Lab #5

Confidence Interval

Donovan Lee - 016741645

Jimmy Tran - 016878652

12/19/2020

Introduction:

The purpose of the lab is to measure a large population of a size N and explore the relation of X bar to the population mean. We would use confidence intervals, 95%, and 99%, to find a population curve and run it 1,000,000 times to see the percentage of the population in the interval. In parts 1 and 2 of the lab, we will first create a population and use confidence intervals to graph out our population. Then the second part of the lab will be to calculate the success of having 3 different n sizes 5, 40, and 120 is compared with the confidence intervals.

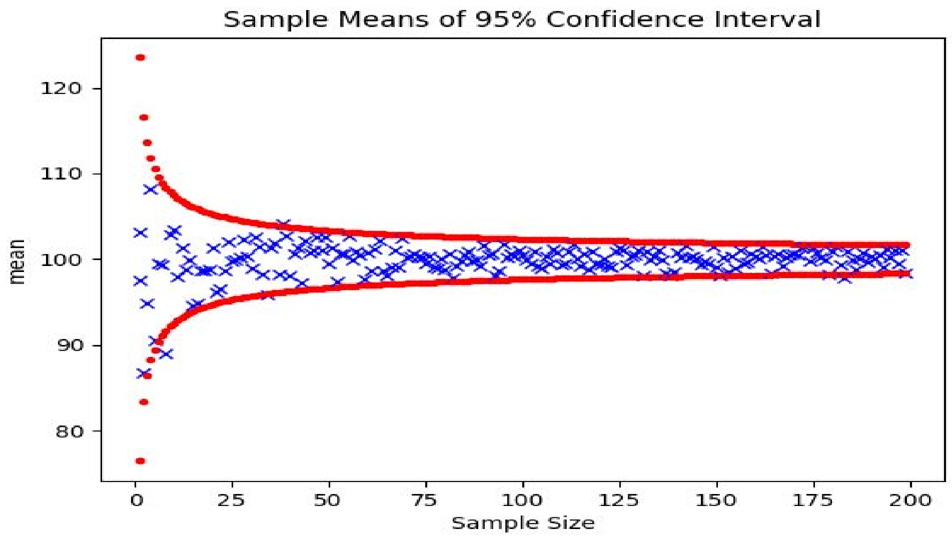
Procedure:

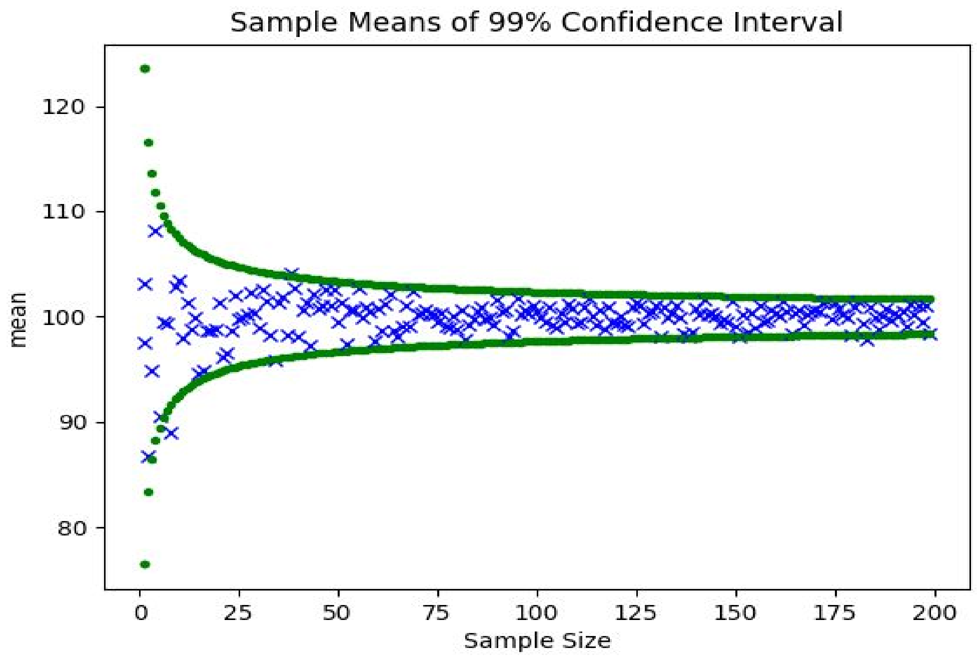
For the first part of the lab, we set our experiment runs to 1,000,000, our mean to 100, our standard deviation to 12, and our max sample size to 200. We then ran a loop that went the range of our sample size. Then we created our population and looped through that too. Inside that loop, we added all the population numbers together to get the max sum for x. Then we get our sample standard deviation. After that, we start calculating our lower and upper bound for our confidence intervals for 95% and 99%. Then we append all the calculations into a list that holds all the information. Then the loop reruns and does the same thing for the next population that would be created.

