COMP312 Group Project

A Study at KK Malaysia Restaurant

A Well-known Local Take-away Store in Wellington CBD

Group number: 6

Date: 30, May 2018

Members:

- Andy Yang 30035616
- Minping Yang 300364234
- Haochen Yin 300377043
- Zhu Xinyu 300394847

INTRODUCTION

KK Malaysian Restaurant is a famous local take-away place in Central Wellington at 54 Ghuznee St (Phone: 385 6698). Our main purpose of this project report is to conduct an investigation on this service unit to gain better knowledge about the inter-arrival time of customers and the average service time. For the detailed procedure, we will firstly demonstrate how we had collected the customer data, how the data was processed and archived to generate an outcome for the observation. Secondly, we will perform an analysis with Chi-Square Goodness of fit Test and retrieve the best distributions for inter-arrival time and service time. Eventually, we take the distributions that best fit our data and simulate them as M2 to further compare with an empirical distribution M3 that we created from scratch. At the end, we will draw a conclusion that which model can best describe and simulate our service unit and get the most accurate result of our choice.

DATA COLLECTION PROCEDURES -ANDY YANG 300356106

Andy is primarily responsible for data collection procedures, contacting and getting permission from shop owner, data archiving and parsing/ calculating the required statistics (for analysis and simulations) from all collective datasets from the group members.

On 10, April, I gained the permission to collect customer arrival information from my manager, Owen Kw Tiow and actually started the data collection process on 16, April. As a part-timer of this Malaysian restaurant, I am interested in investigating where I have been working while applying what we are learning in this course. Our vision is to look into the average time a customer spent in the service unit and give some business suggestions to the shop owner. In total, we had visited KK Malaysia restaurant 12 times on different days to complete data collection task.

Mainly we only focus on the lunch rush period for our observation (11:30am-2:00pm). For each lunch shift, there are 2 front-desk waiters working. There are some major considerations for our chosen time. Firstly, since the size of KK Malaysian Restaurant is rather smaller than usual, meaning the capacity of accommodating customers is limited, we need to choose a time period with more probability that the restaurant can be relatively free (so that we are able to take a seat while concentrating on observing customer behaviors). Secondly, customers who visit our store during lunch period tend to order their meal faster than those who come at dinner time, resulting in a simplicity and ease of recording their service time. As for how we set up/ define our formats of collected data and some variances during observation process will be discussed in the following:

Customer Arrival	The timing when a customer opens the door and enter into our restaurant. (can be walk-in& dine-in, walk-in& take-away or take-away phone order when the phone starts to ring) In a simpler way, we treated a group-arrival (family, group of friends…) as a single arrival.
Service Begin	When a waiter/server approaches a customer or a group of customers to take their orders.
Service Completion	The moment when a customer/a table of customers receives their 'first' ordered meal. Considering the observers' burden of complexity, we treated the first plate of order arrival as service completion otherwise it can be hard for the observer to know exactly how many dishes a customer/ a table has ordered.

Note that we generalised the definition of the queue of our service unit. There will not be necessarily a physical formation of queue in front of the counter. Anyone who has arrived and stayed in the system without being taken his/her orders are perceived as queuing people.

