DNSC 4219 PROJECT

Daily New York City Collision Data

Jonathan Beck, Henrique Cassol, Jesse Mutamba, Matthew Wolf

1. Introduction and Overview

This project focuses on modeling daily motor vehicle collisions in New York City, using time series data collected over a two-year span from January 1, 2016, to December 31, 2017. The primary target variable of interest is COLLISIONS, which records the total number of reported collisions each day. Understanding and forecasting trends in daily collisions is valuable for public safety planning, traffic management, and the allocation of emergency response resources.

In addition to collision counts, the dataset includes the following meteorological variables that could influence traffic patterns and accident rates: AWND: Average daily wind speed (in meters per second), TMAX: Daily maximum temperature (in degrees Celsius), TMIN: Daily minimum temperature (in degrees Celsius), PRCP: Daily precipitation (in millimeters).

These variables will be considered as potential external regressors in time series models to investigate how weather conditions correlate with or help explain variability in collision rates.

Modeling this type of data can provide predictive insights that support proactive decision-making, such as identifying high-risk weather conditions or anticipating surges in collisions during certain times of the year.

Time Series Plot

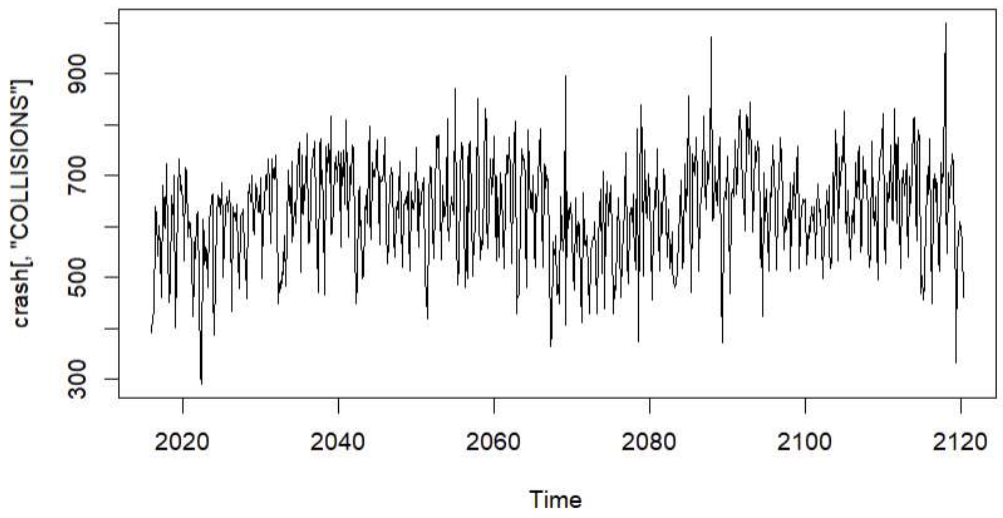


Figure 1

No Clear Long-Term Trend:
The series (*Figure 1)* does not show a strong increasing or decreasing trend over time. Instead, the number of daily collisions appears to fluctuate around a relatively stable mean. This suggests that, at least over the observed period, the overall rate of collisions remains fairly consistent.

Seasonal Behavior:
The time series (*Figure 1*) exhibits visible seasonality or cyclical behavior. Regular oscillations in the number of collisions suggest repeating patterns, possibly related to weekly traffic cycles, weather conditions, or seasonal travel behaviors. For example, increased variability might correspond to winter months when weather impacts driving conditions, or summer months when travel increases.

Short-Term Fluctuations:
The series (*Figure 1*) exhibits considerable short-term volatility, with frequent peaks and dips. These may be due to localized events, holidays, policy changes, or severe weather days.

Outliers:
A few extreme spikes and dips in *Figure 1.1* are noticeable. These could represent unusually high or low collision days, perhaps tied to holidays, snowstorms, COVID lockdown, and major traffic disruptions.

Seasonal Box Plot for Day of the Week

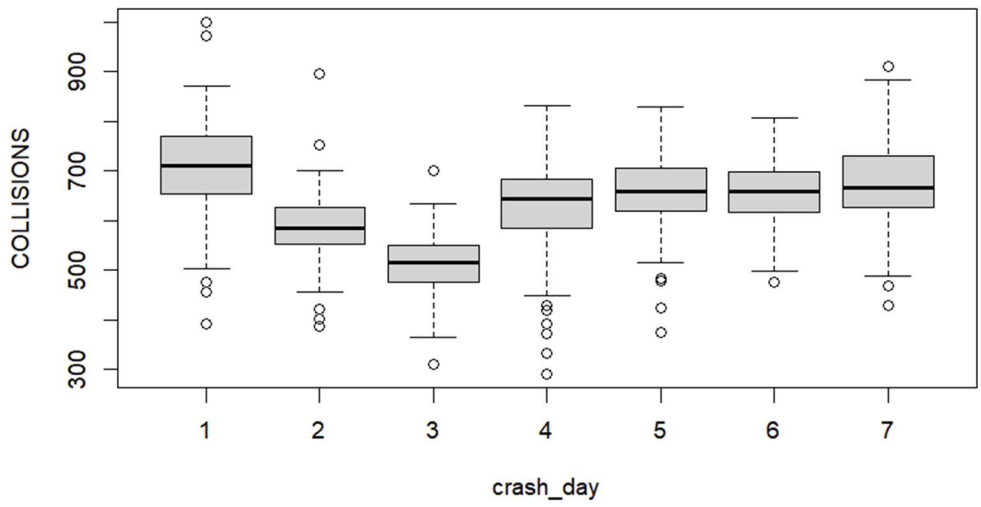


Figure 2

The box plot (*Figure 2*) shows the distribution of daily collisions in NYC by day of the week, with Friday as day 1 and Thursday as day 7. Collisions are highest on Fridays, followed by Thursdays and Mondays, while weekends (especially Sundays) see the lowest collision counts. This pattern highlights clear weekly seasonality, with more collisions occurring on weekdays due to higher commuter traffic and fewer on weekends when traffic volume is reduced.

Seasonal Box Plot for Month

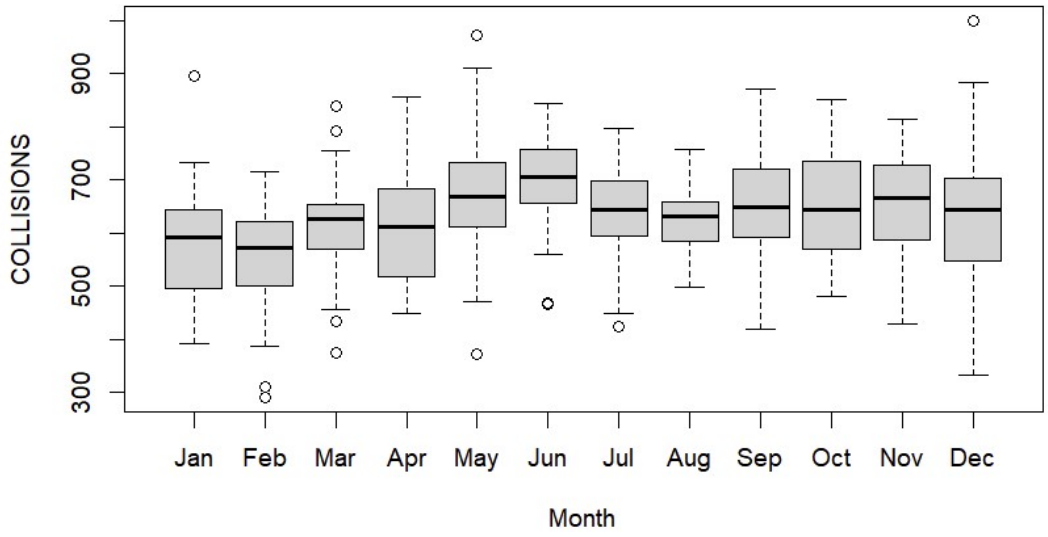


Figure 3

The box plot (*Figure 3*) displays the distribution of daily collisions by month, revealing gradual seasonal variation throughout the year. Collisions tend to be lower in the winter months, especially January and February, and gradually increase into the spring, peaking in May and July. From August through November, collisions remain relatively high before dipping again in December. This pattern suggests that warmer months may see more traffic and activity, contributing to higher collision rates, while harsher winter weather or reduced travel may suppress collisions in colder months.

Periodogram

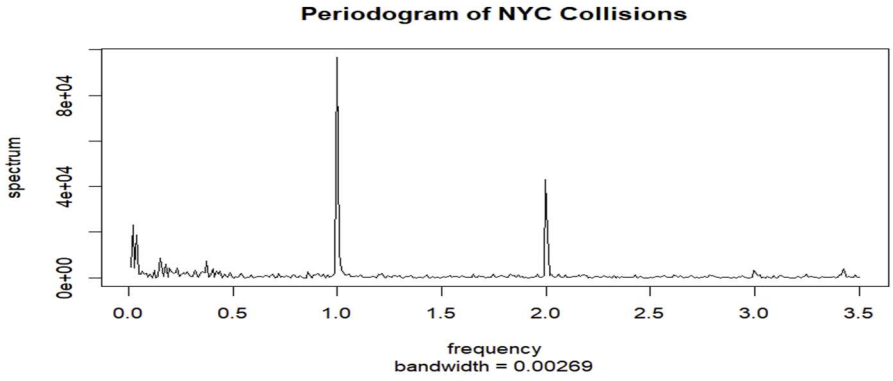


Figure 4

The periodogram (*Figure 4*) shows a dominant peak at frequency = 1, indicating a strong weekly cycle in the collision data, which is consistent with the observed day-of-week

seasonality. A secondary peak at frequency = 2 suggests harmonic repetition of this pattern. These findings confirm that collisions follow a clear 7-day cycle, reinforcing the need to account for weekly seasonality in any time series model.

