A Study of the Bluebridge Cook Strait Ferry Check-in System

**Group 4**

Katie Milne

Jaymesh Master

Daniel Braithwaite

Tim Williams

May, 2016

Victoria University of Wellington

Introduction

This project aimed to build a stochastic model of the process of customers arriving and checking in to the Bluebridge Cook Strait Ferry terminal in Wellington, New Zealand. The system consisted of three check-in counters each operated by one person, and a queuing area in front of the servers. The servers followed a first come first served policy. There were three departures from Wellington per day - morning, midday and evening.

Contents

1. Data collection – Tim Williams
2. Data fitting – Jaymesh Master
3. Data models – Daniel Braithwaite
4. Comparing models to observations- Katie Milne

**1.-Data collection methods**

To record our data, we used a python program Monitor (supplied). Monitor is a simple program useful for observing and recording events in sequence, making note of the time of each event. The program outputs a text file with a line for each event, giving the time of the event and the description given to the event by the user.

In this case we described events using the keywords ‘s1’, ‘s2’, ‘s3’ for a service starting at server 1, 2, or 3, respectively, ‘q’ for a customer joining the queue, ‘e1’ ,’e2’, ‘e3’ for server 1, 2, or 3 becoming empty (a service finishing and the queue empty). Prefixes ‘c’ and ‘o’ were also used to indicate servers closing or opening. Mistakes were donated by an ‘x’ and manually deleted from the file later.

Each booking was counted as a ‘customer’ rather than each passenger. Bookings were most often for one or two passengers with the occasional family or school group.

Some challenges we faced with the data collection arose from the nature of the system. People travelling together did not always arrive together, making it difficult to be sure who was joining the queue and who was effectively already in the queue. Often a customer would not be aware of the correct procedure and go to the counter before filling out luggage tags or just to ask a question. This results in multiple services for a single arrival. Another difficulty was in dealing with servers opening and closing. From time to time a server operator would be required to deal with another task such as baggage handling or dealing with a large group, leaving the counter unattended. In these cases, it was unclear whether the counter was open or closed, as the attendant was still working on some service.

Reading the data

A python program Data\_Parser.py (see a walkthrough of the script in Appendix A.) was written to read data in the format output by Monitor.py. It performs several functions on the file and provides information about the data collection session, including service times, queue times, arrival times, inter-arrival times, number of services performed by each server. We can get the time taken for a service by subtracting one service start time from the previous service start time when a queue exists. When the queue is empty we use the empty server keyword as the end time for the service. The time spent in the queue is obtained by keeping track of the number in the queue and subtracting a customer’s service start time from the time they arrived in the queue.

Example output from one data collection session

Number of services by s1: 25  
Number of services by s2: 23  
Number of services by s3: 30  
  
Total services = 78  
Total number queued = 47

Duration of session: 42.41 minutes  
mean time spent in queue: 68.68 seconds

s1   
*list of service times*  
Mean service time for server: 79.0  
s2  
*list of service times*Mean service time for server: 83.74  
s3  
*list of service times*Mean service time for server: 66.64

*list of interarrival times*  
Mean interarrival time: 32.5985454545

Table 1a below displays the data gathered from all sessions. Without collecting more data, it is hard to make accurate observations, but it appears that early in the weekend is the best time to get larger collections. We gathered 60% of our data from three sessions on Friday morning and Saturday mornings. These busier sessions all have a shorter average inter-arrival time than the weekday sessions we observed. Average service times are fairly similar across all observed sessions.

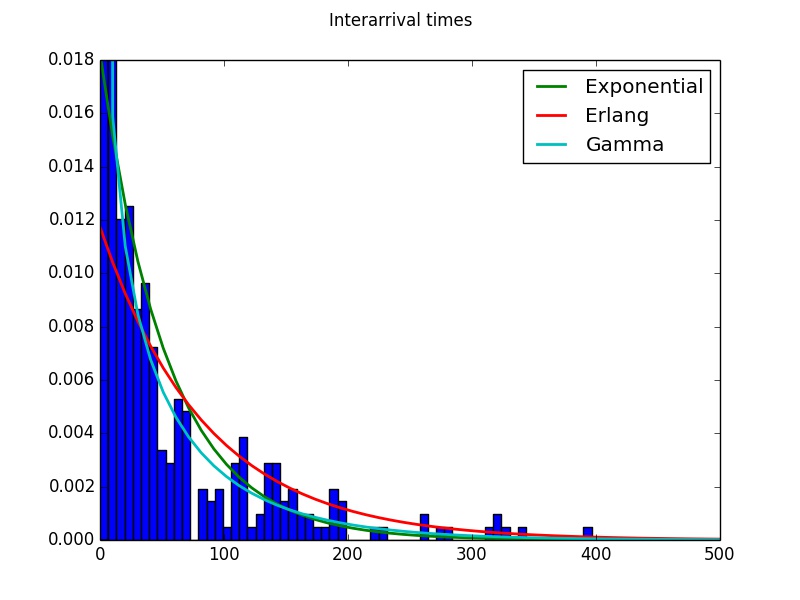
|  |  |  |  |
| --- | --- | --- | --- |
| Session time | No. customers | Av. service | Av. inter-arrival |
| Wed- Evening | 21 | 75.75 | 52.74 |
| Wed- Morning | 28 | 51.43 | 89.27 |
| Sat- Midday | 36 | 104.47 | 80.31 |
| Tues- Midday | 42 | 78.66 | 68.25 |
| Fri- Morning | 55 | 82.32 | 41.64 |
| Sat- Morning | 60 | 91.88 | 57.18 |
| Fri- Midday | 78 | 75.65 | 32.6 |
| Total | 320 | 81.36 | 55.09 |

