Final Project 6306

Tim Morales

12/7/2020

Objective

Although many different regions have taken large hits to business during the global pandemic, few cities in the world were hit as hard as New York City. Claimed to be the epicenter during the initial wave of COVID-19 in the United States, New York City has seen huge loses to local business across a variety of industries including one that is a staple to the city itself, yellow cabs. Yellow cabs have covered the streets of New York City for decades, but with the rise of COVID-19, finds itself in trying times.

In this paper, we look at how the pandemic has affected NYC yellow cabs using individual pick up data from January 1, 2018 and May 31st, 2020. We compare the effects of the pandemic on yellow cabs in comparison to its main competition, for hire vehicles including Lyft and Uber. Through collecting, processing and summarizing over 33GB of individual ride information from the NYC Taxi and Limousine Commission, we use a time series approach to show how COVID-19 has hurt NYC yellow cab predictions in comparison to its for hire competitors.

Data Manipulation

When understanding the data collection, there are a few things to note within the code. There were multiple instances of recording error especially for yellow cab individual ride data. For those errors, whether it be that the data was placed in the incorrect data frame or the year was incorrect, we drooped the observations as we cannot justify a proper way to solve or assure the true meaning behind those recording.

Below is the Sparklyr Code used in collaboration with an AWS EMR cluster to process the for hire vehicle information. (SEE RMD CODE FOR ACTUAL CODE)

```r
install.packages("sparklyr")
install.packages("tidyverse")
library(sparklyr)
library(tidyverse)

config <- spark_config()                            # Create a config to tune memory
config[["sparklyr.shell.driver-memory"]] <- "20G"   # Set driver memory to 20GB

sc <- spark_connect(master = "yarn",               # Connect to the AWS Cluster
                    config = config,
                    spark_home = "/usr/lib/spark")  # This is where AWS puts the Spark Code

#I read in all the data stored in S3 ~ 18.8 GB
DATA_ALL <- spark_read_csv (sc,
                            "data",
                             "s3://stat6306studentfilebucket/Tim Morales/FHV/*.csv")

## BELOW IS FOR 2018
# Multiple iterations due to changes in labeling

#Select only pick up data
Step1 <- DATA_ALL %>%
  select(Pickup_DateTime)%>%
  filter(Pickup_DateTime != "Pickup_DateTime") #omit column names if loading error

#spliting date variable
Step2 <- Step1  %>%
  mutate(Pickup_DateTimeSplit = split(Pickup_DateTime, "-")) %>%
  sdf_separate_column("Pickup_DateTimeSplit", into = c("Year", "Month", "Day_Time"))

#spliting day variable
Step3 <- Step2  %>%
  mutate(Day_TimeSplit = split(Day_Time, " ")) %>%
  sdf_separate_column("Day_TimeSplit", into = c("Day", "Time"))

#making numeric and creating a summary
Step4 <-Step3 %>% filter(Year >= 2018)%>%
  mutate(Year = as.numeric(Year),
         Month = as.numeric(Month),
         Day = as.numeric(Day),
  )%>%
  group_by(Year, Month, Day)%>%    #count every single pick up for
  summarise(n = n())              #each day each month each year


Final <- Step4 %>% collect()            #collecting into a dataframe.

##BELOW I REPEAT THE PROCESS FOR THE 2019-2020 DATA
#NOTICE DIFFERENT NAMING
Step1 <- DATA_ALL %>%
  select(pickup_datetime)%>%
  filter(pickup_datetime != "pickup_datetime")

Step2 <- Step1  %>%
  mutate(pickup_datetimesplit = split(pickup_datetime, "-")) %>%
  sdf_separate_column("pickup_datetimesplit", into = c("Year", "Month", "Day_Time"))


Step3 <- Step2  %>%
  mutate(Day_TimeSplit = split(Day_Time, " ")) %>%
  sdf_separate_column("Day_TimeSplit", into = c("Day", "Time"))


Step4 <-Step3 %>% filter(Year >= 2018)%>%
  mutate(Year = as.numeric(Year),
         Month = as.numeric(Month),
         Day = as.numeric(Day),
  )%>%
  group_by(Year, Month, Day)%>%
  summarise(n = n())


Final1920 <- Step4 %>% collect()


Final1920 <- Final1920 %>%
  arrange(Year, Month, Day)

#WRITE TO BUCKET
spark_write_csv (Step4,
                 "s3://stat6306studentfilebucket/Tim Morales/Fullsets/FHV2018Summary.csv", # => File lo
                 header = TRUE,
                 mode = "overwrite",
                 charset = "UTF-8")
write.csv(Final,"s3://stat6306studentfilebucket/Tim Morales/Fullsets/FHV2018Summary.csv")
aws.s3::put_object("FHV2018Summary.csv",
                   file = "FHV2018Summary.csv",
                   bucket = "s3://stat6306studentfilebucket/Tim Morales/Fullsets/")
```

Below you can see the code used for the NYC yellow cab data. This was done within my local computer at the same time as the AWS cluster. Notice this was done in a loop and only 1 iteration is shown.

```
#i read in the file for the year and month
df <- read.csv(file="/Users/timmorales/Desktop/fhv_data/fhv_tripdata_2020-05.csv")
library(tidyverse)
head(df)

#i select the date variable
Step1 <- df %>%
  select(pickup_datetime)%>%
  separate(pickup_datetime, c("Year","Month","Day_Time"),sep = "-")%>% #separate date
  separate(Day_Time, c("Day", "Time"), sep = " ")%>% #finish date separation
  filter(Year >= 2018)%>% #help avoid recording errors and slowing computation
  mutate(Year = as.numeric(Year), #summarize by month, year, day
         Month = as.numeric(Month),
         Day = as.numeric(Day),
  )%>%
  group_by(Year, Month, Day)%>%
  summarise(n = n())

test = Step1%>%
  filter(Month == 5, Year == 2020)

#saving as CSV
write.csv(test, "/Users/timmorales/Desktop/fhv_data/Summaries/summary_fhv_2020-05.csv")
```

With those summary tables I have created I will use the rbind function to unit them.

Below I review each data frame individual and make sure they load properly and functionally. I also go through each to check for any issues. I do so for each month, each year, each vehicle type. (SEE RMD CODE FOR ACTUAL CODE)

Yellow Cab

```
df12018 <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_yellow_tripdata_2018-01.csv")
df22018 <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_yellow_tripdata_2018-02.csv")
df32018 <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_yellow_tripdata_2018-03.csv")
df42018 <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_yellow_tripdata_2018-04.csv")
df52018 <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_yellow_tripdata_2018-05.csv")
df62018 <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_yellow_tripdata_2018-06.csv")
df72018 <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_yellow_tripdata_2018-07.csv")
df82018 <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_yellow_tripdata_2018-08.csv")
df92018 <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_yellow_tripdata_2018-09.csv")
df102018 <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_yellow_tripdata_2018-10.csv
df112018 <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_yellow_tripdata_2018-11.csv
df122018 <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_yellow_tripdata_2018-12.csv

df_2018_full <- rbind(df12018,df22018,df32018,df42018,df52018,df62018,df72018,df82018,df92018,df102018,
```

I do the same for 2019

```
df12019 <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_yellow_tripdata_2019-01.csv")
df22019 <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_yellow_tripdata_2019-02.csv")
df32019 <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_yellow_tripdata_2019-03.csv")
df42019 <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_yellow_tripdata_2019-04.csv")
df52019 <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_yellow_tripdata_2019-05.csv")
df62019 <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_yellow_tripdata_2019-06.csv")
df72019 <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_yellow_tripdata_2019-07.csv")
df82019 <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_yellow_tripdata_2019-08.csv")
df92019 <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_yellow_tripdata_2019-09.csv")
df102019 <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_yellow_tripdata_2019-10.csv
df112019 <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_yellow_tripdata_2019-11.csv
df122019 <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_yellow_tripdata_2019-12.csv

df_2019_full <- rbind(df12019,df22019,df32019,df42019,df52019,df62019,df72019,df82019,df92019,df102019,
```

and 2020

```
df12020 <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_yellow_tripdata_2020-01.csv")
df22020 <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_yellow_tripdata_2020-02.csv")
df32020 <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_yellow_tripdata_2020-03.csv")
df42020 <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_yellow_tripdata_2020-04.csv")
df52020 <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_yellow_tripdata_2020-05.csv")

df_2020_full <- rbind(df12020,df22020,df32020,df42020,df52020)


df_total <- rbind(df_2020_full,df_2019_full, df_2018_full)
```

