Prediction of queue waiting times

Oriol Prat Font

June 1st, 2023

Abstract—Queuing theory is a branch of mathematics that studies, analyses and predicts the behaviour of queues using queuing models. Even though queuing models were originated in the early 1900s, they are not widely used in modern times. This project uses the Poisson distribution together with exponential distribution and other probability techniques to implement a queuing model that can predict the behaviour of queues and estimate the expected customer waiting times and their lengths. The project also involves the generation of plots for representing and analysing the data. The output data is then exported and uploaded to a dashboard in order to provide visualizations and get crucial insights. Furthermore, alerts are connected to the dashboard so that they set off when a queue exceeds a previously defined limit.

Python is used to implement the queuing model, and the simulations are made in the discrete-event simulation framework SimPy, that is based on Python. The plots are also generated in Python, using the library Matplotlib. On the other hand, the dashboard used to visualize the data is created with Power BI.

Index Terms— behaviour of a queue, discrete event, exponential distribution, Matplotlib, monitorization, M/M/1, Poisson, Power BI, Python, SimPy

1 Introduction

AMUSEMENT parks are popular destinations for people seeking entertainment and adventure. These parks offer a wide range of attractions, from breath-taking roller-coasters to family rides. However, there's something that frustrates everyone: long queues. Queues can lead to a negative experience of the park and can cause discomfort to the visitors.

This project focuses on a big amusement park that has this exact problem. The understanding of the behaviour of its queues is needed in order to find alternatives that can provide the fast services visitors demand. Understanding how queues work and being able to predict them can optimize processes and increase the visitor's satisfaction.

The main objective of the project is to reduce the wait times of the park's queues by accurately predicting their actual waiting times using a queuing model and other probability techniques. Additionally, plots and charts are designed and implemented to easily analyse the queue. Furthermore, by monitoring the system using dashboards the behaviour of the queue can be understood and, therefore, different options for improving the queuing system can be contemplated.

2 State of the art

At present, companies from all kinds of sectors have already designed queuing models and queue management systems. Their objective is to optimize their businesses, and queuing theory can contribute to it because it can provide shorter waiting times, that lead to a higher customer satisfaction.

Queuing theory has been studied throughout the last few years, which has contributed to the creation of diverse technologies used to optimize queues. These first technologies date back to the 1970s, where physical and electronic methods such as barriers and tickets were first developed. Later on, big industries like healthcare and retail began using these technologies, which lead to an improvement in queuing management systems by implementing real-time data and customer feedback tools.

Today, queue management systems are used in mobile apps, virtual queues, and even contactless payments. They are also starting to implement machine learning algorithms to predict customer behaviour and future long waiting times.

In recent years, there has been significant research on queueing models in amusement parks, with the aim of improving queue management and enhancing visitor experience. This specially includes models such as M/M/1 and M/M/k, to predict queue performance based on different factors, like ride capacity, arrival rates, and service times. Research has also focused on the use of simulation and analytical models to evaluate the effectiveness of different strategies for improving queue performance, such as single-rider lines, virtual queues, and express passes [1]. Machine learning and artificial intelligence are also emerging as a promising approach in the entertainment industry.

- *Contact e-mail: Oriol.PratF@autonoma.cat*
- *Project tutored by Antonio Espinosa Morales (Departament d'Arquitectura de Computadors i Sistemes Operatius)*
- *Course 2022/23*

However, there are still challenges in applying queuing models to amusement parks, such as the dynamic nature of the park environment, the complexity of its queue behaviour, and the variability of demand. Therefore, more research in this field is required to further develop and optimise queueing models in this sector, with the ultimate goal of giving visitors a better experience and increasing the park's income.

As new technologies emerge, queuing theory will become more and more valuable. One technology that can benefit from queuing models and simulations is self-driving vehicles. With autonomous cars becoming more and more common, queuing models can be used to optimize traffic flow and minimize waits during congestions. Additionally, going further, in a future world where all vehicles are autonomous, queuing models could completely get rid of congestions.

3 OBJECTIVES

After defining and analysing the problem to be solved, a series of objectives are outlined to successfully conclude this project:

- Accurately predict queue waiting times.
- Understand the queue's behaviour using plots.
- Monitor the system with dashboards.
- Offer support for decision making to reduce waiting times or make queues more enjoyable.

The proposal to achieve them begins with the implementation of a queuing model programmed with Python, which is an easy-to-use programming language that offers a wide range of libraries, such as Pandas [2] for data manipulation and NumPy [3] for numerical computing.

The simulation library used to create the model is SimPy [4]. This discrete-event simulation library is really useful for simulating real-life events in Python. It can easily model active components such as customers and servers, which are required in this project. There are a lot of projects that use this library, some of them being related to queueing theory and queueing systems [5].

