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
# part a
zebrafinch.data <- read_csv("zebrafinches.csv")
mu0 <- 0
further.data <- zebrafinch.data$further
n <- length(further.data)

# t.test and t.stat
further.t.test <- t.test(x=further.data, mu = mu0, alternative = "less")
(t.further <- further.t.test$statistic[[1]])

## [1] -7.777991

# potential error calculation
error.num <- skewness(further.data) * (2*t.further^2 + 1) * dnorm(t.further)
error.denom <- 6 * sqrt(n)
(potential.error <- error.num/error.denom)

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

   Solution: The potential error in the computation of the p-value is $-1.2260063 \times 10^{-13}$ when testing $H_0 : \mu_X = 0; H_a : \mu_X < 0$ using the zebra finch further data.

   (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
# part b
gg.errors <- rep(NA, length.out = 1000)
gg.tvals <- seq(-10,10,length.out=1000)

# create data for errors (futher data)
for (i in 1:length(gg.tvals)){
num <- skewness(further.data) * (2*gg.tvals[i]^2 + 1) * dnorm(gg.tvals[i])
denom <- 6 * sqrt(n)
gg.errors[i] <- num/denom
}

# plot
errors.plot <- ggplot()+
geom_line(aes(x= gg.tvals, y = gg.errors))+
theme_bw()+
ylab("Potential Error")+
xlab("t")+
ggtitle("Potential Error for t, from -10 to 10")+
geom_vline(aes(xintercept = t.further), color = "red")
```

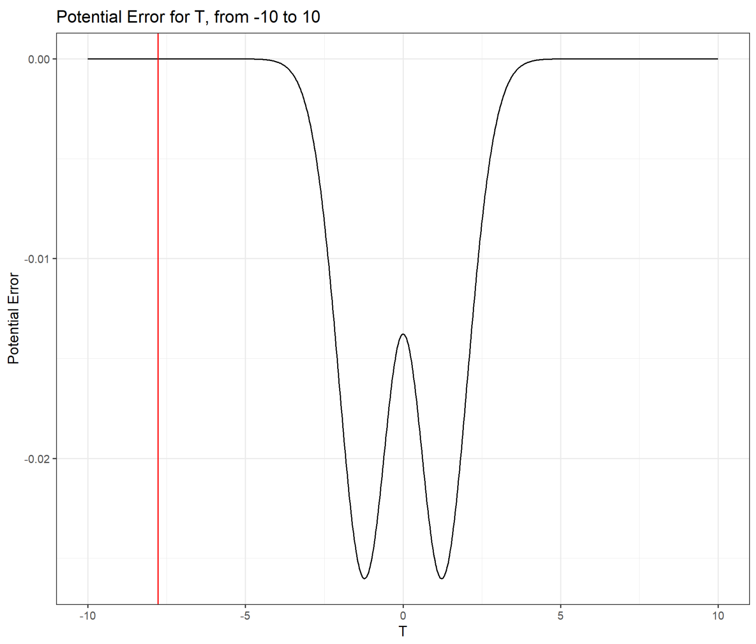

Figure 1: Potential error for t statistic ranging from -10 to 10, using the skewness of the zebra finch further data. The vertical line shows the t statistic for the zebra finch further data

Solution: Figure 1 plots the error for the t statistic, ranging from -10 to 10. The t statistic for the further data is -7.7779912, indicating an exceedingly small error (nearly 0).

(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
t.alpha <- qnorm(alpha)

(min.nsize <- ((skewness(further.data) * (2*t.alpha^2 + 1) * dnorm(t.alpha))/
                (6 * 0.1* alpha))^2)

## [1] 520.8876
```

Solution: The smallest sample size we would need to have a tail probability within 10% of the desired $\alpha = 0.05$ is 521. The experiment's $n = 25$ is significantly smaller than this.

