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?

```
## 1a
zebra.finches.dat <- read_csv("zebrafinches.csv")

further.vec <- zebra.finches.dat$further
skew <- skewness(further.vec)
n <- length(further.vec)

further.test <- t.test(x=further.vec, mu = 0, alternative = "less")
t.far <- further.test$statistic #finding the t value
fz <- dnorm(t.far)
Fz <- pnorm(t.far)
(error = (skew / sqrt(n)) * ((2 * (t.far^2) + 1) / 6) * fz)

##             t
## -1.226006e-13

prob <- error + Fz
```

   (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.

```
t.vecs <- seq(-10,10,0.1)
results <- tibble(
  t = t.vecs,
  fz = dnorm(t.vecs),
  error = (skew / sqrt(n)) * ((2 * (t.vecs^2) + 1) / 6) * dnorm(t.vecs)
)

error.plot <- ggplot(results) +
  theme_bw()+
  geom_line(aes(x = t, y = error))+
  labs(
    x = "T values",
    title = "Edgeworth Approximation Error"
  )
```

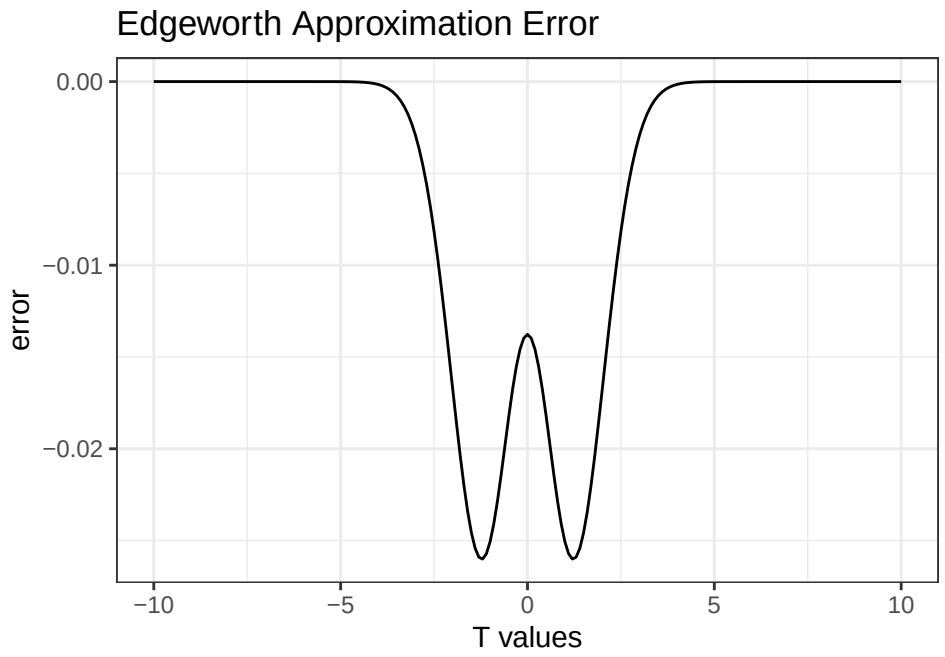

Figure 1: Edgeworth Error across span of t values

(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

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

Answer: We would need a sample size of at least 521 for a left-tailed test.

```
alpha = 0.05
t.val <- qnorm(alpha)
pdf <- dnorm(t.val)
(target.size = ((skew / (6*(0.1*alpha))) * (2 * (t.val^2) + 1) * pdf)^2)

## [1] 520.8876
```

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).

Answer: All of the shifted means of the t values are so small that they are practically 0.

```
##########################
#  Part 2:Boot strapping  #
##########################
R <- 10000
mu0 = 0
fur.sd <- sd(zebra.finches.dat$further)
close.sd <- sd(zebra.finches.dat$closer)
diff.sd <- sd(zebra.finches.dat$diff)

resample <- tibble(close.t=numeric(R),
                   far.t=numeric(R),
                   diff.t=numeric(R),
                   close.mean =numeric(R),
                   far.mean =numeric(R),
                   diff.mean =numeric(R)
)
resamples.shifted <- tibble()


for(i in 1:R){
  close.sample <- sample(x=zebra.finches.dat$closer,
                         size=n,
                         replace=T)
  far.sample <- sample(x=zebra.finches.dat$further,
                       size=n,
                       replace=T)
  diff.sample <- sample(x=zebra.finches.dat$diff,
                        size=n,
                        replace=T)

  resample$close.t[i] = (mean(close.sample)-mu0)/(close.sd/sqrt(n))
  resample$far.t[i] = (mean(far.sample)-mu0)/(fur.sd/sqrt(n))
  resample$diff.t[i] = (mean(diff.sample)-mu0)/(diff.sd/sqrt(n))
  resample$close.mean[i] = mean(close.sample)
  resample$far.mean[i] = mean(far.sample)
  resample$diff.mean[i] = mean(diff.sample)

}

## shifting the means
close.change <- mean(resample$close.t)
far.change <- mean(resample$far.t)
diff.change <- mean(resample$diff.t)

resamples.shifted <- resample |>
  mutate(close.shifted = close.t - close.change)  |>
  mutate(far.shifted = far.t - far.change) |>
  mutate(diff.shifted = diff.t - diff.change) |>
  select(c(close.shifted, far.shifted, diff.shifted ))
(mean(resamples.shifted$close.shifted))

## [1] 1.171951e-16

(mean(resamples.shifted$far.shifted))

## [1] 1.382103e-16

(mean(resamples.shifted$diff.shifted))

## [1] 3.648193e-16
```

