Matthew Russell

Python

May 20, 2019

Python Project: Disney Attractions Wait Times

**Abstract**

Walt Disney World is the most visited vacation resort in the world. According to Magic Guides, the average annual attendance at Walt Disney World is over 52 million visitors ever year. Many guests visit Walt Disney World for its innovative attractions. I recently completed the Fall 2018- Disney College Program working in attractions, and I personally witnessed guests’ willingness to wait hours to ride on their favorite attractions. I noticed that guests seemed happier and the mood in the parks was elevated when attraction wait times were shorter. Cast members want guests to have a great experience, but long lines can be frustrating for guests. Wait times at attractions change from day to day, hour to hour, but I believe that guests would find it helpful to know when they are most likely to find the shortest lines in the parks so they could enjoy their favorite rides without waiting extra time in lines.

**Question**

Based on historical data of attraction wait times, I will try to determine what calendar days the wait times at Disney World attractions are likely to be the shortest as well as what time of day will yield the shortest wait time for different attractions.

**Data Source**

The Walt Disney World Ride Wait Time Datasets I used were provided online by touringplans. touringplans accesses wait times directly from Walt Disney World and provides them for research purposes. The datasets were comma-separated files (CSV) of ASCII text. Each attraction provided approximately 250,000 rows by 4 columns of data for analysis. I utilized the attraction wait time data sets over 5 years for eight specific attractions, two attractions in each of Walt Disney World’s parks, specifically: Expedition Everest, Dinosaurs, Pirates of the Caribbean, Seven Dwarfs Mine Train, Soarin’, Spaceship Earth, Rock ‘n’ Rollercoaster, and Toy Story Mania.

**Description of Analysis**

The first program analyzes the average wait time for a given attraction for each day in the years 2014 through 2018. The program takes in a specific rides csv file and given a defined start date (1/1/2014) and end date (1/1/2019), it jumps to the specific row in the csv file with that start date. It appends the wait times into a day array until the date changes and then averages that array. That average is then associated with that date. Each average is appended to a year array which stores 365 averages which coincides with the date of the year. It then moves on to the next date until it reaches a new year and then it plots that year. The program keeps going until it reaches the specified end date. Five separate plot years can be viewed independently or as a group for comparison for the specified attraction.

The second program analyzes the average wait time each hour for a specific attraction for a any given date in the years 2014 through 2018. The program takes in a specific ride csv file for the date selected through the period 1/1/2014 to 1/1/2019. At the start of the program, it jumps to the first occurrence of that date after the start date. It then appends every wait time within each hour to an hour array and then averages them. The program generates an average wait time for each hour. When the hour changes it appends the average to a day array and continues until the day ends. After that it plots that day of the year into a line graph and then tries to find the next occurrence of that date in the next year. The program keeps going until it reaches the end date given.

**Code**

My first consideration in developing the code was how to consolidate 250,000 rows of data in each file. The data set provided a varying number of wait times (between 150 to 500) per day. I made the decision to have the program average the wait times for a given day to be able to plot 365 points per year. Another consideration was how to get the program to know what the day, year, and hour were in order for me to always know where I was in the data set. To do this I needed to find a way to know when a hour, day or year changed so that I could perform an action. I had to code the program to grab the data from the cell and then split the string to grab the year, date, and hour. Another important consideration was that any given park day may not have actually ended at midnight when the park remained open past midnight. I recognized that the program could not use the column one date but needed to look in column zero to find the specific park operational date in order to graph the data appropriately.

**Attraction Wait Times Over A Year (Program 1)**

import pygal

import matplotlib.pyplot as plt

import csv

import statistics

import pygal.style

hist = pygal.Line(height=500,

width=1000,

dots\_size=2,

x\_label\_rotation=30,

show\_minor\_x\_labels=False,

show\_x\_guides=True,

style=pygal.style.DarkStyle)

Name = " Soarin"

#Soarin

#Pirates of the Caribbean

#Expedition Everest

#Rock n Rollercoaster

#Spaceship Earth

#Seven Dwarfs Mine Train

#Dinosaur

#Toy Story Mania

#Kilimanjaro Safaris

#Avatar Flight of Passage

#Splash Mountain

#Slinky Dog Dash

File = Name+".csv"

RowStart = "1/1/2014"