DATA PARSING INSIGHT -ANDY YANG 300356106

A JOURNEY FROM REAL-LIFE DATA TO QUEUING THEORY SIMULATION

```
No.Arrival/ Arrival time/ Service Begin/ Service completion
*observation time range
From 11:30:00
To 13:10:30
Observer: Andy Yang
1 11:46:23 11:46:59 11:59:02
2 11:51:01 11:51:48 11:58:56
3 11:51:22 11:54:03 11:58:21
4 11:56:33 11:57:12 12:02:13
5 11:58:19 11:59:05 12:03:54
6 12:05:11 12:05:22 12:09:32
7 12:08:12 12:10:10 12:13:00
8 12:10:03 12:12:45 12:16:36
9 12:13:44 12:13:53 12:18:01
10 12:13:54 12:15:51 12:22:00
11 12:18:25 12:18:29 12:22:54
12 12:24:04 12:25:05 12:29:09
13 12:28:14 12:28:43 12:34:22
14 12:28:33 12:32:54 12:38:02
15 12:28:49 12:34:03 12:38:48
16 12:29:10 12:29:55 12:34:01
17 12:41:32 12:41:39 12:43:00
18 12:45:12 12:47:05 12:55:01
19 12:45:39 12:47:34 12:56:30
20 12:47:00 12:49:06 12:57:22
21 12:48:11 12:49:04 12:54:12
22 12:53:02 12:54:16 12:55:50
23 12:59:10 13:04:34 13:08:22
```

Method 1

```
#26
# date: Sat Apr 07, 2018 at 12:25:03  v1.26 Minping Yang
44703.861697  begin
44711.279891  1_A
44725.441920  1_B
44907.931890  2_A
45193.984837  2_B
45196.607965  1_C
45202.623014  2_C
45241.640787  3_A
45259.037637  3_B
45286.936914  4_A
45295.097109  4_B
45384.502446  3_C
45451.859786  5_A
45598.274109  6_A
45644.034713  5_B
45656.262224  6_B
45936.970127  4_C
46017.726423  5_C
46125.536069  6_C
46267.693107  7_A
46309.151072  7_B
46414.019257  7_C
46467.189127  8_A
46474.845139  8_B
46484.550090  9_A
46511.349656  9_B
46628.963574  10_A
46667.235337  10_B
```

Method 2

We have got two different kinds of dataset format corresponding to the preference that the observer feel more comfortable with. Consequently, I need to write 2 different Python codes to parse and store our data (read.py, read2.py) for further analysis. The picture on the right shows the observation result using monitor.py (the event monitor in python provided on Project webpage) and the left-hand-side image is the data format created by manual written approach.

In order to parse the service time and waiting time in the queue from Method 1 dataset format, I split each line of customer instance into 4 blocks by a space blank (named them as No-customer, Arrival, Begin, Completion respectively). Continuously, according to each Arrival, Begin, Completion blocks, I split the entire time block into three sections representing hour, minute, and second individually.

```
for line in fp:
    No_customer, Arrival, Begin, Completion= line.split(" ")
    arrival = [int(n) for n in Arrival.split(":")]
    begin =[int(n) for n in Begin.split(":")]
    complete =[int(n) for n in Completion.split(":")]
```

For the next step, I applied .datetime() function along with dateDiffInSeconds() function (imported from python standard library module) to calculate the time differences and store my result into a list.

```
    #W time spent in system
    W_1= datetime.datetime(2017, 04, 24, arrival[0],arrival[1],arrival[2])
    W_2= datetime.datetime(2017, 04, 24, complete[0],complete[1],complete[2])
    sum_W+=(dateDiffInSeconds(W_1,W_2)/60.0)
    w_list.append(dateDiffInSeconds(W_1,W_2)/60.0)
```

The reason why I still needed to compute Ws and Wq was to confirm that our W was calculated correctly as a sum of Ws and Wq.

```
#Ws
Ws1= datetime.datetime(2017, 04, 24, begin[0],begin[1],begin[2])
Ws2= datetime.datetime(2017, 04, 24, complete[0],complete[1],complete[2])
Ws.append(dateDiffInSeconds(Ws1, Ws2)/60.0)

#Wq
WQ1= datetime.datetime(2017, 04, 24, arrival[0],arrival[1],arrival[2])
WQ2= datetime.datetime(2017, 04, 24, begin[0],begin[1],begin[2])
Wq.append(dateDiffInSeconds(WQ1, WQ2)/60.0)
```

