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)
#Load in Data
dat.finch = read.csv("zebrafinches.csv")

#Question 1
library(moments)#used for calculating statistics
library(ggplot2)

n <- length(dat.finch$further)
x_bar <- mean(dat.finch$further)
s <- sd(dat.finch$further)
t_obs <- x_bar / (s / sqrt(n)) #t-value
skew <- skewness(dat.finch$further)

#Could use this instead shows same thing
#(ttest <- t.test(x = dat.finch£further,
                # mu = 0,
                # alternative = "less"))
# Gaussian PDF and CDF at t
fz <- dnorm(t_obs)
Fz <- pnorm(t_obs)

# Edgeworth approximation error
edgeworth_error <- (skew / sqrt(n)) * ((2 * t_obs^2 + 1) / 6) * fz
```

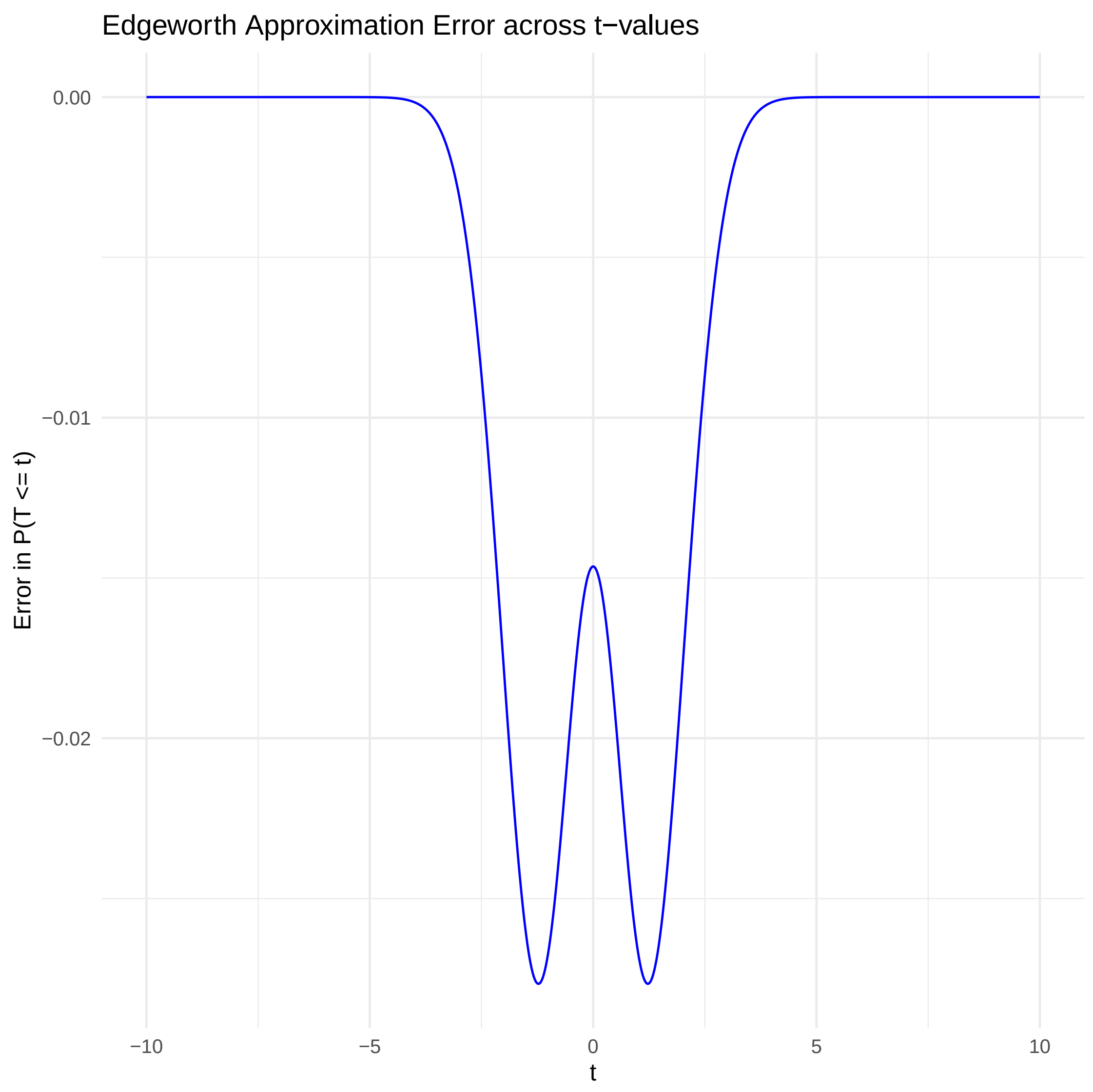
   The Edgeworth error in p-value estimate: -1.303424e-13. This means the potential error is very small only changing the p value by 1.303424e-11 percent which is not important.

   (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_vals <- seq(-10, 10, length.out = 1000)
fz_vals <- dnorm(t_vals)
error_vals <- (skew / sqrt(n)) * ((2 * t_vals^2 + 1) / 6) * fz_vals

error_df <- data.frame(t = t_vals, error = error_vals)

ggplot(error_df, aes(x = t, y = error)) +
  geom_line(color = "blue") +
  labs(title = "Edgeworth Approximation Error across t-values",
       x = "t", y = "Error in P(T <= t)") +
  theme_minimal()
```



