Supermarket Queuing Systems

Group 3

June 3, 2022

1 Introduction (Jack)

Our group has been studying the queuing system of a large supermarket, New World Wellington City (near Oriental Bay). We collected data from three individual queues, each with one server and space for several people to line up, taking down the arrival and departure times of customers in those specific queues. The data has been entered into a text document and python code written to read the data. Histograms and best fitted densities for inter-arrival and service times have been fitted as well as multiple chi-square tests to find the best fitted distribution. Three models have been simulated, MMc (theoretical), our best fitted model (normal distribution), and Empirical distribution; with varying results.

2 Data collection process (Eugenie Neiertz)

First, we have contacted the supermarket and got approval from the manager to come collect our data the Sunday 3rd of April. We chose to come on a weekend day as we thought there would be a regular influx of customers, with low break time periods.

When we arrived, the store's supervisor assigned us to the cashiers located near the end of the shop, next to the self-checkouts. This is a relatively strategic placement as they are the first cashiers available when the customers have finished their shopping, at the end of the shop.

We started to collect the data at about 2:15 pm and finished around 4:30 pm. We initially started with two queues, dividing the team into two groups of two. But we quickly realized that we had the capacity to take on an extra queue and collect more data. Thus, we collected data from three queues at the same time, which allows us to cover more influx of customers. Our first impressions were that the three queues had a similar rhythm, with a constant arrival of customers, and none of them had a specifically long wait during the time we were present, often one or two customers lining up in the queue. There was 6 other cashiers opened during the time of the collection, located sparsely along the cashier line.

We collected 169 datapoints and for each of them, we gathered two information, as followed:
- customer arrival/service begins: when a customer choose a queue and start waiting with his trolley;
- customer leaving/service ends: when the customer leaves the cashier with his trolley.

3 Data Parsing (Kyle Hennessey)

Original Data Format

The original data format was a text file which had two columns: one for the arrival time for a customer and one for the system exit time for a customer. Each customer's arrival and exit time was on it's own line, with each time seperated by a space which created the two 'columns'. The precision of each time was down to the seconds, with the hour, minute and seconds of the time split by a period. For example, one customer's arrival and exit time was formatted as follows:
2.37.08 3.39.41

Script/Program (Written in Python)

The script first begins by defining a Customer class which has fields for the arrival time and exit time. Then the script iterates through each line from the text file and splits the two times as strings into a list by using a space as a delimiter. Each time also has the periods replaced with colons to allow for the string to be easily converted into a time object which will allow for calculations. Each time is converted into a datetime object using the datetime.strptime method. The two times are then used to create a Customer object which is added to a list of all customers which.

The script then has two methods which calculates λ and μ. Each method is given the list of all customers as an input.

The method which calculates λ is also given the time range of when we observed the data. This is calculated by subtracting the time we started observing our data (2:14:14) from the time we finished observing our data (4:24:50). This gives us a datetime object which we can then divide by the number of customers we observed during the time period. This gives us λ which is 46.37 seconds (2dp).

The method which calculates μ iterates through each Customer object in the list of all customers and finds it's service time by subtracting their arrival time from their exit time. The service time is then added to a list which contains each of the customer's service time. Then by using numpy's mean method, we get the average time from the list which gives us μ which is 132.40 seconds (2dp).

Note: to see the raw data file, please see the attached "NW Data.txt" file. To see the data parsing script, please see the "data-parser.ipynb" file.

4 Data analysis (Eugenie Neiertz)

4.1 Choose the best fit distribution

After parsing the data, we have used the chi-square goodness of fit approach to determine which distribution would have the best fit. We have decided to compare the following distributions: Normal, Erlang, Gamma and Exponential.

To get the inter-arrival time, we gathered in a list all the time of customer arrival/beginning of service. To get the service time, we gathered in a list the difference between the customer leaving time and its arrival.

Note: ss represents the scipy.stats library used in our Python code.