RowCap = "1/1/2019"

path = "PATH OF THE FILE"+File

file = open(path,newline='')

reader = csv.reader(file)

AverageFile = open("Average.csv","w+")

AverageFile.write("Date,Average Wait Time\n")

header = next(reader)

data = [row for row in reader]

#Find First Row Number

Row = 0

while True:

if data[Row][0] == RowStart:

break

Row = Row + 1

Day = data[Row][0]

Year = Day[data[Row][0].\_\_len\_\_()-4:data[Row][0].\_\_len\_\_()]

DayArray = []

DateArray = []

YearArray = []

while True:

if data[Row][0] == RowCap:

break

while True:

if data[Row][0] == Day:

if not data[Row][2]== "":

try:

DayArray.append(float(data[Row][2]))

except ValueError:

continue

Row = Row + 1

else:

break

if DayArray.\_\_len\_\_() != 0:

#Print to File to Test

Average = statistics.mean(DayArray)

YearArray.append(Average)

DateArray.append(Day[0:Day.\_\_len\_\_()-5])

AverageFile.write(Day)

AverageFile.write(",")

AverageFile.write(str(Average))

AverageFile.write("\n")

else:

Average = 0

if (data[Row][0])[data[Row][0].\_\_len\_\_()-4:data[Row][0].\_\_len\_\_()] != Year:

#Grpah after each year

hist.title = Name

hist.y\_title="Wait Time in Minutes"

hist.x\_title="Date"

hist.x\_labels = DateArray

hist.x\_labels\_major = ["1/1","1/15","2/1","2/15","3/1","3/15","4/1","4/15","5/1","5/15","6/1","6/15","7/1","7/15","8/1","8/15","9/1","9/15","10/1","10/15","11/1","11/15","12/1","12/15","12/31"]

hist.add(Year,YearArray)

hist.render\_to\_file(Name+'.svg')

YearArray = []

DateArray = []

DayArray = []

Day = data[Row][0]

Year = Day[data[Row][0].\_\_len\_\_()-4:data[Row][0].\_\_len\_\_()]

Average=0

AverageFile.close()

file.close()

print("OK.")

**Attraction Wait Times Over A Day (Program 2)**

import pygal

import matplotlib.pyplot as plt

import csv

import statistics

import pygal.style

hist = pygal.Line(height=500,

width=1000,

dots\_size=2,

x\_label\_rotation=30,

show\_minor\_x\_labels=False,

show\_x\_guides=True,

style=pygal.style.DarkStyle)

DatePicked = "12/31"

Name = " Soarin"

# Soarin

# Pirates of the Caribbean

# Expedition Everest

# Rock n Rollercoaster

# Spaceship Earth

# Seven Dwarfs Mine Train

# Dinosaur

# Toy Story Mania

#Avatar Flight of Passage

#Splash Mountain

#Slinky Dog Dash

#Kilimanjaro Safaris

File = Name + ".csv"

RowStart = "1/1/2014"

RowCap = "1/1/2019"

path = " PATH OF THE FILE " + File

file = open(path, newline='')

reader = csv.reader(file)

AverageFile = open("Average.csv", "w+")

AverageFile.write("Date,Average Wait Time\n")

header = next(reader)

data = [row for row in reader]

# Find First Row Number

Row = 0

while True:

if data[Row][0] == RowStart:

break

Row = Row + 1

Day = data[Row][0]

Year = Day[data[Row][0].\_\_len\_\_() - 4:data[Row][0].\_\_len\_\_()]

while True:

if data[Row][0] == DatePicked + "/" + Year:

break

Row = Row + 1

Day = data[Row][0]

Hour = (data[Row][1])[data[Row][1].\_\_len\_\_() - 5:data[Row][1].\_\_len\_\_() - 3]

HourArray = []

HourofdayArray = []

DayArray = []

DateArray = []

YearArray = []

while True:

if data[Row][0] == RowCap:

break

while True:

if data[Row][0] == DatePicked + "/" + Year:

if (data[Row][1])[data[Row][1].\_\_len\_\_() - 5:data[Row][1].\_\_len\_\_() - 3] == Hour:

if not data[Row][2] == "":

try:

HourArray.append(float(data[Row][2]))

except ValueError:

continue

Row = Row + 1

else:

break

else:

Row = Row + 1

break

try:

if data[Row][0] == RowCap:

break

except IndexError:

break

if HourArray.\_\_len\_\_() != 0:

# Print to File to Test

Average = statistics.mean(HourArray)

YearArray.append(Average)

DateArray.append(Day[0:Day.\_\_len\_\_() - 5])

HourofdayArray.append(Hour)

AverageFile.write(Day)

AverageFile.write(",")

AverageFile.write(str(Average))

AverageFile.write("\n")

#print(Hour)

#print(HourArray)

else:

Average = 0

if data[Row][0] != Day and HourArray.\_\_len\_\_() != 0:

# Grpah after each year

j = 5

while True:

if int(HourofdayArray[0]) != j:

YearArray.insert(0,None)

else:

break

j = j + 1

print(HourofdayArray)

print(YearArray)

print("")

#Grpah after each year

hist.title = Name+" on "+DatePicked

hist.y\_title="Wait Time in Minutes"

hist.x\_title="Hour of the Day"

hist.x\_labels = ["5","6","7","8","9","10","11","12","13","14","15","16","17","18","19","20","21","22","23","0","1","2","3","4"]

hist.x\_labels\_major = ["5","6","7","8","9","10","11","12","13","14","15","16","17","18","19","20","21","22","23","0","1","2","3","4"]

hist.add(Year,YearArray)

FileName = Name+" Day"

hist.render\_to\_file(FileName+'.svg')

YearArray = []

DateArray = []

HourofdayArray = []

DayArray = []

HourArray = []

Hour = (data[Row][1])[data[Row][1].\_\_len\_\_() - 5:data[Row][1].\_\_len\_\_() - 3]

Day = data[Row][0]

Year = Day[data[Row][0].\_\_len\_\_() - 4:data[Row][0].\_\_len\_\_()]

Average = 0

#print(YearArray)

AverageFile.close()

file.close()

print("OK.")

**Description of Graphs**

**Attraction Wait Times Over A Year (Program 1)**

To generate the graph, the program creates an array of all the averages for each day of the year which it graphs at the years end. The dates where added to an array as they were found within the csv file and added to the graph. Different sets of data were assigned different colors to show the data for each of the five years within one graph. I chose a line graph because it displayed the large amount of data well. The ability to toggle the years on and off helps to make each graph even more readable. I also played with the appearance of the graphs increasing the width, decreasing the dot size and changing the colors to enhance the graphs readability. The main challenge I faced in portraying the data was that if the ride was down for the entire day, that day was completely skipped in the csv file and had to be manually added or years would be missing days and look shorter on the graph.

**Attraction Wait Times Over A Day (Program 2)**

To generate this graph, the program creates an array of all the averages for each hour of a chosen date. The array for that specific date is added to the graph at the end of each year as the program read through the csv file. The hours are added to the graph separately in order to keep the hours in terms of a specific park’s operational day (5am-4:59am) rather than a true day (12am-11:59pm). Because the data set used military time, the graphs are also in military time with midnight being zero and 23 being 11 o’clock pm. I decided to use the line graphs again for consistency in the presentation of the information and because it worked well visually.

