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?
   Answer: The error is approximately -1.225×10^{-13}

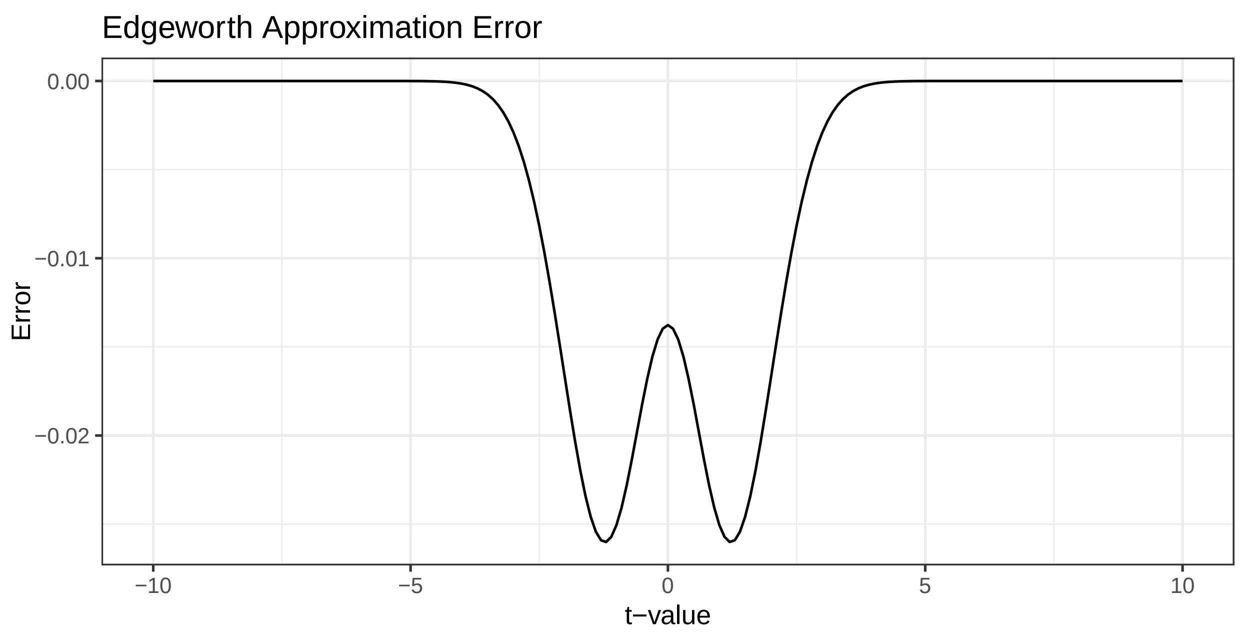
```
################################################################################
further.dat <- finches.dat$further
mu.0 <- 0
t.further <- t.test(x = further.dat,
                    alternative = "less",
                    mu = mu.0)
# t-value
t <- t.further$statistic
n <- length(further.dat)
gauss.pdf <- dnorm(x = t)
gauss.cdf <- pnorm(q = t)
(error <- ((skewness(further.dat) / sqrt(n)) * ((2*t^2 + 1)/6)) * gauss.pdf)

##             t
## -1.226006e-13

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

   (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.

```
################################################################################
tVals <- seq(-10, 10, 0.1)
t.errors = tibble(t = numeric(length(tVals)), error = numeric(length(tVals)))
i = 1
for(t in tVals){
  t.errors$t[i] <- t
  gauss.pdf <- dnorm(x = t)
  gauss.cdf <- pnorm(q = t)
  curr.error <- ((skewness(further.dat) / sqrt(n)) * ((2*t^2 + 1)/6)) * gauss.pdf
  t.errors$error[i] <- curr.error
  i <- i+1
}
first.p <- ggplot(x = t.errors) +
  geom_line(aes(x = t.errors$t, y = t.errors$error)) +
  theme_bw() +
  labs(y = "Error",
       x = "t-value",
       title = "Edgeworth Approximation Error")
################################################################################
```



(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 need a sample size of approximately 520.

