
    fill="#69b3a2",
    color="#e9ecef"
  ) +
  ggtitle("Verizon Repair Time") +
  coord_cartesian(xlim = c(0, 20)) +
  geom_vline(xintercept = mean) +
  geom_vline(xintercept = ci99_low, col="red") +
  geom_vline(xintercept = ci99_high, col="red")
```



4. Using the traditional statistical testing methods we saw in class, find the t-statistic and p-value of the test

T-statistic

```r
t <- (population_mean - 7.6) / population_sderr
glue(t)
```

```
## 2.56076233446444
```

P-value

```
df <- nrow(verizon) - 1
p <- 1 - pt(t, df)
glue(p)
```

```
## 0.00526534229428899
```

5. Briefly describe how these values relate to the Null distribution of t (not graded)

6. What is your conclusion about the advertising claim from this t-statistic, and why?

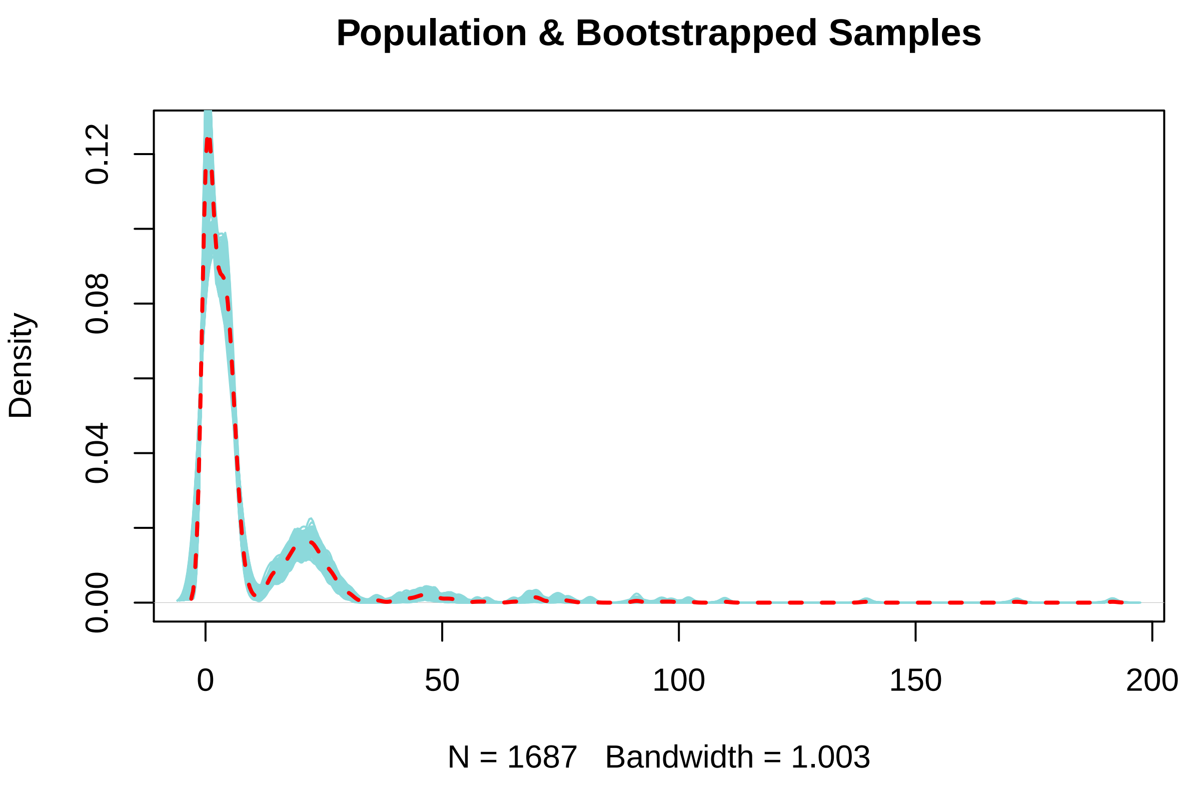
The t-statistics or t-value measures the size of the difference relative to the variation in your sample data. Put another way, t-value is simply the calculated difference represented in units of standard error. The greater the magnitude of t-value, the greater the evidence against the null hypothesis.

In this case, we have t-value which is far away from 0. This means that what Verizon's repair time needs more than Verizon has advertised.

B. Let's use bootstrapping on the sample data to examine this problem:

1. Bootstrapped Percentile: Estimate the bootstrapped 99% CI of the mean

```
bootstrapped_samples <- replicate(1000, sample(verizon$Time, nrow(verizon), replace=TRUE))

plot(
  density(verizon$Time),
  main="Population & Bootstrapped Samples"
)

plot_sample_and_get_mean <- function(sample) {
  lines(
    density(sample),
    col="#8cd9db"
  )
  return(mean(sample))
}

sample_means <- apply(bootstrapped_samples, 2, FUN=plot_sample_and_get_mean)

lines(density(verizon$Time), col="red", lwd=2, lty="dashed")
```



```r
average_sample_mean <- mean(sample_means)
sample_mean_sderror <- sd(sample_means) / (length(sample_means)^0.5)
mean_ci99_low <- average_sample_mean - 2.58 * sample_mean_sderror
mean_ci99_high <- average_sample_mean + 2.58 * sample_mean_sderror
```

The 99% CI of the mean is between 8.5055176 and 8.5628925.

2. Bootstrapped Difference of Means: What is the 99% CI of the bootstrapped difference between the population mean and the hypothesized mean?

```r
diff <- sample_means - 7.6
diff_means <- mean(abs(diff))
diff_sderror <- sd(diff) / (length(diff)^0.5)
diff_ci99_low <- diff_means - 2.58 * diff_sderror
diff_ci99_high <- diff_means + 2.58 * diff_sderror
```

The 99% CI of the mean is between 0.9071753 and 0.9645501.

3. Bootstrapped t-Interval: What is 99% CI of the bootstrapped t-statistic?

```r
# population_sderr <- population_sd / sqrt(nrow(verizon))
# t <- (population_mean - 7.6) / population_sderr

compute_t <- function(sample) {
  sample_sderr <- sd(sample) / sqrt(length(sample))
  return (
    (mean(sample) - 7.6) / sample_sderr
  )
}

sample_t <- apply(bootstrapped_samples, 2, FUN=compute_t)
average_sample_t <- mean(sample_t)
sample_t_sderr <- sd(sample_t) / (length(diff)^0.5)
t_ci99_low <- average_sample_t - 2.58 * sample_t_sderr
t_ci99_high <- average_sample_t + 2.58 * sample_t_sderr
```

The 99% CI of the bootstrapped t-statistic is between 2.4817647 and 2.6209986.

4. Plot separate distributions of all three bootstraps above (for 2 and 3 make sure to include zero on the x-axis)

C. Do the four methods (traditional test, bootstrapped percentile, bootstrapped difference of means, bootstrapped t-Interval) agree with each other on the test?