Time Series Final Project

Yixuan (Sherry) Wu and Rachel Tao

December 8, 2019

Contents

1 Abstract

Our final project focuses on the question of what predictors influence metro traffic volume between two metropolitan cities? This is an everyday problem that many Americans face, so we hope to discover the root of this phenomenon that causes much anguish and distress to many people. We are using traffic and weather data to analyze traffic volume between Minnesota and St. Paul, Minnesota and focusing on the years 2012 to 2014. We first explore the our dependent variable, metro volume, and our weather/days-related predictors. To build our final model, we tested for stationary and autocorrelation. The results showed that our time series object has evidence of positive autocorrleation and can tolerate non-stationarity. Thus, we started with an ARIMA model and progressed to a SARIMA model because of seasonality in the ACF plot of residuals in the ARIMA model. Once we had a final model, we tested it with observations in our dataset from 2015 to 2018. We were able to confirm that our model works for the entire dataset. Lastly, we went through permutations of our model to find our final regression model, followed by forecasting using the dataset from 2015 to 2018. Our analyses showed that rain, holiday, and temperature are significant predictors for metro traffic volume between Minnesota and St. Paul.

2 Introduction

Today, the average U.S. commuter spends about 42 hours in traffic. This is often draining and tiring for the commuter, as no one enjoys sitting in a car or bus for a very long time. There are many reasons for traffic, ranging from bad drivers to the weather. The traffic volume between populated cities in the state can also worsen, as there are more people on the road and those cities usually house many jobs for these commuters. And so, our project is identifying which predictors influence daily 94 westbound metro traffic volume between Minneapolis and St. Paul, Minnesota, as those are the most populated cities in Minnesota. Our independent variables are rain, snow, cloud, temperature, and holiday; our dependent variable is traffic volume.

3 Methods and Materials

3.1 Summary of Problem and Data

The goal of this project is to identify which predictors, among rain, snow, cloud, temperature, and holiday, influence daily metro traffic volume in the route between Minneapolis and St. Paul, MN. The dataset is retrieved from the UCI Machine Learning Repository. The contributors have cited that the dataset is aggregated using traffic data from the Minnesota Transportation Department and weather data from WeatherMap. The original data is in hourly form from 2012 to 2018. However, due to missing hours and varying number of completed hours everyday, we aggregated the dataset into a daily form. We calculated the average of the daily values of the variables by taking the average of the hourly values in a day. There are some hours that have a temperature of 0, and this is impossible because the temperature has a unit of Kelvin. Therefore, when aggregating the data, we dropped all temperature observations with a value of 0 Kelvin. We assumed that this value of 0 represents the fact that the temperature information is missing for that observation. The

`holiday` variable is a categorical value that takes on the values "yes" or "no" depending if the day is a holiday or not. We also refined the data to keep only observations from 2012 to 2014 to have a total of 661 observations. We did this because there are no observations from August 2014 to June 2015. We kept the rest of the dataset from 2015 to 2018 as the test data to assess model reliability in the end.

The data from 2012 to 2014, after aggregation, still has 15 missing days. For those days, we added `NA`s in all other variables so a more accurate `ts` object can be constructed later.

```
Metro = read_csv("Metro_Interstate_Traffic_Volume.csv")
MetroRemoved = Metro[duplicated(Metro$date_time) == F,]
MetroRemoved = MetroRemoved[year(MetroRemoved$date_time) <2015,]

###aggregate by date:
MetroRemoved$ho = MetroRemoved$holiday
MetroRemoved$ho[MetroRemoved$holiday == "None"] = 0
MetroRemoved$ho[MetroRemoved$holiday != "None"] = 1
MetroRemoved$ho = as.numeric(MetroRemoved$ho)

MetroNew = MetroRemoved %>%
  group_by(date(MetroRemoved$date_time)) %>%
  summarise(Volume = mean(traffic_volume),
            Rain = mean(rain_1h),
            Snow = mean(snow_1h),
            Cloud = mean(clouds_all),
            Temp = mean(temp[temp != 0]),
            Holiday = if(mean(ho) > 0){1}else{0})

names(MetroNew)[1] = "Date"

###check if any day is missing:
DateRange = seq(min(MetroNew$Date), max(MetroNew$Date), by = 1)
MissingDate = DateRange[!DateRange %in% MetroNew$Date ]

MetroNew$Date = as.numeric(MetroNew$Date)

for (i in 1:length(MissingDate))
  {
    newrow = as.numeric(c(MissingDate[i], rep(NA, 6)))
    MetroNew = rbind(MetroNew, newrow)
  }
```

```
MetroNew$Date = as.Date(MetroNew$Date)
##Reorder the data by date
MetroNew = MetroNew %>% arrange(Date)
```

3.2 Dependent Variables

The variable of interest, as mentioned above, is the daily metro traffic volume of the route between Minneapolis and St. Paul, MN. The minimum volume is 27.2, with a median volume of 3479, a mean volume of 3292, and a maximum volume of 4681 as shown in the summary. From the histogram and the box plot (*Figure 1*), the volume is left skewed and has about 6 low outliers.

A time series object is constructed with the start at 2012.75, which refers to October 2, 2012, with a frequency of 366, since year 2012 has 366 days. The observations that contain `NA`s are removed. The time series plot is shown in *Figure 2* . It looks relatively stationary, with a decreasing pattern around late October to the end of 2013. There is a gradual decrease to the end of 2013. There are gradual increasing patterns around January to May 2014, and around January to late June 2013. It is hard to tell from the graph if the variability in the data overwhelms such patterns, so a formal test of stationarity is necessary. Since the data is daily, a seasonality with period of 7 days is expected, but it is hard to visualize from this graph. An ACF plot the time series object (*Figure 3*) more clearly shows this seasonality. Since the `frequency` component in the ts object is set to be 365, a lag of 0.018 confirms that there is a seasonality with a period of `0.018*365 = 6.57`, roughly a week. In the same figure of the time series plot (*Figure 2*), the smoothers are also shown. The cubic smoother does not provide much information, as it outputs a straight line. The kernel smoother is better than the lowess smoother, because the lowess smoother, while more flexible, still fails to capture the decreasing trend toward May 2014.

```
y.histogram = ggplot(MetroNew) +
  geom_histogram(aes(x = Volume), fill = "#044bc4", bins = 40) +
```

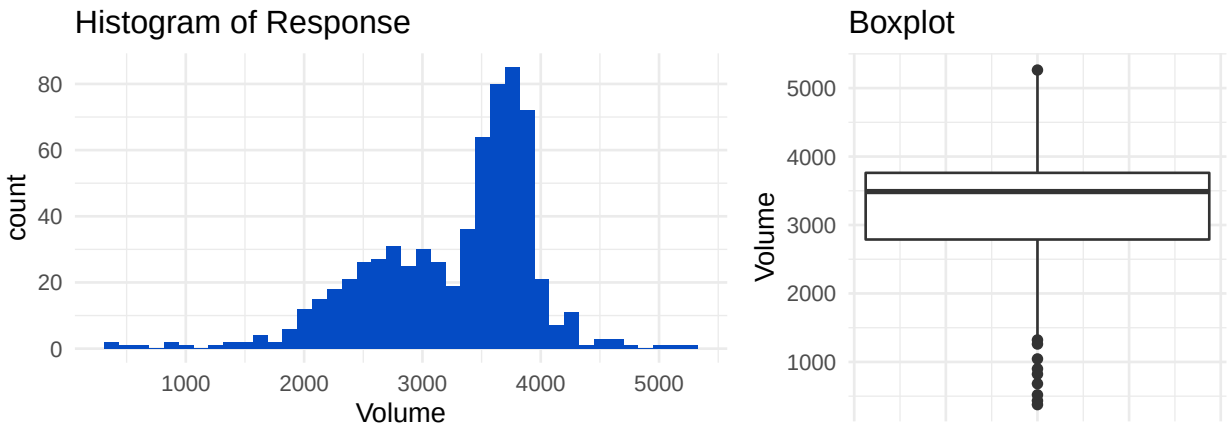

Figure 1: Histogram and Boxplot of Daily Traffic Volume of the Route between Minneapolis and St. Paul, MN

```
  labs(title = "Histogram of Response") +
  theme_minimal()

y.bboxplot = ggplot(MetroNew) +
  geom_boxplot(aes(y = Volume)) +
  labs(title = "Boxplot") +
  theme_minimal() +
  theme(axis.text.x = element_blank())

grid.arrange(y.histogram, y.bboxplot, ncol = 2, widths = c(1.5,1))
```

```
Metro.ts = ts(MetroNew$Volume, start = c(2012.75), frequency = 366)
Metro.ts = na.remove(Metro.ts)
```

```
tsplot = ggplot() +
  geom_line(aes(x = time(Metro.ts), y = Metro.ts), color = "#044bc4") +
  scale_y_continuous() +
  scale_x_continuous() +
  labs(y = "Metro Volume", x = element_blank(),
       title = "TS Plot of Volume") +
  theme_minimal()

ts.Kernel = ksmooth(time(Metro.ts),
                    Metro.ts, "normal", bandwidth = 0.2)
ts.Lowess = lowess(Metro.ts, f= 0.15)
difftime = time(Metro.ts) - mean(time(Metro.ts))
ts.Cubic = lm(Metro.ts ~ difftime + (difftime)^2 + (difftime)^3)
```