```
################################################################################
t <- qnorm(0.05)
n <- length(further.dat)
gauss.pdf <- dnorm(x = t)
gauss.cdf <- pnorm(q = t)
alpha <- 0.05
skew <- skewness(further.dat)
(size <- (((skew/(6*(0.1*alpha))) * ((2*t^2) + 1) * gauss.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).

```r
################################################################################
# re-sampling for further data
n <- length(further.dat)
R <- 10000
resamples.further <- tibble(t.stat = numeric(R), mean = numeric(R))
s <- sd(further.dat)

for(i in 1:R){
curr.sample <- sample(x = further.dat,
                      size = n,
                      replace = T)
t.stat <- mean(curr.sample) / (s/sqrt(n))
resamples.further$t.stat[i] <- t.stat
resamples.further$mean[i] <- mean(curr.sample)
}
################################################################################
# further bootstrapping
f.delta <- mean(resamples.further$t.stat) - 0
resamples.null.further <- resamples.further |>
mutate(t.shifted = resamples.further$t.stat - f.delta)
# re sampling for closer data
closer.dat <- finches.dat$closer
n <- length(closer.dat)
R <- 10000
s <- sd(closer.dat)
resamples.closer <- tibble(t.stat = numeric(R), mean = numeric(R))

for (i in 1:R){
curr.sample <- sample(x = closer.dat,
                      size = n,
                      replace = T)
t.stat <- mean(curr.sample) / (s/sqrt(n))
resamples.closer$t.stat[i] <- t.stat
resamples.closer$mean[i] <- mean(curr.sample)
}
################################################################################
# closer bootstrapping
c.delta <- mean(resamples.closer$t.stat) - 0
resamples.null.closer <- resamples.closer |>
mutate(t.shifted = resamples.closer$t.stat - c.delta)
################################################################################
# re sampling for difference data
difference.dat <- finches.dat$diff
n <- length(closer.dat)
R <- 10000
s <- sd(difference.dat)
resamples.difference <- tibble(t.stat = numeric(R), mean = numeric(R))

for (i in 1:R){
curr.sample <- sample(x = difference.dat,
                      size = n,
                      replace = T)
t.stat <- mean(curr.sample) / (s/sqrt(n))
resamples.difference$t.stat[i] <- t.stat
resamples.difference$mean[i] <- mean(curr.sample)
}
#################################################################################
# diff bootstrapping
d.delta <- mean(resamples.difference$t.stat) - 0
resamples.null.diff <- resamples.difference |>
mutate(t.shifted = resamples.difference$t.stat - d.delta)
################################################################################
```

(b) Compute the bootstrap p-value for each test using the shifted resamples.

```r
################################################################################
# further
further.boot.p <- mean(resamples.null.further$t.shifted <= f.delta )
further.t.p <- (t.test(x = further.dat, mu = 0, alternative = "less"))$p.value
# closer
closer.boot.p <- mean(resamples.null.closer$t.shifted >= c.delta)
closer.t.p <- (t.test(x = closer.dat, mu = 0, alternative = "greater"))$p.value
# difference
low <- mean(resamples.null.diff$t.shifted <= -d.delta)
high <- mean(resamples.null.diff$t.shifted >= d.delta)
diff.boot.p <- low + high
diff.t.p <- (t.test(x = difference.dat, mu = 0, alternative = "two.sided"))$p.value
```

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

data	t.test.p	boot.p
further	0.00	0.00
closer	0.00	0.00
difference	0.00	0.00

Table 1: T-test and Bootstrap p-values

How do these compare to the t-test p-values? **Answer:** The bootstrap p-values are all zero, which is the same a the p-values from the t-test.

(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.

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

closer.boot.ptl <- quantile(resamples.null.closer$t.shifted, 0.95)
closer.t.ptl <- qt(0.95, df = n - 1)

diff.boot.ptl <- quantile(resamples.null.diff$t.shifted, 0.05)
diff.t.ptl <- qt(0.05, df = n - 1)
percentile.summary <- tibble(data = c("further", "closer", "difference"),
                       t.test.p = c(further.t.ptl, closer.t.ptl, diff.t.ptl),
                       boot.p = c(further.boot.ptl, closer.boot.ptl, diff.boot.ptl))
################################################################################
```

data	t.test.p	boot.p
further	-1.71	-1.67
closer	-1.71	-1.60
difference	-1.71	-1.56

Table 2: T-test and Bootstrap 5^{th} percentiles

Answer: We can see that percentiles for the two methods are roughly the same, although the bootstrap values are slightly less extreme and closer to 0.

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

```
################################################################################
# further
further.boot.ci <-quantile(resamples.further$mean, c(0.025, 0.975))
further.t.ci <- t.test(further.dat, mu = 0, conf.level = 0.95,
                       alternative = "two.sided")$conf.int
# closer
closer.boot.ci = quantile(resamples.closer$mean, c(0.025, 0.975))
closer.t.ci <- t.test(closer.dat, mu = 0, conf.level = 0.95,
                       alternative = "two.sided")$conf.int
#difference
diff.boot.ci = quantile(resamples.difference$mean, c(0.025, 0.975))
diff.t.ci <- t.test(difference.dat, mu = 0, conf.level = 0.95,
                       alternative = "two.sided")$conf.int
# further
further.boot.ci # bootstrapping confidence interval