FHV

```
FHV2018<- read.csv("/Users/timmorales/Desktop/fhv_data/Summaries/FHV2018Summary.csv")
FHV20191<- read.csv("/Users/timmorales/Desktop/fhv_data/Summaries/summary_fhv_2019-01.csv")
FHV20192<- read.csv("/Users/timmorales/Desktop/fhv_data/Summaries/summary_fhv_2019-02.csv")
FHV20193<- read.csv("/Users/timmorales/Desktop/fhv_data/Summaries/summary_fhv_2019-03.csv")
FHV20194<- read.csv("/Users/timmorales/Desktop/fhv_data/Summaries/summary_fhv_2019-04.csv")
FHV20195<- read.csv("/Users/timmorales/Desktop/fhv_data/Summaries/summary_fhv_2019-05.csv")
FHV20196<- read.csv("/Users/timmorales/Desktop/fhv_data/Summaries/summary_fhv_2019-06.csv")
FHV20197<- read.csv("/Users/timmorales/Desktop/fhv_data/Summaries/summary_fhv_2019-07.csv")
FHV20198<- read.csv("/Users/timmorales/Desktop/fhv_data/Summaries/summary_fhv_2019-08.csv")
FHV20199<- read.csv("/Users/timmorales/Desktop/fhv_data/Summaries/summary_fhv_2019-09.csv")
FHV201910<- read.csv("/Users/timmorales/Desktop/fhv_data/Summaries/summary_fhv_2019-10.csv")
FHV201911<- read.csv("/Users/timmorales/Desktop/fhv_data/Summaries/summary_fhv_2019-11.csv")
FHV201912<- read.csv("/Users/timmorales/Desktop/fhv_data/Summaries/summary_fhv_2019-12.csv")
FHV20201<- read.csv("/Users/timmorales/Desktop/fhv_data/Summaries/summary_fhv_2020-01.csv")
FHV20202<- read.csv("/Users/timmorales/Desktop/fhv_data/Summaries/summary_fhv_2020-02.csv")
FHV20203<- read.csv("/Users/timmorales/Desktop/fhv_data/Summaries/summary_fhv_2020-03.csv")
FHV20204<- read.csv("/Users/timmorales/Desktop/fhv_data/Summaries/summary_fhv_2020-04.csv")
FHV20205<- read.csv("/Users/timmorales/Desktop/fhv_data/Summaries/summary_fhv_2020-05.csv")

FHV_total <- rbind(FHV2018,FHV20191,FHV20192,FHV20193,FHV20194,FHV20195,FHV20196,FHV20197,FHV20198,FHV2

FHV_total<-FHV_total%>%
arrange(Year, Month, Day)
#write.csv(FHV_total, "/Users/timmorales/Desktop/fhv_data/Summaries/summary_total.csv")
```

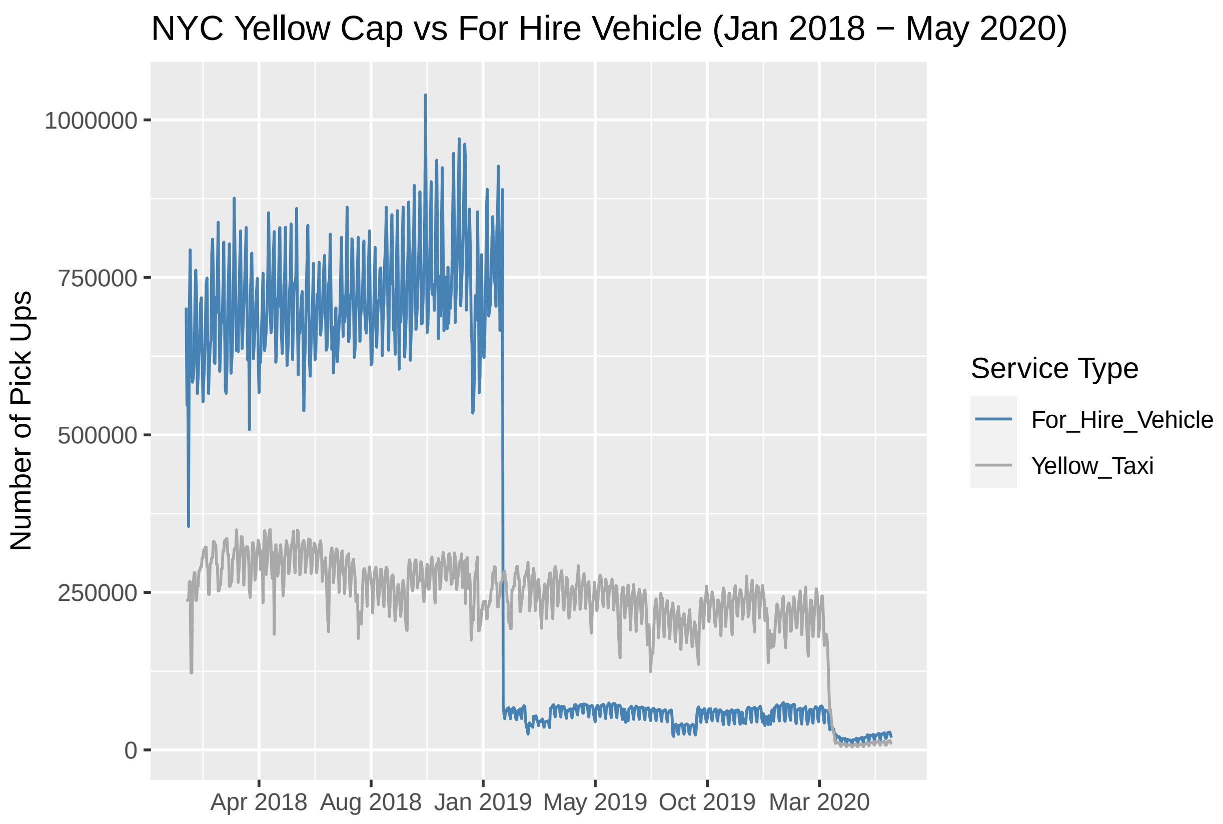
Analysis

EDA

I combine the data frames together below and continue with some EDA and visualization.

```
FHV_total<-read.csv("/Users/timmorales/Desktop/fhv_data/Summaries/summary_total.csv")
#
# df_ts <- df_total %>%
#   arrange(Year, Month, Day)
#
#
#


#write.csv(df_ts,"/Users/timmorales/Desktop/taxi_data/Summaries/summary_complete.csv")

df_ts <- read.csv("/Users/timmorales/Desktop/taxi_data/Summaries/summary_complete.csv")
inds <- seq(as.Date("2018-01-01"), as.Date("2020-05-31"), by = "day")
df_ts$Date <- seq(as.Date("2018-01-01"), as.Date("2020-05-31"), by = "day")
ts <- ts(df_ts$n, start = c(2018, as.numeric(format(inds[1], "%j"))),
           frequency = 365)
df_ts$Yellow_Taxi <- df_ts$n
df_ts$For_Hire_Vehicle<- FHV_total$n

#melting our data so we can create two lines in ggplot
```

```r
df_ggplot <- df_ts %>%
  select(Date, Yellow_Taxi, For_Hire_Vehicle) %>%
  gather(key = "Service Type", value = "value", -Date)


ggplot(df_ggplot, aes(x = Date, y = value)) +
  geom_line(aes(color = `Service Type`)) +
  scale_color_manual(values = c("steelblue", "darkgrey"))+
  scale_x_date(date_breaks = "20 weeks",date_labels = "%b %Y")+
  xlab("")+
  ylab("Number of Pick Ups")+
  ggtitle("NYC Yellow Cap vs For Hire Vehicle (Jan 2018 - May 2020)")
```



In our initial plot of the time series, we see the obvious drop off in picks due to COVID-19 for both service types. There is also a huge drop off in the for hire vehicle pick ups due to the city's cap on for hire vehicles in early 2019. This was done to protect the yellow cab system.

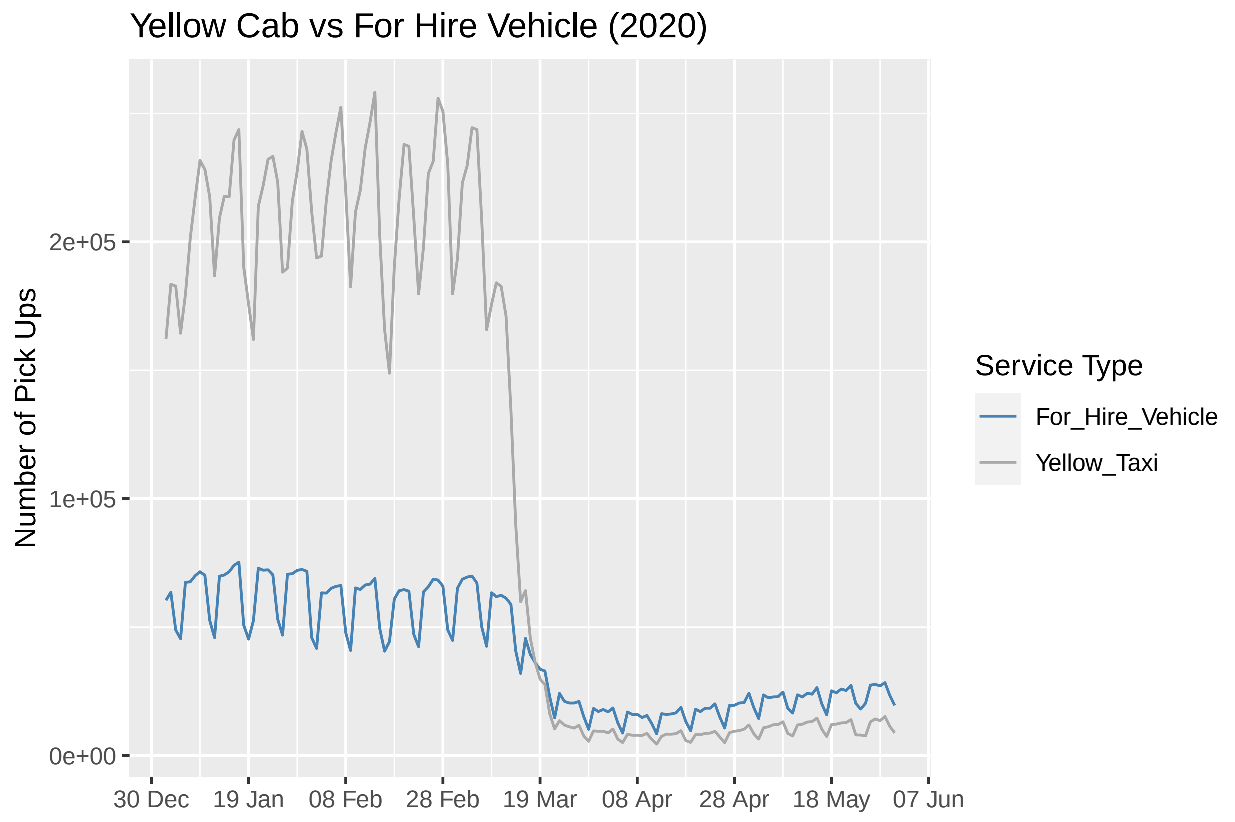
In terms of general trend, there seems to be a long term trend of a slight decrease in pick-ups dating to 2018 for yellow cabs. This raises a concern about stationarity in our data for the yellow cab series.

