DATA304/474 and COMP312 Final Project

Group 1

09 June, 2023

Author	E-mail	ORCID
Duncan Bennie	benniedunc@myvuw.ac.nz	0009-0008-5570-5397
Will Doherty	dohertwill@myvuw.ac.nz	0009-0000-6177-370X
Leo Gaynor	gaynorleo@myvuw.ac.nz	0009-0005-8472-4907
Sharna Granwal	granwashar@myvuw.ac.nz	0000-0003-1010-3284
Samuel Non	nonsamu@myvuw.ac.nz	0009-0003-6581-2704
Taichi Taniguchi (John)	tanigutaic@myvuw.ac.nz	0009-0008-0634-5399

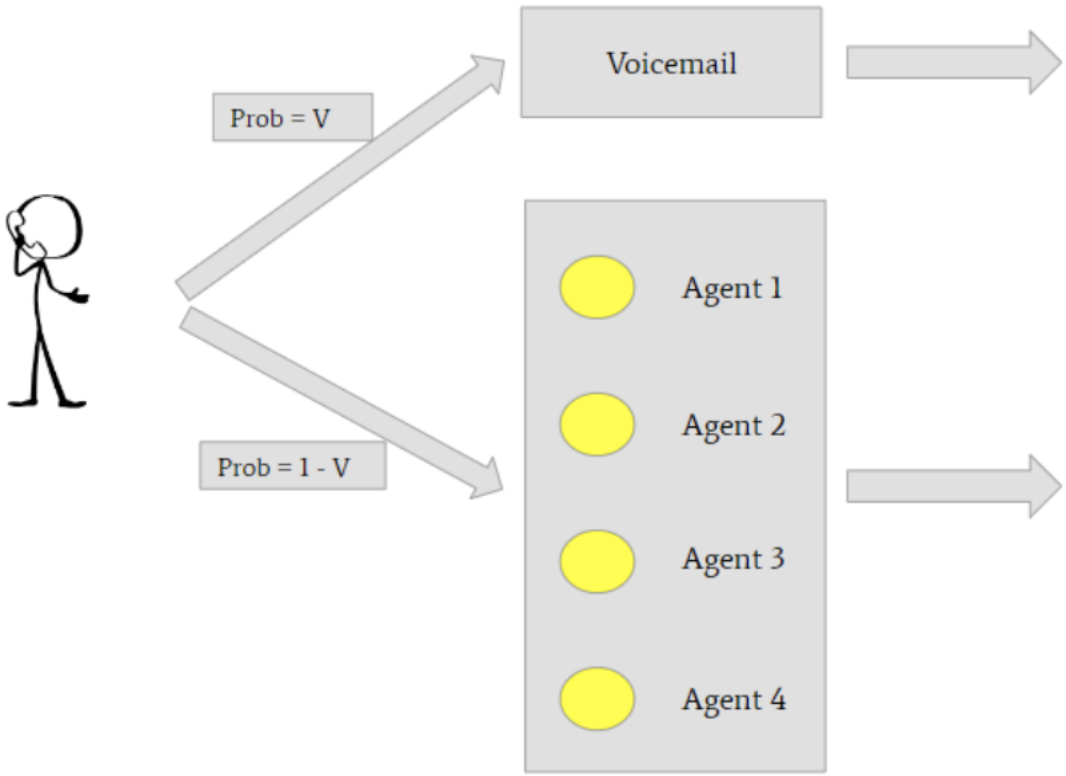
Project Tasks	Team Member(s)
Project Planning	Everyone
Data collection and processing of historic data	Leo, everyone
EDA, model fitting and GOF tests	Everyone
EDA Report (Report 3)	Leo
Simulation - Model 1 (M/M/4)	Samuel, John
Simulation - Model 2 (M/M/4)	Duncan, Leo
Simulation - Model 3 (Empirical)	Will, Sharna
Simulation Report (Report 4)	Sharna

Final Report and Presentation	Team Member
Background	Will
Data Collection and Processing	Leo
EDA - Model Fitting	John
EDA - GOF Tests	Samuel
Simulation Results	Sharna
Discussion	Duncan
Git Repository[1]	Will
Final Report Collation	Leo

[1] https://github.com/willdoherty/DATA474-group-1-phone-queueing-system

Introduction

Our goal was to model a real world phone queueing system. The operator of this system wished to remain anonymous and so will not be referenced in this report. The motivation for this study was to see if the system could handle a large increase in the number of callers due to some external event. In particular, we were interested in what would happen if the arrival rate doubled from the empirically observed arrival rate.
The diagram below gives a basic overview of the system.



- Customers call into the system, usually to book an appointment or ask a question.
- They can choose to either leave a message (with probability V) or speak to an agent (with probability $1 - V$).
- If they choose to leave a message, they are sent straight to voicemail and do not have to queue.
- If they choose to speak to an agent, they enter a single queue.
- There are multiple agents who pick up calls. This number varied slightly over time, but was about four on average.
- Once callers have recorded their voicemail or spoken to an agent, they leave the system.

Methods

Data Collection

The data used in this project is a combination of manual observations, and historical records. After permission to record call activity from the phone system was granted by the Operations Manager of the service, we endeavored to collect at least part of the data manually. This presented several challenges, which are listed below.

- Accuracy: human error in data entry, formatting, missing data points.
- Time Constraints: time consuming for one or more individuals resulting in a high cost to volume ratio.
- Privacy: strict data privacy guidelines and policies made access to the system difficult.
- Interface: the phone system interface was not designed for observation, its primary purpose being to provide information to agents answering calls.