ACF

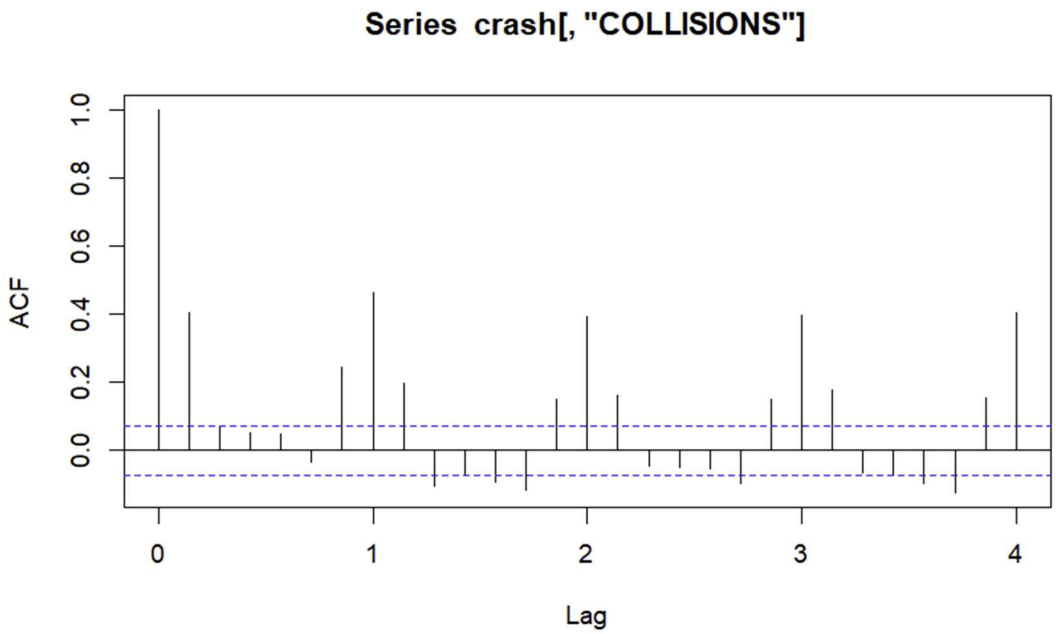


Figure 5

The autocorrelation function (ACF) plot (*Figure 5*) of the COLLISIONS series reveals an extremely slow decay, suggesting the series is **non-stationary** rather than stationary. Consistent spikes, lying outside of the two standard error bounds, appear at lags around 7, 14, 21, and so on, which indicates **weekly seasonality**. This pattern suggests that values tend to repeat in a similar pattern every 7 days.

2. Univariate Time-series models

Two types of Univariate models were created for the Time Series of daily collisions in New York City. The first model was created by using the manually created dummy variables for month and day of week. The second model was created by producing cos and sin pairs for the top 6 periods that had the highest amplitude from the periodogram output. They were identified as periods 6.976, 3.508, 3.488, 300, 7.058, and 200.

2.1 Deterministic Time Series Models (Seasonal Dummies and Trend, Cyclical Trend)

Initially, dummy variables were created for month and day of week by using the as.factor() function, but by using that method, appropriate models were not able to be made in section 4. Therefore, dummy variables were created by manual assignment; m1-m12 for monthly indicators, and d1-d7 for day of the week. If the month column (created in introduction) was

equal to "JAN," then a value of 1 would be assigned to m1 while m2-m12 would be assigned values of 0. The same principle applies to day of week, where instead if the wkdy column was equal to 1, then d1 would be assigned a value of 1 while d2-d7 would be assigned values of 0.

The model comprising trend and dummies for month and day of week was then trained on the first 600 observations of collision data which comprised the train_crash data. M1

```
Call:
lm(formula = train_crash ~ time + m2_train + m3_train + m4_train +
    m5_train + m6_train + m7_train + m8_train + m9_train + m10_train +
    m11_train + m12_train + d2_train + d3_train + d4_train +
    d5_train + d6_train + d7_train)

Residuals:
    Min      1Q  Median      3Q     Max
-290.69  -34.46    4.11   42.63  358.67

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.515e+02  1.209e+01  53.886  < 2e-16 ***
time         6.561e-03  1.796e-02   0.365 0.715025
m2_train    -1.970e+01  1.315e+01  -1.498 0.134548
m3_train     3.268e+01  1.291e+01   2.531 0.011636 *
m4_train     3.020e+01  1.307e+01   2.311 0.021187 *
m5_train     8.667e+01  1.304e+01   6.645 6.97e-11 ***
m6_train     1.192e+02  1.325e+01   8.994  < 2e-16 ***
m7_train     6.289e+01  1.327e+01   4.741 2.68e-06 ***
m8_train     4.966e+01  1.378e+01   3.603 0.000341 ***
m9_train     6.809e+01  1.597e+01   4.265 2.33e-05 ***
m10_train    6.148e+01  1.583e+01   3.883 0.000115 ***
m11_train    6.605e+01  1.608e+01   4.108 4.56e-05 ***
m12_train    3.937e+01  1.599e+01   2.462 0.014087 *
d2_train    -1.156e+02  1.092e+01 -10.585  < 2e-16 ***
d3_train    -1.915e+02  1.092e+01 -17.528  < 2e-16 ***
d4_train    -7.985e+01  1.093e+01  -7.308 8.99e-13 ***
d5_train    -5.355e+01  1.093e+01  -4.901 1.24e-06 ***
d6_train    -5.148e+01  1.096e+01  -4.699 3.27e-06 ***
d7_train    -3.235e+01  1.095e+01  -2.954 0.003267 **
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 71.59 on 581 degrees of freedom
Multiple R-squared:  0.4983,    Adjusted R-squared:  0.4827
F-statistic: 32.05 on 18 and 581 DF,  p-value: < 2.2e-16
```

Figure 6

(January), and d1(Friday) were chosen as the reference month and day of the model. The model output can be seen in *Figure 6* below. Based on the model's output, it can be concluded that the trend component within the seasonal dummy variable model is not statistically significant, as its p-value far exceeds the 0.05 level of significance. Furthermore, it can be concluded that there exists seasonality for the monthly component of the series, because all monthly dummy variables except for m2 (February) have a p-value less than 0.05 for the t-stats. Therefore, they have significant deviation from January beyond trend. Seasonality can also be concluded from day of the week, as all days exhibit a p-value less than the 0.05 level of significance for t-stats. Therefore, they have significant deviation from Friday beyond trend.

Below in *Figure 7*, a predicted versus actual time series plot was produced based on the seasonal dummy model where the blue line represents the actual values, and the red line represents the predicted values. Based on this plot, the model does seem to capture some of the general patterns of the actual data. However, the model does not appear able to generalize effectively to areas in

the TS plot where there are large spikes in variation. This implies that the model may not perform as well on unseen data in the holdout sample.

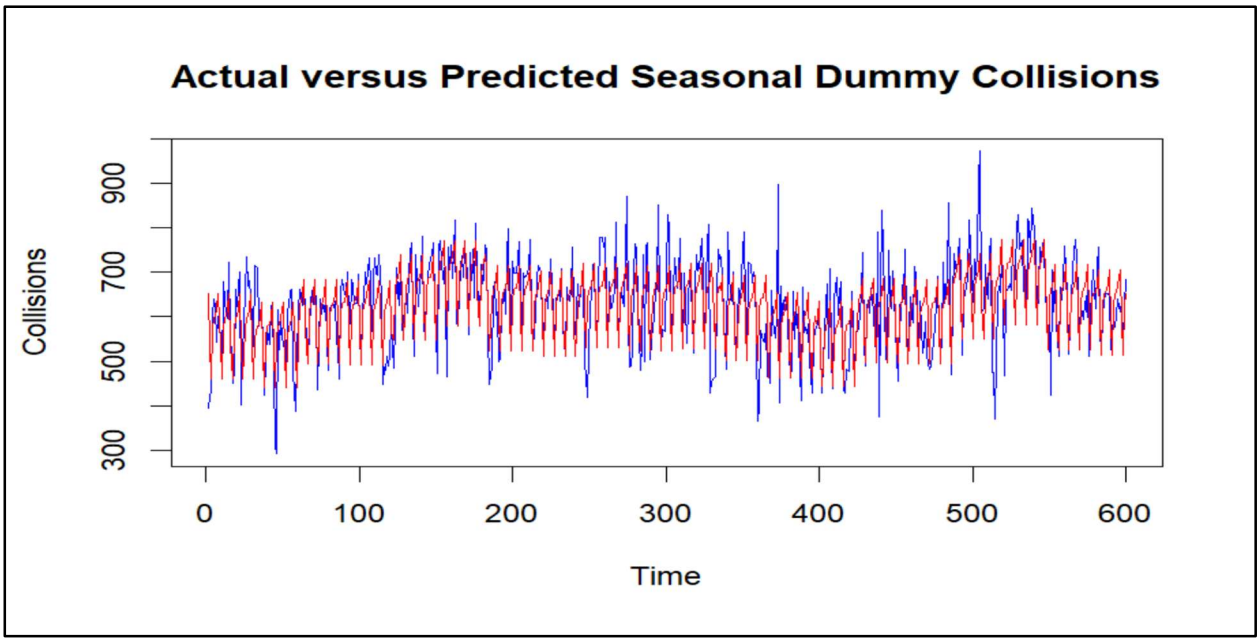


Figure 7

The next model created was the cyclical trend model. In order to determine which periods in the model were significant enough to be used in the model, the periodogram **(*Figure 4 in the Introduction and Overview section*)** was created to identify dominant periods. To best capture the nature of the collision time series, the top 6 periods by amplitude were selected. They were as follows: 6.976, 3.508, 3.488, 300, 7.058, and 200. Each period was then defined with a cos and sin term where cos1 and sin1 represent period 6.976, cos2 and sin2 represent period 3.508, and so on. The model was then trained on the training sample of collision data using trend and the 6 pairs, with the output of the cyclical trend model shown below in *Figure 9*.
Similar to that of the seasonal dummy model, as evidenced by *Figure 9*, the trend component in the cyclical trend model was not statistically significant due to its high p-value. Moreover, all cos and sin pairs used in the model were statistically significant except for cos4. This means that all periods except for period 300 have a significant contribution to explaining the variability in the collision series, as those pairs had a p-value associated with it less than 0.05. Because at least one of the sin and cos pairs for period 300 had a p-value greater than 0.05, it cannot be concluded that period has a significant contribution to explaining the variability in the daily collision data.