We can also see what appears to be weekly seasonality, with what we will assume to be increases in pick ups on the weekend.

Excluding 2020 and pre-2019 for FHV, we see no real signs of a multiplicative effect.

With such a large decrease in pick ups for the for hire vehicles as a result of the policy change in early 2019, we will only use the time points after the FHV cap in our modeling of the FHV estimations.

```r
df_ggplot %>%
  filter(Date > ymd(20200101))%>%
  ggplot(aes(x = Date, y = value)) +
  geom_line(aes(color = `Service Type`)) +
  scale_color_manual(values = c("steelblue", "darkgrey"))+
  scale_x_date(date_breaks = "20 days",date_labels = "%d %b")+
  xlab("")+
  ylab("Number of Pick Ups")+
  ggtitle("Yellow Cab vs For Hire Vehicle (2020)")
```



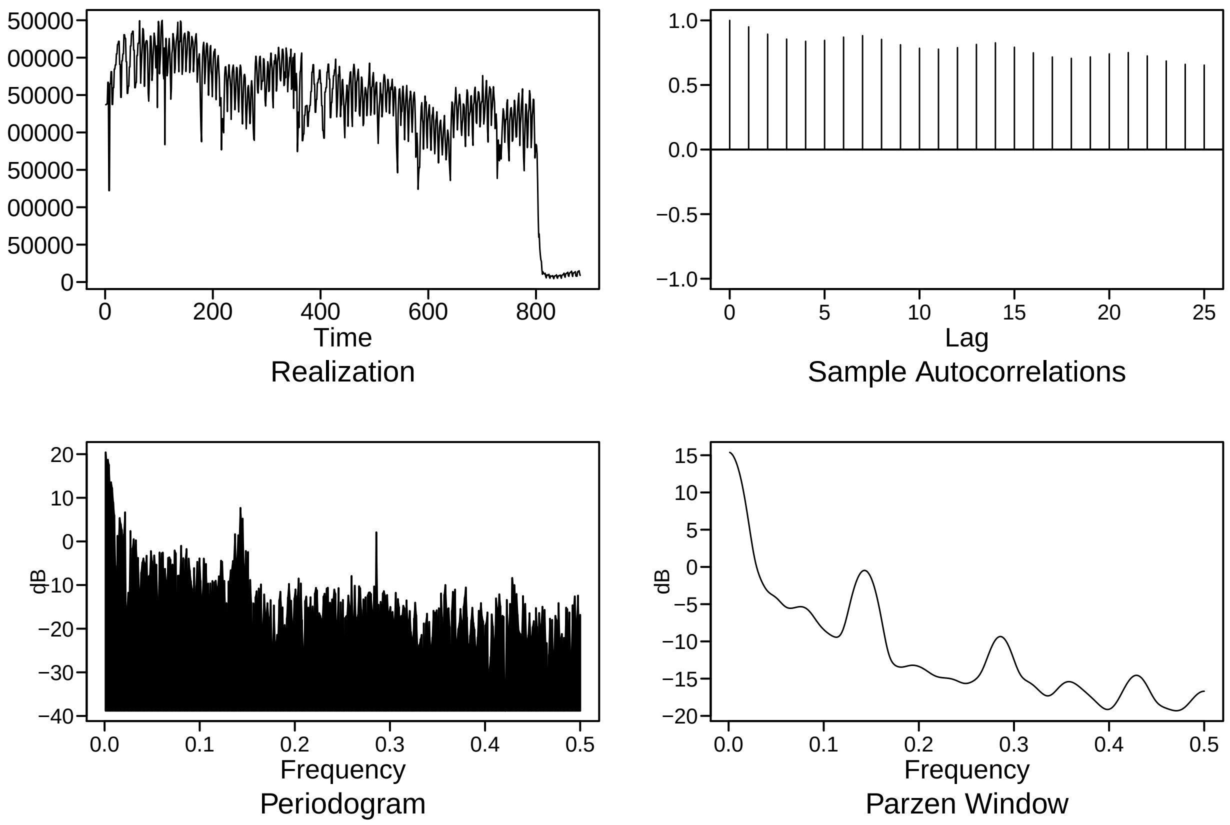
When we zoom in and look at just 2020, we can see the drop off in pick ups is much more severe for the yellow cabs. The large drop in yellow cab pick ups corresponds exactly to the pandemic striking in early - mid March. Although yellow cabs had dominated FHV in pick ups dating back to the cap introduced in 2019, once the pandemic hit, FHV took over as the main service for pickups. FHV also seem to be rebounding from the initial spike faster than yellow cabs.

Yellow Cab

We begin the analysis conducting a SARIMA for the yellow cab time series. We want to see how the pandemic compares with what would have been the prediction without the pandemic striking.

Before the SARIMA, we look at the spectral analysis.

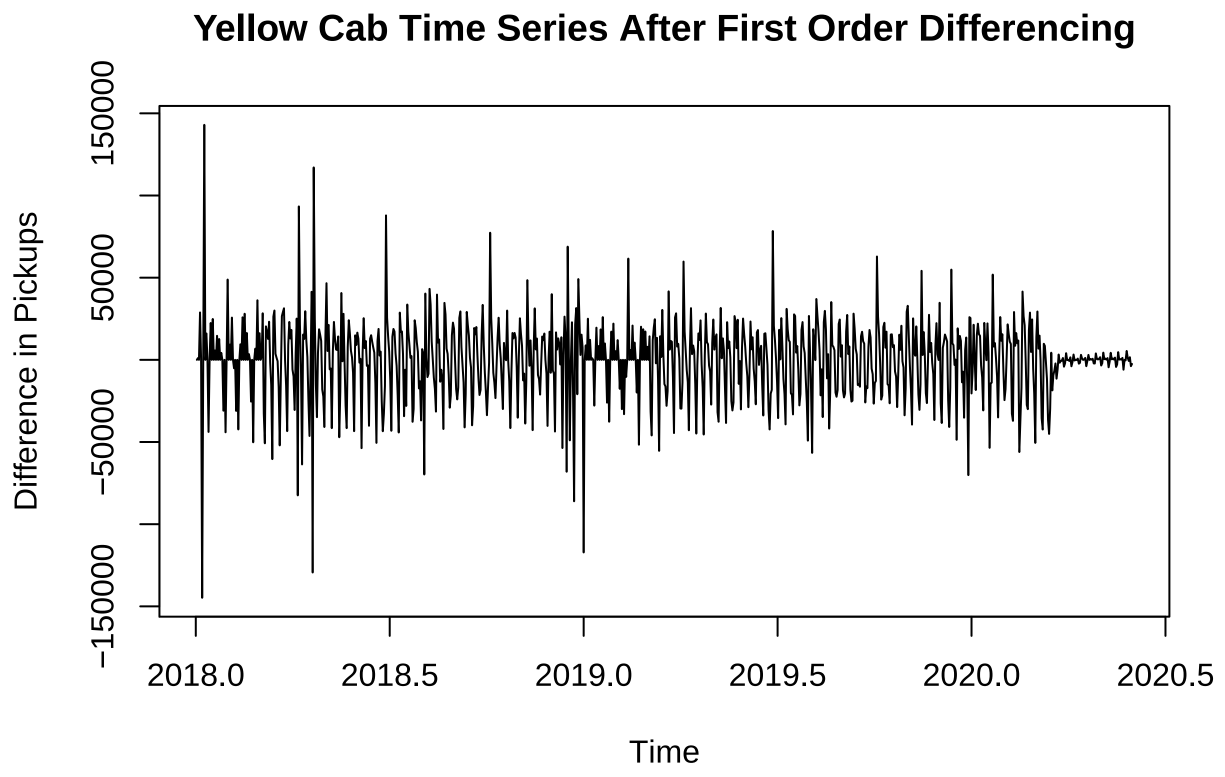
```r
spectral.yellow <-plotts.sample.wge(ts)
```



In the spectral analysis, we see a few things that confirm our initial thoughts. In the autocorrelation function, we see that there is a slow, gradual decline in the autocorrelation. This suggests an autoregressive model. In the periodogram, we see that there is some seasonality the model does not do well in understanding; depicted by the large spike at 0. We also see spikes at frequencies around 0.14, suggesting weekly seasonality.