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

Answer: Table 1 below shows that the comparison of the p-values. While t-tests did produce nonzero p-values, they were so small that they are essentially 0, which is what the bootstrap test shows for each case.

```
# Bootstrap P-Value
boot.p.closer <- mean(resamples.shifted$close.shifted >= close.change)
boot.p.far <- mean(resamples.shifted$far.shifted <= far.change)
low = -diff.change
high = diff.change
p.low <- mean(resamples.shifted$diff.shifted <= low)
p.high <- mean(resamples.shifted$diff.shifted >= high)
boot.p.diff <- p.low +p.high
```

```
# T-Test P-Value
test.close <- t.test(zebra.finches.dat$closer,
                     mu = 0,
                     alternative = "greater")
close.ttest.p <- test.close$p.value

test.far <- t.test(zebra.finches.dat$further,
                     mu = 0,
                     alternative = "less")
far.ttest.p <- test.far$p.value

test.diff <- t.test(zebra.finches.dat$diff,
                    mu = 0,
                    alternative = "two.sided")
diff.ttest.p <- test.diff$p.value

comparisons.p <- tibble(" "= c("close", "far", "diff"), Bootstrapped =
                        c(boot.p.closer, boot.p.far, boot.p.diff),
                      t.test = c(close.ttest.p, far.ttest.p, diff.ttest.p))
```

Data	Bootstrapped	T test
closer	0.00	0.00
further	0.00	0.00
difference	0.00	0.00

Table 1: Comparison of p-values between bootstrapping and T test

(c) What is the 5^{th} percentile of the shifted resamples under the null hypothesis?

Answer: Table 2 below shows that the comparison of the 5^{th} percentile of the shifted resamples: closer = -1.54, further = -1.70, and difference = -1.61. They approximate the T-test, which is -1.71.

```
## Part c
close.5th <- quantile(resamples.shifted$close.shifted, 0.05)
far.5th <- quantile(resamples.shifted$far.shifted, 0.05)
diff.5th <- quantile(resamples.shifted$diff.shifted, 0.05)

close.percentile.t <- qt(0.05, df = n-1)
far.percentile.t <- qt(0.05, df = n-1)
diff.percentile.t <- qt(0.05, df = n-1)

comparisons.percentiles <- tibble(" "= c("close", "far", "diff"), sampled =
                        c(close.5th, far.5th, diff.5th),
                      t.val =c(close.percentile.t, far.percentile.t,
                               diff.percentile.t))
```

Data	Bootstrapped	T-test
Closer	-1.54	-1.71
Further	-1.70	-1.71
Difference	-1.61	-1.71

Table 2: Comparison of the 5th percentile between bootstrapping and T test

Note this value approximates $t_{0.05,n-1}$. Compare these values in each case.

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

Answer: Table 3 below shows that the comparison of the confidence intervals of the shifted resamples with those of the T-tests. Respectively for the closer, further, and difference cases, bootstrapping produced 95% confidence intervals of (0.12, 0.18), (-0.26, -0.17), and (0.27, 0.42) while the T-tests produced CIs of (0.12, 0.20), (-0.26, -0.15), and (0.27, 0.45). They are pretty similar to each other for each case

```r
## Part d
lower.close <- quantile(resample$close.mean, 0.025)
upper.close <- quantile(resample$close.mean, 0.925)
Bootstrap.CI.close <- c(lower.close, upper.close)

lower.far <- quantile(resample$far.mean, 0.025)
upper.far <- quantile(resample$far.mean, 0.925)
Bootstrap.CI.far <- c(lower.far, upper.far)

lower.diff <- quantile(resample$diff.mean, 0.025)
upper.diff <- quantile(resample$diff.mean, 0.925)
Bootstrap.CI.diff <- c(lower.diff, upper.diff)

# t tests confidence intervals
test.close <- t.test(zebra.finches.dat$closer,
                     mu = 0, conf.level = 0.95,
                     alternative = "two.sided")
close.CI <- test.close$conf.int

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

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

comparisons.CI <- tibble(
  "Condition" = c("close", "far", "diff"),
  "Bootstrapped Lower" = c(lower.close, lower.far, lower.diff),
  "Bootstrapped Upper" = c(upper.close, upper.far, upper.diff),
  "t-test Lower" = c(close.CI[1], far.CI[1], diff.CI[1]),
  "t-test Upper" = c(close.CI[2], far.CI[2], diff.CI[2]))
```

