Disney World Regression

Ashley Fidler

Narrative

Being a Disney fan (and always wanting to plan the next trip), I was interested in seeing when the shortest wait times would be for our favorite ride. Using data from touringplans.com, I was able to find a dataset on the Flight of Passage ride in Walt Disney World's Animal Kingdom. This dataset provides information on how long guests actually waited in line to take a ride on a banshee through the world of Pandora. The actual wait time is recorded around three times each day starting in May 2017 up to August 2021. In this project, I would like to explore how wait times fluctuate throughout the day, week, and year. I will focus on the wait times post-covid (Disney World closed its doors on March 13th 2020 and reopened them on July 11th 2020). After exploring which times (throughout the day, week, and year) result in the shortest line, I will create models that will help me predict how long I would have to wait in line for the Flight of Passage ride.

Original Dataset

Variable	Description
date	year-month-day at which the wait time was observed
datetime	the date and time at which the wait time was observed
SACTMIN	actual wait time for the ride
SPOSTMIN	the standby wait time posted by Disney World

Reformed Dataset

Variable	Description
day	day of the month of observation
time	time of day wait time was recorded
SACTMIN	actual wait time for the ride
weekday	day of the week (1 = Sunday, 7 = Saturday)
date	year-month-day at which the wait time was observed
year	the year at which the wait time was observed
month	the month at which the wait time was observed
precovid	TRUE if the data was taken before March 13th, 2020 ; FALSE if not
time_of_day	hour-long block of time in which the time lies
quarter	which quarter of the year the date is in

The dataset I found contained a lot of information. In order to work with it, I cleaned it up a bit. I removed any missing values, removed the SPOSTMIN column, made new columns for the month, day, and year (because working with dates as a whole turns out to be not so easy), and created a new column that would indicate whether the data was taken before Disney World closed its parks for COVID-19 or after it reopened. I also filtered the data to only allow SACTMIN to have a maximum of 720 minutes since the parks

are rarely open for more than 12 hours each day.

Cleaning the Data

```
flight_of_passage <- read_csv("flight_of_passage (1).csv")
```

```
## Rows: 143761 Columns: 4
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): date, datetime
## dbl (2): SACTMIN, SPOSTMIN
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
flight_of_passage <- flight_of_passage %>%
    filter(!is.na(SACTMIN))

flight_of_passage <- flight_of_passage[ , 2:3]

flight_of_passage <- separate(flight_of_passage, datetime, into = c("day", "time"), sep = " ")

flight_of_passage %<>%
    mutate(day = parse_date(day, "%m/%d/%Y"),
           time = parse_time(time, "%H:%M"),
           weekday = wday(day))

flight_of_passage <- flight_of_passage %>%
    filter(SACTMIN < 720)

flight_of_passage <- flight_of_passage %>%
    mutate(date = ymd(day)) %>%
    mutate_at(vars(day), funs(year,month,day))
```

```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

```
flight_of_passage <- flight_of_passage %>%
    mutate(pre_covid = date <= as.Date("2020-03-13"))
```

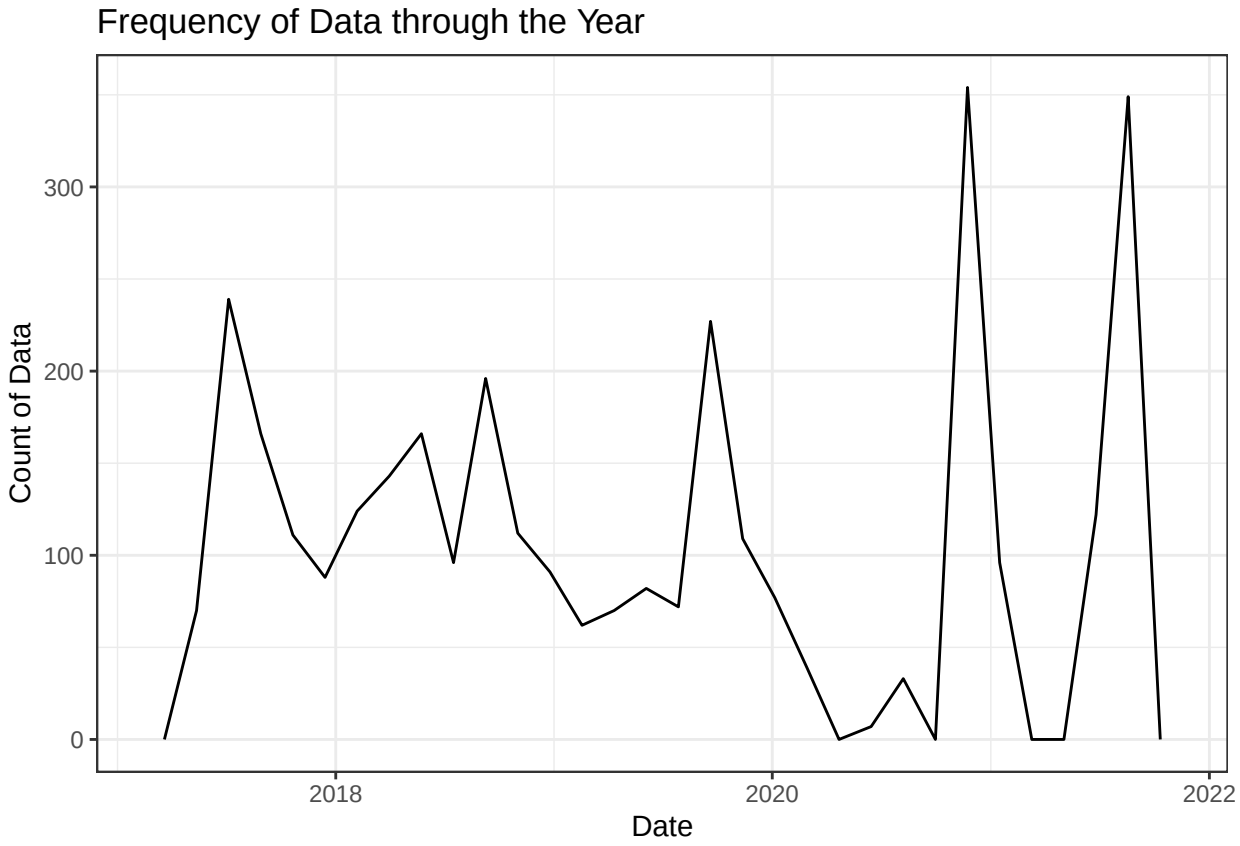
Exploratory Data Anlaysis

I first wanted to get an understanding of how much data I was now working with. To do this, I created frequency graphs that showed me how much data was taken over the years, throughout the week, and throughout each day.

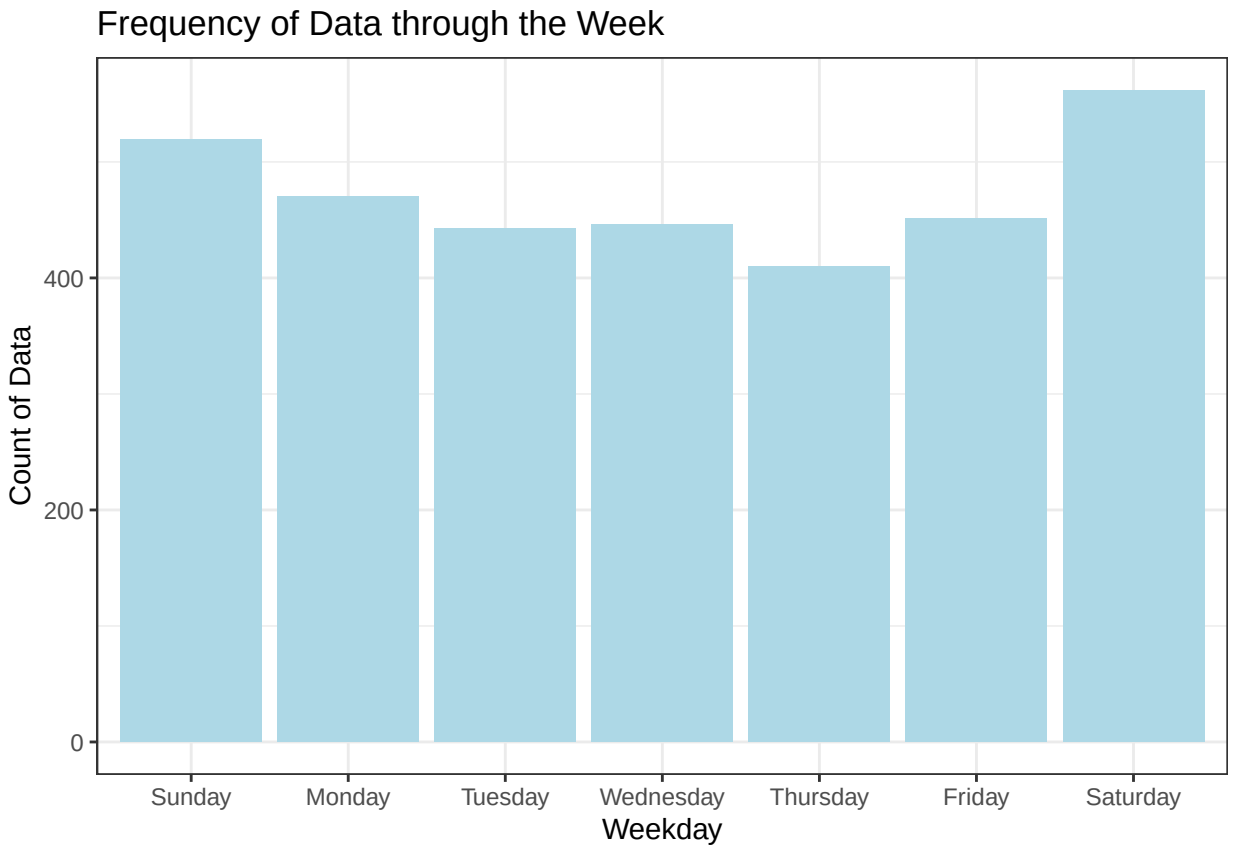
How Much Data Are We Working With?