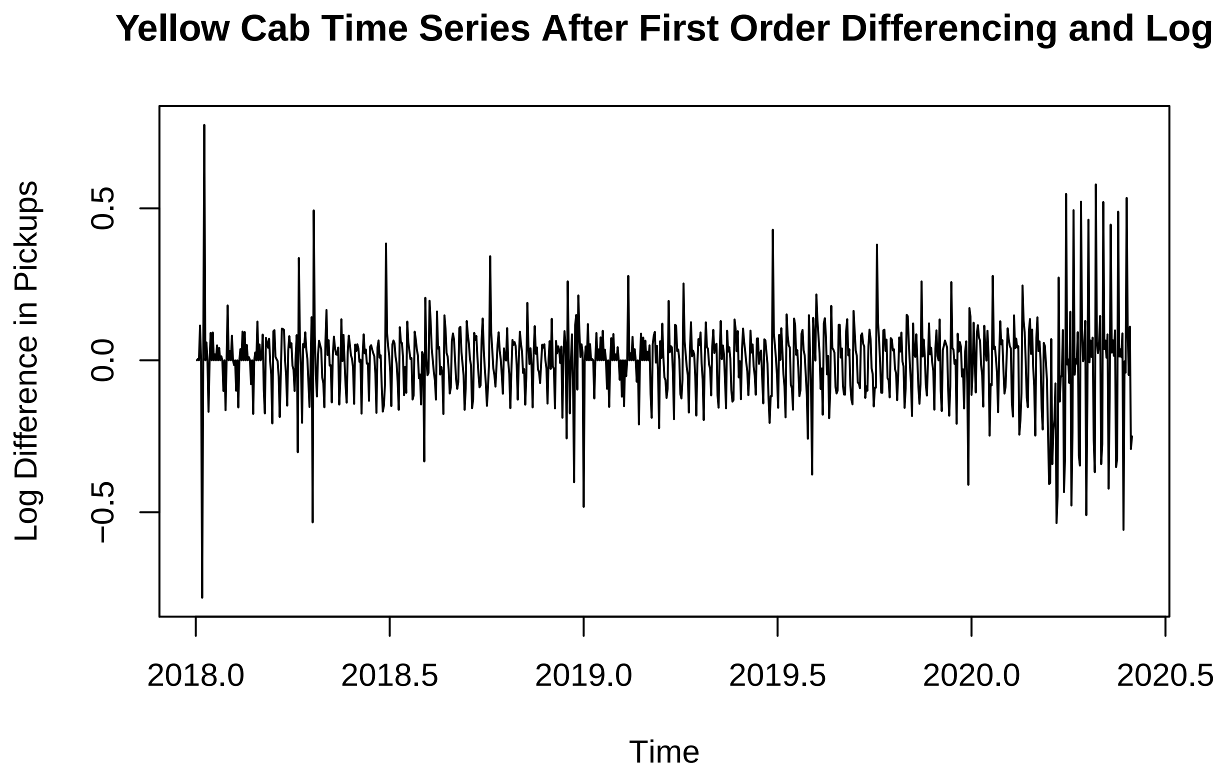
Since we saw some general decreasing trend in the data, we look to see how the graph would look if we used some first order differencing.

```r
diff.ts <- diff(ts)

plot.ts(diff.ts, ylab = "Difference in Pickups")
title("Yellow Cab Time Series After First Order Differencing")
```



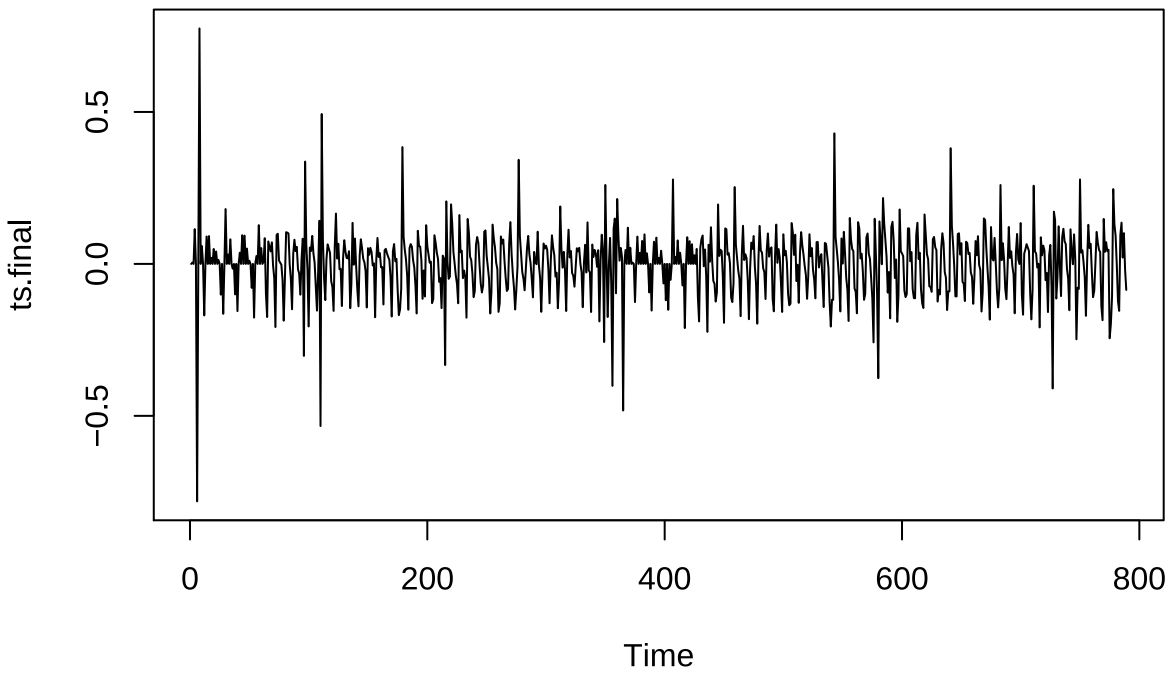
When we look at the first order difference, we actually do see some inconsistent variance, so we will perform a log transform.

```r
log.ts <- log(ts)
ts.final <- diff(log.ts)

plot.ts(ts.final, ylab = "Log Difference in Pickups")
title("Yellow Cab Time Series After First Order Differencing and Log")
```



Looking at the graph, the log transform and differncing creates a relatively stable time series. We just need to drop the post February information.

```r
ts.final <- ts.final[1:789]
ts.plot(ts.final)
```



Finally, we fit the SARIMA using the get.best.arima function. The function works by finding the model with the minimum AIC when allowing parameter values to be any integer between 0 and 5. I do this in parallel.

There is a maximum of 350 lags in the arima function, so I use the 350 lags. This also insures both yellow cab and FHV time series will be the same length.

```r
library(doParallel)
cl <- parallel::makeCluster(3, setup_strategy = "sequential")
registerDoParallel(cl)
#define our get best arima with seasonal
get.best.arima <- function(x.ts, maxord=c(1,1,1,1,1,1))
 {
   best.aic <- 1e8
   n <- length(x.ts)
   for(p in 0:maxord[1]) for (d in 0:maxord[2]) for (q in 0:maxord[3])
     for(P in 0:maxord[4]) for (D in 0:maxord[5]) for (Q in 0:maxord[6])
      {
        fit <- arima(x.ts, order = c(p,d,q),
                           seas = list(order = c(P,D,Q),
                           frequency(x.ts)), method = "CSS")
        fit.aic <- -2 * fit$loglik + (log(n) + 1) * length(fit$coef)
        if (fit.aic < best.aic)
       {
           best.aic <- fit.aic
           best.fit <- fit
           best.model <- c(p,d,q,P,D,Q)
        }
   }
   list(best.aic, best.fit, best.model)
}

fitting.taxi <- log.ts[439:789]
get.best.arima(fitting.taxi,maxord = c(5,5,5,5,5,5))
stopCluster(cl)
```

Since the above code takes so long to run, I just show a picture of the results below.

