1. When conducting the work of Lab 11, we conducted the test that uses the Central Limit Theorem even though the sample size was "small" (i.e., $n < 30$). It turns out, that how "far off" the t-test is can be computed using a first-order Edgeworth approximation for the error. Below, we will do this for the the further observations.

   (a) Boos and Hughes-Oliver (2000) note that

   $$P(T \leq t) \approx F_Z(t) + \underbrace{\frac{\text{skew}}{\sqrt{n}} \frac{(2t^2+1)}{6} f_Z(t)}_{\text{error}},$$

   where $f_Z(\cdot)$ and $F_Z(\cdot)$ are the Gaussian PDF and CDF and skew is the skewness of the data. What is the potential error in the computation of the p-value when testing $H_0 : \mu_X = 0; H_a : \mu_X < 0$ using the zebra finch further data?

```
library(tidyverse)
library(xtable)
library(VGAM)
library(e1071)
library(pwr)
library(effectsize)
### Part A ### -> using further observations
dat.finches <- read.csv("zebrafinches.csv")

further <- dat.finches$further
mu0 <- 0
t.further <- t.test(further,
                    mu = mu0,
                    alternative = "less")
t.stat.further <- t.further$statistic
n <- length(further)
skewness <- skewness(further)
fz <- dnorm(t.stat.further)
Fz <- pnorm(t.stat.further)

(edgeworth.approx.error <- (skewness/sqrt(n)) *
    (((2*(t.stat.further)^2 + 1)/6)*(fz)))

##             t
## -1.226006e-13

# Potential Error in the computation of the p-value is -1.226006e-13.
probability <- Fz + edgeworth.approx.error
```

   We found the Edgeworth approximation error to be -1.226006e-13 which is very close to 0. This means that the t-test is not "far-off" due to the potential error in the computation of the p-value when testing the hypotheses for the further data being so low.

   (b) Compute the error for t statistics from -10 to 10 and plot a line that shows the error across t. Continue to use the skewness and the sample size for the zebra finch further data.

```
### Part B ###
t.values <- seq(-10,10, length = 1000)
fz2 <- dnorm(t.values)

error.vals <- (skewness/sqrt(n)) * (((2*(t.values)^2 + 1)/6)*(fz2))
```

```r
error.tvals <- tibble(
  t = t.values,
  error = error.vals
)
(summary(error.tvals))
```

```
##        t           error
##  Min.   :-10   Min.   :-2.602e-02
##  1st Qu.: -5   1st Qu.:-8.061e-03
##  Median :  0   Median :-2.560e-06
##  Mean   :  0   Mean   :-5.172e-03
##  3rd Qu.:  5   3rd Qu.: 0.000e+00
##  Max.   : 10   Max.   : 0.000e+00
```

```r
error.t.plot <- ggplot()+
  geom_line(data = error.tvals,
            aes(x = t, y = error),
            color = "lightblue")+
  theme_bw()+
  labs(title = "Edgeworth Approximation Error",
       x = "T Values (-10,10)",
       y = "Error")
```

