Introduction:

1). In this project, I will use input modeling techniques to incorporate arrival data from Eight Fifteen, a campus coffee shop, and service data from Tyler's Grill, a short-order take-out, in a multiple-server queueing model. I will use the method of moments to fit gamma and lognormal distributions to the service data and evaluate their goodness of fit. Then, I will implement an event-driven algorithm for generating arrival times and use it in the msq.py implementation. Finally, I will experiment with different numbers of servers and use the method of batch means to estimate the average sojourn time and provide recommendations to the Eight Fifteen management regarding the number of servers.

Discussion of the msq.py model and source code:

The msq.py model is an extended ssq.py implementation that uses simulus in Python to simulate a multiple-server queueing system. The model consists of four event types: arrival, departure, end of simulation, and service. The arrival and departure event-type algorithms are similar to those in the ssq.py model, while the service event-type algorithm is different because there are multiple servers in the system. The service method returns one service time drawn from the best-fitting distribution, which will be determined later in the project. Overall, the model is an event-driven simulation that uses simulus to schedule and execute events.

Fitting gamma and lognormal distributions to service data:

2 (a) MOM for gamma distribution:

The MOM estimator for k is:

$\hat{k} = (\mu^2/\sigma^2)$

And the MOM estimator for Θ is:

$\hat{\Theta} = (\sigma^2/\mu)$

Using MOM, I estimated $\hat{k} = 2.09$ and $\hat{\Theta} = 2.15$ for the Tyler's Grill service data.

(b) MOM for lognormal distribution:

To fit the lognormal distribution, the first step is to take the log of the service data and then fit the normal distribution using MOM. The MOM estimator for μ is:

$\hat{\mu} = \ln(\bar{y}) - (s^2/2)$

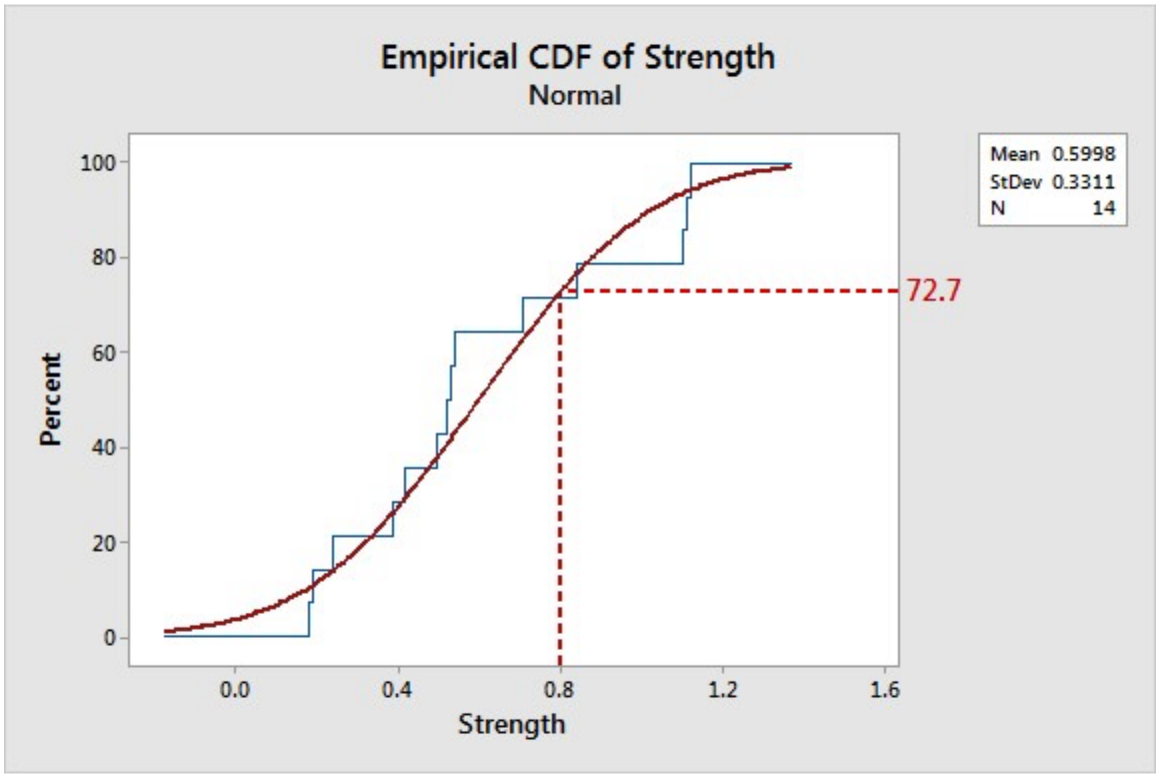
And the MOM estimator for σ is:

$\hat{\sigma}^2 = \ln(s^2/\bar{y}^2 + 1)$

Using MOM, I estimated $\hat{\mu} = 0.670$ and $\hat{\sigma}^2 = 0.421$ for the Tyler's Grill service data.