**Table 1a.** Comparison of observation sessions at various times. Number of customers was varied and appear to be associated with interarrival times.

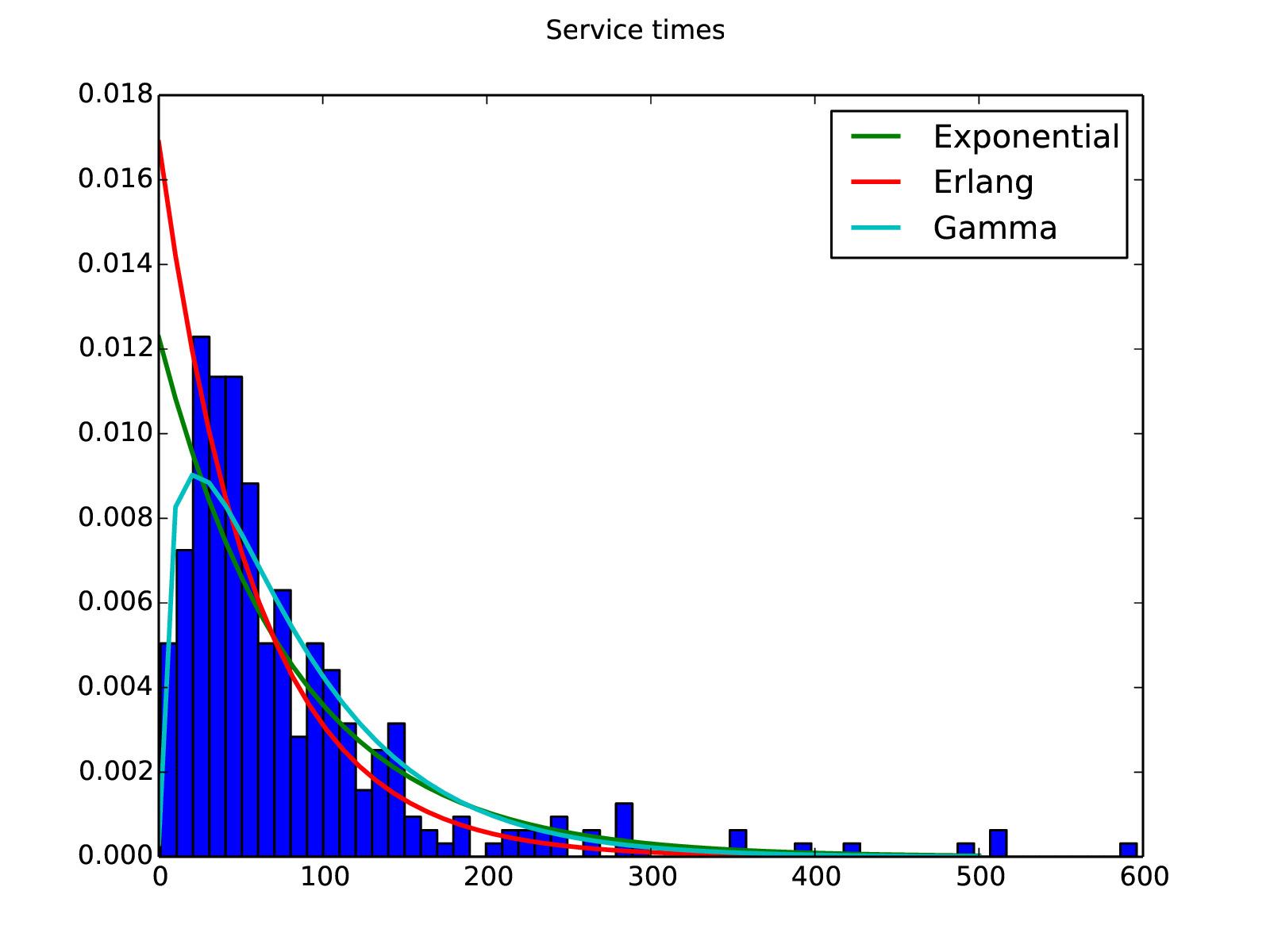
**2. Data fitting**

The data collected at Bluebridge Cook Strait Ferry Terminal for interarrival times and service times were modelled by three distributions: Exponential, Erlang and Gamma. In order to successfully fit each distribution, Python programming language was utilised as the primary resource to perform a chi squared goodness of fit test.

|  |
| --- |
| **Parameter Hypothesis:** |
| H0 : The data is consistent with a specified distribution.  Ha : The data is not consistent with a specified distribution. |
|
| A higher p-value/test statistic does not reject H0 and indicates a stronger fit to allow selection of the test with the highest p-value/test statistic |

  
***Figure 2a****: Interarrival time data fitting to compare Exponential, Erlang and ,Gamma distributions. It is observed that Erlang shows a uniqueness as it intercepts around 0.012 mins and decreases significantly while exponential dips more. Gamma decreases steadily and identifies to fit closer.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | | | **chi-squared test** | **p-value** |
| Exponential | loc, beta | 0, 54.7985931569 | 58.743715 | 3.8x10 -8 |
| Erlang | f, loc, beta | 1, 0, 85.1986853693 | 122.780038 | 1.6 x10 -18 |
| Gamma | alpha, loc, beta | 0.643186078892, 0, 85.1986853693 | 23.260845 | 0.025591 |



***Figure 2b****: Service Time data fitting to compare Exponential, Erlang and ,Gamma distributions. Erlang intercepts around 0.017 and decreases steadily while gamma appeared to reveal a stronger fit.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | | | **chi-squared test** | **p-value** |
| Exponential | loc, beta | 0, 81.4455727819 | 48.116649 | 0.000003 |
| Erlang | f, loc, beta | 1, 0, 59.2293360366 | 77.665087 | 1.4x10 -13 |
| Gamma | alpha, loc, beta | 1.37508883625,0, 59.2293360366 | 33.630288 | 0.000213 |