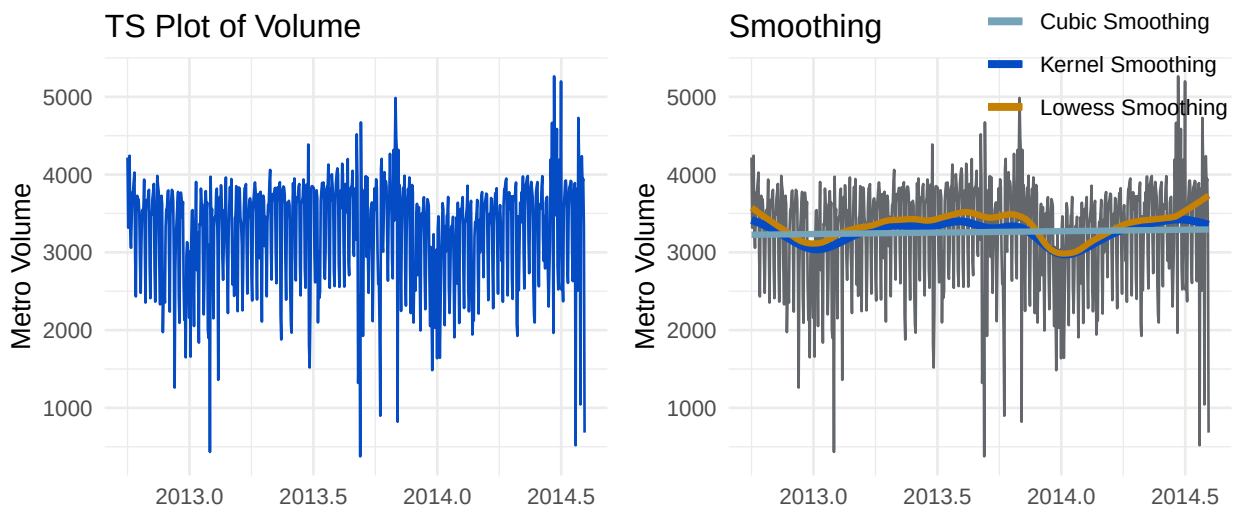

Figure 2: Time Series Plot of the Dependent Variable

```
expts = ts(MetroNew$Volume, start = c(2012, 10,2))
expts = na.remove(expts)

smooth = ggplot() +
  geom_line(aes(x = time(Metro.ts), y = Metro.ts), color = "#63666A") +
  geom_line(aes(x = ts.Kernel$x, y = ts.Kernel$y,
                color = "Kernel Smoothing"), size = 1.5) +
  geom_line(aes(x = ts.Lowess$x, y = ts.Lowess$y,
                color = "Lowess Smoothing"), size = 1.2) +
  geom_smooth(method = "lm", aes(x = time(Metro.ts), y = Metro.ts,
                                 color = "Cubic Smoothing"),
              formula = (y~x + x^2 + x^3), size = 1.2, se = F) +
  labs(x = element_blank(), y = "Metro Volume", title = "Smoothing ") +
  scale_y_continuous() +
  scale_x_continuous() +
  scale_color_manual(values = c("#75A4B9", "#044bc4", "#c48104")) +
  theme_minimal() +
  theme(legend.title=element_blank(), legend.position = c(0.75,1))

grid.arrange(tsplot, smooth, nrow = 1)
```

```
autoplot(acf(Metro.ts, plot = FALSE)) + theme_minimal()
```

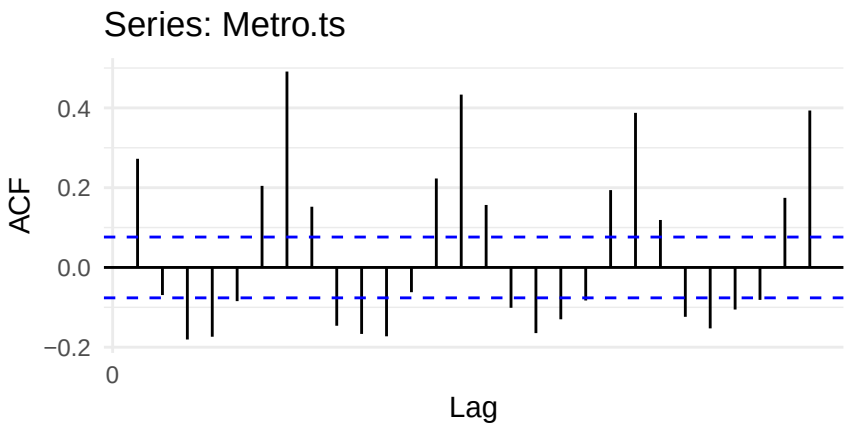

Figure 3: ACF of the Time Series

3.3 Independent Variables

The independent variables are `Rain`, `Snow`, `Cloud`, `Temp`, and `Holiday`. `Rain` is skewed strongly to zero, as shown from the summary statistics (*Table 1*). Out of 676 observations, 502 observations have a 0 for rain with a mean of 0.11 and a maximum of 4.68. It has a right skewed distribution with many outliers according to the histogram and boxplot (*Figure 4*). All observations of `Snow` from 2012 to 2014 are 0s, which is unusual for Minnesota, located in the Midwest. Therefore, this variable is excluded from further analysis. `Cloud` has a minimum of 0 and a maximum of 93, with a mean of 48.41 and a median of 52. It is close to a uniform distribution and is a little skewed left without any outliers, according to the histogram and the boxplot (*Figure 5*). `Temp` refers to temperature. It has a minimum of 249.0 Kelvin (-11.47 Fahrenheit) and a maximum of 303.3 Kelvin (85.73 Fahrenheit), with a mean of 40.73 F a median of 38.93 F. The histogram and boxplot (*Figure 6*) show that `Temp` has bi-modal distribution and right skewed, without any outliers. Lastly, `Holiday` is a categorical variable with two levels, 0, meaning that there is no holiday, and, 1, meaning that there is a holiday. Out of the 676 observations, 643 are days without holidays (*Table 2*).

```
kable(summary(MetroNew[,c(3)]), booktabs = TRUE,
  caption = 'Summary Table for Rain')
```

Table 1: Summary Table for Rain

	Rain
	Min. :0.00000
	1st Qu.:0.00000
	Median :0.00000
	Mean :0.11296
	3rd Qu.:0.01042
	Max. :4.67563
	NA's :15

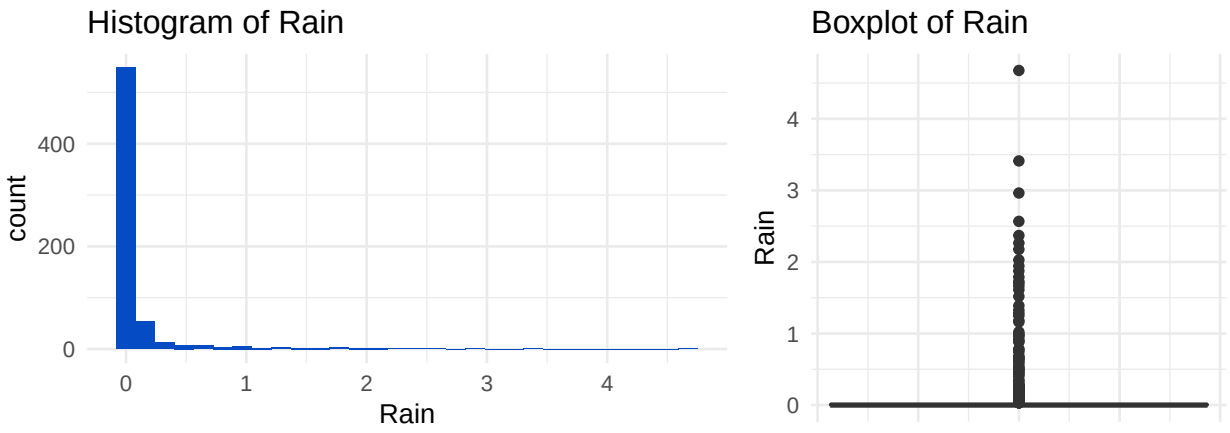

Figure 4: Histogram and Boxplot of Rain

```
rain.hist = ggplot(MetroNew, aes(x = Rain)) +
  geom_histogram(fill = "#044bc4") +
  labs(title = "Histogram of Rain") +
  theme_minimal()

rain.box = ggplot(MetroNew, aes(y = Rain)) +
  geom_boxplot() +
  labs(title = "Boxplot of Rain") +
  theme_minimal() +
  theme(axis.text.x = element_blank())

grid.arrange(rain.hist, rain.box, ncol = 2, widths = c(1.5,1))
```

```
holidaytable = table(MetroNew$Holiday)
kable(holidaytable, booktabs = TRUE, caption = "Table for Holiday")
```

Table 2: Table for Holiday

Var1	Freq
0	643
1	18

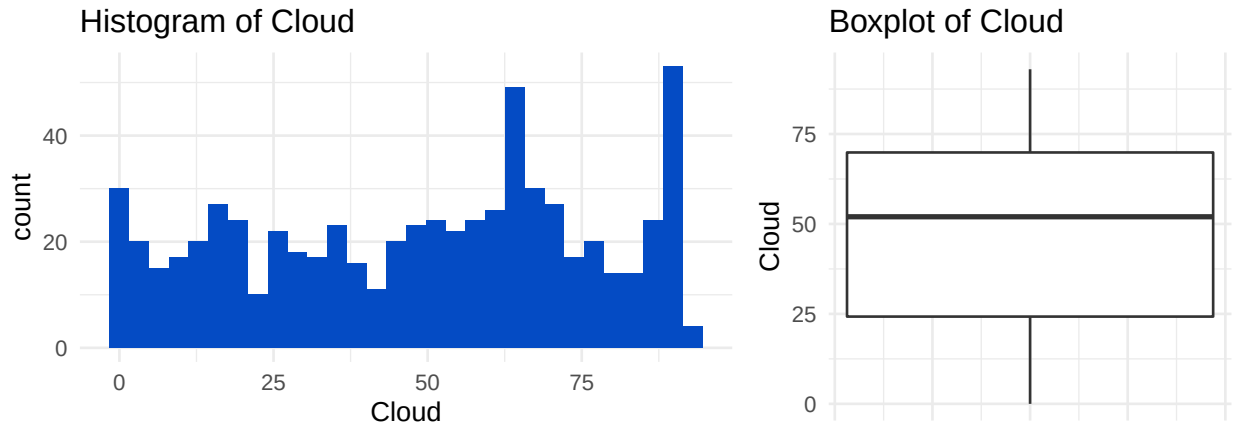

Figure 5: Histogram and Boxplot of Variable Cloud

```
cloud.hist = ggplot(MetroNew, aes(x = Cloud)) +
  geom_histogram(fill = "#044bc4") +
  labs(title = "Histogram of Cloud") +
  theme_minimal()

cloud.box = ggplot(MetroNew, aes(y = Cloud)) +
  geom_boxplot() +
  labs(title = "Boxplot of Cloud") +
  theme_minimal() +
  theme(axis.text.x = element_blank())

grid.arrange(cloud.hist, cloud.box, ncol = 2, widths = c(1.5,1))
```

```
temp.hist = ggplot(MetroNew, aes(x = Temp)) +
  geom_histogram(fill = "#044bc4") +
  labs(title = "Histogram of Temperature") +
  theme_minimal()

temp.box = ggplot(MetroNew, aes(y = Temp)) +
  geom_boxplot() +
  labs(title = "Boxplot of Temp") +
  theme_minimal() +
```

