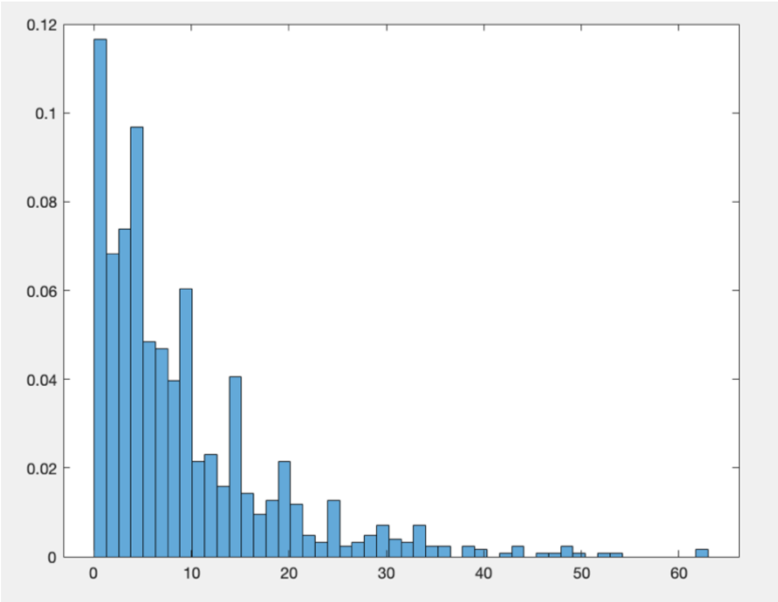
STU33009: Statistical Methods for Computer Science - Mid Term Assignment 2019-2020

Student name: Davy Nolan
Student number: 17330208

The following questions were answered through Matlab.

Question 1

(a) Below is a screenshot of the histogram plot produced after calling the **hist_pmf()** function with user 0's data:



(b) Calling the function **mean()** with user 0's data searches through the data to see which timings are more than 10ms and then increments a count variable. After this, the count is divided by the number of lines of data from user 0 to calculate the estimated empirical mean:

$$Prob(X_0 = 1) = 0.3060$$

(c) The function **con_inter()** produces 95% confidence intervals using the Central Limit Theorem (CLT) the Chebyshev inequality and the function **con_inter_bstrap()** produces 95% confidence intervals with bootstrapping. The bootstrapped methods run on 1000 random samples (with replacement) of 20% of the dataset. They produce the following results for user 0:

CLT	0.276855 < X < 0.335145
Chebyshev	0.240829 < X < 0.371171
Bootstrapped CLT	0.277035 < X < 0.335335
Bootstrapped Chebyshev	0.241003 < X < 0.371367

Central Limit Theorem (CLT)

Pros:

- ⇒ Gives a full distribution of $\overline{X}$
- ⇒ Only requires mean and variance to fully describe this distribution.

Cons:

- ⇒ It is an approximation when N is finite, and it is difficult to be sure how accurate it is.

Chebyshev's Inequality

Pros:

- ⇒ Provides an actual bound and not an approximation.
- ⇒ Works for all N.

Cons:

- ⇒ It is generally loose.

Bootstrapping

Pros:

- ⇒ Gives a full distribution of $\overline{X}$ and doesn't assume normality.

Cons:

- ⇒ It is an approximation when N is finite, and it is difficult to be sure how accurate it is.
- ⇒ Requires availability of all N measurements.

Question 2

Using the **mean()** function created in Q1(b), the estimated empirical means for the remaining users are as follows:

$$\begin{aligned} \mathit{Prob}(X_1 = 1) &= 0.5400 \\ \mathit{Prob}(X_2 = 1) &= 0.4080 \\ \mathit{Prob}(X_3 = 1) &= 0.2280 \end{aligned}$$

Question 3

The probability that the time a request **n** takes to complete exceeds 10ms is:

$$P(Z_n > 10) = \sum_i (Z_n > 10 \mid U_n = i) \cdot P(U_n = i)$$

Since $P(Z_n > 10 \mid U_n = i)$ is the same as ***Prob***(**X**$_i$ **= 1**):

$$P(Z_n > 10) = \sum_i Prob(X_i = 1) \cdot P(U_n = i)$$

The function **mean_zn()** returns the following estimated result:

$P(Z_n > 10)$ = 0.3260

Question 4

Calculate $P(U_n = 0|Z_n > 10)$ Using Baye's rule:

$$P(U_n = 0|Z_n > 10) = \frac{P(Z_n > 10|U_n = 0) \cdot P(U_n = 0)}{P(Z_n > 10)}$$

$$= \frac{Prob(X_0 = 1) \cdot P(U_n = 0)}{P(Z_n > 10)}$$

$$= \frac{0.306 \cdot 0.541703871975470}{0.326009778012153}$$

$$= 0.50845525502$$

Question 5

The **stoc_sim()** function runs a user-specified number of request simulations and for each simulation it randomly chooses the user from which the request was made using the provided probabilities and it also randomly checks to see if the request's duration was longer than 10ms.