The model created with SimPy is a variation of the M/M/1 queuing model [6], which is widely used to analyse the behaviour of single-server queues. This model assumes that customer arrivals follow a Poisson distribution, which means users arrive randomly and independently over time with an average rate λ. The service times for each customer are distributed exponentially, with an average rate μ, and there is a single server in the system, which means that only one customer can be served at a time.

As said before, the model used in this project shares similarities with the M/M/1 queuing model. However, it incorporates a different service time distribution and the number of people served at the same time is higher. Instead of assuming an exponential distribution, the service time of the attractions in the amusement park represents the length of time in which the attraction is operational and running. Therefore, the service times are independent for each attraction and depend on the running time of each ride. Furthermore, this model has single-server queues, but it has a distinction regarding the number of people attended at the same time. While the M/M/1 model assumes that only one customer is served at a time, in the amusement park multiple people get in each attraction at the same time and this number varies for each ride.

Modifying both the service times and number of people attended at the same time in each attraction makes this model more accurate and represents more realistic attraction queues. The more precise these parameters are for each attraction, the more accurate results the simulations will obtain.

The next step, once the model is implemented, is to create plots and charts using the output data from the simulations. This is also made in Python using the library Matplotlib, which provides a wide range of visualization options and customizations while being easy to use and supporting other libraries [7]. Most of these plots use queuing and service times, but they are also useful to visualize the number of people in queue at any time. All these visualizations are fundamental in order to better comprehend and easily analyse the queues.

Additionally, the data from the simulations can be visualized through a dashboard created with Power BI [8], which is recognized as one of the best tools for data visualization and analysis, making it the perfect choice for the most visual part of the project. This tool stands out due to its easily connection to Excel spreadsheets and a lot of other options, the user-friendly and interactive dashboards it provides and the availability to use real-time data. This last feature is not applicable to the amusement park at the moment but they are looking forward to implementing other ways of obtaining data from the queues in order to be able to get real-time information about the number of people waiting in each ride.

The dashboard designed in Power BI provides crucial information from the waiting times throughout the day and from all the attractions in the park. The number of people in the attractions is also displayed next to a limit for

each attraction defined by the park's managers. This last chart will be really useful in the future, once the park implements a way of obtaining real-time data of the queues, because it will provide real-time information using the Power BI dashboard. Furthermore, alarms will also be sent to the park's staff when queue lengths exceed these limits previously defined.

Finally, different alternatives to the current queue management are provided to the company in order to try and reduce the waiting times, manage them more efficiently or find other ways to make the users' experience better.

4 Methodology and planning

The methodology used to complete this project is based in the Scrum framework. As only one person is actively involved in the project, this framework has been adapted to suit the needs [9].

In the project, the Scrum responsibilities of product owner, scrum master and development team are slightly modified so that they can all be assigned to a single person.

The agile methodology used is based in sprints. In the longer tasks there have been more than one sprint and they have been divided on smaller parts of the task. The sprints have lasted between one week and two, depending on the complexity of each task. In addition, there have been weekly meetings with the project tutor. The tasks planning is shown in figure 1.

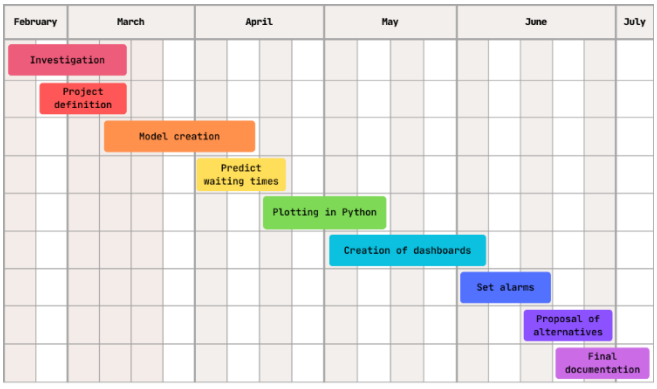

Figure 1: Planning of the project

As there's only one actively involved person in this project, no planning, tracking or control tools have been used. On the other hand, Microsoft Teams [10] and Outlook have been used as communication tools with the project tutor, although there have also been in-person meetings.

5 Development

As mentioned previously, the proposal to achieve the objectives is based in a series of steps that must be carried out in a specific way, following a systematic order. Some of these tasks have already been completed and will be presented in the following sections while the next steps will be described later on in this document.

5.1 Creation of the model

The first thing needed to start implementing the model was the data, so a dataset containing the data of 50 attractions from the amusement park was created. This data contains the number of people who join the queue of each attraction every five minutes, being the columns the time of the day and each row an attraction in the park. The columns begin at 10:00 and end at 19:55 as this is the park's schedule.

The model has been created in Python using the simulation framework SimPy. This model consists of a Queue Simulation class that receives two variables: the number of users and the arrival rate. This class has a function called *run()* that loops over each of the users and processes them in the simulated queue.