For the second part of the lab, we first calculated the sample mean and sample standard deviation. Once we calculated those, we used those values and corresponding zscores and tscores for 95% and 99% confidence levels to find the normal distribution and Student's t distribution. For each value of n=5, n=40, and n=120, we calculated the normal distribution and Student's t distribution mean bounds and if the actual mean was between the mean bounds, then the distribution was a success. After we calculated the percentage of success of the trials for each n value.

Results and Discussion:

Part 1:

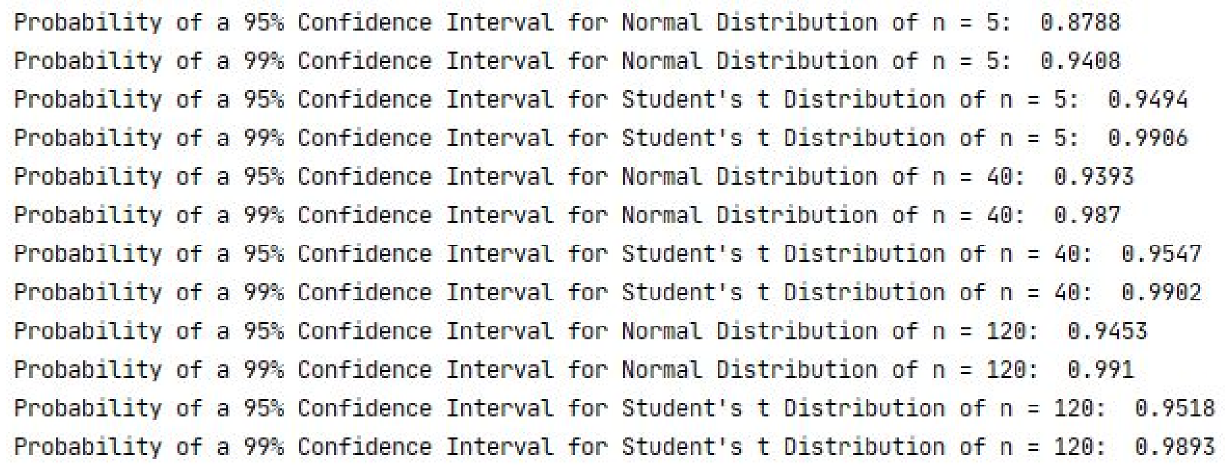




Our results look correct when compared to the lab handout. So our results were what we were expecting. With our 95% interval, you could see that some of the x's are on the line and more spread out than the 99% interval that has the x's really close to each other and in a tight position for the graph.

Part 2:

n	95%(normal)	99%(normal)	95%(Student's t)	99%(Student's t)
5	0.8788	0.9408	0.9494	0.9906
40	0.9393	0.987	0.9547	0.9902
120	0.9453	0.991	0.9518	0.9893



```
Probability of a 95% Confidence Interval for Normal Distribution of n = 5:  0.8788
Probability of a 99% Confidence Interval for Normal Distribution of n = 5:  0.9408
Probability of a 95% Confidence Interval for Student's t Distribution of n = 5:  0.9494
Probability of a 99% Confidence Interval for Student's t Distribution of n = 5:  0.9906
Probability of a 95% Confidence Interval for Normal Distribution of n = 40:  0.9393
Probability of a 99% Confidence Interval for Normal Distribution of n = 40:  0.987
Probability of a 95% Confidence Interval for Student's t Distribution of n = 40:  0.9547
Probability of a 99% Confidence Interval for Student's t Distribution of n = 40:  0.9902
Probability of a 95% Confidence Interval for Normal Distribution of n = 120:  0.9453
Probability of a 99% Confidence Interval for Normal Distribution of n = 120:  0.991
Probability of a 95% Confidence Interval for Student's t Distribution of n = 120:  0.9518
Probability of a 99% Confidence Interval for Student's t Distribution of n = 120:  0.9893
```

Our results correlate with how normal distribution and Student's t distribution should be for each confidence interval. The normal distribution is more accurate for n = 5 than the Student's t, whereas the Student's t distribution is more accurate for n = 40 and n = 120 than the normal distribution.