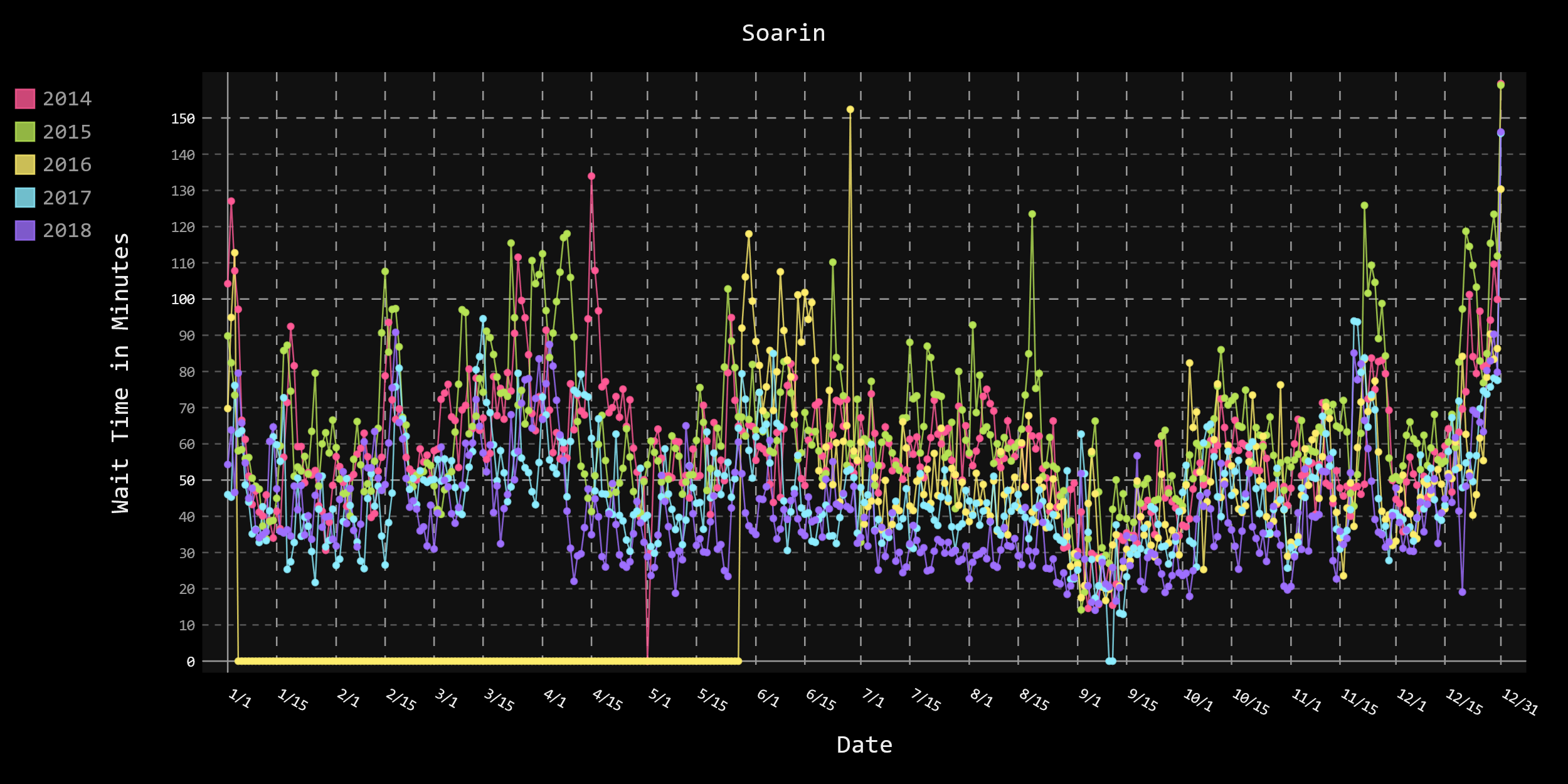
**Results**

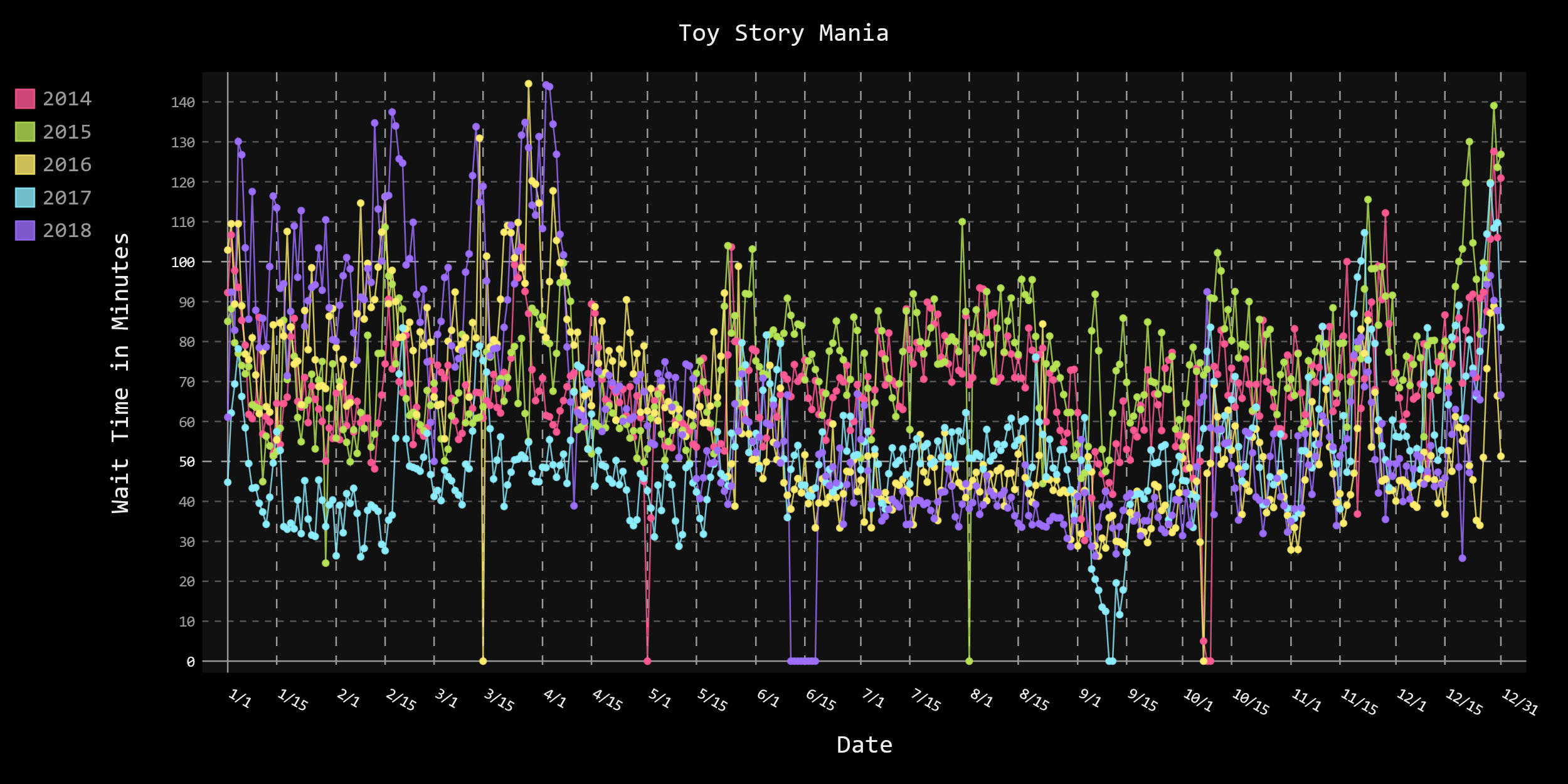
There seem to be many interesting stories told in the data, but let’s look at the main questions first: *What calendar days are the wait times at Disney World attractions the shortest and what time of day will yield the shortest wait times at attractions?*  It appears, based on analyzing the data for the 8 attractions, that the best time of the year to go to Walt Disney World, if you want to avoid crowds and spend less time waiting in line, happens historically between September 1st and September 15th (Graph 1 and 2). Upon graphing the data for all 8 attractions, and accounting for major discrepancies, the data showed that the lowest wait times happened consistently during that period of time, over the five years, across the attractions. One important issue to account for when reading the graphs was weekends. Weekend days typically showed longer average wait times and dates for weekends shift over time (Graph 3). Discrepancies also showed up for issues like shutting down for hurricanes, with the rides going down for maintenance, with the addition of additional tracks or nearby attractions, etc.

A question that I didn’t ask but that did show up in the data was: *When is the worse time to go to Disney to ride the attractions?* The data reveals that visiting New Year’s Eve Day will have you waiting in the longest lines of the year. For example, the data for the September 1-15, show that the average wait time for Soarin’ was around 20 minutes, while the wait time on average for New Year’s Eve day was 140 minutes (Graph 1). That is a difference of 2 hours waiting in line for the same attraction. Once in the parks, it appears that the shortest wait times for attractions is the first few hours of opening and the last few hours before closing. The graphs also show somewhat shorter wait times around lunch and dinner and a decrease in wait times around scheduled parade times (Graph 3 and 4).