A time series plot was again generated to compare the predicted vs actual values of the model using the model. Shown below in *Figure 8*, the blue line represents actual values, while the red line represents predicted values from the cyclical trend model. Like that of the seasonal dummy plot which compared predicted values, the cyclical trend does seem to capture some of the general patterns. Where the cyclical trend appears to be doing even considerably worse than

```
Call:
lm(formula = train_crash ~ time + cos1 + sin1 + cos2 + sin2 +
    cos3 + sin3 + cos4 + sin4 + cos5 + sin5 + cos6 + sin6)

Residuals:
    Min      1Q  Median      3Q     Max
-313.79  -40.99    8.16   47.15  353.16

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 636.22911    6.88144  92.456  < 2e-16 ***
time         -0.03186    0.02041  -1.561 0.119008
cos1         43.92816    4.41546   9.949  < 2e-16 ***
sin1         30.12364    4.41561   6.822 2.24e-11 ***
cos2         29.46599    4.41546   6.673 5.80e-11 ***
sin2         14.30704    4.41544   3.240 0.001262 **
cos3        -28.53391    4.41546  -6.462 2.17e-10 ***
sin3        -15.28111    4.41544  -3.461 0.000578 ***
cos4         -2.73078    4.41546  -0.618 0.536513
sin4        -40.35086    4.82630  -8.361 4.56e-16 ***
cos5        -22.67169    4.41546  -5.135 3.85e-07 ***
sin5        -21.55449    4.41562  -4.881 1.36e-06 ***
cos6        -20.69855    4.41546  -4.688 3.44e-06 ***
sin6        -25.40174    4.60254  -5.519 5.12e-08 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 76.48 on 586 degrees of freedom
Multiple R-squared:  0.4225,    Adjusted R-squared:  0.4097
F-statistic: 32.98 on 13 and 586 DF,  p-value: < 2.2e-16
```

Figure 9

the seasonal dummy, is at points of large variation, the model does not generalize to it much at

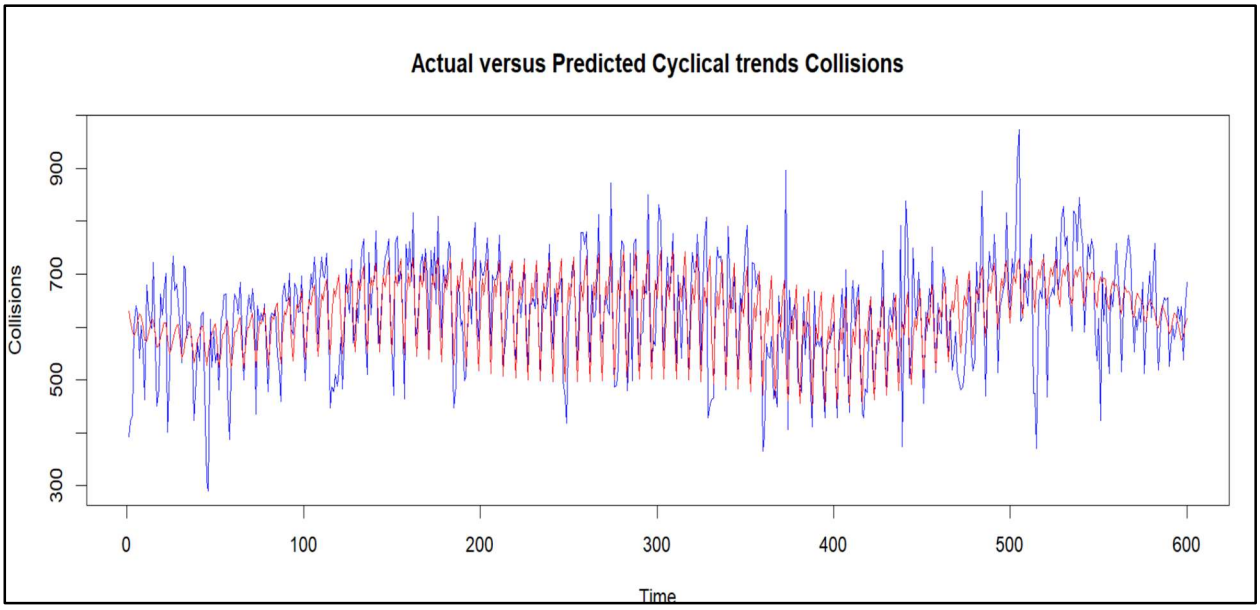


Figure 8

all.

2.2 Comparison of "candidate" models in terms of fit

The univariate time series models were evaluated using R-square, MAPE, RMSE, and MAE. In order to facilitate efficient comparison of model metrics, two tables were created to

Metric <chr>	Training_set <dbl>	Holdout_set <dbl>
MAPE	0.08753549	0.09033148
RMSE	70.44960663	77.80325642
MAE	51.44189077	56.85556672
R2	0.49825944	0.38915910

Figure 10

summarize the performance of each model. A holdout sample size of 131 days was used for evaluation of the models. The seasonal dummy variable model evaluation table, *Figure 10*, is shown above. The model has a moderate fit, with an R2 of 0.498, MAPE of 8.75%, and RMSE of 70.44 on training data. When used on the holdout sample, performance across all metrics drops marginally, with MAPE increasing to 9.03%, RMSE increasing to 77.8, and R2 dropping to 0.389.

The second table, containing the evaluation metrics of the cyclical trend model is shown below in *Figure 11*. This model had a worse fit across the board, it had a MAPE of 9.67%, R2 of 0.422, and RMSE of 75.57. When the cyclical trend model was used to predict values of the

Metric <chr>	Training_set <dbl>	Holdout_set <dbl>
MAPE	0.09673193	0.1571088
RMSE	75.57964381	133.6968750
MAE	56.54141130	104.1497154
R2	0.42252691	-0.8037447

Figure 11

holdout set, it resulted in a performance drop more steep than the previous seasonal dummy model. MAPE increased to 15.7%, RMSE nearly doubled to 133.7, and the holdout R2 was a much different -0.803. This indicates that on holdout data, the cyclical trend model performed worse than just using the mean of the series; therefore, the cyclical trend model does not provide an accurate representation of the data as is.

Overall, the seasonal dummy model performed better across the board in comparison to the cyclical trend model. The seasonal dummy model also generalized to unseen data in a more promising fashion. Therefore, out of the two univariate time series models, the seasonal dummy variable model gives the most accurate representation of the collision time series data.

2.3 Looking at residuals of the model(s) to see if they are stationary or not.

In order to test if the residuals of both models were stationary or not, ACFs were created as well as having a Box Pierce test performed. Shown below in *Figure 13* is the ACF of the regression residuals for the seasonal dummy model. Multiple correlations surpass the two

standard error bounds computed by R at different lags, while exhibiting extremely slow decay to 0 (**nonstationary behavior**), so the null hypothesis that the regression residuals of the seasonal dummy variable model are white noise can be rejected. Additionally, when using a box pierce test **(***Figure 12***)**, a p-value much less than the level of significance 0.05 is calculated. This confirms the decision to reject the null hypothesis, suggesting that the residuals of the seasonal model exhibit nonstationary behavior. Thus, until this correlation between residuals is handled,

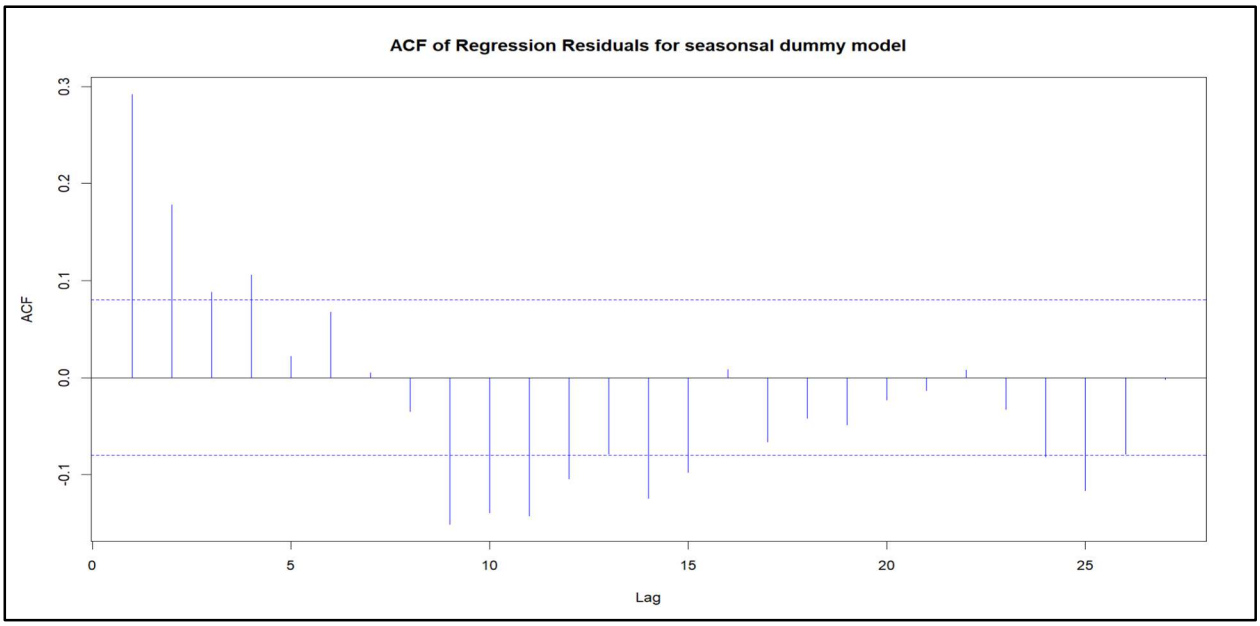


Figure 13

```
        Box-Pierce test

data:  res_dummy
X-squared = 153.53, df = 20, p-value < 2.2e-16
```

Figure 12

the seasonal dummy variable model could not be reliably used for forecasting

When looking at the ACF of the residuals for the Cyclical trend model **(***Figure 14***)**, a similar behavior is demonstrated as that seen from the seasonal dummy variable - multiple lags exhibit autocorrelations exceeding the two standard error bounds. Therefore, the null hypothesis that the residuals are white noise can be rejected. In a similar vein, the Box-Pierce test **(***Figure 15***)** confirms that the residuals are not white noise, as the p-value associated with it was much less than the 0.05 level of significance. Thus, we can conclude that the cyclical trend model

exhibits nonstationary behavior. Moreover, this model could not be used for forecasting due to the correlated residuals.

```
        Box-Pierce test

data:  res_CM
X-squared = 162.52, df = 20, p-value < 2.2e-16
```