```
#distribution among years
flight_of_passage %>%
    ggplot()+
    geom_freqpoly(aes(date))+
    ggtitle("Frequency of Data through the Year")+
    labs(x = "Date" , y="Count of Data")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

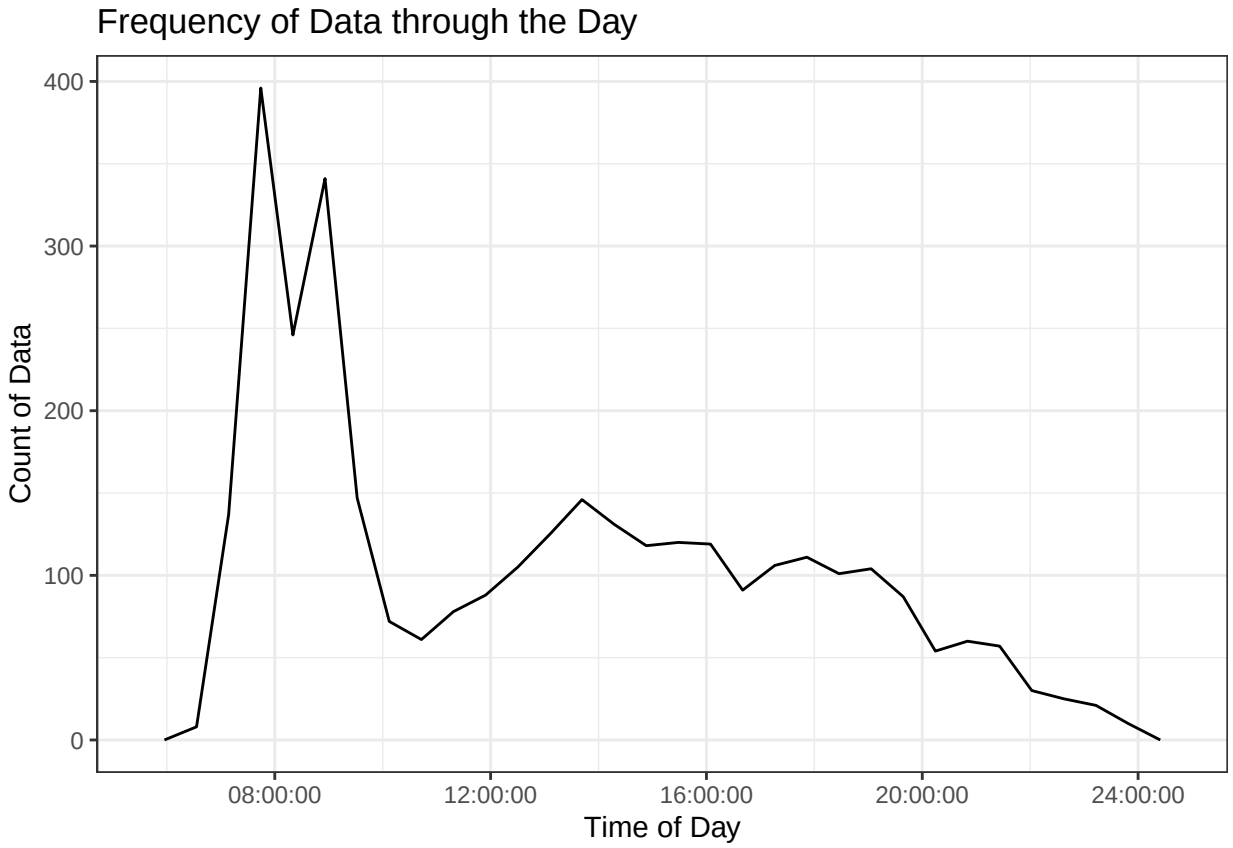



```
#distribution among weekdays
flight_of_passage %>%
    ggplot()+
    geom_bar(aes(weekday), fill = "lightblue")+
    ggtitle("Frequency of Data through the Week")+
    xlim("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday")+
    labs(x = "Weekday" , y="Count of Data")
```




```
#distribution from 6am to 12am
flight_of_passage %>%
    filter(hour(time) >= 06 & minute(time) >= 00) %>%
    ggplot() +
    geom_freqpoly(aes(x = time))+
    ggtitle("Frequency of Data through the Day")+
    labs(x = "Time of Day" , y="Count of Data")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



We notice that COVID-19 had a substantial impact on our data. On March 12, 2020, Disney World officially closed due to COVID. It only reopened on July 11th of 2020, so we have a significant amount of time where there is little to no data as we can see in the "Frequency of Data through the Year" graph. The "Frequency of Data through the Week" shows that there is less data taken during the week compared to the weekends. Thursday seems to have the least amount of data while Saturday has the most. This is unsurprising since weekends are expected to be much more busy and therefore more important/easier to collect wait time data. The third graph in this section, "Frequency of Data through the Day", shows us that much more data is taken as the parks open in the morning compared to when the parks close in the evening. There seems to be a lull before lunch, but sampling begins to pick back up again once everyone is fed!

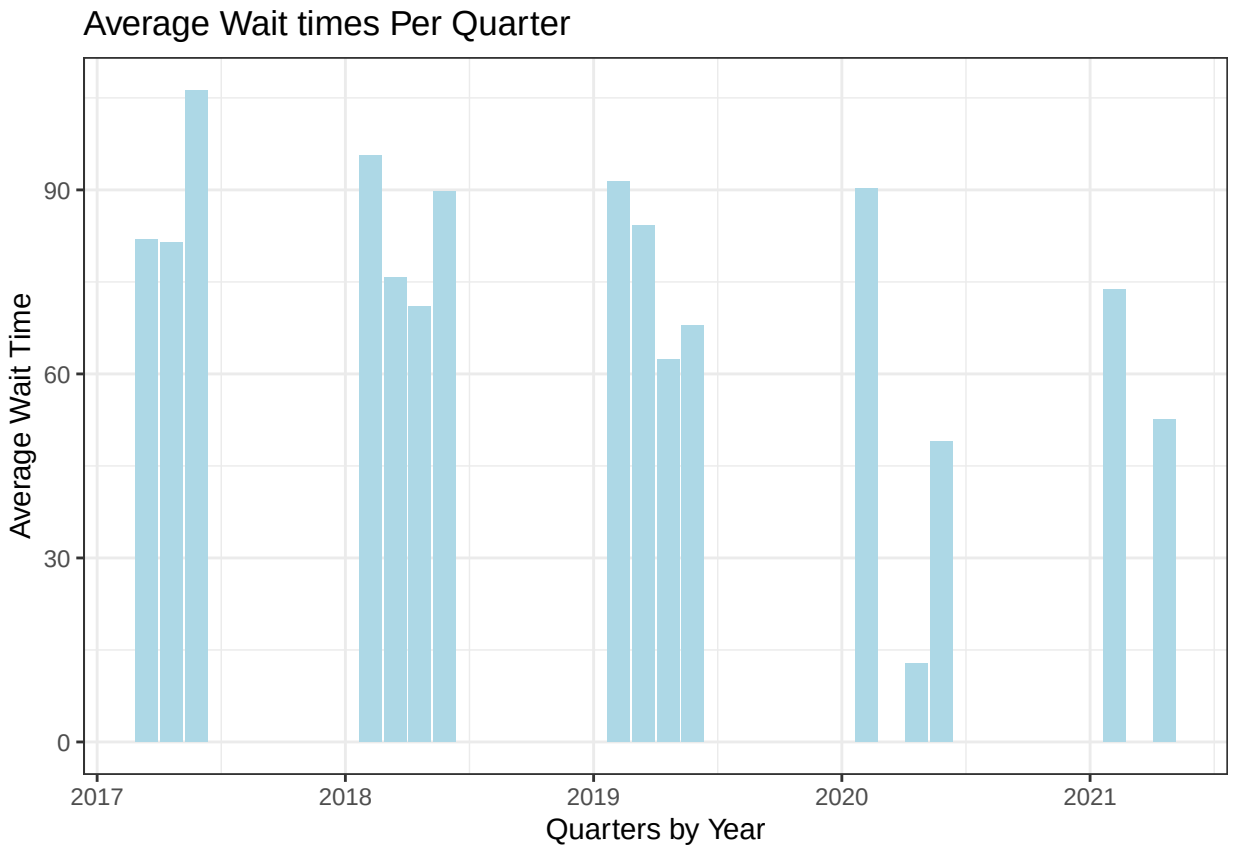
To continue this project, I will intermittently include summaries from before the closure for COVID and after reopening. For the integrity of the project, I really want to focus on present-day information. By focusing on the results from July 11th 2020 and onward, I can begin to predict what future wait times will be in a post-COVID world.

Average Wait Time During the Year

Now knowing what the data count looks like over the past several years, I wanted to focus on the relationship between wait time and year. To do this, I created quarters for each year (Quarter 1 being January, February and March, Quarter 2 being April, May and June, etc.) and then made a graph to show average wait times during those quarters from 2017 to 2021. I then wanted to see how the wait times compare between months, so I made a bar chart that shows the average wait time for each month both pre and post covid. To get a more broad picture, I made another plot that shows the average wait times for each day of the year once the parks reopened.

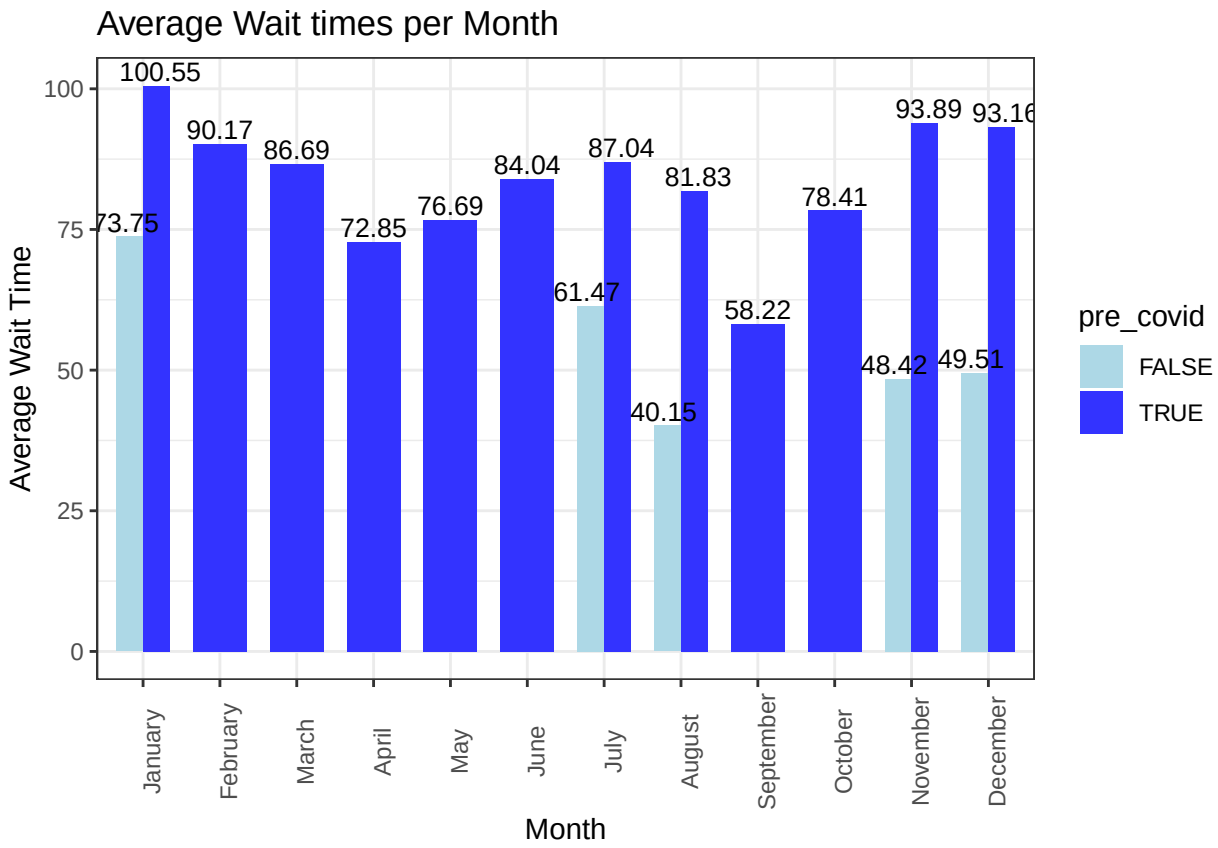
```
#Creating quarters
flight_of_passage <- flight_of_passage %>%
    mutate(date = date, quarter = quarter(date, with_year = TRUE))
```