(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

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

```
alpha <- 0.05
target_error <- 0.10 * alpha  # 10% of alpha
t_alpha <- qnorm(alpha)  # for left-tailed test
fz_alpha <- dnorm(t_alpha)
```

```
# Solve for n
numerator <- skew * (2 * t_alpha^2 + 1) * fz_alpha
n_required <- (numerator / (6 * target_error))^2
```

The required sample size to keep the Edgeworth approximation error within 10% of the tail probability $\alpha = 0.05$ is approximately $n = 589$.

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)

closer <- dat.finch$closer
further <- dat.finch$further
diff <- dat.finch$diff

# Sample sizes
n_closer <- length(closer)
n_further <- length(further)
n_diff    <- length(diff)

# Original standard deviations
s_closer <- sd(closer)
s_further <- sd(further)
s_diff <- sd(diff)

R <- 10000
# Resample under null hypothesis: shifted to be consistent with t = 0
resamples.null.closer <- tibble(t = replicate(R, {
  samp <- sample(closer, n_closer, replace = TRUE)
  xbar <- mean(samp)
  t <- (xbar - mean(closer)) / (s_closer / sqrt(n_closer))  # shift xbar so mean is 0
  return(t)
}))

resamples.null.further <- tibble(t = replicate(R, {
  samp <- sample(further, n_further, replace = TRUE)
  xbar <- mean(samp)
  t <- (xbar - mean(further)) / (s_further / sqrt(n_further))
  return(t)
}))

resamples.null.diff <- tibble(t = replicate(R, {
  samp <- sample(diff, n_diff, replace = TRUE)
  xbar <- mean(samp)
  t <- (xbar - mean(diff)) / (s_diff / sqrt(n_diff))
  return(t)
}))
```

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

```
# Observed t-statistics
t_obs_closer <- (mean(closer) - 0) / (s_closer / sqrt(n_closer))
t_obs_further <- (mean(further) - 0) / (s_further / sqrt(n_further))
t_obs_diff <- (mean(diff) - 0) / (s_diff / sqrt(n_diff))

# Two-sided bootstrap p-values
pval_boot_closer <- mean(abs(resamples.null.closer$t) >= abs(t_obs_closer))
pval_boot_further <- mean(abs(resamples.null.further$t) >= abs(t_obs_further))
pval_boot_diff <- mean(abs(resamples.null.diff$t) >= abs(t_obs_diff))

# Compare to t-tests
pval_ttest_closer <- t.test(closer, mu = 0)$p.value
pval_ttest_further <- t.test(further, mu = 0)$p.value
pval_ttest_diff <- t.test(diff, mu = 0)$p.value

tibble(
  method = c("t-test", "bootstrap"),
  closer = c(pval_ttest_closer, pval_boot_closer),
  further = c(pval_ttest_further, pval_boot_further),
  diff = c(pval_ttest_diff, pval_boot_diff)
)

## # A tibble: 2 x 4
##   method          closer      further         diff
##   <chr>            <dbl>        <dbl>        <dbl>
## 1 t-test    0.0000000163 0.0000000517 0.0000000104
## 2 bootstrap 0            0            0
```

Both the p values for for bootstrapping and t-test are both 0 (or close) and both or less than 0.05 the significance level.

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

```
# Compare quantile of null resamples to actual t critical values
tibble(
  stat = c("bootstrap", "t-test"),
  closer = c(quantile(resamples.null.closer$t, 0.05), qt(0.05, df = n_closer - 1)),
  further = c(quantile(resamples.null.further$t, 0.05), qt(0.05, df = n_further - 1)),
  diff = c(quantile(resamples.null.diff$t, 0.05), qt(0.05, df = n_diff - 1))
)