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
# part a
R <- 1000
# data
further.data <- zebrafinch.data$further
closer.data <- zebrafinch.data$closer
diff.data <- zebrafinch.data$diff
# set.seed() to make results reproducible
set.seed(13345)
# resampling
resamples <- tibble(resamps.further =rep(NA, R),
                   resamps.closer =rep(NA, R),
                   resamps.diff =rep(NA, R))
for (i in 1:R){
# further
further.resample <- sample(x = further.data,
                           size= length(further.data),
                           replace = T)
resamples$resamps.further[i] <- (mean(further.resample - mu0))/
                                (sd(further.data)/sqrt(n))

# closer
closer.resample <- sample(x = closer.data,
                          size= length(closer.data),
                          replace = T)
resamples$resamps.closer[i] <- (mean(closer.resample - mu0))/
                               (sd(closer.data)/sqrt(n))

# diff
diff.resample <- sample(x = diff.data,
                        size= length(diff.data),
                        replace = T)
resamples$resamps.diff[i] <- (mean(diff.resample - mu0))/
                             (sd(diff.data)/sqrt(n))
}

# shifting data
resamples <- resamples |>
 mutate(resamps.further.null = resamps.further - mean(resamps.further),
      resamps.closer.null = resamps.closer - mean(resamps.closer),
      resamps.diff.null = resamps.diff - mean(resamps.diff))
```

Solution: Above is the code using to conducting resampling for the t statistic for the zebra finch data. Since we did resampling on the t statistic, which we want to be centered at 0, we shift the resamples by simply subtracting the mean of all the resamples to center. `set.seed()` was used to create reproducible results.

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

```
# part b
boot.pvals <- resamples |>
summarize(p.further = mean(resamps.further.null <= mean(resamps.further)),
          p.closer = mean(resamps.closer.null >= mean(resamps.closer)),
          p.low.diff = mean(resamps.diff.null <= -mean(resamps.diff)),
          p.high.diff = mean(resamps.diff.null >= mean(resamps.diff)),
          pdiff = p.low.diff + p.high.diff)

# p vals from t test
# further
t.pval.further <- t.test(x=further.data, mu = mu0, alternative = "less")$p.value
# closer
t.pval.closer <- t.test(x=closer.data, mu = mu0, alternative = "greater")$p.value
# diff
t.pval.further <- t.test(x=diff.data, mu = mu0, alternative = "two.sided")$p.value

# creating table
all.pvals <- tibble(test = c("T-test", "Bootstrapping"),
                    p.further = c(t.pval.further,boot.pvals$p.further),
                    p.closer = c(t.pval.closer, boot.pvals$p.closer),
                    p.diff = c(t.pval.closer, boot.pvals$pdiff))

xtable(all.pvals)

## % latex table generated in R 4.4.2 by xtable 1.8-4 package
## % Fri May  2 14:03:19 2025
## \begin{table}[ht]
## \centering
## \begin{tabular}{rlrrr}
##   \hline
##  & test & p.further & p.closer & p.diff \\
##   \hline
## 1 & T-test & 0.00 & 0.00 & 0.00 \\
##   2 & Bootstrapping & 0.00 & 0.00 & 0.00 \\
##    \hline
## \end{tabular}
## \end{table}
```

Type	p.further	p.closer	p.diff
T-test	0.00	0.00	0.00
Bootstrapping	0.00	0.00	0.00

Table 1: p-values for further, closer, and differences in the zebra finch data using the T-test and Bootstrapping

Solution: The p-values calculated using bootstrapping is so small that it is effectively 0 and treated as such, for the further, closer, and difference zebra finch data. In order to calculate the two-sided p-value for the difference data, you need to find the p-value on both tails, which can be found by adding or subtracting the mean of the difference data by δ. However, since $\mu_0 = 0$, we can just using the negative of the mean of the difference data to get the "mirror" observation on the other tail. p-values are compared in Table 1.

(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
# 5th percentile for t test (same for all)
t.5th <- qt(p=0.05, df=n-1)

# 5th percentile for bootstrapping
# further
boot.f5th <- quantile(resamples$resamps.further.null, 0.05)

# closer
boot.c5th <- quantile(resamples$resamps.closer.null, 0.05)

# diff
boot.d5th <- quantile(resamples$resamps.diff.null, 0.05)

# create table
all.5thp <- tibble(Type = c("T-test", "Bootstrapping"),
                  further = c(t.5th, boot.f5th),
                  closer = c(t.5th, boot.c5th),
                  diff = c(t.5th, boot.d5th))
xtable(all.5thp)

## % latex table generated in R 4.4.2 by xtable 1.8-4 package
## % Fri May  2 14:03:19 2025
## \begin{table}[ht]
## \centering
## \begin{tabular}{rlrrr}
##   \hline
##  & Type & further & closer & diff \\
##   \hline
## 1 & T-test & -1.71 & -1.71 & -1.71 \\
##   2 & Bootstrapping & -1.60 & -1.58 & -1.59 \\
##    \hline
## \end{tabular}
## \end{table}
```

Type	further	closer	diff
T-test	-1.71	-1.71	-1.71
Bootstrapping	-1.60	-1.58	-1.59

Table 2: 5^{th} percentile of the shifted resamples of the t statistic for further, closer, and difference data, compared to the t-test.