Results

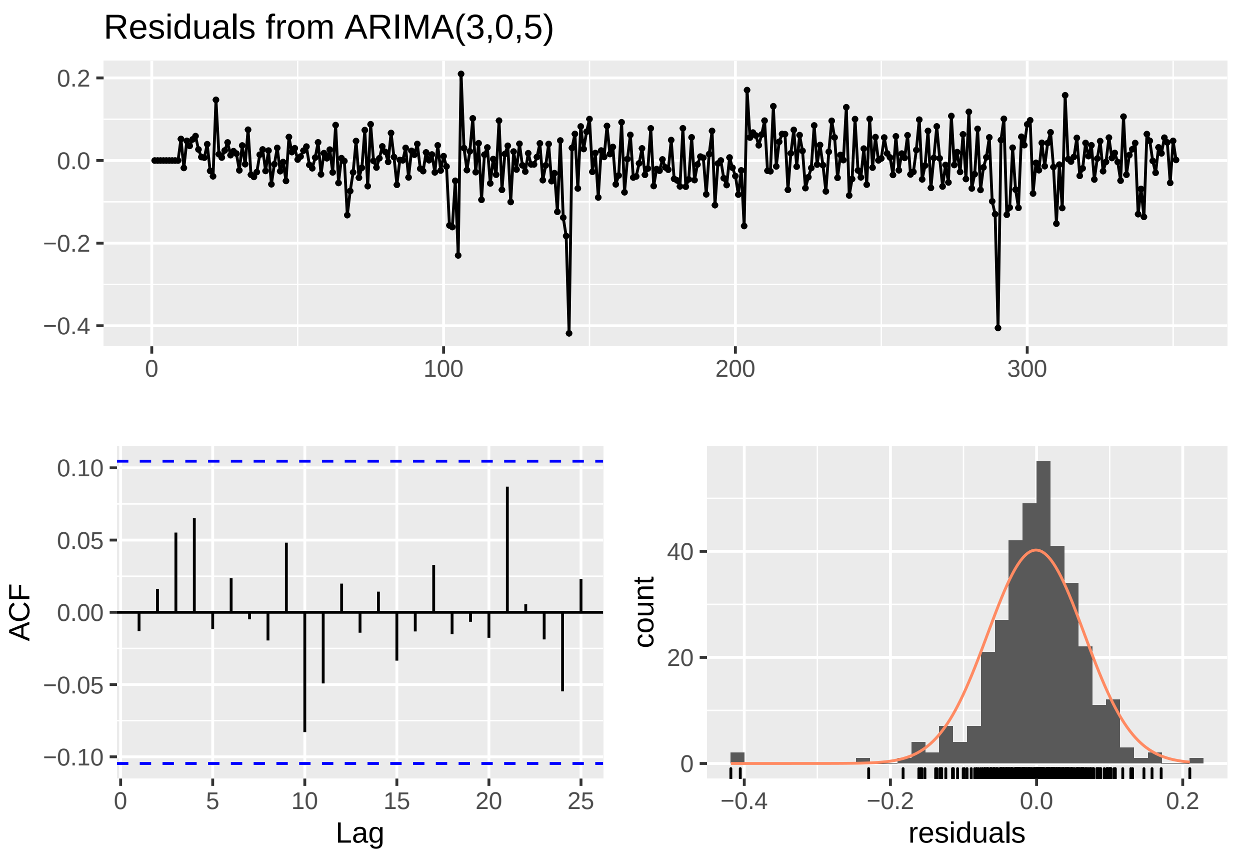
We see that the best model if an SARIMA(3,0,5,5,1,3). It is interesting that it opted for the seasonal differencing.

We then fit that optimal model.

```r
fitting.taxi <- log.ts[439:789]
opt.taxi<-arima(fitting.taxi,order=c(3,0,5), seasonal = list(order=c(5,1,3),frequency(fitting.taxi)),me
```

```
## Warning in arima(fitting.taxi, order = c(3, 0, 5), seasonal = list(order =
## c(5, : possible convergence problem: optim gave code = 1
```

```r
checkresiduals(opt.taxi)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(3,0,5)
## Q* = 8.644, df = 3, p-value = 0.03442
##
## Model df: 16.   Total lags used: 19
```

The residual plots look good and very much like white noise except for a few points that we believe to be holidays.

I conduct a ljungbox to test of lag 1 autocorrelation in residuals.

```
Box.test(opt.taxi$residuals, lag = 1, type = c( "Ljung-Box"))
```

```
##
##  Box-Ljung test
##
## data:  opt.taxi$residuals
## X-squared = 0.059674, df = 1, p-value = 0.807
```

We fail to say there is signficant autocorrelation. We are confident in trusting this models predictions.

Below I get some predictions for the COVID months to compare what was projected and what was actually observed for yellow cab pick ups.

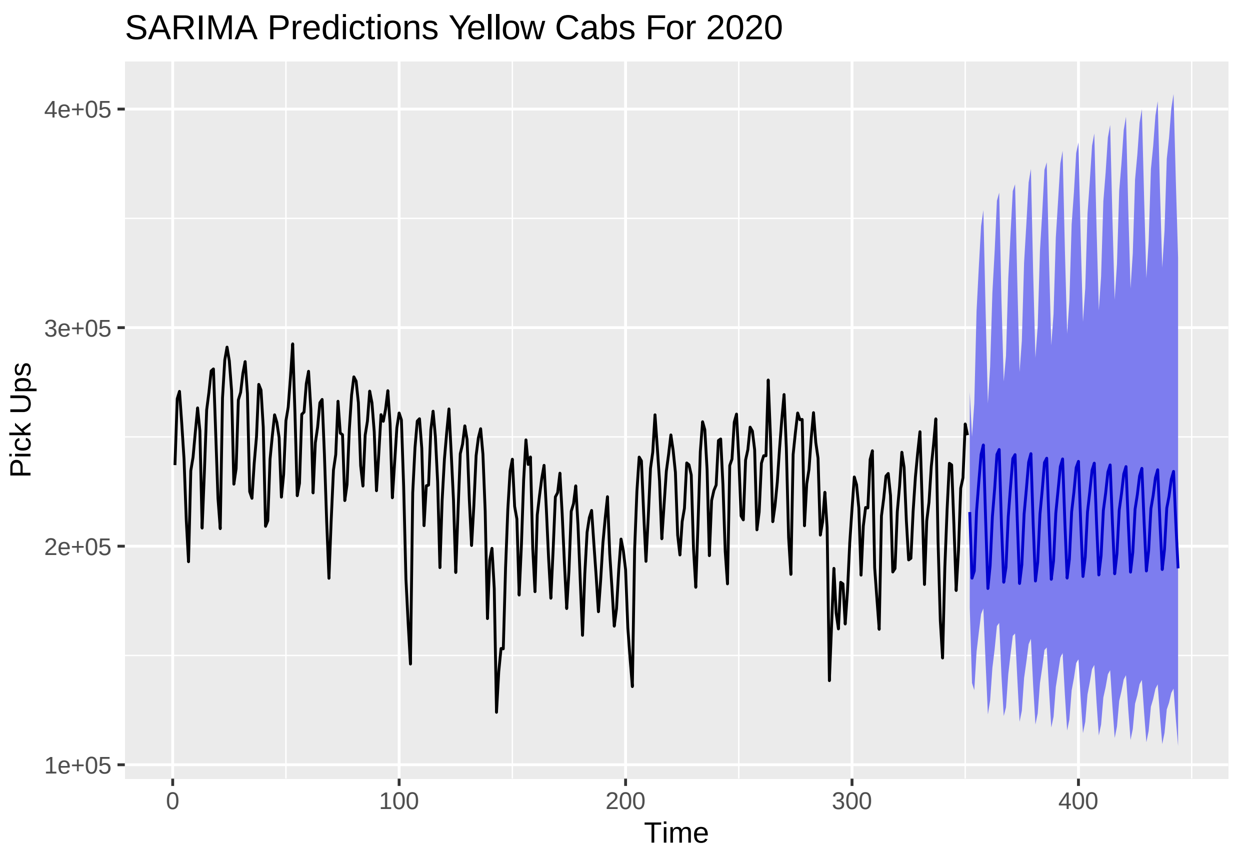
Although the data was log transformed, I do not use the correction factor when converting back to original units since it was not in terms of means, but rather a count.

Below is a plot with a 99.99% confidence interval for the yellow cab pick up numbers.

```r
forecast.taxi<-forecast(opt.taxi,level=c(99.9),h=93)
```

```
## Warning in predict.Arima(object, n.ahead = h): MA part of model is not
## invertible
```

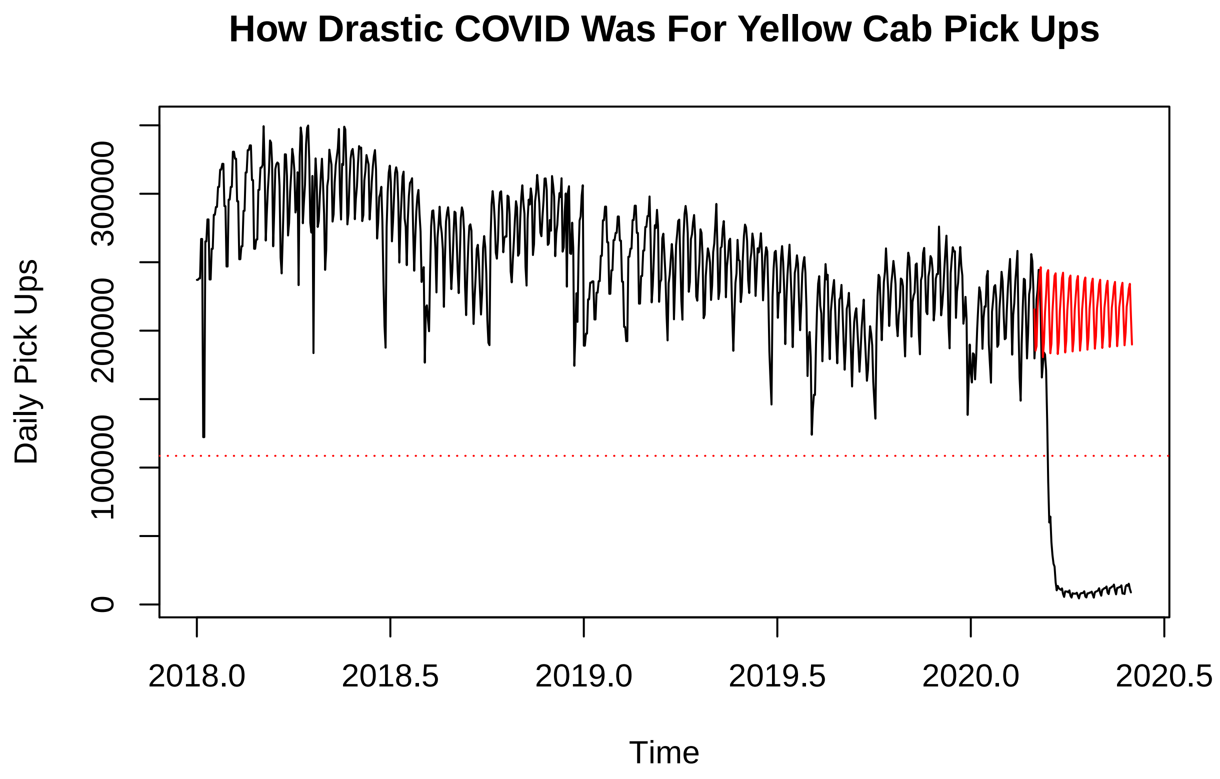
```r
forecast.taxi$x <-exp(forecast.taxi$x)
forecast.taxi$fitted <-exp(forecast.taxi$fitted)
forecast.taxi$mean<-exp(forecast.taxi$mean)
forecast.taxi$upper<-exp(forecast.taxi$upper)
forecast.taxi$lower<-exp(forecast.taxi$lower)
autoplot(forecast.taxi, main="SARIMA Predictions Yellow Cabs For 2020", ylab='Pick Ups')
```



```r
lowest.lowerbound.value<-min(forecast.taxi$lower)
```

