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?

```
#################
#  PART A
#################
#initializing data
finch_data = read_csv("zebrafinches.csv")

## Rows:  25 Columns:  3
## -- Column specification --------------------------------------------------------
## Delimiter:  ","
## dbl (3):  closer, further, diff
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.

further_data = finch_data$further
closer_data = finch_data$closer
diff_data = finch_data$diff
skew_further = skewness(further_data)
n=nrow(finch_data)
#Calculating t statistic
t_val = mean(further_data)/(sd(further_data)/sqrt(n))

#Caculating potential error
potential_err = (skew_further/sqrt(n))*((2*t_val^2+1)/6)*dnorm(t_val)
```

   (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
#################
#Initializing
t_vals = seq(from = -10, to = 10, by = 0.01)
R = length(t_vals)
err_tibble = tibble(err=rep(NA,R))

#Looping over each t value
for (i in 1:R){
  t_val = t_vals[i]
  #Calculating potential error
  potential_err = (skew_further/sqrt(n))*((2*t_val^2+1)/6)*dnorm(t_val)

  #Adding error to tibble
  err_tibble$err[i] = potential_err
}
#view(err_tibble)

#Making plot of values
error_plot = ggplot(data=err_tibble, aes(t_vals, err)) +
  geom_line() +
  theme_bw() +
  geom_hline(yintercept = 0) +
  xlab("T Value") +
  ylab("Potential Error")
```

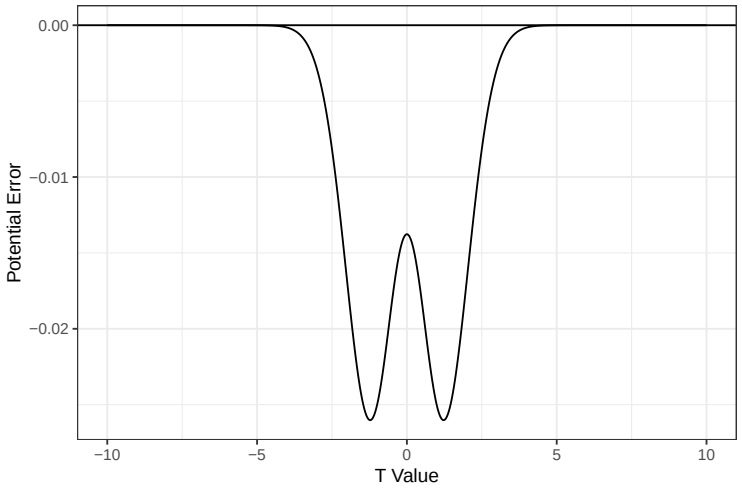

Figure 1: Error Across t

(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

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

```
#################
#  PART C
#################
alpha = 0.05
t_val = qnorm(alpha)
n = ((skew_further/(6*0.1*alpha))* (2*t_val^2+1)*dnorm(t_val))^2
```

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 <- 10000
resamples <- tibble(further = rep(NA, R),
                    closer = rep(NA, R),
                    diff = rep(NA, R))

for(i in 1:R){
  #Making a sample for further
```

```
  curr.resample <- sample(further_data,
                          size = length(further_data),
                          replace = T)

  resamples$further[i] <- mean(curr.resample)/(sd(further_data)/sqrt(length(further_data)))



  #Making a sample for closer
  curr.resample <- sample(closer_data,
                          size = length(closer_data),
                          replace = T)

  resamples$closer[i] <- mean(curr.resample)/(sd(closer_data)/sqrt(length(closer_data)))

  #Making a sample for difference
  curr.resample <- sample(diff_data,
                          size = length(diff_data),
                          replace = T)

  resamples$diff[i] <- mean(curr.resample)/(sd(diff_data)/sqrt(length(diff_data)))
}
resamples_shifted = resamples |>
  mutate(further = further-mean(further_data)/(sd(further_data)/sqrt(25))) |>
  mutate(closer = closer-mean(closer_data)/(sd(closer_data)/sqrt(25))) |>
  mutate(diff = diff-mean(diff_data)/(sd(diff_data)/sqrt(25)))

#view(resamples)
#view(resamples_shifted)
```

