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?

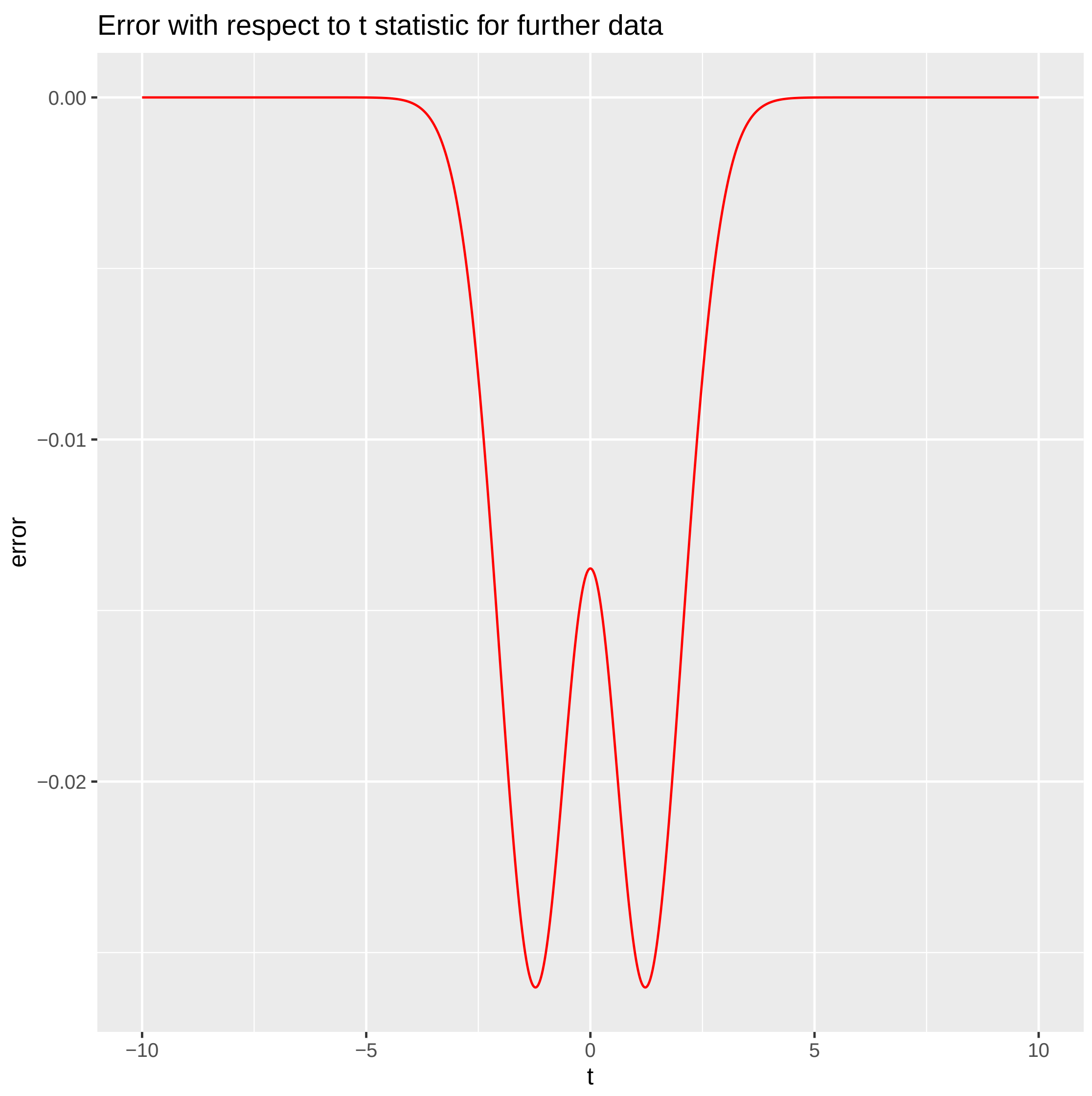
```r
library(tidyverse)
library(e1071)
data = read_csv("zebrafinches.csv")
far.vec = data$further #further data
mu0 = 0
n = 25
far.t.stat = (mean(far.vec) - mu0)/(sd(far.vec)/sqrt(n)) #far t statistic
far.pdf = dnorm(far.t.stat) #pdf using the far t statistic
#potential error for further data
(far.error = (skewness(far.vec)/sqrt(n))*((2*far.t.stat^2 + 1)/6)*far.pdf)

## [1] -1.226006e-13
```

   As you can see, the potential error is essentially zero.