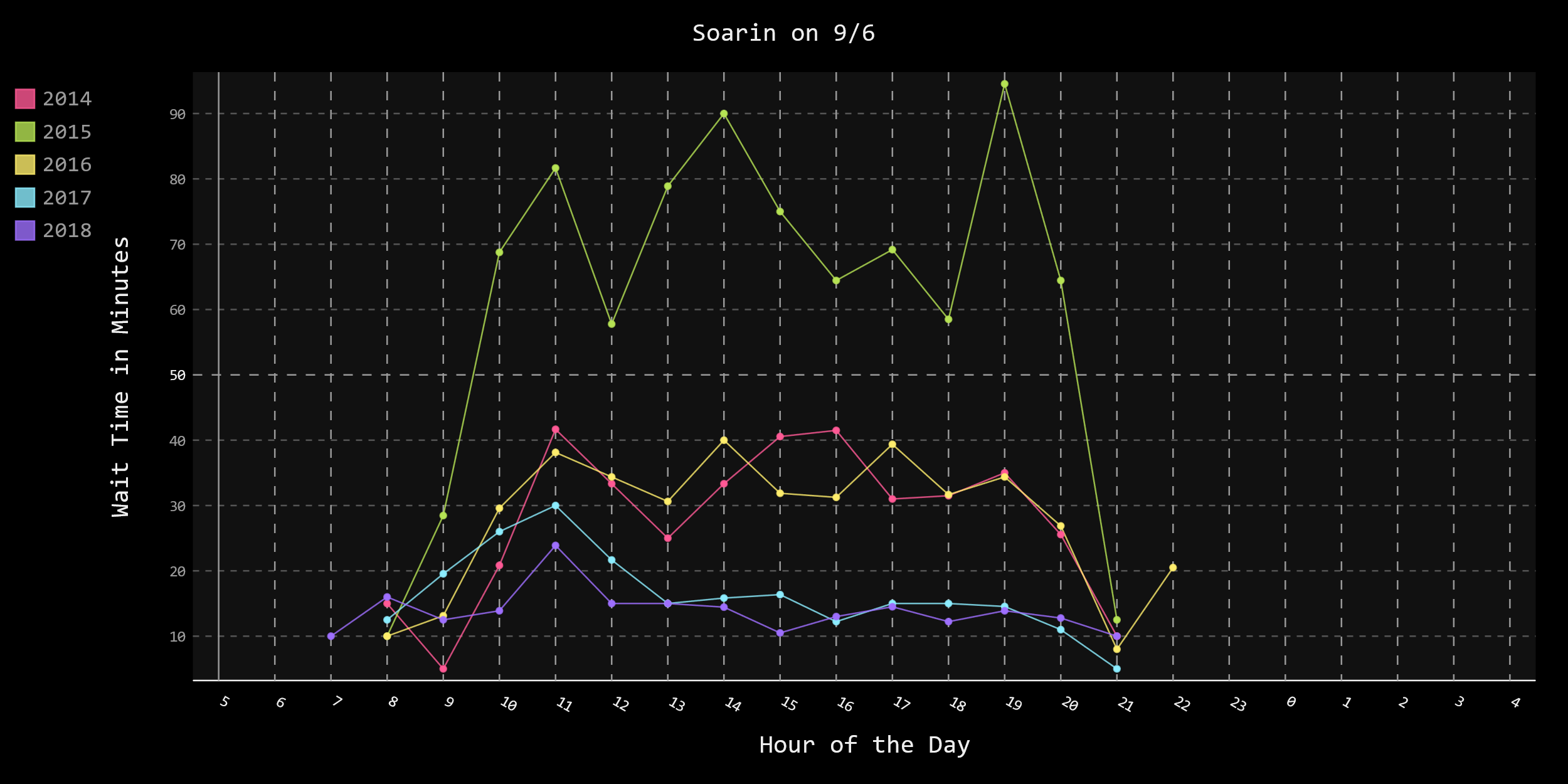
The Soarin’ graph is an interesting one. It shows that Soarin’ went down between Jan. 4th and May 27th in 2016. Following a spike in wait times for a month following the reopening, the average wait times then decreased from the previous years. In researching, it became clear that the reason for the decrease in wait times was because Disney had added a third bay to the ride. They were able to get more guests on the ride at one time so wait times went down. This same thing happened with the attraction, Toy Story Mania. In May 2016 Disney added an additional track to the attraction. This can be seen clearly within the graph for Toy Story Mania with a decrease in wait times following that addition. For example, looking at New Year’s Eve Day in 2015, the average wait time for Toy Story Mania was 125 minutes. Once the new track opened the average wait times for New Year’s Eve day were 50 in 2016, 80 minutes in 2017, and 65 minutes in 2018. The graph also reveals that Jan. 1, 2018 the third track was taken out of operation while Disney was reconfiguring the attraction’s entrance. The wait times returned to their previous 2 track levels until the 3rd track reopened on June 19th. In fact, the average wait time levels went to an all-time low following that date. Why would that happen? Research revealed that date to be the opening of the new Toy Storyland which added two new attractions. So, it appears that the new attractions helped to decrease the wait times at Toy Story Mania.