The monitor.py already keeps the recorded time in seconds so that I did not need to further parse and convert the figures through python datetime() function. Each recording method has its advantages and drawbacks. For example, the event monitor did not arrange the order of [Arrival, Begin, Completion] for each customer so that it was a hard job to clean the data (e.g. missing data) and sort the data. 76% $(\frac{214}{279})$ out of 279 total observations were established using Method 3, while the rest of them 23% $(\frac{65}{279})$ used Method 2. These computed variables are listed below:

Variables	How do we compute?	Descriptions
Wq	$\text{Service Begin} - \text{Arrival}$	The average time a customer spent in the queue.
Ws	$\text{Service Completion} - \text{Service Begin}$	The average time a customer spent to wait for the service to complete.
W	$Ws + Wq$	The average time a customer spent in the entire system (our store).
$Interarrival$	$(n+1)\_Arrival - n\_Arrival$	The time gap between 2 arrivals. (in minute)
$E(S)$	AVG [$n\_C - n\_B$ (Method 2) or the third column of time subtracts the second column of time (Method 1)]	The averaged service time

These are some preliminary numerical statistical figures computed from the parser python code (read.py and read2.py) which are what we need to use or to be compared with the simulation results later on.

$$Total\ number\ of\ observations = 279$$

$$Total\ observation\ time = 1299.27 \text{ minutes}$$

$$Arrival\ rate\% = 0.215 \text{ per minutes } (12.8/\text{hour})$$

$$E(S) = \frac{1}{\mu} = \frac{1}{0.2123} = 4.71\ minutes\ (average\ service\ time)$$

$$P_0 = \left[\sum_{n=0}^{c-1} \frac{\rho^n}{n!} + \frac{\rho^c}{c!} \left(\frac{1}{1 - \frac{\rho}{c}} \right) \right]^{-1}$$

The M/M/c python code given as an image was what I had done to estimate some critical results from the theoretical queuing formula. It is important to note that we got $\frac{\rho}{c} < 1$, which indicated that the steady state distribution for our model exists. (see the next page for the output from the code)

```
================== RESTART: /Users/andy-yang/Desktop/m1.py
W= 5.91275543146 minutes (theoretical result)
Wq= 1.20323843952
L= 1.27124241776
Lq 0.258696264497
π0= 0.242984899809
π1= 0.246033425603
π2= 0.124560099335
π3= 0.0630614247159
π4= 0.0319263015078
π5= 0.0161634268899
π6= 0.00818310786047
λ= 0.215 (arrival rate)
Mu= 0.212336 (service rate)
E(S)= 4.70951699194 minutes (expected service time)
ρ/c = 0.506273076633
∴ SSD∃
```

```
import random
import numpy
import math
#Theoretical Result from M/M/c formula
"""observed W= 5.9608 minutes
(overall average waiting time for a customerin the system)"""

def MMc(lamb, mu, c):
    rho= lamb/mu
    #get Pi[0]:
    pi0= ((rho**0)/math.factorial(0)+(rho**1)/math.factorial(1)+
          ((rho**c)/math.factorial(c)*(1/(1-rho/c)**2)))**(-1)
    Lq= (rho**c/math.factorial(c)*pi0)*(rho/c/(1-rho/c)**2)
    Ls= rho
    L=Lq+Ls
    W=L/lamb
    print "W=" ,W, "minutes (theoretical result)"
    print "Wq=", Lq/lamb
    print "L=", L
    print "Lq", Lq
    print "π0=", pi0
    print "π1=", pi0*(rho)
    print "π2=", pi0*(rho**2/math.factorial(2))
    print "π3=", pi0*(rho**3/(math.factorial(2)*(2**1)))
    print "π4=", pi0*(rho**4/(math.factorial(2)*(2**2)))
    print "π5=", pi0*(rho**5/(math.factorial(2)*(2**3)))
    print "π6=", pi0*(rho**6/(math.factorial(2)*(2**4)))
    print"λ=", lamb, "(arrival rate)"
    print"Mu=", mu, "(service rate)"
    print"E(S)=", 1/mu, "minutes (expected service time)"
    print "ρ/c =",(rho)/c
    if (rho/c)<1:
        print "∴ SSD∃"
    else:
        print" >1 ∴SSD not exists"
#Calculate with our observation data results
overall_lamb= 0.215
overall_mu= 0.212336
c=2
MMc(overall_lamb, overall_mu, c)
```