After one test run to see how the system worked, we ended up recording data from two different days. On day one we recorded 43 arrivals over 103 minutes, and on day two we recorded 42 arrivals over 145 minutes, totaling to 85 observations over 248 minutes (4 hours and 8 minutes). The length of recording was determined by when the service could make the system available to us. For the majority of the time, we found there were 4 agents on roster to take calls, this number varied occasionally from 3 to 6 over all the sessions, but is well approximated by 4 agents.
Except for the high time investment required, the most frequent issue that occurred while recording was to do with the interface of the phone system (not included due to the security policy of the observed system). As it is designed for a single agent to accept and handle calls, there was limited access to view the queue of calls. Calls would "bounce" around different agents until they were picked up, making tracking the calls through arrival, service and departure additionally challenging. There was very little visual difference between calls that were queued and calls that were being served, and when calls departed, they just disappeared.
After consulting with the Operations Manager, we were provided with historical data, amounting to 305 observations over 24 hours after cleaning. The historical data was easily available, and was much more accurate than the manually recorded data. It also included records from the phone system's 'Voice Assistant'.

Exploratory Data Analysis

Inter Arrival Times

To calculate the inter arrival times, we had to transform the data first. As the actual start of the recording sessions were not measured, we had to set the first index to an interval of zero seconds to emulate a start time. The data was also not stored in separate sessions, so using the absolute inter arrival times, we could identify large jumps (of an hour or more) that we could split on to separate the data into the sessions.

From there we were able to fit the inter arrival times to both gamma and exponential distributions using the `fitdistr` function from the `MASS` package in R.

A histogram with Kernal Density Estimate, estimated Exponential Distribution and estimated Gamma Distribution was then generated using `ggplot2`.

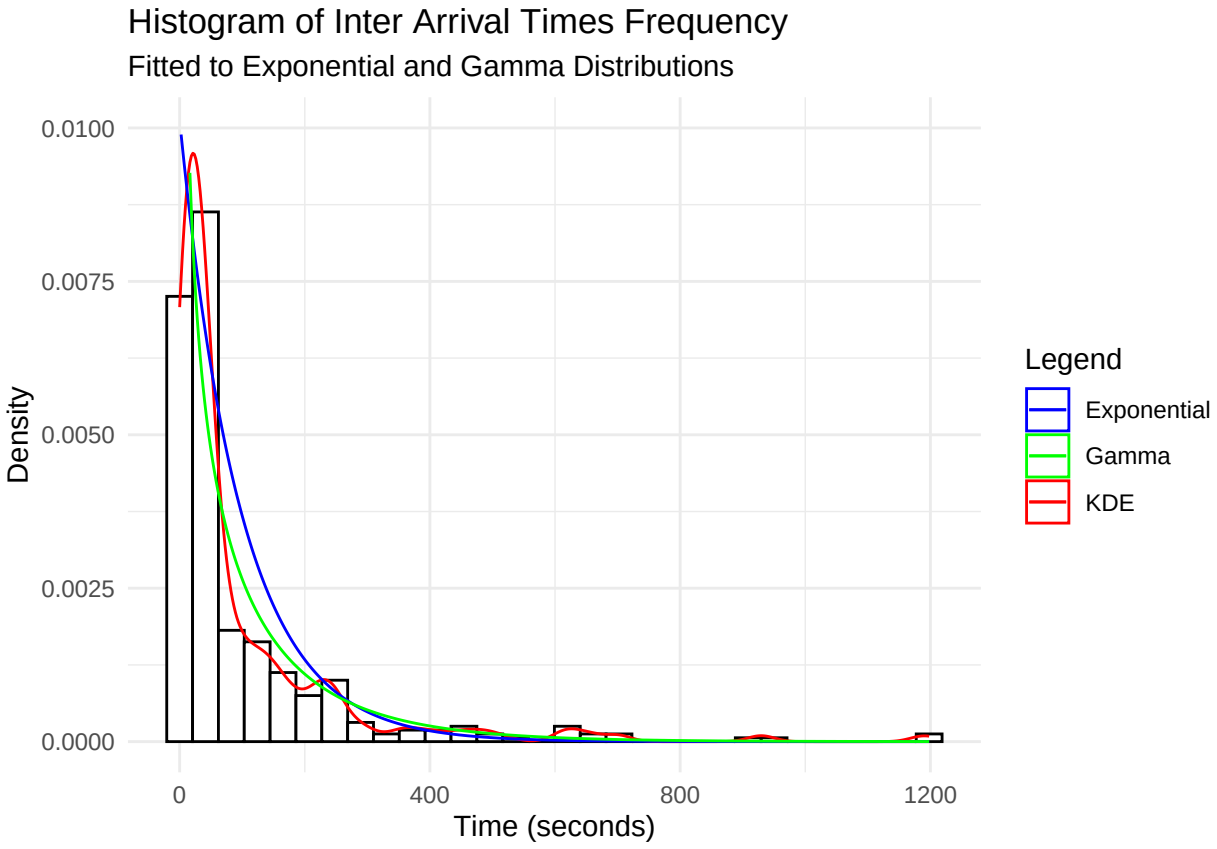


Figure 1

Goodness of Fit Test

Below are the Hypotheses for each Chi-Squared GOF test, with the specified distribution referring to either exponential or gamma:

H_0 : The observed data follow the specified distribution.

H_1 : The observed data do not follow the specified distribution.

In the goodness of fit tests below, we chose to divide the data into a specific number of bins such that the expected number of observations for the estimated distribution was at least 5. This is a rule of thumb for contingency tables evaluated by the chi-squared GOF test (Cochran's Rule). The results of this transformation are below:

Table 4: Exponential Counts

	Observed	Expected
(0,63.1]	256	179

	Observed	Expected
(63.1,126]	40	98
(126,189]	31	63
(189,252]	25	17
(252,315]	8	17
(315,378]	4	7
(378,1.2e+03]	23	6

Table 5: Gamma Counts

	Observed	Expected
(0,63.1]	256	237
(63.1,126]	40	61
(126,189]	31	36
(189,252]	25	25
(252,315]	8	5
(315,378]	4	7
(378,441]	4	6
(441,504]	6	5
(504,1.2e+03]	13	5

Service Times

The Service times included several invalid observations that had to be removed, for example calls that went though to the 'Virtual Assistant' did not have a recorded service time.

