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(e1071)
dat1 = 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.

dat = dat1$further
t.val = as.numeric(t.test(dat, alternative = "less", mu = 0)$statistic)
cdf.val = pnorm(t.val)
pdf.val = dnorm(t.val)
skew = skewness(dat)
n = length(dat)
(error = cdf.val + (skew/sqrt(n)*(2*(t.val)^2+1)/6*pdf.val))

## [1] -1.189164e-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.

```
total.error = vector()
seq = seq(-10,10,.01)
for(t in seq){
  curr.cdf.val = pnorm(t)
  curr.pdf.val = dnorm(t)
  skew = skewness(dat)
  n = length(dat)
  curr.error = (skew/sqrt(n)*(2*(t)^2+1)/6*curr.pdf.val)
  total.error = append(total.error, curr.error)
}
total.error = tibble(x = total.error) |>
mutate(seq)
```

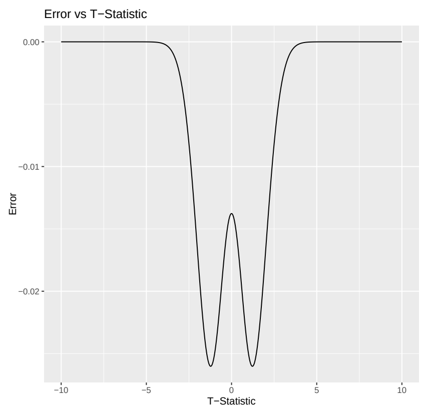

Figure 1:

(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

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

```r
t.val.c = qnorm(.05)
pdf.val.c = dnorm(t.val.c)
(n.c = (skew/(6*(.1*.05))*(2*t.val.c^2+1)*pdf.val.c)^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
#Farther
t.dist.dat = vector()
s = sd(dat)
for(i in 1:10000){
curr.val = (mean(sample(dat, replace = TRUE)) - 0)/(s/sqrt(n))
t.dist.dat = append(t.dist.dat, curr.val)
}
ncp = -1*(mean(t.dist.dat))
resamples.null.farther = t.dist.dat + ncp
(mean(resamples.null.farther))

## [1] 3.934297e-16

#Closer
t.close.dat = vector()
s = sd(dat1$closer)
for(i in 1:10000){
curr.val = (mean(sample(dat1$closer, replace = TRUE)) - 0)/(s/sqrt(n))
t.close.dat = append(t.close.dat, curr.val)
}
ncp2 = -1*(mean(t.close.dat))
resamples.null.closer = t.close.dat + ncp2
(mean(resamples.null.closer))

## [1] 5.778711e-16

#Difference
t.diff.dat = vector()
s = sd(dat1$diff)
for(i in 1:10000){
curr.val = (mean(sample(dat1$diff, replace = TRUE)) - 0)/(s/sqrt(n))
t.diff.dat = append(t.diff.dat, curr.val)
}
ncp3 = -1*(mean(t.diff.dat))
resamples.null.diff = t.diff.dat + ncp3
(mean(resamples.null.diff))

## [1] 8.424261e-16
```

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

```r
#Farther p-value
x.bar.far = mean(t.dist.dat)
(p.val.far = mean(resamples.null.farther <= x.bar.far))

## [1] 0

#Closer p-value
x.bar.close = mean(t.close.dat)
(p.val.close = mean(resamples.null.closer >=x.bar.close))

## [1] 0

#Difference p-value
x.bar.diff = mean(t.diff.dat)
x.bar.diff.lower = 0-x.bar.diff
x.bar.diff.upper = 0+x.bar.diff

(p.val.diff = mean(resamples.null.diff <= x.bar.diff.lower) + mean(resamples.null.diff >= x.bar.diff.upper))

## [1] 0
```

(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.

```r
#Farther
quantile(resamples.null.farther, prob = .025)

##     2.5%
## -2.02765

#Closer
quantile(resamples.null.closer, prob = .025)

##      2.5%
## -1.902959

#Difference
quantile(resamples.null.diff, prob = .025)

##      2.5%
## -1.834268
```