In our selected server unit (2-server unit), the result of Wq of each customer can vary from time to time due to customer behaviors and also the occupation of the system. During lunch period, our store tends to gain more businessmen customers longing for quick meal with colleagues. Sometimes we have a businessman comes and takes a table, waiting for his or her co-workers to arrive for a lunch or a mom with children come (as a family arrival), waiting for her husband to park the car nearby to dine in. With these scenarios occurring, Wq values will increase and further affect W. Besides, both Wq and Ws can be influenced by system congestion. Say if in a certain section of our dataset when a sequence of $interarrival\ time$ have a comparatively smaller value, we can make an inference that Wq and Ws among these customers will increase. Since the system is suddenly filled up with a surge of arrivals during peak time (e.g. lunchtime normally between 12:15 to 13:00), it is more likely that all servers (waiters), kitchenhands and chefs are occupied (busy) under this time-period. Therefore, the food processing time becomes longer and the probability which a waiter can be idle to take your order gets lower.

The results that we are interested in for our investigation purpose:

$$observed\ Wq = 1.2515 * 76\% + 1.2507 * 23\% = 1.2387\ minutes$$

$$observed\ Ws = 4.5132 * 76\% + 5.369 * 23\% = 4.6649\ minutes$$

$$observed\ W = Ws + Wq = 1.2387 + 4.6649 \cong 5.9608\ minutes$$

Moreover, observers have been encountering many regular customers who visit our restaurant very frequently, which means that they prefer to step in and tell the waiter what they want to order directly while the server is guiding them to the seats. In this circumstance, Wq turns out to be smaller. By contrast, there were customers coming (or calling) to grab a take-away meal who left and came back later to collect the order. Thus, in this case, those particular $E(S)$ service times will increase and further affect Ws since we had assumed the service completion is when a customer receives his or her order.

DATA ANALYSING- MINGPING YANG 300364234

I have taken part in every part through the entire project. Furthermore, I mainly responsible for analysing data part and the model 3 of modelling the performance of three different models part.

In the data analysis part, chi-square goodness of fit approach is used to check the best fit distribution for both interarrival times and service time. Python program is used to implement the chi-square goodness of fit.

Firstly, load observed interarrival time and service time from txt files into two different lists on the python program. Secondly, use below codes to find the parameter of corresponding distribution.

Find alpha and beta parameter of gamma distribution:

```
fit_alpha, fit_loc, fit_beta = ss.gamma.fit(data, floc=0)
```

ss represents the scrip.stats library. data is a list containing either interarrival time or service time. the parameter estimation in ss.gamma is E(X) = alpha*beta, V(X) = alpha*beta*beta.

Find alpha and beta parameter of Erlang distribution:

```
fit_alpha, fit_loc, fit_beta = ss.erlang.fit(data, floc=0)
```

the parameter estimation in ss. erlang is $\mathrm{E(X)} = \frac{alpha}{beta}$ $\mathrm{V(X)} = \frac{alpha}{beta*beta}$

Find lambda of Exponential distribution:

```
fit_lamda, fit_loc, = ss.expon.fit(data, floc=0)
```

the parameter estimation in ss.expon is $\mathrm{E(X)} = \frac{1}{lamda}$, $\mathrm{V(X)} = \frac{1}{lamda}$

Thirdly, calculate the number of bins by

Original formula: **Number of bins = $(1+\log_2 N)$** → **in python:** **k=int(1+math.log(279,2))**

N is the number of loaded data. If data is interarrival time, the number of loaded data is 270. If data is service time, the number of loaded data is 279. Because we collected data in ten different periods, which means that there are 9 interarrival times which cannot be calculated out from observed data set.

