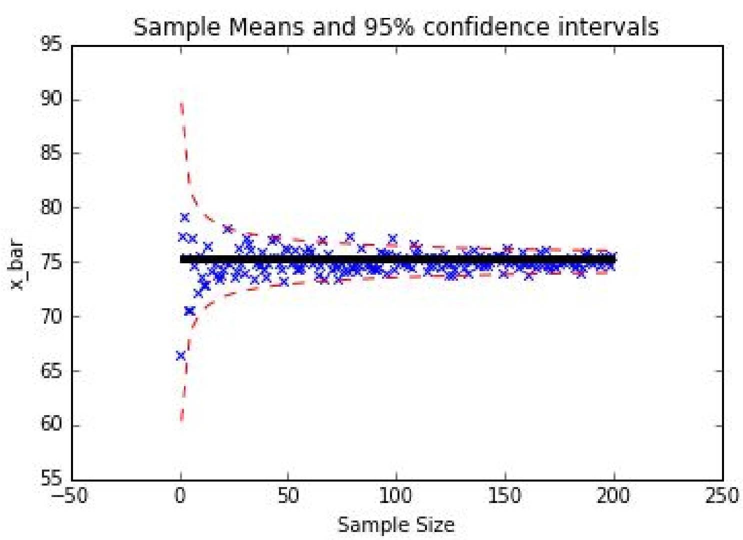
Roberto Sanchez (014587792)
November 13, 2017
EE 381 – Probability and Statistics with Applications to Computing
Lab 05 - Confidence Intervals

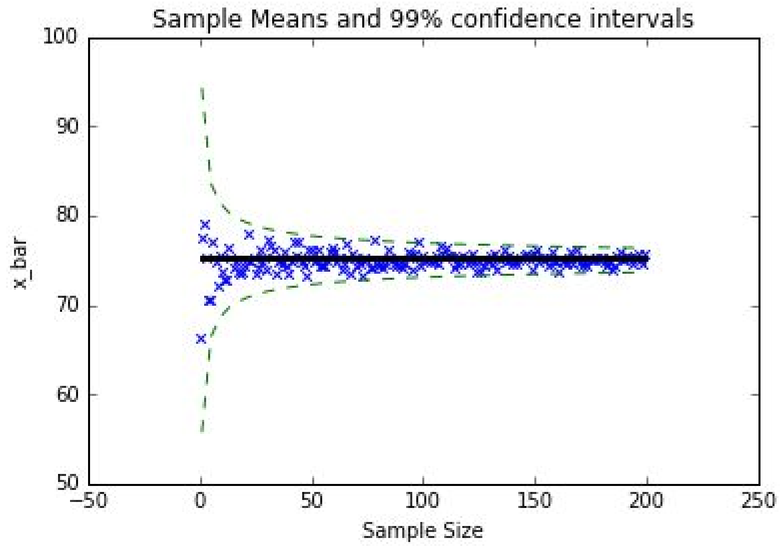
1) Effect of a sample size on confidence intervals
    a) **INTRODUCTION:**
        i) We are creating the effect on a sample size on the confidence interval on a plot. Our inputs are N=1,000,000 for number of bearings, sample size is n = 200, with mean = 75 grams, and population standard deviation of 7.5 grams.
    b) **METHODOLOGY:**
        i) We start by generating values for the given… Then we plot this data by using the z values for both 99% and 95% confidence interval we can plot this data and generate a graph.
    c) **RESULTS AND CONCLUSIONS:**

For 95% confidence interval:

```
In [1]: runfile('/home/beryl/.spyder2-py3/temp.py', wdir='/home/beryl/.spyder2-py3')
```



For 99% confidence interval:



2) Exponentially Distributed Random Variables
   a) **INTRODUCTION:** We are going to create a simulation experiment by creating 1,000,000 bearings and a random sample of 5 and calculate using the formulas provided in the handout. We are going to create two graphs that shows the confidence
   b) **METHODOLOGY:**

First generate 1,000,000 ball bearings with a normal distribution. The sample size will determine the confidence interval values. For the 95% confidence interval use the normal distribution to fill the entries. Then check if the confidence includes the actual mean. If the mean the actual mean is included then it will count as a success. Then calculate the number success out of 10,000 runs. Repeat for 99% confidence level.

   c) **RESULTS AND CONCLUSIONS:**

x	**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	87.78	93.85	94.94	98.91
40	93.78	98.61	94.47	98.97
120	95.05	99.03	95.27	99.20