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

```r
#Farther
n = length(dat1$further)
sd.f = sd(dat1$further)
lower.f = quantile(t.dist.dat, prob = .025)
x.bar.lower.f = lower.f/sqrt(n)*sd.f
upper.f = quantile(t.dist.dat, prob = .975)
x.bar.upper.f = upper.f/sqrt(n)*sd.f

c(x.bar.lower.f,x.bar.upper.f)

##       2.5%      97.5%
## -0.2556874 -0.1554911

#Closer
sd.c = sd(dat1$closer)
lower.c = quantile(t.close.dat, prob = .025)
x.bar.lower.c = lower.c/sqrt(n)*sd.c
upper.c = quantile(t.close.dat, prob = .975)
x.bar.upper.c = upper.c/sqrt(n)*sd.c

c(x.bar.lower.c,x.bar.upper.c)

##      2.5%     97.5%
## 0.1200736 0.1926468

#Difference
sd.d = sd(dat1$diff)
lower.d = quantile(t.diff.dat, prob = .025)
x.bar.lower.d = lower.d/sqrt(n)*sd.d
upper.d = quantile(t.diff.dat, prob = .975)
x.bar.upper.d = upper.d/sqrt(n)*sd.d

c(x.bar.lower.d,x.bar.upper.d)
```

```
##      2.5%     97.5%
## 0.2810613 0.4434062
```

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

```
############
# Further Data
############
mu0 = 0
R = 10000
rand = tibble(xbars = rep(NA, 10000))
samp.mean = mean(dat1$further)
x.shift = dat1$further - mu0
#Doing the randomization process
for(i in 1:R){
  curr.samp = x.shift * sample(x = c(-1,1),
                                   size = length(x.shift),
                                   replace = T)
  rand$xbars[i] = mean(curr.samp)
}
#Shifting back
rand1 = rand |>
  mutate(xbars = xbars + mu0)
############
# Closer Data
############
rand.cl = tibble(xbars = rep(NA, 10000))
samp.mean.cl = mean(dat1$closer)
x.shift = dat1$further - mu0
#Doing the randomization process
for(i in 1:R){
  curr.samp = x.shift * sample(x = c(-1,1),
                               size = length(x.shift),
                               replace = T)
  rand.cl$xbars[i] = mean(curr.samp)
}
#Shifting back
rand.cl = rand.cl |>
  mutate(xbars = xbars + mu0)
#############
# Different Data
#############

rand.diff = tibble(xbars = rep(NA, 10000))
samp.mean.diff = mean(dat1$diff)
x.shift = dat1$diff - mu0
#Doing the randomization process
for(i in 1:R){
  curr.samp = x.shift * sample(x = c(-1,1),
                               size = length(x.shift),
                               replace = T)
  rand.diff$xbars[i] = mean(curr.samp)
}
#Shifting back
rand.diff = rand.diff |>
  mutate(xbars = xbars + mu0)
```

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

```
#Farther
(furth.p = mean(rand1<=samp.mean))

## [1] 0

#Closer
(closer.p = mean(rand>=samp.mean.cl))

## [1] 4e-04

#Different
delta = samp.mean.diff
low = mu0 - delta
high = mu0 + delta
(diff.p = mean(rand.diff>=high) + mean(rand.diff<=low))

## [1] 0
```