Fourthly, make sure to adjust the number of parameters of corresponding distribution.Exponential distribution has one parameter.Both gamma distribution and erlang distribution have two parameters.

Finally, using python program and above information to calculate the test statistic (X^2) and the p-value for both interarrival time and service time with gamma distribution, erlang distribution and exponential distribution.

	Gamma	Erlang	Exponential
Interarrival time	test statistic: 21.038 p-value: 0.00031	test statistic: 281.79 p-value: $6.498*10^{-58}$	test statistic: 20.76 p-value: 0.00089
Service time	test statistic: 1.05 p-value: 0.305329	test statistic: 68.04 p-value: $1.60*10^{-16}$	test statistic: 130.074 p-value: $2.295*10^{-26}$

Before make conclusion from above data, each pair of test statistic and p-value should be manually checked if they are all reasonable. In other words, if higher p-value should be corresponding to lower test statistic. Otherwise, they are invalid.

For a chi-square goodness of fit test, the hypotheses take the following form.

Ho: The data are consistent with a specified distribution.

Ha: The data are not consistent with a specified distribution.

Therefore, as above table shown, Interarrival data is most likely are Exponential distributed, since the p value of Exponential distribution is highest in the three distributions, which means that it mostly unlikely to against Hypothesis null statement. Similarly, service time data is most likely are Gamma distributed with parameters (alph:3.876165 beta:1.214991). since the p value of Gamma distribution is highest in the three distributions.

DRAWING TWO HISTOGRAMS- MARK 300394847

I mainly took part in drawing histogram for data analysis. Using the given code with a little tweaking it is not hard to fit our data in and calculate the corresponding p-values and test statistics for exponential distribution, gamma distribution and erlang distribution. Service times are most likely gamma distributed because the p-value calculated for it is the highest (p-value = 0.305329) among the three kinds of distribution. Following the same reasoning inter-arrival times are most likely exponentially distributed as the p-value for it is the highest (p-value = 0.000899).

Something worth noticing is that the highest p-value for service time is 0.305329, which is much higher than either 0.05, 0.01 or 0.1, whereas the calculated p-value for inter-arrival time is only 0.000899. The latter can be interpreted that the probability of wrongly rejecting null hypothesis (The data is consistent with an exponential distribution) is 0.000899. In other words, the probability of our inter-arrival times being exponentially distributed is exceptionally low. We would say the chance that service times are ***not exponential distributed*** is with probability 1- 0.000899 = 0.999101.

Going on to drawing histograms, the code below is used for drawing histogram.

```
myHist = plt.hist(data, k, density=True)
# plot the density of this gamma rv
x = np.linspace(0.001,max(data)+10)
h = plt.plot(x, rv.pdf(x), lw=2)
#show them together
plt.show() # if you want to see your picture uncomment this line
```

```
myHist = plt.hist(data, k, density=True)
```

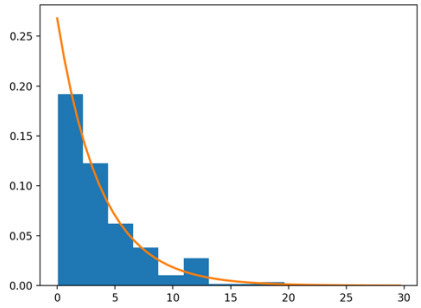
This line plots a histogram using 2 parameters. Data is our collected data parsed into a collection in python, k is the number of bins.

