Project 5 - Confidence Intervals

CECS 381 - Sec 06

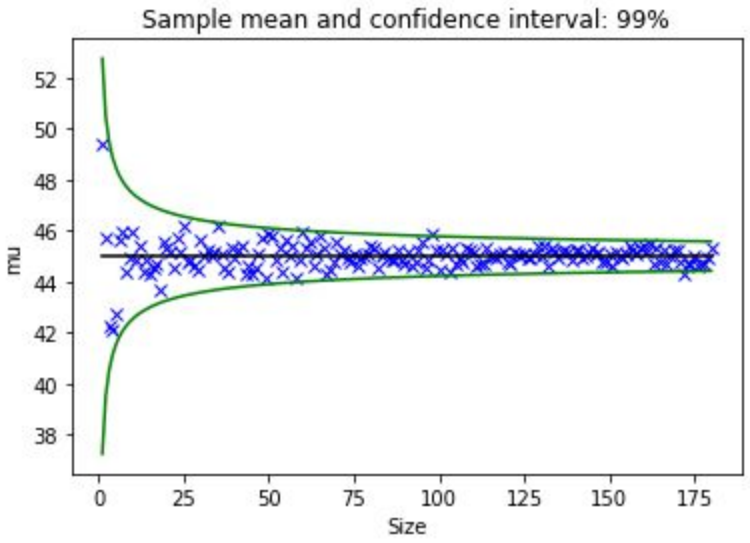
Dustin Martin - 015180085

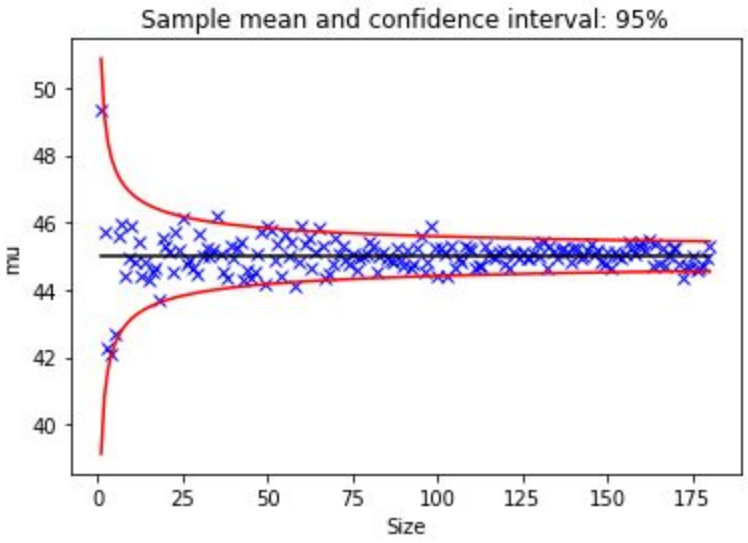
Problem 1:

- **Introduction**
  Problem 1 demonstrates the effect of sample size on the confidence intervals. Using a mean of 45 and std of 3, we graph the 95% and 99% confidence interval using sample sizes from 1 to 180.

- **Methodology**
  The code is fairly simple. Using the given values, we generate random samples from sizes 1-180 and plot them (blue cross marks) on the graph.
  To compare them with the theoretical values, we also graph the upper and lower values which the estimated mean should fall between with 95% and 99% confidence.

- **Results**





- **Appendix/Source Code**

```
import numpy as np
import matplotlib.pyplot as plt
import math

################ Problem 1

N = 1200000 #total number of bearings

mean = 45 #grams

std = 3 #grams

n = 180
#sample sizes = 1 to 180

small_x = np.zeros((n,1))

sigma_small_x = np.zeros((n,1))

sample = np.zeros((n,1))

mu_small_x = np.full((n,1), mu)

for i in range(1, n+1):
    X = np.random.normal(mu, std, i)
    small_x[i-1] = np.sum(X) / i
    sigma_small_x[i-1] = std/math.sqrt(i)
    sample[i-1] = i

#95
conf_95_pos = np.zeros((n,1))
conf_95_neg = np.zeros((n,1))

for i in range(0,n):
    conf_95_pos[i] = mean + (sigma_small_x[i]*1.96)
    conf_95_neg[i] = mean - (sigma_small_x[i]*1.96)
    plt.plot(sample[i], small_x[i], 'b', marker = 'x')

plt.title("Sample mean and confidence interval: 95%")
plt.xlabel("Size")
plt.ylabel("mu")
plt.plot(sample, conf_95_pos, 'r')
plt.plot(sample, mu_small_x, 'k')
plt.plot(sample, conf_95_neg, 'r')
```

```
plt.show()

#95
conf_99_pos = np.zeros((n,1))
conf_99_neg = np.zeros((n,1))

for i in range(0,n):
    conf_99_pos[i] = mean + (sigma_small_x[i]*2.58)
    conf_99_neg[i] = mean - (sigma_small_x[i]*2.58)
    plt.plot(sample[i], small_x[i], 'b', marker = 'x')

plt.title("Sample mean and confidence interval: 99%")
plt.xlabel("Size")
plt.ylabel("mu")
plt.plot(sample, conf_99_pos, 'g')
plt.plot(sample, mu_small_x, 'k')
plt.plot(sample, conf_99_neg, 'g')
plt.show()
```