It's important to acknowledge that, in order to make the simulations as similar as possible to a real attraction, the number of people who get in a ride at the same time is different for every attraction. Therefore, every time the server is available, multiple people will start being attended at the same time and will also depart when the service is completed at the same instance. Additionally, the service time of each attraction isn't computed using the exponential distribution as in the M/M/1 queuing model because the time of service in a park is the running time of each ride, which is constant. Therefore, the service time of each attraction is an integer which is unique for each ride and is stored before running the simulations and is retrieved once the simulation of that attraction begins.

The simulated queue follows the same process as a real amusement park's queue:

1. First of all, the arrival time of the user is generated using Poisson distribution and the arrival rate defined in the class.
2. The user then waits for the simulation to get to this *Arrival time*.
3. When this instance arrives, the user is placed in the last position of the queue and it waits until the server is available and he is the next in line, with the server being unavailable referring to it attending other users in that moment.
4. When the server is available and the user is the next, it will be attended. This instance is defined as the *Queue end time*, which will also be the *Service start time*.

5. Once the user starts being attended, it will stay there until the service time is completed. That instance will be the *Departure time*.
6. At that moment the user will depart and the next user in line will start the service.

A flowchart of this process is shown in figure 2.

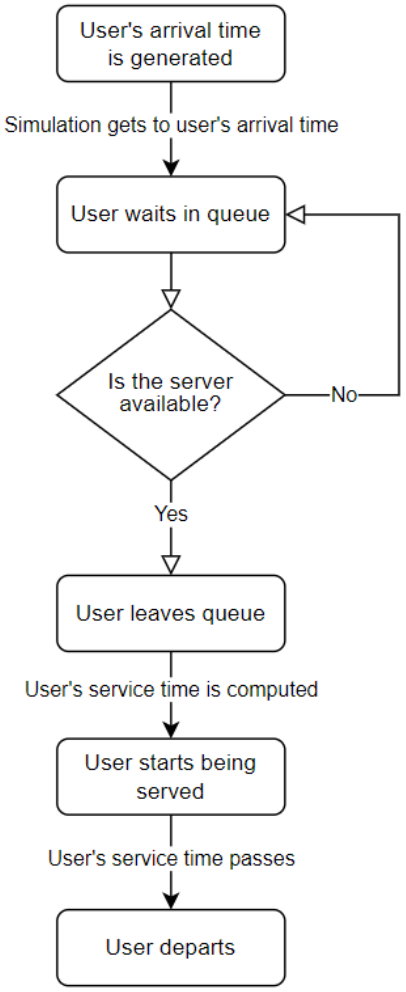

Figure 2: Simulated queue's flowchart

Once a user departs, its simulation times are saved in a Pandas dataframe where the rows refer to each of the users in the simulation. Later on, the *Time in queue* and *Service time* variables are computed using the following formulas:

$$Time\ in\ queue = Queue\ end\ time - Arrival\ time$$

$$Service\ time = Departure\ time - Service\ start\ time$$

These two variables are also saved in the dataframe with the rest of the simulation times. When all the users in the queue have been served, the simulation ends and the Queue Simulation class returns this dataframe.

However, not everything explained before will be the same way once the project is finished, as I have lately designed four different input spreadsheets that have different ranges of people in queue at each attraction. In the next weeks the queuing model will be able to plot all of them in order to get a better understanding of how the queues react to more or less people waiting, but at the moment this feature is still not available.

These four models represent different situations that can happen in the park divided by the number of people. These models are, from more to less people, special events such as Halloween, weekends, weekdays, and days with bad weather like rainy days. The purpose of having four different models is to give the company the possibility to use each one of them depending on their needs and which one of them they expect to be more similar to the reality in their park.

To conclude, the model may suffer changes in these next weeks in order to implement these four models and get a more realistic look at the number of people who go to the amusement park depending on the day of the week and other variables that can't be controlled such as the weather.

5.2 Prediction of wait times

Once the model was implemented, the data was extracted from the dataset and loaded in a dataframe using Python's library Pandas. Each of the values in the dataframe refer to the number of people who get in a specific attraction in a range of 5 minutes from the park's opening schedule. These values are used in the Queue Simulation class as the number of users.

When all the simulation are completed and the *run()* function returns the dataframe of simulation times, the average of all the *Time in queue* times is computed. This value will be the average waiting time for a specific attraction in a specific hour and it is stored in a new dataframe. The average waiting times for all attractions and times are computed and stored in this dataframe as soon as each simulation is completed.

As a result, when all the simulations are done, the script exports the average waiting times for every 15 minutes for each attraction. Additionally, the source code prints the average wait time of all the values in the exported dataframe in order to get a better understanding of how the queues have performed in the simulations.

5.3 Plotting in Python

When all the simulations are completed and the wait times

have been exported, the plotting part of the code begins. In this section a Gantt chart and a histogram of wait times are generated using Python's Matplotlib library [11]. A Gantt chart for one of the simulations is shown in figure 3.

