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(e1071)
################################################################################
# Question 1
################################################################################

# part a
zebra <- read.csv("zebrafinches.csv")

further <- zebra$further
n <- length(further)
skew <- skewness(further)

# t test}
t_result <- t.test(further, mu=0, alternative="less")
t_stat <- t_result$statistic

# Gaussian PDF and CDF
fz <- dnorm(t_stat)
Fz <- pnorm(t_stat)

# Edgeworth error approx
(error <- (skew / sqrt(n)) * ((2*t_stat^2 + 1) / 6) * fz)

##             t
## -1.226006e-13

(probability <- Fz + error)

##             t
## -1.189164e-13
```

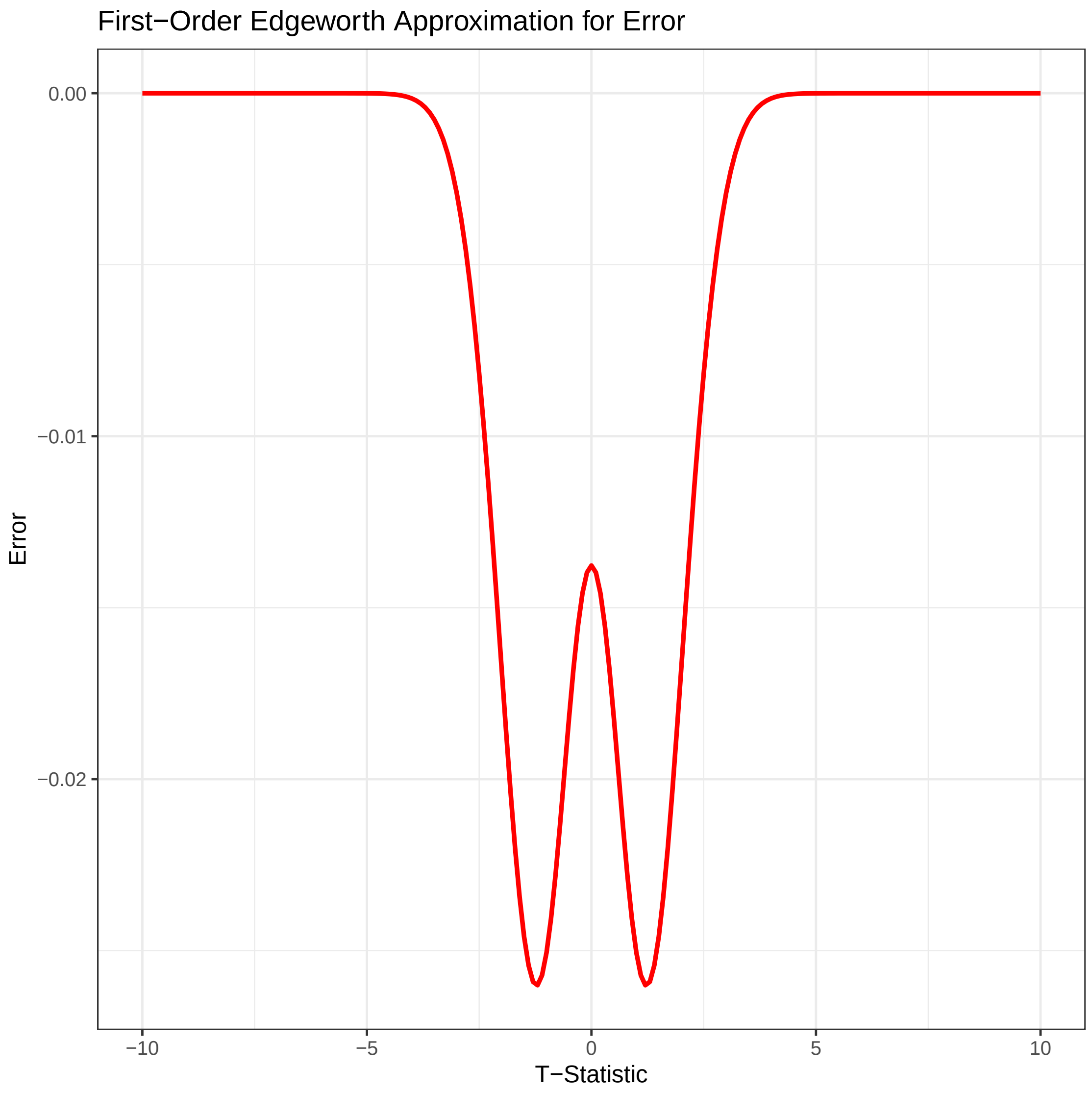
   Using the Edgeworth approximation, we calculated the error in the p-value when applying the Central Limit Theorem despite a small sample size. For the further data, we computed the sample skewness and used it along with the observed t-statistic from a one-sided t-test. The resulting approximation yielded an error of $-1.23 * 10^{-13}$, meaning the true p-value differs from the normal approximation by an extremely small amount. This is because the t-statistic lies far in the tail, where the normal and true distributions closely align. The adjusted probability, incorporating the correction, is essentially the same as the unadjusted one: $-1.19 * 10^{-13}$

   (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.