The density estimates and plotted distributions were done the same way as the inter arrival times. The estimated density for the gamma distribution is again the better fit visually for the majority of the data, but in both this plot and the previous one there is low-alignment in the middle of the range. This is indicative of the density under fitting the longer time period data.

Table 6: Exponential Parameter Estimate

Rate
0.007879

Table 7: Gamma Parameter Estimate

Shape	Rate
1.387	0.01093

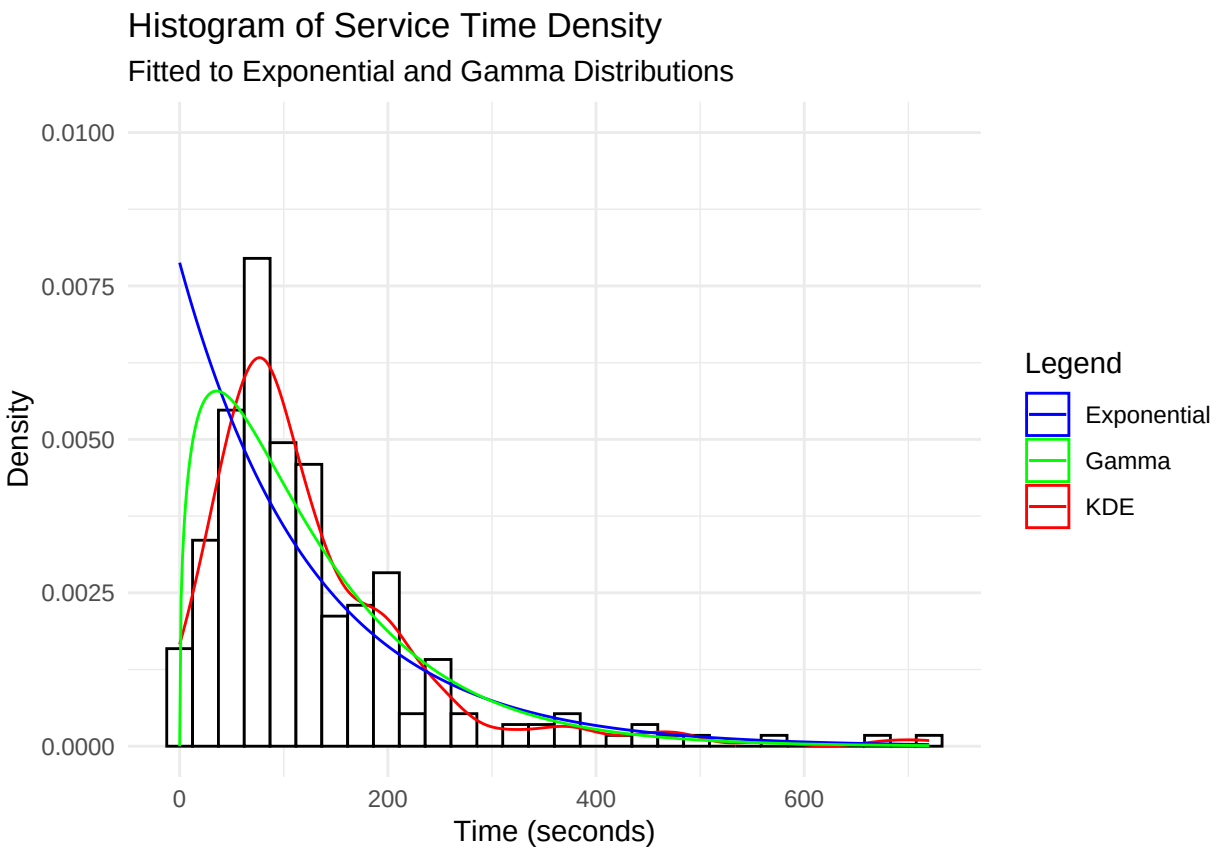


Figure 2

Goodness of Fit Test

We will be using the same Hypotheses for the Service Time Chi-Squared GOF test:

H_0 : The observed data follow the spedified distribution.
H_1 : The observed data do not follow the spedified distribution.

Below we have repeated the Chi-Squared GOF test transformation for the Service Time data.

Table 8: Exponential Counts

	Observed	Expected
(0,37.9]	28	55
(37.9,75.8]	48	45
(75.8,114]	60	22
(114,152]	30	29
(152,189]	18	24
(189,227]	18	22
(227,265]	9	8

	Observed	Expected
(265,720]	17	22

Table 9: Gamma Counts

	Observed	Expected
(0,37.9]	28	39
(37.9,75.8]	48	58
(75.8,114]	60	42
(114,152]	30	30
(152,189]	18	19
(189,227]	18	18
(227,720]	26	22

Simulated Models

Models

The service's phone queueing system was simulated using three models:

- M/M/4, using the estimated parameters from our investigation.
- G/G/4, using the best-fit distribution and estimated parameters from our investigation. This was the gamma distribution for both the interarrival and service times.
- Empirical model, using samples taken from the empirical distribution of the interarrival and service times to simulate the service.

Additionally, we test the G/G/4 model with our scenario of doubling the arrival rate of calls.

Assumptions

Each model contained four service units representing four agents taking calls. Although the number of agents fluctuated throughout the day, we chose a constant number based on the typical number of agents rostered on to simplify the models.