In summary, after performing each test on all three distributions using chi squared testing, it can be concluded that the exponential distribution in both interarrival times and service times has a lower p-value which means that we can not reject the null hypothesis. Erlang was also tested using chi squared testing and was observed to have a lower p value in which meant we again could not reject the null hypothesis. Interestingly, Gamma distribution resulted in having a higher p value in both cases (p-value(Service times) = 0.000213 and p-value(Interarrival times) = 0.025591) and we therefore can reject the null hypothesis and thus indicating a stronger fit. Each of these experiments tested at 95% confidence interval.

Applying Pollaczek-Khinchin (PK) formula

The PK formula can derive steady-state performance measures for M/G/1/FCFS systems. To apply PK we must approximate that arrival rate follows an exponential distribution (found through data fitting) and find the mean and variance of the service times (service times can follow any general distribution). To compare this with the other models, they must be run with a single server as PK only applies to a single server system.

E(S) = 81.44 Var(S) = 7142.83 E(S^2) = Var(S) + (E(S))^2 = 13775.30 Lambda = 1/54.8

Rho = lambda\*E(S) = **1.49**- As Rho was found to be greater than zero, steady state did not exist with just one server and we could not apply PK.

**3. Data models**

Performance measures were obtained through running 50 random replications of the model to generate a point estimate and a 95% confidence interval for each measure. Performance measures were produced for both 3 servers and 2 servers as the system was observed to switch between 2 and 3 servers over the session. As sessions generally had around 60 people and lasted for under 2 hours- performance measures were generated under these conditions, N= 60, maxTime = 7200 (Table 3a) as well as for N = 10000, maxTime = 2000000 (Table 3b). The shorter simulation was expected to be affected by the ‘warm up’ period whereas the longer simulation would represent performance measures at steady state.