Figure 15

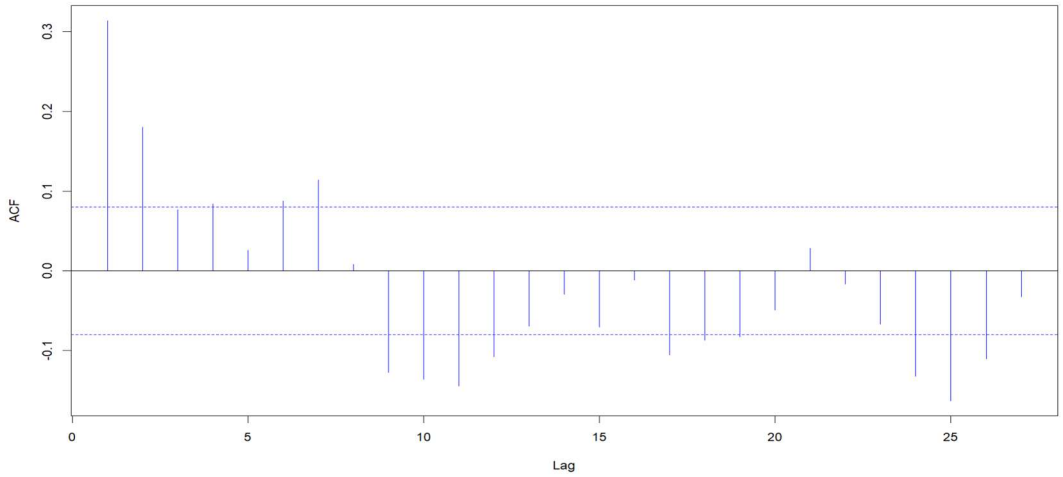


Figure 14

3. Time Series Regression Models

This Section looks at the effectiveness of time series regression models in predicting daily traffic collisions in New York City on selected weather variables. The goal is to assess the predictive power of variables like, wind speed, temperature, and precipitation using correlation analysis, regression modeling, and residual diagnostics. The analysis also includes evaluating candidate models based on goodness-of-fit metrics, testing model performance on holdout samples, and checking for residual stationarity.

3.1 Correlation Analysis and Scatter Plots:

When examining the relationship between weather variables and daily traffic collisions in NYC, the independent variables: Average Wind Speed (**AWND**), Maximum Temperature (**TMAX**), Minimum Temperature (**TMIN**), and Precipitation (**PRCP**). A correlation matrix and scatter plot analysis were conducted using these variables against dependent variables (**COLLISIONS).**

The correlation matrix (*Figure 16.a*) highlighted that **TMAX** and **TMIN** are the strongest predictors among the selected features, with correlation coefficients of 0.257 and 0.229. This suggests that warmer or colder temperatures may be associated with changes in collision frequency. **AWND** had a weak negative correlation(-0.13) with collisions, while **PRCP** showed minimal linear association (0.0025), suggesting that there is a limited standalone predictive power

Table for Matrix

Variable pair	**Correlation**
Collisions & TMAX	**0.257**
Collisions & TMIN	**0.229**
Collisions & AWND	**-0.130**
Collisions & PRCP	**0.025**

.

Figure 16.a

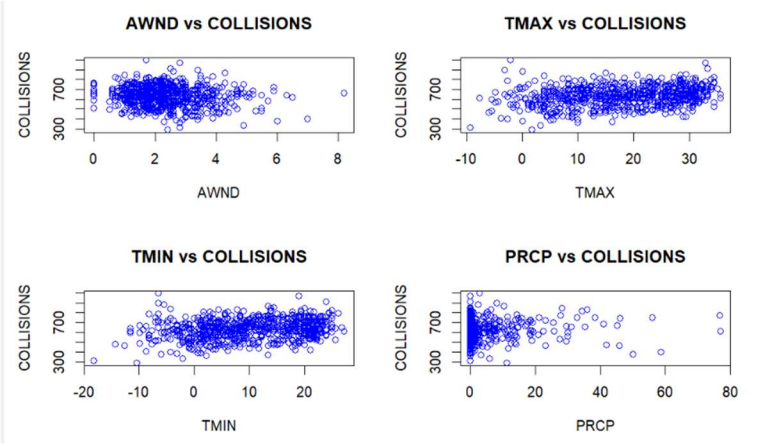


Figure 1617.b

Figure 16.b showed these patterns:

- **TMAX** and **TMIN** showed linear relationships with collisions.
- **AWND** showed a weak negative trend with some noise.
- **PRCP** demonstrated a highly scattered pattern, there are no clear trends - indicating the need for transformation in future interactions.

The correlation results and scatter plots suggest that temperature variables may be more predictive of daily collisions than wind or precipitation. Despite some multicollinearity present between TMAX and TMIN (correlation = 0.956), both were kept for modeling to explore their individual effects.

3.2 Comparison of Candidate models (Fit & Holdout Evaluation)

We chose to compare the performance metrics of the two "best" regression models in this section.

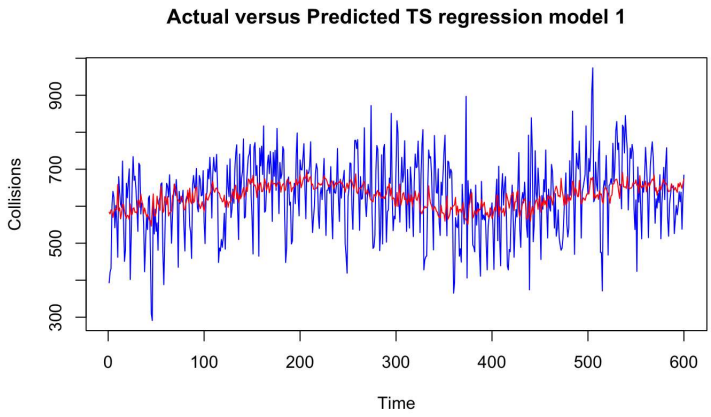


Figure 18.a

Although "Regression Model 1" was identified to be our best model in this section, its fit wasn't promising, as evidenced by the fit of the predicted values against the actual values (*Figure 18.a*).

Model (REG_1): Includes all variables - AWND, TMAX,TMIN, PRCP

Training Performance	**Holdout Performance**
R-squared: 0.098	**R-squared: -0.116**
MAPE: 12.67%	**MAPE: 12.98%**
RMSE: 94.44	**RMSE: 94.44**
MAE: 74.28	**MAE: 83.67**

Figure18.b

Model (REG_2): Excludes PRCP

Training Performance	**Holdout Performance**
R-squared: 0.094	**R-squared: 0.107**

MAPE: 12.71%	**MAPE: 12.89%**
RMSE: 94.66	**RMSE: 104.75**
MAE: 74.59	**MAE: 82.96**

Figure 19.c

Both models perform similarly with low R-squared values and moderate errors. Model 1 (*Figure 18.b*) slightly outperforms Model 2 (*Figure 18.c*) on the holdout set in MAPE and MAE, though both models generalize poorly beyond the training data.

These results show that although the selected variables may have a weak linear relationship with collisions, they do not accurately explain the variance in collisions across time. The negative R-squared on the holdout set implies the model performs worse than simply predicting the mean.

3.3 Residual analysis and stationary check

To asses the residuals of REG_1, the ACF plot and Box-Pierce test were conducted:

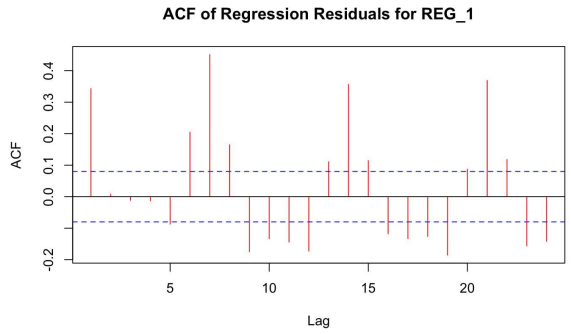


Figure 20

The ACF plot (*Figure 21*) showed significant autocorrelation across multiple lags. The Box-Piece Test: X-squared = 444.81 df= 20, P-value = <2.2e-16.

Thus, we can conclude that the residuals from REG_1 are not white noise, indicating the presence of autocorrelation. This violates one of the core assumptions of linear regression and suggests the need for a more robust time series model. The presence of autocorrelation and poor holdout fit for both show that linear regression may not be suitable for modeling the temporal structure of the data.