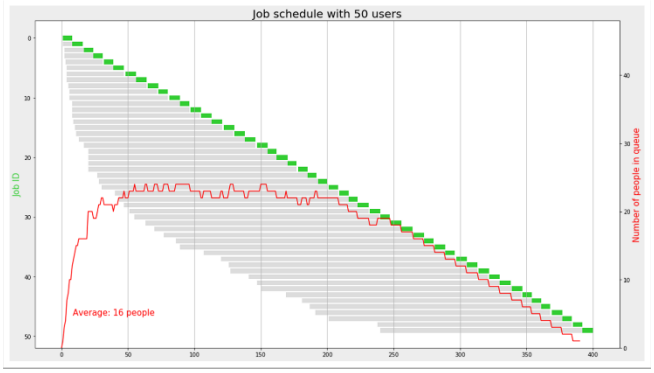

Figure 3: Gantt chart of a simulation

This chart represents the instances when every user in one of the simulations joins and waits in queue, starts being served and departs. The part where a user is waiting in queue is shown with a grey bar, while the part where it is being served is painted in green. It can be seen that the arrival times, which are the beginning of the grey bars, show a Poisson distribution. On the other hand, the service times, which are the lengths of the green bars, are almost the same in all the instances because they represent the ride times, which are constant in parks.

The number of people waiting in the queue at any moment can also be seen in colour red. We can see that as the attraction opens its gates the number of people in queue starts rising until it reaches about 20 people waiting. At that point the people in queue maintains this number until every user in that simulation has joined the queue and it starts going down as the rides are completed.

The figure 4 compares the difference on waiting times from an underloaded, a standard and an overloaded attraction respectively. The waiting times are plotted using a histogram in order to bring into focus how the waiting times vary depending on the flow of the queue and the number of users. There's also a red vertical line that indicates the average waiting time of all the simulations to provide additional information.

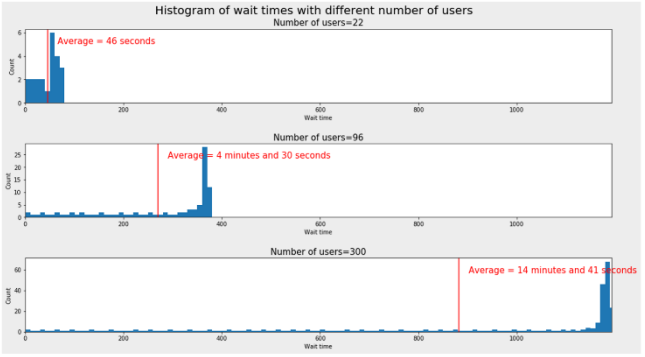

Figure 4: Histogram of waiting times

5.4 Power BI dashboard

Once all the simulations are done and all the charts are plotted using Matplotlib, the wait times of all the simulations are saved in a dataframe and exported to an Excel spreadsheet. Excel provides a versatile platform and is really useful and easy-to-use for manipulating and organizing data, which makes it the perfect tool for exporting the outputs of the model. The Excel book that is generated contains the average waiting times of each attraction for every hour it is operating, so it uses the same format as the original spreadsheet that is input in the model but with the values for every hour instead of every five minutes.

To make the most of the data obtained in the simulations, Power BI project has been designed. Power BI can provide interactive but at the same time intuitive visualizations and it can easily connect to Excel spreadsheets. These visualizations can help the company easily visualize the data obtained and get crucial insights about the queues' behaviour and their waiting times throughout the day.

As it can be seen in figure 5, the dashboard has been designed to have just one page where information from all the attractions is displayed. There are charts related to both waiting times and the number of people in the queues. The visualizations also show data throughout the day and between the two parks that define the whole amusement park. Additionally, there's also a filter on the top right corner of the figure that can filter all the charts on the dashboard between this two parks.

Power BI also has the feature of filtering the charts depending on what you click in the dashboard. For example, it is possible to filter by one of the attractions or by a specific hour of the day by simply clicking on what you want to filter by. Once you click what you want to filter by, all the charts in the dashboard will automatically and instantly update, and by clicking again the filtering will be removed.

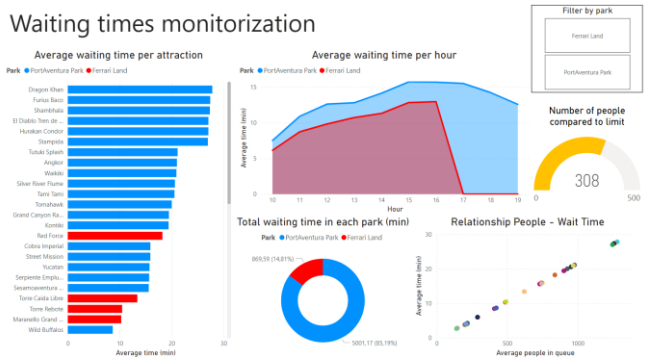

Figure 5: Power BI dashboard

6 Results

This section focuses on presenting and analysing the outcomes obtained throughout this project, going from the queuing model to the Power BI dashboard.