Gamma distribution, fitted with 2 prameters

```
fit alpha, fit loc, fit beta = ss.gamma.fit(data, floc=0)
```

Erlang dsitribution, fitted with 2 parameters

```
fit alpha, fit loc, fit beta = ss.erlang.fit(data)
```

Normal distribution, fitted with 2 parameters

```
fit alpha, fit loc = ss.norm.fit(data)
```

Exponential distribution, fitted with 2 parameters

```
fit alpha, fit loc = ss.expon.fit(data)
```

Here below are the results of the chi-square and p-value that we obtained for all of them:

	Gamma	Erlang	Normal	Exponential

Inter-arrival times	chi-square = 30.39
	p-value = 0.17
		chi-square = 29.20
		p-value = 0.21
			chi-square = 61.06
			p-value = 0.000
				chi-square = 15.18
				p-value = 0.89

Service times	chi-square = 26.21
	p-value = 0.45
		chi-square = 24.03
		p-value = 0.57
			chi-square = 44.06
			p-value = 0.005
				chi-square = 67.31
				p-value = 0.000

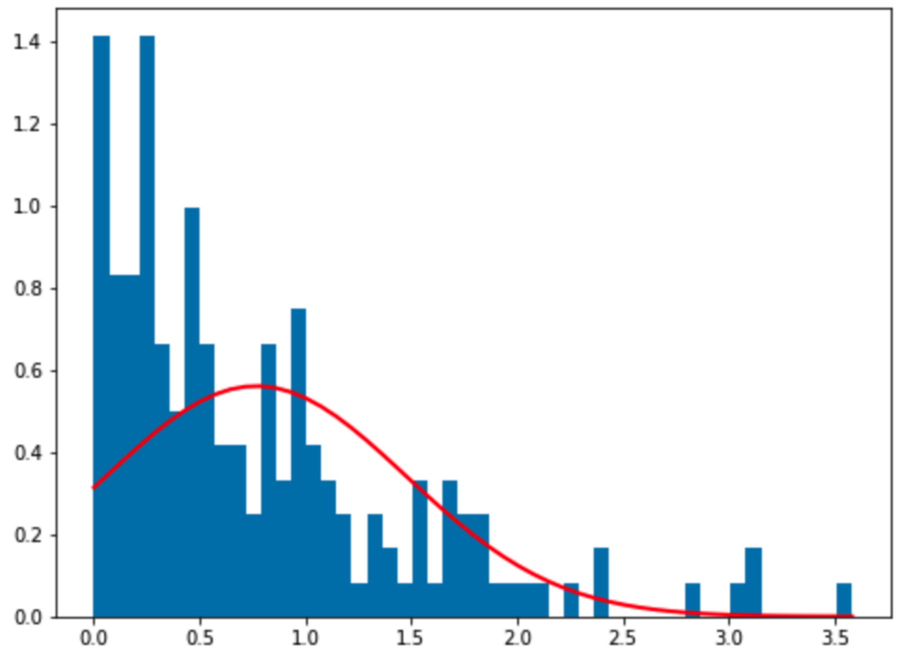
After testing for those 4 distributions, and considering the following hypothesis:
H0: the data fits the specified distribution
H1: the data doesn't fit the specified distribution

We could conclude that for the inter-arrival time, the best fit distribution would be Exponential, and for the service time, the best fit distribution would be Erlang. However, we decided to use the Normal distribution for both inter-arrival times and service times to fit our data on the following plot with the reason behind it being mainly due to the nature of business. We realized that the sample we took was a small portion of the business operation and not representative enough of the actual movement in the store over time. But we had observe a larger period of time, we would expect the plot to fit well a normal distribution.

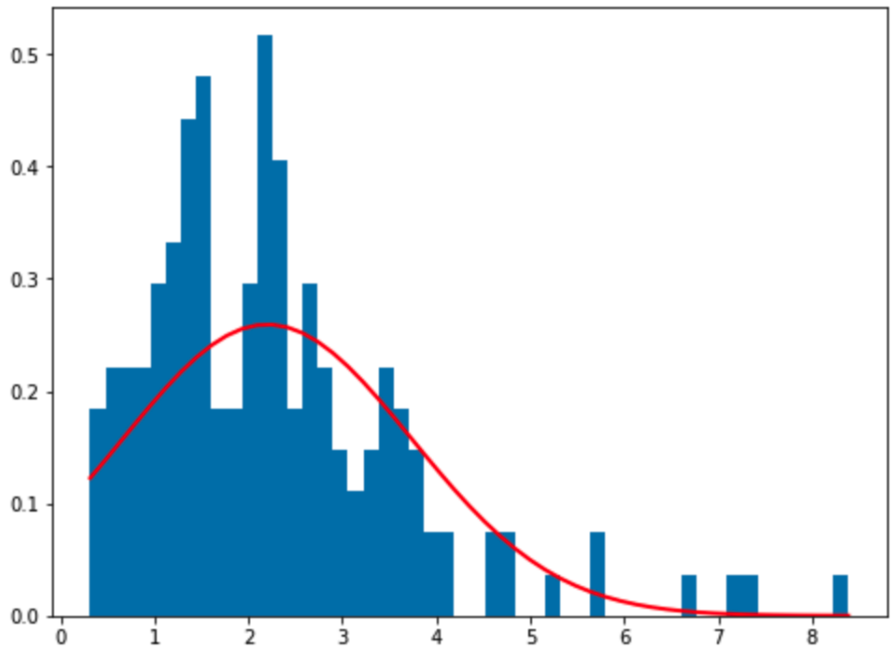
4.2 Histograms plots for inter-arrival time and service time

After choosing the best fit distribution, we have run two histograms, one for the inter-arrival time and one for the service time with normal distribution density fit.