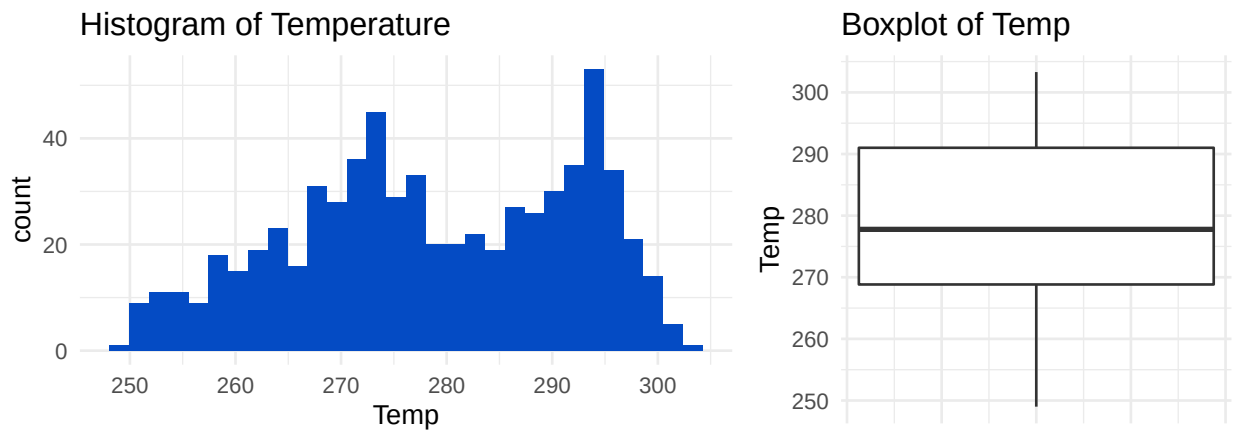

Figure 6: Histogram and Boxplot of Temperature

```
  theme(axis.text.x = element_blank())

grid.arrange(temp.hist, temp.box, ncol = 2, widths = c(1.5,1))
```

3.4 Procedures

After the exploratory analysis of both the independent and dependent variables, we will fit models to capture the relationship between independent and dependent variables. We will start with testing for autocorrelation and stationarity to start model selection process for Metro Volume, the dependent variable. We will then compare different models using criteria including AIC, BIC, sigma^2. Then, the selected models' adequacy will be evaluated by looking at the ACF plots. After a model is selected, we will use the model to perform forecasting. The reliability of the model will also be tested using both the train data by removing some values and test data. Lastly, with a model that can account for potential autocorrelation, trend, and seasonality, metro volume will be fitted against the independent variables. A permutation of all potential models will be performed, and the models will be selected based on criteria including AIC, and BIC. The final model's adequacy will again be evaluated. Then, forecasting with regressions will be performed.

4 Results:

4.1 Test of Autocorrelation and Stationarity:

To test the autocorrelation of the dependent variable, a Durbin-Watson Test (a null hypothesis that true autocorrelation is 0; an alternative hypothesis that the true autocorrelation is greater than 0) shows a DW statistic of 1.4325 ($p = 1.024e - 13$). Thus, we reject the null hypothesis and conclude that there is positive autocorrelation between time and metro volume.

We then test the stationarity of the time series using the Dickey-Fuller test (a null hypothesis that there is non-stationarity; an alternative hypothesis that we can tolerate non-stationarity in the time series). With a Dickey-Fuller statistic of -7.0345 ($p = 0.01$), we reject the null hypothesis and conclude that the time series can tolerate non-stationarity. Thus, there is no need to perform a model transformation to stationarize the series, meaning that when fitting the model, we can set the differencing component to be 0.

```
lmmetro = lm(Metro.ts ~ time(Metro.ts))
lmtest::dwtest(lmmetro)
##
##  Durbin-Watson test
##
## data:  lmmetro
## DW = 1.4325, p-value = 1.024e-13
## alternative hypothesis: true autocorrelation is greater than 0
adf.test(Metro.ts)
##
##  Augmented Dickey-Fuller Test
##
## data:  Metro.ts
## Dickey-Fuller = -7.0345, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

4.2 Model Selection

4.2.1 ARMA Model

All permutations of the ARMA model with order up to 3 are fitted. From *Table 3*, one can conclude that ARMA(2,2) has the lowest AIC value. This is confirmed with autoselection. An ACF plot of ARMA(2,2) is shown in *Figure 7*. One can still see seasonality pattern in this ACF plot. Therefore, it is necessary to fit seasonlity components in the model as well.

```
arima100 = arima(Metro.ts, order = c(1,0,0))
arima001 = arima(Metro.ts, order = c(0,0,1))
arima101 = arima(Metro.ts, order = c(1,0,1))
arima200 = arima(Metro.ts, order = c(2,0,0))
arima002 = arima(Metro.ts, order = c(0,0,2))
arima102 = arima(Metro.ts, order = c(1,0,2))
arima201 = arima(Metro.ts, order = c(2,0,1))
arima300 = arima(Metro.ts, order = c(3,0,0))
arima003 = arima(Metro.ts, order = c(0,0,3))
arima301 = arima(Metro.ts, order = c(3,0,1))
arima103 = arima(Metro.ts, order = c(1,0,3))
arima202 = arima(Metro.ts, order = c(2,0,2))

arimas = as.table(matrix(c(arima100$aic, arima001$aic, arima101$aic,
                           arima200$aic, arima002$aic, arima102$aic,
                           arima201$aic, arima202$aic, arima300$aic,
                           arima003$aic, arima301$aic, arima103$aic)))

arimanames = c("arima100", "arima001",
               "arima101", "arima200",
               "arima002", "arima102",
               "arima201", "arima202",
               "arima300", "arima003",
               "arima301", "arima103")

rownames(arimas) = arimanames
colnames(arimas) = "AIC"

kable(arimas, booktabs = TRUE,
  caption = 'AIC of potential ARMA models')
```

Table 3: AIC of potential ARMA models

	AIC
arima100	10477.00
arima001	10467.88
arima101	10469.86
arima200	10462.04
arima002	10469.83
arima102	10467.78
arima201	10453.41
arima202	10356.52
arima300	10452.80
arima003	10458.53
arima301	10451.19
arima103	10460.01

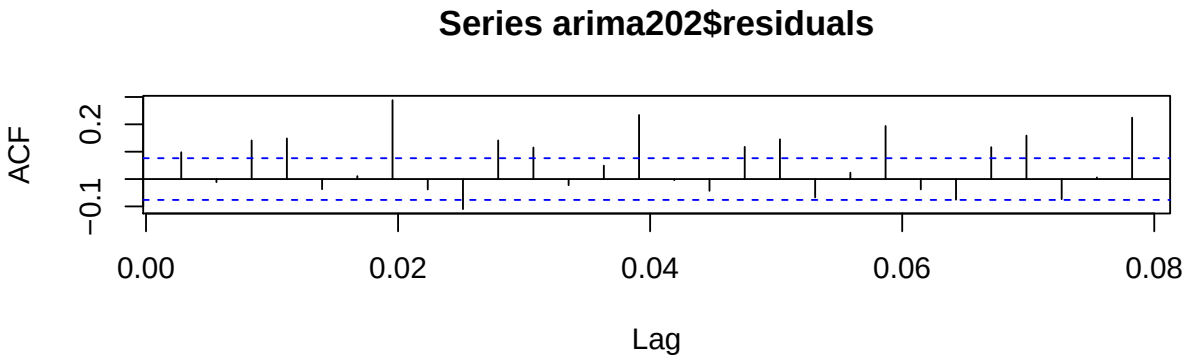

Figure 7: ACF of the ARIMA(2,0,2) Model

```
acf(arima202$residuals)
```

```
auto.arima(Metro.ts)
## Series: Metro.ts
## ARIMA(2,0,2) with non-zero mean
##
## Coefficients:
##          ar1      ar2      ma1     ma2       mean
##       1.2245  -0.9790  -1.1368  0.8678  3258.7636
## s.e.  0.0127   0.0108   0.0387  0.0283    22.8269
##
```

```
## sigma^2 estimated as 369570:  log likelihood=-5173.26
## AIC=10358.52   AICc=10358.65   BIC=10385.48
```

4.2.2 SARMA Model

We still keep the seasonal differencing component to be 0 because from the time series plot, it is not obvious that the the seasonal component interacts with time. The period of seasonality is 7, because from common sense, we expect that traffic volume has patterns that repeat every week. Also, from the ACF plot of both the time series (*Figure 3*) and the ACF plot of ARMA(2,2) (*Figure 7*), we can see the seasonality with a period of 7.

We then fitted permutations of a SARMA model with orders up to 2. Again, SARMA(2,2)(1,1)7 has the lowest AIC value (*Table 4*). An ACF plot of SARMA(2,2)(1,1)7 is shown in *Figure 8*. From the ACF plot, only one lag at 14 is still a little significant. However, this lag can be significant due to random chance, and it is also not worth it to go up to order 14 just to capture the significance of this one lag. Therefore, the final model chosen is SARMA(2,2)(1,1)7.