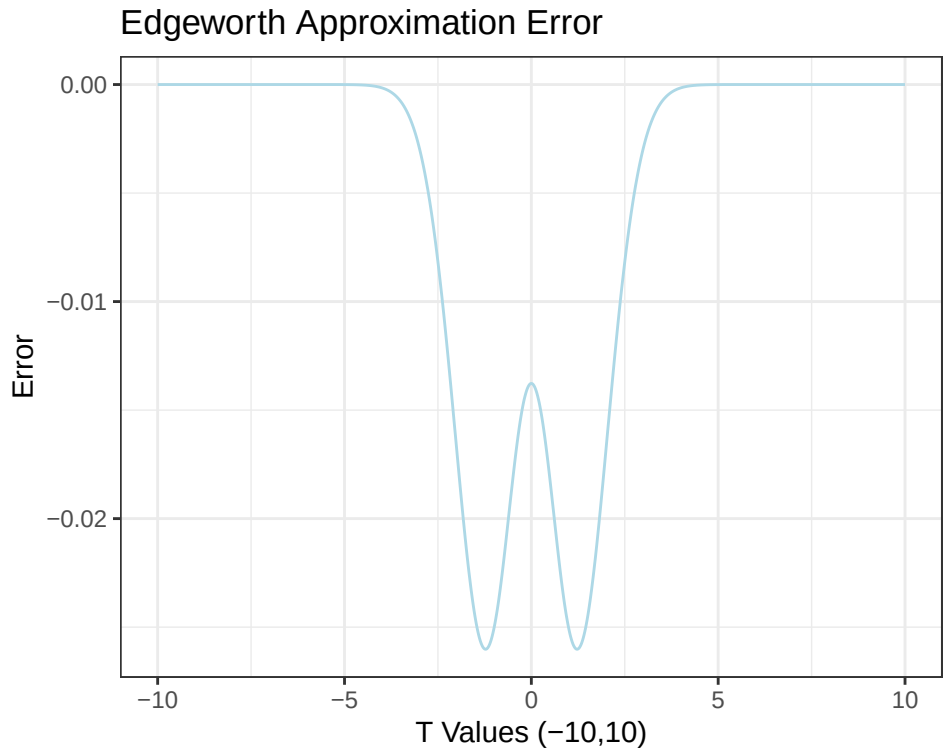

Figure 1: Edgeworth Approximation Error Across t values

For the error across the t-values from -10 to 10, we can see that the maximum error is essentially 0 with the median being around -2.56e-06 at around t=0.

(c) Suppose we wanted to have a tail probability within 10% of the desired $\alpha = 0.05$. Recall we did

a left-tailed test using the further data. How large of a sample size would we need? That is, we need to solve the error formula equal to 10% of the desired left-tail probability:

$$0.10\alpha \overset{set}{=} \underbrace{\frac{\text{skew}}{\sqrt{n}} \frac{(2t^2+1)}{6} f_Z(t)}_{\text{error}},$$

which yields

$$n = \left(\frac{\text{skew}}{6(0.10\alpha)} (2t^2+1) f_Z(t) \right)^2.$$

```
### Part C ### - error in the rejection region
skewness <- skewness(further)
tval <- qnorm(0.05)
fz3 <- dnorm(tval)

(min.n <- ((skewness/(6*(0.10*0.05))) * (2*tval^2 + 1) *fz3)^2)

## [1] 520.8876
```

We would need a sample size of n = 520.8876 for a left-tailed test. This is much larger than the current n.

2. Complete the following steps to revisit the analyses from lab 11 using the bootstrap procedure.

   (a) Now, consider the zebra finch data. We do not know the generating distributions for the closer, further, and difference data, so perform resampling to approximate the sampling distribution of the T statistic:

   $$T = \frac{\bar{x}_r - 0}{s/\sqrt{n}},$$

   where $\bar{x}_r$ is the mean computed on the rth resample and s is the sample standard deviation from the original samples. At the end, create an object called `resamples.null.closer`, for example, and store the resamples shifted to ensure they are consistent with the null hypotheses at the average (i.e., here ensure the shifted resamples are 0 on average, corresponding to $t = 0$, for each case).

```
library(boot)
### Part A ###

# Closer Resamples
R <- 10000
s.closer <- sd(dat.finches$closer)
n.closer <- length(dat.finches$closer)
resamples.closer <-  tibble(tstat=rep(NA, R),
                            xbar=rep(NA, R))
for(i in 1:R){
  curr.resample <- sample(x = dat.finches$closer,
                          size = n.closer,
                          replace = T)
  resamples.closer$tstat[i] <- (mean(curr.resample)-0)/(s.closer/sqrt(n.closer))
  resamples.closer$xbar[i] <- mean(curr.resample)
}
# shift so H0 is true
delta.closer <- mean(resamples.closer$tstat) - 0 # null mu0 = 0
```

```r
resamples.null.closer <- resamples.closer |>
  mutate(tstat.shifted = tstat - delta.closer)

(mean(resamples.null.closer$tstat.shifted))
```

```
## [1] 7.819495e-16
```

```r
# Further Resamples
R <- 10000
s.further <- sd(dat.finches$further)
n.further <- length(dat.finches$further)
resamples.further <- tibble(tstat=rep(NA, R),
                            xbar=rep(NA, R))
for(i in 1:R){
  curr.resample <- sample(x = dat.finches$further,
                          size = n.further,
                          replace = T)
  resamples.further$tstat[i] <- (mean(curr.resample)-0)/(s.further/sqrt(n.further))
  resamples.further$xbar[i] <- mean(curr.resample)
}
# shift so H0 is true
delta.further <- mean(resamples.further$tstat) - 0 # null mu0 = 0
resamples.null.further <- resamples.further |>
  mutate(tstat.shifted = tstat - delta.further)

(mean(resamples.null.further$tstat.shifted))
```

```
## [1] 2.557732e-16
```

```r
# Difference Resamples
R <- 10000
s.diff <- sd(dat.finches$diff)
n.diff <- length(dat.finches$diff)
resamples.diff <-  tibble(tstat=rep(NA, R),
                          xbar=rep(NA, R))
for(i in 1:R){
  curr.resample <- sample(x = dat.finches$diff,
                          size = n.diff,
                          replace = T)
  resamples.diff$tstat[i] <- (mean(curr.resample)-0)/(s.diff/sqrt(n.diff))
  resamples.diff$xbar[i] <- mean(curr.resample)
}
# shift so H0 is true
delta.diff <- mean(resamples.diff$tstat) - 0 # null mu0 = 0
resamples.null.diff <- resamples.diff |>
  mutate(tstat.shifted = tstat - delta.diff)

(mean(resamples.null.diff$tstat.shifted))
```

```
## [1] -3.541167e-16
```