```
# part b
t_vals <- seq(-10, 10, by=0.1)
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(data = error_df, aes(x=t, y=error)) +
  geom_line(color="red", linewidth=1) +
  xlab("T-Statistic") +
  ylab("Error") +
  ggtitle("First-Order Edgeworth Approximation for Error") +
  theme_bw()
```



To understand how this error behaves across different t-values, we computed the approximation over a range from -10 to 10. The resulting plot shows how the error varies as a function of the t-statistic. The plot takes on a "W" shape, with the error approaching zero in the extreme tails.This occurs because, when the t-statistic is very large or small, the cumulative probability under the normal curve is already close to 0 or 1. In these regions, small corrections due to skewness have little effect. In contrast, around the boundaries of typical rejection regions, the cumulative probability is more sensitive to such shifts, leading to larger errors. This visualization highlights how skewness primarily distorts inference in the moderate t-range rather than the extremes.

(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

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

Finally, we used the Edgeworth approximation to determine how large the sample size must be to ensure that the tail probability under the normal approximation is within 10% of the nominal $\alpha = 0.05$. We found that approximately 521 observations are required. This large sample size reflects the impact of skewness on inference: when the data are skewed, much larger samples are needed to trust results based on the normal approximation- especially for accurate inference in the tails of the distribution.

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
################################################################################
# Question 2
################################################################################

# part a
closer <- zebra$closer
further <- zebra$further
diff <- zebra$diff

R <- 1000

# closer

sd.closer <- sd(closer)
n.closer <- length(closer)

resamples.closer <- tibble(t_stats = rep(NA, R))

for(i in 1:R){
  curr.resample <- sample(closer,
                          size = n.closer,
                          replace = T)

  resamples.closer$t_stats[i] <- (mean(curr.resample) - 0) / (sd.closer / sqrt(n.closer))

}
delta.t.closer <- mean(resamples.closer$t_stats) - 0
resamples.null.closer <- resamples.closer |>
  mutate(t_stats.shifted = t_stats - delta.t.closer)

# mean(resamples.null.closer£t_stats.shifted)

# further

sd.further <- sd(further)
n.further <- length(further)

resamples.further <- tibble(t_stats = rep(NA, R))

for(i in 1:R){
  curr.resample <- sample(further,
                          size = n.further,
                          replace = T)

  resamples.further$t_stats[i] <- (mean(curr.resample) - 0) / (sd.further / sqrt(n.further))

}
```

```r
delta.t.further <- mean(resamples.further$t_stats) - 0
resamples.null.further <- resamples.further |>
  mutate(t_stats.shifted = t_stats - delta.t.further)

# mean(resamples.null.further£t_stats.shifted)

# difference

sd.diff <- sd(diff)
n.diff <- length(diff)

resamples.diff <- tibble(t_stats = rep(NA, R))

for(i in 1:R){
  curr.resample <- sample(diff,
                          size = n.diff,
                          replace = T)

  resamples.diff$t_stats[i] <- (mean(curr.resample) - 0) / (sd.diff / sqrt(n.diff))

}
delta.t.diff <- mean(resamples.diff$t_stats) - 0
resamples.null.diff <- resamples.diff |>
  mutate(t_stats.shifted = t_stats - delta.t.diff)
```