```
#Wait time by quarters
flight_of_passage %>%
    group_by(quarter) %>%
    mutate(avg_wait = mean(SACTMIN)) %>%
    distinct(quarter, avg_wait) %>%
    ggplot() +
    geom_col(aes(quarter, avg_wait), fill = "lightblue")+
    ggtitle("Average Wait times Per Quarter")+
    labs(y = "Average Wait Time" , x = "Quarters by Year")
```



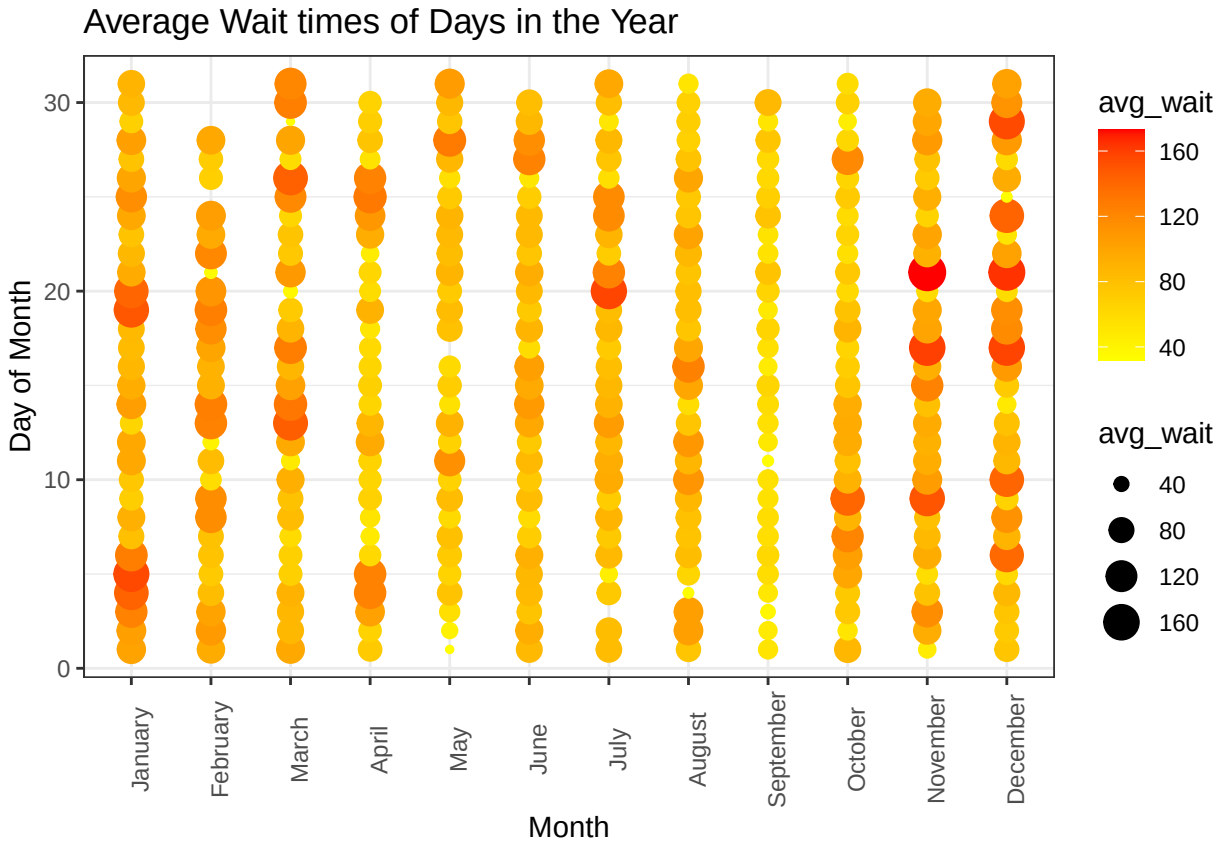

```
#Wait time by month
flight_of_passage %>%
    group_by(month, pre_covid) %>%
    summarise(avg_wait = mean(SACTMIN)) %>%
    ggplot(aes(month, avg_wait, fill = pre_covid))+
    geom_bar(stat = "identity", position = "dodge", aes(group = pre_covid), width = .7)+
    geom_text(aes(label = round(avg_wait, digits = 2)), position = position_dodge(width = .9), vjust = -
    xlim("January", "February", "March", "April", "May", "June", "July", "August", "September", "Octobe
    theme(axis.text.x = element_text(angle = 90))+
    scale_fill_manual(values = c("#add8e6", "#3333FF"))+
    ggtitle("Average Wait times per Month")+
    labs(x = "Month", y = "Average Wait Time")
```

```
## `summarise()` has grouped output by 'month'. You can override using the `.groups` argument.
```




```
#Wait time by day
flight_of_passage %>%
    filter(pre_covid == "TRUE") %>%
    group_by(month, day) %>%
    summarise(avg_wait = mean(SACTMIN)) %>%
    ggplot(aes(month, day))+
    geom_point(aes(size = avg_wait, color = avg_wait))+
    scale_color_gradient(low = "yellow", high = "red")+
    xlim("January", "February", "March", "April", "May", "June", "July", "August", "September", "Octobe
    theme(axis.text.x = element_text(angle = 90))+
    ggtitle("Average Wait times of Days in the Year")+
    labs(x = "Month", y = "Day of Month")
```

```
## `summarise()` has grouped output by 'month'. You can override using the `.groups` argument.
```