   (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.

```r
t.stat = seq(-10,10, by = 0.01)
error.vec = (skewness(far.vec)/sqrt(n))*((2*t.stat^2 + 1)/6)*dnorm(t.stat)
error.dat = data.frame(t = t.stat, error = error.vec)
error.plot = ggplot(data = error.dat, aes(x = t, y = error))+
  geom_line(color = "red") +
  labs(title = "Error with respect to t statistic for further data")
error.plot
```



As you can see, the error is essentially zero when the magnitude of the t statistic is between 5 and 10. This is because small shifts don't matter very much when the magnitude of the t statistic is so large. As the magnitude of t gets smaller, the magnitude of the error begins to increase towards -0.026. However, between t values of roughly -1 and 1, the error gets a little smaller (approaches -0.013). This is because there is an inflection point somehwere near 1 and -1.

(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

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

```r
alpha = 0.05
t = qnorm(alpha) #t statistic at alpha = 0.05
#required sample size to get a tail probability within 10% of alpha
(n.required = ((skewness(far.vec)/(6*.1*alpha))*((2*t^2 + 1))*dnorm(t))^2)
```

```
## [1] 520.8876
```

It would require a sample size of 521 in order to get a tail probability within 10% of alpha (0.05).

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).

```r
far.vec = data$further
close.vec = data$closer
diff.vec = data$diff
R = 10000 #simulations
n = 25
far.resamples = tibble(t.stats = rep(NA, R))
close.resamples = tibble(t.stats = rep(NA, R))
diff.resamples = tibble(t.stats = rep(NA, R))
#resamples for all 3 types of data
for(i in 1:R){
  curr.far.resample = sample(far.vec,
                             size = n,
                             replace = T)
  curr.close.resample = sample(close.vec,
                               size = n,
                               replace = T)
  curr.diff.resample = sample(diff.vec,
                              size = n,
                              replace = T)
  #resampled t statistics for close, far, and diff data
  far.resamples$t.stats[i] = mean(curr.far.resample)/(sd(far.vec)/sqrt(n))
  close.resamples$t.stats[i] = mean(curr.close.resample)/(sd(close.vec)/sqrt(n))
  diff.resamples$t.stats[i] = mean(curr.diff.resample)/(sd(diff.vec)/sqrt(n))
}
#shifted resamples
resamples.null.far = far.resamples$t.stats - mean(far.resamples$t.stats)
resamples.null.close = close.resamples$t.stats - mean(close.resamples$t.stats)
resamples.null.diff = diff.resamples$t.stats - mean(diff.resamples$t.stats)
#mean of shifted resamples
mean(resamples.null.far)
```

```
## [1] 4.799938e-16
```

```
mean(resamples.null.close)

## [1] -5.931033e-16

mean(resamples.null.diff)

## [1] 1.362466e-16
```

You can see that after shifting all of the resampled data by their mean, to show what the distribution would look like if the null was true, the means of the shifted data are all roughly zero.

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

```
far.t.stat = (mean(far.vec) - mu0)/(sd(far.vec)/sqrt(n)) #far t statistic
close.t.stat = (mean(close.vec) - mu0)/(sd(close.vec)/sqrt(n)) #far t statistic
diff.t.stat = (mean(diff.vec) - mu0)/(sd(diff.vec)/sqrt(n)) #far t statistic
#p values for each test
(far.p = mean(resamples.null.far <= far.t.stat))

## [1] 0

(close.p = mean(resamples.null.close >= close.t.stat))

## [1] 0

(diff.p = mean(resamples.null.diff >= abs(diff.t.stat)) + mean(resamples.null.diff <= -abs(dif

## [1] 0
```

As you can see, the p-values for all three tests are zero. For the t-tests, the p-values were very very close to zero. The reason these p-values are actually zero rather than approximately, is because I calculated the proportion of shifted observations that were as/more extreme than their corresponding t-statistic. Because of how large the magnitude of the three t-statistics are, none of the shifted observations were as extreme, resulting in p-values of zero. The t-test calculates the p-values as a theoretical probability. So even though the probability is very small, there is still some area under the curve. So although the p-values were tiny for the t-tests, the bootstrap procedure resulted in even smaller ones.