4. Stochastic Time Series Models

4.1 ARIMA models (for the variable of interest).

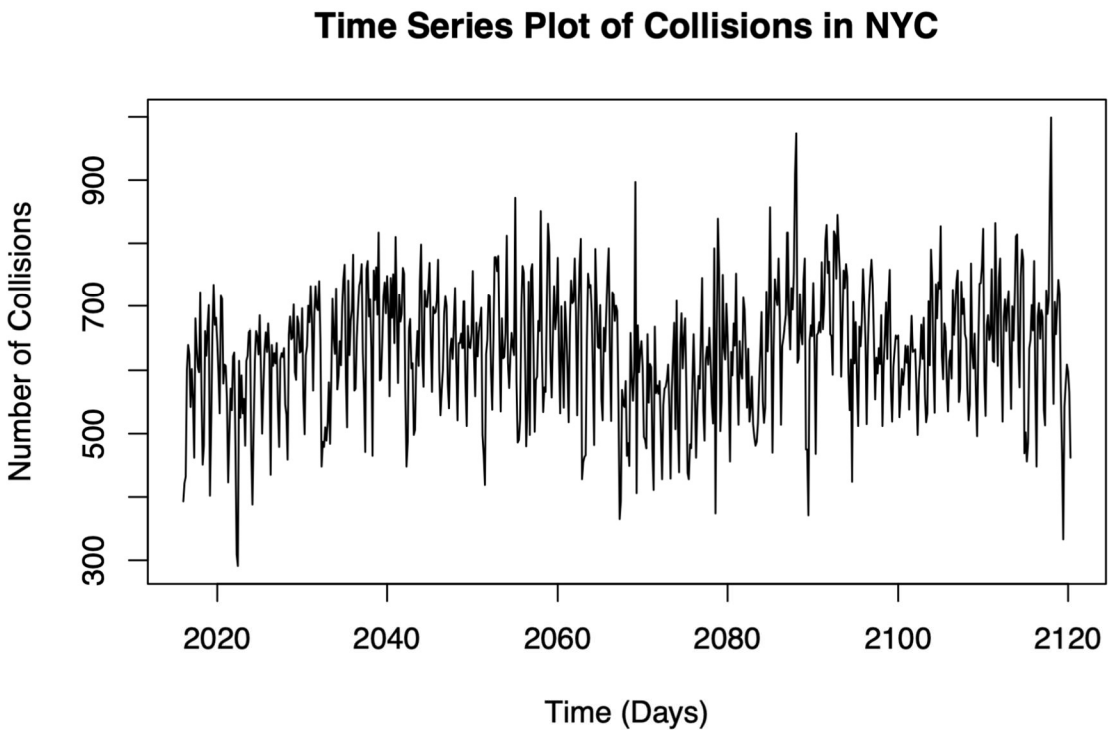


Figure 21

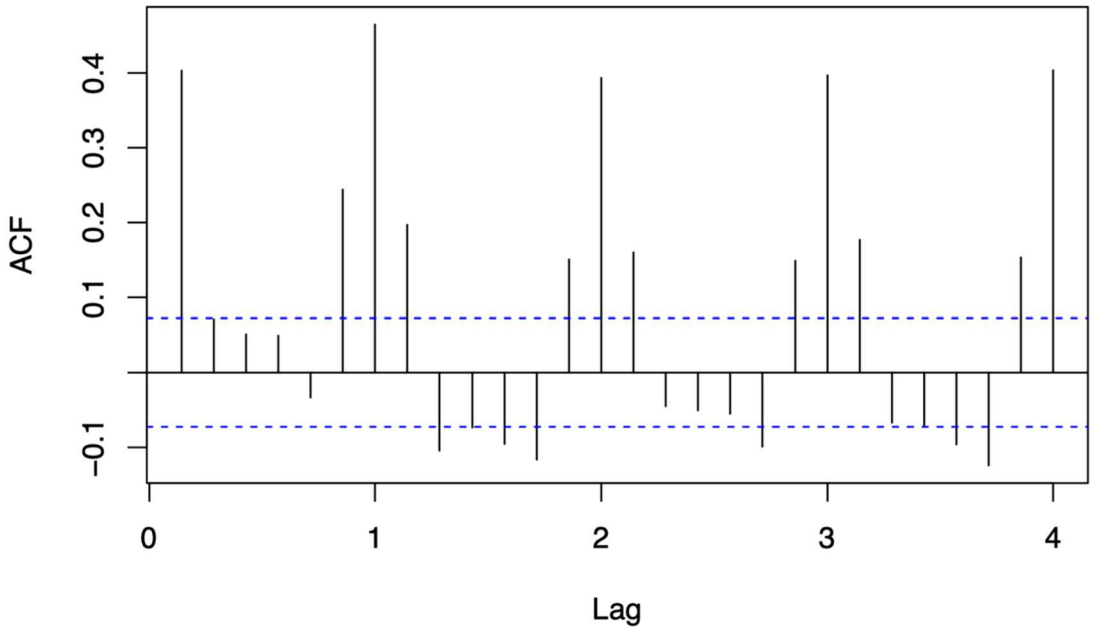


Figure 22

The Time Series plot (*Figure 22*) shows that there is no clear trend or cyclical component in the series; thus, differencing at lag 1 to stabilize the mean and variance of the series is unnecessary. However, as *Figure 23* shows, there seems to be strong autocorrelation between lag 0 (time t=0) and every lag that is a factor of 7 (lags 7, 14, 21, etc.). Thus, the time series needs to be differenced at the (daily) seasonal level to remove the seasonal component so that the structure of the data is stationary (constant mean, constant variance, and *consistent autocorrelation amongst lags, where the autocorrelation is a function of the lag and not of time*). In other words, although there seems to be a constant mean and variance in the time series, which indicates that there is no (clear and obvious) trend component in the data, there is still strong residual seasonal autocorrelation at lags 7, 14, 21 (which implies that there is considerable weekly/daily seasonality in the data). Additionally, because there seems to be nonstationary seasonal behavior in the data (extremely slow decay at the seasonal lags, at factors of 7), seasonal differencing is necessary at s=7, to remove the correlation in the data at the seasonal level.

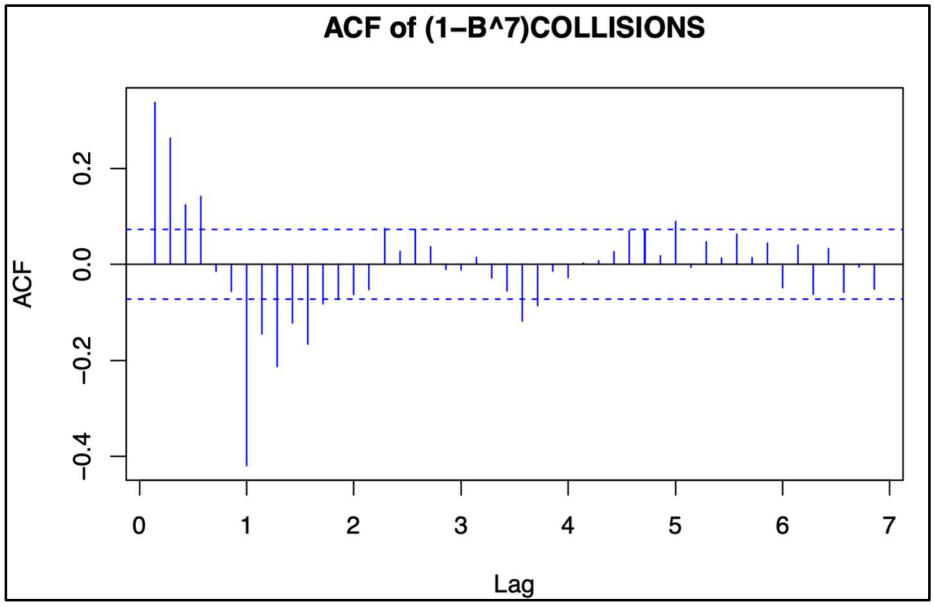


Figure 23

After differencing at the seasonal level (s=7), the structure of the series appears stationary (*Figure 24*).

SEASONAL COMPONENTS: After differencing at the seasonal lags, PACF (*Figure 25*) seems to be decaying slowly at lags 7, 14, 21, 28, before reaching 0 autocorrelation. ACF (*Figure 24*) is chopped off after lag 7; thus, an MA(1)s=7 process could be reasonably fitted to model the series.

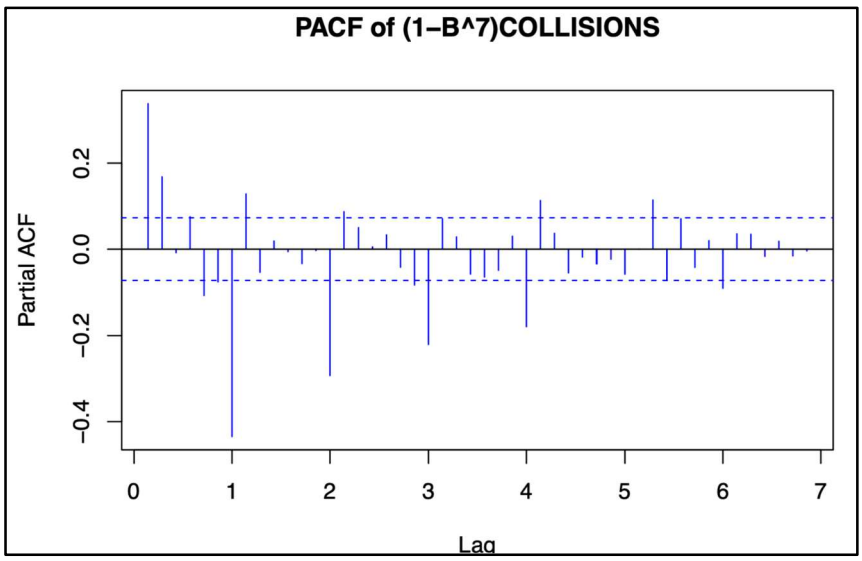


Figure 25

NONSEASONAL COMPONENTS: Perhaps a chopping off after lag 1 (nonseasonal component) in the PACF (*Figure 25*). ACF (*Figure 24*) at nonseasonal lags seems to gradually decay to 0.

By applying a Moving Average Component to the seasonality, the underlying autocorrelation at the weekly seasonal level [lag (s=7)] was effectively removed. Adding the Moving Average component to the difference of Yt and Yt-7 effectively handles and captures most of the residual relationship left over that influences today's number of collisions in NYC with last weeks. However, after fitting the MA(1)s=7 process to the series, there was still correlation at the nonseasonal lags left to be captured. The Box Pierce test returned an extremely small p value, indicating that the autocorrelation at lags 1-24 were not yet white noise. Moreover, there appeared to be extremely slow decay in both the ACF and PACF at the nonseasonal lags, with random lags having autocorrelation and partial autocorrelation statistically different than 0. Thus, nonseasonal differencing was applied to remove any underlying, hard to detect trend/cyclical component in the series.

After adding the nonseasonal differencing to the model, the ACF appeared to decay to 0 in a stationary manner and the PACF appeared to chop off after lag 1. Thus, an AR(1) process was added to the model, to capture the autocorrelations of the nonseasonal component of the series (*not shown)*.

We then argued that the PACF exhibited decay, after adding the AR(1) process to the model, and the ACF chopped off after lag 2; thus, we added an MA(2) process to the model (*not*

```
Series: train_crash
ARIMA(1,1,2)(0,1,1)[7]
Box Cox transformation: lambda= 0

Coefficients:
         ar1      ma1     ma2     sma1
      0.7208  -1.3843  0.4037  -0.9773
s.e.  0.0853   0.1072  0.1001   0.0221

sigma^2 = 0.01412:  log likelihood = 409.99
AIC=-809.99   AICc=-809.89   BIC=-788.07

Training set error measures:
                     ME     RMSE      MAE       MPE     MAPE      MASE
Training set -1.643512 69.56998 50.76395 -1.556262 8.579936 0.6144068
                    ACF1
Training set -0.001224739
```

Figure 24

shown), at the nonseasonal component (to get a more favorable p value from the Box Pierce Test.)

