HW1 - ISYE 6644

Marcos Grillo

January 2023

1

1.1 (a)

For the given data, μ is the following with a 95% confidence interval;

$$\mu \approx 1.602 \pm 0.5143$$

This result was obtained through the following code:

```
import numpy as np
import scipy.stats as st

ex1a = np.array([.43, .15, .54, .18, .93, .21, .64, .51, .72, .81,\
                 .87, .59, .98, .02, .28, .52, .51, .24, .77, .66])

def OneAFunction(x):
    return (2-(-2))*(np.exp((-2 + (2-(-2))*x)))/(1 + np.exp((-2 + (
                                        2-(-2))*x)**2))

output = OneAFunction(ex1a)

mean = np.mean(output)

interval = st.t.interval(0.95, len(output)-1, loc=mean, scale=st.
                                    sem(output))

print(mean)

print(mean - interval[0])
```

1.2 (b)

The required sample size to obtain a CI with half-length ≤ 0.02 is 515. This result was obtained via the following code:

```
import math

def OneBFunction(s_size, s_interval, d_interval):
    return math.ceil(s_size*(s_interval/d_interval))

print(OneBFunction(20, 0.5143, 0.02))
```

2

The true value of μ is:

$$\int_{-2}^{2} \left(\frac{e^t}{1+e^{t^2}}\right) dt \approx 1.42511$$

From a single run with 100 trials of 100 uniform(0, 1) observations, the fraction of 95% confidence intervals that contain the true value of μ is $\frac{93}{100}$ or 93%.

The following Python code was created to run this experiment:

```
def TwoFunction(num_samples, sample_size, conf, true_val):
    counter = 0
    for i in range(num_samples):
        sample = np.random.uniform(size=sample_size)
        output = OneAFunction(sample)
        mean = np.mean(output)
        interval = st.t.interval(conf, len(output)-1, loc=mean,
                                 scale=st.sem(output))
        if interval[0] <= true_val <= interval[1]:
            counter += 1
    return counter/num_samples

print(TwoFunction(100, 100, .95, 1.42511))
```

3

The following code was used to complete the two experiments requested:

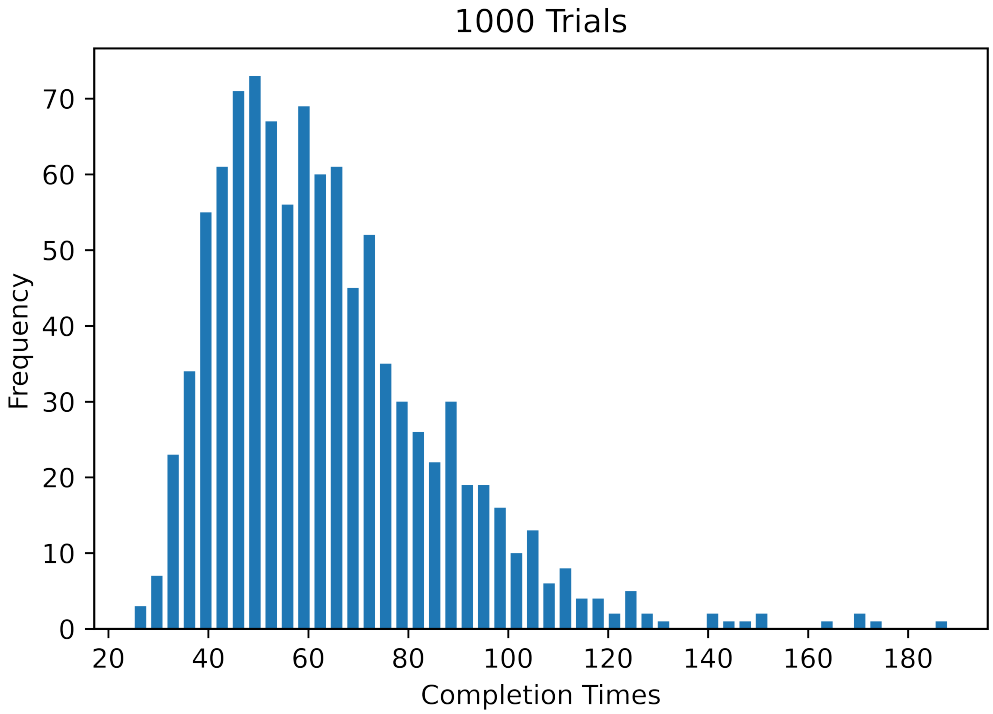
```
from matplotlib import pyplot as plt

def NRV(dist):
    return np.random.normal(dist, dist/4)

def exp(dist):
    return np.random.exponential(dist)

def ThreeFunction(trials):

    project_duration_total = np.array([])

    for i in range(trials):
        OneTwo = NRV(13)

        OneThree = NRV(5.5)

        TwoThree = exp(7)

        TwoFour = NRV(5.2)

        TwoSix = exp(16.5)

        ThreeSix = exp(14.7)

        FourFive = exp(6)

        FourSeven = exp(10.3)

        FiveSix = exp(20)

        FiveEight = exp(4)

        SixNine = NRV(3.2)

        SevenEight = NRV(3.2)

        EightNine = exp(16.5)

        arc_1369 = OneThree + ThreeSix + SixNine

        arc_1269 = OneTwo + TwoSix + SixNine

        arc_124789 = OneTwo + TwoFour + FourSeven + SevenEight +
                                        EightNine

        arc_124569 = OneTwo + TwoFour + FourFive + FiveSix +
                                        SixNine

        arc_124589 = OneTwo + TwoFour + FourFive + FiveEight +
                                        EightNine

        arc_12369 = OneTwo + TwoThree + ThreeSix + SixNine

        duration = max([arc_1369, arc_1269, arc_124789, arc_124569,
                                        arc_124589, arc_12369])
```

```
        project_duration_total = np.append(project_duration_total,
                                           duration)

    mean_time = np.mean(project_duration_total)

    interval = st.t.interval(0.95, len(project_duration_total)-1,
                                     loc=mean_time, scale=st.sem(
                                     project_duration_total))

    Five_per = np.percentile(project_duration_total, 5)

    TwentyFive_per = np.percentile(project_duration_total, 25)

    Fifty_per = np.percentile(project_duration_total, 50)

    SeventyFive_per = np.percentile(project_duration_total, 75)

    NinetyFive_per = np.percentile(project_duration_total, 95)

    percentiles = [Five_per, TwentyFive_per, Fifty_per,
                                     SeventyFive_per,
                                     NinetyFive_per]

    return (project_duration_total, mean_time, interval,
                                     percentiles)

output_1000 = ThreeFunction(1000)

hist, bins = np.histogram(output_1000[0], bins = 50)

width = 0.7 * (bins[1] - bins[0])
center = (bins[:-1] + bins[1:]) / 2
plt.bar(center, hist, align='center', width=width)
plt.title('1000 Trials')
plt.xlabel('Completion Times')
plt.ylabel('Frequency')
plt.show()

print('Confidence intervals for 1000 trials: ', output_1000[2])

print('Percentiles for 1000 trials:', output_1000[3])

output_10000 = ThreeFunction(10000)

hist, bins = np.histogram(output_10000[0], bins = 50)

width = 0.7 * (bins[1] - bins[0])
center = (bins[:-1] + bins[1:]) / 2
plt.bar(center, hist, align='center', width=width)
plt.title('10000 Trials')
plt.xlabel('Completion Times')
plt.ylabel('Frequency')
plt.show()

print('Confidence intervals for 1000 trials: ', output_10000[2])

print('Percentiles for 1000 trials:', output_10000[3])
```

The outputs for one run of each experiment are as follows:

For 1,000 trials:

Histogram:



μ with 95% confidence intervals (rounded to 3 d.p.):

$$\mu \approx 63.913 \pm (62.518, 65.309)$$

Percentiles are as follows (rounded to 3 d.p.):