```
#Correlation
cor(flight_of_passage$SACTMIN, flight_of_passage$day)
```

```
## [1] 0.01455967
```

```
cor(flight_of_passage$SACTMIN, flight_of_passage$month)
```

```
## [1] -0.1816571
```

```
cor(flight_of_passage$SACTMIN, flight_of_passage$year)
```

```
## [1] -0.31504
```

The first graph gave me the understanding that the wait time for Flight of Passage has generally been decreasing. It seems that the first quarter of the year (January, February and March) has the highest wait times. This doesn't surprised me since I expected New Years to be the most busy and therefore have the longest wait times, and for back to school season result in the shortest lines. The graph also tells me that the wait times have still been decreasing even after people have been allowed back in the parks.

The second graph indicated what I was already expecting. Before the parks closed, November, December, and January had the longest wait times. There is a lapse of data from February to June post-covid since the parks haven't been open these months since 2019. However, I can expect that the wait times from January - June post-covid mimic what the January-June pre-covid wait times were like. It seems that people still want to be in Disney for the holidays, regardless of the wait times, weather, or potential health risk. The average wait time now is at least 30 minutes less than what it used to be which may be caused by its age and the attraction of newer rides within the park.

The third graph is the most interesting to me because it shows which specific days of the months are busiest during the pre-covid season. While I would have preferred to have information from 2020 and 2021 in this graph, there wasn't enough information available post-covid to form any conclusions. Therefore this

visual uses information prior to 2020.

As I pointed out before, the holidays seem to be the most popular. Christmas and New Years really increase the wait time, as well as the end of March (spring break season) and the end of July (summer vacation). Things really slow down as kids go back to school in September and pick back up when fall break comes around in October. It's also interesting to see a slight increase in wait time around Valentine's Day. From this graph, I am starting to expect that the end of September would be the best time to go to Disney.

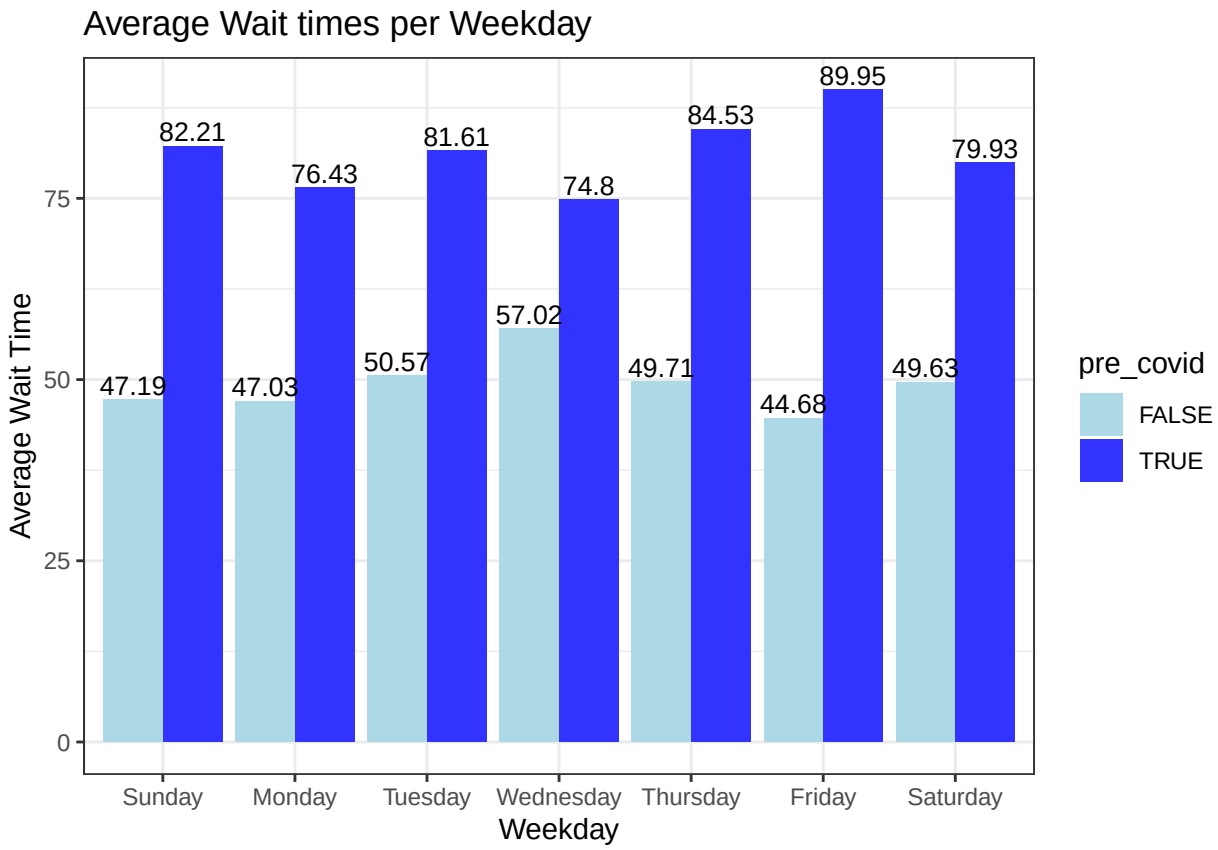
The last information I included in this section was the correlation between wait time and day, month, and year. None of these correlations were too high, but I will still consider them valuable predictors for wait time since I don't have too many.

Average Wait Time Throughout the Week

Since I had looked at the wait times through the year, I also wanted to focus on the wait times through the week. I created the graph below to show how weekdays affect the ride lines.

```
#Pre-COVID
flight_of_passage %>%
    group_by(weekday, pre_covid) %>%
    summarise(avg_wait = mean(SACTMIN)) %>%
    ggplot(aes(weekday, avg_wait, fill = pre_covid))+
    geom_bar(stat = "identity", position = "dodge", aes(group = pre_covid))+
    geom_text(aes(label = round(avg_wait, digits = 2)), position = position_dodge(width = .9), vjust = -
    xlim("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday")+
    scale_fill_manual(values = c("#add8e6", "#3333FF"))+
    ggtitle("Average Wait times per Weekday")+
    labs(x = "Weekday", y = "Average Wait Time")
```

```
## `summarise()` has grouped output by 'weekday'. You can override using the `.groups` argument.
```




```
#Correlation
cor(flight_of_passage$SACTMIN, flight_of_passage$weekday)
```

```
## [1] 0.01559506
```

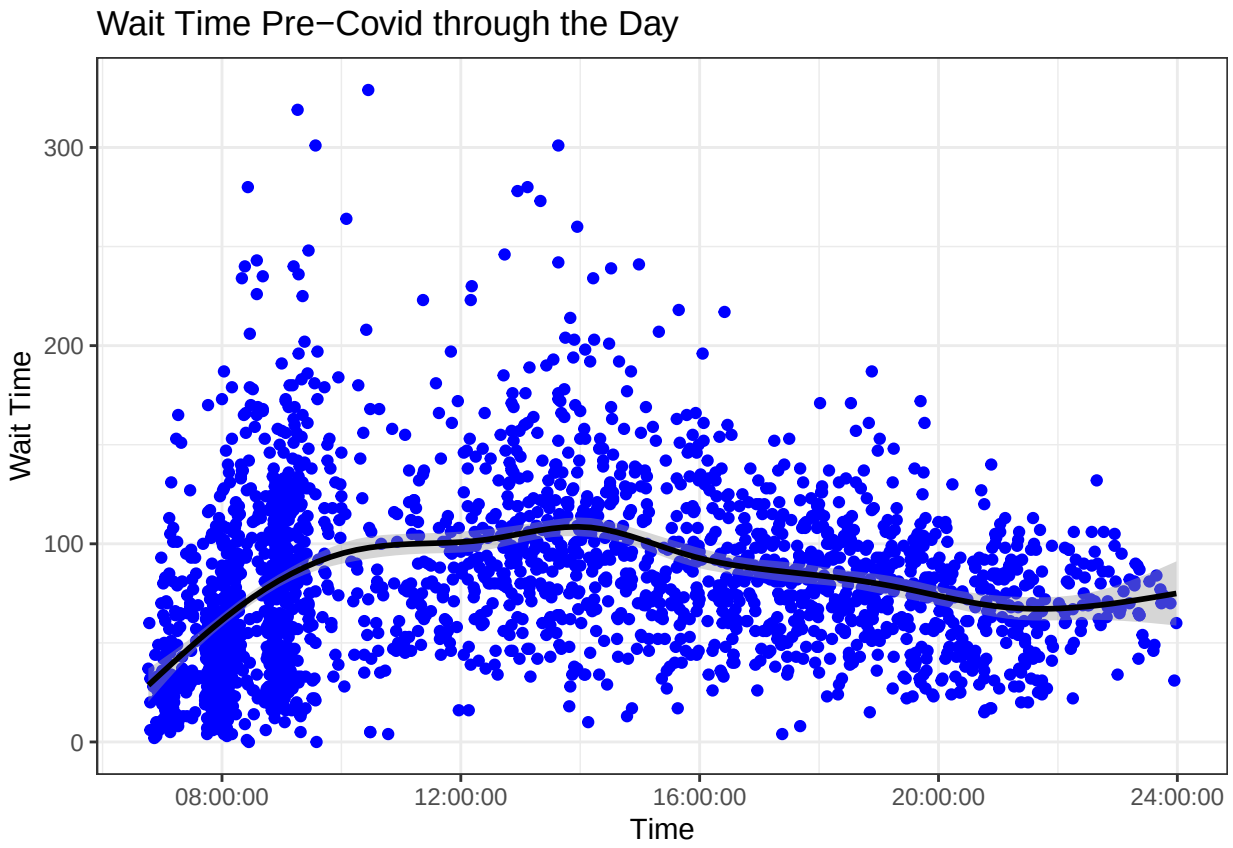
Interestingly, the wait times pre-covid and post-covid follow opposite trends. While the weekends resulted in longer lines before the shut-down, it now seems that the longest lines are during the week (specifically Wednesday). Again we can see that the lines are at least 20 minutes quicker now than they were before. I also ran code to see the correlation between weekday and wait time, which showed me that it is not that strong of a predictor on its own. However, I will still consider it in my model.

Average Wait Time During the Day

Even more specifically than year, month, or weekday is the time of day. To begin, I created scatterplots that show which times of day have been most popular before and after the closure. To remove some of the outliers and make my graph easier to read, I limited the hours to be from 6AM to midnight. Then I chunked my time into hour long blocks to get a visual of which sections of times were busiest.

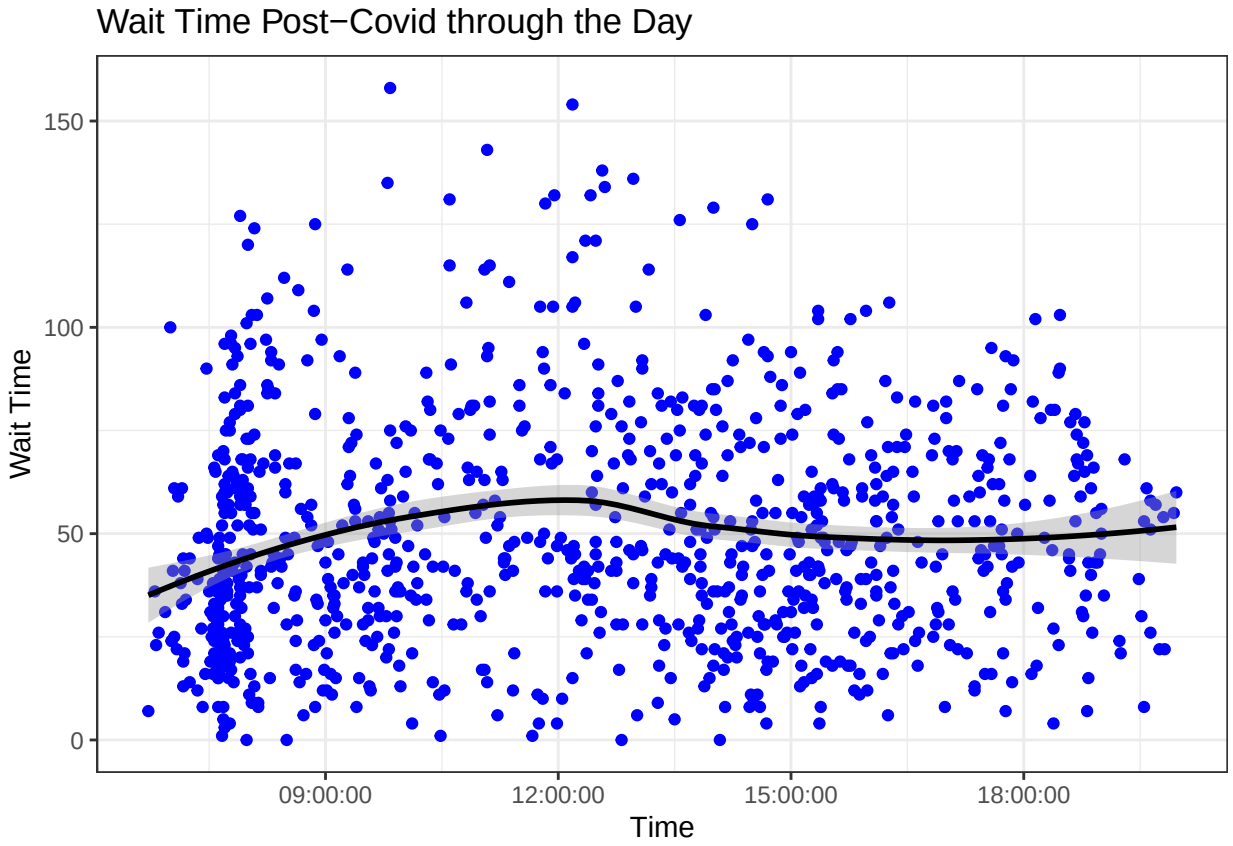
```
#Day beginning at 6 AM pre-covid
flight_of_passage %>%
    filter(pre_covid == "TRUE" & hour(time) >= 06 & minute(time) >= 00) %>%
    ggplot()+
    geom_point(aes(time, SACTMIN), color = "blue")+
    geom_smooth(aes(time, SACTMIN), color = "black")+
    ggtitle("Wait Time Pre-Covid through the Day")+
    labs(x = "Time", y = "Wait Time")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
#Day beginning at 6 AM post-covid
flight_of_passage %>%
    filter(pre_covid =="FALSE" & hour(time) >= 06 & minute(time) >= 00) %>%
    ggplot()+
    geom_point(aes(time, SACTMIN),
               color = "blue")+
    geom_smooth(aes(time, SACTMIN), color = "black")+
    ggtitle("Wait Time Post-Covid through the Day")+
    labs(x = "Time", y = "Wait Time")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



