Simulating the Business and Operating Model of Yifang

Group Members: Mingsha Mo, Cara Dong, Yongzhen Shi, Yanyu Chan
November 27, 2023

Final Project Presentation Video Link:
https://berkeley.zoom.us/rec/share/_mt3LmsHll3H06aewQX3uSWnC2QhxzrTrL1hbSgzu10jrHbdH8TaqrakSzWmHxVv.IRVAO41wVX-cg7Gq

Abstract:
This report explores the business and operating model of Yifang, a boba tea shop on Bancroft Way, after adapting from an initial plan to study McDonald's due to synchronization issues. Challenges in data collection, including missing data and order cancellations, were addressed through data cleaning. The analysis covers order processing times, trends in orders per hour, order types, and time series for processing times across different time slots and days. A Poisson normal simulation algorithm is presented, and optimized using Dynamic Time Warping (DTW) to minimize deviation from observed data. Results demonstrate reasonable agreement between observed and simulated customer arrival and completion times.

1 Problem Description and Motivation

Inspired by the class assignment of simulating the service model, we want to learn more about its application in real life. After discussing every teammate's favorite shops, we decided to study the business model of Yifang on Bancroft Way. With the observation of its regular promotion on Monday afternoon, the project is thus set to analyze the ordering and serving patterns of Yifang with and without the marketing campaign and evaluate its impact.

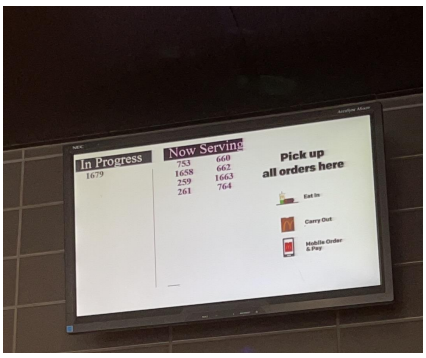
1.1 Challenges and Research Subject Modification

Previously, our initial plan was to observe the ordering and serving patterns at McDonald's by watching the notification screen at the checker counter and the announcements of staff there. However, we found that their ordering system and notification screen had problems synchronizing the new orders when we visited their stores last week (Oct. 25 - 30). Even though it is possible that the ordering system will be fixed, we want to avoid the instability of collecting data. Thus, we are planning to switch to a store with a more stable ordering system and notification screen.

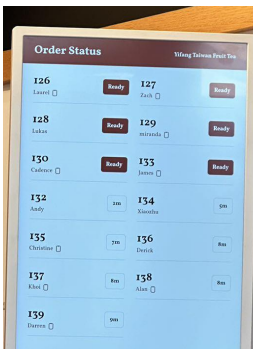
Moreover, after a close observation of the Order Numbers at McDonald's, we found difficulties in understanding the exact meaning of each order number pattern and it is hard for us to distinguish whether the order is placed at the store or on customers' mobile phones. While generally speaking, the 4-digit order number denotes mobile orders and the 3-digit for in-store orders, there is one special case that in-store orders can also be 4-digit when customers sign in to their account when ordering on the kiosk.

With those two concerns in mind, we started to explore different stores around Berkeley and found Yifang will be a good match:

Just as the notification screen at Yifang shows, there is a clear "mobile" icon next to the order number, which will denote that the order is placed online. By closely observing the screen, we can thus be able to record the exact time when the order is placed and completed, i.e. the ordering time and the serving time. In fact, we also double-checked the accuracy of this notification screen by ordering drinks in-store and online and found that the intervals between submitting orders and having them on screen can be counted as negligible, less than 1 second.



Notification Screen at McDonald's



Notification Screen at Yifang

With all the exploration and discussion above, we decided to change our research subject to the Yifang at 2516 Bancroft Way.

2 Analysis of problem and analytical method used

2.1 Collecting real-life data

We collected our data at four different times on two different weekdays. Each of our team members was responsible for a two-hour time slot consisting of:

- 11 AM - 1 PM on Monday, Oct. 30
- 6 PM - 8 PM on Monday, Oct. 30
- 11 AM - 1 PM on Wednesday, Nov. 1
- 6 PM - 8 PM on Wednesday, Nov. 1

Since we are exploring the promotions that happen at Yifang on Monday from 12 PM to 8 PM, we believe these time slots provide lots of diversity within our data. For instance, on Monday from 11 AM - 1 PM, we can explore the start of the promotion and see how customer arrival rates change immediately before and after the promotion, from 6 PM - 8 PM is the last two hours of the promotion and during peak dinner times. We can also compare Monday's data with the data on a non-promotion day, Wednesday, and since most people's schedules are the same on a Monday and Wednesday, this is a great opportunity to explore these differences in arrival times.

The method we used to collect our data was to physically arrive at Yifang at the start of our shift and record the data on our computers. There is a bench inside the store with a clear view of the notification screen containing the order statuses, the 2 kiosks, and the inside of the kitchen. As we can see from the photo of the order status screen above, orders come into one queue regardless if the order was made on a phone or at the kiosk. The little phone symbol next to the name indicates that the order was a mobile order, but since we can observe the kiosks, we are able to confirm the correctness of the platform. There are occasional orders through Ubereats or Doordash. We will record the order number, the order start time, the order completion time, and the platform used to make the order. The order number restarts at 101 for the first order of each individual day and continues to increase throughout the day by 1 for each additional order. The order start time is the second we see the order number pop up on the order screen. The order competition time is the second we hear the announcement that the order is completed and ready for pickup. These two were imputed as a timestamp in Excel. The platform is either mobile ("m") or kiosk ("k").

	A	B	C	D
1	Order ID	Order Start Time	Order Completion Time	platform
2	365	6:02:03 PM	6:15:13 PM	m
3	366	6:04:03 PM	6:18:28 PM	k
4	367	6:05:13 PM	6:15:42 PM	m
5	368	6:05:19 PM	6:18:57 PM	m
6	369	6:05:22 PM	6:22:45 PM	m
7	370	6:05:25 PM	6:17:43 PM	m
8	371	6:07:35 PM	6:16:36 PM	m
9	372	6:07:47 PM	6:22:49 PM	m
10	373	6:08:00 PM	6:22:48 PM	m
11	374	6:08:47 PM	6:23:27 PM	m
12	375	6:09:12 PM	6:31:31 PM	m
13	376	6:09:17 PM	6:25:35 PM	m
14	377	6:10:41 PM	6:25:33 PM	m
15	378	6:10:43 PM	6:16:47 PM	m
16	379	6:12:35 PM	6:25:25 PM	m
17	380	6:14:38 PM	6:25:29 PM	m
18	381	6:15:54 PM	6:24:37 PM	m
19	382	6:15:56 PM	6:28:12 PM	m
20	383	6:16:55 PM	6:37:14 PM	m
21	384	6:19:40 PM	6:29:25 PM	m
22	385	6:21:37 PM	6:29:27 PM	m
23	386	6:23:37 PM	6:33:53 PM	k