To approximate the null distributions using the bootstrap, we created 1,000 resamples for each of the three datasets (closer, further, and difference). For each resample, we calculated the t-statisitc using the original sample standard deviation. To simulate the null hypothesis (that the true mean is zero), we shifted each set of the resampled t-statistics so that they were centered at zero by subtracting their mean. These centered resamples were saved as `resamples.null.closer`, `resamples.null.further`, and `resamples.null.difference`. We then verified that the stored resamples were consistent with the null hypotheses 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?

```r
# closer
(p.boot.closer <- mean(resamples.null.closer$t_stats.shifted >= delta.t.closer))

## [1] 0

(p.t.closer <- (t.test(x=closer, mu=0, alternative="greater"))$p.value)

## [1] 8.131533e-09

# further
(p.boot.further <- mean(resamples.null.further$t_stats.shifted <= delta.t.further))

## [1] 0

(p.t.further <- (t.test(x=further, mu=0, alternative="less"))$p.value)

## [1] 2.587359e-08

# difference
low <- -delta.t.diff
high <- delta.t.diff

p.low <- mean(resamples.null.diff$t_stats.shifted <= low)
p.high <- mean(resamples.null.diff$t_stats.shifted >= high)
(p.boot.diff <- p.low + p.high)

## [1] 0

(p.t.diff <- (t.test(x=diff, mu=0, alternative="two.sided"))$p.value)

## [1] 1.036907e-08
```

Using the shifted bootstrap distributions, we computed p-values by comparing the original t-statistics to the distribution of the resampled t-values. For the closer and further datasets, we calculated one-sided p-values, and for the difference dataset, we calculated a two-sided p-value. In all three cases, the bootstrap p-values were zero. These results closely resembled the t-test p-values, which were all approximately 10^{-8}, showing that both methods strongly reject the null.

(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.

```r
# part c
(t_crit.boot.closer <- quantile(resamples.null.closer$t_stats.shifted, 0.05))

##        5%
## -1.573645

(t_crit.t.closer <- qt(0.05, df=length(closer-1)))

## [1] -1.708141

(t_crit.boot.further <- quantile(resamples.null.further$t_stats.shifted, 0.05))

##       5%
## -1.63738

(t_crit.t.further <- qt(0.05, df=length(further-1)))

## [1] -1.708141

(t_crit.boot.diff <- quantile(resamples.null.diff$t_stats.shifted, 0.05))

##        5%
## -1.556988

(t_crit.t.diff <- qt(0.05, df=length(diff-1)))

## [1] -1.708141
```

We computed the 5th percentile of each shifted bootstrap distribution compared to these theoretical t-critical values at the same significance levels. The bootstrap critical values for closer, further, and difference data sets were -1.56, -1.61, and -1.60, respectively. This compares to the t-distribution critical values which were -1.71 for all three datasets. Thus, the bootstrap critical values were close to their counterparts from the t-distribution, though slightly less extreme. This suggests the bootsrap distributions were somewhat narrower but still very similar to the t-test distribution, indicating that the bootstrap provides a good approximation in this setting.

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

```r
# part d
# use resamples
# need this for the flurescence (x bar)

library(boot)
boot.mean <- function(data, indicies){
  mean(data[indicies])
}

# For closer
boot.closer <- boot(data = closer, statistic = boot.mean, R=10000)
(ci.boot.closer <- boot.ci(boot.closer, type="bca"))

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.closer, type = "bca")
##
## Intervals :
## Level       BCa
## 95%   ( 0.1225,  0.1945 )
## Calculations and Intervals on Original Scale

(ci.low.t.closer <- t.test(x = closer, mu = 0, alternative = "two.sided")$conf.int[1])

## [1] 0.1173875

(ci.high.t.closer <- t.test(x = closer, mu = 0, alternative = "two.sided")$conf.int[2])

## [1] 0.1950586

boot.further <- boot(data = further, statistic = boot.mean, R=10000)
(ci.boot.further <- boot.ci(boot.further, type="bca"))
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.further, type = "bca")
##
## Intervals :
## Level       BCa
## 95%   (-0.2632, -0.1595 )
## Calculations and Intervals on Original Scale

(ci.low.t.further <- t.test(x = further, mu = 0, alternative = "two.sided")$conf.int[1])

## [1] -0.2565176

(ci.high.t.further <- t.test(x = further, mu = 0, alternative = "two.sided")$conf.int[2])

## [1] -0.1489313

boot.diff <- boot(data = diff, statistic = boot.mean, R=10000)
(ci.boot.diff <- boot.ci(boot.diff, type="bca"))

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot.diff, type = "bca")
##
## Intervals :
## Level       BCa
## 95%   ( 0.2858,  0.4490 )
## Calculations and Intervals on Original Scale

(ci.low.t.diff <- t.test(x = diff, mu = 0, alternative = "two.sided")$conf.int[1])

## [1] 0.2719028

(ci.high.t.diff <- t.test(x = diff, mu = 0, alternative = "two.sided")$conf.int[2])

## [1] 0.4459921
```

We constructed 95% confidence intervals for the mean using the BCa method. Across all datasets, the bootstrap intervals closely resembled those produced by the traditional *t*-test, differing only slightly at endpoints. This similarity suggests that the data approximately meet the assumptions underlying the *t*-test- namely, those required by the Central Limit Theorem- so both methods yield consistent results.