3) Appendix
   a) Code for #1

```python
# Name: Roberto Sanchez
# Data: November 12, 2017
# Assignment 05 - Confidence Intervals
# Part 1
#----------------------------------------
import numpy as np
import matplotlib.pyplot as plt

def SampleConfInterval(sampleSize):
    # Declaring given data
    totalBearings = 1000000
    popMean = 75
    popStdDev = 7.5
    # calculate
    bearings = np.random.normal(popMean, popStdDev, totalBearings)
    #
    result = []
    #
    for i in range(sampleSize):
        sample = np.random.choice(bearings, (i + 1))
        result.append(sum(sample) / (i + 1))
    # Z-Values for Conf. Intervals
    z_95 = 1.95
    z_99 = 2.57
    # Plotting Figure 1
    plt.figure(1)
    plt.title("Sample Means and 95% confidence intervals")
    plt.xlabel("Sample Size")
    plt.ylabel("x_bar")
    plt.barh(popMean, sampleSize, height=0.5, color='black')
    # Creating evenly spaced numbers over a specific interval
    x = np.linspace(1, sampleSize)
    plt.plot(x, popMean + z_95 * popStdDev / (x ** (1 / 2)), color='red', linestyle='--')
    plt.plot(x, popMean - z_95 * popStdDev / (x ** (1 / 2)), color='red', linestyle='--')
    plt.scatter(range(sampleSize), result, marker='x')
    plt.show()
    # Plotting Figure 2
    plt.figure(2)
    plt.title("Sample Means and 99% confidence intervals")
    plt.xlabel("Sample Size")
    plt.ylabel("x_bar")
    plt.barh(popMean, sampleSize, height=0.5, color='black')
    plt.plot(x, popMean + z_99 * popStdDev / (x ** (1 / 2)), color='green', linestyle='--')
    plt.plot(x, popMean - z_99 * popStdDev / (x ** (1 / 2)), color='green', linestyle='--')

    plt.scatter(range(sampleSize), result, marker='x')
    plt.show()

SampleConfInterval(200)
```

b) Code for #2

```python
#----------------------------------------
# Name: Roberto Sanchez
# Data: November 12, 2017
# Assignment 05 - Confidence Intervals
# Part 2
#----------------------------------------
import numpy as np
import matplotlib.pyplot as plt

def ConfInterval(n):
    # Given values from handout
    NumBaring = 1000000
    mean = 75
    stddev = 7.5
    successes_n95 = 0
    successes_t95 = 0
    successes_n99 = 0
    successes_t99 = 0
    M = 10000
    #calculate normal
    bearings = np.random.normal(mean, stddev, NumBaring)

    for j in range(M):
        sample = np.random.choice(bearings, n)

        mean_s = sum(sample)/n
        stddev_s = 0

        for i in range(len(sample)):
            stddev_s = stddev_s + ((sample[i]-mean_s)**2)

        stddev_s = (stddev_s/(n-1))**(1/2)

        z_95 = 1.96
        z_99 = 2.58
        #Finding upper and lower limits
        lowerLimit_n95 = mean_s-z_95*stddev_s/(n**0.5)
        upperLimit_n95 = mean_s+z_95*stddev_s/(n**0.5)

        lowerLimit_n99 = mean_s-z_99*stddev_s/(n**0.5)
        upperLimit_n99 = mean_s+z_99*stddev_s/(n**0.5)
        #Giving t values based on metrics
        if n == 5 :
            t_95 = 2.78
            t_99 = 4.60

```

```

        lowerLimit_n99 = mean_s-z_99*stddev_s/(n**0.5)
        upperLimit_n99 = mean_s+z_99*stddev_s/(n**0.5)
        #Giving t values based on metrics
        if n == 5 :
            t_95 = 2.78
            t_99 = 4.60

        if n == 40:
            t_95 = 2.02
            t_99 = 2.70

        if n == 120:
            t_95 = 1.98
            t_99 = 2.62
        #
        lowerLimit_t95 = mean_s-(t_95*stddev_s/(n**0.5))
        upperLimit_t95 = mean_s+(t_95*stddev_s/(n**0.5))

        lowerLimit_t99 = mean_s-(t_99*stddev_s/(n**0.5))
        upperLimit_t99 = mean_s+(t_99*stddev_s/(n**0.5))
        #
        if mean >= lowerLimit_n95 and mean <= upperLimit_n95:
            successes_n95 = successes_n95 + 1

        if mean >= lowerLimit_t95 and mean <= upperLimit_t95:
            successes_t95 = successes_t95 + 1

        if mean >= lowerLimit_n99 and mean <= upperLimit_n99:
            successes_n99 = successes_n99 + 1

        if mean >= lowerLimit_t99 and mean <= upperLimit_t99:
            successes_t99 = successes_t99 + 1
    # Calculating the successes for 99 and 95
    successes_n95 = successes_n95 / M * 100
    successes_t95 = successes_t95 / M * 100
    successes_n99 = successes_n99 / M * 100
    successes_t99 = successes_t99 / M * 100

    print("Normal Confidence 95% n =",n,":",successes_n95)
    print("Student Confidence 95% n=",n,":",successes_t95)

    print("Normal Confidence 99% n =", n, ":", successes_n99)
    print("Student Confidence 99% n=", n, ":", successes_t99)

ConfInterval(5)
ConfInterval(40)
ConfInterval(120)
```