6.1 Results of the predictions and plots

As mentioned before, the results obtained at the moment in the queuing model do not represent the final and completed results of the project as there are a few more changes to implement to the model. I have designed four different input spreadsheets that have different numbers of people in the attraction queues in order to get more realistic simulations depending on the day of the week or the weather.

After clarifying that, the actual results of the model are really realistic as the waiting times are higher in the most popular attractions and lower in the least. The hour of the day is also an important variable, as in the first and last hours there's less people in the queues and, therefore, the waiting times are shorter. This can be seen in figure 6, which is the output for one of the executions of the queuing model coloured depending on the waiting times.

AttractionName	Park	10	11	12	13	14	15	16	17	18	19
Angkor	PortAventura Park	11,72	17,01	19,24	21,25	22,62	25,55	22,13	26,71	22,44	21,22
Armadillos	PortAventura Park	2,37	3,7	4,14	4,42	4	4,38	5,32	5,44	4,16	3,71
Buffalo Rodeo	PortAventura Park	2,59	2,99	3,59	3,97	4,18	5,32	5,27	5,44	4,05	3,92
Canoes	PortAventura Park	5,09	6,96	8,54	7,36	9,01	9,98	10,2	9,41	9,45	8,34
Carousel	PortAventura Park	1,89	3,72	3,6	3,5	4,93	4,91	5,55	4,54	4,1	3,35
Cobra Imperial	PortAventura Park	8,17	12,73	14,27	14,23	18,53	19,9	20,52	18,94	16,38	15,95
Coco Piloto	PortAventura Park	2,6	2,9	4,05	3,64	4,23	5,03	4,53	4,4	4,28	3,69
Crazy Barrels	PortAventura Park	2,24	3,27	3,48	4,33	4,62	5,45	5,25	4,76	4,66	3,8
Dragon Khan	PortAventura Park	16,85	24,82	26,25	28,17	27,63	29,56	34,64	33,18	31,15	27,14
El Diablo Tren de la Mina	PortAventura Park	16,3	21,57	26,74	26,91	27,98	31,02	31,07	33,64	31,11	24,79
El Salto de Blas	PortAventura Park	5,29	7,68	7,41	8,19	8,94	11,15	9,99	9,43	8,97	9,13
Estacio del Nord	PortAventura Park	2,13	2,92	4	3,23	4,02	5,14	4,53	4,61	4,04	4,26
Furius Baco	PortAventura Park	16,42	23,87	27,77	24,31	26,78	33,2	36,78	29,5	32,34	24,68
Grand Canyon Rapids	PortAventura Park	11,63	15,42	18,49	17,47	21,39	22,18	23,56	23,54	20,91	20,34
Hurakan Condor	PortAventura Park	14,73	20,54	25,14	24,94	31,58	36,38	29,99	34,67	29,87	23,79
Kiddie Dragons	PortAventura Park	1,93	3,27	3,35	4,22	3,76	5,43	4,69	5,08	4,62	4,07
Kontiki	PortAventura Park	12,39	15,63	20,39	18,1	19,97	21,7	22,08	26,43	19,84	17,28
La Granja de Elmo	PortAventura Park	1,86	3,41	3,81	4,22	4,97	4,66	4,51	4,27	4,79	4,16
Los Potrillos	PortAventura Park	2,31	3,3	3,42	3,49	4,43	5,24	4,71	4,82	4,41	3,54
Magic Fish	PortAventura Park	2,11	2,46	3,18	3,2	4,03	4,85	4,61	4,36	4,28	3,62
Mariposas Saltarinas	PortAventura Park	2,13	3,18	3,89	3,78	3,9	4,72	4,97	4,73	3,76	3,42
Penitence Station	PortAventura Park	2,48	2,86	4,14	4,13	4,23	5,17	4,28	5,47	3,85	3,59
Port de la Drassana	PortAventura Park	2,33	3,17	3,62	4,35	4,17	4,5	5,06	4,84	5	3,94
Serpiente Emplumada	PortAventura Park	8,06	13,31	15,97	15,72	16,74	19,57	17,1	18	16,44	16,47
Sesamoaventura Station	PortAventura Park	8,55	13,11	15,36	15,25	16,34	16,9	19,28	17,68	18,8	14,39
Shambhala	PortAventura Park	15,59	24,5	28,88	25,27	30,36	30,52	29,83	30,26	28,14	28,7
Silver River Flume	PortAventura Park	11,75	17,98	20,1	20,81	22,59	24,12	23,2	25,77	21,07	19,04
Stampida	PortAventura Park	15,99	25,08	25,25	28,23	26,91	29,79	33,77	33,78	26,98	25,79
Street Mission	PortAventura Park	8,19	13,24	15,7	14,39	18,45	18,01	19,95	17,81	17,11	14,88
Tami Tami	PortAventura Park	10,54	16,22	18,81	20,96	24,42	26,79	23,22	23,34	23,04	18,28
Tea Cups	PortAventura Park	2,2	2,84	4,25	3,84	4,35	5,02	4,66	4,38	4,47	3,76
Tomahawk	PortAventura Park	11,27	15,85	19,92	20,86	21,24	23,5	23,83	23,1	23,14	17,27
Tutuki Splash	PortAventura Park	11,6	15,43	20,43	21,59	23,6	26,5	24,15	24,22	22,63	21,08
Volpaiute	PortAventura Park	2,27	3,8	4,13	3,31	4,18	4,34	5,01	5,51	4,08	3,4
Waikiki	PortAventura Park	13,02	17,07	17,57	21,46	20,13	26,71	27,09	22,9	22,42	20,73
Waitan Port	PortAventura Park	4,36	7,15	8,43	8,12	9,57	9,97	10,93	10,35	9,26	8,65
Wild Buffalos	PortAventura Park	5,37	6,65	7,48	9,1	10,16	10,47	10,49	10,25	8,69	8,1
Yucatan	PortAventura Park	9,88	14,27	13,85	16,46	16,39	19,56	19,75	16,66	16,21	14,99