```
sarima100 = arima(Metro.ts, order = c(2,0,2),
                  seasonal = list(order = c(1,0,0), period = 7))
sarima001 = arima(Metro.ts, order = c(2,0,2),
                  seasonal = list(order = c(0,0,1), period = 7))
sarima101 = arima(Metro.ts, order = c(2,0,2),
                  seasonal = list(order = c(1,0,1), period = 7))
sarima200 = arima(Metro.ts, order = c(2,0,2),
                  seasonal = list(order = c(2,0,0), period = 7))
sarima002 = arima(Metro.ts, order = c(2,0,2),
                  seasonal = list(order = c(0,0,2), period = 7))
sarima201 = arima(Metro.ts, order = c(2,0,2),
                  seasonal = list(order = c(2,0,1), period = 7))
sarima102 = arima(Metro.ts, order = c(2,0,2),
                  seasonal = list(order = c(1,0,2), period = 7))
sarima202 = arima(Metro.ts, order = c(2,0,2),
                  seasonal = list(order = c(2,0,2), period = 7))

sarimas = as.table(as.matrix(c(sarima100$aic, sarima001$aic, sarima101$aic,
                               sarima200$aic, sarima002$aic, sarima102$aic,
```

Table 4: AIC of Potential SARMA Models

	AIC
sarima100	10302.20
sarima001	10313.08
sarima101	10227.36
sarima200	10264.59
sarima002	10291.94
sarima102	10229.23
sarima201	10229.23
sarima202	10231.26

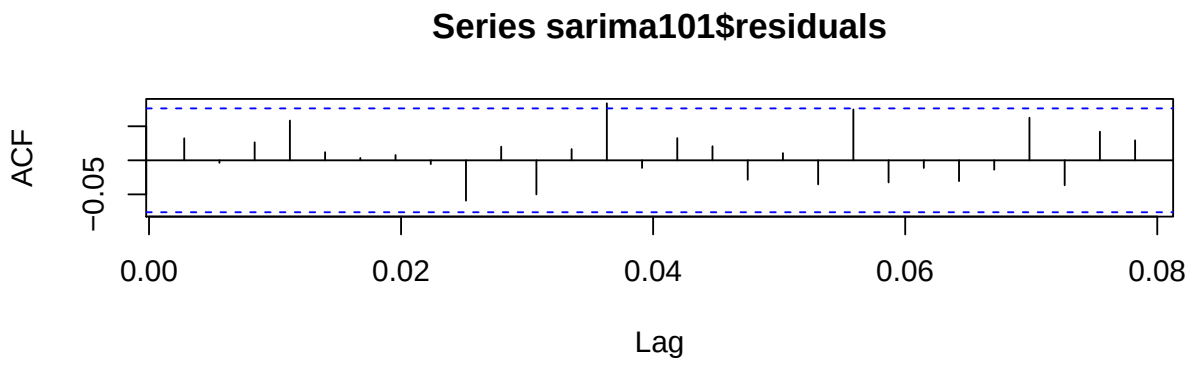

Figure 8: ACF Plot of ARMA(2,0,2) Model with Seasonal Component of (1,0,1)7

```
                                sarima201$aic, sarima202$aic)))

sarimanames = c("sarima100", "sarima001", "sarima101", "sarima200",
                "sarima002", "sarima102", "sarima201", "sarima202")
rownames(sarimas) = sarimanames
colnames(sarimas) = "AIC"
kable(sarimas, booktabs = TRUE, ncol(5),
  caption = 'AIC of Potential SARMA Models')
```

```
acf(sarima101$residuals)
```

Table 5: 1-week ahead forecasting

	Prediction	Lower CI	Higher CI
Day1	1820.455	2891.363	749.5467
Day2	3499.195	4590.296	2408.0942
Day3	3725.111	4818.850	2631.3717
Day4	3491.389	4587.994	2394.7836
Day5	3919.154	5022.709	2815.5989
Day6	3388.536	4493.421	2283.6512
Day7	2793.768	3899.330	1688.2065

4.3 Forecasting with Confidence Interval

Using `predict` from package `forecast`, we can predict the following one week's traffic volume with 95 percent confidence interval. The predicted values are shown in *Table 5*.

```
FinalModel = sarima101
Pred = predict(FinalModel, n.ahead = 7)
Pred.up = Pred$pred + 1.96 * Pred$se
Pred.lo = Pred$pred - 1.96 * Pred$se

predtable = as.table(cbind(Pred$pred, Pred.up, Pred.lo))
rownames(predtable) = c("Day1", "Day2", "Day3", "Day4", "Day5",
                        "Day6", "Day7")
colnames(predtable) = c("Prediction", "Lower CI", "Higher CI")
kable(predtable, booktabs = TRUE,
  caption = '1-week ahead forecasting')
```

We also predicted 1-year ahead metro volume using similar methods but by setting the `n.ahead` to be 365 days. *Figure 9* displays the forecast.

The forecast along with its 95 percent confidence interval converges to the volume mean. This makes sense because the differencing and seasonal differencing components are both 0, thus this is a mean-stationary model, and such pattern is expected.

```
FinalModel = sarima101
Pred = predict(FinalModel, n.ahead = 365)
Pred.up = Pred$pred + 1.96 * Pred$se
Pred.lo = Pred$pred - 1.96 * Pred$se
```

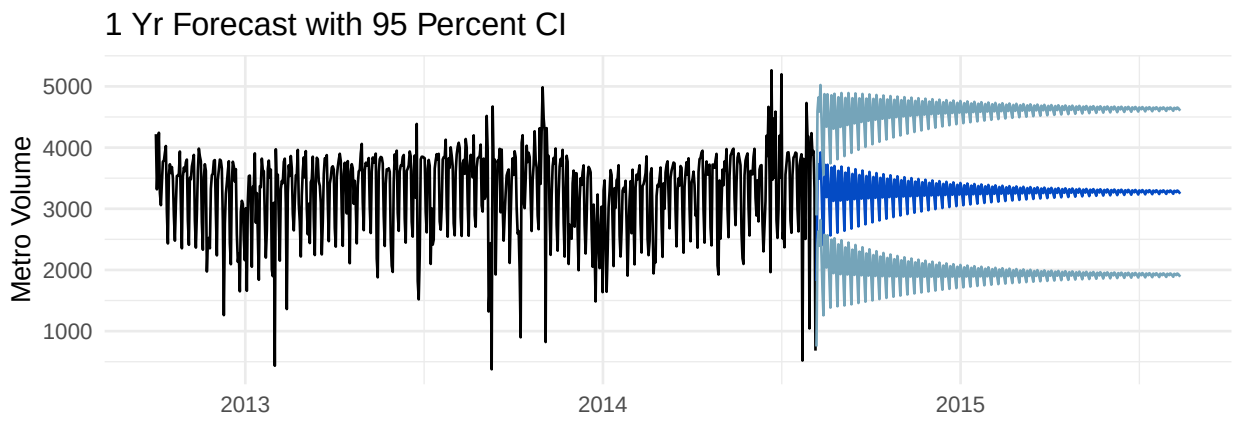

Figure 9: 1 Year Ahead Forecast with 95 Confideince Interval

```
ggplot() +
  geom_line(aes(x = time(Metro.ts), y = Metro.ts)) +
  geom_line(aes(x = time(Pred$pred), y = as.matrix(Pred$pred)),
            color = "#044bc4") +
  geom_line(aes(x = time(Pred$pred), y = as.matrix(Pred.up)),
            color = "#75A4B9") +
  geom_line(aes(x = time(Pred$pred), y = as.matrix(Pred.lo)),
            color = "#75A4B9") +
  labs(x = element_blank(), y = "Metro Volume",
       title = "1 Yr Forecast with 95 Percent CI") +
  scale_x_continuous() +
  scale_y_continuous() +
  theme_minimal()
```

4.4 Model Reliability

4.4.1 Check with Current Data

To verity our fitted model's reliability, we removed 1 month of data and reconstructed the time series object to rebuild the same SARMA model, namely SARMA(2,2)(1,2). The graph is shown in *Figure 10*. The model predicts the first two and half weeks of the last month well. However, since the original data suddenly becomes more variable, the model does not have the ability to capture such sudden change of variability, which is expected. However, when the original data is not more volatile than usual, the model does a good job of forecasting.

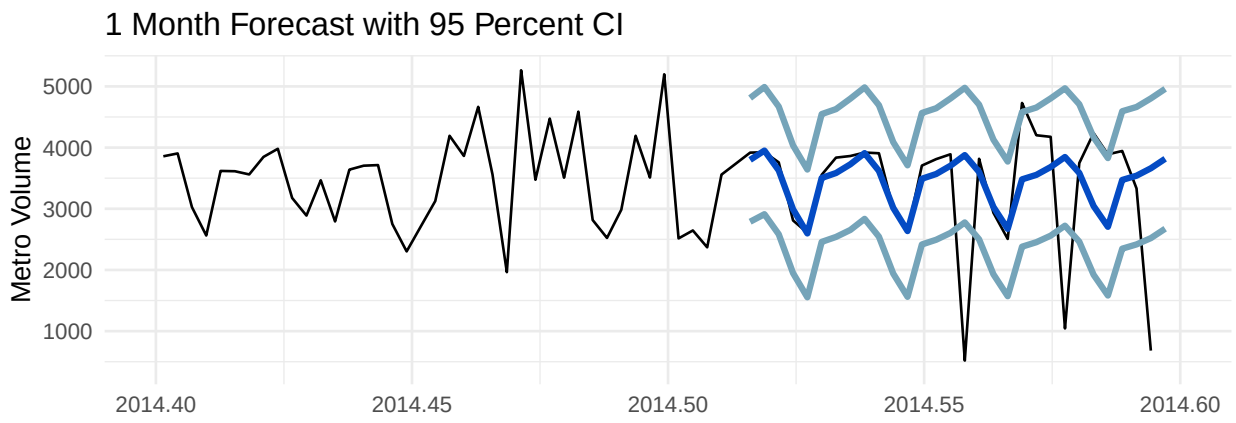

Figure 10: 1 Month Forecast with 95 Percent Confidence Interval

```
FinalModel2 = arima(window(Metro.ts, end = 2014.515), order = c(2,0,2),
                    seasonal = list(order = c(1,0,1), period = 7))
Pred1 = predict(FinalModel2, n.ahead = 30)
Pred1.up = Pred1$pred + 1.96 * Pred1$se
Pred1.lo = Pred1$pred - 1.96 * Pred1$se
```

```
ggplot() +
  geom_line(aes(x = time(Metro.ts), y = Metro.ts)) +
  geom_line(aes(x = time(Pred1$pred), y = as.matrix(Pred1$pred)),
            color = "#044bc4", size = 1.2) +
  geom_line(aes(x = time(Pred1$pred), y = as.matrix(Pred1.up)),
            color = "#75A4B9", size = 1.2) +
  geom_line(aes(x = time(Pred1$pred), y = as.matrix(Pred1.lo)),
            color = "#75A4B9", size = 1.2) +
  labs(x = element_blank(), y = "Metro Volume",
       title = "1 Month Forecast with 95 Percent CI") +
  scale_x_continuous(limits = c(2014.4, 2014.6),
                     breaks = c(2014.4, 2014.45, 2014.5, 2014.55, 2014.6)) +
  scale_y_continuous() +
  theme_minimal()
```

4.4.2 Check with the Test Data (2015-2018)

To further verify the adequacy of our model, we checked it with the data from years 2015 to 2018. This part of the data was dropped in our previous analysis in order to keep our time series object continuous. Now, this serves as our test data. We cleaned the data using similar

methods as before, and removed the NAs in the time series object, which contained 9 NAs.

We built a new time series object that started with 2015.44, namely June 2015. The frequency is set to be 365. This time series is plotted, as shown in *Figure 11*. We also repeated our methods for the entire dataset (2012 to 2018) for when we check with forecasting. The trend and seasonlity looks similar to the previous dataset. There is one part around June 2016 that has a really low metro traffic volume.