## # A tibble: 2 x 4
##   stat      closer further  diff
##   <chr>      <dbl>   <dbl> <dbl>
## 1 bootstrap  -1.63   -1.66 -1.54
## 2 t-test     -1.71   -1.71 -1.71
```

The values for the bootstrap method are slightly lower in the closer and difference data, and slightly higher in the further data. The values for closer and difference data is noticeably lower having values of -1.59 and -1.57 respectively compared to -1.71.

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

```
# Resample means
resample_means_closer <- replicate(R, mean(sample(closer, n_closer, replace = TRUE)))
resample_means_further <- replicate(R, mean(sample(further, n_further, replace = TRUE)))
resample_means_diff <- replicate(R, mean(sample(diff, n_diff, replace = TRUE)))

# Bootstrap CIs (percentile method)
ci_boot_closer <- quantile(resample_means_closer, probs = c(0.025, 0.975))
ci_boot_further <- quantile(resample_means_further, probs = c(0.025, 0.975))
ci_boot_diff <- quantile(resample_means_diff, probs = c(0.025, 0.975))

# t-test CIs
ci_ttest_closer <- t.test(closer, mu = 0)$conf.int
ci_ttest_further <- t.test(further, mu = 0)$conf.int
ci_ttest_diff <- t.test(diff, mu = 0)$conf.int

tibble(
  method = c("t-test", "bootstrap"),
  CI_closer_low = c(ci_ttest_closer[1], ci_boot_closer[1]),
  CI_closer_high = c(ci_ttest_closer[2], ci_boot_closer[2]),
  CI_further_low = c(ci_ttest_further[1], ci_boot_further[1]),
  CI_further_high = c(ci_ttest_further[2], ci_boot_further[2]),
  CI_diff_low = c(ci_ttest_diff[1], ci_boot_diff[1]),
  CI_diff_high = c(ci_ttest_diff[2], ci_boot_diff[2])
)

## # A tibble: 2 x 7
##   method CI_closer_low CI_closer_high CI_further_low CI_further_high CI_diff_low
##   <chr>          <dbl>          <dbl>          <dbl>           <dbl>       <dbl>
## 1 t-test         0.117          0.195         -0.257          -0.149       0.272
## 2 boots~         0.121          0.193         -0.256          -0.155       0.282
## # i 1 more variable: CI_diff_high <dbl>
```

The difference in confidence intervals between t test and bootstraps is very small. All of the differences are less than 0.01.

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
mu0 <- 0

## -- Closer data --
rand_closer <- tibble(xbars = rep(NA, R))
x.shift <- closer - mu0

for (i in 1:R) {
  curr.rand <- x.shift * sample(c(-1, 1), size = length(x.shift), replace = TRUE)
```

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

rand_closer <- rand_closer |>
  mutate(xbars = xbars + mu0)

## -- Further data --
rand_further <- tibble(xbars = rep(NA, R))
x.shift <- further - mu0

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

rand_further <- rand_further |>
  mutate(xbars = xbars + mu0)

## -- Difference data --
rand_diff <- tibble(xbars = rep(NA, R))
x.shift <- diff - mu0

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

rand_diff <- rand_diff |>
  mutate(xbars = xbars + mu0)
```

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

```r
# Function to compute p-value
compute_pval <- function(data, rand_dist, mu0 = 0) {
  delta <- abs(mean(data) - mu0)
  low <- mu0 - delta
  high <- mu0 + delta

  mean(rand_dist$xbars <= low) + mean(rand_dist$xbars >= high)
}

# Calculate p-values
pval_closer <- compute_pval(closer, rand_closer)
pval_further <- compute_pval(further, rand_further)
pval_diff <- compute_pval(diff, rand_diff)