Figure 6: Output spreadsheet of the model

As the actual data is not real and it had to be created synthetically, it is evident that the waiting times have a big dependency on the attraction, as the best rides always have the longest queues. On the other hand, this is actually what happens in real parks, as there are a few attractions where everybody wants to ride and are always full.

If we take a closer look at the numbers, we can see that they are all between approximately 2 and 35 (minutes). This would not be a really packed day at the park, while it wouldn't also be the emptiest one. So, once I implement the four models mentioned before, these would be the results of a weekend as there could be simulations with a few more and a lot less people.

Now, let's take a closer look at one of these simulations; specifically at one that had 142 people waiting in that attraction and hour. In this case the number of people waiting maintains in approximately 60 users along most of the simulation, until all the people have joined the queue and the remaining rides are completed, where we can see that the number of people waiting is constantly getting lower.

This could represent that there's a limit where people would consider that the queue is too long and it's not worth it to wait for that much time, so they go to other attractions or spend their time somewhere else along the park. These insights will be really useful later on for the proposal of alternatives and final conclusions.

Figure 7: Gantt chart of a simulation

If we move on to figure 8, which is the last plot made with Python's Matplotlib, we can see the comparisons between the most and least popular attractions in the simulations. The first plot represents the one with least people, the last one is the one with the most people, and the plot in the middle represents the attraction that is closer to the average waiting time of all the simulations made.

We can see in colour red that the waiting times go from almost two minutes all the way to more than half an hour in this case, having the average in about 11 minutes. These

values can give us some ideas of if that day was really busy for the employees or if it was a quiet day but, as commented before, this will be clearer in the last delivery of the project once the four models are implemented in the queuing system.

As these plots are histograms, the bars represent the number of times each value in the x axis has appeared in the data. So, it is really obvious that the most repeated waiting times in these simulations are the higher ones. This can also be seen in figure 7 where, as mentioned before, the maximum number of people waiting at the same time is repeated through most of the simulation. This means that the arrival rate is really similar to the departure rate, or the rate in which people leave the queue to get in the rides. Once again, this demonstrates that it is possible to define a limit where people no longer feel like the attraction is worth all the wait time.

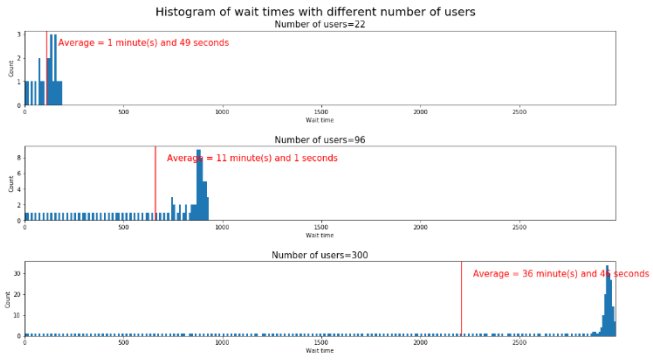

Figure 8: Histogram of waiting times

6.2 Results of the dashboard

Last but not least, we will take a look at the results and visualizations of the Power BI dashboard. Starting with the first plot, this is probably the one that gives the best information to the company as it gives the average waiting times through all the simulations of each attraction. This chart basically tells which are the most popular attractions at the moment, colouring them based on which park they are from. It will be seen in the next plots that this colouring is applicable to most of them.

