Stat 243 – Test 2 Take home part

Richmond Yevudza

March 15, 2022

1

a.

The effective coverage rate is meant to evaluate the performance of a confidence interval estimator. On a normal circumstance, our Confidence interval should have the highest effective coverage rate value. This isn't easy because large prediction values, for example a prediction value of 95%, might consist of all of the values but these large values are mostly very wide prediction intervals with little practical value. So althogh it is not completely abnormal, an effective coverage rate that is close to 95% could be a surprise.

b.

```
qnorm(1-0.025, 0, 1)
```

```
## [1] 1.959964
```

c.

```
isInside = function(param, ciBounds) {
  ciBounds[1] <= param & param <= ciBounds[2]
}
enn <- 7
zStar <- qnorm(1-0.025, 0, 1)
runResults <- do(10000) * {
  sampledVals <- rnorm(enn, 70, 3);
  xbar <- mean(~ sampledVals);
  ess <- sd(~ sampledVals);
  myCI <- xbar + c(-1,1) * zStar * ess / sqrt(enn);
  isInside(70, myCI)
}
prop(runResults == TRUE)    # the effective coverage rate of the simulation
```

```
## prop_TRUE
##    0.9022
```

d.

```
isInside = function(param, ciBounds) {
  ciBounds[1] <= param & param <= ciBounds[2]
}
enn <- 7
tStar <- qt(0.975, df = enn - 1)
runResults <- do(10000) * {
  sampledVals <- rexp(enn, 0.02);
  xbar <- mean(~ sampledVals);
  ess <- sd(~ sampledVals);
  myCI <- xbar + c(-1,1) * zStar * ess / sqrt(enn);
  isInside(70, myCI)
}
prop(runResults == TRUE)    # the effective coverage rate of the simulation
```

```
## prop_TRUE
##    0.6081
```

e.

```
isInside = function(param, ciBounds) {
  ciBounds[1] <= param & param <= ciBounds[2]
}
enn <- 35
tStar <- qt(0.975, df = enn - 1)
runResults <- do(10000) * {
  sampledVals <- rexp(enn, 0.02);
  xbar <- mean(~ sampledVals);
  ess <- sd(~ sampledVals);
  myCI <- xbar + c(-1,1) * zStar * ess / sqrt(enn);
  isInside(70, myCI)
}
prop(runResults == TRUE)    # the effective coverage rate of the simulation
```

```
## prop_TRUE
##      0.35
```

f.

```
isInside = function(param, ciBounds) {
  ciBounds[1] <= param & param <= ciBounds[2]
}
enn <- 7
tStar <- qt(0.95, df = enn - 1)
runResults <- do(10000) * {
  sampledVals <- rnorm(enn, 70, 3);
  xbar <- mean(~ sampledVals);
  ess <- sd(~ sampledVals);
  myCI <- xbar + c(-1,1) * tStar * ess / sqrt(enn);
  isInside(70, myCI)
}
prop(runResults == TRUE)    # the effective coverage rate of the simulation
```

```
## prop_TRUE
##    0.9005
```

2

a.

```
sleeptrouble0 <- filter(NHANES, SleepTrouble == 'Yes')
sleeptrouble1 <- filter(NHANES, SleepTrouble == 'No')

mean(~Height, data=sleeptrouble0, na.rm = TRUE) -
  mean(~Height, data=sleeptrouble1, na.rm = TRUE)
```

```
## [1] -1.081902
```

```
manyDiffs = do(5000) * (mean(~Height, data=resample(sleeptrouble0), na.rm = TRUE) -
                          mean(~Height, data=resample(sleeptrouble1), na.rm = TRUE))

sd(~result, data=manyDiffs)
```

```
## [1] 0.2592468
```

```
qdata(~result, data=manyDiffs, c(0.025,0.975))
```

```
##       2.5%      97.5%
## -1.5869598 -0.5776994
```

b.

1.Assuming we have a really big bag containing slips with details of each person (i.e age, gender, etc.; just like in NHANES). 2.We then bring 2 other bags let's say called sample1 and sample2. We take time in picking out slips from the really big bag and put them into either sample1 bag or sample2 bag depending on the person's answer to whether or not he/she has trouble sleeping. Let's make sample1 bag the bag with slips of people who have trouble sleeping, and sample2 bag the bag with slips of people who don't have trouble sleeping. 3.Now after we do this, from sample1 bag, we then take outeach slip and record the person's age, as well as counting the number of slips taken out. Now we add all the ages and divide the result by the number of slips to get the mean. This make this mean mean1. 4.We then repeat the previous step for sample2 bag but this time name the mean mean2. 5. Lastly we subtract both means (mean1 - mean2) which gives us a single bootstrap statistic.