```
MetroTest = Metro[duplicated(Metro$date_time) == F,]
MetroTest = MetroTest[year(MetroTest$date_time) >2014,]

MetroTest$ho = MetroTest$holiday
MetroTest$ho[MetroTest$holiday == "None"] = 0
MetroTest$ho[MetroTest$holiday != "None"] = 1
MetroTest$ho = as.numeric(MetroTest$ho)

MetroNewTest = MetroTest %>%
  group_by(date(MetroTest$date_time)) %>%
  summarise(Volume = mean(traffic_volume),
            Rain = mean(rain_1h),
            Snow = mean(snow_1h),
            Cloud = mean(clouds_all),
            Temp = mean(temp[temp != 0]),
            Holiday = if(mean(ho) > 0){1}else{0})

names(MetroNewTest)[1] = "Date"

###check if any day is missing:
DateRangeTest = seq(min(MetroNewTest$Date), max(MetroNewTest$Date), by = 1)
MissingDateTest = DateRangeTest[!DateRangeTest %in% MetroNewTest$Date]
MetroNewTest$Date = as.numeric(MetroNewTest$Date)
for (i in 1:length(MissingDateTest))
  {
    newrow = as.numeric(c(MissingDateTest[i], rep(NA, 6)))
    MetroNewTest = rbind(MetroNewTest, newrow)
  }

MetroNewTest$Date = as.Date(MetroNewTest$Date)
##Reorder the data by date
MetroNewTest = MetroNewTest %>% arrange(Date)
Metro.ts1 = ts(MetroNewTest$Volume, start = c(2015.44), frequency = 365)
Metro.ts1 = na.remove(Metro.ts1)
```

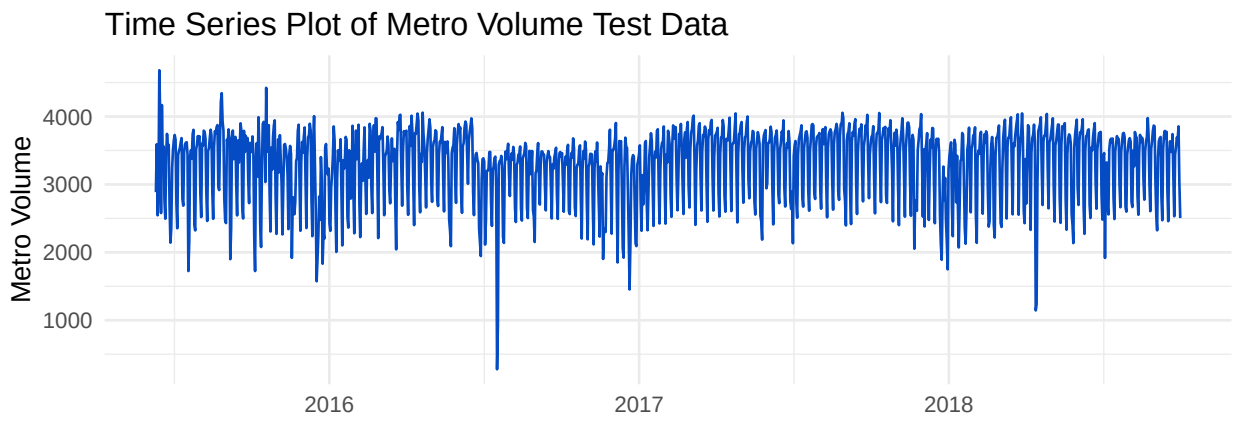

Figure 11: Time Series Plot of the Volume from 2015 to 2018

```
ggplot() +
  geom_line(aes(x = time(Metro.ts1), y = Metro.ts1), color = "#044bc4") +
  scale_y_continuous() +
  scale_x_continuous() +
  labs(y = "Metro Volume", x = element_blank(),
       title = "Time Series Plot of Metro Volume Test Data") +
  theme_minimal()
```

The model fit of a SARIMA model (202)(101) is shown below.

```
m101 = arima(Metro.ts1, order = c(2,0,2),
             seasonal = list(order = c(1,0,1), period = 7))
summary(m101)
##
## Call:
## arima(x = Metro.ts1, order = c(2, 0, 2), seasonal = list(order = c(1, 0, 1),
##     period = 7))
##
## Coefficients:
##          ar1      ar2      ma1      ma2    sar1     sma1  intercept
##       1.0162  -0.1388  -0.5535  -0.2026  0.9844  -0.7849   3302.698
## s.e.  0.1882   0.1013   0.1861   0.0626  0.0055   0.0253    191.684
##
## sigma^2 estimated as 99128:  log likelihood = -8603.36,  aic = 17220.73
##
## Training set error measures:
##                ME RMSE MAE MPE MAPE
## Training set NaN  NaN NaN NaN  NaN
```

The auto arima model suggests a model of ARIMA(5,1,2), and this is a weird suggestion since the `adf.test` results show that the stationarity assumption is met for the time series. To compare the fit, we fitted additional models of ARIMA(2,1,2), ARIMA(5,0,2), and ARIMA(5,1,2). The plots of the four models are shown in *Figure 12*. It is obvious to see the seasonal pattern from the ACF plots of ARIMA(2,1,2) and ARIMA(2,0,2), but not for the models with AR(5). Therefore, we infer that since the seasonality pattern is "accounted for" by model ARIMA(5,1,2), so it has the lowest AIC value according to the `bestAIC` function. However, it is more appropriate to use an ARIMA model with lower orders but including the seasonal components.

```
auto.arima(Metro.ts1)
## Series: Metro.ts1
## ARIMA(5,1,2) with drift
##
## Coefficients:
##          ar1      ar2      ar3      ar4      ar5      ma1     ma2   drift
##       0.1751  -0.7592  -0.2388  -0.3644  -0.5248  -0.6948  0.4397  0.0902
## s.e.  0.0443   0.0281   0.0383   0.0237   0.0329   0.0507  0.0418  2.9213
##
## sigma^2 estimated as 135969:  log likelihood=-8778.33
## AIC=17574.67   AICc=17574.82   BIC=17620.46
```

```
nsarima212 = arima(Metro.ts1, order = c(2,1,2))
nsarima202 = arima(Metro.ts1, order = c(2,0,2))
nsarima502 = arima(Metro.ts1, order = c(5,0,2))
nsarima512 = arima(Metro.ts1, order = c(5,1,2))

acf1 = autoplot(acf(nsarima212$residuals, plot = F)) + theme_minimal()
acf2 = autoplot(acf(nsarima202$residuals, plot = F)) + theme_minimal()
acf3 = autoplot(acf(nsarima512$residuals, plot = F)) + theme_minimal()
acf4 = autoplot(acf(nsarima502$residuals, plot = F)) + theme_minimal()

grid.arrange(acf1, acf2, acf3, acf4, ncol = 2)
```

To verify our assumption, we fitted all four model with a seasonal component of (1,0,1) with a period of 7. The resulting ACF plots are shown in *Figure 13*. Here, the resulting ACF for ARIMA(2,0,2)(1,0,1)7 is not less meaningful than ARIMA model with higher AR orders.

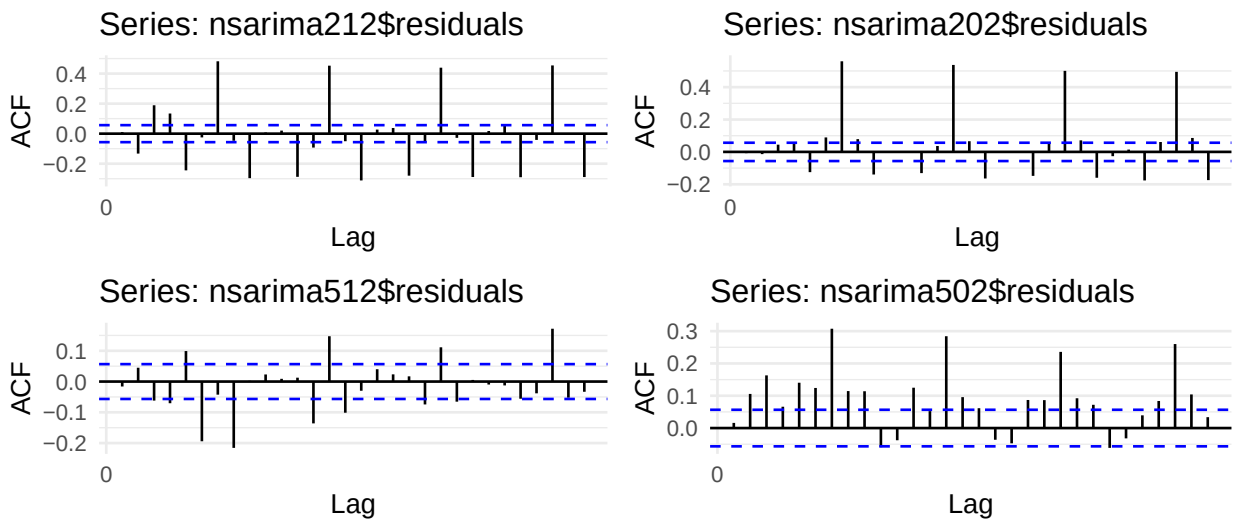

Figure 12: ACF of all four ARIMA models fitted with AR and MA Order shown from their Names

Therefore, we conclude that the model we fitted before in the beginning of the project, which is ARIMA(2,0,2)(1,0,1)7, is still the best model. A line plot of fitted values is shown together with the original data in *Figure 14*. From the figure, the model does a great job of modeling the 2015-2018 data, since the blue and gray line almost overlap with each other, and it was even necessary to bolded the grey line so it is not completely covered by the blue line.