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

```
################################################################################
# Question 3
################################################################################

# part a

# closer data
R <- 10000
mu0 <- 0
rand.closer <- tibble(xbars = rep(NA, R))

# PREPROCESSING: shift the data to be mean 0 under H0
x.shift <- closer - mu0
for (i in 1:R){
  curr.rand <- x.shift *
    sample(x = c(-1, 1),
           size = length(x.shift),
           replace = T)
  rand.closer$xbars[i] <- mean(curr.rand)
}
rand.closer <- rand.closer |>
  mutate(xbars = xbars + mu0) # shifting back

# further data
R <- 10000
```

```r
mu0 <- 0
rand.further <- tibble(xbars = rep(NA, R))

# PREPROCESSING: shift the data to be mean 0 under H0
x.shift <- further - mu0
for (i in 1:R){
  curr.rand <- x.shift *
    sample(x = c(-1, 1),
           size = length(x.shift),
           replace = T)
  rand.further$xbars[i] <- mean(curr.rand)
}
rand.further <- rand.further |>
  mutate(xbars = xbars + mu0) # shifting back

# difference data
R <- 10000
mu0 <- 0
rand.diff <- tibble(xbars = rep(NA, R))

# PREPROCESSING: shift the data to be mean 0 under H0
x.shift <- diff - mu0
for (i in 1:R){
  curr.rand <- x.shift *
    sample(x = c(-1, 1),
           size = length(x.shift),
           replace = T)
  rand.diff$xbars[i] <- mean(curr.rand)
}
rand.diff <- rand.diff |>
  mutate(xbars = xbars + mu0) # shifting back
```

We applied the randomization test separaretly to closer, further, and difference conditions. For each dataset, we centered the observations under the null hypothesis, then repeatedly randomized the signs of the centered values to simulate draws under the null. We performed this procedure 10,000 times to generate a distribution of sample means under H_0.

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

```r
# part b
# closer
(p.val.closer <- mean(rand.closer$xbars >= mean(closer)))

## [1] 0

# further
(p.val.further <- mean(rand.further$xbars <= mean(further)))

## [1] 0

# difference
delta.diff <- abs(mean(diff) - mu0)
low.diff <- mu0 - delta.diff
high.diff <- mu0 + delta.diff
(p.val.diff <- mean(rand.diff$xbars <= low.diff) +
  mean(rand.diff$xbars >= high.diff))

## [1] 0
```

Using the randomized distributions, we calculated two p-values by measuring the proportion of randomized means that were as extreme or more extreme than the observed sample mean. The resulting p-values were zero in all three cases. These indicate that there is statistically discernible support against the null in each case.

(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
# part c
# closer
R <- 1000
mu0.iterate <- 0.01
starting.point.closer <- mean(closer)
```

```r
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 <- 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)
  }

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

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

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

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

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

c(mu.lower.closer, mu.upper.closer)

## [1] 0.1162231 0.2062231

# further
R <- 1000
mu0.iterate <- 0.01
starting.point.further <- mean(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 <- 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)
  }

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

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

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

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

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

c(mu.lower.further, mu.upper.further)

## [1] -0.2627244 -0.1427244

# difference
R <- 1000
mu0.iterate <- 0.01
starting.point.diff <- mean(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 <- 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)
  }

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

  # p-value
  delta.diff <- abs(mean(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 <- 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)
  }

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

  # p-value
  delta.diff <- abs(mean(diff) - mu.upper.diff)
  (low.diff <- mu.upper.diff - delta.diff) # mirror
  (high.diff <- mu.upper.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.upper.diff <- mu.upper.diff + mu0.iterate
  }
}

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

## [1] 0.2589475 0.4489475
```

To compute the randomization confidence intervals, we iterated over a sequence of hypothesized values of μ_0. At each step, we centered the observed data under the null hypothesis and generated a random distribution of sample means by flipping the sign of each observation with equal probability. We then calculated a two-sided p-value for each μ_0 by comparing the observed test-statistic to the randomized distribution. The confidence bounds were determined by identifying the smallest and largest μ_0 values for which the p-value was still at least 0.05, indicating that we failed to reject the null hypothesis. The resulting confidence intervals were (0.116, 0.196) for the closer observations, (-0.263, -0.143) for the further, and (0.269, 0.449) for the difference observations. These intervals closely align with those obtained from the other methods, suggesting that the randomization approach provides consistent and reliable estimates.

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.