We were asked to preform resampling across the closer, further, and difference data because we do not know the generating distributions for them. We use resampling to approximate the sampling distribution of the T statistic. For all three data sets, the shifted resamples are close to 0 on average which corresponds to t = 0 for each case. This ensures that the resamples shifted are consistent with the null hypothesis at the average.

(b) Compute the bootstrap p-value for each test using the shifted resamples. How do these compare to the t-test p-values?

```
################## Part B ###################
## Closer ##
# Bootstrap P-Value
p.boot.closer <- mean(resamples.null.closer$tstat.shifted >= delta.closer)
# T-Test P-Value
p.t.closer <- t.test(dat.finches$closer,
                     mu = 0,
                     alternative = "greater")
p.t.closer <- p.t.closer$p.value

## Further ##
# Bootstrap P-Value
p.boot.further <- mean(resamples.null.further$tstat.shifted <= delta.further)
# T-Test P-Value
p.t.further <- t.test(dat.finches$further,
                      mu = 0,
                      alternative = "less")
p.t.further <- p.t.further$p.value

## Difference ##
# Bootstrap P-Value
low <- 0 - delta.diff
high <- 0 + delta.diff
p.low <- mean(resamples.null.diff$tstat.shifted <= low)
p.high <- mean(resamples.null.diff$tstat.shifted >= high)
p.boot.diff <- p.low + p.high
# T-Test P-Value
p.t.diff <- t.test(dat.finches$diff,
                   mu = 0,
                   alternative = "two.sided")
p.t.diff <- p.t.diff$p.value

comparison.pvals <- tibble(
  Data = c("Closer", "Further", "Difference"),
  `Bootstrap P-value` = c(p.boot.closer,
                          p.boot.further,
                          p.boot.diff),
  `T-Test P-value` = c(p.t.closer,
                       p.t.further,
                       p.t.diff)
)
comparison.pvals.1 <- xtable(comparison.pvals,
       caption = "Comparing Bootstrap p-values to t-test p-values",
       label = "tab:bootpcomp")
```

Data	Bootstrap P-value	T-Test P-value
Closer	0.00	0.00
Further	0.00	0.00
Difference	0.00	0.00

Table 1: Comparing Bootstrap p-values to t-test p-values

(c) What is the 5^{th} percentile of the shifted resamples under the null hypothesis? Note this value approximates $t_{0.05,n-1}$. Compare these values in each case.

```
################## Part C ###################
# Want to use shifted t stat for the p-value
#Closer
percentile.boot.closer <- quantile(resamples.null.closer$tstat.shifted, 0.05)
percentile.t.closer <- qt(0.05, df = n.closer -1)

#Further
percentile.boot.further <- quantile(resamples.null.further$tstat.shifted, 0.05)
percentile.t.further <- qt(0.05, df = n.further -1)

#Difference
percentile.boot.diff <- quantile(resamples.null.diff$tstat.shifted, 0.05)
percentile.t.diff <- qt(0.05, df = n.diff -1)

comparison.percentile <- tibble(
  Data = c("Closer", "Further", "Difference"),
  `Bootstrap Percentile` = c(percentile.boot.closer,
                             percentile.boot.further,
                             percentile.boot.diff),
  `T-Test Percentile` = c(percentile.t.closer,
                          percentile.t.further,
                          percentile.t.diff)
)
comparison.percent.1 <- xtable(comparison.percentile,
       caption = "Comparing Bootstrap percentiles to t-test percentiles",
       label = "tab:bootperccomp")
```