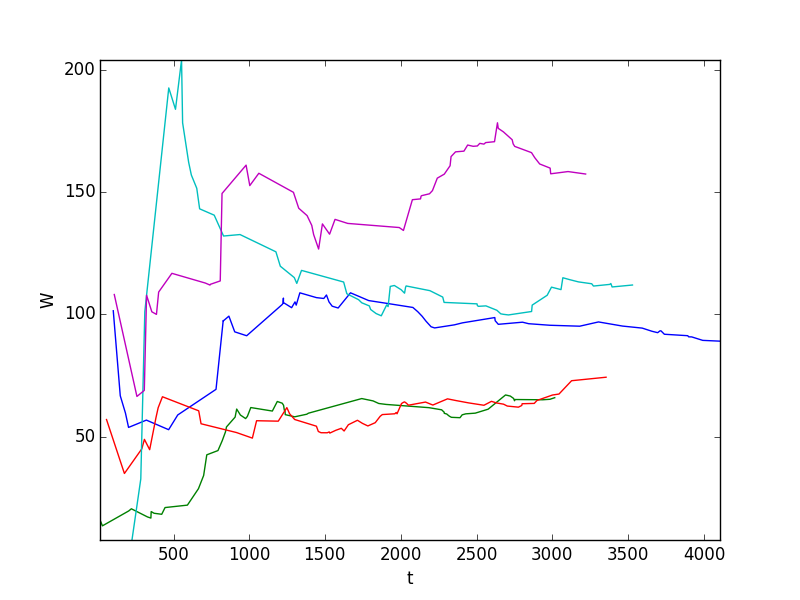
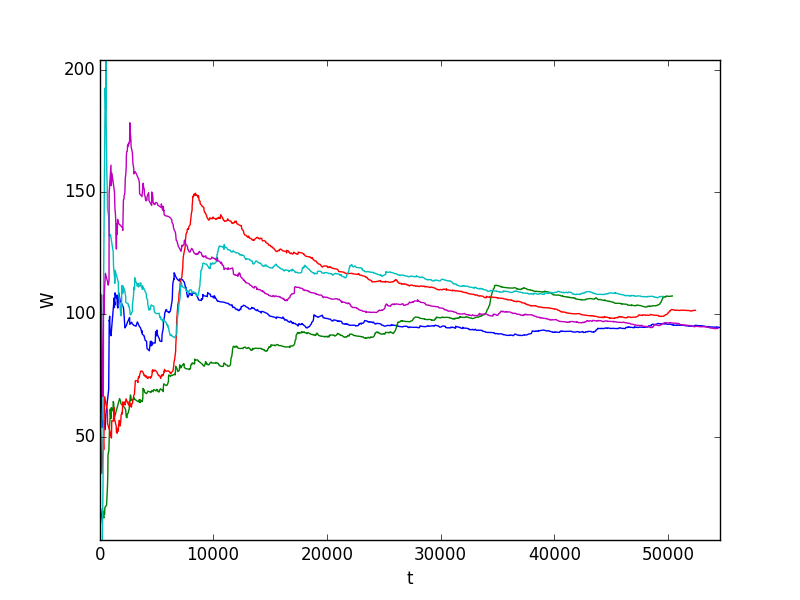
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 50 replications  N = 60  maxTime = 7200 | | | | | | |
|  | **M/M/3** | **M/M/2** | **G/G/3** | **G/G/2** | **Empirical (3)** | **Empirical (2)** |
| **W** | 90.84 *(86.10,95.60)* | 142.46  *(126.6,158.4)* | 100.43  *(94.8,106.78)* | 165.69  *(147.8,183.6)* | 99.80  *(92.7, 106.9)* | 176.19  (156.8, 195.6) |
| **L** | 1.62  *(1.53,1.72)* | 2.54  *(2.23,2.85)* | 1.84  *(1.69,1.98)* | 3.02  *(2.61,3.43)* | 1.95  *(1.79, 2.12)* | 3.37  (2.93, 3.80) |
| **WQ** | 9.14  *(6.55,11.73)* | 61.34  *(47.18,75.50)* | 19.59  *(15.00,24.18)* | 85.33  *(68.53,102.13)* | 22.17  (16.22, 28.11) | 87.47  (67.76, 107.19 |
| **LQ** | 0.17  *(0.12,0.21)* | 1.11  *(0.85,1.38)* | 0.37  (0.28,0.46) | 1.59  *(1.23,1.95)* | 0.45  (16.22, 28.11) | 1.68  (1.28, 2.09) |
| **U** | 0.79  *(0.77,0.81)* | 0.84  *(0.82,0.87)* | 0.49  *(0.47,0.51)* | 0.71  *(0.68, 0.75)* | 0.5  *(0.48, 0.53)* | 0.73  *(0.71, 0.76)* |