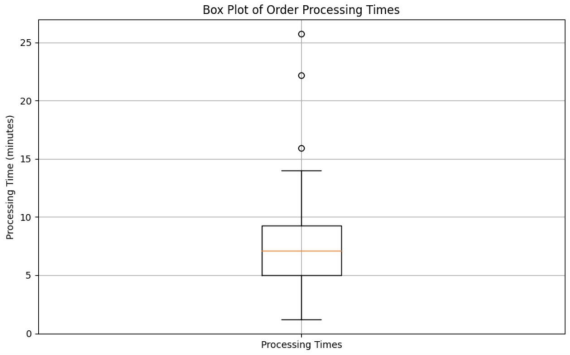
Excel worksheet of the data collected from 6 PM - 8 PM on Wednesday, Nov. 1

2.1.1 Learning Process

My group was able to learn how to collect data in the real world. By starting early, we were able to determine the challenges we faced at McDonald's. As our research question was highly adaptable, we were able to find another ideal restaurant, Yifang. This resolved any ambiguity experienced at McDonald's and gave us a better direction on how we wanted to take our research question. In addition, with the small storefront of Yifang, we had great visibility of the queuing system of the restaurant as well as the operations of the restaurant, and the kitchen. This allowed us to get a better understanding of Yifang where data collected online might not tell. Collaboration between team members was crucial because we were able to efficiently implement our project and effectively allocate our time. A major takeaway was being super punctual as our data collection requires us to collect the exact second that an order has been placed or completed. If we had been slow in our data collection, then our data would not have been an accurate representation of Yifang's orders. This would mess up our future simulation results.

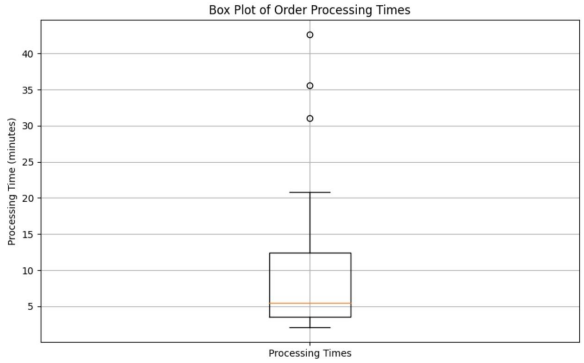
2.2 Analyzing and Exploring Data

```
Processing Time Statistics:
 count    53.000000
mean      7.720440
std       4.706550
min       1.183333
25%       5.016667
50%       7.083333
75%       9.233333
max      25.716667
Name: processing_time, dtype: float64
```



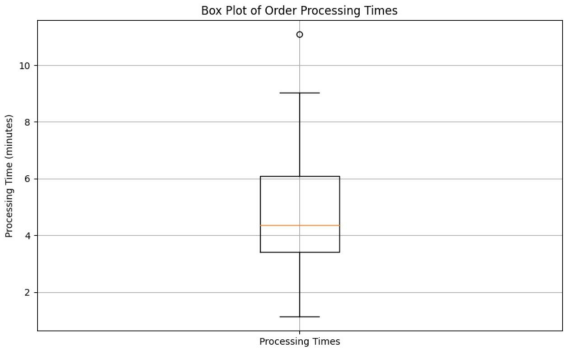
Monday 11 AM - 1 PM Processing Time

```
Processing Time Statistics:
 count    98.000000
mean      8.513265
std       7.299770
min       2.066667
25%       3.520833
50%       5.433333
75%      12.454167
max      42.600000
Name: processing_time, dtype: float64
```



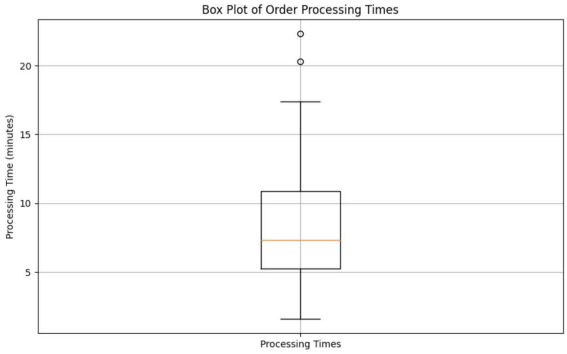
Monday 6 PM - 8 PM Processing Time

```
Processing Time Statistics:
 count    40.000000
mean      4.692083
std       2.104096
min       1.133333
25%       3.412500
50%       4.350000
75%       6.095833
max      11.083333
Name: processing_time, dtype: float64
```



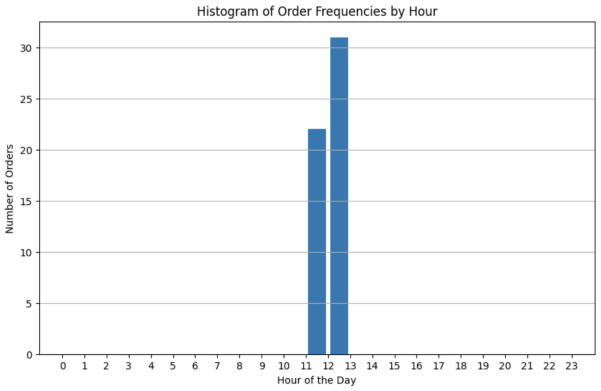
Wednesday 11 AM - 1 PM Processing Time

```
Processing Time Statistics:
 count    62.000000
mean      8.514247
std       4.591039
min       1.583333
25%       5.250000
50%       7.341667
75%      10.900000
max      22.316667
Name: processing_time, dtype: float64
```