Each model also includes the variable `va_prob`, representing the proportion of calls that went to the "voice assistant". Callers are greeted with an automated message with a choice of either speaking to an agent, or leaving a message with the voice assistant. The voice assistant can be considered as an infinite server system, so no queuing takes place. If the caller chooses to speak to an agent, however, they are redirected to the queue which is what we were primarily interested in monitoring. The estimated value for `va_prob` was 150/304. This was based on historical data from the phone management system which captures the different types of calls. Our models assume that this proportion is accurate for all days of the week.

Simulation

The code and output for each model and simulation is provided below in the Simulation section. Each simulation is run for a maximum time of 29,700 seconds, which is the maximum amount of time the phone lines are open per day. The random seeds are set to ensure replicability. We captured the following performance measure for each model:

- W: Average time in the system (including arrivals of calls sent to voice assistant)
- L: Average number of callers in the system
- B: Proportion of time the system is busy
- W_s: Average time in system (calls sent to agents only)
- W_q: Average time in the queue (calls sent to agents only)
- λ_{eff}: Effective arrival rate (calls arriving at agents)
- V: Proportion of calls sent to voice assistant

Results

Theoretical Models

Table 10: Inter Arrival Time Chi-Squared GOF Test for Exponential Distribution

Test statistic	df	P value
64.45	6	5.592e-12 * * *

Table 11: Inter Arrival Time Chi-Squared GOF Test for Gamma Distribution

Test statistic	df	P value
11.03	8	0.2001

Table 12: Service Time Chi-Squared GOF Test for Exponential Distribution

Test statistic	df	P value
28.46	7	0.0001813 * * *

Table 13: Service Time Chi-Squared GOF Test for Gamma Distribution

Test statistic	df	P value
6.286	6	0.3919

The inter arrival histograms (*fig.1* and *fig.2*) show that there are periods of high and low activity for arrival times, and interestingly the graph for the service time showed similar results. The gamma distribution fits the observed data better than the exponential distribution in both cases. This is evident in the plotted distribution estimates, where the curve of the gamma distribution more closely matches the observed data. We noticed that there are instances where service and arrival times are significantly longer than the mean. These instances are considered delays and appear as spikes in the histogram during the higher time periods, for example at 400 seconds and 800 seconds there is a bump.

Finally, the chi-square goodness-of-fit test suggests that in both cases the gamma distribution fits the observed data better. The p-value for the gamma distribution is greater than 0.05, indicating that there is insufficient evidence to reject the hypothesis that the observed data follow a gamma distribution. Our analysis suggests that the gamma distribution is a more accurate model for both service and arrival times in this phone system.

Simulated Models

The point estimates and 95% confidence intervals for each performance measure is given in the table below:

	Actual Data	Model 1 (M/M/4)	Model 2 (G/G/4)	Model 3 (Empirical)	Scenario (G/G/4)
W (s)	109	63.685	64.626	66.66	65.766
95% CI		(61.643, 65.727)	(63.097, 66.155)	(65.384, 67.936)	(64.472, 67.059)
L	1.11	0.653	0.664	0.712	1.342
95% CI		(0.631, 0.675)	(0.639, 0.688)	(0.697, 0.726)	(1.305, 1.379)
B	0.2	0.477	0.453	0.441	0.694
95% CI		(0.466, 0.489)	(0.441, 0.464)	(0.435, 0.448)	(0.684, 0.704)
W_s (s)	167	127.074	127.471	132.552	129.713
95% CI		(123.974, 130.174)	(125.25, 129.693)	(132.243, 132.861)	(127.778, 131.648)
W_q (s)	36	0.139	0.252	0.435	1.937
95% CI		(0.044, 0.235)	(0.146, 0.358)	(0.328, 0.541	(1.58, 2.294)
λ_{eff}	0.006	0.005	0.005	0.005	0.01
95% CI		(0.005, 0.005)	(0.005, 0.005)	(0.005, 0.005)	(0.01, 0.011)
V	0.492	0.498	0.492	0.497	0.492
95% CI		(0.49, 0.507)	(0.485, 0.499)	(0.488, 0.506	(0.487, 0.497)

What is of immediate interest is the large difference between W, W_s and W_q. When a caller is passed to the Voice Assistant (with $P(V)$), there is no recorded service time in both the model, and the original data. Since $P(V)$ is close to 0.5 in this system, this has the effect of reducing the average time in system for all call types by approximately half. i.e.

$$W = P(V)W_v + (1 - P(V))W_s$$

where W_s is the average time in system for all queue types, and W_v is the average time in system for calls going to the Voice Assistant. In our simulations, W_v is essentially 0.

Considering only calls redirected to agents, the average time in the system (W_s) from our collected data was 167 seconds, split into an average service time of 131 seconds and an average queue time (W_q) of 36 seconds. Noticeably, the W_q calculated from our data (36 seconds) is significantly higher than the estimates from all three theoretical models, each of which estimated this to be less than half a second. This will be discussed further in the discussion section.

We attribute this to the fact that even though an agent may be available in the system, they may not necessarily be free to answer a call. For example, they might be completing "follow up" tasks from a previous call and there is some delay before they can pick up the next caller. We noticed this behaviour during the data collection stage, but it is currently not captured in any of the models. We suggest this could be modelled in future by adding a fixed amount of time to the service times, or drawing the additional times from a distribution that models this activity.

The effective arrival rate of calls arriving at agents(λ_{eff}) from our collected data, averaged across all days, was 0.006. All of our models returned a relatively close estimate of 0.005. The proportion of time the system is busy with at least one caller (B) estimated by all three models was between 0.4-0.5, which is more than double of that calculated from our collected data (0.2). Despite over-estimating the "busyness" of the system, all three models still suggest the system is perhaps underutilised.

Due to the differences in the metrics between the simulated models and our collected data, we were not confident that any of the simulations modelled the system accurately. Regardless, we tested

the Model 2 with our proposed scenario of doubling the arrival rate to see how the system would cope. The estimated proportion of time that the model was busy was 0.694, meaning that this system could easily manage an influx of calls. This result is inline with our understanding of the system after collecting the data which we believed to be largely underutilised.