(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.

```
(far.5th.percentile = quantile(resamples.null.far, 0.05))

##        5%
## -1.674054

(close.5th.percentile = quantile(resamples.null.close, 0.05))

##        5%
## -1.594521

(diff.5th.percentile = quantile(resamples.null.diff, 0.05))

##        5%
## -1.562238
```

The first thing I notice here is that these values are all relatively close to the 5th percentile of standard normal distribution, which is -1.64. For the far data, the 5th percentile is the most negative, meaning more extreme values are required to reject the null. It makes sense that these values are all pretty similar given that they all have the same sample size (25), and they are all under the assumed null distribution.

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

```
library(boot)
boot.mean <- function(d, i){
  mean(d[i])
}
#bootstrap confidence intervals
boots.far <- boot(data = far.vec,
              statistic = boot.mean,
              R = R)
boot.far.ci = boot.ci(boots.far, type="bca")

boot.far.ci$bca[4:5]

## [1] -0.2630417 -0.1593101

boots.close <- boot(data = close.vec,
                statistic = boot.mean,
                R = R)
boot.close.ci = boot.ci(boots.close, type="bca")

boot.close.ci$bca[4:5]

## [1] 0.1219668 0.1944105

boots.diff <- boot(data = diff.vec,
                statistic = boot.mean,
                R = R)
boot.diff.ci = boot.ci(boots.diff, type="bca")

boot.diff.ci$bca[4:5]

## [1] 0.2852404 0.4500516
```

These confidence intervals are all similar to their corresponding t-test confidence intervals. For the most part, the bootstrap intervals are pulled in from one side, and pulled out from the other side, in relation to the t-test intervals. This is due to the skewness of the distributions. But all in all, the intervals are quite similar, and still exclude zero, suggesting a difference.

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 = 10000
n = 25
far.rand = tibble(xbars = rep(NA, R))
close.rand = tibble(xbars = rep(NA, R))
diff.rand = tibble(xbars = rep(NA, R))
# RANDOMIZE / SHUFFLE
for(i in 1:R){
  curr.far.rand <- far.vec *
    sample(x = c(-1, 1),
           size = n,
           replace = T)
  curr.close.rand <- close.vec *
    sample(x = c(-1, 1),
           size = n,
           replace = T)
  curr.diff.rand <- diff.vec *
    sample(x = c(-1, 1),
           size = n,
           replace = T)

  far.rand$xbars[i] <- mean(curr.far.rand)
  close.rand$xbars[i] <- mean(curr.close.rand)
  diff.rand$xbars[i] <- mean(curr.diff.rand)
}
```

Here there was no shifting to do because mu0 is zero, so all there was to do was shuffle each data vector by randomly multiplying by 1 and -1. I then stored the mean of each shuffled data. We can now use these shuffled means to find the p-values.

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

```
mu0 = 0
delta.diff <- abs(mean(diff.vec) - mu0)
low.diff <- mu0 - delta.diff
high.diff <- mu0 + delta.diff
#p-values
(far.p = mean(far.rand$xbars <= mean(far.vec)))

## [1] 0

(close.p = mean(close.rand$xbars >= mean(close.vec)))

## [1] 0

(diff.p = mean(diff.rand$xbars <= low.diff) + mean(diff.rand$xbars >= high.diff))

## [1] 0
```

Again, the p-values are zero for all three tests. Just like for the bootstrapping test, this is because the magnitude of the test statistics are too large for any observation under the assumption of the null to be more extreme.

(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.