We see the lowest points of the 99.9% confidence intervals through May 2020 are around 108,589 daily pick ups. We will use this minimum lower bound in the next section.

```r
#i make the prediction
pred.taxi<-predict(opt.taxi, n.ahead=93)
# i adjust for the log transform
pred.taxi$pred<- exp(pred.taxi$pred)
pred.inds <- seq(as.Date("2020-03-01"), as.Date("2020-05-31"), by = "day")
pred.ts <- ts(pred.taxi$pred, start = c(2020, as.numeric(format(pred.inds[1], "%j"))),
           frequency = 365)

#the plot
ts.plot(ts,pred.ts,col=c("black","red"), main="",ylab="Daily Pick Ups")
abline(h=lowest.lowerbound.value, lty=3, col = "red")
title("How Drastic COVID Was For Yellow Cab Pick Ups")
```



In the above plot where we superimpose predictions on the actual time series, we can see how devasting the pandemic was for the yellow cabs and how drastic the difference is between projection and actuality.

We use the red dotted line to denote the minimum bound of the 99.9% confidence interval for any day between March and May 2020. Even when having 99.9% confidence, the actual number of pick ups is well below that minimum bound of our prediction within a matter of days of the pandemic!

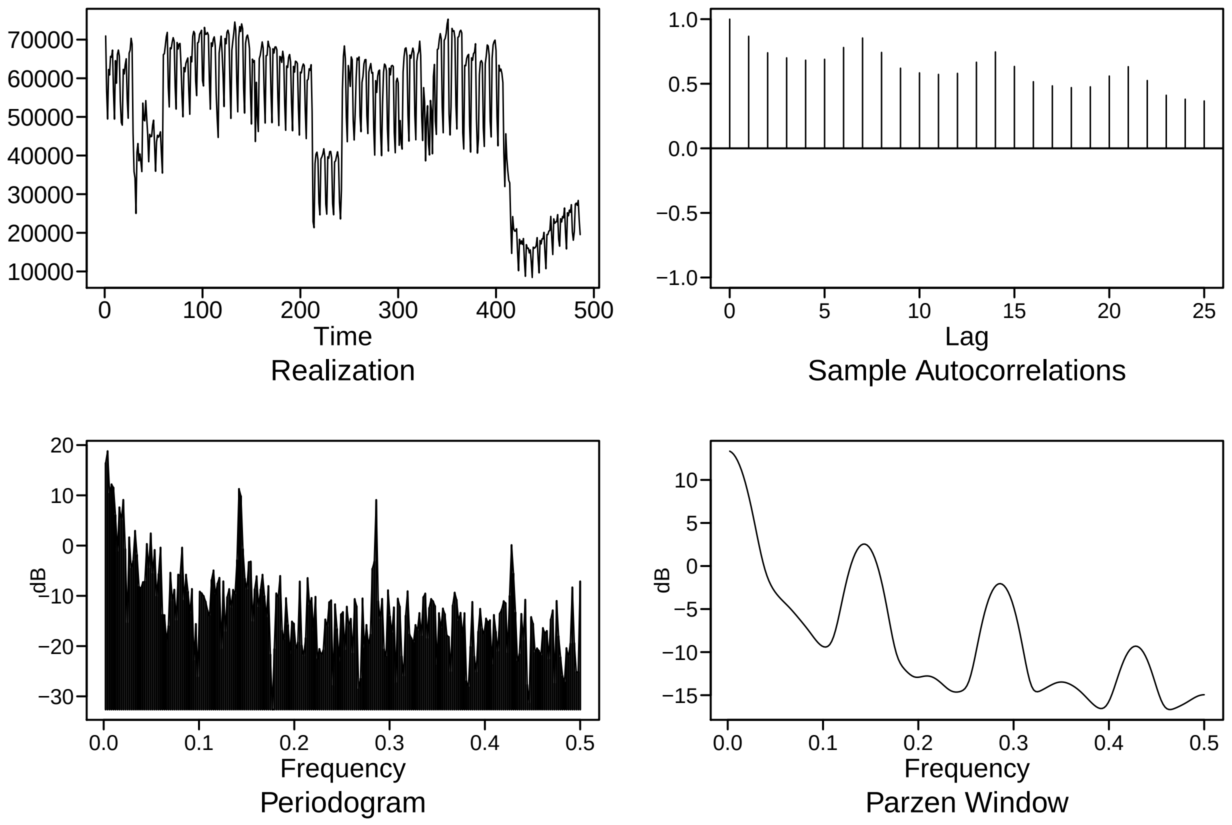
For Hire Vehicles

Now we turn our focus to the FHV and see how drastic the pandemic was for this service in comparison. As mentioned before, we will only focus on post-February 2019 due to the policy changes within the city. Using this limited time frame also allows us to use 350 lags and keep a consistent length for both series.

```r
#i respecify the indicies to make sure
inds <- seq(as.Date("2019-02-01"), as.Date("2020-05-31"), by = "day")
ts.fhv <- ts(FHV_total$n[397:882], start = c(2019, as.numeric(format(inds[1], "%j"))),
           frequency = 365)
```

Spectral Analysis

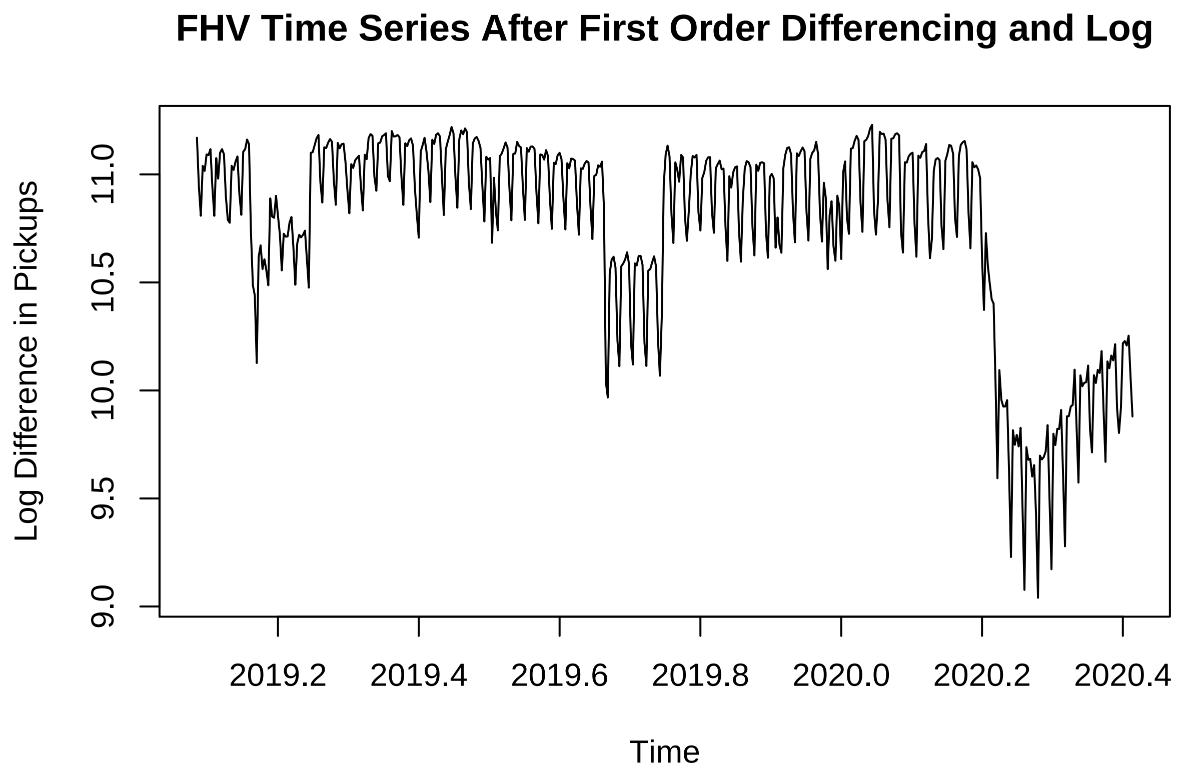
```r
spectral<-plotts.sample.wge(ts.fhv)
```



In the spectral analysis of truncated series, we see some very similar things to the yellow cab analysis such as the large spikes in frequency corresponding to weekly cycles and the large spike around 0. The time series plot itself is interesting as there are random spikes and decreases seen at times ~30 and ~200. This suggests it may be difficult to model.