Conclusion

In summary, we found that the system modeled by our simulations would be able to handle the increased arrival rate with ease. However, the system they modeled was not the intended system that we were observing. As mentioned in the results for the simulated models, we attribute the large discrepancy in model metrics to delays in agent availability. Even though all elements of the simulation were modelled off the observed data, and the derived distributions both passed goodness of fit tests, we were not confident that any of the simulated models accurately represented the intended system, or could be utilised in good faith to analyse the intended system under different conditions.

Discussion

Several factors complicated the system and made it more difficult to model and simulate accurately. First, the system had a dynamic number of agents (servers) and as such throughout the day the value of c would change. These changes were neither random nor exactly timed. The changing value of c meant that taking a single value (the closest integer to the mean, or the mode) could not accurately model the system, only create a best estimate of the system. The non-random and non-regular changing of the number of agents also meant that these changes could not accurately be simulated because we could not create an algorithm or function that would be able to exactly simulate these changes.

We also noticed that calls would bounce around the queue (i.e. calls would be ringing and not picked up immediately) because answering calls was not the only responsibility of the agents, so they could be busy with another task and not be able to pick up the call. There could be other reasons why the calls would go unanswered, which could generally be categorised as human factors, for example being logged in to the system but away from their workstation. This created 2 problems. The first is that it would reduce the effective number of servers without reducing the recorded number of servers, i.e. there could be 2 servers ready to receive calls but the system would record the number of servers as 4 because 4 accounts were logged in and active in the phone/computer system. The other problem this bouncing created was the existence of wait times while the queue was empty; calls would come in and not immediately be picked up for the aforementioned reasons, and despite there being no queue for the system the customer would have a wait time regardless. This is contradictory to the queueing models used in this course because normally any system with an open server should not have any queue and therefore have a queue wait time of 0. This inconsistency created small deviations between our modelled system and the real system, and as such would have likely impacted on the statistics we gathered while simulating the various models.

The system also had a low utilization rate. This meant that queue forming was rare. While this is good for customer experience, it made studying the system more difficult as queue forming an important part of the statistics that we gather. This meant that minute-for-minute, data recording was less valuable, as we would need to be recording for a longer than normal time for a queue to

form. This made in-person recording less important and the system recorded historical data more important, since this data didn't require us to be present at the time of recording.

Finally, the voice assist system added complexity to the system which made it harder to model. Specifically it created 2 problems. One of which was having a wait time despite the queue being empty and servers being available (The consequences of this is discussed above), this is because customers would listen/interact with the voice assist before being connected to the server. The other is that if the customer chose to go to voice mail, they would leave the system. As such some key metrics were unusual. For example, the average time in the system was lower than the average service time. A system typically has a larger average time in system than average service time because average time in system is the sum of the average time in queue and the average time being served.

Suggestions

Several of the challenges mentioned in the discussion could be avoided by modifying our approach. For example, we could use only the historical data that was recorded by the system. This would be important for several reasons. Firstly it would eliminate any human error in the data gathering process, meaning the data will be more accurate. Secondly, it would include all the data on the voicemail system. This is important because in the live data gathering, we were not able to gather any data on the calls going to voicemail and as such it limited our ability to model this part of the system. Thirdly it would allow us to use gather significantly more data. The system automatically records datapoints and so we would be able to have a much larger dataset. We could collect data across days, weeks or months. This would allow us to see trends depending on the seasons, days, or week. This could provide an interesting insight into the system. For practical reasons in-person recording of this amount of data is not viable (we would need to be in the office recording at all times for months).

Further, we could explore models with or without voice mail. Modelling the current system without the voicemail and having all customers go through an agent would enable us to predict how the system would manage if the voicemail broke. This could also help create a cost/benefit analysis for the current system. Conversely, this could be used to add a voicemail system to a different system which currently does not have one in order to see if it could benefit from a voicemail system being added.

Finally, we could add a "buffer time" in before a free agent picks up a call in the simulations. This would help the simulation better reflect the real-life dynamic of an agent being available but the customer not instantly being served. Doing this would make the simulation a more accurate representation of real life and would help reduce some of the inconsistencies between the real-life metrics and the simulated metrics, for example the average wait time in the simulation would be closer to the average wait time in our recorded data. This could be done by adding a wait time randomly generated from an appropriate distribution and having the server wait this amount of time before beginning service.