c.

```
sleeptrouble0 <- filter(NHANES, SleepTrouble == "Yes")
sleeptrouble1 <- filter(NHANES, SleepTrouble == "No")
diff <- mean(~Height, data = sleeptrouble0, na.rm = TRUE) - mean(~Height, data = sleeptrouble1, na.rm =
std1 <- sd(~Height, data = sleeptrouble0, na.rm = TRUE)
std1
```

```
## [1] 9.775389
```

```
std2 <- sd(~Height, data = sleeptrouble1, na.rm = TRUE)
std2
```

```
## [1] 10.15603
```

```
#Used favstats to find n values for both data sets
favstats(Height ~ SleepTrouble, data=NHANES)
```

```
##   SleepTrouble   min    Q1 median    Q3   max     mean        sd    n missing
## 1           No 137.3 161.7  169.1 176.4 200.4 169.1188 10.156032 5756      43
## 2          Yes 134.5 160.8  167.7 175.1 199.2 168.0369  9.775389 1956      17
```

```
enn1 <- 5756
enn2 <- 1956
stderr <- sqrt((std1*std1/enn1) + (std2*std2/enn2))
dof <- 3269.643845 #Using Welch's formula and manually calculating the dof
tStar <- qt(0.975, df = dof)
lower <- (diff - (tStar*stderr))
upper <- (diff + (tStar*stderr))
c(lower, upper)
```

```
## [1] -1.5981786 -0.5656256
```

The CI of this as compared with that of 2a is substantially the same.

3

a.

$H_0 : \mu_Y - \mu_N = 0$, $H_0 : \mu_Y - \mu_N \neq 0$

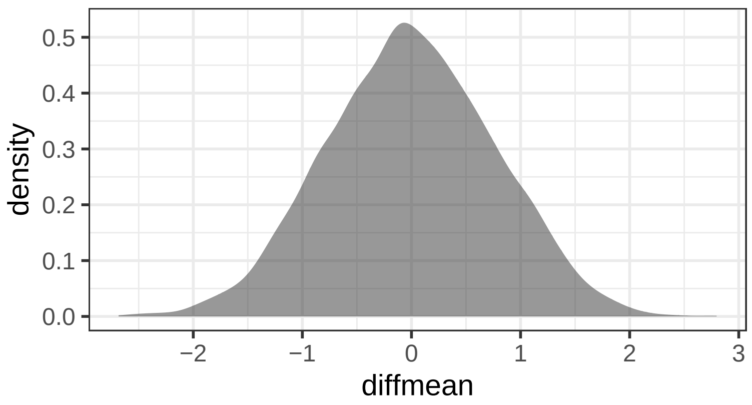
```
favstats(Weight ~ SleepTrouble, data=NHANES)
```

```
##   SleepTrouble  min   Q1 median    Q3   max     mean       sd    n missing
## 1           No 37.0 65.9   78.2 92.70 230.7 81.07340 20.97112 5751      48
## 2          Yes 38.7 67.1   80.4 95.35 188.9 83.05847 22.13269 1955      18
```

```
testStat <- diffmean(Weight ~ SleepTrouble, data = NHANES, na.rm = TRUE)
testStat
```

```
## diffmean
##  1.98507
```

```
manyDiffs <- do(5000) * diffmean(Weight ~ shuffle(SleepTrouble), data = NHANES, na.rm = TRUE)
gf_density(~diffmean, data = manyDiffs)
```



```
p_value <- nrow(filter(manyDiffs, abs(diffmean) >= abs(testStat)))/5000
p_value
```

```
## [1] 0.008
```

The p value is the sum of the relative frequencies. This tells us how likely it is that our data would have occurred by random chance. Since it is low in this case, there is a low chance.

b.

We take a big bag of all the data we need being in slips. This time around we shuffle everything together and pick slips. If that slip has sleeptrouble to be no, we put it in one bag, if yes we put it in the other bag. We do this then after we take two vectors in a given dataset (let's call them OurData and YourData), take the mean of the data and subtract it from a resampled data which gives us a randomization distribution.

c.

```
sleeptrouble0 <- filter(NHANES, SleepTrouble == "Yes")
sleeptrouble1 <- filter(NHANES, SleepTrouble == "No")
diff <- mean(~Height, data = sleeptrouble0, na.rm = TRUE) - mean(~Height, data = sleeptrouble1, na.rm =
std1 <- sd(~Weight, data = sleeptrouble0, na.rm = TRUE)
std1
```

```
## [1] 22.13269
```

```
std2 <- sd(~Weight, data = sleeptrouble1, na.rm = TRUE)
std2
```

```
## [1] 20.97112
```

```
#Used favstats to find n values for both data sets
favstats(Weight ~ SleepTrouble, data=NHANES)
```

```
##   SleepTrouble  min   Q1 median    Q3   max     mean       sd    n missing
## 1           No 37.0 65.9   78.2 92.70 230.7 81.07340 20.97112 5751      48
## 2          Yes 38.7 67.1   80.4 95.35 188.9 83.05847 22.13269 1955      18
```

```
enn1 <- 5751
enn2 <- 1955
stderr <- sqrt((std1*std1/enn1) + (std2*std2/enn2))
dof <- 3541.339668 #Using Welch's formula and manually calculating the dof
tStar <- qt(0.95, df = dof)
lower <- (diff - (tStar*stderr))
upper <- (diff + (tStar*stderr))
c(lower, upper)
```

```
## [1] -1.9981538 -0.1656504
```

```
t <- (diff - 0)/ stderr
t
```

```
## [1] -1.942737
```

```
p_value <- 2*(pt(t, df = dof))
p_value
```

```
## [1] 0.05212708
```

The p values are different when compared.