A screenshot of a cell phone

Description automatically generated

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

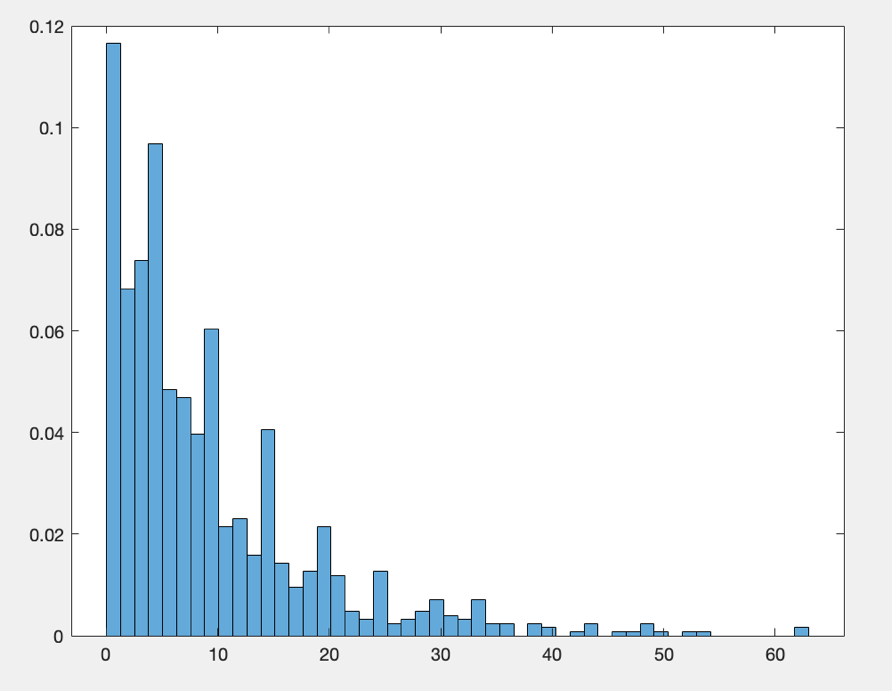
Student name: Davy Nolan

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*(X0  = 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 X̅
* 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 X̅ 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:

*Prob*(X1 = 1) = 0.5400

*Prob*(X2  = 1) = 0.4080

*Prob*(X3  = 1) = 0.2280

**Question 3**

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

Since is the same as ***Prob*(Xi= 1)**:

The function **mean\_zn()** returns the following estimated result:

= 0.3260

**Question 4**

Calculate Using Baye’s rule:

**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 .

As you can see, the result is close to the answer, however there is a margin of error of (0.0075).

Next, I decided to increase the number of request simulations to 100,000 to increase accuracy. This produced a result of . This is much closer to the actual answer from Question 3 with a margin of error of only 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