Data	Bootstrap Percentile	T-Test Percentile
Closer	-1.56	-1.71
Further	-1.66	-1.71
Difference	-1.58	-1.71

Table 2: Comparing Bootstrap percentiles to t-test percentiles

(d) Compute the bootstrap confidence intervals using the resamples. How do these compare to the t-test confidence intervals?

```
# Confidence Interval
### use resamples for boot -- need this for x bar (floruorences)
# want to use resample x bars (not shifted) for the confidence interval
# Closer
(CI.boot.closer <- quantile(resamples.null.closer$xbar, c(0.025, 0.975)))

##      2.5%     97.5%
## 0.1214700 0.1922559

CI.t.closer <- t.test(x=dat.finches$closer, mu = 0, conf.level = 0.95, alternative = "two.sided
(CI.t.closer <- CI.t.closer$conf.int)

## [1] 0.1173875 0.1950586
## attr(,"conf.level")
## [1] 0.95
```

```r
# Further
(CI.boot.further <- quantile(resamples.null.further$xbar, c(0.025, 0.975)))
```

```
##       2.5%      97.5%
## -0.2548966 -0.1568618
```

```r
CI.t.further <- t.test(x=dat.finches$further, mu = 0, conf.level = 0.95, alternative = "two.si
(CI.t.further <- CI.t.further$conf.int)
```

```
## [1] -0.2565176 -0.1489313
## attr(,"conf.level")
## [1] 0.95
```

```r
# Diff
(CI.boot.diff <- quantile(resamples.null.diff$xbar, c(0.025, 0.975)))
```

```
##      2.5%     97.5%
## 0.2797737 0.4419639
```

```r
CI.t.diff <- t.test(x=dat.finches$diff, mu = 0, conf.level = 0.95, alternative = "two.sided")
(CI.t.diff <- CI.t.diff$conf.int)
```

```
## [1] 0.2719028 0.4459921
## attr(,"conf.level")
## [1] 0.95
```

3. Complete the following steps to revisit the analyses from lab 11 using the randomization procedure.

   (a) Now, consider the zebra finch data. We do not know the generating distributions for the closer, further, and difference data, so perform the randomization procedure

```r
################################################################################
# Closer (#3)
################################################################################
#Since we do not know the generating distribution for
# closer, further, and difference data,
# we need to preform randomization procedure.
mu0 <- 0
R <- 10000
rand.closer <- tibble(xbars = rep(NA, R))

# PREPROCESSING: shift the data to be mean 0 under H0
x.shift.closer <- dat.finches$closer - mu0
# RANDOMIZE / SHUFFLE
for(i in 1:R){
  curr.rand <- x.shift.closer *
    sample(x = c(-1, 1),
           size = length(x.shift.closer),
           replace = T)

  rand.closer$xbars[i] <- mean(curr.rand)
}
rand.closer <- rand.closer |>
  mutate(xbars = xbars + mu0) # shifting back

################################################################################
```

```r
# Further (#3)
################################################################################
# We need to preform randomization procedure.
mu0 <- 0
R <- 10000
rand.further <- tibble(xbars = rep(NA, R))

# PREPROCESSING: shift the data to be mean 0 under H0
x.shift.further <- dat.finches$further - mu0
# RANDOMIZE / SHUFFLE
for(i in 1:R){
  curr.rand <- x.shift.further *
    sample(x = c(-1, 1),
           size = length(x.shift.further),
           replace = T)

  rand.further$xbars[i] <- mean(curr.rand)
}
rand.further <- rand.further |>
  mutate(xbars = xbars + mu0) # shifting back

################################################################################
# Diff (#3)
################################################################################
# We need to preform randomization procedure.
mu0 <- 0
R <- 10000
rand.diff <- tibble(xbars = rep(NA, R))

# PREPROCESSING: shift the data to be mean 0 under H0
x.shift.diff <- dat.finches$diff - mu0
# RANDOMIZE / SHUFFLE
for(i in 1:R){
  curr.rand <- x.shift.diff *
    sample(x = c(-1, 1),
           size = length(x.shift.diff),
           replace = T)

  rand.diff$xbars[i] <- mean(curr.rand)
}
rand.diff <- rand.diff |>
  mutate(xbars = xbars + mu0) # shifting back
```