```
narima212 = arima(Metro.ts1, order = c(2,1,2),
                  seasonal = list(order = c(1,0,1), period = 7))
narima202 = arima(Metro.ts1, order = c(2,0,2),
                  seasonal = list(order = c(1,0,1), period = 7))
narima502 = arima(Metro.ts1, order = c(5,0,2),
                  seasonal = list(order = c(1,0,1), period = 7))
narima512 = arima(Metro.ts1, order = c(5,1,2),
                  seasonal = list(order = c(1,0,1), period = 7))

acf202 = autoplot(acf(narima202$residuals, plot = F)) + theme_minimal()
acf212 = autoplot(acf(narima212$residuals, plot = F)) + theme_minimal()
acf502 = autoplot(acf(narima502$residuals, plot = F)) + theme_minimal()
acf512 = autoplot(acf(narima512$residuals, plot = F)) + theme_minimal()

grid.arrange(acf202, acf212, acf502, acf512, ncol = 2)
```

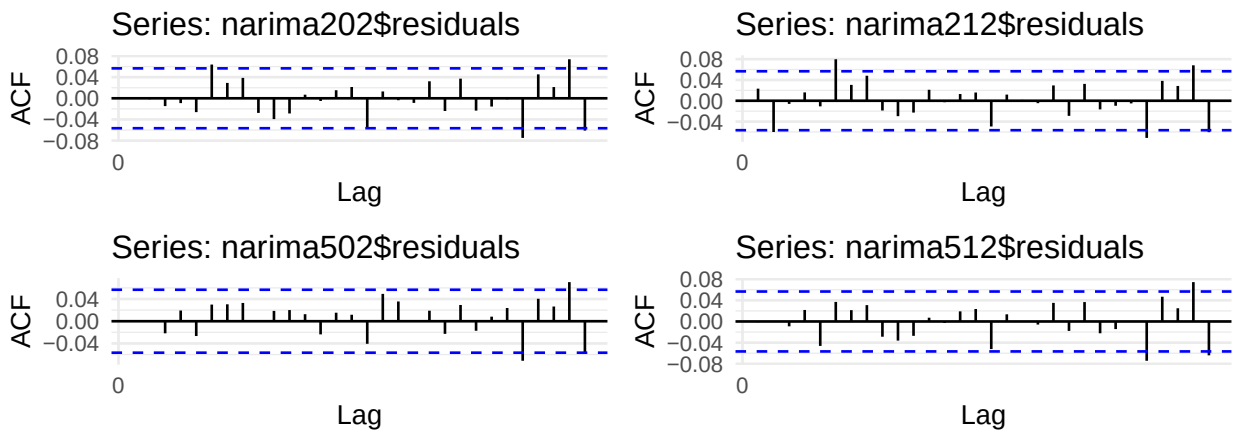

Figure 13: ACF of all four SARIMA models fitted all with Seasonal Component of (1,0,1)

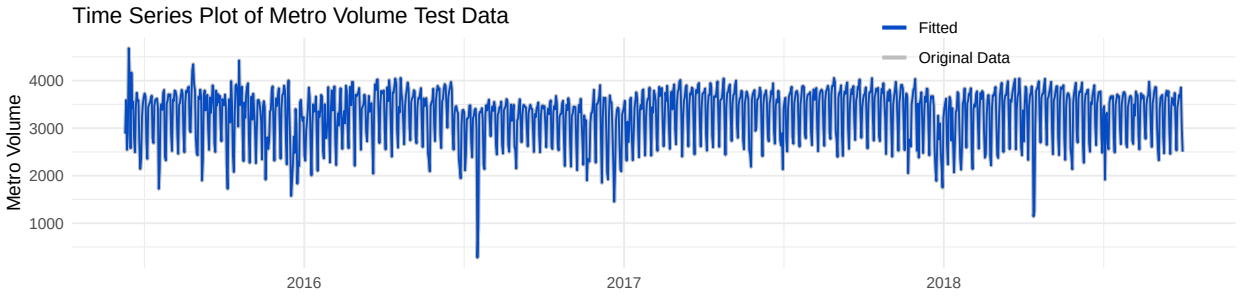

Figure 14: Time Series Plot with Fitted Values from Arima

```
themodel = forecast::Arima(Metro.ts1, order = c(2,1,2),
                           seasonal = list(order = c(1,0,1), period = 7))
ggplot() +
  geom_line(aes(x = time(Metro.ts1), y = Metro.ts1, color = "Original Data"),
            size = 1.1) +
  geom_line(aes(x = time(Metro.ts1), y = themodel$x, color = "Fitted")) +
  scale_y_continuous() +
  scale_x_continuous() +
  scale_color_manual(values = c("#044bc4", "grey")) +
  labs(y = "Metro Volume", x = element_blank(),
       title = "Time Series Plot of Metro Volume Test Data") +
  theme_minimal() +
  theme(legend.title=element_blank(), legend.position = c(0.75,1))
```

4.5 Regression Model

4.5.1 Bivariate Regression Models

We then fitted all the potential bivariate regression models. Comparing AIC, BIC, and sigma^2 (*Table 6*), the best bivariate model contains the variable `holiday` and `temperature`, as it has a sigma^2 of 271012, an AIC of 10168.02, and a BIC of 10214.96. Among all the models, the ones that contain `holiday` have lower AIC, BIC, and sigma^2 than other models without `holiday`.

```
MetroNew = MetroNew[!is.na(MetroNew$Volume),]
## Permutations
crMatrix <- MetroNew %>%
  dplyr::select(Cloud, Rain)
crreg <- arima(Metro.ts, order=c(2,0,2),
               seasonal=list(order=c(1,0,1), period=7), xreg=crMatrix)
ctMatrix <- MetroNew %>%
  dplyr::select(Cloud, Temp)
ctreg <- arima(Metro.ts, order=c(2,0,2),
               seasonal=list(order=c(1,0,1), period=7), xreg=ctMatrix)
chMatrix <- MetroNew %>%
  dplyr::select(Cloud, Holiday)
chreg <- arima(Metro.ts, order=c(2,0,2),
               seasonal=list(order=c(1,0,1), period=7), xreg=chMatrix)
rtMatrix <- MetroNew %>%
  dplyr::select(Rain, Temp)
rtreg <- arima(Metro.ts, order=c(2,0,2),
               seasonal=list(order=c(1,0,1), period=7), xreg=rtMatrix)
rhMatrix <- MetroNew %>%
  dplyr::select(Rain, Holiday)
rhreg <- arima(Metro.ts, order=c(2,0,2),
               seasonal=list(order=c(1,0,1), period=7), xreg=rhMatrix)
htMatrix <- MetroNew %>%
  dplyr::select(Holiday, Temp)
htreg <- arima(Metro.ts, order=c(2,0,2),
               seasonal=list(order=c(1,0,1), period=7), xreg=htMatrix)

bivariateselection = matrix(c(crreg$sigma2, crreg$aic, BIC(crreg),
                              ctreg$sigma2, ctreg$aic, BIC(ctreg),
                              chreg$sigma2, chreg$aic, BIC(chreg),
                              rtreg$sigma2, rtreg$aic, BIC(rtreg),
```

Table 6: Bivariate Model Selection

	Sigma^2	AIC	BIC
Cloud & Rain	296103.4	10225.96	10272.90
Cloud & Temp	294529.3	10222.44	10269.38
Cloud & Holiday	273750.6	10174.58	10221.52
Rain & Temp	292321.0	10217.46	10264.40
Rain & Holiday	271341.5	10168.76	10215.70
Holiday & Temp	270349.1	10166.42	10213.36

```
                              rhreg$sigma2, rhreg$aic, BIC(rhreg),
                              htreg$sigma2, htreg$aic, BIC(htreg)),
                            ncol = 3, byrow = T)
colnames(bivariateselection) = c("Sigma^2", "AIC", "BIC")
rownames(bivariateselection) = c("Cloud & Rain", "Cloud & Temp",
                                 "Cloud & Holiday", "Rain & Temp",
                                 "Rain & Holiday", "Holiday & Temp")
kable(bivariateselection, booktabs = T, caption = "Bivariate Model Selection")
```

4.5.2 Trivariate Regression Models

With a sigma^2 of 267482.3, an AIC of 10161.33, and a BIC of 10212.76, the model with `Holiday`, `Temp`, and `Rain` top the others by all sigma^2, AIC, and BIC values (*Table 7*). This model is also a better fit than the best bivariate model, which contains only `Temp` and `Holiday`.

```
# let's try rain holiday temp since they both have lower BIC's
rhtMatrix <- MetroNew %>%
  dplyr::select(Rain, Holiday, Temp)
rhtreg <- Arima(Metro.ts, order=c(2,0,2),
                seasonal=list(order=c(1,0,1), period=7),
                xreg=as.matrix(rhtMatrix), include.mean=F)
# let's compare that with cloud holiday temp
chtMatrix <- MetroNew %>%
  dplyr::select(Cloud, Holiday, Temp)
chtreg <- arima(Metro.ts, order=c(2,0,2),
                seasonal=list(order=c(1,0,1), period=7), xreg=chtMatrix)

# let's compare with cloud holiday rain
```

Table 7: Trivariate Model Selection

	Sigma^2	AIC	BIC
Rain & Temp & Holiday	271715.5	10162.67	10207.60
Cloud & Temp & Holiday	270132.0	10167.89	10219.33
Cloud & Holiday & Rain	271309.2	10170.67	10222.11
Cloud & Temp & Rain	292263.2	10219.31	10270.75

```
chrMatrix <- MetroNew %>%
  dplyr::select(Cloud, Holiday, Rain)
chrreg <- arima(Metro.ts, order=c(2,0,2),
                seasonal=list(order=c(1,0,1), period=7), xreg=chrMatrix)

# let's compare with cloud temp rain
ctrMatrix <- MetroNew %>%
  dplyr::select(Cloud, Temp, Rain)
ctrreg <- arima(Metro.ts, order=c(2,0,2),
                seasonal=list(order=c(1,0,1), period=7), xreg=ctrMatrix)


trivatriateregression = matrix(c(rhtreg$sigma2, rhtreg$aic, BIC(rhtreg),
                                 chtreg$sigma2, chtreg$aic, BIC(chtreg),
                                 chrreg$sigma2, chrreg$aic, BIC(chrreg),
                                 ctrreg$sigma2, ctrreg$aic, BIC(ctrreg)),
                               ncol = 3, byrow = T)
colnames(trivatriateregression) = c("Sigma^2", "AIC", "BIC")
rownames(trivatriateregression) = c("Rain & Temp & Holiday",
                                    "Cloud & Temp & Holiday",
                                    "Cloud & Holiday & Rain",
                                    "Cloud & Temp & Rain")

kable(trivatriateregression, booktabs = T,
      caption = "Trivariate Model Selection")
```

4.5.3 Final Model and Assumption Check

We then compared the best trivariate model found above with the model that considers all four variables. The two models are shown below. Even though `allreg` model, compared to `rhtreg`, has slightly better sigma^2 (267459 vs. 267482) it has a larger AIC and BIC value (10165.28 vs. 10161.38, and 10220.82, vs. 10214.44), meaning that is is penalized for including

one more variable. Therefore, we choose the model that contains `Rain`, `Holiday`, and `Temp` as our final regression model. Now the ACF of the final model residuals is shown in *Figure 15*. There are no significant lags, which shows that the model is adequate. The coefficients' p-values are shown in *Table 8*, and all the coefficients have significant p-values. Our final model has the form of

$$\hat{Z}_t = 1.03Z_{t-1} - 0.87Z_{t-2} - 0.88w_{t-1} + 0.77w_{t-2} + 0.95Z_{t-7} - 0.76w_{t-7} + 870.57 - 131.23x_1 - 925.94x_2 + 8.72x_3$$

where x_1 is Rain, x_2 is Holiday, x_3 is Temp.