We can observe similarities between the two plots: when there is a peak of arrival, we can mirror a peak in the service time plot, with slight delay, but we can identify the few busy periods.



Inter-arrival time



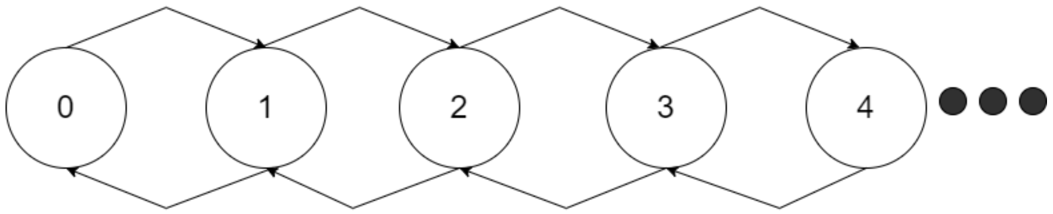
Service time

5 M/M/S Model (Kyle Hennessey)

M/M/3 Queue

$\lambda_i = \lambda = 46.37 seconds$

$\mu_j =$
0 if j = 0
j$\mu = j \times 132.4$ if j = 1,2,3
s$\mu = s \times 132.4$ if s = s+1, s+2 ...



$$\begin{cases} \lambda_i = \lambda = 46.37 \\ \mu_j = \begin{cases} 0 \text{ if } j = 0 \\ j\mu = j * 132.4 \text{ if } j = 1,\ 2\ ,3 \\ s\mu = s * 132.4 \text{ if } s = s+1,\ s+2,\ s+3\ \ldots \end{cases} \end{cases}$$

Simulation results from SimPy

To simulate the performance of the queue beyond the number of customers that we observed, we created a M/M/3 simulation model using SimPy. The model used the λ and the μ from our observed data and simulated 50 separate instances of 10,000 customers entering and leaving our system. For each simulation we collected performance measures which included: the average waiting time, average number of customers queued in the system, the average service time and the average effective lambda that the model observed.

The performance measure results were as follows:
Average waiting time: 536.28 seconds (2dp)
Confidence interval: (459.12 seconds, 613.43 seconds) (2dp)

Average number of customers in system: 11.72 (2dp)

Confidence interval: (9.93, 13.51) (2dp)

Average service time: 166.90 seconds (2dp)
Confidence interval: (163.28 seconds, 170.53 seconds) (2dp)

Average proportion busy: 98.08 percent (2dp)
Confidence interval: (97.67 percent, 98.50 percent) (2dp)

Lambda effective: 0.02165 per second (5dp)
Confidence interval: (0.02135 per second, 0.021943 per second) (5dp)

Analysis of the Simulation

Two of the performance measures returned expected results within a region of leeway, but the other three performance measures were wildly unexpected and unrealistic.

The best performance measure was the Lambda effective. The observed lambda from our system was a customer arriving every 46.37 seconds. As the calculated Lambda effective is per second, we need to multiply its value by 46.37 to find how accurate Lambda was. The closer this value is to one, the more similar it is to the observed lambda. 0.02135 * 46.37 = 0.98999 which shows that the lamda effective was nearly perfect and it lies within the confidence interval for the observed lambda.

The second best performance measure was the average service time, which is arguably the most important performance measure together with Lambda effective. The average service time performed well as it was only 44 seconds longer than the observed average service time. The performance measure's confidence interval did not fall within the observed average service time so the service time is unlikely to be exponentially distributed.

While the previous two performance measures performed fairly well, all other performance measures performed terribly. The average waiting time performance measure was 536.28 seconds while the observed waiting time was 224.38 seconds. The performance measure was almost double of what we observed and its value is very unrealistic when applied to the real world system as it is extremely unlikely that on average a customer would have to wait just under nine minutes in the system. The lowest bound of the performance measure is still 235 seconds longer than what was observed which is a very large difference.