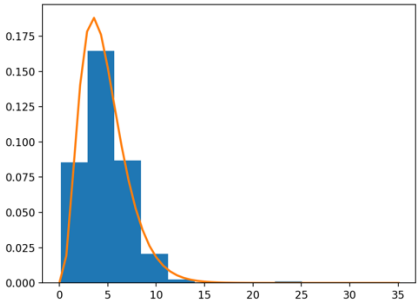
```
x = np.linspace(0.001,max(data)+10)
```

This line specifies the interval between the points. The starting value of the sequence is 0.001, and the end point of the sequence is max(data) + 10

```
h = plt.plot(x, rv.pdf(x), lw=2)
```

This line draws the density plot using the estimated gamma random variate rv. Output histograms are shown below:





Inter-arrival Histogram **Service time Histogram**

Density plot smooths out sharp data in a way that highlights the curve that shows what kind of distribution it is. In this graph it is clearly exponential distribution.

MODELING (M1,M2)- HAOCHEN YIN 300377043

In this part, we will use three different models to calculate the Average time spending in the queue (Wq), average time spending in the service (Ws) and average time spending in the system. And compare to the observed performance measures that we calculated from data analysis part to find out which simulation best suits our experiment. We choose to compare time instead of number of customer is because time is more straightforward. For example, if you are in a restaurant and waiting to be served, a serve tells you there are 5 people before you or a serve tell you that you need to wait 5 minutes to be served. It is easy to see time is more clearly than number of customer, so we choose to use average time to compare our simulations.

For the first M/M/c model, because in our experiment, the number of server is 2, so our first model is M/M/2. To modify a model with Poisson arrival and exponential service time. We need to know the arrival rate (lambda) and service rate (mu) before we start to perform the model. In the data analysis part, we already get the Lamda and Mu from the data that we collected. We calculate the Lambda using the formula $Lamda = \frac{1}{mean}(Interarrival\ time)$ (lamda = 0.267770516023) and Mu using the formula $Mu = \frac{1}{mean}(Service\ time)$. (mu= 0.212336486777)

After we get the lamda and mu, we can easily put these values into the simpy program (q3.py). We compute Estimate of W, Wq and Ws and Confidence interval of W, Wq and Ws to compare with the observed W, Wq and Ws.

Finally, we get the result of our M/M/2 model.

- The estimate of W is 7.82337206711 and with 95% confidence interval, the W is (7.809121917413742, 7.837622216805047).
- The estimate of Wq is 3.11337407736 and with 95% confidence interval, the Wq is (3.0948807792369624, 3.1318673754882353).
- The estimate of Ws is 4.70833321818 and with 95% confidence interval, the Ws is (4.704664109569529, 4.7120023267810565).

Comparing to the observed W we calculate from Data Analysis part (W = 5.9608), the W we get from the M/M/1 model has significant difference. So, it is not a suitable simulation for our experiment.

From the result we get from data analysis, the interarrival time best fit distribution is exponential distribution with parameter (lambda = 0.267770516023) and the service time best fit distribution is Gamma distribution with

parameter (alpha = 3.876165, beta = 1.214991). The detailed description how we calculated this result is motioned in the data analysis part.

After we get these important parameter, we can build up M2 with exponential interarrival time and gamma service time. Finally, we get the result of M2 model. There is one thing need to be mentioned here, in the random library, the default probability distribution function for gamma distribution is

$$pdf(x) = \frac{x^{alpha-1} * math.\exp(\frac{-x}{beta})}{math.gamma(alpha) * beta^{alpha}}$$

- Estimate of W: 6.7056019466 and with 95% confidence interval, the W is (6.679096919947189, 6.73210697325375)
- Estimate of Wq: 1.99837885424 and with 95% confidence interval, the Wq is (1.9742759920546804, 2.022481716421073)
- Estimate of Ws: 4.70721556874 and with 95% confidence interval, the Ws is (4.703045184310918, 4.7113859531765705)

Compare to the observed W we calculate from Data Analysis part (W = 5.9608), the W we get from M2 is about 0.8 minutes higher. However, it is better than the result we get from M1.

EMPIRICAL MODEL (M3)- MINGPING YANG 300364234

Coding a python program to build an empirical distribution model. Then, use simpy to build a simulation that the both interarrival time and service time are empirical distributed.