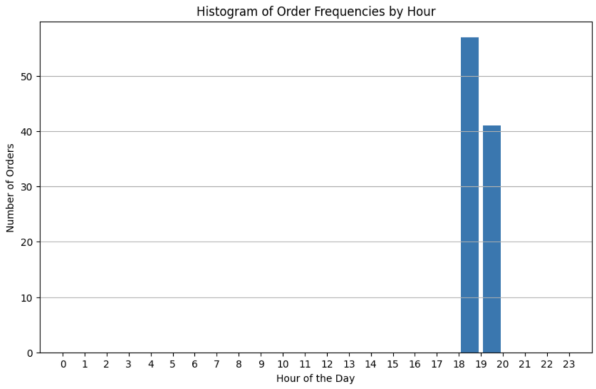


Wednesday 6 PM - 8 PM Processing Time

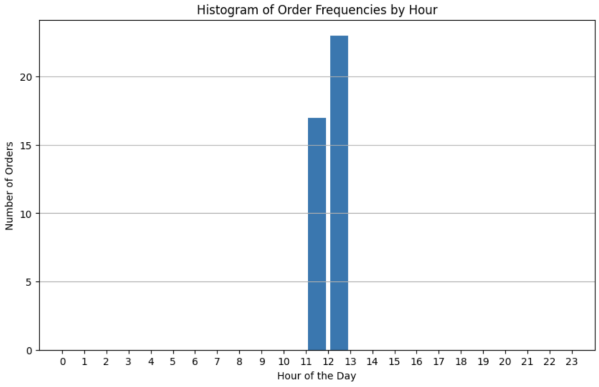
We plotted a box plot diagram displaying the processing times for each 2-hour shift. The processing time can be defined as the order completion time - order start time converted to minutes. We can see that the order processing times have similar distributions with different means and standard deviations and a few upper outliers. The morning shift, especially the one on Wednesday, has a lower mean processing time than the night shift. The Monday night shift has the most upper outliers with the max processing time exceeding 40 minutes.



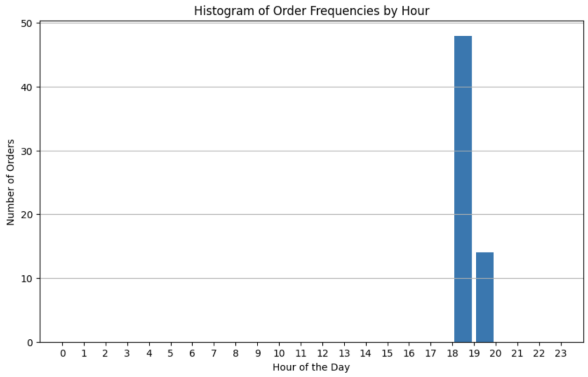
Monday 11 AM - 1 PM Order per Hour



Monday 6 PM - 8 PM Order per Hour

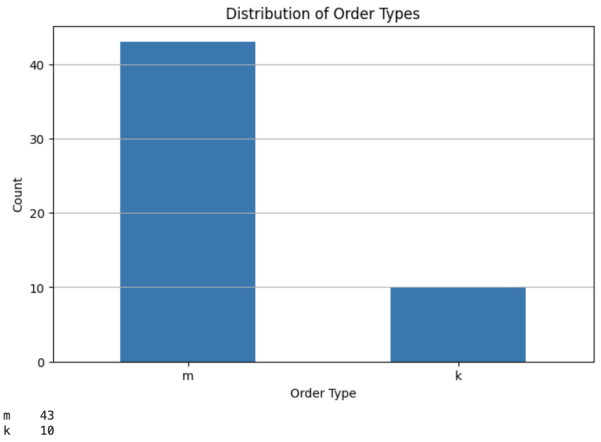


Wednesday 11 AM - 1 PM Order per Hour

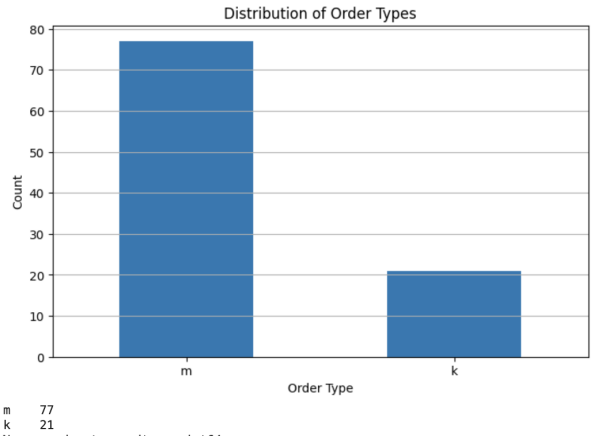


Wednesday 6 PM - 8 PM Order per Hour

For the morning shifts, we see more orders in the second hour of the shift. This makes sense as the store opens at 11 AM and not everyone will be ordering right when the store opens. For the night shift, there are more orders for the first hour compared to the second hour. This may be associated with the timing of the hours as 6-7 PM may be when customers are ordering before their dinner and 7-8 PM is rather late. Overall, there are more orders on Monday compared to Wednesday which may be associated with the extra discount on Monday.