Another performance measure which performed terribly was the average number of customers in the system. The observed average number of customers in the system was 4.40 while the performance measure value was 11.72. This is a worse result than the average waiting time as it is almost three times larger than what was observed in the real world system. The lower bound of the performance measure is still more than double than the observed value and realistically the performance measure value is wildly unrealistic in the real world system.

The final performance measure was the average proportion of time a server was busy serving a customer. The observed value was 92.20 percent while the performance measure result was 98.08 percent. The lowest bound of the performance measure's confidence interval was 97.67 percent which is still fairly larger than the observed value. While the two values may not seem to be wildly different, the performance value suggests that over a 2 hour and 10 minute time period, a server would have only 2 minutes where they would not be serving any customers. What we observed was that there would be 10 minutes where a server would not be serving any customer which is 5 times more than what the performance measure suggests. This is a very large difference and the performance measure value is very unrealistic when applied to the real world system as a server would never be busy for 59 minutes every hour

Conclusion of the M/M/3 Model

The M/M/3 simulation model did not perform well, with most performance measures performing terribly, providing contradictory results to what was observed and painting an unrealistic picture of the system. While two performance measures performed generally well, the other three measures performed so poorly that it is clear that the system does not operate as a M/M/3 exponentially distributed model. Additionally, the initial analysis of the data supports the results from the simulation as the chi-square tests predicted that the exponential distribution was not a good fit and that other models would perform much better

Note: to view the source code for the M/M/3 simulation model, please see the attached MM3.ipynb file.

6 M2 - Best Fit Model (Samuel)

As a group we found that the best fit distribution for the data we collected for both the service and arrival times happened to be the Normal Distribution. To further prove this, we simulated the times spent waiting in queue (W), the average number in systems (L), and the proportion of time servers were busy (B). We then compared these results to the data we collected and saw that the results from the simulation model based on the Normal Distribution was closer to the numbers from the data and therefore was the best fit model out of the M/M/S and Empirical Models. These results were gathered from N=169 over 50 repititions. Below are the results:

```
   ==========================================
Estimate of W: 266.9239902563536
Conf int of W: (224.38298832969568, 309.4649921830115)
Estimate of L: 5.293930473825759
Conf int of L: (4.402479867137253, 6.185381080514264)
Estimate of B: 0.9356656917244526
Conf int of B: (0.9221848401769442, 0.9491465432719609)
Estimate of LambdaEffective: 0.019472179777797614
Conf int of LambdaEffective: (0.019077285654000135, 0.019867073901595093)
============================================
```

7 Empirical Model (Jack)

To simulate an Empirical Model we used the provided draw empirical method which takes a data-set and a random number and returns a random variate from the empirical CDF based on the data. Using this method a SimPy simulation was created, using draw empirical(inter-arrival times, random.random()) to simulate arrival times; draw empirical(service times, random.random()) to simulate service times. The following metrics were gathered:

```
c = 3
Estimate of W: 373.85
Conf int of W: (327.16, 420.54)

   Estimate of L: 7.79
Conf int of L: (6.81, 8.77)

   Estimate of S: 140.82
Conf int of S: (138.33, 143.32)

   Estimate of B: 0.99
Conf int of B: (0.98, 0.99)
```

These results were gathered from N=169 over 50 repetitions. We found that the cost of having a higher N (more customers) in this model was extreme, taking minutes with higher values. As well the more customers simulated the more skewed the metric estimations become. Overall this model did not seem to simulate our data well (the estimations only seem reasonable because of the low N).

8 Conclusion (Kyle)

After all the analysis of the data we observed, we came to find that the supermarket checkout system that we observed followed a normal distribution, both for the inter-arrival and service times of customers. The decding factor for this conclusion was the SimPy simulations in which the normal distribution model performed by far the best, and most consistent across all the performance measures.
Looking back at the histograms of our data, and considering some basic domain knowledge, this is what you would expect for a supermarket. In our plots it seems that we are seeing the tail end of a normal distribution due to the fact that we observed our data from the tail end of the peak hours for customers. It is likely that had we observed the system for the entire day we would have seen a complete normal distribution graph, with each tail of the distribution spanning within the morning hours and the late afternoon hours, with the peak starting at noon.

If we had to advise a change to be made to the system to help improve its efficiency in serving customers, it would be to shuffle the number of checkouts operating in accordance with the peak times. By redirecting checkout staff from the quite hours into the peak hours of early afternoon, it is likely that we would see the system perform in a steady, flat state for the majority of the time, rather than what we observed with peaks and dips.