```
# let's compare with all four
predictorMatrix <- MetroNew %>%
  dplyr::select(Temp, Rain, Cloud, Holiday)

allreg <- stats::arima(Metro.ts, order=c(2,0,2),
                       seasonal=list(order=c(1,0,1),
                  period=7), xreg=predictorMatrix)
summary(allreg)
##
## Call:
## stats::arima(x = Metro.ts, order = c(2, 0, 2), seasonal = list(order = c(1,
##     0, 1), period = 7), xreg = predictorMatrix)
##
## Coefficients:
##          ar1      ar2      ma1     ma2    sar1     sma1  intercept    Temp
##       1.0253  -0.8687  -0.8840  0.7738  0.9510  -0.7584   884.4362  8.7008
## s.e.  0.0645   0.0614   0.0769  0.0735  0.0193   0.0472   763.5325  2.6934
##            Rain    Cloud    Holiday
##       -129.3169  -0.1759  -925.1607
## s.e.    50.1332   0.7597   118.4486
##
## sigma^2 estimated as 267459:  log likelihood = -5070.64,  aic = 10165.28
##
## Training set error measures:
##                      ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -6.999023 517.1642 336.451 -6.55231 15.35682 0.5780765
##                     ACF1
## Training set 0.04268034
summary(rhtreg)
```

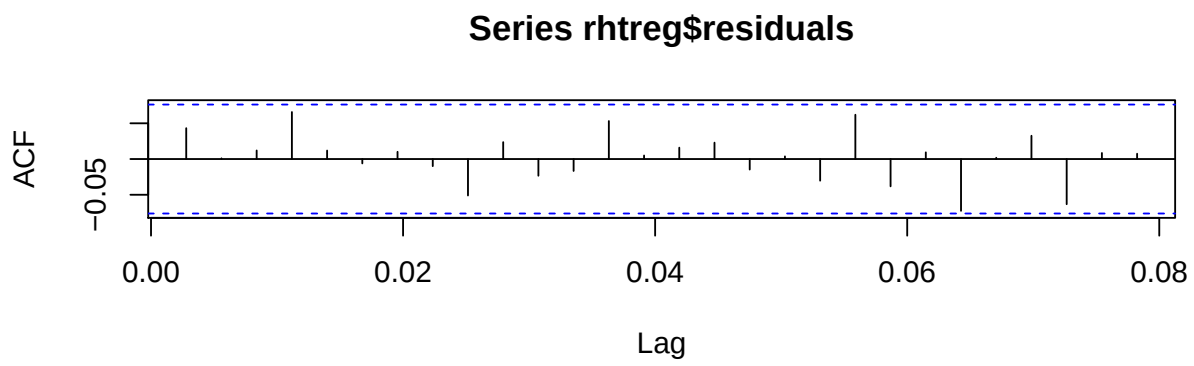

Figure 15: Final Model ACF

```
## Series: Metro.ts
## Regression with ARIMA(2,0,2)(1,0,1)[7] errors
##
## Coefficients:
##          ar1      ar2      ma1     ma2    sar1     sma1       Rain
##       1.0246  -0.8652  -0.8813  0.7683  0.9523  -0.7632  -133.9779
## s.e.  0.0655   0.0628   0.0784  0.0750  0.0189   0.0462    49.2059
##         Holiday     Temp
##       -924.1015  11.7892
## s.e.   118.6043   0.3220
##
## sigma^2 estimated as 271716:  log likelihood=-5071.33
## AIC=10162.67   AICc=10163.01   BIC=10207.6
##
## Training set error measures:
##                      ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -5.033189 517.7025 337.7805 -6.456484 15.37226 0.4728682
##                     ACF1
## Training set 0.04322155

BIC(allreg) #10220.82
## [1] 10219.21
BIC(rhtreg) #10214.44
## [1] 10207.6
```

```
acf(rhtreg$residuals)
```

Table 8: P-Values for the Final Model's Coefficients

	P
ar1	0.0000000
ar2	0.0000000
ma1	0.0000000
ma2	0.0000000
sar1	0.0000000
sma1	0.0000000
Rain	0.0064731
Holiday	0.0000000
Temp	0.0000000

```
pval = as.matrix((1-pnorm(abs(rhtreg$coef)/sqrt(diag(rhtreg$var.coef))))*2)
colnames(pval) = c("P")
kable(pval, booktabs = T,
      caption = "P-Values for the Final Model's Coefficients")
```

4.5.4 Model Reliability and Forecasting with Predictors

We first checked the fitted values together with the original data from 2012 to 2014 (*Figure 16*). It is not much different from the fitted values of the SARIMA model without regressors (*Figure 17*). They both are capture the original data well.

Then, we proceeded to check the model fit using our test data from 2015 to 2018. The result is shown in *Figure 18*. Since there is an outlier in the forecasted values (2015 to 2018), we changed the y limits so that the reader can better see the trend and fitted values *Figure 19*. However, we do want to acknowledge that the model without regressors as shown in *Figure 14* better handles the data from 2015 to 2018 compared to the model with regressors. A potential reason is that for `Rain`, the mean and maximum in the test dataset (0.4 and 410, respectively) are much larger than the mean and maximum in the train dataset (0.11 and 4.68, respectively). Therefore, the coefficient in the train dataset may not work well in the test dataset.

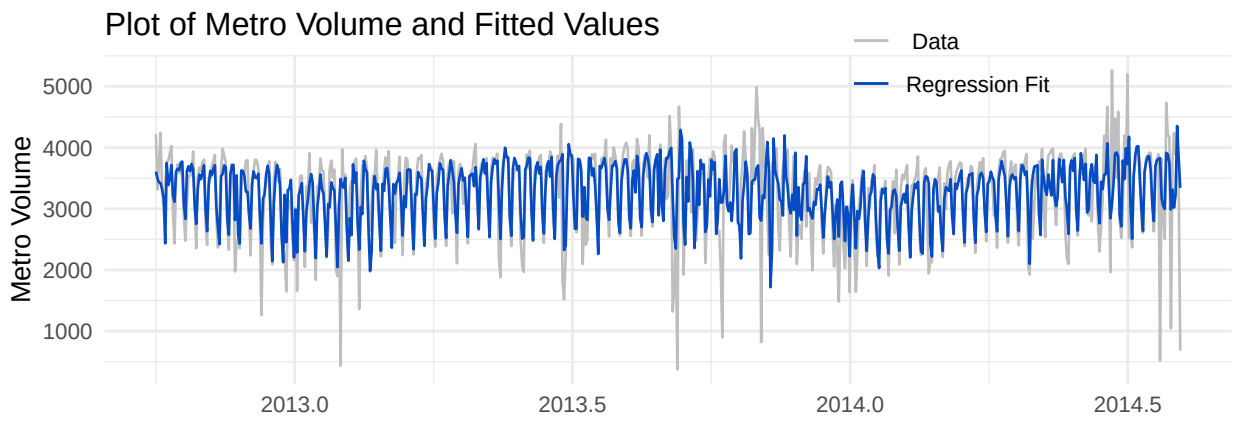

Figure 16: Fitted Values from Regression

```
cutmatrix = MetroNew %>%
  dplyr::select(Rain, Holiday, Temp)

ggplot() +
  geom_line(aes(x = time(Metro.ts), y = Metro.ts, color = " Data")) +
  geom_line(aes(x = time(Metro.ts), y = fitted(rhtreg),
                color = "Regression Fit")) +
  scale_y_continuous() +
  scale_x_continuous() +
  scale_color_manual(values = c("grey", "#044bc4")) +
  labs(y = "Metro Volume", x = element_blank(),
       title = "Plot of Metro Volume and Fitted Values") +
  theme_minimal() +
  theme(legend.title=element_blank(), legend.position = c(0.75,1))
```