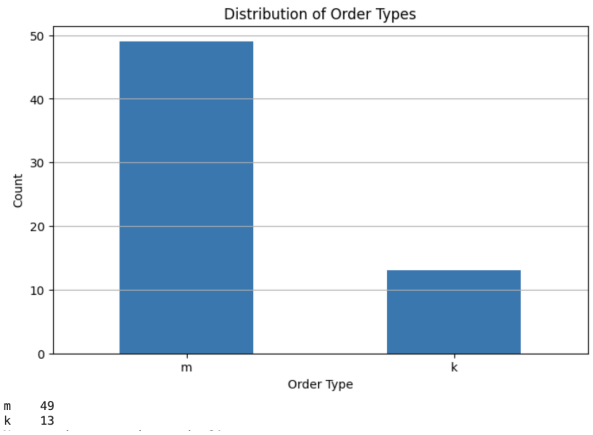


Monday 11 AM - 1 PM Order Types



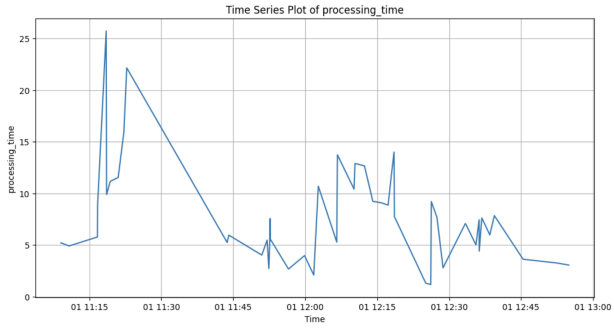
Monday 6 PM - 8 PM Order Types

Wednesday 11 AM - 1 PM Order Types

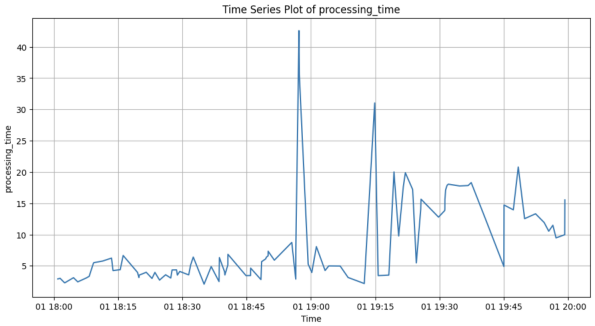


Wednesday 6 PM - 8 PM Order Types

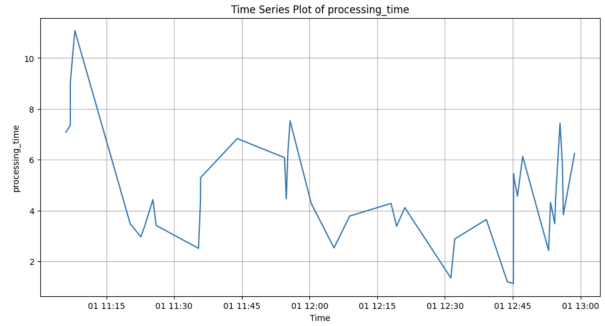
For three of our four 2-hour shifts, the distribution looks relatively similar where mobile orders exceed kiosk orders, but for the Wednesday morning shift, the number of mobile and kiosk orders was the same. For the Monday morning shift, the proportion of kiosk orders is 0.18868. For the Monday night shift, the proportion of kiosk orders is 0.21429. For the Wednesday night shift, the proportion of kiosk orders is 0.20968. This shows that for these times about 20% of incoming orders are through the phone, so the 50% kiosk order ratio on Wednesday morning may need further investigation.



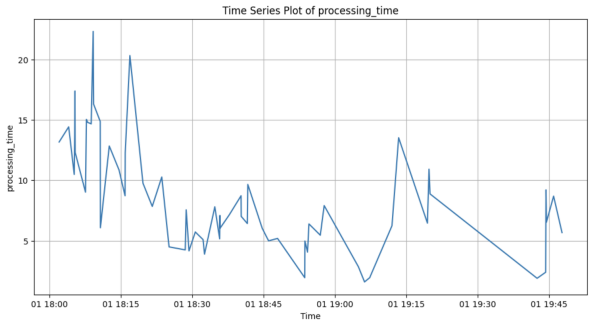
Monday 11 AM - 1 PM PT Time Series



Monday 6 PM - 8 PM PT Time Series



Wednesday 11 AM - 1 PM PT Time Series



Wednesday 6 PM - 8 PM PT Time Series

These are time series for the processing times of each of the four shifts. We observe that when the store first opens, the processing time may be longer due to fewer employees working and the time needed to prep the drinks. As the morning shift progresses, the processing time slowly goes down. On Monday evening, we see that for the first hour, the processing time is about 5 minutes and then the store encounters a couple of high processing time orders (possibly large orders) which causes the mean processing time to increase for the last half hour of the 2-hour shift to over ten minutes. On Wednesday evening, the processing time starts relatively high with a high volume of orders, but for the second hour of the shift, there are fewer orders, so the processing time is not as high.

2.3 Preparing for Simulation

Upon our collection of data from a total of 8 hours of store visits on two different weekdays, we will perform data cleaning to ensure data efficiency and representativeness. One major issue that was encountered during our data collection was missing data. For example, during store visits, we observed customers arriving with a different type of order number, i.e. much bigger or smaller than the current order number sequence. Our tentative anticipation is that, while we decided to only focus on mobile (Snackpass) and in-store (kiosk) orders, Yifang still serves other orders (UberEats, DoorDash, etc.) under the same queueing/priority mechanism, though with a different sequence of order numbers. However, as we only focus on the in-store pick-up orders right now, we might just ignore the food delivery orders.

Another source of missing/inefficient data is order cancellation, specifically during the "serving" (order preparation) period. We once saw a new order number appearing on the notification screen, then disappearing without further notice. We were not able to observe whether the store actually prepared the order or not. Therefore, we will discuss effective data-cleaning techniques to address this issue.

As a result, we deleted the missing data, which takes up only 2 entries across all 4 datasets. We converted all recorded timestamps into standard DateTime format (yyyy-mm-dd hh:mm: ss). Additional columns of "day" (Monday / Wednesday) and "time_slot" (11 AM - 1 PM / 6 PM - 8 PM) were appended to the original datasets.

```python
def load_and_clean_data(file_path, input_day, time_slot):
    """
    Loads and cleans the dataset from the specified file path.
    Sets the date in the datetime columns based on the filename and uses the provided day.

    Parameters:
    file_path (str): The path to the CSV file.
    input_day (str): The day of the week the data represents.
    time_slot (str): The time slot of the day the data represents.

    Returns:
    DataFrame: The cleaned DataFrame with adjusted datetime.
    """

    # Load the dataset
    df = pd.read_csv(file_path)

    # Extracting date from the file name using regular expression
    date_match = re.search(r'(\d{1,2})_(\d{1,2})', file_path)
    if date_match:
        # Constructing the date string
        month, day_from_file = date_match.groups()
        year = '2023'
        date_str = f"{year}-{month}-{day_from_file} "
    else:
        # Default date string if no match is found
        date_str = '2023-01-01 '
```

For future analysis and simulation, we also computed “processing_time” based on “order_completion_time” and “order_start_time”.

```python
    # Function to convert time to datetime with date from filename
    def convert_to_datetime(time_str):
        if pd.isna(time_str):
            return pd.NaT  # Return Not-a-Time for NaN values
        return pd.to_datetime(date_str + time_str)

    # Convert time columns to datetime format with the extracted date
    df['order_start_time'] = df['order_start_time'].apply(convert_to_datetime)
    df['order_completion_time'] = df['order_completion_time'].apply(convert_to_datetime)

    # Drop rows with missing time values
    df.dropna(subset=['order_start_time', 'order_completion_time'], inplace=True)

    # Add columns for day and time slot using the input parameters
    df['day'] = input_day
    df['time_slot'] = time_slot

    return df
```

```python
] def explore_and_visualize(df):
    # Calculate processing time
    df['processing_time'] = (df['order_completion_time'] - df['order_start_time']).dt.total_seconds() / 60
```