Data	Bootstrapped Lower	Bootstrapped Upper	t-test Lower	t-test Upper
close	0.12	0.18	0.12	0.20
far	-0.26	-0.17	-0.26	-0.15
diff	0.27	0.42	0.27	0.45

Table 3: Comparison of the Confidence Interval between bootstrapping and T test, separated into lower and upper bounds

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
mu0 <- 0
R <- 10000
randomized <- tibble(close.means = numeric(R), far.means = numeric(R),
                     diff.means = numeric(R))

shifted.x.close <- zebra.finches.dat$closer - mu0
shifted.x.far <- zebra.finches.dat$further - mu0
shifted.x.diff <- zebra.finches.dat$diff - mu0

for(i in 1:R){
  curr.rand1 <- shifted.x.close *
    sample(x = c(-1, 1),
           size = length(shifted.x.close),
           replace = T)
  curr.rand2 <- shifted.x.far *
    sample(x = c(-1, 1),
           size = length(shifted.x.far),
           replace = T)
  curr.rand3 <- shifted.x.diff *
    sample(x = c(-1, 1),
           size = length(shifted.x.diff),
           replace = T)

  randomized$close.means[i] <- mean(curr.rand1)
  randomized$far.means[i] <- mean(curr.rand2)
```

```
  randomized$diff.means[i] <- mean(curr.rand3)
}

randomized <- randomized |>
  mutate(close.means = close.means + mu0,
         far.means = far.means + mu0,
         diff.means = diff.means + mu0 ) # shifting data back
```

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

Answer: The p value is 0 for all three cases, so the close, further, and difference data are in the tailed regions of their respective null distributions. The closer data is in its right-tailed zone, further in its left-tailed zone, and difference in both tailed zones, so they all reject the null hypothesis.

```
##########################################
## 3b
## found observed data
observed.close.mean <- mean(zebra.finches.dat$closer -mu0)
observed.far.mean <- mean(zebra.finches.dat$further -mu0)
observed.diff.mean <- mean(zebra.finches.dat$diff -mu0)

## p-values for each value type
rand.p.close <- mean(randomized$close.means >= observed.close.mean)
rand.p.far <- mean(randomized$far.means <= observed.far.mean)

(delta <- abs(mean(zebra.finches.dat$diff) - mu0))

## [1] 0.3589475

(low <- mu0 - delta) # mirror

## [1] -0.3589475

(high<- mu0 + delta)   # xbar

## [1] 0.3589475

rand.p.diff <- mean(randomized$diff.means <= low) +
  mean(randomized$diff.means >= high)

c(rand.p.close,rand.p.far,rand.p.diff)

## [1] 0 0 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.

Answer: We have to numerically compute the confidence interval of a randomization test, so we slowly decreased the mean until the p-value hit 0.05 to find the lower bound and slowly increased the mean until the same condition to find the upper bound for each case. Our final result showed a confidence interval for the closer data as (0.1189, 0.1941), for the further data (-0.2627, -0.1427), and for the difference data (0.2689, 0.4489).

```
##########################################
## 3c
## creating randomized Confidence Intervals

R <- 1000
mu0.iterate <- 0.0001
starting.point <- mean(zebra.finches.dat$closer)
mu.lower.close <- starting.point

repeat{
  rand <- tibble(xbars = rep(NA, R))

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

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

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

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


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


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

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

    rand$xbars[i] <- mean(curr.rand)
  }

  rand <- rand |>
    mutate(xbars = xbars + mu.upper.close) # shifting back

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

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


## CI for further
starting.point <- mean(zebra.finches.dat$further)
mu.lower.far <- starting.point

repeat{
  rand <- tibble(xbars = rep(NA, R))

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

    rand$xbars[i] <- mean(curr.rand)
  }

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

  # p-value
  delta <- abs(mean(zebra.finches.dat$further) - mu.lower.far)
```

```
  low <- mu.lower.far - delta # mirror
  high<- mu.lower.far + delta   # xbar
  p.val <- mean(rand$xbars <= low) +
      mean(rand$xbars >= high)


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


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

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

    rand$xbars[i] <- mean(curr.rand)
  }

  rand <- rand |>
    mutate(xbars = xbars + mu.upper.far) # shifting back

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

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


## CI for diff
starting.point <- mean(zebra.finches.dat$diff)
mu.lower.diff <- starting.point

repeat{
  rand <- tibble(xbars = rep(NA, R))

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

    rand$xbars[i] <- mean(curr.rand)
  }

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

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


  if(p.val < 0.05){
    break
```

```
  }else{
    mu.lower.diff <- mu.lower.diff - mu0.iterate
  }
}


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

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

    rand$xbars[i] <- mean(curr.rand)
  }

  rand <- rand |>
    mutate(xbars = xbars + mu.upper.diff) # shifting back

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

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


rand.CI <- tibble(" " = c("close", "far", "diff"),
                  lower.limit = c(mu.lower.close, mu.lower.far, mu.lower.diff),
                  upper.limit = c(mu.upper.close, mu.upper.far, mu.upper.diff))
```

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.