(c) Compute the randomization confidence interval by iterating over values of μ_0.

```
  ##################
# Farther
##################

# Lower val
R = 1000
starting.point = mean(dat1$further)
lower.val = starting.point
samp.mean = mean(dat1$further)
repeat{
  rand = tibble(xbars = rep(NA, R))
  x.shift = dat1$further - lower.val
  #Doing the randomization process
  for(i in 1:R){
    curr.samp = x.shift * sample(x = c(-1,1),
                                 size = length(x.shift),
                                 replace = T)
    rand$xbars[i] = mean(curr.samp)
  }
  #Shifting back
  rand = rand |>
    mutate(xbars = xbars + lower.val)

  #P-value
  delta = abs(samp.mean - lower.val)
  lower = lower.val - delta #Mirror?
  upper = lower.val + delta
  p.val = mean(rand$xbars <= lower) + mean(rand$xbars >= upper)

  if(p.val < .05) {#Should it be .025 or .5?
    break
  }
  else{
    lower.val = lower.val - .0001
  }
}
lower.val

## [1] -0.2534244

#Upper val
upper.val = starting.point
samp.mean = mean(dat1$further)
repeat{
  rand = tibble(xbars = rep(NA, 1000))
  x.shift = dat1$further - upper.val
  #Doing the randomization process
  for(i in 1:R){
    curr.samp = x.shift * sample(x = c(-1,1),
                                   size = length(x.shift),
                                   replace = T)
    rand$xbars[i] = mean(curr.samp)
  }
  #Shifting back
  rand = rand |>
    mutate(xbars = xbars + upper.val)

  #P-value
  delta = abs(samp.mean - upper.val)
  lower = upper.val - delta #Mirror?
  upper = upper.val + delta
  p.val = mean(rand$xbars <= lower) + mean(rand$xbars >= upper)
  if(p.val < .05) {#Should it be .025 or .5?
    break
  }
  else{
    upper.val = upper.val + .0001
  }
}
upper.val

## [1] -0.1525244

c(lower.val,upper.val)

## [1] -0.2534244 -0.1525244
```

```
###############
# Closer
###############

# Lower val
R = 1000
starting.point = mean(dat1$closer)
lower.val = starting.point
samp.mean = mean(dat1$closer)
repeat{
  rand = tibble(xbars = rep(NA, R))
  x.shift = dat1$closer - lower.val
  #Doing the randomization process
  for(i in 1:R){
    curr.samp = x.shift * sample(x = c(-1,1),
                                 size = length(x.shift),
                                 replace = T)
    rand$xbars[i] = mean(curr.samp)
  }
  #Shifting back
  rand = rand |>
    mutate(xbars = xbars + lower.val)

  #P-value
  delta = abs(samp.mean - lower.val)
  lower = lower.val - delta #Mirror?
  upper = lower.val + delta
  p.val = mean(rand$xbars <= lower) + mean(rand$xbars >= upper)

  if(p.val < .05) {#Should it be .025 or .5?
    break
  }
  else{
    lower.val = lower.val - .0001
  }
}
lower.val

## [1] 0.1204231

#Upper val
upper.val = starting.point
samp.mean = mean(dat1$closer)
repeat{
  rand = tibble(xbars = rep(NA, 1000))
  x.shift = dat1$closer - upper.val
  #Doing the randomization process
  for(i in 1:R){
    curr.samp = x.shift * sample(x = c(-1,1),
                                 size = length(x.shift),
                                 replace = T)
    rand$xbars[i] = mean(curr.samp)
  }
  #Shifting back
  rand = rand |>
    mutate(xbars = xbars + upper.val)

  #P-value
  delta = abs(samp.mean - upper.val)
  lower = upper.val - delta #Mirror?
  upper = upper.val + delta
  p.val = mean(rand$xbars <= lower) + mean(rand$xbars >= upper)
  if(p.val < .05) {#Should it be .025 or .5?
    break
  }
  else{
    upper.val = upper.val + .0001
  }
}
upper.val

## [1] 0.1937231

c(lower.val,upper.val)

## [1] 0.1204231 0.1937231

###############
# Difference
###############
```

```r
# Lower val
R = 1000
starting.point = mean(dat1$diff)
lower.val = starting.point
samp.mean = mean(dat1$diff)
repeat{
  rand = tibble(xbars = rep(NA, R))
  x.shift = dat1$diff - lower.val
  #Doing the randomization process
  for(i in 1:R){
    curr.samp = x.shift * sample(x = c(-1,1),
                                 size = length(x.shift),
                                 replace = T)
    rand$xbars[i] = mean(curr.samp)
  }
  #Shifting back
  rand = rand |>
    mutate(xbars = xbars + lower.val)

  #P-value
  delta = abs(samp.mean - lower.val)
  lower = lower.val - delta #Mirror?
  upper = lower.val + delta
  p.val = mean(rand$xbars <= lower) + mean(rand$xbars >= upper)

  if(p.val < .05) {#Should it be .025 or .5?
    break
  }
  else{
    lower.val = lower.val - .0001
  }
}
lower.val

## [1] 0.2745475

#Upper val
upper.val = starting.point
samp.mean = mean(dat1$diff)
repeat{
  rand = tibble(xbars = rep(NA, 1000))
  x.shift = dat1$diff - upper.val
  #Doing the randomization process
  for(i in 1:R){
    curr.samp = x.shift * sample(x = c(-1,1),
                                 size = length(x.shift),
                                 replace = T)
    rand$xbars[i] = mean(curr.samp)
  }
  #Shifting back
  rand = rand |>
    mutate(xbars = xbars + upper.val)

  #P-value
  delta = abs(samp.mean - upper.val)
  lower = upper.val - delta #Mirror?
  upper = upper.val + delta
  p.val = mean(rand$xbars <= lower) + mean(rand$xbars >= upper)
  if(p.val < .05) {#Should it be .025 or .5?
    break
  }
  else{
    upper.val = upper.val + .0001
  }
}
upper.val

## [1] 0.4411475

c(lower.val,upper.val)

## [1] 0.2745475 0.4411475
```

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.

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.