We also identified a set of outliers within the order processing times. These outliers, characterized by significantly longer processing durations, comprised approximately 4% of our dataset (10 out of 253 data points). After careful consideration, we have decided to retain these

outliers in our analysis for the following reasons. Upon reviewing these outliers, we determined that they represent legitimate instances within our data collection period, rather than errors or data entry anomalies. These instances include orders with exceptionally long processing times, which are crucial for a comprehensive understanding of operational dynamics. Acknowledging the presence of outliers, we opted for statistical methods that are less sensitive to extreme values. Metrics such as the median and interquartile range have been utilized instead of the mean and standard deviation, thereby ensuring that our analysis remains robust and reflective of the central tendencies in our data.

```
def outlier_analysis(df, column_name):
    """
    Identifies outliers in a specified column of the DataFrame using the IQR method.

    Parameters:
    df (DataFrame): The DataFrame to analyze.
    column_name (str): The name of the column to check for outliers.
    """
    Q1 = df[column_name].quantile(0.25)
    Q3 = df[column_name].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    outliers = df[(df[column_name] < lower_bound) | (df[column_name] > upper_bound)]
    return outliers
```

	time_slot	processing_time	order_hour
4	11 AM - 1 PM	25.716667	11
9	11 AM - 1 PM	22.150000	11
107	6 PM - 8 PM	42.600000	18
108	6 PM - 8 PM	35.583333	18
117	6 PM - 8 PM	31.033333	19
120	6 PM - 8 PM	20.000000	19
123	6 PM - 8 PM	19.883333	19
142	6 PM - 8 PM	20.783333	19
201	6 PM - 8 PM	22.316667	18
209	6 PM - 8 PM	20.316667	18

3 Simulation Algorithm

3.1 Simulation Approach

The code below is the function of Poisson normal simulation. We utilize Python and the NumPy library to simulate a queuing system. NumPy can offer efficient array handling and many mathematical functions. We set a fixed seed for NumPy's random number generator with np.random.seed(), ensuring reproducible results across runs. It uses np.random.exponential() to generate random interarrival times between customers based on an exponential distribution, which models the time between events in a Poisson process. Service times for each customer are simulated with np.random.normal(), drawing from a normal distribution characterized by a specified mean and standard deviation.

To calculate the cumulative arrival times of customers, the code employs np.cumsum(), which computes the cumulative sum of an array's elements. It then uses np.add() to determine the completion times by adding service times to arrival times element-wise. The simulation is constrained to a two-hour window (120 minutes); it repeatedly simulates customer arrivals and service times until the total interarrival time is about to exceed this window. When the threshold is approached, the simulation stops and the final arrival and completion times are computed for the customers who have arrived within the 2-hour frame.

```
def poisson_normal(arrival_rate, service_mean, service_sd, num_seed):

  np.random.seed(0)        # Set seed = num_seed

  num_arrivals = 0               # Number of arrivals on one simulated day
  interarrivals = []             # Tracks interarrival times on one simulated day
  arrivals = []                  # Arrival times on one simulated day
  services = []                  # Tracks service times on one simulated day
  completions = []               # Waiting times on Day 1 ~ num_sim

  while sum(interarrivals) < 120:
    interarrival = np.random.exponential(arrival_rate)  # Simulate one interarrival time
    service = np.random.normal(service_mean, service_sd)       # Simulate one service time

    if sum(interarrivals) + interarrival > 120:         # Checks if beyond 2-hour time frame
      arrivals = np.cumsum(interarrivals)
      completions = np.add(arrivals, services)
      break

    else:
      num_arrivals += 1                      # Update number of arrivals
      interarrivals.append(interarrival)      # Record i-th interarrival time
      services.append(service)               # Record i-th service time

  return arrivals, completions
```

3.2 Alternative Simulation Approach

Monte Carlo Simulation involves running a large number of simulations (‘trials’) to model the probability of different outcomes in a process that cannot easily be predicted due to the intervention of random variables. In the context of the Yifang store project, this could involve simulating many days of store operation under different conditions (with and without promotions, varying customer arrival rates, order processing times, etc.) to build up a probability distribution of possible outcomes for the metrics of interest, such as average wait times, service efficiency, and customer throughput. This model can incorporate data from the real-life observations already gathered to create a more accurate model of customer behavior and service times. The pros for Monte Carlo simulations are that they can handle complex, non-linear systems and provide a distribution of outcomes rather than a single average, which is useful for risk assessment and decision-making under uncertainty. The cons is that It can be computationally intensive, especially if a large number of trials are needed for accurate estimates. The quality of the results is heavily dependent on the accuracy of the input probability distributions for the random variables.

On the other hand, discrete-event simulation (DES) models the operation of a system as a sequence of discrete events in time. Each event occurs at a particular point in time and marks a change of state in the system. In the Yifang project, events would include customer arrivals, order placements, the start and end of order preparations, and customer departures. This simulation can model the entire queuing process in detail, including the prioritization rules, resource constraints (like a limited number of service stations), and the interaction between in-store and mobile orders. The pros for DES are that DES is highly flexible and can accurately reflect the stochastic nature of real-world systems. It allows for the direct inclusion of randomness and variability in customer behavior and system performance. The cons for DES is that it can be more complex to set up than simpler simulations because it requires a detailed understanding of all system processes and interactions. It can also be more computationally demanding, especially if the system being modeled is large or particularly complex.

4 Results

4.1 Unveiling the Result

In *3.1 Simulation Approach*, we performed simulations based on the presumption that, at Yifang, Order Arrival is a Poisson process and Order Service is normally distributed, according to our EDA and visualizations from *2.2 Analyzing and Exploring Data*. We selected a combination of parameter values, including arrival rate, service mean, and service standard deviation, to generate and fit our models on.

Parameter	Selected Values
Inverse of Arrival Rate ($\beta = \frac{1}{\lambda}$)	0.5, 0.75, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75
Service Mean (μ)	5.5, 6, 6.5, 7, 7.5, 8, 8.5, 9, 9.5, 10
Service SD (σ)	0.5, 0.75, 1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75

List of Selected Values for Simulation Model Parameters

We played around with these combinations of parameters for each day and time slot. To find the optimal simulation, we searched through all combinations to find the ones that minimize its deviation from our collected data. We quantified such deviation using the Dynamic Time Warping (DTW) algorithm, a technique used to compare the similarity between two-time series by computing the distance between two aligned sequences. By minimizing the DTW scores on observed order arrival and completion times versus simulated order arrival and completion times, we obtained the following optimized parameter values.