We use a log transform for the inconsistent variance throughout.

```r
log.ts.fhv<-log(ts.fhv)
plot.ts(log.ts.fhv, ylab = "Log Difference in Pickups")
title("FHV Time Series After First Order Differencing and Log")
```



The log transformation seems to really help with the mutliplciative effect. Since there are obvious needs for some pretty severe differencing, we just fit the optimal SARIMA using the same function as above.

```r
library(doParallel)
cl <- parallel::makeCluster(3, setup_strategy = "sequential")
registerDoParallel(cl)
#define our get best arima with seasonal
get.best.arima <- function(x.ts, maxord=c(1,1,1,1,1,1))
 {
   best.aic <- 1e8
   n <- length(x.ts)
   for(p in 0:maxord[1]) for (d in 0:maxord[2]) for (q in 0:maxord[3])
     for(P in 0:maxord[4]) for (D in 0:maxord[5]) for (Q in 0:maxord[6])
      {
        fit <- arima(x.ts, order = c(p,d,q),
                         seas = list(order = c(P,D,Q),
                         frequency(x.ts)), method = "CSS")
        fit.aic <- -2 * fit$loglik + (log(n) + 1) * length(fit$coef)
        if (fit.aic < best.aic)
       {
          best.aic <- fit.aic
          best.fit <- fit
          best.model <- c(p,d,q,P,D,Q)
        }
   }
   list(best.aic, best.fit, best.model)
}


get.best.arima(log.ts.fhv[43:393],maxord = c(5,5,5,5,5,5))
```

```
stopCluster(cl)
```

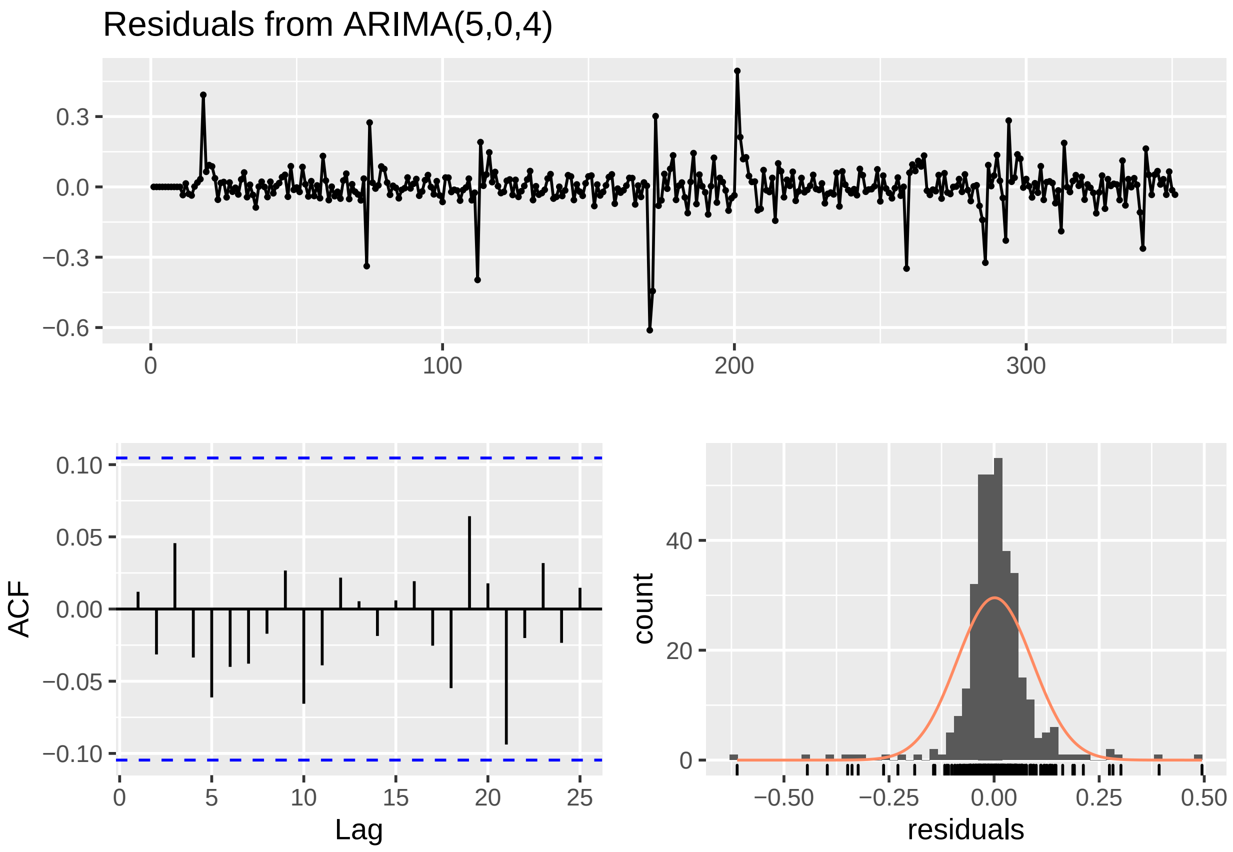
Results

We see the results are an SARIMA(5,0,4,4,1,5)

```
opt.fhv<-arima(log.ts.fhv[43:393],order=c(5,0,4), seasonal = list(order=c(4,1,5),frequency(fitting.taxi
```

```
## Warning in arima(log.ts.fhv[43:393], order = c(5, 0, 4), seasonal = list(order =
## c(4, : possible convergence problem: optim gave code = 1
```

```
checkresiduals(opt.fhv)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(5,0,4)
## Q* = 13.246, df = 3, p-value = 0.004134
##
## Model df: 18.   Total lags used: 21
```

Although there does seem to be some random spikes in the residuals, it again may be due to something like holidays. We again follow up with a Ljung-Box test.

```
Box.test(opt.fhv$residuals, lag = 1, type = c( "Ljung-Box"))
```

```
##
##  Box-Ljung test
##
## data:  opt.fhv$residuals
## X-squared = 0.050495, df = 1, p-value = 0.8222
```

We again fail to reject the null hypothesis that there is autocorrelation in the residuals. We therefore believe that these predictions will be reasonable and continue with the analysis.

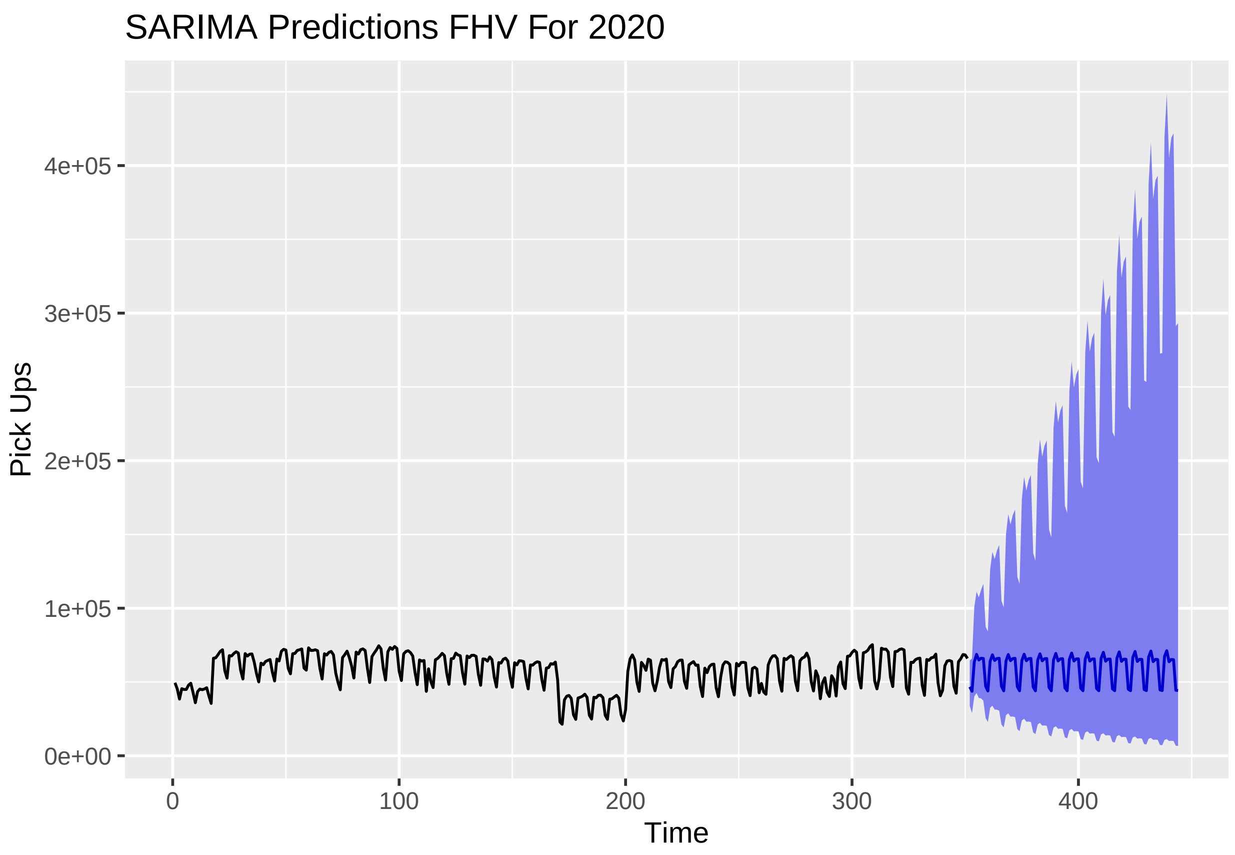
We use the same analysis getting the forecast with a confidence interval at the 99.9% confidence level, but this time for FHV.

```
forecast.fhv<-forecast(opt.fhv,level=c(99.9),h=93)
```

```
## Warning in predict.Arima(object, n.ahead = h): MA part of model is not
## invertible
```

```
## Warning in predict.Arima(object, n.ahead = h): seasonal MA part of model is not
## invertible
```

```
forecast.fhv$x <-exp(forecast.fhv$x)
forecast.fhv$fitted <-exp(forecast.fhv$fitted)
forecast.fhv$mean<-exp(forecast.fhv$mean)
forecast.fhv$upper<-exp(forecast.fhv$upper)
forecast.fhv$lower<-exp(forecast.fhv$lower)
autoplot(forecast.fhv, main="SARIMA Predictions FHV For 2020", ylab='Pick Ups', color= 'red')
```



```
lowest.lowerbound.value<-min(forecast.fhv$lower)
```