(b) Compute the randomization test p-value for each test.

```r
# Closer
# p-value
obs.mean.closer <- mean(dat.finches$closer)
(p.rand.closer <- mean(rand.closer$xbars >= obs.mean.closer))
```

```
## [1] 0
```

```r
# Further
# p-value
obs.mean.further <- mean(dat.finches$further)
```

```r
(p.rand.further <- mean(rand.further$xbars <= obs.mean.further))
```

```
## [1] 0
```

```r
# Difference
# p-value
delta.diff <- abs(mean(dat.finches$diff) - mu0)
low.diff <- mu0 - delta.diff # mirror
high.diff <- mu0 + delta.diff   # xbar
(p.rand.diff <- mean(rand.diff$xbars <= low.diff) +
  mean(rand.diff$xbars >= high.diff))
```

```
## [1] 0
```

(c) Compute the randomization confidence interval by iterating over values of μ_0.
Hint: You can "search" for the lower bound from Q_1 and subtracting by 0.0001, and the upper bound using Q_3 and increasing by 0.0001. You will continue until you find the first value for which the two-sided p-value is greater than or equal to 0.05.

```r
##############################################################################
# Closer (#3)
##############################################################################
## Confidence Interval ##
R <- 1000
mu0.iterate <- 0.01
starting.point.closer <- mean(dat.finches$closer)
mu.lower.closer <- starting.point.closer
repeat{
  rand.closer <- tibble(xbars = rep(NA, R))
  # PREPROCESSING: shift the data to be mean 0 under H0
  x.shift.closer <- dat.finches$closer - mu.lower.closer
  # RANDOMIZE / SHUFFLE
  for(i in 1:R){
    curr.rand <- x.shift.closer *
      sample(x = c(-1, 1),
             size = length(x.shift.closer),
             replace = T)

    rand.closer$xbars[i] <- mean(curr.rand)
  }
  # Thinking is hard
  rand.closer <- rand.closer |>
    mutate(xbars = xbars + mu.lower.closer) # shifting back

  # p-value  (one-sided)
  obs.mean.closer <- mean(dat.finches$closer)
  p.val.closer <- mean(rand.closer$xbars >= obs.mean.closer)
if(p.val.closer < 0.05){
  break
}else{
  mu.lower.closer <- mu.lower.closer - mu0.iterate
}
}
mu.upper.closer <- starting.point.closer
repeat{
```

```
  rand.closer <- tibble(xbars = rep(NA, R))
  # PREPROCESSING: shift the data to be mean 0 under H0
  x.shift.closer <- dat.finches$closer - mu.upper.closer
  # RANDOMIZE / SHUFFLE
  for(i in 1:R){
    curr.rand <- x.shift.closer *
      sample(x = c(-1, 1),
             size = length(x.shift.closer),
             replace = T)

    rand.closer$xbars[i] <- mean(curr.rand)
  }
  # Thinking is hard
  rand.closer <- rand.closer |>
    mutate(xbars = xbars + mu.upper.closer) # shifting back

  # p-value  (one-sided)
  obs.mean.closer <- mean(dat.finches$closer)
  p.val.closer <- mean(rand.closer$xbars <= obs.mean.closer)

  if(p.val.closer < 0.05){
    break
  }else{
    mu.upper.closer <- mu.upper.closer + mu0.iterate
  }
}
(closer.rand.CI <- c(mu.lower.closer, mu.upper.closer))

## [1] 0.1162231 0.1962231

##############################################################################################
# Further (#3)
##############################################################################################
## Confidence Interval ##
R <- 1000
mu0.iterate <- 0.01
starting.point.further <- mean(dat.finches$further)

mu.lower.further <- starting.point.further
repeat{
  rand.further <- tibble(xbars = rep(NA, R))

  # PREPROCESSING: shift the data to be mean 0 under H0
  x.shift.further <- dat.finches$further - mu.lower.further
  # RANDOMIZE / SHUFFLE
  for(i in 1:R){
    curr.rand <- x.shift.further *
      sample(x = c(-1, 1),
             size = length(x.shift.further),
             replace = T)

    rand.further$xbars[i] <- mean(curr.rand)
  }
  # Thinking is hard
```

```
  rand.further <- rand.further |>
    mutate(xbars = xbars + mu.lower.further) # shifting back

  # p-value  (one-sided)
  obs.mean.further <- mean(dat.finches$further)
  p.val.further <- mean(rand.further$xbars >= obs.mean.further)

  if(p.val.further < 0.05){
    break
  }else{