I will be comparing the results to the answer from Question 3 which is **0.3260**. At first, I ran the function with the number of request simulations set to 10,000 ((i.e) req_sims=10000).
This gave a result of $P(Z_n > 10) = 0.3335$.
As you can see, the result is close to the answer, however there is a margin of error of $\pm$(0.0075).
Next, I decided to increase the number of request simulations to 100,000 to increase accuracy. This produced a result of $P(Z_n > 10) = 0.32571$. This is much closer to the actual answer from Question 3 with a margin of error of only $\pm$0.00029.

These results show that increasing the number of request simulations, thus increasing the run-time of the application, increases the accuracy of the simulation.

Appendix – Data

My downloaded user data can be found here: https://github.com/davynolan1/STU33009-Midterm-/blob/master/Data.txt
My downloaded user probabilities can be found here: https://github.com/davynolan1/STU33009-Midterm-/blob/master/Probabilities.txt

Appendix – Matlab Code

```
data = readtable('data.txt');
probs = readtable('probabilities.txt');

user0 = data.Var1;
user1 = data.Var2;
user2 = data.Var3;
user3 = data.Var4;

all_times = [user0,user1,user2,user3];
parsed_probs = probs.Var2;

%Q1_a
hist_pmf(user0);

%Q1_b
fprintf("\nQuestion 1 (b)");
fprintf("\nProb(X0 = 1) = %f",mean(user0));

fprintf("\n");

%Q1_c
fprintf("\nQuestion 1 (c)");
con_inter(user0);
con_inter_bstrap(user0);

fprintf("\n");

%Q2
fprintf("\nQuestion 2");
fprintf("\nProb(X1 = 1) = %f",mean(user1));
fprintf("\nProb(X2 = 1) = %f",mean(user2));
fprintf("\nProb(X3 = 1) = %f",mean(user3));

fprintf("\n");

%Q3
fprintf("\nQuestion 3");
fprintf("\nP(Zn > 10) = %f",mean_zn(all_times, parsed_probs));

fprintf("\n");

%Q5
fprintf("\nQuestion 5");
```

```
fprintf("\nSimulation P(Zn > 10) = %f",stoc_sim(all_times, parsed_probs,
100000));

function hist_pmf(u_data)
    histogram(u_data, 50,'Normalization', 'pdf');
end

function m = mean(u_data)
    onesCount = 0;
    for n = 1 : length(u_data)
        if u_data(n) > 10
            onesCount = onesCount + 1;
        end
    end

    m = onesCount / length(u_data);

end

function con_inter(u_data)
    N = length(u_data);
    m = mean(u_data);
    std_dev = sqrt(m - (m^2));

    cheby_higher = m + (std_dev / sqrt(0.05 * N));
    cheby_lower = m - (std_dev / sqrt(0.05 * N));

    clt_higher = m + (2 * (std_dev / sqrt(N)));
    clt_lower = m - (2 * (std_dev / sqrt(N)));

    fprintf("\nChebyshev .95 confidence interval: %f < X < %f",
cheby_lower, cheby_higher);
    fprintf("\nCLT .95 confidence interval: %f < X < %f", clt_lower,
clt_higher);
end

function con_inter_bstrap(u_data)
    N = length(u_data);
    S = 1000;

    means = [];
    for n = 1: S
        k = 0.2 * N; % 20 percent of dataset with replacement
        sample = randsample(u_data, k);
        means = [means, mean(sample)];
    end

    m = sum(means) / length(means);
    std_dev = sqrt (m - (m^2));

    cheby_higher = m + (std_dev / sqrt(0.05 * N));
    cheby_lower = m - (std_dev / sqrt(0.05 * N));

    clt_higher = m + (2 * (std_dev / sqrt(N)));
    clt_lower = m - (2 * (std_dev / sqrt(N)));

    fprintf("\nBootstrapped Chebyshev .95 confidence interval: %f < X <
%f", cheby_lower, cheby_higher);
```

```
    fprintf("\nBootstrapped CLT .95 confidence interval: %f < X < %f",
clt_lower, clt_higher);
end


function sum_mean = mean_zn(all_u_data, par_probs)
    sum_mean = 0;
    for i=1 : length(par_probs)
        sum_mean = sum_mean + mean(all_u_data(:,i)) * par_probs(i);
    end

end

function sz = stoc_sim(all_u_data, par_probs, req_sims)
    prob_timing = [];
    for k=1 : length(par_probs)
        prob_timing = [prob_timing, mean(all_u_data(:,k))];
    end

    count = 0;
    for i=1: req_sims
        ran_user_p = unifrnd(0,1);
        user_j = -1;
        sum_probs = 0;
        for j=1: length(par_probs)
            sum_probs = sum_probs + par_probs(j);
            if ran_user_p <= sum_probs
                user_j = j;
                break;
            end
        end
        ran_req_p = unifrnd(0,1);
        if ran_req_p <= prob_timing(user_j)
            count = count + 1;
        end
    end
    sz = count / req_sims;
end
```