Interestingly, the model does not have very strong confidence and has a pretty large interval, especially in the increasing direction.

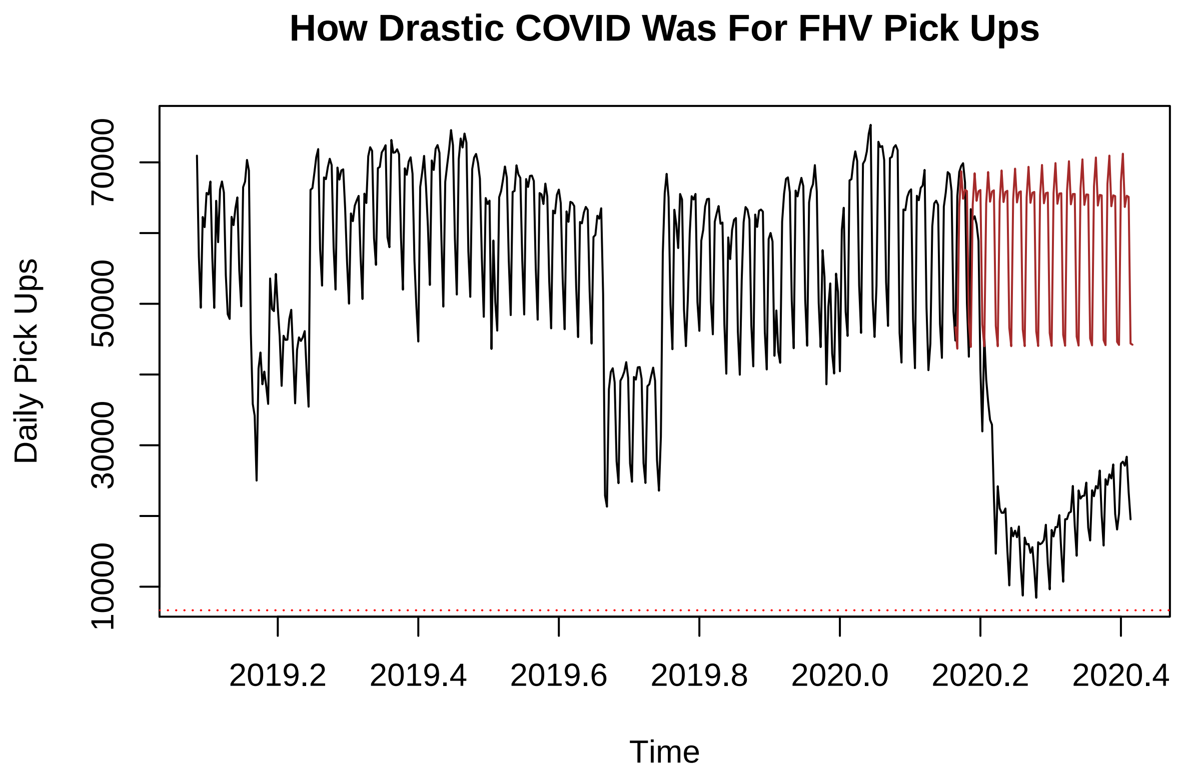
We see the lowest points of the 99.9% confidence intervals through May are around 6,668 daily pick ups.

```
#i make the prediction
pred.fhv<-predict(opt.fhv, n.ahead=93)
```

```
## Warning in predict.Arima(opt.fhv, n.ahead = 93): MA part of model is not
## invertible
```

```
## Warning in predict.Arima(opt.fhv, n.ahead = 93): seasonal MA part of model is
## not invertible
```

```
# i adjust for the log transform
pred.fhv$pred<- exp(pred.fhv$pred)
pred.inds <- seq(as.Date("2020-03-01"), as.Date("2020-05-31"), by = "day")
pred.ts.fhv <- ts(pred.fhv$pred, start = c(2020, as.numeric(format(pred.inds[1], "%j"))),
           frequency = 365)

#the plot
ts.plot(ts.fhv,pred.ts.fhv,col=c("black","brown"), main="",ylab="Daily Pick Ups")
abline(h=lowest.lowerbound.value, lty=3, col = c("red"))
title("How Drastic COVID Was For FHV Pick Ups")
```



When we superimposed the predictions and minimum confidence interval bound on the full time series for the FHV, we see that these COVID-19 drop offs are actually WITHIN the 99.9% confidence intervals for the FHV vehicles!

This suggests that the for hire vehicle industry may have been able to actually account for and predict losses as large as the once seen from the pandemic.

For the FHV, we also see a relatively quick increase in pick ups from March to May 2020, this may because of how resilient the FHV industry has proven to be. After taking those huge losses in early 2019 from the NYC policy changes, maybe FHVs have been more prepared for future loses.

Conclusion

Our analysis shows in a visual manor that yellow cabs have been hit harder by the 2020 pandemic than their for hire vehicle counter parts. This is in terms of the sheer decrease in the number of pick ups and how unpredictable the loses were for the yellow cabs. What was a true black swan event for the yellow cab industry, actually appeared to be relatively predictable for the for hire vehicle companies.

In the future, it appears that for hire vehicles will again retake control over the vehicle service industry as they did before the 2019 NYC policy change.

Predictions

Regression

For the regression task, I fit four different neural networks, one for each year used in the prediction. The data for each year was the corresponding month and year included in the test set data. In my training set data which was 500,000 rows from the same month as the prediction set I did an 80/20 test split. For each set, I feature engineered date and time variables to get a variable about which hour it was in the day and created 4 different flag variables. The flag variables denoted if it was late night/past 5pm, if the drive was over a mile, under 5 minutes or over ten mintes. I centered and scaled all numeric variables.

Using tensorflow and Keras, I used keras sequential API to fit a three layer neural network for each year. The neural network used gloral normalization for first layer weights and had 12 neurons on the first layer. I used relu activation functions for all layers other than the final layer which I used linear. The second layer had 5 units and the final layer had 1 single unit.

I trained the model using RMS Prop with a mean squared error activation function. The model was training for 30 epochs with a minibatch size of 128. For each epoch the training set was split into a 80/20 training and validation set to monitor overfitting.

The model was then used to predict the test set and test RMSEs can be seen below. The models were used to predict the same year they were fit on for the final data predictions saved in CSVs.

List of variables included:

hour,passenger_count,trip_distance,fare_amount,dropoff_longitude,dropoff_latitude, day,surcharge,late_night,over_mile,o

RMSE on test set below (additonal cleaning done to just that year)

TEST SET RMSE

2020 : 1.647215 (involved outlier removal did not use long and lat)

2018 : 1.62216 (did not use long and lat)

2011 : 1.3708

2009 : 1.520322

Classification

For classification, I again used neural networks with the same feature engineering. The main differences in this neural network was the number of neurons for each layer were 12, 4, 2 and the activation functions were relu, relu, sigmoid respectively. The optimizer for the classification was adam and the model was trained using a binary cross entropy loss function. This model only was trained for 5 epochs but with a minibatch size of 25.

List of vars included :

hour,passenger_count,trip_distance,fare_amount,tip_amount,late_night,over_mile,over_ten, over_five,

TEST SET ACCURACY

2020:

Accuracy : 0.9071

Sensitivity : 0.8297

Specificity : 0.9848

2019:

Accuracy : 0.8651

Sensitivity : 0.8033

Specificity : 0.9432

2017:

Accuracy : 0.9207

Sensitivity : 0.8411

Specificity : 0.9997

2015:

accuracy 0.9317

Sensitivity : 0.8622

Specificity : 0.9979