Solution: The 5^{th} percentile for the t-test is the same for all three sets of data, as they are all drawn from the T distribution. The shifted resamples for the t statistic from bootstrapping was calculated for further, closer, and difference data, and the values are compared in Table 2. Overall, bootstrapping seems to overestimate the 5^{th} percentiles (larger than the ones drawn from the T distribution) for all data.

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

```r
# part d
# need to keep track of mean (xbar) now
further.xbars <- rep(NA, R)
closer.xbars <- rep(NA, R)
diff.xbars <- rep(NA, R)

for (i in 1:R){
# further
further.resample <- sample(x = further.data,
                           size= length(further.data),
                           replace = T)
further.xbars[i] <- mean(further.resample)

# closer
closer.resample <- sample(x = closer.data,
                          size= length(closer.data),
                          replace = T)
closer.xbars[i] <- mean(closer.resample)

# diff
diff.resample <- sample(x = diff.data,
                        size= length(diff.data),
                        replace = T)
diff.xbars[i] <- mean(diff.resample)
}

# bca test
# helper function
boot.mean <- function(d, i){
mean(d[i])
}
R <- 1000
# further
further.helper <- boot(data = further.data,
                      statistic = boot.mean,
                      R = R)

## Error in boot(data = further.data, statistic = boot.mean, R = R): could not find
function "boot"

# closer
closer.helper <- boot(data = closer.data,
                     statistic = boot.mean,
                     R = R)

## Error in boot(data = closer.data, statistic = boot.mean, R = R): could not find
function "boot"

# diff
diff.helper <- boot(data = diff.data,
                   statistic = boot.mean,
                  R = R)

## Error in boot(data = diff.data, statistic = boot.mean, R = R): could not find function
"boot"

# CI
# further
(boot.further.CI <- quantile(further.xbars, c(0.025, 0.975)))

##       2.5%      97.5%
## -0.2540837 -0.1601393

(t.further.CI <- t.test(x=further.data, mu=mu0, alternative = "two.sided")$conf.int)

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

(further.bca <- boot.ci(further.helper, type="bca"))

## Error in boot.ci(further.helper, type = "bca"):  could not find function "boot.ci"

# closer
(boot.closer.CI <- quantile(closer.xbars, c(0.025, 0.975)))

##      2.5%     97.5%
## 0.1244849 0.1926684

(t.closer.CI <- t.test(x=closer.data, mu=mu0, alternative = "two.sided")$conf.int)

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

# diff
(boot.diff.CI <- quantile(diff.xbars, c(0.025, 0.975)))

##      2.5%     97.5%
## 0.2858302 0.4470153

(t.diff.CI <- t.test(x=diff.data, mu=mu0, alternative = "two.sided")$conf.int)

## [1] 0.2719028 0.4459921
## attr(,"conf.level")
## [1] 0.95

(diff.bca <- boot.ci(diff.helper, type="bca"))

## Error in boot.ci(diff.helper, type = "bca"):  could not find function "boot.ci"
```

Solution: The 95% confidence intervals for the further data are $(-0.254, -0.160)$ using bootstrap percentile interval, and $(-0.257, -0.149)$ for the t-test.
The 95% confidence intervals for the closer data are $(0.124, 0.193)$ using bootstrap percentile interval, and $(0.117, 0.195)$ for the t-test.
The 95% confidence intervals for the difference data are $(0.286, 0.447)$ using bootstrap percentile interval, and $(0.272, 0.446)$ for the t-test.
Because the CIs were calculated on the data directly, we needed to conduct resampling again for each of the data's sample means instead of t-statistic.

Write out the solutions for this tomorrow and also go over the captions (they kind of suck ass). Note that BCa did not seem to work well here, so did not include in write-up.