    mu.lower.further <- mu.lower.further - mu0.iterate
  }
}

mu.upper.further <- starting.point.further
repeat{
  rand.further <- tibble(xbars = rep(NA, R))

  # PREPROCESSING: shift the data to be mean 0 under H0
  x.shift.further <- dat.finches$further - mu.upper.further
  # RANDOMIZE / SHUFFLE
  for(i in 1:R){
    curr.rand <- x.shift.further *
      sample(x = c(-1, 1),
             size = length(x.shift.further),
             replace = T)

    rand.further$xbars[i] <- mean(curr.rand)
  }
  # Thinking is hard
  rand.further <- rand.further |>
    mutate(xbars = xbars + mu.upper.further) # shifting back

  # p-value  (one-sided)
  obs.mean.further <- mean(dat.finches$further)
  p.val.further <- mean(rand.further$xbars <= obs.mean.further)

  if(p.val.further < 0.05){
    break
  }else{
    mu.upper.further <- mu.upper.further + mu0.iterate
  }
}

(further.rand.CI <- c(mu.lower.further, mu.upper.further))

## [1] -0.2527244 -0.1527244

##############################################################################
# Difference (#3)
##############################################################################
## Confidence Interval ##
R <- 1000
mu0.iterate <- 0.01
```

```r
starting.point.diff <- mean(dat.finches$diff)

mu.lower.diff <- starting.point.diff
repeat{
  rand.diff <- tibble(xbars = rep(NA, R))

  # PREPROCESSING: shift the data to be mean 0 under H0
  x.shift.diff <- dat.finches$diff - mu.lower.diff
  # RANDOMIZE / SHUFFLE
  for(i in 1:R){
    curr.rand <- x.shift.diff *
      sample(x = c(-1, 1),
             size = length(x.shift.diff),
             replace = T)

    rand.diff$xbars[i] <- mean(curr.rand)
  }
  # Thinking is hard
  rand.diff <- rand.diff |>
    mutate(xbars = xbars + mu.lower.diff) # shifting back

  # p-value  (one-sided)
  (delta.diff <- abs(mean(dat.finches$diff) - mu.lower.diff))
  (low.diff <- mu.lower.diff - delta.diff) # mirror
  (high.diff <- mu.lower.diff + delta.diff)   # xbar
  (p.val.diff <- mean(rand.diff$xbars <= low.diff) +
      mean(rand.diff$xbars >= high.diff))

  if(p.val.diff < 0.05){
    break
  }else{
    mu.lower.diff <- mu.lower.diff - mu0.iterate
  }
}

mu.upper.diff <- starting.point.diff
repeat{
  rand.diff <- tibble(xbars = rep(NA, R))

  # PREPROCESSING: shift the data to be mean 0 under H0
  x.shift.diff <- dat.finches$diff - mu.upper.diff
  # RANDOMIZE / SHUFFLE
  for(i in 1:R){
    curr.rand <- x.shift.diff *
      sample(x = c(-1, 1),
             size = length(x.shift.diff),
             replace = T)

    rand.diff$xbars[i] <- mean(curr.rand)
  }
  # Thinking is hard
  rand.diff <- rand.diff |>
    mutate(xbars = xbars + mu.upper.diff) # shifting back
```

```
  # p-value
  (delta.diff <- abs(mean(dat.finches$closer) - mu.upper.diff))
  (low.diff <- mu.upper.diff - delta.diff) # mirror
  (high.diff <- mu.upper.diff + delta.diff)   # xbar
  (p.val.diff <- mean(rand.closer$xbars <= low.diff) +
      mean(rand.diff$xbars >= high.diff))

  if(p.val.diff < 0.05){
    break
  }else{
    mu.upper.diff <- mu.upper.diff + mu0.iterate
  }
}

(diff.rand.CI <- c(mu.lower.diff, mu.upper.diff))

## [1] 0.2689475 0.3589475
```

4. **Optional Challenge:** In this lab, you performed resampling to approximate the sampling distribution of the T statistic using

$$T = \frac{\bar{x}_r - 0}{s/\sqrt{n}}.$$

I'm curious whether it is better/worse/similar if we computed the statistics using the sample standard deviation of the resamples (s_r), instead of the original sample (s)

$$T = \frac{\bar{x}_r - 0}{s_r/\sqrt{n}}.$$

   (a) Perform a simulation study to evaluate the Type I error for conducting this hypothesis test both ways.

   (b) Using the same test case(s) as part (a), compute bootstrap confidence intervals and assess their coverage – how often do we 'capture' the parameter of interest?

References

Boos, D. D. and Hughes-Oliver, J. M. (2000). How large does n have to be for z and t intervals? *The American Statistician*, 54(2):121–128.