5th percentile $\approx$ 36.880

25th percentile $\approx$ 47.417

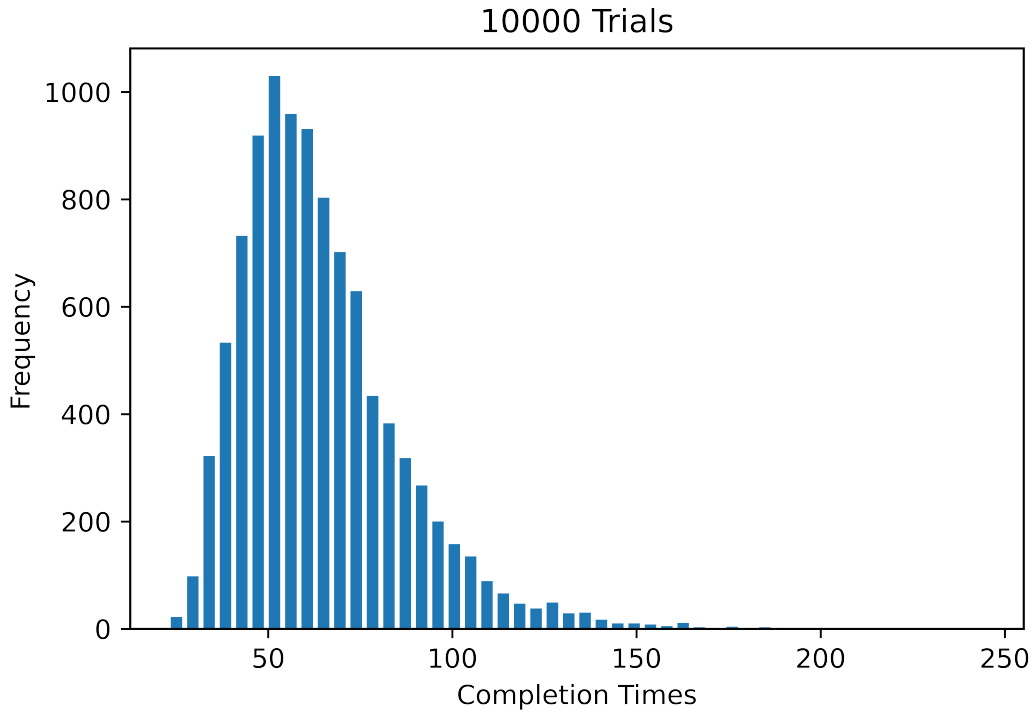
50th percentile $\approx$ 60.000

75th percentile $\approx$ 74.923

95th percentile $\approx$ 104.762

For 10,000 trials:

Histogram:



μ with 95% confidence intervals (rounded to 3 d.p.):

$$\mu \approx 64.140 \pm (62.715, 64.566)$$

Percentiles are as follows (rounded to 3 d.p.):

5th percentile ≈ 36.905

25th percentile ≈ 48.930

50th percentile ≈ 60.080

75th percentile ≈ 74.673

95th percentile ≈ 104.519

As the number of trials increases, we can see that the confidence intervals become tighter, while the percentiles remain roughly the same. to further illustrate this point, a run with 10,000 trials had the following statistics:

μ with 95% confidence intervals (rounded to 3 d.p.):

$$\mu \approx 63.967 \pm (63.832, 64.102)$$

Percentiles are as follows (rounded to 3 d.p.):

5th percentile $\approx$ 36.790

25th percentile $\approx$ 48.521

50th percentile $\approx$ 59.860

75th percentile $\approx$ 74.838

95th percentile $\approx$ 105.307

This, as seen in class, is empirical evidence of error diminishing with samples size (since our confidence intervals shrink) but our risk remaining the same, since our percentiles remain roughly in place throughout the different numbers of trials.

A stochastic activity network such as the one explored here will always have the same risk of a given length (assuming the network stays the same throughout) no matter the amount of trials undergone. The level of certainty we have in our outputs, however, will increase as trials increase.