```
mu0 = 0
R <- 1000
mu0.iterate <- 0.0001

far.mu.lower <- quantile(far.vec, 0.25)
close.mu.lower <- quantile(close.vec, 0.25)
diff.mu.lower <- quantile(diff.vec, 0.25)
repeat{
  far.rand <- tibble(xbars = rep(NA, R))
  close.rand <- tibble(xbars = rep(NA, R))
  diff.rand <- tibble(xbars = rep(NA, R))
```

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

  far.rand$xbars[i] <- mean(far.curr.rand)
  close.rand$xbars[i] <- mean(close.curr.rand)
  diff.rand$xbars[i] <- mean(diff.curr.rand)
}
far.rand <- far.rand |>
  mutate(xbars = xbars + far.mu.lower) # shifting back
close.rand <- close.rand |>
  mutate(xbars = xbars + close.mu.lower) # shifting back
diff.rand <- diff.rand |>
  mutate(xbars = xbars + diff.mu.lower) # shifting back
# p-value
far.delta <- abs(mean(far.vec) - far.mu.lower)
close.delta <- abs(mean(close.vec) - close.mu.lower)
diff.delta <- abs(mean(diff.vec) - diff.mu.lower)
far.low <- far.mu.lower - far.delta
close.low <- close.mu.lower - close.delta
diff.low <- diff.mu.lower - diff.delta
far.high<- far.mu.lower + far.delta
close.high<- close.mu.lower + close.delta
diff.high<- diff.mu.lower + diff.delta
far.p.val <- mean(far.rand$xbars <= far.low) +
    mean(far.rand$xbars >= far.high)
close.p.val <- mean(close.rand$xbars <= close.low) +
  mean(close.rand$xbars >= close.high)
diff.p.val <- mean(diff.rand$xbars <= diff.low) +
  mean(diff.rand$xbars >= diff.high)
if(diff.p.val < 0.05){
  diff.mu.lower <- diff.mu.lower + mu0.iterate
}
if(close.p.val < 0.05){
  close.mu.lower <- close.mu.lower + mu0.iterate
}
if(far.p.val < 0.05){
  far.mu.lower <- far.mu.lower + mu0.iterate
  }
  if((diff.p.val>0.05) & (close.p.val>0.05) & (far.p.val>0.05)){
    break
  }
}

far.mu.upper <- quantile(far.vec, 0.75)
close.mu.upper <- quantile(close.vec, 0.75)
diff.mu.upper <- quantile(diff.vec, 0.75)
repeat{
  far.rand <- tibble(xbars = rep(NA, R))
  close.rand <- tibble(xbars = rep(NA, R))
  diff.rand <- tibble(xbars = rep(NA, R))

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

    far.rand$xbars[i] <- mean(far.curr.rand)
    close.rand$xbars[i] <- mean(close.curr.rand)
    diff.rand$xbars[i] <- mean(diff.curr.rand)
  }
  far.rand <- far.rand |>
    mutate(xbars = xbars + far.mu.upper) # shifting back
  close.rand <- close.rand |>
    mutate(xbars = xbars + close.mu.upper) # shifting back
  diff.rand <- diff.rand |>
    mutate(xbars = xbars + diff.mu.upper) # shifting back
  # p-value
  far.delta <- abs(mean(far.vec) - far.mu.upper)
  close.delta <- abs(mean(close.vec) - close.mu.upper)
  diff.delta <- abs(mean(diff.vec) - diff.mu.upper)
  far.low <- far.mu.upper - far.delta
  close.low <- close.mu.upper - close.delta
  diff.low <- diff.mu.upper - diff.delta
  far.high<- far.mu.upper + far.delta
  close.high<- close.mu.upper + close.delta
  diff.high<- diff.mu.upper + diff.delta
  far.p.val <- mean(far.rand$xbars <= far.low) +
    mean(far.rand$xbars >= far.high)
  close.p.val <- mean(close.rand$xbars <= close.low) +
    mean(close.rand$xbars >= close.high)
  diff.p.val <- mean(diff.rand$xbars <= diff.low) +
    mean(diff.rand$xbars >= diff.high)
  if(diff.p.val < 0.05){
    diff.mu.upper <- diff.mu.upper - mu0.iterate
  }
  if(close.p.val < 0.05){
    close.mu.upper <- close.mu.upper - mu0.iterate
  }
  if(far.p.val < 0.05){
    far.mu.upper <- far.mu.upper - mu0.iterate
  }
  if((diff.p.val>0.05) & (close.p.val>0.05) & (far.p.val>0.05)){
    break
  }
}
(far.ci = c(far.mu.lower, far.mu.upper))

##        25%        75%
## -0.2595606 -0.1467054

(close.ci = c(close.mu.lower, close.mu.upper))

##      25%      75%
## 0.119518 0.192770

(diff.ci = c(diff.mu.lower, diff.mu.upper))

##       25%       75%
## 0.2740407 0.4459148
```

We again can see that the confidence intervals are very similar to the corresponding t-test confidence intervals

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.