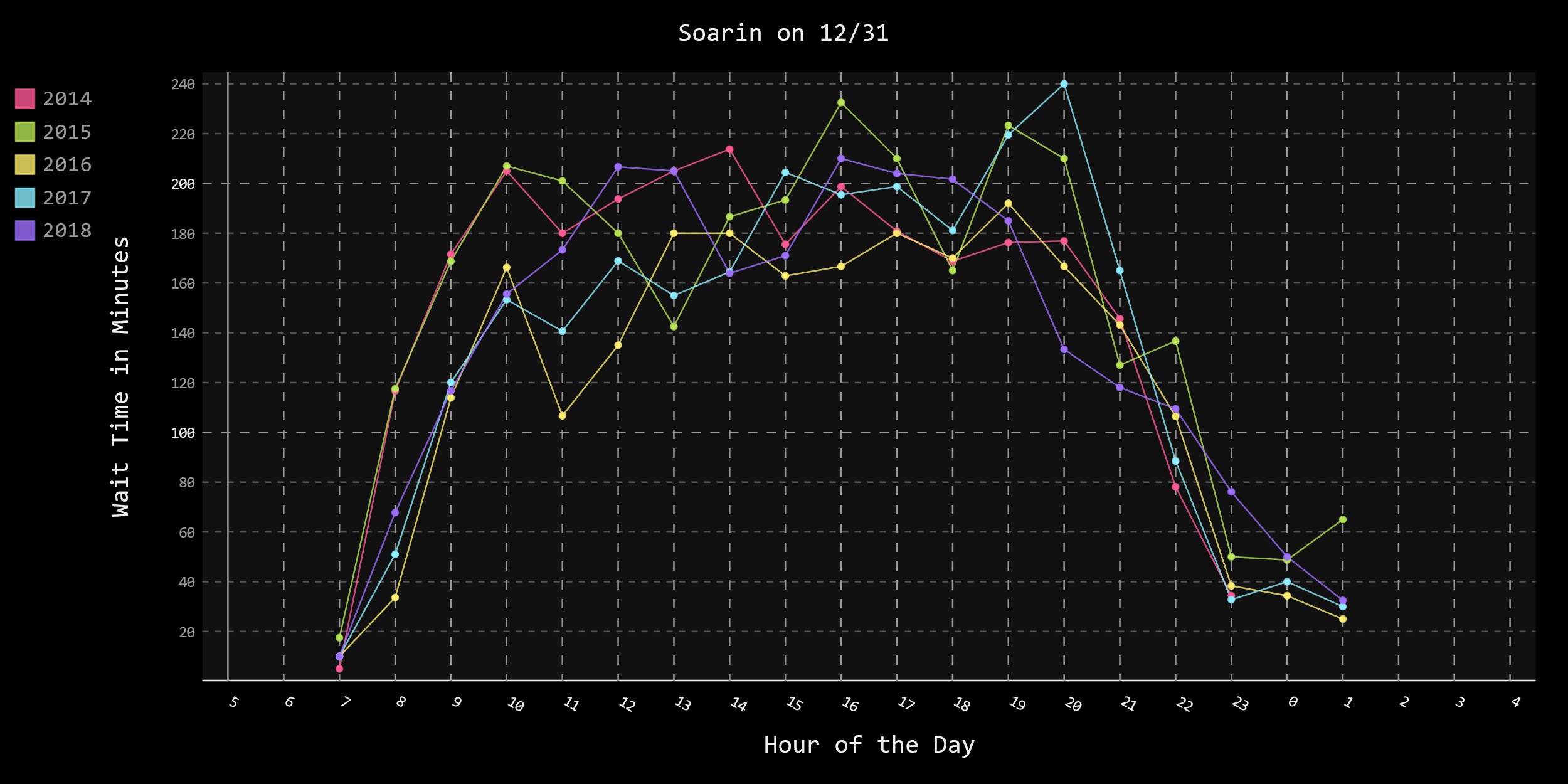
**Graph Examples (Attraction wait times over a year)**