Appendix

R Code (EDA and Hypothesis Testing)

```
#Loading Required Packages and defining helper functions:
library(readxl)
library(dplyr)
library(pander)
# Reading in data
data <- read_xlsx("Group 1 data.xlsx", sheet = 3)
n <- length(data$`Arrival time`)

# Calculate the time differences between observation n and n+1
time.dif <- c(0, data$`Arrival time`[2:n] - data$`Arrival time`[1:n-1])
iat <- as.double(time.dif, units='secs')
# Find the bounds, i.e. where the times differ by more than 1 hour
bounds <- which(abs(iat) > 3600)

# Split the data at each bound
s1 <- data[1:(bounds[1]-1),]
s2 <- data[(bounds[1]+1):(bounds[2]-1),]
s3 <- data[bounds[2]:n,]

# Add the inter arrival times to the original data.frame
data$inter.arrival.time <- time.dif

# Create a vector of all valid inter arrival times
iat <- as.double(data$inter.arrival.time)
iat <- iat[abs(iat)<3600]
# Ensure positive values for interarrival time
iat <- abs(iat)
# Avoid divide by zero errors in parameter estimation
iat[iat==0] <- 0.001
# Load Required Packages
library(MASS)
library(ggplot2)

# fit using MLE
fit.exp <- fitdistr(iat, "exponential")
fit.gamma <- fitdistr(iat, "gamma")

pander(pandoc.table(fit.exp$estimate, style="rmarkdown", col.names = c("Rate"), caption="Expone
pander(pandoc.table(fit.gamma$estimate, style="rmarkdown", col.names = c("Shape", "Rate"), capt
pred.density <- density(iat)
x <- seq(0, max(iat), length.out = length(pred.density$y))

fit.exp.density <- dexp(x, fit.exp$estimate) # mean
```

```
fit.gamma.density <- dgamma(x, fit.gamma$estimate[1],
                            fit.gamma$estimate[2]) #shape and rate


cols <- c("KDE"="red","Exponential"="blue","Gamma"="green")

ggplot() +
  geom_histogram(aes(x=iat, y=..density..), color = 'black', fill ='white') +
  geom_density(aes(x=iat, color='KDE')) +
  geom_line(aes(x=x, y=fit.exp.density, color='Exponential')) +
  geom_line(aes(x=x, y=fit.gamma.density, color='Gamma')) +
  scale_color_manual(name="Legend",values=cols) +
  ylim(c(0, 0.01)) +
  labs(title = "Histogram of Inter Arrival Times Frequency",
       subtitle = "Fitted to Exponential and Gamma Distributions")+
  ylab("Density") +
  xlab("Time (seconds)") +
  theme_minimal()
library(EnvStats)
# Sample variates using estimated parameters
set.seed(123)
exp.variates <- rexp(length(iat), rate= fit.exp$estimate)
set.seed(123)
gamma.variates <- rgamma(length(iat), fit.gamma$estimate[1],
                         fit.gamma$estimate[2]) # shape and rate

# Initial split of observations into 20 bins
# Expected counts should be >=5
bins.for.chisq <- seq(0, max(iat), length.out=20)

observed <- table(cut(iat, bins.for.chisq))
expected_exp <- table(cut(exp.variates, bins.for.chisq))
expected_gam <- table(cut(gamma.variates, bins.for.chisq))
# Bins for exp distribution.
bins.for.chisq.exp <- append(bins.for.chisq[1:7], 1198)
observed_exp <- table(cut(iat, bins.for.chisq.exp))
expected_exp <- table(cut(exp.variates, bins.for.chisq.exp))

# Bins for gamma distribution.
bins.for.chisq.gam <- append(bins.for.chisq[1:9], 1198)
observed_gam <- table(cut(iat, bins.for.chisq.gam))
expected_gam <- table(cut(gamma.variates, bins.for.chisq.gam))


# Create contingency tables of observed vs expected
counts_table_exp <- data.frame(observed_exp)
counts_table_exp$Expected_Exp <- expected_exp
```

```r
# reassign row names
bin_names <- counts_table_exp[1]
counts_table_exp <- counts_table_exp[,-1]
rownames(counts_table_exp) <- bin_names[,1]
colnames(counts_table_exp) <- c("Observed_Exp", "Expected_Exp")

# Repeat for gam
counts_table_gam <- data.frame(observed_gam)
counts_table_gam$Expected_Gam <- expected_gam
# reassign row names
bin_names <- counts_table_gam[1]
counts_table_gam <- counts_table_gam[,-1]
rownames(counts_table_gam) <- bin_names[,1]
colnames(counts_table_gam) <- c("Observed_Gam", "Expected_Gam")

pander(pandoc.table(counts_table_exp, style="rmarkdown", col.names = c("Observed", "Expected")
pander(pandoc.table(counts_table_gam, style="rmarkdown", col.names = c("Observed", "Expected")

exp_iat_result <- chisq.test(x=counts_table_exp)
gam_iat_result <- chisq.test(x=counts_table_gam)
library(lubridate)
# Process service times, convert to seconds
st <- data$`Service time`
st_times <- format(st, format = "%H:%M:%S")
st_seconds <- period_to_seconds(hms(st_times))

# remove 0 service times (voice assistant) and outliers
st_seconds<- st_seconds[st_seconds > 0 & st_seconds<3600]
# fit to distributions using MLE
s_fit.exp <- fitdistr(st_seconds, "exponential")
s_fit.gamma <- fitdistr(st_seconds, "gamma")
s_pred.density <- density(st_seconds)
x <- seq(0, max(st_seconds), length.out = length(s_pred.density$y))

s_fit.exp.density <- dexp(x, s_fit.exp$estimate)
s_fit.gamma.density <- dgamma(x, s_fit.gamma$estimate[1], s_fit.gamma$estimate[2])

pander(pandoc.table(s_fit.exp$estimate, style="rmarkdown", col.names = c("Rate"), caption="Expo
pander(pandoc.table(s_fit.gamma$estimate, style="rmarkdown", col.names = c("Shape", "Rate"), ca
cols <- c("KDE"="red","Exponential"="blue","Gamma"="green")

ggplot() +
  geom_histogram(aes(x=st_seconds, y=..density..), col = 'black', fill ='white') +
  geom_density(aes(x=st_seconds, color='KDE')) +
  geom_line(aes(x=x, y=s_fit.exp.density, color='Exponential')) +
  geom_line(aes(x=x, y=s_fit.gamma.density, color='Gamma'))+
  scale_color_manual(name="Legend",values=cols) +
```

```
    ylim(c(0, 0.01)) +
  labs(title = "Histogram of Service Time Density",
       subtitle = "Fitted to Exponential and Gamma Distributions")+
  ylab("Density") +
  xlab("Time (seconds)") +
  theme_minimal()

# Sample variates using estimated parameters
set.seed(123)
s_exp.variates <- rexp(length(st_seconds), rate= s_fit.exp$estimate) # mean
set.seed(123)
s_gamma.variates <- rgamma(length(st_seconds), s_fit.gamma$estimate[1],
                           s_fit.gamma$estimate[2]) # shape and rate


# Initial split of observations into 20 bins
# Expected counts should be >=5
bins.for.chisq <- seq(0, max(st_seconds), length.out=20)

observed <- table(cut(st_seconds, bins.for.chisq))
expected_exp <- table(cut(s_exp.variates, bins.for.chisq))
expected_gam <- table(cut(s_gamma.variates, bins.for.chisq))

# Bins for exp distribution.
bins.for.chisq.exp <- append(bins.for.chisq[1:8], 720)

observed_exp <- table(cut(st_seconds, bins.for.chisq.exp))
expected_exp <- table(cut(s_exp.variates, bins.for.chisq.exp))

# Bins for gamma distribution.
bins.for.chisq.gam <- append(bins.for.chisq[1:7], 720)

observed_gam <- table(cut(st_seconds, bins.for.chisq.gam))
expected_gam <- table(cut(s_gamma.variates, bins.for.chisq.gam))

# Create contingency tables of observed vs expected
counts_table_exp <- data.frame(observed_exp)
counts_table_exp$Expected_Exp <- expected_exp
# reassign row names
bin_names <- counts_table_exp[1]
counts_table_exp <- counts_table_exp[,-1]
rownames(counts_table_exp) <- bin_names[,1]
colnames(counts_table_exp) <- c("Observed_Exp", "Expected_Exp")


# Repeat for gam
counts_table_gam <- data.frame(observed_gam)
```

```r
counts_table_gam$Expected_Gam <- expected_gam
# reassign row names
bin_names <- counts_table_gam[1]
counts_table_gam <- counts_table_gam[,-1]
rownames(counts_table_gam) <- bin_names[,1]
colnames(counts_table_gam) <- c("Observed_Gam", "Expected_Gam")

exp_st_result <- chisq.test(x=counts_table_exp)
gam_st_result <- chisq.test(x=counts_table_gam)
pander(pandoc.table(counts_table_exp, style="rmarkdown", col.names = c("Observed", "Expected")
pander(pandoc.table(counts_table_gam, style="rmarkdown", col.names = c("Observed", "Expected")
### Chi square test
# 1. Compare observed with expected exponential counts
pander(exp_iat_result,
       caption="Inter Arrival Time Chi-Squared GOF Test for Exponential Distribution")
# p-value = 5.592e-12

#2. Compare observed with expected gamma counts
pander(gam_iat_result, caption="Inter Arrival Time Chi-Squared GOF Test for Gamma Distribution
# p-value = 0.2001
### Chi square test
# 1. Compare observed with expected exponential counts

pander(exp_st_result, caption="Service Time Chi-Squared GOF Test for Exponential Distribution"

# 2. Compare observed with expected gamma counts

pander(gam_st_result, caption="Service Time Chi-Squared GOF Test for Gamma Distribution") # p-
```