(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
##################
pval_further = (mean(resamples_shifted$further <= mean(resamples$further)))/R
pval_closer = (mean(resamples_shifted$closer >= mean(resamples$diff)))/R
pval_diff = (mean(resamples_shifted$diff >= mean(resamples$diff)))/R

pval_further

## [1] 0

pval_closer

## [1] 0

pval_diff

## [1] 0
```

(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.

```
##################
#  PART C
##################
resample_further_5th = quantile(resamples_shifted$further, c(0.05, 0.95))
resample_closer_5th = quantile(resamples_shifted$closer, c(0.05, 0.95))
resample_diff_5th = quantile(resamples_shifted$diff, c(0.05, 0.95))

resample_further_5th

##        5%       95%
## -1.693523  1.588579

resample_closer_5th

##        5%       95%
## -1.598645  1.668597

resample_diff_5th

##        5%       95%
## -1.539859  1.668092
```

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

```
# Confidence Interval
resample_further_ci = quantile(resamples$further, c(0.025, 0.975))
resample_closer_ci = quantile(resamples$closer, c(0.025, 0.975))
resample_diff_ci = quantile(resamples$diff, c(0.025, 0.975))

resample_further_ci

##      2.5%     97.5%
## -9.817407 -5.964390

resample_closer_ci

##      2.5%     97.5%
##  6.413167 10.280816

resample_diff_ci

##      2.5%     97.5%
##  6.683322 10.520937
```

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

```
#################
#  PART A
#################
closer_data = finch_data$closer
diff_data = finch_data$diff

R <- 10000
rand <- tibble(further = rep(NA, R),
               closer = rep(NA, R),
               diff = rep(NA, R))

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

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

#Randomization for closer
x.shift <- closer_data - 0
# RANDOMIZE / SHUFFLE
for(i in 1:R){
  curr.rand <- x.shift *
    sample(x = c(-1, 1),
           size = length(x.shift),
           replace = T)

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


#Randomization for difference
x.shift <- diff_data - 0
# RANDOMIZE / SHUFFLE
for(i in 1:R){
  curr.rand <- x.shift *
    sample(x = c(-1, 1),
           size = length(x.shift),
           replace = T)

  rand$diff[i] <- mean(curr.rand)
}
#view(rand)
```

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

```
#################
#  PART B
#################
# p-value for further
(delta <- abs(mean(further_data) - 0))

## [1] 0.2027244

(low <- 0 - delta) # mirror

## [1] -0.2027244

(high<- 0 + delta)   # xbar

## [1] 0.2027244

pval_further = mean(rand$further <= low) + mean(rand$further >= high)

# p-value for closer
(delta <- abs(mean(closer_data) - 0))

## [1] 0.1562231

(low <- 0 - delta) # mirror

## [1] -0.1562231

(high<- 0 + delta)   # xbar

## [1] 0.1562231

pval_closer = mean(rand$closer <= low) + mean(rand$closer >= high)

# p-value for further
(delta <- abs(mean(diff_data) - 0))

## [1] 0.3589475

(low <- 0 - delta) # mirror

## [1] -0.3589475

(high<- 0 + delta)   # xbar

## [1] 0.3589475

pval_diff = mean(rand$diff <= low) + mean(rand$diff >= high)
```

(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.

```
#################
#  PART C
#################
R <- 1000
mu0.iterate <- 0.001
starting.point <- mean(further_data)

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

# PREPROCESSING: shift the data to be mean 0 under H0
x.shift <- further_data - 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)
}
```

```r
# Thinking is hard
rand <- rand |>
  mutate(xbars = xbars + mu.lower) # shifting back

# p-value
(delta <- abs(mean(further_data) - 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
}
}

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

# PREPROCESSING: shift the data to be mean 0 under H0
x.shift <- further_data - 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_data) - 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_ci = c(mu.lower, mu.upper)


###################
#    CLOSER
###################
starting.point <- mean(closer_data)

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

# PREPROCESSING: shift the data to be mean 0 under H0
x.shift <- closer_data - 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
```

```r
# p-value
(delta <- abs(mean(closer_data) - 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
}
}

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

# PREPROCESSING: shift the data to be mean 0 under H0
x.shift <- closer_data - 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(closer_data) - 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_ci = c(mu.lower, mu.upper)



####################
#    DIFFERENCE
####################
starting.point <- mean(diff_data)

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

# PREPROCESSING: shift the data to be mean 0 under H0
x.shift <- diff_data - 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(diff_data) - mu.lower))
(low <- mu.lower - delta) # mirror
```

```r
(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
}
}

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

# PREPROCESSING: shift the data to be mean 0 under H0
x.shift <- diff_data - 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(diff_data) - 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
}
}

diff_ci = c(mu.lower, mu.upper)

further_ci

## [1] -0.2587244 -0.1497244

closer_ci

## [1] 0.1182231 0.1972231

diff_ci

## [1] 0.2709475 0.4459475
```

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.