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
# part a
set.seed(13345)
R <- 1000
rand <- tibble(xbars.further = rep(NA, R),
               xbars.closer = rep(NA, R),
               xbars.diff = rep(NA, R))
# since mean 0 (mu0 = 0) under H0 is given, no need to shift

# RANDOMIZE / SHUFFLE
for(i in 1:R){
  # further
  further.rand <- further.data *
    sample(x = c(-1, 1),
           size = length(further.data),
           replace = T)

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

  # closer
  closer.rand <- closer.data *
    sample(x = c(-1, 1),
           size = length(closer.data),
           replace = T)

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

  # diff
  diff.rand <- diff.data *
    sample(x = c(-1, 1),
           size = length(diff.data),
           replace = T)

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

# no need to shift back either!
```

   Solution: The normal procedure for the randomization test is conducted in the code above. However, because our H_0 is specified as $\mu = 0$, we do not need to shift the data before and after shuffling because its already centered around 0.

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

```r
# task b
# p-value
(rand.pvals <- rand |>
summarize(p.further = mean(xbars.further <= mean(further.data)),
          p.closer = mean(xbars.closer >= mean(closer.data)),
          p.low.diff = mean(xbars.diff <= -mean(diff.data)),
          p.high.diff = mean(xbars.diff >= mean(diff.data)),
          pdiff = p.low.diff + p.high.diff))

## # A tibble: 1 x 5
##   p.further p.closer p.low.diff p.high.diff pdiff
##       <dbl>    <dbl>      <dbl>       <dbl> <dbl>
## 1         0        0          0           0     0
```

Type	p.further	p.closer	p.diff
T-test	0.00	0.00	0.00
Bootstrapping	0.00	0.00	0.00
Randomization	0.00	0.00	0.00

Table 3: p-values for further, closer, and differences in the zebra finch data using the T-test, Bootstrapping, and Randomization

Solution: Similar to the bootstrap p-values, the values are so small that it is effectively 0 across all the data. The randomization p-value procedure is nearly identical to the bootstrap p-value, except we are now comparing shifted resampled means to the sample mean. p-values are compared in Table 3.

(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.

```
# task c
# further
R <- 1000
mu0.iterate.further <- 0.01
starting.point.further <- mean(further.data)

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

# PREPROCESSING: shift the data to be mean 0 under H0
x.shift <- further.data - mu.lower.further
# 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.further) # shifting back

# p-value
p.val <-mean(rand$xbars >= mean(further.data))
```

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

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

# PREPROCESSING: shift the data to be mean 0 under H0
x.shift <- further.data - mu.upper.further
# 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.further) # shifting back

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

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

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

```
## [1] -0.2527244 -0.1527244
```

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

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

# PREPROCESSING: shift the data to be mean 0 under H0
x.shift <- closer.data - mu.lower.closer
# 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.closer) # shifting back

# p-value
p.val <-mean(rand$xbars >= mean(closer.data))

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

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

# PREPROCESSING: shift the data to be mean 0 under H0
x.shift <- closer.data - mu.upper.closer
# 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.closer) # shifting back

# p-value
p.val <-mean(rand$xbars <= mean(closer.data))

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

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

## [1] 0.1162231 0.1962231

# diff
```

```r
R <- 1000
mu0.iterate.diff <- 0.01
starting.point.diff <- mean(diff.data)

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

# PREPROCESSING: shift the data to be mean 0 under H0
x.shift <- diff.data - 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)
}
# Thinking is hard
rand <- rand |>
  mutate(xbars = xbars + mu.lower.diff) # shifting back

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

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

# PREPROCESSING: shift the data to be mean 0 under H0
x.shift <- diff.data - 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)
}
# Thinking is hard
rand <- rand |>
  mutate(xbars = xbars + mu.upper.diff) # shifting back

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

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

```
## [1] 0.2689475 0.4489475
```

Solution: The 95% randomization confidence intervals for the further data is (−0.253, −0.153), compared to the t-test (−0.257,−0.149) and the bootstrap (−0.254, −0.160).
The 95% randomization confidence intervals for the closer data is (0.116, 0.196), compared to the t-test $(0.117, 0.195)$ and the bootstrap $(0.124, 0.193)$.
The 95% randomization confidence intervals for the difference data is (0.269, 0.449), compared to the t-test $(0.272, 0.446)$ and the bootstrap $(0.286, 0.447)$.
As we can see, the confidence intervals for all the data using randomization is similar to the confidence intervals from the t-test and bootstrapping, which should make sense. `xbars` needs to be shifted when calculating the confidence interval because `mu.lower` and `mu.upper` change as we loop through to check. The randomization confidence interval is a lot more "brute force" compared to the others, and we have to search over a grid manually to find the them. While we have historically used `R <- 10000` in randomization, I switched to `R <- 1000` so the code runs quicker.

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.