Firstly, I coded an own-defined python function:

```
def empericalvariate(fList,wList):

    l1= getK(fList)

    r=l1[0]

    k=l1[1]

    v1= (wList[k+1]-wList[k]) * (r-fList[k])/(fList[k+1]-fList[k])

    x= wList[k]+v1

    return x
```

Basically, the above method implements below formula to return a random variable x which are empirical distributed.

$$x = W^{(k)} + (\frac{r - F(k)}{F(k+1) - F(k)})(W^{(k+1)} - W^{(k)}) \qquad \text{where } W^{(0)} = F^{(0)} = 0$$

r is a random number from a uniform U(0,1) distribution. Then 0<=r<1.

Find K which is the largest k such that

$$F^{(k)} <= r$$

Use python program to implement find the corresponding k, shown as below:

```
def getK(fList):
    r= random.random()
    k=0
    result=[]
    for f in fList:
      if r<f:
          k=fList.index(f)-1
          break
    result.append(r)
    result.append(k)
    return result
```

In above python function, fList represents the F List which can be calculated from below formula: F list={$F^{(1)}$, $F^{(2)}$, $F^{(3)}$, $F^{(4)}$, $F^{(5)}$, $F^{(6)}$,.....}

$$F^{(i)} = \frac{number\ of\ observations\ Xj\ \ that\ are <= W(i)}{n}$$

n is the size of sample data.

i is from 0 to number of unique data in the sample data.

j is from 0 to size of sample data.

Using python code to find F List shown as below:

```
def empericalDist(tempList,xList):
    tempList.sort()
    n =float(len(xList))
    result= getList(tempList,xList,n)
    return result
```

The parameter tempList represents to w List= { $W^{(1)}$, $W^{(2)}$, $W^{(3)}$, $W^{(4)}$, $W^{(5)}$, $W^{(6)}$,.....}

The parameter xList represents to x List= {$X^{(1)}$, $X^{(2)}$, $X^{(3)}$, $X^{(4)}$, $X^{(5)}$, $X^{(6)}$,.....}

X list is the list of the sample data, in the project, X list is the list of interarrival time or a list of service time.

W is sorted list of X list without duplicates value.

From above information, once a sample dataset is loaded into programme, an empirical distribution random variable would be generated according to the sample dataset. Furthermore, because, in model 3, both interarrival time and service time are empirical distributed, their empirical distribution would be different due to different dataset loaded.

Therefore, server time are empirical distributed by using python program:

```
t = empericalvariate(fList_server, wList_server)
```

fList_server represents the F List is generated from service time data.

wList_server represents the W List is generated from service time data.

Interarrival time are empirical distributed by using python program:

```
t = empericalvariate(fList_arrive, wList_arrive)
```

fList_arrive represents the F List is generated from interarrival time data.

wList_arrive represents the W List is generated from interarrival time data.

By using model 3, the estimated W, estimated Ws and estimated Wq is the most closely to the observed W, observed Ws and observed Wq, in three different simulation models. So, the model 3 is most likely correct because of the reasonable conclusion.

To sum up, by comparing the estimate (W, Wq, Ws) from three different simulations and observed (W= 5.9608, Wq= 1.2387, Ws= 4.6948). Model 3 is the fittest model for our experiment. The average waiting time in the system(W=6.51688285922) is only 0.6 minutes higher than the observed value (W = 5.9608). This is the best simulation from three different simulations.

CONCLUSION

In conclusion, based on our 279 observations data, we applied the Chi-Square Goodness of fit approach, and found out that the fittest distributions for our inter-arrival time and service time is exponential distribution and gamma distribution respectively. As for the simulation modelling section, by comparing observed W and estimated W from three different simulation models, we draw a conclusion that the empirical model performs best, the fittest distribution model performs better than M/M/c model. As a result, the empirical model can represent our service unit most accurately.