```
ggplot() +
  geom_line(aes(x = time(Metro.ts), y = Metro.ts, color = " Data")) +
  geom_line(aes(x = time(Metro.ts), y = fitted(sarima101),
                color = "SARIMA Fit")) +
  scale_y_continuous() +
  scale_x_continuous() +
  scale_color_manual(values = c("grey", "#044bc4")) +
  labs(y = "Metro Volume", x = element_blank(),
       title = "Plot of Metro Volume and Fitted Values") +
  theme_minimal() +
  theme(legend.title=element_blank(), legend.position = c(0.75,1))
```

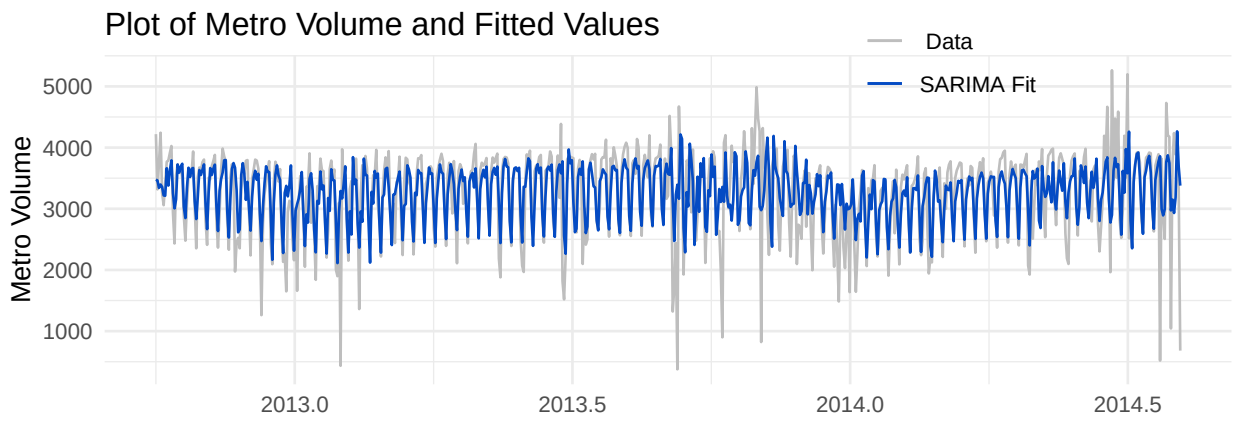

Figure 17: Plot of Metro Volume and SARIMA Fitted Values

```
rhtMatrixTest = MetroNewTest[!is.na(MetroNewTest$Volume),] %>%
  dplyr::select(Rain, Holiday, Temp)

rhtpredict.1 = forecast::forecast(rhtreg,
                                  xreg = as.matrix(rhtMatrixTest,
                                                   ncol = 3))

predictedrht.1 = ts(rhtpredict.1$mean, start=c(2015.44), frequency=365)

ggplot() +
  geom_line(aes(x = time(predictedrht.1), y = predictedrht.1,
                color = "Forecasted Values")) +
  geom_line(aes(x = time(Metro.ts1), y = Metro.ts1,
                color = "Training Data")) +
  scale_y_continuous() +
  scale_x_continuous() +
  scale_color_manual(values = c("grey", "#044bc4")) +
  labs(y = "Metro Volume", x = element_blank(),
       title = "Plot of Metro Volume and Forecasted Values") +
  theme_minimal() +
  theme(legend.title=element_blank(), legend.position = c(0.75,0.25))
```

```
ggplot() +
  geom_line(aes(x = time(Metro.ts1), y = Metro.ts1,
                color = "Original Data")) +
  geom_line(aes(x = time(predictedrht.1), y = predictedrht.1,
                color = "Forecasted Values")) +
  ylim(c(0,5000)) +
  scale_x_continuous() +
```

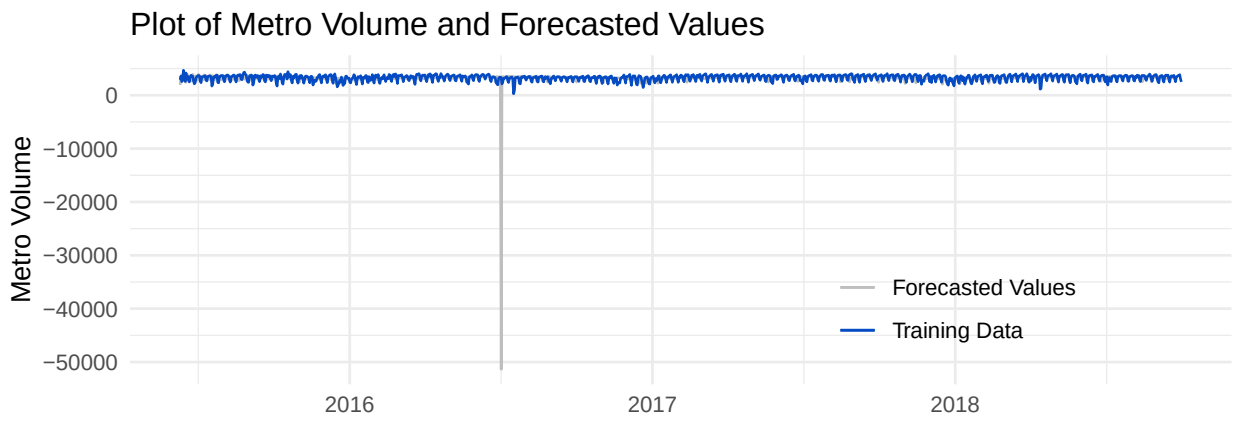

Figure 18: Fitted Values from Regression on Data from 2015 to 2018

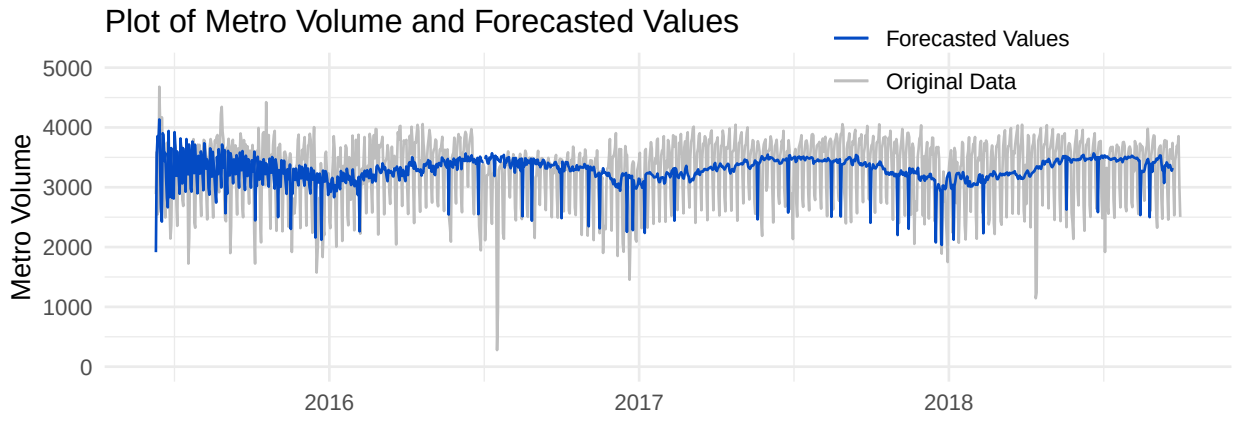

Figure 19: Fitted Values from Regression on Data from 2015 to 2018 (Modified Y-Axis)

```
scale_color_manual(values = c("#044bc4", "grey")) +
labs(y = "Metro Volume", x = element_blank(),
     title = "Plot of Metro Volume and Forecasted Values") +
theme_minimal() +
theme(legend.title=element_blank(), legend.position = c(0.75,1))
```

5 Discussion

Our project's analyses allowed us to achieve our goal of identifying which predictors influence daily metro traffic volume from Minneapolis and St. Paul, Minnesota. From our results we can see that rain, holiday, and temperature are our most significant predictors

for metro traffic volume. We are ultimately not surprised by these results. Rain can cause drivers to drive slower, and rain can also cause crashes, which can worsen traffic. During holidays, there is usually an increased amount of traffic, especially on populated highways such as the I-94, because people are usually traveling to visit relatives or to leave the state. Additionally, the airport is off of I-94, and there is usually an increased amount of people going to the airport during the holidays. Lastly, temperature can increase traffic because if it is too cold the road may be slippery, which can slow down peoples' driving and can also cause accidents. If the temperature is too hot and sunny, then people may be distracted by the weather and cause crashes or slowdowns. However, it is interesting that in our time frame that rain and holiday mainly take the value of "No Rain" and "No Holiday", so we wonder if I-94 is a populated highway where drivers experience daily traffic no matter the weather conditions and other external conditions.

We also know that our model holds up for the entire dataset because our test data, after 2015, follows the same ARIMA(2,0,2) and SARIMA(1,0,1) model as our model before 2015. Our model is adequate because the ACF plot of the residuals shows no significant lags. However, when we forecasted the data on our test data timeline, there was one observation #337 that led a huge drop in our forecasted value. We think that this may be due to the fact that there was incorrect data recorded on that day, so our regression model could not handle it. Also to note is that the SARIMA model without regressors handle the 2015-2018 data much better than the model with regressors. Given the time constraint for this project, we can explore more and potentially address this issue later.

In the future, we want to find data on construction projects and road closures. We are interested in examining how our current analyses would be impacted by these two variable; construction will almost always cause slowdowns and increase traffic, and road closures also slow down traffic as vehicles have to take detours and roads that they are unfamiliar with to get to their destinations.

Overall, our project was good in answering our initial question. We can confirm that a

SARIMA (2,0,2)(1,0,1)7 provides us a good fit throughout the entire dataset (2012 to 2018). This SARIMA model with regressors perform well in the train dataset (2012-2014), and all coefficients have significant p-values. Although this model with regressors capture less information compared to the model without regressors in the test dataset (2015-2018), it still captures the general trend.

6 References

Frank, Jacqui. "Here's How Much Time and Money You Waste Sitting in Traffic a Year." Business Insider, Business Insider, 23 Feb. 2017, https://www.businessinsider.com/time-money-spent-traffic-per-year-us-cities-new-york-los-angeles-san-francisco-atlanta-2017-2.

Hyndman R, Athanasopoulos G, Bergmeir C, Caceres G, Chhay L, O'Hara-Wild M, Petropoulos F, Razbash S, Wang E, Yasmeen F (2019). *forecast: Forecasting functions for time series and linear models*. R package version 8.10, <URL: http://pkg.robjhyndman.com/forecast>.

Hyndman RJ, Khandakar Y (2008). "Automatic time series forecasting: the forecast package for R." *Journal of Statistical Software*, *26*(3), 1-22. <URL: http://www.jstatsoft.org/article/view/v027i03>.

"Metro Interstate Traffic Volume Data Set." UCI Machine Learning Repository: Metro Interstate Traffic Volume Data Set, 7 May 2019, https://archive.ics.uci.edu/ml/datasets/Metro Interstate Traffic Volume?fbclid=IwAR2FTsUt6M43RC0SkstGD-lboHvNp16bUxekejsn2XOSsl_KdhsInOY4Xng#.

Nag, Oishimaya Sen. "The 10 Biggest Cities In Minnesota." WorldAtlas, 14 Jan. 2019, https://www.worldatlas.com/articles/the-10-biggest-cities-in-minnesota.html.