Conclusion:

What we learned from the lab was creating confidence intervals that had values between them. Aspects from the lecture that we were able to apply to the code were the formulas for the confidence interval, standard deviation., and creating sample means. Challenges that we were able to overcome were finding out how to create the outside lines to show the area in which the population would be held within. The part that we had trouble was making the lines containing the population, but we were able to figure it out.

Appendix:

```
import numpy as np
import random
from random import randint
import math
import matplotlib
import matplotlib.pyplot as plt
from numpy import histogram


def partOne():
    N = 1000000
    mean = 100
    std = 12
    sample_size = 200
    results = []
    for i in range(sample_size + 1):
        if (i == 0):#start at 1 so the incrementing makes sense from 1- 200
instead of 0-199
            i += 1
        sample_Data = []
        sample_Data.append(i)
        sum_x = 0
        N_x = np.random.normal(mean, std, i)
        for sample in range(len(N_x)):
            sum_x += N_x[sample]#adds all the pop together

        sample_std = std / math.sqrt(i)#get sample std

        # 95% confidence intervals
        lower_95 = mean - (1.96 * sample_std)
        upper_95 = mean + (1.96 * sample_std)
        # 99% confidence interval
        lower_99 = mean - (2.58 * sample_std)
        upper_99 = mean + (2.58 * sample_std)

        sample_Data.append(sum_x / i)#get the mean
        sample_Data.append(lower_95)
        sample_Data.append(upper_95)
        sample_Data.append(lower_99)
        sample_Data.append(upper_99)
        results.append(sample_Data)# add sample data into list to so it would be
a tuple

    # Plot for 95%
    fig1 = plt.figure(1)
    plt.title("Sample Means of 95% Confidence Interval")
    plt.xlabel("Sample Size")
    plt.ylabel("mean")

    for i in range(sample_size):
        nx_sample = results[i][0]#grabs the sample
        nx_bar = results[i][1]#grabs the mean
        low_bound = results[i][2]#grabs the lower bound
        up_bound = results[i][3]#grabs upper bound

        plt.plot(nx_sample, nx_bar, 'xb')#plot the population
        plt.plot(nx_sample, up_bound, '.r')#creates the upper curve graph as
color red
        plt.plot(nx_sample, low_bound, '.r')#creates the lower curve
    plt.show()
    # Plot for 99%
    fig2 = plt.figure(2)
    plt.title("Sample Means of 99% Confidence Interval")
    plt.xlabel("Sample Size")
    plt.ylabel("mean")

    for i in range(sample_size):
        nx_sample = results[i][0]#same as above but for the 99% confidence
interval
```

```
        nx_bar = results[i][1]
        low_bound = results[i][2]
        up_bound = results[i][3]

        plt.plot(nx_sample, nx_bar, 'xb')
        plt.plot(nx_sample, up_bound, '.g')#color green
        plt.plot(nx_sample, low_bound, '.g')
    plt.show()
```

```
def partTwoCalc(n, d, confidence, mean, std):
    s = 0
    #zscore for each confidence level
    zscore = [1.96, 2.58]
    #corresponding tscore for each confidence level
    fivevt = [2.776, 4.604]
    fourtyvt = [2.0227, 2.7079]
    onetwentyvt = [1.9801, 2.6178]
    #distibution
    val = np.random.normal(mean, std, n)
    sum_x = np.sum(val)
    #sample mean
    bar_x = sum_x / n
    #sample standard deviation
    for i in range(len(val)):
        s += (val[i] - bar_x) ** 2
    s_hat = math.sqrt(s/(n-1))
    #using the normal distribution
    if (d == 0):
        if (confidence == 95):
            mean_upper = bar_x + zscore[0] * (s_hat/math.sqrt(n))
            mean_lower = bar_x - zscore[0] * (s_hat/math.sqrt(n))
        if (confidence == 99):
            mean_upper = bar_x + zscore[1] * (s_hat / math.sqrt(n))
            mean_lower = bar_x - zscore[1] * (s_hat / math.sqrt(n))
    #using the Student's t distrubtion
    if (d == 1):
        if(n == 5):
            if (confidence == 95):