*Table 3a: performance measures for shorter simulation*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 50 replications  N = 10000  maxTime = 2000000 | | | | | | |
|  | **M/M/3** | **M/M/2** | **G/G/3** | **G/G/2** | **Empirical (3)** | **Empirical (2)** |
| **W** | 93.53  *(92.30,94.78)* | 186.10  *(176.8,195.4)* | 100.94  *(99.4,102.5)* | 200.15  *(191.5,208.8)* | 99.77  *(97.7, 101.9)* | 217.30  *(203.6, 231.0)* |
| **L** | 1.70  *(1.67,1.73)* | 3.41  *(3.22,3.60)* | 1.84  *(1.80,1.89)* | 3.64  *(3.45,3.83)* | 1.95  *(1.88, 2.01)* | 4.24  *(3.93, 4.55)* |
| **WQ** | 12.37  *(11.50,13.24)* | 104.49  *(95.52,113.5)* | 19.27  *(18.05,20.49)* | 118.83  *(110.4,127.3)* |  |  |
| **LQ** | 0.23  *(0.21,0.24)* | 1.92  *(1.74,2.10)* | 0.35t  *(0.33,0.38)* | 2.17  (1.99,2.34) |  |  |
| **U** | 0.78  *(0.78,0.79)* | 0.85  *(0.84,0.86)* | 0.50  *(0.49,0.50)* | 0.74  (0.73, 0.75) | 0.5  *(0.49, 0.51)* | 0.75  *(0.74, 0.76)* |

*Table 3b: performance measures for longer simulation*

(Left image is warm up period for (N=10000, maxTime=2000000) and Right for the other, both for empirical simulation)



*Figures 3a and 3b: Warm up periods*

*These figures show the warm up period for the empirical distribution. Left figure shows the longer simulation and right figure shows the shorter. Each line on the graph represents a different replication of the simulation.*

As the two graphs demonstrate the shorter simulation with less people is significantly affected by the ‘warm up’ of the system. So the longer simulation better represents the steady state.

Comparing Tables 3a and 3b revealed that the warm up period had negligible effect on performance measures with 3 servers, whereas with 2 servers the measures were significantly different.

The 95% confidence intervals of performance measures obtained for a 3 server system through the G/G/3 model consistently overlapped with the empirical model results indicating that the G/G/3 model was able to accurately model the system. As the empirical model required a lot of computing power, using the G/G/3 model will be an advantage in future simulations. Were as with the 2 server simulations all the point estimates for the G\G\2 and empirical models are close but not all the 95% confidence intervals overlap, indicating that for a two server system the G\G\2 model doesn't fit as well.

Performance measures obtained through the M/M/c model were close to the empirical results however 95% confidence intervals did not always overlap.

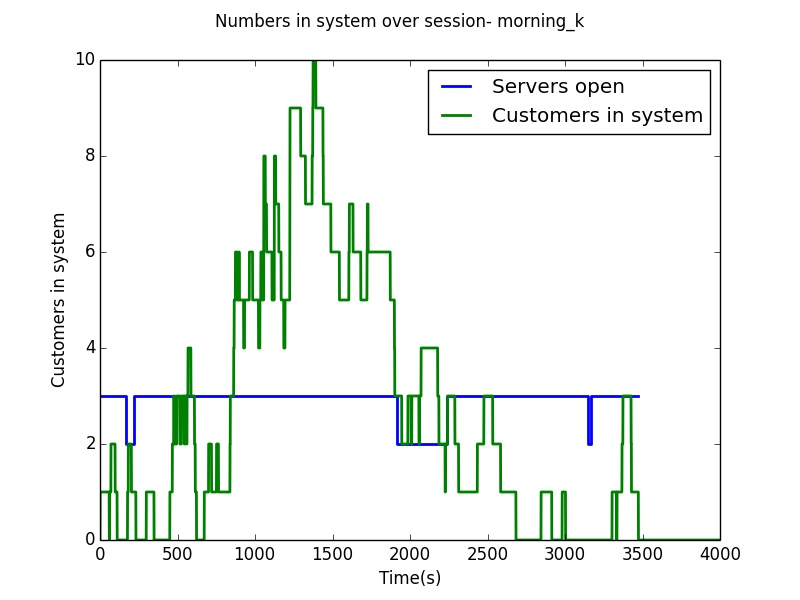
Reflection

Observing the results from Table 3b we see that the utilization is quite low for the 3 server system (around 0.5). Whereas for the 2 server system the utilization is much higher (around 0.74). For the following reasons we would conclude that a 2 server system is the better choice.

* The service unit is a check in service so it is unlikely that balking will occur as people need to collect their tickets to board.
* Less servers will save the business the cost of employing another worker or alternatively allow an employee to do something more useful.