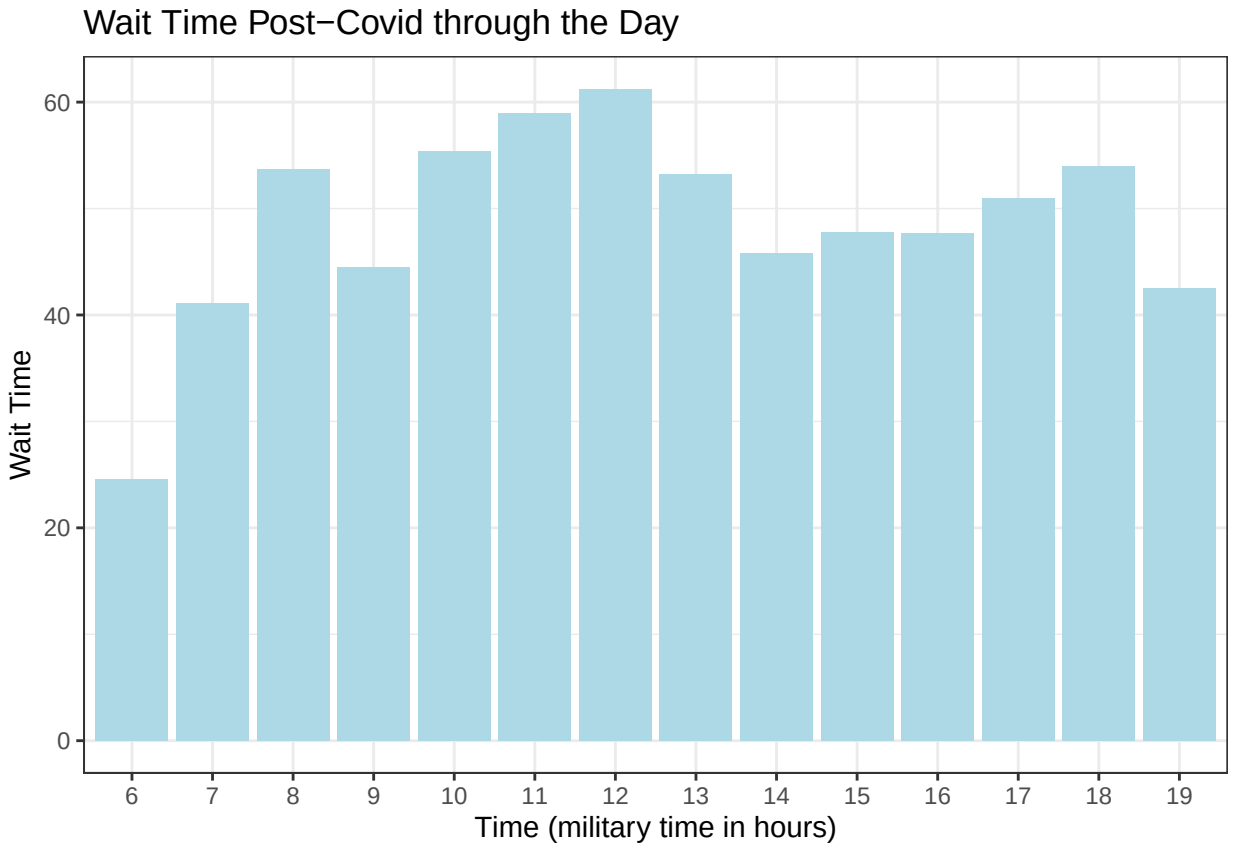

```
#Grouping Time into military time
flight_of_passage <- flight_of_passage %>%
    mutate(time_of_day = case_when(
        hms("00:00:00") <= time & time < hms("01:00:00") ~ "00",
        hms("01:00:00") <= time & time < hms("02:00:00") ~ "1",
        hms("02:00:00") <= time & time < hms("03:00:00") ~ "2",
        hms("03:00:00") <= time & time < hms("04:00:00") ~ "3",
        hms("04:00:00") <= time & time < hms("05:00:00") ~ "4",
        hms("05:00:00") <= time & time < hms("06:00:00") ~ "5",
        hms("06:00:00") <= time & time < hms("07:00:00") ~ "6",
        hms("07:00:00") <= time & time < hms("08:00:00") ~ "7",
        hms("08:00:00") <= time & time < hms("09:00:00") ~ "8",
        hms("09:00:00") <= time & time < hms("10:00:00") ~ "9",
        hms("10:00:00") <= time & time < hms("11:00:00") ~ "10",
        hms("11:00:00") <= time & time < hms("12:00:00") ~ "11",
        hms("12:00:00") <= time & time < hms("13:00:00") ~ "12",
        hms("13:00:00") <= time & time < hms("14:00:00") ~ "13",
        hms("14:00:00") <= time & time < hms("15:00:00") ~ "14",
        hms("15:00:00") <= time & time < hms("16:00:00") ~ "15",
        hms("16:00:00") <= time & time < hms("17:00:00") ~ "16",
        hms("17:00:00") <= time & time < hms("18:00:00") ~ "17",
        hms("18:00:00") <= time & time < hms("19:00:00") ~ "18",
        hms("19:00:00") <= time & time < hms("20:00:00") ~ "19",
        hms("20:00:00") <= time & time < hms("21:00:00") ~ "20",
        hms("21:00:00") <= time & time < hms("22:00:00") ~ "21",
        hms("22:00:00") <= time & time < hms("23:00:00") ~ "22",
        hms("23:00:00") <= time & time <= hms("23:59:99") ~ "23"
```

```
    ))



#Order of times for the next graph
flight_of_passage$time_of_day <- factor(flight_of_passage$time_of_day, levels = c("00", "1", "2", "3",

#Which hours have the longest wait times
flight_of_passage %>%
    filter(pre_covid == "FALSE") %>%
    group_by(time_of_day) %>%
    summarise(avg_wait = mean(SACTMIN)) %>%
    ggplot(aes(time_of_day, avg_wait))+
    geom_col(fill = "lightblue")+
    ggtitle("Wait Time Post-Covid through the Day")+
    labs(x = "Time (military time in hours)", y = "Wait Time")
```




```
#Correlation
flight_of_passage$time_of_day <- as.numeric(flight_of_passage$time_of_day)
cor(flight_of_passage$SACTMIN, flight_of_passage$time_of_day)
```

```
## [1] 0.1231846
```

The first graph shows that the pre-covid wait time was on a steady increase until around 2PM when it began to shorten until the park closed for the day. The peak of the wait seemed to be almost 2 hours! Post-covid, it seems like there is a similar trend where the wait time steadily increases until around 1PM where it shortens. However, the maximum wait time post-covid is ~60 minutes and while this is significantly

shorter than the pre-covid peak, the wait time remains to be around 50 minutes for the rest of the day, just like pre-covid. Interestingly, I think that the steady line we see in the post-covid graph is partly because of Disney's fastpass+ system. This will be discussed further at the end of the report.