                mean_upper = bar_x + fivevt[0] * (s_hat/math.sqrt(n))
                mean_lower = bar_x - fivevt[0] * (s_hat / math.sqrt(n))
            if (confidence == 99):
                mean_upper = bar_x + fivevt[1] * (s_hat/math.sqrt(n))
                mean_lower = bar_x - fivevt[1] * (s_hat / math.sqrt(n))
        if (n == 40):
            if (confidence == 95):
                mean_upper = bar_x + fourtyvt[0] * (s_hat / math.sqrt(n))
                mean_lower = bar_x - fourtyvt[0] * (s_hat / math.sqrt(n))
            if (confidence == 99):
                mean_upper = bar_x + fourtyvt[1] * (s_hat / math.sqrt(n))
                mean_lower = bar_x - fourtyvt[1] * (s_hat / math.sqrt(n))
        if (n == 120):
            if (confidence == 95):
                mean_upper = bar_x + onetwentyvt[0] * (s_hat / math.sqrt(n))
                mean_lower = bar_x - onetwentyvt[0] * (s_hat / math.sqrt(n))
            if (confidence == 99):
                mean_upper = bar_x + onetwentyvt[1] * (s_hat / math.sqrt(n))
                mean_lower = bar_x - onetwentyvt[1] * (s_hat / math.sqrt(n))
    #seeing if the mean is between the bounds
    if(mean_lower<mean<mean_upper):
        #success
        return 1
    else:
        #failure
        return 0
```

```
def partTwo():
   #number of trials
   N = 10000
   #sample size values
   n = [5,40,120]
   #normal distribution and student's t distribution token
   nd, td = 0, 1
   mean = 100
   std = 12

   #n=5
   counter = 0
   for i in range(N):
       counter += partTwoCalc(n[0], nd, 95, mean, std)
   print("Probability of a 95% Confidence Interval for Normal Distribution of n = 5:
", counter / N)
   counter = 0
   for i in range(N):
       counter += partTwoCalc(n[0], nd, 99, mean, std)
   print("Probability of a 99% Confidence Interval for Normal Distribution of n = 5:
", counter / N)
   counter = 0
   for i in range(N):
       counter += partTwoCalc(n[0], td, 95, mean, std)
   print("Probability of a 95% Confidence Interval for Student's t Distribution of n =
5: ", counter / N)
   counter = 0
   for i in range(N):
       counter += partTwoCalc(n[0], td, 99, mean, std)
   print("Probability of a 99% Confidence Interval for Student's t Distribution of n =
5: ", counter / N)
```

```
   #n=40
   counter = 0
   for i in range(N):
       counter += partTwoCalc(n[1], nd, 95, mean, std)
   print("Probability of a 95% Confidence Interval for Normal Distribution of n = 40:
", counter / N)
   counter = 0
   for i in range(N):
       counter += partTwoCalc(n[1], nd, 99, mean, std)
   print("Probability of a 99% Confidence Interval for Normal Distribution of n = 40:
", counter / N)
   counter = 0
   for i in range(N):
       counter += partTwoCalc(n[1], td, 95, mean, std)
   print("Probability of a 95% Confidence Interval for Student's t Distribution of n =
40: ", counter / N)
   counter = 0
   for i in range(N):
       counter += partTwoCalc(n[1], td, 99, mean, std)
   print("Probability of a 99% Confidence Interval for Student's t Distribution of n =
40: ", counter / N)

   #n=120
   counter = 0
   for i in range(N):
       counter += partTwoCalc(n[2], nd, 95, mean, std)
   print("Probability of a 95% Confidence Interval for Normal Distribution of n = 120:
", counter / N)
   counter = 0
   for i in range(N):
       counter += partTwoCalc(n[2], nd, 99, mean, std)
```

```
    print("Probability of a 99% Confidence Interval for Normal Distribution of n = 120: 
", counter / N)
    counter = 0
    for i in range(N):
        counter += partTwoCalc(n[2], td, 95, mean, std)
    print("Probability of a 95% Confidence Interval for Student's t Distribution of n = 
120: ", counter / N)
    counter = 0
    for i in range(N):
        counter += partTwoCalc(n[2], td, 99, mean, std)
    print("Probability of a 99% Confidence Interval for Student's t Distribution of n = 
120: ", counter / N)




def main():
    partOne()
    partTwo()

main()
```