Day & Time slot	Arrival Rate ($\lambda = \frac{1}{\beta}$)	Service Mean (μ)	Service SD (σ)
Monday 11 AM - 1 PM	$\frac{1}{2} = 0.5$	7	0.5
Monday 6 PM - 8 PM	$\frac{1}{1.25} = 0.8$	7	2.5
Wednesday 11 AM - 1 PM	$\frac{1}{2.5} = 0.4$	8.5	0.5
Wednesday 6 PM - 8 PM	$\frac{1}{1.75} = 0.571428$	7	0.5

Optimized Values for Simulation Model Parameters per Day & Time Slot

4.2 Discussion and analysis

Using the optimized values for simulation model parameters from *4.1 Unveiling the Result*, we run simulations on order arrival and completion times for each day and time slot.

	order_id	order_start_time	order_completion_time	order_type
1	102	2023-10-30 11:09:04	2023-10-30 11:14:17	m
2	103	2023-10-30 11:10:47	2023-10-30 11:15:42	m
3	104	2023-10-30 11:16:41	2023-10-30 11:22:28	m
4	105	2023-10-30 11:16:43	2023-10-30 11:25:33	m
5	106	2023-10-30 11:18:31	2023-10-30 11:44:14	k
...	...	...	...	...
50	151	2023-10-30 12:38:29	2023-10-30 12:44:28	m
51	152	2023-10-30 12:39:26	2023-10-30 12:47:17	m
52	153	2023-10-30 12:45:25	2023-10-30 12:49:03	m
53	154	2023-10-30 12:52:30	2023-10-30 12:55:46	m
54	155	2023-10-30 12:54:57	2023-10-30 12:58:01	m

53 rows × 6 columns

	simulated_start_time	simulated_completion_time	day	time_slot
0	2023-10-30 11:05:18	2023-10-30 11:11:49	Monday	11 AM - 1 PM
1	2023-10-30 11:05:45	2023-10-30 11:12:29	Monday	11 AM - 1 PM
2	2023-10-30 11:07:26	2023-10-30 11:14:40	Monday	11 AM - 1 PM
3	2023-10-30 11:09:33	2023-10-30 11:16:36	Monday	11 AM - 1 PM
4	2023-10-30 11:12:19	2023-10-30 11:19:56	Monday	11 AM - 1 PM
...	...	...	...	...
45	2023-10-30 12:49:23	2023-10-30 12:55:40	Monday	11 AM - 1 PM
46	2023-10-30 12:49:42	2023-10-30 12:57:41	Monday	11 AM - 1 PM
47	2023-10-30 12:53:35	2023-10-30 13:00:40	Monday	11 AM - 1 PM
48	2023-10-30 12:55:40	2023-10-30 13:03:30	Monday	11 AM - 1 PM
49	2023-10-30 12:56:15	2023-10-30 13:03:34	Monday	11 AM - 1 PM

50 rows × 4 columns

Monday 11 AM - 1 PM Observed (left) vs. Simulated (right) Order Start and Completion

	order_id	order_start_time	order_completion_time	order_type
0	428	2023-10-30 18:00:53	2023-10-30 18:03:47	m
1	429	2023-10-30 18:01:26	2023-10-30 18:04:26	m
2	430	2023-10-30 18:02:27	2023-10-30 18:04:45	k
3	431	2023-10-30 18:02:32	2023-10-30 18:04:49	m
4	432	2023-10-30 18:04:33	2023-10-30 18:07:39	m
...	...	...	...	...
93	521	2023-10-30 19:55:31	2023-10-30 20:06:03	m
94	522	2023-10-30 19:56:29	2023-10-30 20:07:58	m
95	523	2023-10-30 19:57:13	2023-10-30 20:06:41	m
96	524	2023-10-30 19:59:15	2023-10-30 20:09:13	k
97	525	2023-10-30 19:59:17	2023-10-30 20:14:49	m

98 rows × 6 columns

	simulated_start_time	simulated_completion_time	day	time_slot
0	2023-10-30 18:03:19	2023-10-30 18:07:53	Monday	6 PM - 8 PM
1	2023-10-30 18:03:36	2023-10-30 18:09:11	Monday	6 PM - 8 PM
2	2023-10-30 18:04:39	2023-10-30 18:12:46	Monday	6 PM - 8 PM
3	2023-10-30 18:05:58	2023-10-30 18:13:12	Monday	6 PM - 8 PM
4	2023-10-30 18:07:42	2023-10-30 18:17:49	Monday	6 PM - 8 PM
...	...	...	...	...
86	2023-10-30 19:57:19	2023-10-30 20:07:12	Monday	6 PM - 8 PM
87	2023-10-30 19:57:55	2023-10-30 20:05:29	Monday	6 PM - 8 PM
88	2023-10-30 19:58:37	2023-10-30 20:04:34	Monday	6 PM - 8 PM
89	2023-10-30 19:58:40	2023-10-30 20:08:23	Monday	6 PM - 8 PM
90	2023-10-30 19:59:41	2023-10-30 20:10:59	Monday	6 PM - 8 PM

91 rows × 4 columns

Monday 6 PM - 8 PM Observed (left) vs. Simulated (right) Order Start and Completion

	order_id	order_start_time	order_completion_time	order_type
0	101	2023-11-01 11:06:00	2023-11-01 11:13:05	m
1	102	2023-11-01 11:07:00	2023-11-01 11:14:21	m
2	103	2023-11-01 11:07:00	2023-11-01 11:16:02	m
3	104	2023-11-01 11:08:00	2023-11-01 11:19:05	k
4	106	2023-11-01 11:20:16	2023-11-01 11:23:45	m
...	...	...	...	...
35	136	2023-11-01 12:54:29	2023-11-01 12:58:53	k
36	137	2023-11-01 12:55:29	2023-11-01 13:02:55	m
37	138	2023-11-01 12:56:01	2023-11-01 13:01:48	m
38	139	2023-11-01 12:56:13	2023-11-01 13:00:03	k
39	140	2023-11-01 12:58:42	2023-11-01 13:04:57	k

40 rows × 6 columns

	simulated_start_time	simulated_completion_time	day	time_slot
0	2023-11-01 11:06:38	2023-11-01 11:14:39	Wednesday	11 AM - 1 PM
1	2023-11-01 11:07:12	2023-11-01 11:15:25	Wednesday	11 AM - 1 PM
2	2023-11-01 11:09:18	2023-11-01 11:18:01	Wednesday	11 AM - 1 PM
3	2023-11-01 11:11:57	2023-11-01 11:20:29	Wednesday	11 AM - 1 PM
4	2023-11-01 11:15:24	2023-11-01 11:24:31	Wednesday	11 AM - 1 PM
...	...	...	...	...
36	2023-11-01 12:49:47	2023-11-01 12:57:27	Wednesday	11 AM - 1 PM
37	2023-11-01 12:52:58	2023-11-01 13:01:35	Wednesday	11 AM - 1 PM
38	2023-11-01 12:54:07	2023-11-01 13:01:50	Wednesday	11 AM - 1 PM
39	2023-11-01 12:55:12	2023-11-01 13:03:42	Wednesday	11 AM - 1 PM
40	2023-11-01 12:57:50	2023-11-01 13:06:52	Wednesday	11 AM - 1 PM