Python Code (Simulation)

Imports and utility functions

```
import numpy
import random
import scipy
import pylab
from SimPy.Simulation import *
import math
import time
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

def conf(L):
    """confidence interval"""
    lower = numpy.mean(L) - 1.96*numpy.std(L)/math.sqrt(len(L))
    upper = numpy.mean(L) + 1.96*numpy.std(L)/math.sqrt(len(L))
    return (round(lower, 3),round(upper, 3))
```

Model 1 (M/M/4)

```
# M/M/4 Model
class Source(Process):
    """generate random arrivals"""
    def run(self, N,lamb,mu):
        for i in range(N):
            a = Caller(str(i))
            activate(a, a.run(mu))
            t = random.expovariate(lamb)
            yield hold, self, t

class Caller(Process):
    n = 0 # number in system
    def run(self,mu):
        Caller.n += 1
        arrivetime = now()
        # Observe total number in system
        G.numbermon.observe(Caller.n)
        if (Caller.n>0):
            G.busymon.observe(1)
        else:
            G.busymon.observe(0)
```

```
        if random.random() > G.va_prob:
            # Gone to agent
            G.voiceassistmon.observe(0)
            yield request, self, G.agent

            # record waiting time in queue
            G.waitmon.observe(now() - arrivetime)

            t = random.expovariate(mu)
            yield hold, self, t
            yield release, self, G.agent
            # observe time that agents are busy
            delay = now() - arrivetime
            G.servicemon.observe(delay)
        else:
            # Gone to voice assistant
            G.voiceassistmon.observe(1)
            G.waitmon.observe(0)

        Caller.n -= 1
        G.numbermon.observe(Caller.n)
        if (Caller.n>0):
            G.busymon.observe(1)
        else:
            G.busymon.observe(0)
        # observe time that system is busy
        delay = now()-arrivetime
        G.delaymon.observe(delay)

class G:
    va_prob = 150/304 # probability that calls go to virtual assistant
    agent = 'dummy' # resource of c agents
    delaymon = 'Monitor' # monitor total time in system for all callers
    numbermon = 'Monitor' # monitor number of callers
    busymon = 'Monitor' # monitor busy time of system
    servicemon = 'Monitor' # time in system for callers who talk to agents
    voiceassistmon = 'Monitor' # prop. of calls going to voice assistant
    waitmon = "Monitor" # monitor of wait time


def model1(c, N, lamb,mu,maxtime, rvseed):
    # setup
    initialize()
    random.seed(rvseed)
    G.delaymon = Monitor()
    G.numbermon = Monitor()
    G.busymon = Monitor()
```

```
    G.servicemon = Monitor()
    G.voiceassistmon = Monitor()
    G.waitmon = Monitor()

    G.agent = Resource(capacity=c, monitored=True)
    Caller.n = 0

    # simulate
    s = Source('Source')
    activate(s, s.run(N,lamb,mu))
    simulate(until=maxtime)

    # gather performance measures
    W = G.delaymon.mean()
    L = G.numbermon.timeAverage()
    B = G.busymon.timeAverage()

    Ws = G.servicemon.mean()
    Wq = G.waitmon.mean()
    V = G.voiceassistmon.mean()

    Eff_lamb = L/Ws

    return W, L, B, Ws, Wq, Eff_lamb, V

# M/M/4 Simulation
allW = []
allL = []
allB = []
allWs = []
allWq = []
allEffLamb = []
allV = []
agents=4

for k in range(50):
    seed = 123*k
    result = model1(c=agents, N=10000,lamb=0.01013,mu=0.007879,
                    maxtime=29700, rvseed=seed)
    allW.append(result[0])
    allL.append(result[1])
    allB.append(result[2])
    allWs.append(result[3])
    allWq.append(result[4])
    allEffLamb.append(result[5])
    allV.append(result[6])
```

```
# Print results:
...
```