The bar graph shows similar information. The longest wait time is between 11AM and 1PM and then quickly drops until it picks up again at 3PM. We can see that the park's opening automatically means a drastic increase in wait time. I also computed the correlation between wait time and the time of day (by hours). It came out to be .12, so I will use it as a predictor in my model as well.

Regression Techniques

The next step of this report was to create the models. I decided to use a multiple linear regression model and a penalized regression model. Splitting my data into training and testing sets, I was able to evaluate their RMSE and R-squared values.

Multiple Linear Regression

```
#Splitting the data
split <- initial_split(drop_na(flight_of_passage), prop = .9)

train_data <- training(split)
test_data <- testing(split)


#Selecting Predictors
wait_time_lm <- lm(SACTMIN ~ ., train_data)
summary(wait_time_lm)
```

```
##
## Call:
## lm(formula = SACTMIN ~ ., data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -87.901 -26.917  -4.892  20.213 252.518
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    2.096e+06  9.492e+05   2.209   0.0273 *
## day           -2.767e+00  1.319e+00  -2.098   0.0360 *
## time          -6.505e-04  6.544e-04  -0.994   0.3203
## weekday        5.691e-01  3.471e-01   1.639   0.1012
## date           2.900e+00  1.320e+00   2.198   0.0281 *
## year          -1.225e+03  4.864e+02  -2.519   0.0118 *
## month         -9.467e+01  4.032e+01  -2.348   0.0190 *
## pre_covidTRUE  1.846e+01  3.240e+00   5.698 1.33e-08 ***
## quarter        1.614e+02  2.948e+01   5.475 4.75e-08 ***
## time_of_day    3.185e+00  2.361e+00   1.349   0.1775
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 39.24 on 2960 degrees of freedom
## Multiple R-squared:  0.145,  Adjusted R-squared:  0.1424
## F-statistic: 55.77 on 9 and 2960 DF,  p-value: < 2.2e-16
```

```
ols_step_forward_aic(wait_time_lm)
```

```
##
##                               Selection Summary
## ------------------------------------------------------------------------------
## Variable          AIC         Sum Sq          RSS         R-Sq      Adj. R-Sq
## ------------------------------------------------------------------------------
## pre_covid      30331.135    604151.761    4727577.117    0.11331      0.11301
## time_of_day    30301.974    653493.949    4678234.929    0.12257      0.12198
## date           30283.293    685957.544    4645771.334    0.12866      0.12777
## quarter        30265.245    717211.672    4614517.206    0.13452      0.13335
## year           30254.506    736961.115    4594767.764    0.13822      0.13677
## month          30241.141    760670.154    4571058.724    0.14267      0.14093
## day            30238.723    767465.662    4564263.216    0.14394      0.14192
## weekday        30238.113    771473.767    4560255.111    0.14469      0.14238
## ------------------------------------------------------------------------------
```

```
ols_step_both_aic(wait_time_lm)
```

```
##
##
##                                    Stepwise Summary
## ---------------------------------------------------------------------------------------------
## Variable        Method        AIC           RSS           Sum Sq       R-Sq      Adj. R-Sq
## ---------------------------------------------------------------------------------------------
## pre_covid      addition    30331.135    4727577.117    604151.761    0.11331      0.11301
## time_of_day    addition    30301.974    4678234.929    653493.949    0.12257      0.12198
## date           addition    30283.293    4645771.334    685957.544    0.12866      0.12777
## quarter        addition    30265.245    4614517.206    717211.672    0.13452      0.13335
## year           addition    30254.506    4594767.764    736961.115    0.13822      0.13677
## month          addition    30241.141    4571058.724    760670.154    0.14267      0.14093
## day            addition    30238.723    4564263.216    767465.662    0.14394      0.14192
## weekday        addition    30238.113    4560255.111    771473.767    0.14469      0.14238
## ---------------------------------------------------------------------------------------------
```

```
ols_step_backward_aic(wait_time_lm)
```

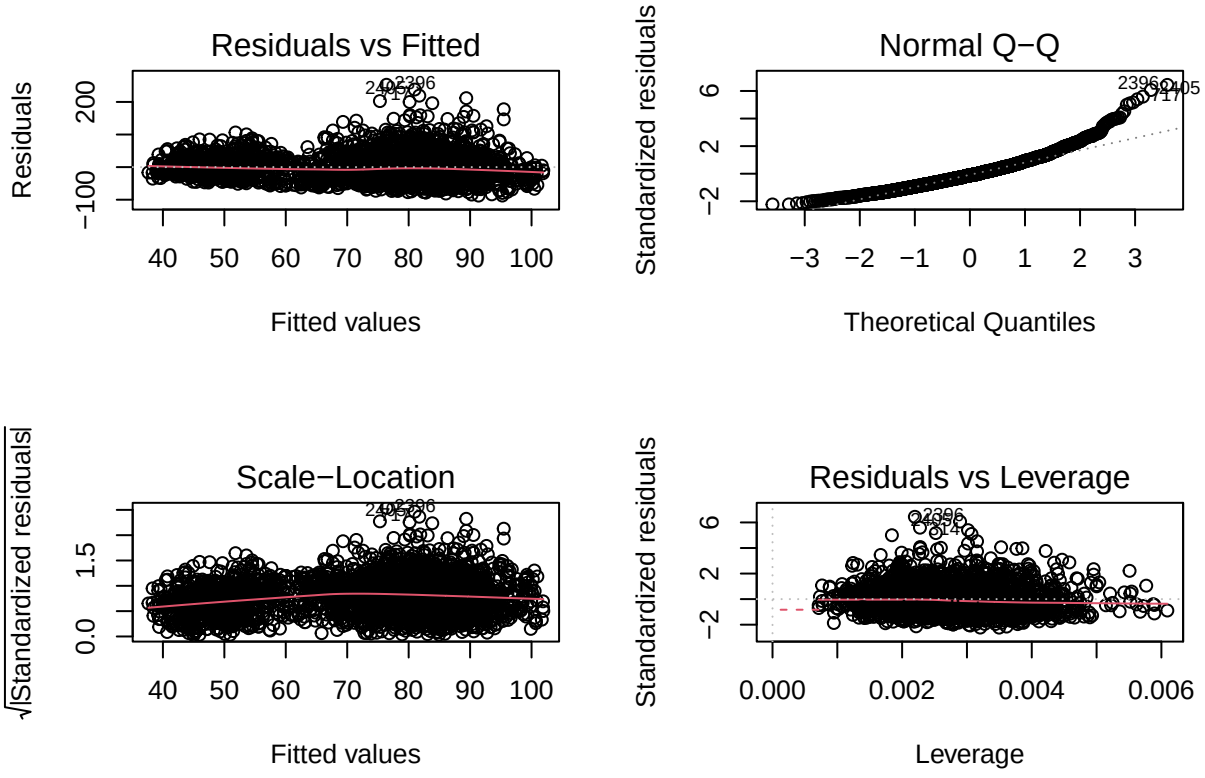
```
##
##
##                           Backward Elimination Summary
## ------------------------------------------------------------------------------
## Variable         AIC           RSS           Sum Sq       R-Sq      Adj. R-Sq
## ------------------------------------------------------------------------------
## Full Model    30239.122    4558733.191    772995.687    0.14498      0.14238
## time          30238.113    4560255.111    771473.767    0.14469      0.14238
## ------------------------------------------------------------------------------
```

```
wait_time_lm_adjusted <- lm(SACTMIN ~day + weekday + year + month + pre_covid + time_of_day + quarter, d
summary(wait_time_lm_adjusted)
```

```
##
## Call:
## lm(formula = SACTMIN ~ day + weekday + year + month + pre_covid +
##     time_of_day + quarter, data = train_data)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -87.374 -26.632  -4.521  20.418 252.609
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   10579.7676  2047.4670   5.167 2.53e-07 ***
## day               0.1241     0.0815   1.523    0.128
## weekday           0.5533     0.3472   1.594    0.111
## year           -159.4873    29.3651  -5.431 6.05e-08 ***
## month            -6.1039     0.9297  -6.566 6.09e-11 ***
## pre_covidTRUE    16.5570     3.1318   5.287 1.34e-07 ***
## time_of_day       0.8190     0.1609   5.091 3.79e-07 ***
## quarter         154.2703    29.2556   5.273 1.44e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 39.27 on 2962 degrees of freedom
## Multiple R-squared:  0.1433, Adjusted R-squared:  0.1413
## F-statistic: 70.77 on 7 and 2962 DF,  p-value: < 2.2e-16
```

```
par(mfrow = c(2,2))
plot(wait_time_lm_adjusted)
```




```
#RMSE of LM Model Predictors
full_model_predictors <- predict(wait_time_lm, data.frame(test_data))
wait_time_lm_rmse <- sqrt(mean(test_data$SACTMIN - full_model_predictors)^2)
wait_time_lm_rmse
```

```
## [1] 0.4644973
```

```
#MLR Model Making and Evaluating
wait_time_lm_model <- linear_reg() %>%
    set_engine("lm") %>%
    set_mode("regression")
wait_time_lm_fit <- wait_time_lm_model %>%
    fit(SACTMIN ~ day + weekday + year + month + pre_covid + time_of_day + quarter, data = train_data)
tidy(wait_time_lm_fit)
```

```
## # A tibble: 8 x 5
##   term           estimate std.error statistic  p.value
##   <chr>             <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   10580.    2047.          5.17 2.53e- 7
## 2 day               0.124    0.0815      1.52 1.28e- 1
## 3 weekday           0.553    0.347       1.59 1.11e- 1
## 4 year           -159.      29.4        -5.43 6.05e- 8
## 5 month            -6.10     0.930      -6.57 6.09e-11
## 6 pre_covidTRUE    16.6      3.13        5.29 1.34e- 7
## 7 time_of_day       0.819    0.161       5.09 3.79e- 7
## 8 quarter         154.      29.3         5.27 1.44e- 7
```

```
glance(wait_time_lm_fit)
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic  p.value    df  logLik    AIC    BIC
##       <dbl>         <dbl> <dbl>     <dbl>    <dbl> <dbl>   <dbl>  <dbl>  <dbl>
## 1     0.143         0.141  39.3      70.8 6.85e-95     7 -15112. 30241. 30295.
## # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