**Graph 1**

**Graph 2**

**Graph Examples (Attraction wait times over a day)**

**Graph 3**

**Graph 4**

**Reflections**

I really enjoyed doing this project especially the research on why the patterns in the data showed some occasional discrepancies. Having worked attractions and observed how wait times affect the atmosphere, it was really interesting to analyze how wait times change throughout the day and year and to actually look into why. I was happy that the data revealed definitive results for the best time to visit the parks to avoid crowds. I noticed that the data yielded the same recommendations that can be found throughout Disney related sites on the internet for the best week to visit Disney in general. While working, I had personally witnessed that wait times were often the best in the final few hours before closing. I also know firsthand that for very popular rides like *Seven Dwarfs Mine Train* and *Flight of Passage* that getting in line at morning rope drop and just as the park is closing often gets you through the line quickest. So, it does seem that the data provided the results that I was expecting. The biggest challenge to overcome with this project was figuring out how to deal with some of the gaps in Disney’s data. It became obvious that something was off when some of the years didn’t show 365 days. I had to figure out what was causing the issue and then I had to tell the code how to work around it or I had to fill it in manually. It took some time to do this, but I was able to successfully accomplish it. I am returning to Disney next month for a Professional internship in Engineering and this project helped get me in the Disney mindset.

**Bibliography**

"avatar\_flight\_of\_passage.csv", Disney World Ride Wait Time Datasets, TouringPlans.com, January 2019, https://www.touringplans.com/walt-disney-world/crowd-calendar/#DataSets, Accessed 1 May 2019.

"dinosaur.csv", Disney World Ride Wait Time Datasets, TouringPlans.com, January 2019, https://www.touringplans.com/walt-disney-world/crowd-calendar/#DataSets, Accessed 1 May 2019.

"expedition\_everest.csv", Disney World Ride Wait Time Datasets, TouringPlans.com, January 2019, https://www.touringplans.com/walt-disney-world/crowd-calendar/#DataSets, Accessed 1 May 2019.

"kilimanjaro\_safari.csv", Disney World Ride Wait Time Datasets, TouringPlans.com, January 2019, https://www.touringplans.com/walt-disney-world/crowd-calendar/#DataSets, Accessed 1 May 2019.

"pirates\_of\_the\_caribbean.csv", Disney World Ride Wait Time Datasets, TouringPlans.com, January 2019, https://www.touringplans.com/walt-disney-world/crowd-calendar/#DataSets, Accessed 1 May 2019.

"rock\_n\_rollercoaster.csv", Disney World Ride Wait Time Datasets, TouringPlans.com, January 2019, https://www.touringplans.com/walt-disney-world/crowd-calendar/#DataSets, Accessed 1 May 2019.

"seven\_dwarfs\_mine\_train.csv", Disney World Ride Wait Time Datasets, TouringPlans.com, January 2019, https://www.touringplans.com/walt-disney-world/crowd-calendar/#DataSets, Accessed 1 May 2019.

"slinky\_dog\_dash.csv", Disney World Ride Wait Time Datasets, TouringPlans.com, January 2019, https://www.touringplans.com/walt-disney-world/crowd-calendar/#DataSets, Accessed 1 May 2019.

"soarin.csv", Disney World Ride Wait Time Datasets, TouringPlans.com, January 2019, https://www.touringplans.com/walt-disney-world/crowd-calendar/#DataSets, Accessed 1 May 2019.

"spaceship\_earth.csv", Disney World Ride Wait Time Datasets, TouringPlans.com, January 2019, https://www.touringplans.com/walt-disney-world/crowd-calendar/#DataSets, Accessed 1 May 2019.

"splash\_mountain.csv", Disney World Ride Wait Time Datasets, TouringPlans.com, January 2019, https://www.touringplans.com/walt-disney-world/crowd-calendar/#DataSets, Accessed 1 May 2019.

"toy\_story\_mania.csv", Disney World Ride Wait Time Datasets, TouringPlans.com, January 2019, https://www.touringplans.com/walt-disney-world/crowd-calendar/#DataSets, Accessed 1 May 2019.