Model 2 (G/G/4)

```
class Source(Process):
    def run(self, N):
        for i in range(N):
            c = Caller(name = f"Caller {i}")
            activate(c, c.run())
            t = random.gammavariate(G.arr_shape, G.arr_mu)
            yield hold, self, t

class Caller(Process):
    n = 0
    def run(self):
        Caller.n += 1
        arrivetime = now()
        G.numbermon.observe(Caller.n)
        if (Caller.n > 0):
            G.busymon.observe(1)
        else:
            G.busymon.observe(0)

        if random.random() > G.va_prob:
            G.voiceassistmon.observe(0)
            # Gone to agent
            yield request, self, G.agent
            G.waitmon.observe(now() - arrivetime)
            t = random.gammavariate(G.ser_shape, G.ser_mu)
            yield hold, self, t
            yield release, self, G.agent
            delay = now() - arrivetime
            G.servicemon.observe(delay)
        else:
            # Gone to voice assistant
            G.voiceassistmon.observe(1)
            G.waitmon.observe(0)
        Caller.n -= 1
        G.numbermon.observe(Caller.n)
        if (Caller.n > 0):
            G.busymon.observe(1)
        else:
            G.busymon.observe(0)

        delay = now() - arrivetime
```

```
        G.delaymon.observe(delay)

class G:
    # Estimated Gamma Parameters
    arr_mu = 1/0.005845
    arr_shape = 0.577
    ser_mu = 1/0.01093
    ser_shape = 1.387

    va_prob = 150/304 # probability that calls go to virtual assistant
    agent = 'dummy' # resource of c agents
    delaymon = 'Monitor' # monitor total time in system for all callers
    numbermon = 'Monitor' # monitor number of callers
    busymon = 'Monitor' # monitor busy time of system
    servicemon = 'Monitor' # time in system for callers who talk to agents
    voiceassistmon = 'Monitor' # prop. of calls going to voice assistant
    waitmon = "Monitor" # monitor of wait time


def model2(c, N, maxtime = 200000, rvseed = 12345):
    # setup
    initialize()
    random.seed(seed)
    G.busymon = Monitor()
    G.waitmon = Monitor()
    G.delaymon = Monitor()
    G.numbermon = Monitor()
    G.servicemon = Monitor()
    G.voiceassistmon = Monitor()

    G.agent = Resource(capacity=c, monitored=True)
    Caller.n = 0

    # simulate
    s = Source()
    activate(s, s.run(N))
    simulate(until=maxtime)

    # gather performance metrics
    L = G.numbermon.timeAverage()
    Wq = G.waitmon.mean()
    B = G.busymon.timeAverage()
    Ws = G.servicemon.mean()
    W = G.delaymon.mean()
    V = G.voiceassistmon.mean()
    Eff_lamb = L/Ws
```

```
    return W, L, B, Ws, Wq, Eff_lamb, V
# G/G/4 Simulation
allW = []
allL = []
allB = []
allWs = []
allWq = []
allEffLamb = []
allV = []
agents=4

for k in range(50):
    seed = 123*k
    result = model2(c=agents, N=10000, maxtime=29700, rvseed=seed)
    allW.append(result[0])
    allL.append(result[1])
    allB.append(result[2])
    allWs.append(result[3])
    allWq.append(result[4])
    allEffLamb.append(result[5])
    allV.append(result[6])

# Print results:
...
```

Model 3 (Empirical)

```
data = pd.read_excel('~/Project/Group 1 data.xlsx', sheet_name='Data')

arrivalt = data["Arrival time"]
interarrivaltimes = []
for i in range(len(arrivalt)-1):
    j = arrivalt[i]
    j = (j.second + j.minute*60 + j.hour*60*60)
    k = arrivalt[i+1]
    k = (k.second + k.minute*60 + k.hour*60*60)
    if k-j < 3600 and k-j > 0:
        interarrivaltimes.append(k-j)

servicetime = []
st = data["Service time"]
for i in range(len(st)-1):
    j = st[i]
    j = (j.second + j.minute*60 + j.hour*60*60)
    if j > 0 and j < 3600:
```

```
        servicetime.append(j)

# Generate 10,000 samples of new Interarrival and Service times
N = 10000
numpy.random.seed(123)
IAT_new = [draw_empirical(interarrivaltimes, r) for r in
           numpy.random.uniform(size=N)]
ST_new = [draw_empirical(servicetime, r) for r in
          numpy.random.uniform(size=N)]

# Simulation using empirical data
allW = []
allL = []
allB = []
allWs = []
allWq = []
allEffLamb = []
allV = []
agents = 4

for k in range(50):
    seed = 123*k
    result = model3(c=agents, N=10000, maxtime=29700, rvseed=seed)
    # if increasing N, you must also increase the number of sampled times
    allW.append(result[0])
    allL.append(result[1])
    allB.append(result[2])
    allWs.append(result[3])
    allWq.append(result[4])
    allEffLamb.append(result[5])
    allV.append(result[6])
# Print results:
...
```