# View results
pval_closer

## [1] 0

pval_further

## [1] 0

pval_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.

```
# Function to compute two-sided p-value for a given mu0
compute_p_value <- function(data, mu0, R = 10000) {
  x_shift <- data - mu0
  rand_means <- replicate(R, mean(x_shift * sample(c(-1, 1), length(x_shift), replace = TRUE))
  rand_means <- rand_means + mu0
  delta <- abs(mean(data) - mu0)
  low <- mu0 - delta
  high <- mu0 + delta
  mean(rand_means <= low | rand_means >= high)
}

# Function to find confidence interval bounds
find_ci_bounds <- function(data, step = 0.0001, alpha = 0.05) {
  sample_mean <- mean(data)
  # Lower bound
  mu0_lower <- sample_mean
  while (compute_p_value(data, mu0_lower) < alpha) {
    mu0_lower <- mu0_lower - step
  }
  # Upper bound
  mu0_upper <- sample_mean
  while (compute_p_value(data, mu0_upper) < alpha) {
    mu0_upper <- mu0_upper + step
  }
  c(lower = mu0_lower, upper = mu0_upper)
}

# Compute confidence intervals
ci_diff <- find_ci_bounds(diff)
ci_closer <- find_ci_bounds(closer)
ci_further <- find_ci_bounds(further)

# Display results
cat("95% Confidence Interval for diff: [", ci_diff["lower"], ", ", ci_diff["upper"], "]\n")

## 95% Confidence Interval for diff: [ 0.3589475 ,  0.3589475 ]

cat("95% Confidence Interval for closer: [", ci_closer["lower"], ", ", ci_closer["upper"], "]\

## 95% Confidence Interval for closer: [ 0.1562231 ,  0.1562231 ]

cat("95% Confidence Interval for further: [", ci_further["lower"], ", ", ci_further["upper"], 

## 95% Confidence Interval for further: [ -0.2027244 ,  -0.2027244 ]
```

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.

```r
# Parameters
R <- 1000   # bootstrap resamples
S <- 1000   # simulations
n <- 30     # sample size
alpha <- 0.05
mu <- 0     # true mean

# Store rejections
type1_results <- tibble(
  reject_fixed = logical(S),
  reject_flex  = logical(S)
)

for (i in 1:S) {
  x <- rnorm(n, mean = mu, sd = 1)
  s <- sd(x)
  xbar <- mean(x)

  resamples <- replicate(R, sample(x, n, replace = TRUE))
  xbars <- colMeans(resamples)
  srs <- apply(resamples, 2, sd)

  t_fixed <- (xbars - 0) / (s / sqrt(n))
  t_flex <- (xbars - 0) / (srs / sqrt(n))

  t_obs_fixed <- (xbar - 0) / (s / sqrt(n))
  t_obs_flex <- (xbar - 0) / (srs / sqrt(n))

  type1_results$reject_fixed[i] <- mean(abs(t_fixed) >= abs(t_obs_fixed)) < alpha
  type1_results$reject_flex[i]  <- mean(abs(t_flex) >= abs(t_obs_flex)) < alpha
}

# Type I error rates
type1_results %>%
  summarise(
    type1_fixed = mean(reject_fixed),
    type1_flex  = mean(reject_flex)
  )

## # A tibble: 1 x 2
##   type1_fixed type1_flex
##         <dbl>      <dbl>
## 1           0          0
```

The type 1 error is 0 for both standard deviations which means they have a p value of close to 0 which makes sense given the p values we found in prior parts of the lab.

(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?

```r
# Store CI coverage results
ci_results <- tibble(
  cover_fixed = logical(S),
  cover_flex  = logical(S)
)

for (i in 1:S) {
  x <- rnorm(n, mean = mu, sd = 1)
  s <- sd(x)

  resamples <- replicate(R, sample(x, n, replace = TRUE))
  xbars <- colMeans(resamples)
  srs <- apply(resamples, 2, sd)

  # Fixed-s approach CI
  ci_fixed <- quantile(xbars, probs = c(0.025, 0.975))

  # Flex-s approach: use pivot or adjusted method
  # Here: percentile method on resampled t-stats
  t_flex <- (xbars - mean(x)) / (srs / sqrt(n))
  t_quantiles <- quantile(t_flex, probs = c(0.025, 0.975))
  ci_flex <- mean(x) - rev(t_quantiles) * (s / sqrt(n))

  ci_results$cover_fixed[i] <- mu >= ci_fixed[1] && mu <= ci_fixed[2]
  ci_results$cover_flex[i] <- mu >= ci_flex[1] && mu <= ci_flex[2]
}

# CI coverage
ci_results %>%
  summarise(
    coverage_fixed = mean(cover_fixed),
    coverage_flex  = mean(cover_flex)
  )

## # A tibble: 1 x 2
##   coverage_fixed coverage_flex
##            <dbl>         <dbl>
## 1          0.925         0.939
```

The coverage of the flex (using the standard deviation of the resamples sr) is slightly higher than the coverage of fixed (standard deviation of original sample), both of which are around 0.95 which makes sense because the significance level is 0.05.

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.