This sequential procedure of integrating a Multiplicative ARMA process to fit to the series, removed (most of) the significant autocorrelations (*Figure 27*) and partial autocorrelations (*Figure 28*) at all nonseasonal and seasonal lags of found in the series, while keeping our model parsimonious to diminish the likelihood of overfitting. Moreover, by moving from using strictly an MA(1)s=7 process to a Mixed ARIMA (1,1,2)x(0,1,1)s=7 process to capture correlations at seasonal and nonseasonal lags (after differencing), the p value of the box pierce test significantly (from 2.2e-16 to 0.2357). This also increased training fit significantly across the board of all

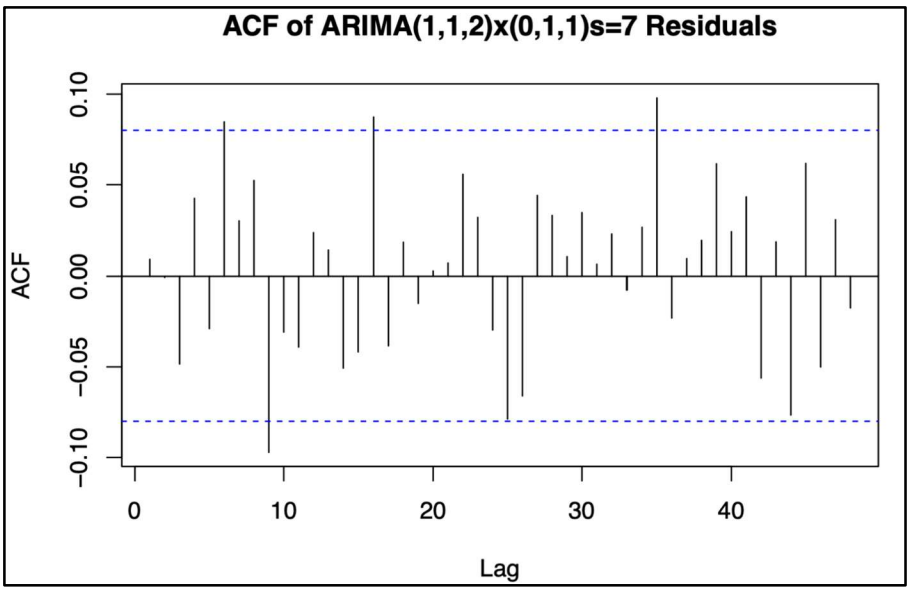


Figure 25

viable error metrics, with each of the coefficients for the phi and theta regressors being statistically significant and different than 0 (*Figure 26*).

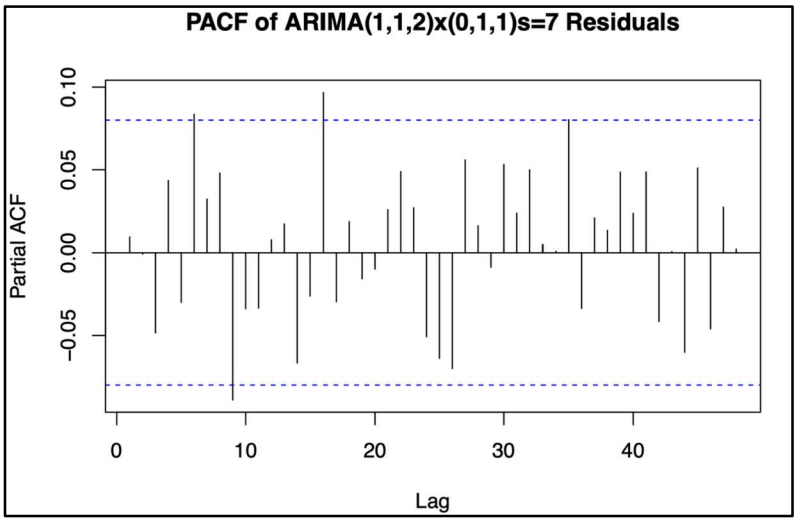


Figure 26

Because the p value of the Box pierce test is now greater than our level of significance (0.05), we fail to reject the null hypothesis that the residuals are no different than white noise. Thus, because the residuals resemble white noise, the correlation between errors at lags 1-24 are statistically equal to 0. Lastly, all coefficients in the model are statistically significant at the 5% level of significance (*Figure 26*). The model can now effectively be used for forecasting at the 5% level of significance.

	Metric	Training_set	Holdout_set
1	MAPE	8.5799361	9.4448525
2	RMSE	69.5699780	78.4295399
3	MAE	50.7639476	58.1144393
4	R2	0.5107106	0.3792855

Figure 27

Fit and predictive metrics for ARIMA(1,1,2)x(0,1,1)s=7 model (*Figure 29*).

4.2) Analysis and modeling of regression model residuals.

Using the best derived time series regression model from section 3.3.

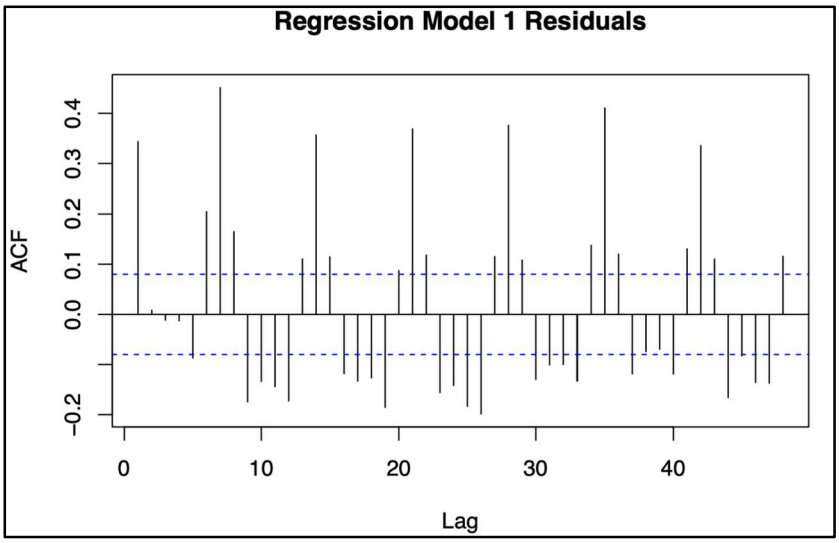


Figure 28

Nonstationary residuals at the seasonal (s=7) lags (*Figure 30*). Thus, need to difference

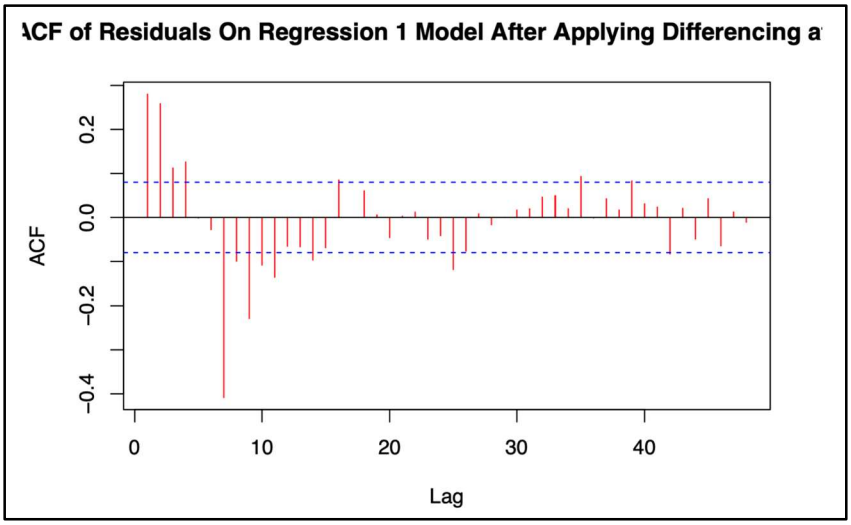


Figure 29

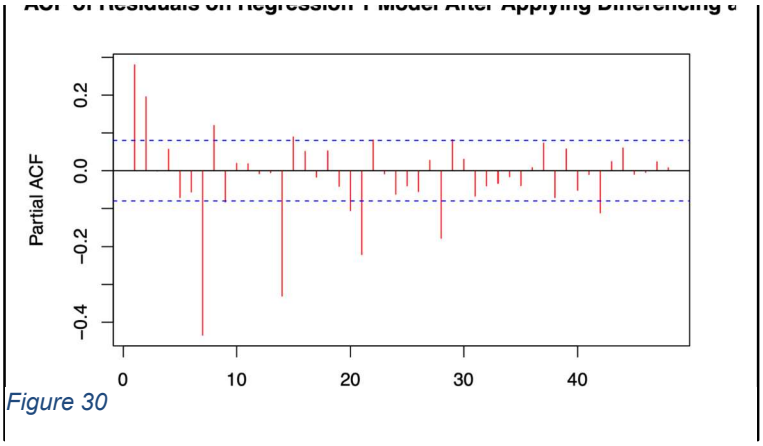


Figure 30

dependent variable at the seasonal level.

After differencing, there appears to be gradual decay in PACF at seasonal lags (*Figure 32*). Chopping off at ACF, lag 7 *(Figure 31*). Therefore, because differenced series indicates stationarity, will try fitting an MA(1) process at the first seasonal lag (7).

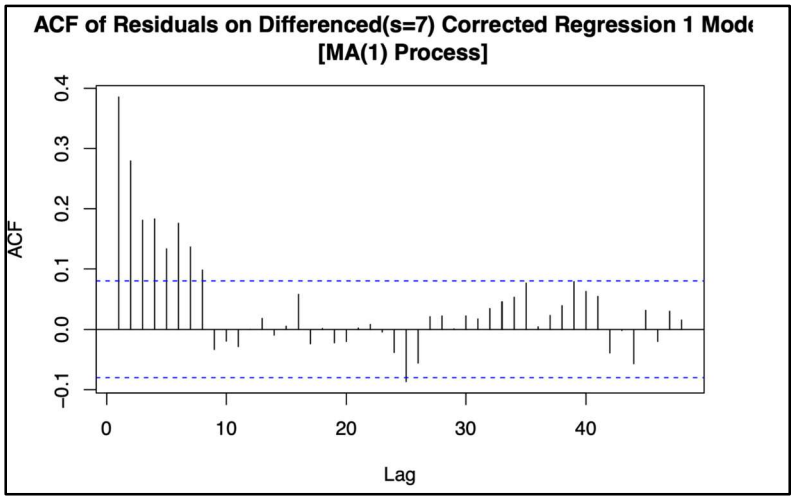


Figure 31

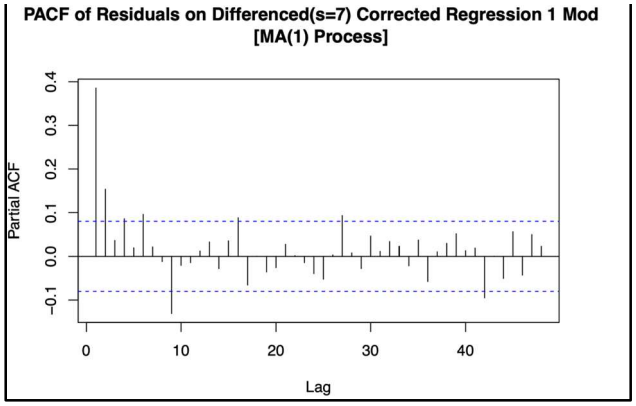


Figure 32

After fitting the MA(1) process to the seasonally differenced series, autocorrelation (*Figure 33*) and partial autocorrelation (*Figure 34*) at the seasonal level seems to be removed; however, the p value returned by the Box Pierce test was still extremely small. Moreover, there seems to be gradual decay in the ACF at the nonseasonal lags (*Figure 33*), with PACF showing a chopping

```
 Box-Pierce test

data:  regarima$residuals
X-squared = 2.6412, df = 1, p-value = 0.1041
```

Figure 33

off at lag 1 (*Figure 34*). Therefore, we added an AR(1) process at the nonseasonal lags to the model.