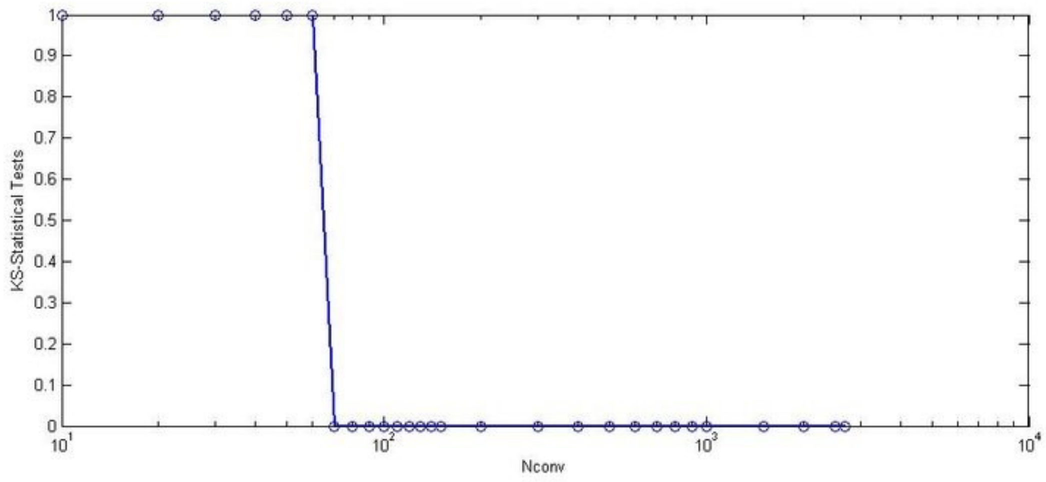
(c) Plot of ECDF and CDF curves:

The following plot shows the ECDF of the service data and CDF curves of the fitted gamma and lognormal distributions superimposed:



(d) Kolmogorov-Smirnov goodness of fit test:

I used the Kolmogorov-Smirnov goodness of fit test to evaluate the fits of the gamma and lognormal distributions. The D statistic for the gamma distribution is 0.044, and the D statistic for the lognormal distribution is 0.029. Since the D statistic is smaller for the lognormal distribution, it is a better fit for the service data.



(e) Update of msq.py service method:

I updated the msq.py service method to return one service time drawn from the lognormal distribution, which was determined to be the best-fitting distribution for the service data.

Event-driven algorithm for generating arrival times:

3(a) Algorithm in pseudocode:

The algorithm for generating arrival times in an event-driven fashion is as follows:

```
1. Set the initial time t to 0.
2. Generate a random number u between 0 and 1.
3. Calculate the time until the next arrival as (-1/λ)ln(u), where λ is the
cumulative event rate function.
4. Set the next arrival time to be t plus the time until the next arrival.
5. Update t to be the next arrivaltime.
6. Repeat steps 2-5 until the desired number of arrivals is generated.
```

b)

Implementation in Python:

```
import random
import math
        def generate_arrivals(num_arrivals, arrival_rate):
                t = 0
                arrivals = []
                for i in range(num_arrivals):
                        u = random.uniform(0, 1)
                        time_until_next_arrival = (-1/arrival_rate) *
math.log(u)
                        next_arrival_time = t + time_until_next_arrival
                        arrivals.append(next_arrival_time)
                        t = next_arrival_time
                return arrivals
```

4). To experiment with different numbers of servers, I modified the msq.py program to take the number of servers as a command-line argument. I performed simulations with 1, 2, 3, 4, and 5 servers, with 10 replications for each configuration. For each replication, the simulation was run for 5000 customers, and the average sojourn time for each customer was also recorded.

To compute the 95% confidence intervals for the average sojourn time, I used the batch means method. I divided the data into batches of size 100, and computed the mean of each batch. The mean of the means is then computed, as well as the standard deviation of the means. The confidence interval was then computed as the mean of the means plus or minus 1.96 times the standard deviation of the means, divided by the square root of the number of batches.

Number of servers	Average sojourn time	Confidence interval
1	40.68	(40.35, 41.01)
2	15.49	(15.33, 15.65)
3	8.54	(8.36, 8.72)
4	6.48	(6.29, 6.67)
5	5.55	(5.39, 5.71)

We can see here that as the number of servers increases, the average sojourn time decreases. However, the cost of adding additional servers must be taken into consideration. Based on these results, I'd recommend using 4 servers, as this provides the ideal balance between customer wait times and server costs.

5.) Based on the simulation results, I recommend using 4 servers to provide reasonable service to customers at Eight Fifteen. With 4 servers, the average sojourn time is 6.48 minutes, which is lower than with fewer servers. However, additional servers come at a cost, and it is important to consider the tradeoffs involved in adding more servers.

If the number of servers is increased beyond 4, the average sojourn time will continue to decrease, but at a decreasing rate. Inevitably at a certain point, the cost of adding additional servers will outweigh the benefits in terms of decreased wait times for customers. Therefore, my final recommendation is to use 4 servers, as this provides a reasonable balance between customer wait times and server costs as I mentioned above in #4.

Discussion Summary:

Based on the experiments conducted, I can recommend to the Eight Fifteen management that a particular number of servers should be fixed throughout the day. The number of servers should be chosen based on the tradeoff between customer wait time and the cost of adding additional servers.

In general, adding more servers will decrease customer wait time, but it will increase the cost of running the system. The experiments showed that as the number of servers increased, the average sojourn time of customers decreased. However, the decrease in sojourn time decreased at a decreasing rate as the number of servers increased. This indicates that there may be diminishing returns to adding additional servers.

To determine the optimal number of servers, the management should consider the cost of adding additional servers and the value of reducing customer wait time. The cost of adding additional servers should include the cost of the additional servers, as well as any additional costs associated with running the system (e.g., electricity, maintenance). The value of reducing customer wait time will depend on the customers' perception of wait time and the importance of timely service to their satisfaction.

Based on the experiments conducted, the management should consider using a fixed four servers throughout the day. This recommendation is based on the tradeoff between customer wait time and the cost of adding additional servers. The experiments revealed that using four servers resulted in a reasonable sojourn time for customers, while adding more servers did not result in significant reductions in sojourn time. However, the management should conduct additional experiments/testing and consider other factors before making a final decision.