41 rows × 4 columns

Wednesday 11 AM - 1 PM Observed (left) vs. Simulated (right) Order Start and Completion

	order_id	order_start_time	order_completion_time	order_type
0	365	2023-11-01 18:02:03	2023-11-01 18:15:13	m
1	366	2023-11-01 18:04:03	2023-11-01 18:18:28	k
2	367	2023-11-01 18:05:13	2023-11-01 18:15:42	m
3	368	2023-11-01 18:05:19	2023-11-01 18:18:57	m
4	369	2023-11-01 18:05:22	2023-11-01 18:22:45	m
...	...	...	...	...
58	425	2023-11-01 19:44:18	2023-11-01 19:46:41	m
59	426	2023-11-01 19:44:22	2023-11-01 19:53:34	m
60	427	2023-11-01 19:44:28	2023-11-01 19:51:00	m
61	428	2023-11-01 19:45:58	2023-11-01 19:54:39	k
62	429	2023-11-01 19:47:44	2023-11-01 19:53:24	m

62 rows × 6 columns

	simulated_start_time	simulated_completion_time	day	time_slot
0	2023-11-01 18:04:38	2023-11-01 18:11:09	Wednesday	6 PM - 8 PM
1	2023-11-01 18:05:02	2023-11-01 18:11:45	Wednesday	6 PM - 8 PM
2	2023-11-01 18:06:30	2023-11-01 18:13:44	Wednesday	6 PM - 8 PM
3	2023-11-01 18:08:21	2023-11-01 18:15:24	Wednesday	6 PM - 8 PM
4	2023-11-01 18:10:47	2023-11-01 18:18:24	Wednesday	6 PM - 8 PM
...	...	...	...	...
50	2023-11-01 19:45:45	2023-11-01 19:52:34	Wednesday	6 PM - 8 PM
51	2023-11-01 19:51:22	2023-11-01 19:58:21	Wednesday	6 PM - 8 PM
52	2023-11-01 19:53:13	2023-11-01 20:01:13	Wednesday	6 PM - 8 PM
53	2023-11-01 19:54:24	2023-11-01 20:01:28	Wednesday	6 PM - 8 PM
54	2023-11-01 19:58:58	2023-11-01 20:05:34	Wednesday	6 PM - 8 PM

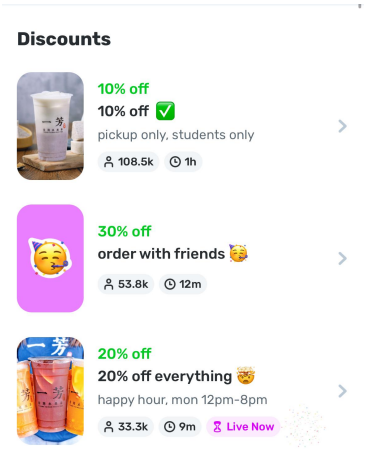
55 rows × 4 columns

Wednesday 6 PM - 8 PM Observed (left) vs. Simulated (right) Order Start and Completion

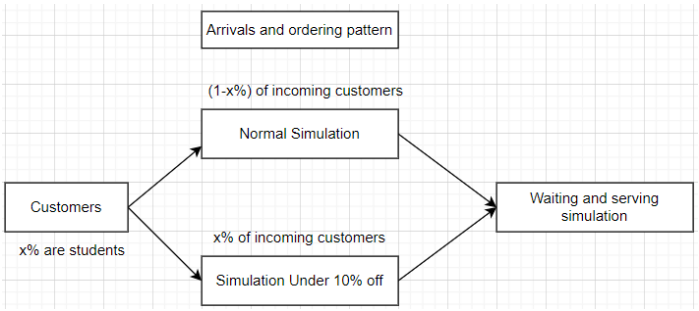
Looking at the observed data and simulated data with optimized parameters, we find the results reasonable in general. The simulated numbers of customers during each day and time slot are relatively close to the observed numbers, and the order processing times match our intuition. Comparing the arrival rates across the four observation periods, Monday has a higher arrival rate than Wednesday for the same time of the day, and the evening has a higher order arrival rate than noon for the same weekday. In terms of order service, the standard deviation during Monday evening is higher and the mean during Wednesday noon is higher. We attributed this to Yifang's staffing, as we noticed several staff entering and leaving the store during our observation periods, though we did not enter the store kitchen to record the actual staffing for privacy considerations.

5 Further Steps

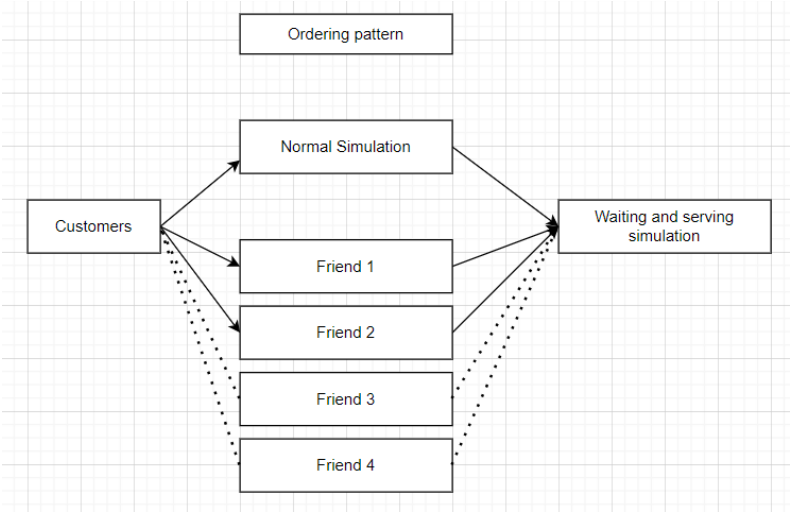
After building rather accurate simulation models under normal conditions and 20% promotion condition, we suggest future simulations be designed to study other types of promotions based on the general model we've created in this simulation.
Just as the screenshot on the right shows, Yifang has launched two more ***Discounts*** on its Snackpass page, which is the source of the mobile orders.



