EE 381 Lab 5 Report

Confidence Intervals

Daniel Duong and Long Nguyen

Instructor: Dr. Duc Tran

Due 12/18/20

Table of Contents

Introduction:

The lab helps us simulate the process of sampling with different sample sizes and means. It also helps us analyze the difference between 95% confidence interval and 99% interval. For the second question we compare how the size of the samples affect the accuracy when using normal distribution and student T distribution.

Procedure:

Task 1:

1. Generate a population size of 1 million and save them into a list.
2. Pick random sample of size n from 1 → 200 using random.sample() taken from the population generated in previous step
3. Calculate the mean of the sample using statistic.mean() and save it to a list
4. Calculate the interval using a function that we made that take in the mean, confidence level and the function return a list of lower bound, and upper bound.
5. For each sample size, then we save the upper bound and lower bound to 2 different lists.
6. We repeat the step 3 to step 5 for all different sample sizes from 1-->200
7. We plot the graph using the sample mean calculated above and the confidence interval value that are saved in 2 different lists.

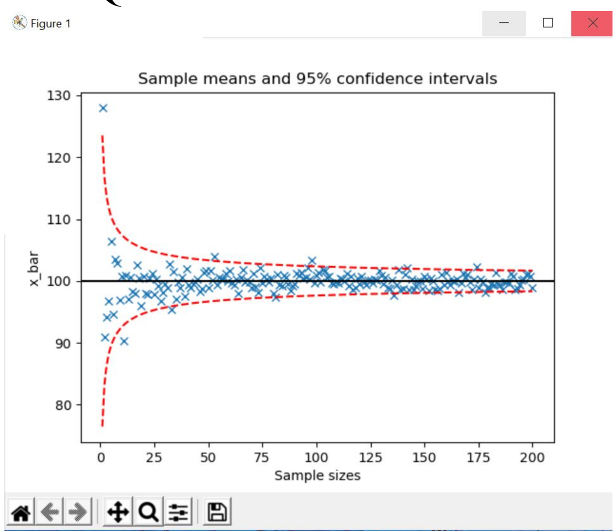
Task 2:

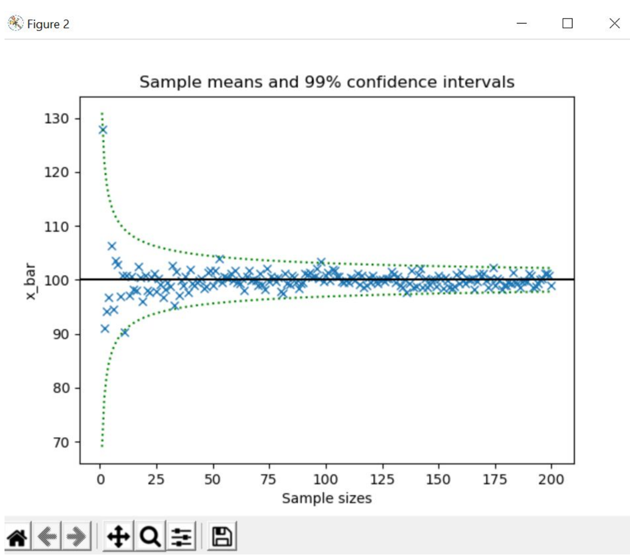
1. Generate a population of size 1 million and save it into a list
2. Calculate the population mean generated above using statistic.mean()
3. We create a sample size of 5, 40, 120
4. We calculate the mean and standard deviation of each of these sample size

5. We calculate the lower and upper confidence interval using a function for both student T and normal distribution.
6. Check to see if population mean calculated in step 2 to see it is in between the upper and lower confidence interval. Increase the counter if it is in the interval.
7. We repeat step 3 to step 6 10000 times

Results and Discussions:

Lab 5 Q1:





Lab 5 Q2:

n	95% normal	99% normal	95% student ‘t	99% student ‘t
5	0.88	0.936	0.9476	0.9902
40	0.9457	0.9865	0.9922	1.0
120	0.9451	0.9891	0.9937	1.0

Conclusion:

We have applied what we have learned in lecture 10 and 11. The most important topics regarding this lab was to compare the accuracy of student ‘t and the gaussian distribution relative to the sample size. We were not sure which mean to use for the confidence interval for task 1 since there are population mean and the samples mean. But then we tried both values on the graph and compared with the example provided in the lab prompt and figured that the correct one was the population mean. We also learnt how to graph individual points and confidence levels on the graph and represent them using symbol “x”. In order to graph individual data points, we had to set (linestyle = “ ”) and we learnt how to do that by researching. We also learned that “linestyle = “--” “ will make dashed line graph while “linestyle= “:” will graph a line with dots. We learnt as the sample size gets larger the result returned by student T and normal distribution are very close to each other. For task 2, since the degree of freedom v = n -1, but since there is no value on the student T table provided by the class textbook for a degree of freedom of 39 and 119 so therefore we just used the closest value to that which is 40 and 120 in the student T table.