##       2.5%      97.5%
## -0.2552970 -0.1545911

further.t.ci # t.test confidence interval

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

# closer
closer.boot.ci
```

```
##      2.5%     97.5%
## 0.1215823 0.1928475

closer.t.ci

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

# difference
diff.boot.ci

##      2.5%     97.5%
## 0.2810074 0.4433528

diff.t.ci

## [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

```
################################################################################
# randomization for further data
shifted.further <- further.dat - mu.0
# perform (randomization) shuffling
further.rand <- tibble(mean = numeric(R))
s <- sd(further.dat)
n <- length(further.dat)

for(i in 1:R){
  curr.rand <- shifted.further *
    sample(x = c(-1, 1),
           size = length(shifted.further),
           replace = T)
  further.rand$mean[i] <- mean(curr.rand)
}

further.rand <- further.rand |>
  mutate(mean = mean + mu.0)

# randomization for closer data
shifted.closer <- closer.dat - mu.0
# perform (randomization) shuffling
closer.rand <- tibble(mean = numeric(R))
s <- sd(closer.dat)
n <- length(closer.dat)
for(i in 1:R){
  curr.rand <- shifted.closer *
    sample(x = c(-1, 1),
           size = length(shifted.closer),
           replace = T)
  closer.rand$mean[i] <- mean(curr.rand)
}
# shift back
closer.rand <- closer.rand |>
  mutate(mean = mean + mu.0)
# randomization for difference data
shifted.difference <- difference.dat - mu.0
# perform (randomization) shuffling
diff.rand <- tibble(mean = numeric(R))
s <- sd(difference.dat)
n <- length(difference.dat)
for(i in 1:R){
  curr.rand <- shifted.difference *
    sample(x = c(-1, 1),
           size = length(shifted.difference),
           replace = T)
  diff.rand$mean[i] <- mean(curr.rand)
}
# shift back
diff.rand <- diff.rand |>
  mutate(mean = mean + mu.0)
```

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

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

```
################################################################################
# further data
delta <- abs(mean(further.dat) - mu.0)
low <- mu.0 - delta # mirror
high<- mu.0 + delta   # xbar
further.rand.p <- mean(further.rand$mean <= low) +
  mean(further.rand$mean >= high)
# closer data
delta <- abs(mean(closer.dat) - mu.0)
low <- mu.0 - delta # mirror
high<- mu.0 + delta  # xbar
closer.rand.p <- mean(closer.rand$mean <= low) +
  mean(closer.rand$mean >= high)
# difference data
delta <- abs(mean(difference.dat) - mu.0)
low <- mu.0 - delta # mirror
high<- mu.0 + delta   # xbar
diff.rand.p <- mean(diff.rand$mean <= low) +
  mean(diff.rand$mean >= high)
random.p.summary <- tibble(data = c("further", "closer", "difference"),
                           p_value = c(further.rand.p, closer.rand.p, diff.rand.p))
################################################################################
```

data	p_value
further	0.00
closer	0.00
difference	0.00

(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: The confidence interval for the further data is (-0.203 -0.151), the closer data is (0.118, 1.195) and difference data is (0.276, 0.446)

```
################################################################################
# for further data
R <- 1000
mu0.iterate <- 0.001
starting.point <- mean(further.dat)
mu.lower <- starting.point
repeat{
rand <- tibble(xbars = rep(NA, R))
# PREPROCESSING: shift the data to be mean 0 under H0
x.shift <- further.dat - mu.lower
# RANDOMIZE / SHUFFLE
for(i in 1:R){
  curr.rand <- x.shift *
    sample(x = c(-1, 1),
           size = length(x.shift),
           replace = T)
  rand$xbars[i] <- mean(curr.rand)
}
# Thinking is hard
rand <- rand |>
  mutate(xbars = xbars + mu.lower) # shifting back
# p-value
delta <- abs(mean(further.dat) - mu.lower)
low <- mu.lower - delta # mirror
high<- mu.lower + delta   # xbar
p.val <- mean(rand$xbars <= low) +
    mean(rand$xbars >= high)

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

mu.upper <- starting.point
```

```r
repeat{
rand <- tibble(xbars = rep(NA, R))
# PREPROCESSING: shift the data to be mean 0 under H0
x.shift <- further.dat - mu.upper
# 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)
}
# thinking is hard
rand <- rand |>
  mutate(xbars = xbars + mu.upper) # shifting back

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

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

# for closer data
R <- 1000
mu0.iterate <- 0.001
starting.point <- mean(closer.dat)
mu.lower <- starting.point
repeat{
rand <- tibble(xbars = rep(NA, R))
# PREPROCESSING: shift the data to be mean 0 under H0
x.shift <- closer.dat - mu.lower
# RANDOMIZE / SHUFFLE
for(i in 1:R){
  curr.rand <- x.shift *
    sample(x = c(-1, 1),
           size = length(x.shift),
           replace = T)
  rand$xbars[i] <- mean(curr.rand)
}
# Thinking is hard
rand <- rand |>
  mutate(xbars = xbars + mu.lower) # shifting back
# p-value
delta <- abs(mean(closer.dat) - mu.lower)
low <- mu.lower - delta # mirror
high<- mu.lower + delta   # xbar
p.val <- mean(rand$xbars <= low) +
    mean(rand$xbars >= high)
if(p.val < 0.05){
  break
}else{
  mu.lower <- mu.lower - mu0.iterate
}
}

mu.upper <- starting.point
repeat{
rand <- tibble(xbars = rep(NA, R))
# PREPROCESSING: shift the data to be mean 0 under H0
x.shift <- closer.dat - mu.upper
# 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)
}
# Thinking is hard
rand <- rand |>
```