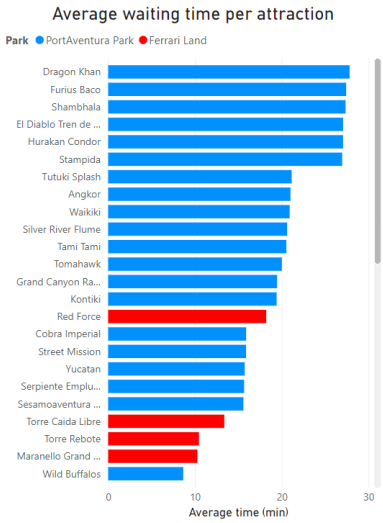

Figure 9: Dashboard's column chart

The following plot also visualizes the average waiting times, but this time in relation to the hour of the day. We can see that there is one park that has higher waiting times on average, mainly because it is the principal park and has the most well-known rides. We can also appreciate that the waiting times drop to zero for the red park once it's 17:00. The reason to this is that that park closes at this time and all the data goes to zero at that moment. There's not a lot more to see in this figure except for the detail that the longest queues tend to form between 15:00 and 17:00. This is really valuable information for the park as they can plan their resources and personnel's schedule based on this chart.

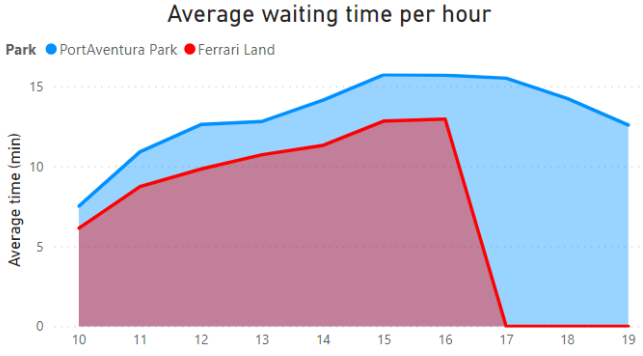

Figure 10: Dashboard's area chart

Moving on to the next chart, this one is really simple as it shows the total number of minutes that people have spent their time waiting in a queue for all the simulations. This visualization gives us information about the amount of people who go to each of the parks and it can also be useful for managing the number of employees in each park.

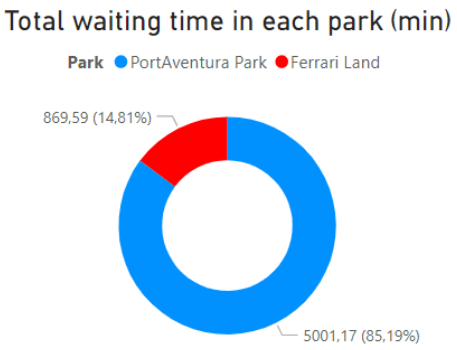

Figure 11: Dashboard's doughnut chart

In figure 12 the relation between the number of people and the waiting time is displayed. At the moment it is obvious that the relationships follow a really straight line. This is due to the fact that the data used in the simulations is synthetic data. If it was real data from the amusement park the dots in the figure would still be in a straight-line-like distribution but it wouldn't be that obvious.

By looking at the lines of the grid we can say that, approximately, for every 500 people who get in a queue there are 10 minutes of waiting time. This relationship seems pretty similar to reality as these are numbers you would expect in a park's queue.

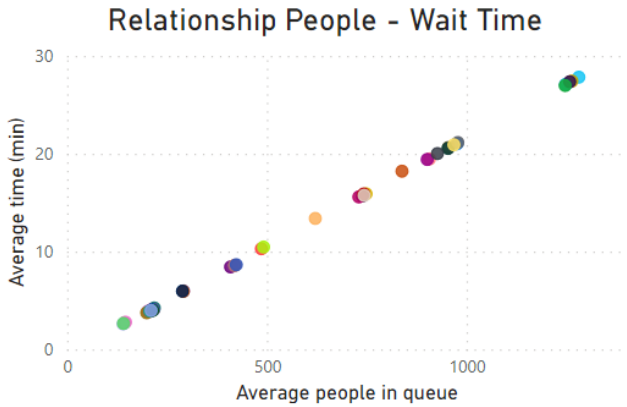

Figure 12: Relationship scatter plot

The last chart represents the average number of people waiting in the queues compared to a previously specified limit. This limit is not defined yet, but in the future the park will define a limit for each attraction. Once the number of people in a queue is higher than the limit defined the company thinks that there are too many people in that queue and they need to do something to prevent it from rising even more.

This visualization is directly related to the next steps as the alarms will be activated once the limit defined is surpassed and the proposals of conclusions may give ideas of how to act once an alarm goes off.

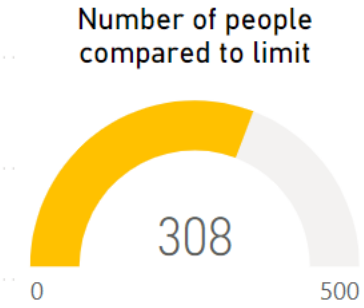

Figure 13: Dashboard's gauge chart

7 Provisional conclusions

The results obtained at the moment may not be the same as the ones that will be identified in the final delivery because there are still some more steps to be done before completing the project. But taking a look at all the results and outcomes obtained, there are definitely some conclusions that can be identified.