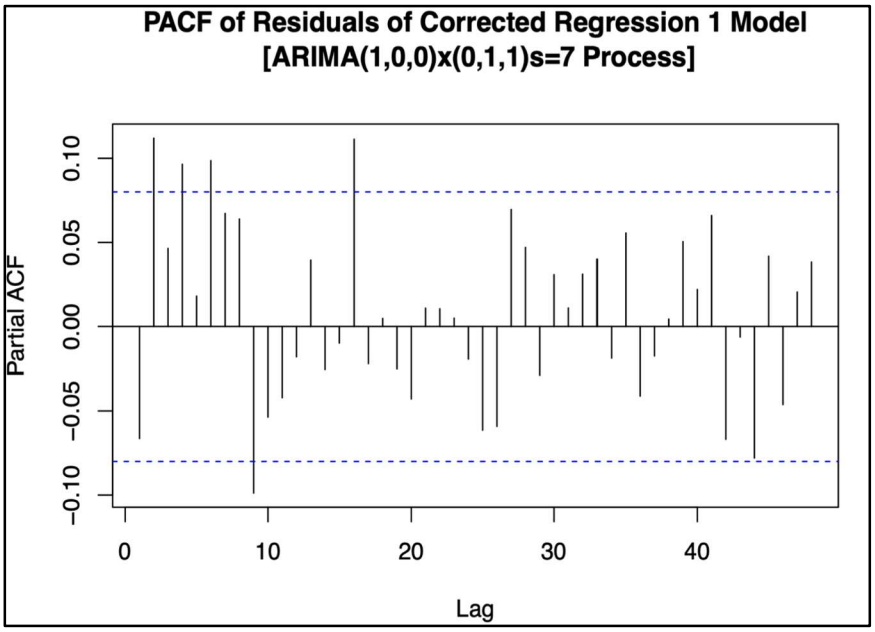


Figure 34

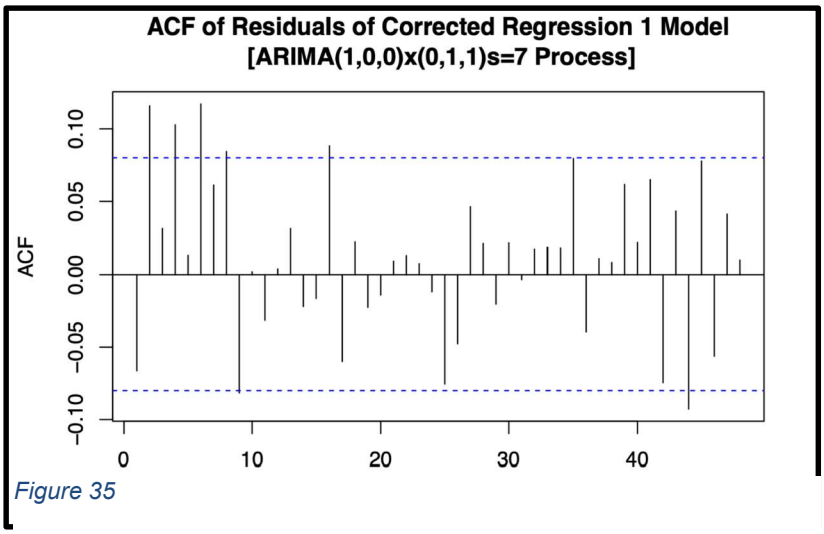


Figure 35

```
Series: train_crash
Regression with ARIMA(1,0,0)(0,1,1)[7] errors

Coefficients:
         ar1     sma1  train_AWND  train_TMAX  train_TMIN
      0.3942  -0.9768      2.0944      4.1983     -1.5587
s.e.  0.0383   0.0222      3.0388      1.0086      1.1205
```

Figure 36

MULTIPLICATIVE ARMA(1,0,0) x (0,1,1) appears to be best parsimonious model at this point (*Figure 37 & 38*). At the 5% level of significance (using the p-value derived from the Box-Pierce Test, at lag=1, *Figure 35*), we fail to reject the null hypothesis that the residuals are white noise; thus, this model can be used for modeling the TS. However, it is notable that neither the 'train_AWND' nor the 'train_TMIN' features are significant (*Figure 38*). They can be removed from the model; thus, will dropped these features from the model and increased the lag in the Box Pierce Test from 1 to 24 to get a more practical and accurate understanding of the

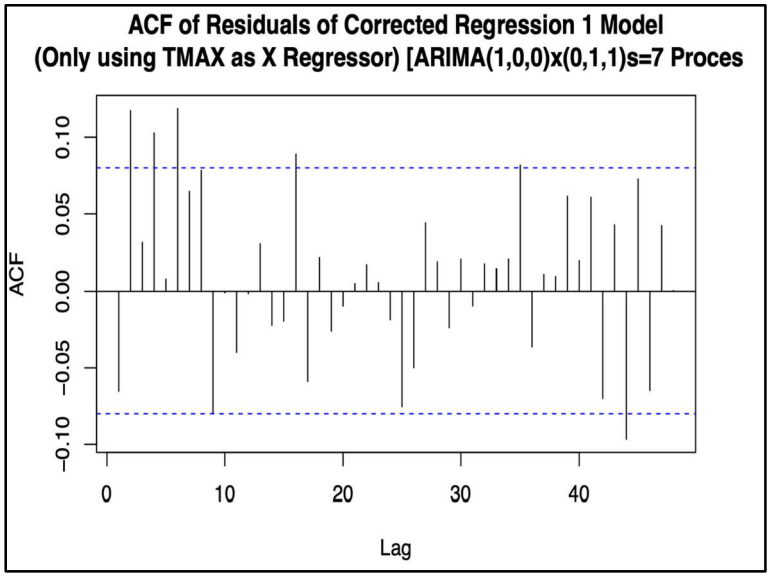


Figure 37

correlation amongst residuals.

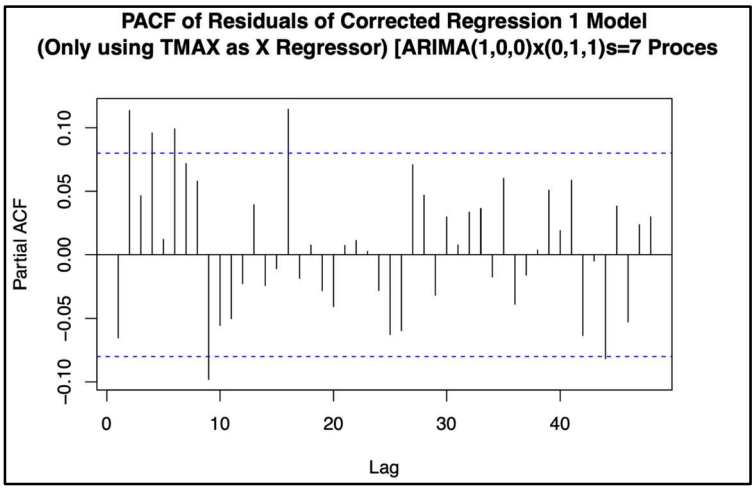


Figure 39

```
 Box-Pierce test

data:  regarima$residuals
X-squared = 46.254, df = 24, p-value = 0.004126
```

Figure 38

After changing the lag to 24 (thus changing the calculated value of the Box Pierce Test), we noticed that the p value of model for this model was still extremely small (*Figure 40)*, indicating that the residuals of the model were still correlated various lags 1-24 and not yet white noise. Moreover, after more closely inspecting the ACF of the residuals (*Figure 39*), there appears to be extremely slow decay to zero at spurious lags throughout, which could be evidence of nonstationarity. Thus, differencing at the nonseasonal lags will likely be beneficial.

```
Series: train_crash
Regression with ARIMA(1,0,0)(0,1,1)[7] errors

Coefficients:
          ar1      sma1   xreg
      0.3953  -0.9791  2.834
s.e.  0.0382   0.0230  0.430
```

Figure 40

However, it is worth mentioning that all coefficients in the model (phi, theta, and the x independent regressor) are now statistically significant after dropping the redundant regressors (*Figure 42*).