```
predict(wait_time_lm_fit, new_data = test_data)
```

```
## # A tibble: 331 x 1
##    .pred
##    <dbl>
##  1  87.7
##  2  84.3
##  3  77.0
##  4  86.0
##  5  86.0
##  6  76.9
##  7  88.4
##  8  86.6
##  9  90.7
## 10  89.7
## # ... with 321 more rows
```

```
test_results <- predict(wait_time_lm_fit, new_data = test_data) %>%
    bind_cols(test_data)
test_results
```

```
## # A tibble: 331 x 11
##    .pred   day time   SACTMIN weekday date        year month pre_covid quarter
##    <dbl> <int> <time>   <dbl>   <dbl> <date>     <dbl> <dbl> <lgl>       <dbl>
##  1  87.7    27 07:01       22       7 2017-05-27  2017     5 TRUE        2017.
##  2  84.3    29 06:59       21       2 2017-05-29  2017     5 TRUE        2017.
##  3  77.0     4 09:28       65       1 2017-06-04  2017     6 TRUE        2017.
##  4  86.0    10 15:49       88       7 2017-06-10  2017     6 TRUE        2017.
```

```
##  5  86.0    10 15:49       89       7 2017-06-10  2017     6 TRUE        2017.
##  6  76.9    12 07:59       45       2 2017-06-12  2017     6 TRUE        2017.
##  7  88.4    12 21:47       72       2 2017-06-12  2017     6 TRUE        2017.
##  8  86.6    13 18:03      129       3 2017-06-13  2017     6 TRUE        2017.
##  9  90.7    13 23:39       84       3 2017-06-13  2017     6 TRUE        2017.
## 10  89.7    14 21:13       92       4 2017-06-14  2017     6 TRUE        2017.
## # ... with 321 more rows, and 1 more variable: time_of_day <dbl>
```

```
#RMSE of LM Fitted Model Predictors
rmse(test_results, truth = SACTMIN, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        41.6
```

```
rsq(test_results, truth = SACTMIN, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.140
```

In the multiple linear regression model, we can see that the model chooses day, weekday, year, month, pre-covid, time of day, and quarter as its predictors. The only predictor it neglects to use is time. Once I established which predictors I wanted, I found the RMSE of the multiple linear regression model that was based on those predictors. Using the training data to form this model resulted in the RMSE being 1.328 when assessing the testing data. The r-squared of this model was also .143. Overall, these values could definitely be better. Since this model didn't perform very well, I decided to compare it to another model that used LASSO regression.

Penalized Regression - LASSO

```
#Penalized Regression Split Attempt
folds <- vfold_cv(train_data, v = 10)           #splits train data again

wait_time_recipe <- recipe(SACTMIN ~ + year + month + pre_covid + time_of_day, data = flight_of_passage)
    step_center(all_numeric_predictors()) %>%
    step_scale(all_numeric_predictors())

wait_time_model <- linear_reg(mixture = 1, penalty = tune()) %>%
    set_engine("glmnet")

wait_time_workflow <- workflow() %>%
    add_recipe(wait_time_recipe) %>%
    add_model(wait_time_model)

grid <- grid_regular(penalty(), levels = 50)     #makes 50 penalties to test

tuned_grid <- tune_grid(wait_time_workflow,
                        resamples = folds,
                        grid = grid)

best_fit <- tuned_grid %>%                       #finds best penalty based on rsme
    select_best(metric = "rmse")
```

```
wait_time_workflow <- wait_time_workflow %>%                        #best penalty goes into workflow
    finalize_workflow(best_fit)

wait_time_pr_fit <- wait_time_workflow %>%                         #considers train and test data
    last_fit(split = split)

wait_time_pr_fit$.workflow[[1]] %>%                      #shows estimates for predictors with best penalty
    extract_fit_parsnip() %>%
    tidy()
```

```
## # A tibble: 5 x 3
##   term        estimate      penalty
##   <chr>          <dbl>        <dbl>
## 1 (Intercept)    60.7  0.0000000001
## 2 year           -6.31 0.0000000001
## 3 month          -4.29 0.0000000001
## 4 pre_covid      15.9  0.0000000001
## 5 time_of_day     3.45 0.0000000001
```

```
wait_time_pr_metrics <- wait_time_workflow %>%                      #gives rmse and r2 for the model
    last_fit(split = split) %>%
    collect_metrics()
wait_time_pr_metrics
```

```
## # A tibble: 2 x 4
##   .metric .estimator .estimate .config
##   <chr>   <chr>          <dbl> <chr>
## 1 rmse    standard      41.7   Preprocessor1_Model1
## 2 rsq     standard       0.137 Preprocessor1_Model1
```

```
wait_time_pr_predictions <- wait_time_pr_fit$.workflow[[1]] %>%          #shows predictions and actual
    last_fit(split) %>%
    collect_predictions()
wait_time_pr_predictions
```