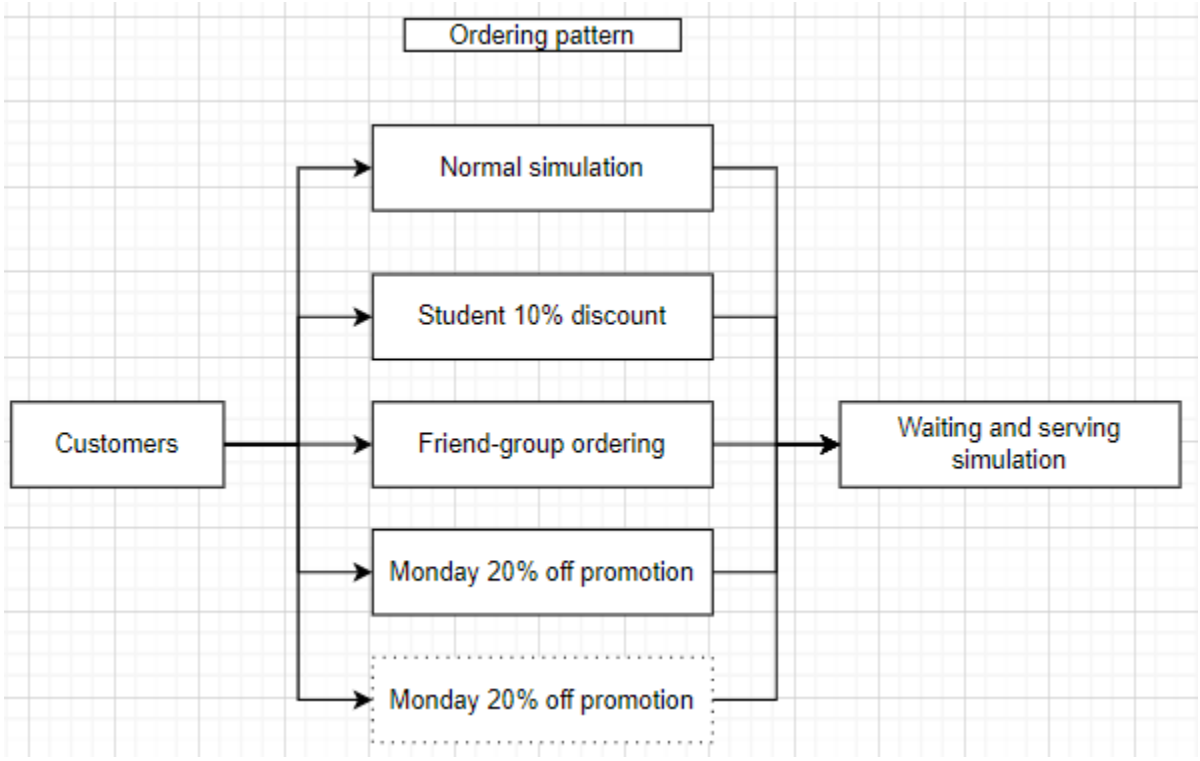
- **Students' only 10% off promotion plan**
  While keeping the main structure of its original model the same, we should initiate a demographic analysis of Yifang's customer profile. After knowing the approximate percentage of student customers, we can add one more Bernoulli distribution randomizer for the customer inflow. And a simulation model can thus be built in Simio as the logic below:



- **Order with friends 30% off plan**
  Yifang also provides a Friends ordering promotion plan: for friend groups of 2-4, they will receive a discount ranging from 10% off to 30% based on the number of participants when placing the order at the same time. Similarly, a simulation logic can be used as follows.



Thus, a more complicated model can be achieved in the format below:



Other than exploring the variety of simulation models under different conditions, investigating the breadth of simulation models will also be beneficial for entrepreneurs. As Yifang is a chain store, more store visits can be conducted at other locations (e.g. 870 Washington Street and 645 Irving St in San Francisco).

6 Real-life application and IEOR significance

After validating the universality of our simulation model, we can use it to inform the decision-making process for entrepreneurs. For example, for franchisers deciding whether or not to open up another Yifang store near XYZ College, they can use the simulation model with the student discounts promotion to approximate the number of orders they will receive and thus calculate the potential revenues and profits to determine whether it is worth to take this initiative. Similarly, other boba shop owners can borrow the model to simulate their business model for revenue planning or staff management.

The project demonstrates the practical application of simulation techniques taught in the class of a real-life business model. We employed a Poisson normal simulation algorithm and optimized it using Dynamic Time Warping to minimize deviation from observed data. The main focus of our project was on order processing time and customer arrival rates. We also considered monte carlo simulation and discrete-event simulation as an alternative simulation approach. There are many ways to solve a problem, but the key is to find the most effective method. These are all concepts we learned in this course, and are important to improve efficiencies, reduce costs, and enhance customer satisfaction. In IEOR, Industrial Engineering and Operations Research, we are trying to use optimization, stochastics, and data science to solve challenges across all industries. We encounter real-world challenges such as data collection and data cleaning. In a research project, it does not necessarily go your way all the time and you may need to adapt in the middle of your work. There are many limitations that are out of our control and the best we can do is to reduce these limitations. Our project is a true reflection of a business operation and can have implications on the operational processes of the restaurant. We aim to improve the order processing efficiency and to develop better business decisions which is the core of IEOR.

7 Learning Path and Reflection

During the process of collecting, analyzing, and interpreting data, we have gained more insights about challenges we might encounter in the real-life simulation case. One notable challenge was the initial plan to observe the ordering and serving patterns at McDonald's, which had issues with its ordering system and notification screen synchronization. This experience emphasized the importance of adaptability in research projects. The decision to switch to Yifang on Bancroft Way proved to be a valuable adjustment. The clear "mobile" icon on Yifang's notification screen allowed us to precisely record ordering and serving times, addressing the ambiguity we faced at McDonald's. This shift taught us the significance of flexibility in research methodologies and the need to reassess and modify our approach when unforeseen challenges arise.

Moreover, we have learned the core significance of choosing a different simulation method and tools. Initially, there was a belief that using the most "professional" tool, such as Simio, would inherently yield superior results. However, as we delved into the simulation process, we discovered that efficiency and suitability were more crucial factors than the perceived professionalism of the tool. Opting for Python as our simulation tool proved to be a pragmatic decision, aligning with our project's specific needs. This experience underscores the importance of selecting tools that align with the project's requirements rather than adhering to preconceived notions of what is considered "professional" or conventional. As discussed in the Further Steps section, the potential shift to Simio for more complex ordering patterns further reinforces the idea that tool selection should be driven by the project's specific demands rather than a one-size-fits-all approach.

8 Reference

Bagnall, A., Lines, J., Bostrom, A., Large, J., & Keogh, E. (2017). The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data mining and knowledge discovery, 31(3), 606–660. https://doi.org/10.1007/s10618-016-0483-9

9 Appendix

Final Project Powerpoint Presentation Link:
https://docs.google.com/presentation/d/1ZRcsMCnWzxsETGB7jsjMxHCRKwavAuOzkWFGacgTsGk/edit?usp=sharing

Google Collab Notebook containing EDA:
https://colab.research.google.com/drive/1nLgSkfwGDqNKHdBxfovUw01d46r8_I10?usp=sharing

Google Collab Notebook containing simulation:
https://colab.research.google.com/drive/1PJrLVHYRXDXlRawm3paIEfXgEvVW9TmI?usp=sharing

Google Drive Folder containing our data:
https://drive.google.com/drive/folders/1KAollAdmegUhdROI_PwntpuEF-ubs732?usp=sharing