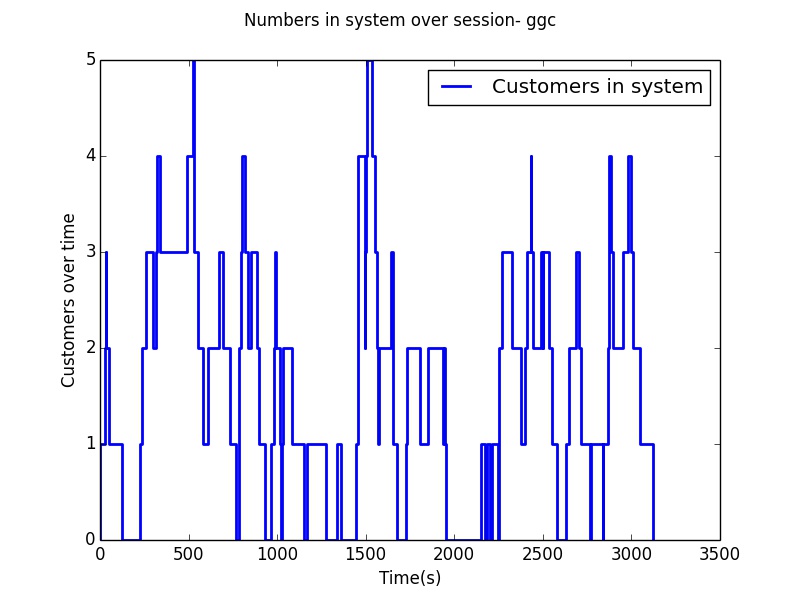
**4. Comparing data models to observations**

The number of customers in the system over time was plotted for a simulation of 60 customers using the G/G/3 model (Figure 4b) and compared with an observed session which also had 60 customers (Figure 4a).



*Figure 4a: An observed session*

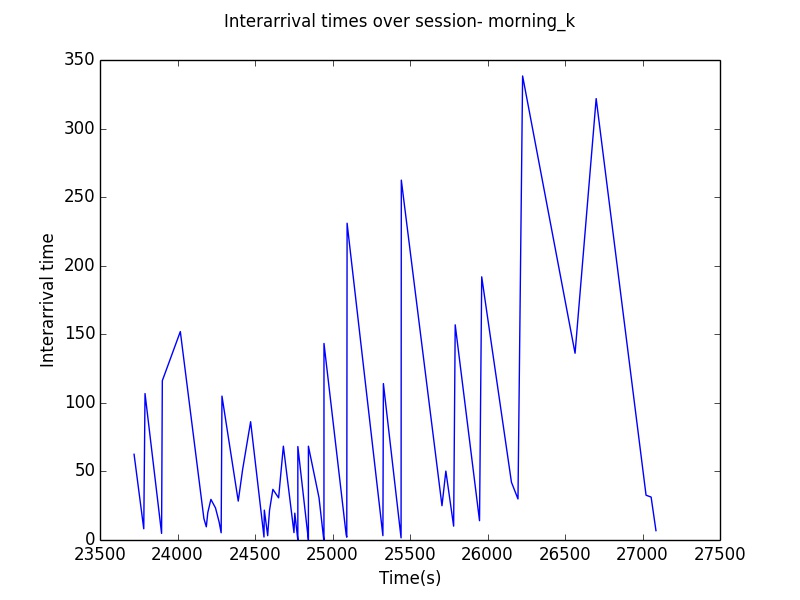
*This figure shows the number of customers in green, and the number of servers open in blue for an observed session on a Saturday morning- in this case there were three servers for the majority of the time meaning it could be reasonably approximated that there were 3 servers over the whole session.*



*Figure 4b: G/G/3 simulation*

*This figure shows the number of customers in blue for the G/G/3 simulation model created in the Data models section.*

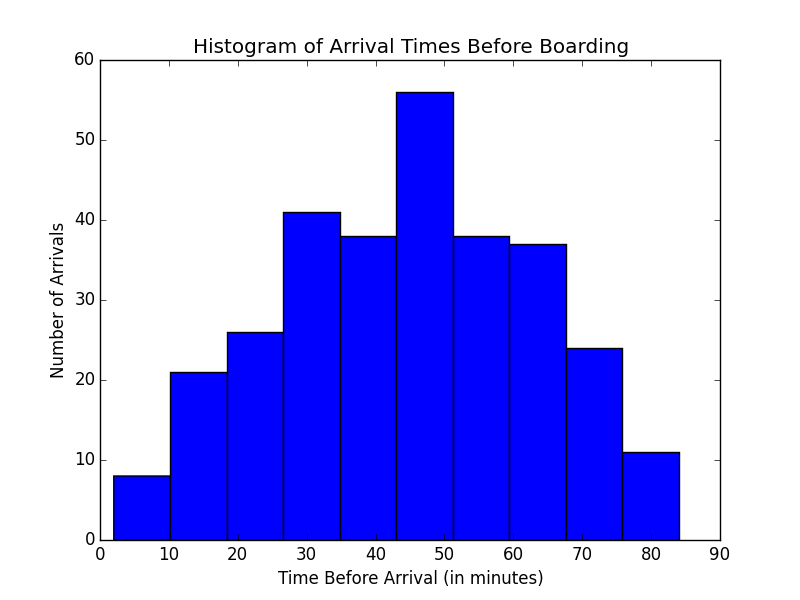
Comparing Figures 4a and 4b revealed that customers were a lot more spread out over the session in the G/G/3 model than in the observed session which suggested that the arrival rate was not constant. Inter-arrival times for the observed session were then plotted over time to investigate this further (Figure 4c).