The model works. It is difficult to know the accuracy of the results because there is no real data to compare it to, but it definitely works. It can correctly simulate queues where each user gets in a queue, waits until it is attended and then leaves. This was something that looked really hard for me to implement before starting this project, considering that it was my first time using a simulation library and doing any project related to this topic. Additionally, I have made both the input data and the simulations as realistic as possible in order to obtain the most precise outputs. In order to do so I've used different service times and served a different amount of people at the same time for each attraction. I've always tried to use the closest numbers to reality, even when I didn't know the real values, and I think I did pretty good based on the outputs from the model.

On the other hand, the model can definitely be improved. First of all, by using the ideal values for the parameters such as the arrival rate. I have tried different values for this parameter but I'm sure it can still be improved in order to get more accurate results. Furthermore, being able to use real data and compare the simulation outputs with waiting times from previous days in the real park would definitely help me get a lot better results and find the ideal values for the parameters in the model, but after trying it a few times I still don't have access to these data.

I also think that if I had real data from the beginning of the project, I could have identified other ways to move forward in the project and more challenges to try and overcome. Additionally, I am sure that these project results wouldn't have the same importance for the company if

they were obtained using their data, as the results I'm obtaining at the moment don't necessarily apply to their park. Even so, as I commented previously, I always tried to get the most precise and realistic results and I think I kind of accomplished it.

Carrying on with the conclusions, thanks to the Matplotlib plots (figure 7 and figure 8) I have seen that there is a limit in each attraction where people would no longer feel like the attraction is worth all the wait time. This is a really important insight for the company as it would set a unique limit for each attraction that they could monitor in the future using real-time data, the Power BI dashboard, and alarms, taking different decisions once an attraction is about to surpass this limit. This limits may vary depending on the number of people in the park on that day, but this hasn't been tested yet as the four models haven't been implemented in the queuing system at the moment.

8 Next Steps

The tasks presented so far represent most of the project, but there's still a few more things to do before completing all the steps proposed initially. As seen in figure 1, the creation of alarms and proposal of alternatives to the actual queue management are still pending.

But, as commented in the previous section, the first thing to do consists of implementing the four models with different numbers of people in the queues as it will give a lot more value to the comparisons between simulations with more and less people.

Additionally, the creation of alarms and connection of them with the Power BI dashboard will be able to give real-time feedback to the park's managers and, therefore, give them the opportunity to make crucial decisions before long queues emerge in the busiest attractions. Unfortunately, these alarms will not be really useful to the park at the moment because right now they don't have the equipment necessary to work with real-time data, but they expect to be able to do so in the future and these alarms will be really useful then.

Last but not least, by presenting different alternatives to the current queue management in the park, we can identify more efficient and user-friendly ways to improve the overall experience of the amusement park.

References

[1] M/M/1 queuing system (2021). EventHelix. [consulted: March 6, 2023]. Available on the Internet: https://www.eventhelix.com/congestion-control/m-m-1/

[2] SimPy overview (2016). Team SimPy. [consulted: March 5, 2023]. Available on the Internet: https://simpy.readthedocs.io/en/latest/

[3] Introduction to simulation with SimPy (2022). Darío Weitz. [consulted: March 12, 2023]. Available on the Internet: https://towardsdatascience.com/introduction-to-simulation-with-simpy-322606d4ba0c

[4] Matplotlib (2012). The Matplotlib development team. [consulted: April 6, 2023]. Available on the Internet: https://matplotlib.org/

[5] Power BI (2015). Microsoft. [consulted: March 12, 2023]. Available on the Internet: https://powerbi.microsoft.com/en-us/

[6] Virtual queuing in amusement parks (2021). Attractions.io. [consulted: March 8, 2023]. Available on the Internet: https://attractions.io/learn/virtual-queuing-the-future-of-theme-park-experiences

[7] Adapting Agile Scrum methodology for individuals (2017). Lucidchart. [consulted: March 8, 2023]. Available on the Internet: https://www.lucidchart.com/blog/scrum-for-one

[8] Microsoft Teams (2020). Microsoft. [consulted: March 12, 2023]. Available on the Internet: https://www.microsoft.com/en-us/microsoft-teams/group-chat-software

[9] Pandas documentation (2023). Pandas team. [consulted: April 7, 2023]. Available on the Internet: https://pandas.pydata.org/docs/

[10] NumPy documentation (2020). NumPy developers. [consulted: April 7, 2023]. Available on the Internet: https://numpy.org/doc/stable/

[11] Gantt charts with Python's Matplotlib (2021). Thiago Carvalho. [consulted: April 2, 2023]. Available on the Internet: https://towardsdatascience.com/gantt-charts-with-pythons-matplotlib-395b7af72d72