```
## # A tibble: 331 x 5
##    id               .pred  .row SACTMIN .config
##    <chr>            <dbl> <int>   <dbl> <chr>
##  1 train/test split  84.6     4      22 Preprocessor1_Model1
##  2 train/test split  83.9    10      21 Preprocessor1_Model1
##  3 train/test split  84.7    40      65 Preprocessor1_Model1
##  4 train/test split  89.3    75      88 Preprocessor1_Model1
##  5 train/test split  89.3    76      89 Preprocessor1_Model1
##  6 train/test split  83.2    89      45 Preprocessor1_Model1
##  7 train/test split  93.8    97      72 Preprocessor1_Model1
##  8 train/test split  91.5   102     129 Preprocessor1_Model1
##  9 train/test split  95.3   105      84 Preprocessor1_Model1
## 10 train/test split  93.8   107      92 Preprocessor1_Model1
## # ... with 321 more rows
```

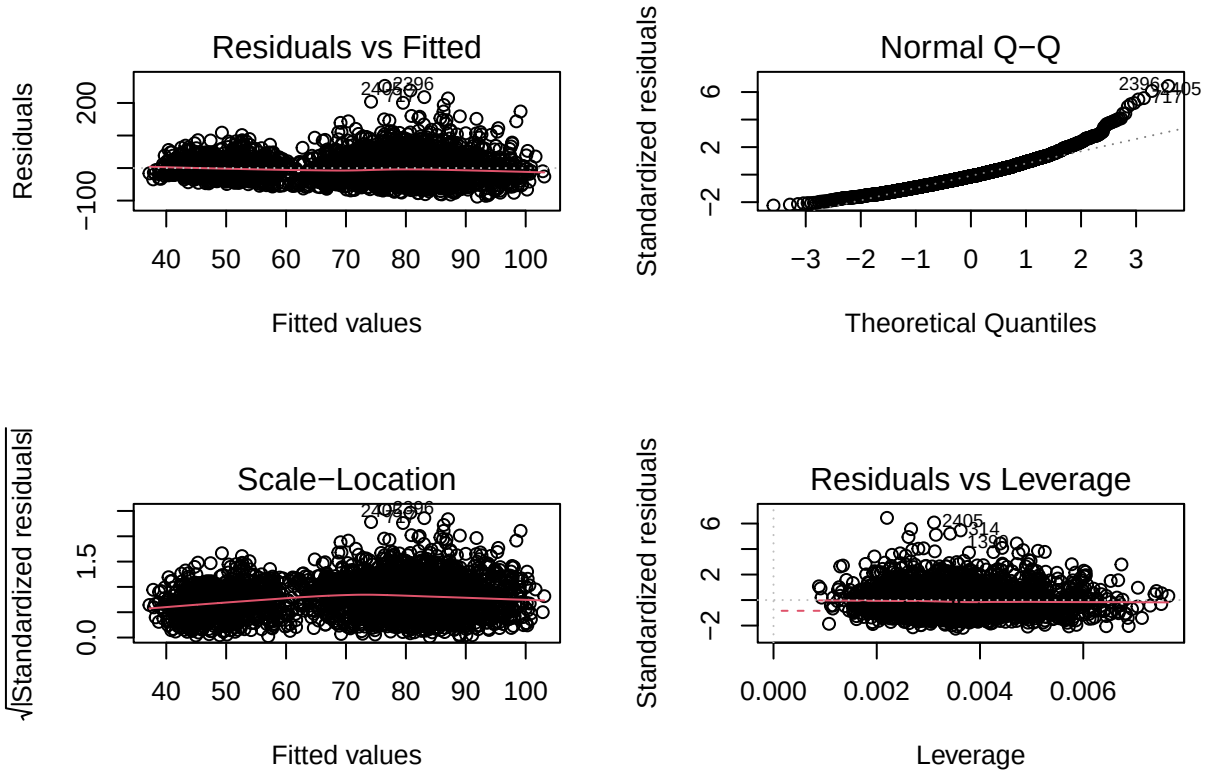
The LASSO regression model seems to have done an even worse job than the multiple linear regression model. The RMSE when using the same predictors as before is now 40.021 and the R-squared is 0.171. When analyzing the estimates of the predictors, day and weekday are shown to be the least important. This makes sense since the correlation coefficients of these predictors were close to zero. Because of this, I decided to run the model again without the predictors day and weekday. My new LASSO regression model had an RSME of 39.854 and an R-squared of .180. This is slightly better than before! Considering that we want a low RMSE

and a high R-squared though, I am not happy with this model.

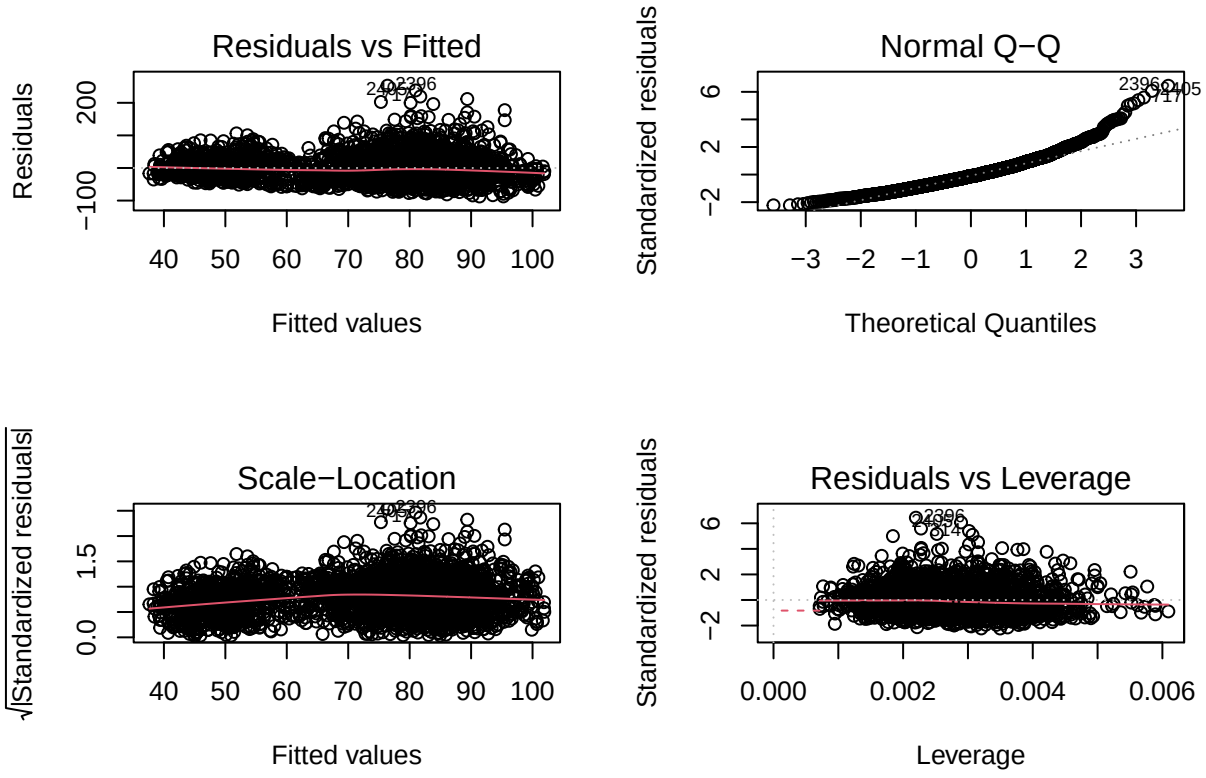
Assumptions

Although I had an idea of what the assumptions were for my models, I wanted to clearly see whether there was constant variance and if the Normal Q-Q plot was well fit.

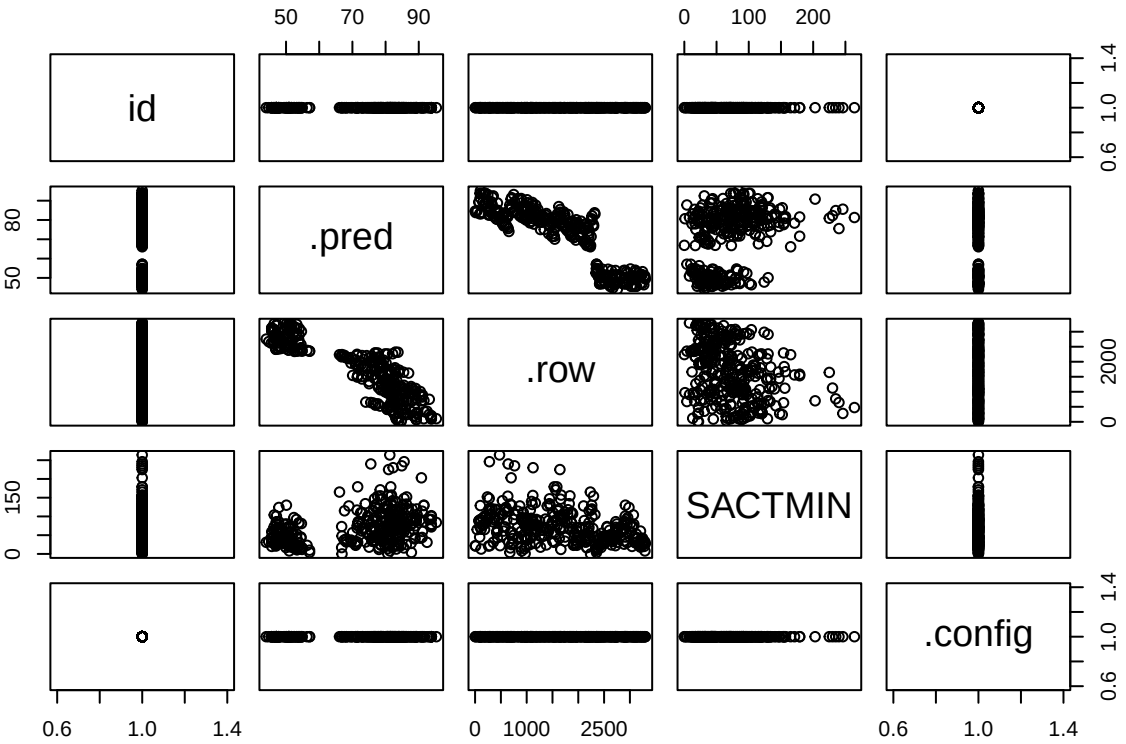
```
par(mfrow = c(2,2))
plot(wait_time_lm)
```




```
par(mfrow = c(2,2))
plot(wait_time_lm_adjusted)
```




```
plot(wait_time_pr_predictions)
```



Evidently, the assumptions of my linear regression model were not great. The Normal Q-Q plot trailed away at the tail and there were so many points in the Residuals plot that it is hard to tell if there's constant variance (even though it seems like the blob gets wider at the end). My penalized regression model had a weird relationship too where the predictions weren't aligned very well with the wait time.

Final Discussion

Shortcomings and Possible Improvements

As touched on before, there were some executive decisions made when analyzing this data. To remove outliers, I narrowed the time of day to 6am to midnight. I am unsure how data was collected between midnight and 6am since the parks would be closed, but I decided to mark those few observations as outliers. I also noticed that when I parted by time into hour long chunks, the times that were strictly on the hours (for example, 7:00:00), were marked as NA. Although this didn't happen often, it is important to note that none of the observations involving the hours throughout the day included on-the-dot times.

Aside from the shortcomings resulting from the data itself, I wanted to mention some of the systems Disney implements for their visitors. In the past, Disney offered a program called fastpass+ which allowed guests to reserve a spot in the "fast lane". The fast lane allows guests to jump to the front of the line on a ride of their choosing one time between a designated hour during the day. Fastpass+ users are able to use a second queue lane which turns their time in line to be at most 20 minutes. There is a limit to how many/which rides guests can use their fastpass+ on, but this must have had an effect on the data. Since reopening in July, Disney has made a lot of adjustments to their fastpass+ system and queue lane in general. Visitors can now "stand in line" virtually, allowing them to have a seat saved without being physically near so many other people. The fastpass+ system has since been renamed and now comes with a price tag (surprising, right?).

Overall, there are a lot of factors that go into how long people wait in line. Weather, covid rates, travel restrictions, etc. all play a part in how long lines become. I don't have data on all of this yet alone the time in my day to work with it all, so it's important to recognize the limitations of the data I'm working with in this project. When I found the dataset, there was no description of how the wait times were obtained, so I'd be interested to see what method they used and how it affects the interpretation of my model. Regardless, I think that the data reflects significant trends in guests and can provide a glimpse into what Disney-goers consider when planning their next trip.