*Figure 4c: Inter-arrival times over time in observed session*

*This figure shows inter-arrival times plotted over the observed session- the same observed session represented in Figure 4a*

Figure 4c revealed a decrease in inter-arrival times from 24250-25000s followed by a gradual increase confirming that the arrival was not constant (as previously assumed). Considering this it was decided to investigate the distribution of arrivals for all of our observations. This was done through extracting the *arrival time before scheduled departure* for every customer over all observed sessions (Figure 4d). For example, for a ferry crossing departing at 7am, a customer may like to arrive 30 minutes before departure.



*Figure 4d: Arrival times before departure*

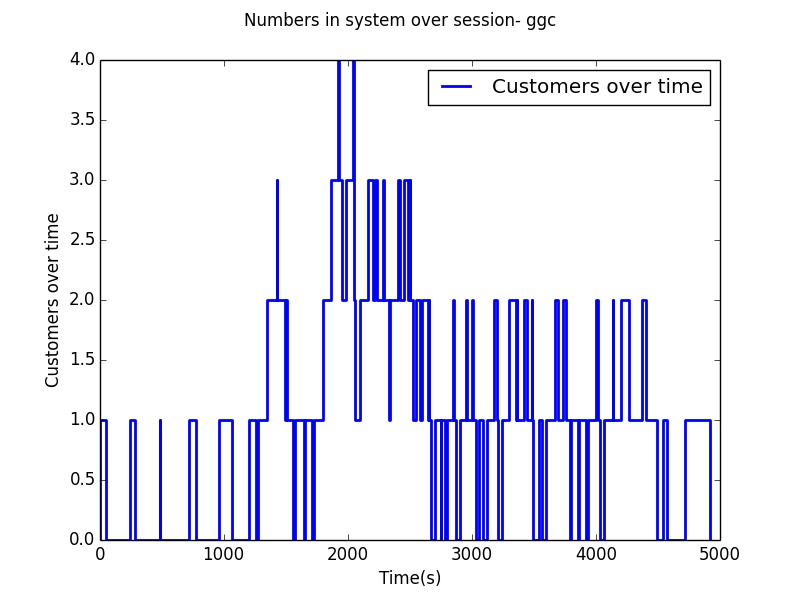
*This figure shows a histogram of the number of customers who arrived within certain time periods before departure - for example for the first bin on the left ~8 customers arrived between 0-10 minutes before departure*

Figure 4d revealed that most people arrived around 45 minutes before departure (the arrival time recommended by Bluebridge) and other arrivals were distributed about this. This made sense for a check-in service when you are arriving for a pre-scheduled event, such as a ferry crossing or flight. Other studies involving check-in services were investigated. Instead of finding the arrival rate, many studies found the proportion of customers that arrived within certain time intervals leading up to departure(Joustra & Van Dijk, 2011;Araujo & Repolho, 2015). This was then done for the Bluebridge service (Table 4a).

|  |  |
| --- | --- |
| **Time before departure (minutes)** | **Proportion of customers** |
| 90-70 | 8.7 |
| 70-60 | 13.7 |
| 60-50 | 16.7 |
| 50-40 | 19.3 |
| 40-30 | 15.7 |
| 30-20 | 14.3 |
| 20-0 | 11.6 |

*Table 4a: Proportion of customers arriving within time intervals*

A simplified model was created in which 60 customers were allocated to time intervals based on the proportions in Table 4a above. Arrival times were then determined by evenly distributing customers across their associated time interval (a deterministic approach). Service times remained in the gamma distribution used in the G/G/c model. This model could be described as D/G/3. Figure 4e shows the number of customers in the system over the session under this model.



*Figure 4e: Proportions model- number of customers in system*

*This figure shows the number of customers in the system over time for the proportions model (D/G/3).*

While the shape of the model was closer to the observed sessions (clear peak), this model was limited in that it was not able to simulate arrivals stochastically. To do this, the arrival rate and distribution would need to be found as a function of time (a non-stationary Poisson Process). An advantage of using proportions is that the same model can be used for different numbers of customers. As the number of customers will often be known by the business prior to check-in, this can be used to allocate the appropriate number of staff/servers (assuming the proportions aren’t dependent on customer numbers).