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

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

closer.rand.ci <- c(mu.lower, mu.upper)
# for difference data
R <- 1000
mu0.iterate <- 0.001
starting.point <- mean(difference.dat)
mu.lower <- starting.point
repeat{
rand <- tibble(xbars = rep(NA, R))
# PREPROCESSING: shift the data to be mean 0 under H0
x.shift <- difference.dat - mu.lower
# RANDOMIZE / SHUFFLE
for(i in 1:R){
  curr.rand <- x.shift *
    sample(x = c(-1, 1),
           size = length(x.shift),
           replace = T)

  rand$xbars[i] <- mean(curr.rand)
}
# Thinking is hard
rand <- rand |>
  mutate(xbars = xbars + mu.lower) # shifting back
# p-value
delta <- abs(mean(difference.dat) - mu.lower)
low <- mu.lower - delta # mirror
high<- mu.lower + delta # xbar
p.val <- mean(rand$xbars <= low) +
    mean(rand$xbars >= high)

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

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

# PREPROCESSING: shift the data to be mean 0 under H0
x.shift <- difference.dat - mu.upper
# RANDOMIZE / SHUFFLE
for(i in 1:R){
  curr.rand <- x.shift *
    sample(x = c(-1, 1),
           size = length(x.shift),
           replace = T)
  rand$xbars[i] <- mean(curr.rand)
}
# Thinking is hard
rand <- rand |>
  mutate(xbars = xbars + mu.upper) # shifting back
# p-value
delta <- abs(mean(difference.dat) - mu.upper)
low <- mu.upper - delta # mirror
high<- mu.upper + delta
p.val <- mean(rand$xbars <= low) +
    mean(rand$xbars >= high)

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

```
}
difference.rand.ci <- c(mu.lower, mu.upper)
################################################################################
```

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.
   We see that conducting the hypothesis test by using the standard deviation of the resample increases the Type I error rate significantly (from approximately 5% to 12%.

```
################################################################################
R <- 10000
n <- 25
alpha <- 0.05
mu <- 0
type1_fixed <- numeric(R)
type1_resample <- numeric(R)
for (i in 1:R) {
  #x <- resamples.further£t.stat
  x <- rnorm(n, mean = mu)
  s_fixed <- sd(x) # using original sd
  t_fixed <- mean(x) / (s_fixed / sqrt(n))
  resample <- sample(x,
                     size = n,
                     replace = TRUE)
  s_resample <- sd(resample) # using resample sd
  t_resample <- mean(resample) / (s_resample / sqrt(n))
  type1_fixed[i] <- (t_fixed < qt(0.05, n - 1))
  type1_resample[i] <- (t_resample < qt(0.05, n - 1))
}

mean(type1_fixed)

## [1] 0.0516

mean(type1_resample)

## [1] 0.1265

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

   (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?
   We see that we use the standard of the original data, the parameter of interest is captured 92.2% compared to 95.6% of the time when the resample standard deviation is used.

```
################################################################################
R <- 1000
n <- 25
mu <- 0
coverage_fixed <- numeric(R)
coverage_resample <- numeric(R)
# function for CI using original sd
boot.stat.fixed <- function(d, i){
  s_fixed <- sd(x)
  mean(d[i]) / (s_fixed / sqrt(n))
}
# function for CI using resample sd
boot.stat.resample <- function(d, i) {
  x_i <- d[i]
  mean(x_i) / (sd(x_i) / sqrt(n))
}
for (i in 1:R) {
  x <- rnorm(n, mean = mu)
  boots_fixed <- boot(data = x,
                      statistic = boot.stat.fixed,
                      R = 1000)
  ci_fixed <- boot.ci(boots_fixed,
                      type = "bca")$bca[4:5]
  coverage_fixed[i] <- (mu >= ci_fixed[1] & mu <= ci_fixed[2])
  # boot ci using resample sd
  boots_resample <- boot(data = x, statistic = boot.stat.resample, R = 1000)
  ci_resample <- boot.ci(boots_resample, type = "bca")$bca[4:5]
  coverage_resample[i] <- (mu >= ci_resample[1] & mu <= ci_resample[2])
}
mean(coverage_fixed)

## [1] 0.932

mean(coverage_resample)

## [1] 0.957

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

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.