Problem 2:

- **Introduction**
  Problem 2 demonstrates the accuracy of using confidence intervals on varying sample sizes, calculating the percentage of samples which accurately estimate the mean with 95% and 99% confidence for sample sizes of 5, 40, and 120.
  Problem 2 also uses the t-distribution table to find accurate critical values to generate upper and lower values the mean might fall within. The results of these are then compared with the z-distribution (normal) table.

- **Methodology**
  To calculate successes, a random sample of 5 is chosen. Using the below equations, the mean and STD are calculated.

$$\bar{X} = \frac{1}{n}\sum_{j=1}^{n} X_j \quad \text{and} \quad \hat{S} = \sqrt{\frac{1}{n-1}\sum_{j=1}^{n}(X_j - \bar{X})^2}$$

  To create the 95% confidence interval, we use:

$$[\mu_{\text{Lower}}, \mu_{\text{Upper}}] = [\bar{X} - 1.96\frac{\hat{S}}{\sqrt{n}}, \bar{X} + 1.96\frac{\hat{S}}{\sqrt{n}}]$$

  We then check if the mean for the whole population fits within these bounds. If it does, it is considered a success. If not, then it is a failure. This is repeated 10,000 times to get an accurate percentage of success.
  The same experiment is also ran for the 99% interval as well, swapping 1.96 with 2.58.
  Because the interval equation is not fit for samples less than thirty, we also run it using crit values from the t-table. The Z-value (normal) crits remain the same as we re-run the experiment for sample sizes 40 and 120, but the t-values shift, requiring the use of a t-distribution table to confer with.

- **Results**

Sample Size (n)	**95% Confidence** (Using Normal Distribution)	**99% Confidence** (Using Normal Distribution)	**95% Confidence** (Using Student's t Distribution)	**99% Confidence** (Using Student's t Distribution)
5	87.52%	93.72%	95.08%	99.04%
40	87.67%	93.81%	88.9%	94.27%
120	87.96%	94.3%	88.17%	93.52%

Appendix/Source Code:

```
import numpy as np
import matplotlib.pyplot as plt
import math


mean = 45 #grams

std = 3 #grams


############## Problem 2
M = 10000
def confidence_success(n, M, mean, std, crit):
    succ = 0
    for i in range(0, M):
        X = np.random.normal(mean, std, n)
        small_x = np.sum(X) / n

        temp = np.zeros((n,1))
        for j in range(0,n):
            temp[j] = (X[j]-small_x)**2
        S = math.sqrt(np.sum(temp)/(n-1))

        high = small_x + (crit * (S/math.sqrt(n)))
        low = small_x - (crit * (S/math.sqrt(n)))

        if (mean<high and mean>low):
            succ+=1
    return succ/M

n1 = 5

n2 = 40

n3 = 120

print("n= ", n1)
print("conf_95_perc: ",confidence_success(n1, M, mean, std, 1.96))
print("conf_99_perc: ",confidence_success(n1, M, mean, std, 2.58))

print("t_conf_95_perc: ",confidence_success(n1, M, mean, std, 2.7763))
print("t_conf_99_perc: ",confidence_success(n1, M, mean, std, 4.6022))
```

```
print("\nn= ", n2)
print("conf_95_perc: ",confidence_success(n1, M, mean, std, 1.96))
print("conf_99_perc: ",confidence_success(n1, M, mean, std, 2.58))

print("t_conf_95_perc: ",confidence_success(n1, M, mean, std, 2.0227))
print("t_conf_99_perc: ",confidence_success(n1, M, mean, std, 2.7079))


print("\nn= ", n3)
print("conf_95_perc: ",confidence_success(n1, M, mean, std, 1.96))
print("conf_99_perc: ",confidence_success(n1, M, mean, std, 2.58))

print("t_conf_95_perc: ",confidence_success(n1, M, mean, std, 1.9801))
print("t_conf_99_perc: ",confidence_success(n1, M, mean, std, 2.6178))
```