Another interesting factor in our model was the process behind opening and closing servers. As the arrival rate changed it made sense that when arrival rates are lower, less servers were required. It was observed at Bluebridge that the number of open servers often decreased after or before the peak arrival time however this was not always consistent over the different sessions and there were many periods in which an open server was idle. The number of servers open at each time period is something that could be optimised through the use of an accurate model and would be an interesting avenue to explore in future.

Reflection

As the arrival rate was not constant the system was more complex than our initial models. This meant that it was difficult to create an accurate model of the system from which to draw any useful conclusions and recommendations. If we had more time it would be interesting to construct a stochastic model that could model a non-stationary Poisson process.

**Recommendations and Conclusion**

Modelling recommendations:

* **Build models around observations made at a particular time of day and day of the week**

It was clear from comparing the number of customers, interarrival times, and service times, that there was a large amount of variation between sessions (Table 1a). At least part of this variation is likely due to the time of day, day of the week, and perhaps time of year. For example, people are likely less inclined to arrive early to a 7am sailing compared with a midday or evening sailing, and customer numbers are likely higher on weekends and during holiday periods.

* **Build a stochastic model based on a changing arrival rate (non-stationary Poisson process)**

An accurate model could allow you to investigate several interesting areas including: optimising the number of open servers over each time period, investigating the addition of self check-in stations or online check-in, and adjusting the recommended customer arrival time. Building a model based on proportions would also allow the model to be applied for different numbers of customers.

Business Recommendation:

* **Consider improvements to service time through online check-in options for customers and the possibility of self-service stations**

In an interview with Bluebridge staff (Personal communication, May 23rd, 2016) the efficiency of service was highlighted as a key limitation in this type of system. As an improvement on the data collection, it can be suggested for future experimentation that the different processes and other key issues during check-in are further observed and collected; tasks performed at each counter include printing boarding pass (and checking in) along with dropping baggage off (the baggage drop off are manned and the agents print a tag and attach it to the bags). Many other check-in systems, such as the Wellington Airport check-in for various airlines, now utilise online check-in and self-service stations to increase service efficiency. Bluebridge has neither of these and may benefit from one or both of these technologies.

Conclusion

In conclusion, while this investigation revealed several interesting features of the Bluebridge check-in system, the number of observations as well as limited time and expertise meant that our models were not able to accurately reflect the system. We did, however, gain valuable understanding of the complications and difficulties involved in building a stochastic model for a check-in system which will be useful in future projects.

**References**

Joustra, P., Van Dijk, N. (2011). Simulation of check-in at airports. Proceedings of the 2001 Winter Simulation Conference, 1023 - 1028.

Araujo, Gerson E., & Repolho, Hugo M.. (2015). Optimizing the Airport Check-In Counter Allocation Problem. *Journal of Transport Literature*, *9*(4), 15-19

**Appendix A:**

Data\_Parser.py walkthrough

**Functions**

**split\_data\_into\_individual \_servers(services)**

*Takes the list of all events and splits populates the server lists*

for every string (line) in services

if current event if a service start or service empty

append line to the server n list of events

**get\_service\_times(services)**

*takes a list of services from an individual server returns a list of service times*

for each item in server event list (services

if current event is not a server empty event

get service start time from current event

get service end time from next event

calculate service time

add to list of service times

return list of service times

**get\_queue\_times(services)**

*build and return a list of times spent in the queue*

-list for queue times

-list for queue service start times

-for each item in server event list (services)

-if current event is a queue join

-record queue start time

-for each event in server even list (inner loop)

-if current event +1 is a service start and is not in queue

service start times list

-record queue exit time

-add current event +1 to queue service start times list

-exit inner loop

calculate time spent in queue

add to list of queue times

return list of queue times

**arrival\_times(all\_events):**

*creates a list of customer arrival times*

-arrival times list

-number in queue

-for each event in all events list

-if event is queue joining

-add queue time to arrival times

-increment number in queue

-else if event is service start

-if queue is not empty

decrement number in queue

-else

add service start time to arrival times

return arrival times list

**print\_service\_times(service\_times, server\_number)**

*prints out the list of service times from a given server*

**Program structure**

open a data file using a file opener dialog

read data file and add all data lines into a list holding all events

*split\_data\_into\_individual\_servers(all events)*

Calculate total time of observation session

count number of services in each server

*get\_service\_times(*each server event list)

*get\_queue\_times(*all events list)

*arrival\_times(*all events list*)*

calculate interarrival times from list of arrivals