Appendix:

Lab5Q1.py

```
import math
import random
import statistics
import numpy as np
import scipy.stats
import matplotlib.pyplot as plt

def main():
    N = 1_000_000
    MEAN = 100
    sigma = 12
    Z1 = 1.96
    Z2 = 2.58

    population = np.random.normal(MEAN, sigma, N)
    # print(type(population))

    populationList = population.tolist()

    meanList = []
    posInList = []
    negaInList = []
    posInList99 = []
    negaInList99 = []
    n_value = []

    #n = random.randint(1, 100)
    for n in range(1, 201):
        n_value.append(n)
        sample = random.sample(populationList, n)
        sample_mean = statistics.mean(sample)
        meanList.append(sample_mean)

        #Calculate the confidence interval of 95
        interList = calInterval(MEAN, sigma, n, 95)
        #Add the positive and negative interval to approriate list
        posInList.append(interList[0])
        negaInList.append(interList[1])

        #Calculate the confidence interval of 99
        interList99 = calInterval(MEAN, sigma, n, 99)
        #Add the positive and negative interval to approriate list
```

```
        posInList99.append(interList99[0])
        negaInList99.append(interList99[1])

        #print("Mean: ", mean)
    print("Meanlist: ", meanList)
    print("MeanList size: ", len(meanList))
    print("PosList: ", posInList)
    print("PosList len: ", len(posInList))
    print("NegaList: ", negaInList)
    print("NegaList len: ", len(negaInList))
    print("n value: ", n_value)

    #95 confidence graph
    figure1 = plt.figure(1)
    plt.plot(n_value, meanList, linestyle=' ', marker="x")
    plt.plot(n_value, posInList, 'r', linestyle='--')
    plt.plot(n_value, negaInList, 'r', linestyle='--')
    plt.axhline(y=100, color="black")
    plt.xlabel("Sample sizes")
    plt.ylabel("x_bar")
    plt.title("Sample means and 95% confidence intervals")

    #99 confidence graph
    figure2 = plt.figure(2)
    plt.plot(n_value, meanList, linestyle=' ', marker="x")
    plt.plot(n_value, posInList99, 'g', linestyle=':')
    plt.plot(n_value, negaInList99, 'g', linestyle=':')
    plt.axhline(y=100, color="black")
    plt.xlabel("Sample sizes")
    plt.ylabel("x_bar")
    plt.title("Sample means and 99% confidence intervals")

    plt.show()

    print("n: ", n)
    print("sample: ", sample)
    #
    # print(population)
    # print("Size: ", population.size)
    # print("list: ", populationList)
    # print("Size: ", len(populationList))

def calInterval(mean, sigma, sample_size, con_level):
    interval = []
    if (con_level == 95):
        result = mean + 1.96 * (sigma / math.sqrt(sample_size))
```

```
        resultNegative = mean - 1.96 * (sigma / math.sqrt(sample_size))
        interval.append(result)
        interval.append(resultNegative)
    elif (con_level == 99):
        result = mean + 2.58 * (sigma / math.sqrt(sample_size))
        resultNegative = mean - 2.58 * (sigma / math.sqrt(sample_size))
        interval.append(result)
        interval.append(resultNegative)
    return interval
main()
```

Lab5Q2V2.py

```
import math
import random
import statistics
import numpy as np
import scipy.stats
import matplotlib.pyplot as plt


def calInterval(mean, sigma, sample_size, zValue):
    interval = []
    upperLim = mean + zValue * (sigma / math.sqrt(sample_size))
    lowerLim = mean - zValue * (sigma / math.sqrt(sample_size))
    interval.append(lowerLim)
    interval.append(upperLim)
    return interval

def calIntervalNormal(mean, sigma, sample_size, con_level):
    interval = []
    if (con_level == 95):
        result = mean + 1.96 * (sigma / math.sqrt(sample_size))
        resultNegative = mean - 1.96 * (sigma / math.sqrt(sample_size))
        interval.append(resultNegative)
        interval.append(result)

    elif (con_level == 99):
        result = mean + 2.58 * (sigma / math.sqrt(sample_size))
        resultNegative = mean - 2.58 * (sigma / math.sqrt(sample_size))
        interval.append(resultNegative)
        interval.append(result)

    return interval

def main():
```

```
N = 1_000_000
MEAN = 100
sigma = 12
Z1 = 1.96
Z2 = 2.58

student5_95 = 2.78
student5_99 = 4.6
student40_95 = 2.02        #look at 97.5 percent tile for 40 degree of freedom
student40_99 = 2.70
student120_95 = 1.98
student120_99 = 2.62

population = np.random.normal(MEAN, sigma, N)
# print(type(population))

populationList = population.tolist()
#print(type(populationList))
pop_mean = statistics.mean(populationList)
print("Pop mean: ", pop_mean)

number_run = 10000
sam_size = 120
counterT = 0
counterT_99 = 0
counterN = 0
counterN_99 = 0

for i in range(number_run):
    sample = random.sample(populationList, sam_size)
    sam_mean = statistics.mean(sample)
    sam_dev = statistics.stdev(sample)

    interval_T95 = calInterval(sam_mean, sam_dev, sam_size, student5_95)
    interval_T99 = calInterval(sam_mean, sam_dev, sam_size, student5_99)
    interval_N95 = calIntervalNormal(sam_mean, sam_dev, sam_size, 95)
    interval_N99 = calIntervalNormal(sam_mean, sam_dev, sam_size, 99)

    if (pop_mean >= interval_T95[0] and pop_mean < interval_T95[1]):
        counterT += 1
    if (pop_mean >= interval_T99[0] and pop_mean < interval_T99[1]):
        counterT_99 += 1

    #Normal
    if (pop_mean >= interval_N95[0] and pop_mean < interval_N95[1]):
        counterN += 1
```

```
        if (pop_mean >= interval_N99[0] and pop_mean < interval_N99[1]):
            counterN_99 += 1


    successT = counterT / number_run
    successT_99 = counterT_99 / number_run
    successN = counterN / number_run
    successN_99 = counterN_99 / number_run
    print("Sample size: ", sam_size)
    print("Success T: ", successT)
    print("Success T 99% conf: ", successT_99)
    print("Success N: ", successN)
    print("Success N 99% conf: ", successN_99)

main()
```