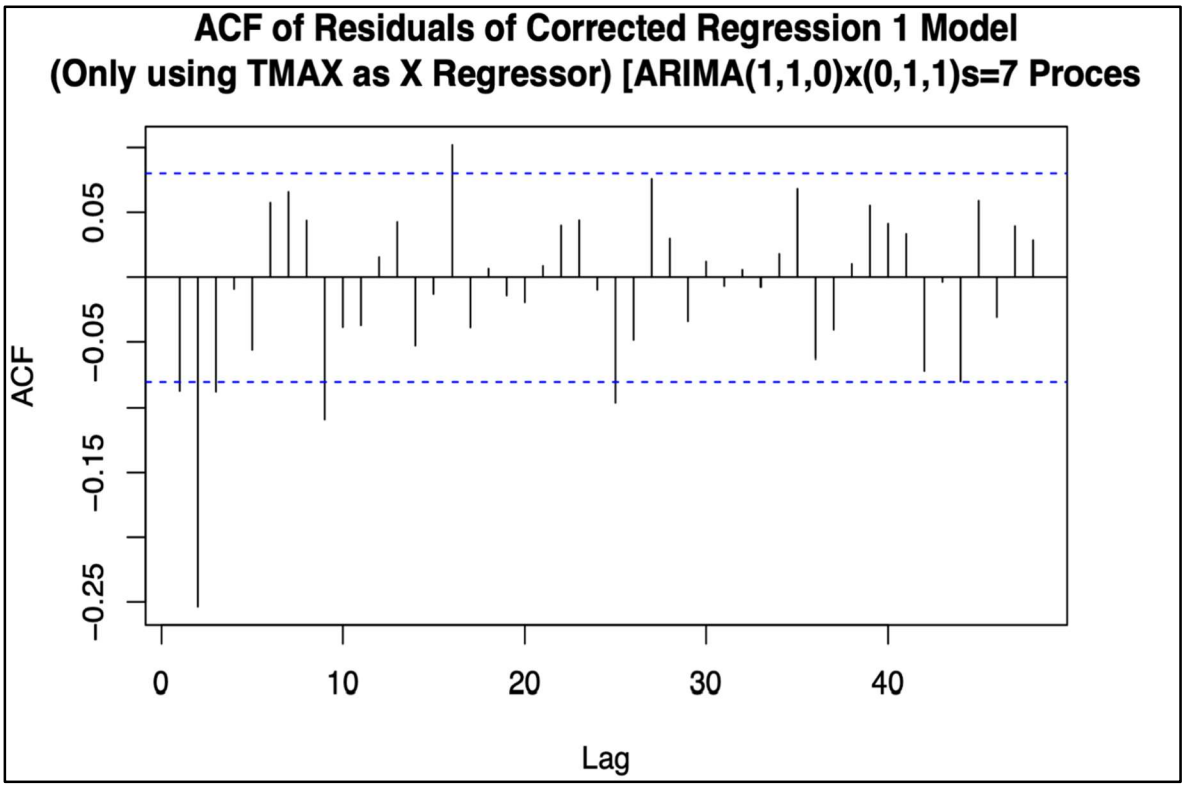


Figure 41

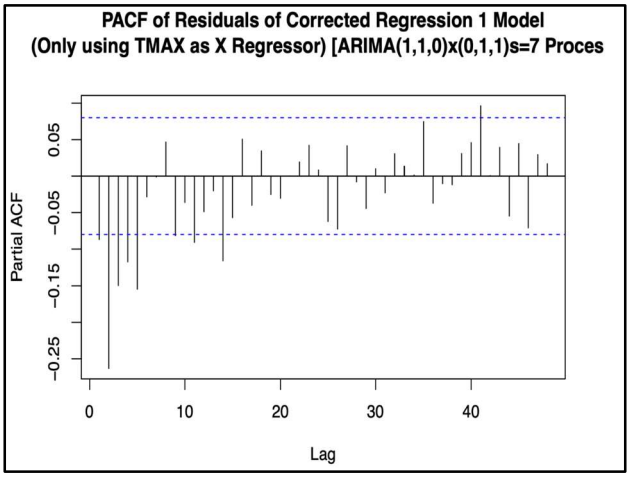


Figure 42

The Box Pierce Test p-value was still extremely small after making the aforementioned adjustments (*Figure 44*); therefore, the corrective time series regression model could be improved. It is positive, however, that the ACF shows more gradual decay to zero at lags 1-48, indicating differencing at nonseasonal level makes residuals more stationary (*Figure 43*). However, may also be easier to interpret a gradual decay in the PACF (*Figure 45*) and a chopping off at lag 2 in the ACF (*Figure 43*); thus, will leverage an AR(2) process to the nonseasonal differenced series, rather than an AR(1) process.

```
 Box-Pierce test

data:  regarima$residuals
X-squared = 76.937, df = 24, p-value = 1.857e-07
```

Figure 43

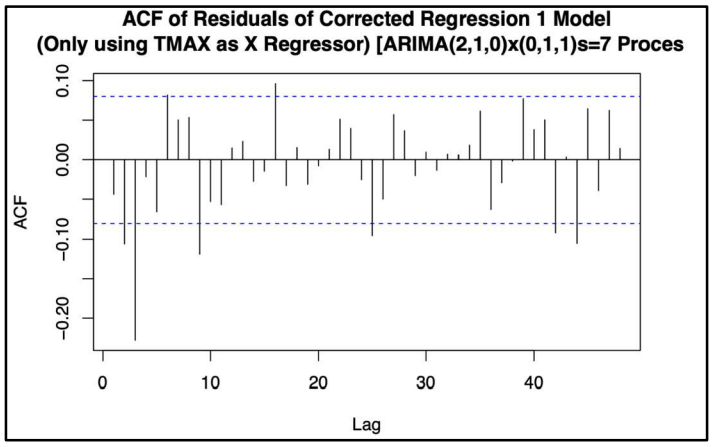


Figure 46

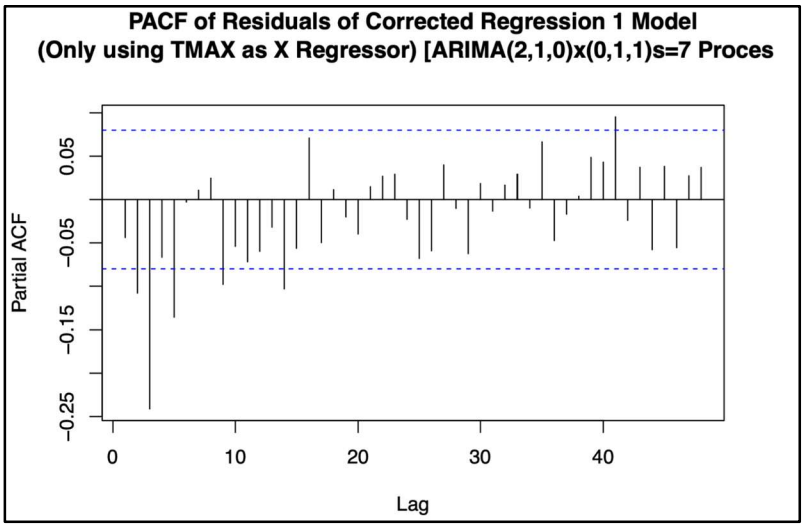


Figure 45

```
 Box-Pierce test

data:  regarima$residuals
X-squared = 72.164, df = 24, p-value = 1.023e-06
```

Figure 44

Box Pierce p-value was still incredibly small (*Figure 48)*. We interpreted a gradual decay in residuals (lags 1-48) in PACF (*Figure 47*) and chopping off at lag 3 in ACF (*Figure 46*). Thus, we leveraged an ARMA(2,1,3) at the nonseasonal level, rather than an AR(2,1,0) process to see how this fit changed performance outputs and the calculated value of Box Pierce test.

```
Series: train_crash
Regression with ARIMA(2,1,3)(0,1,1)[7] errors

Coefficients:
          ar1     ar2      ma1      ma2     ma3     sma1     xreg
      -0.1671  0.6982  -0.5292  -0.8512  0.3803  -0.9669  2.5442
s.e.   0.0867  0.0801   0.1097   0.0944  0.0997   0.0216  0.5133
```

Figure 49

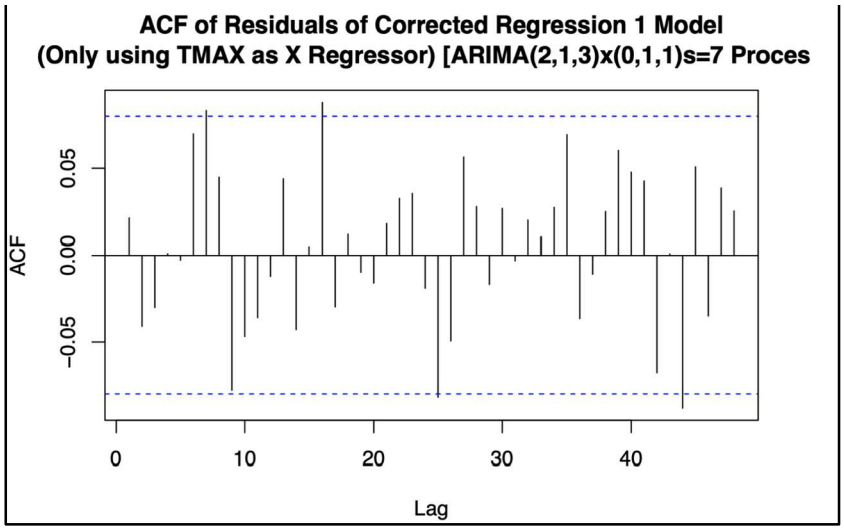


Figure 48

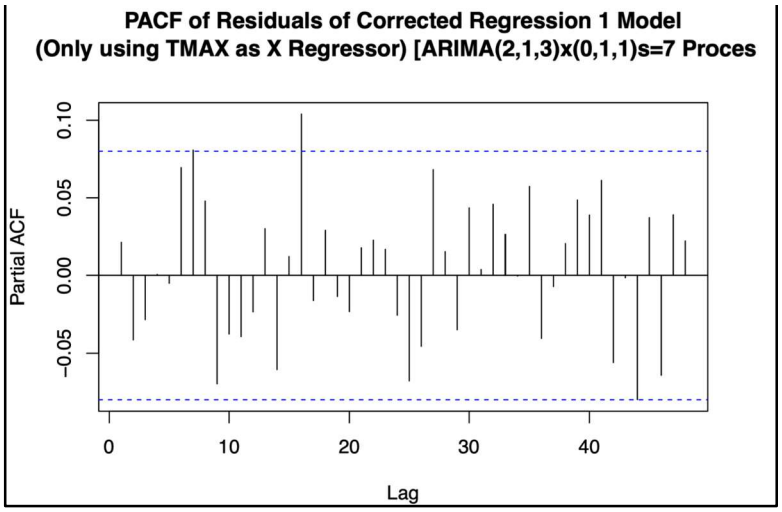


Figure 47

All phi coefficients, theta coefficients, and independent regressor coefficients in the ARIMA(2,1,3)x(0,1,1)s=7 model are statistically significant (*Figure 51*).

The Box Pierce Test of this Multiplicative ARIMA process returned a value of .38 (*Figure 52*) and none of the lags of the residuals in ACF appear to explicitly lie outside of the S.E. bounds (*Figure 50*), indicating that autocorrelations between residuals at the aforementioned lags have been removed. Therefore, at the 5% level of significance (using the p-value derived from the Box-Pierce Test, for lags 1-24), we fail to reject the null hypothesis that the residuals are white noise.

Additionally, only one lag in the PACF (lag 16) lies drastically outside of the S.E. bounds (*Figure* 49) and trying to remove this partial correlation would unnecessarily increase model

```
 Box-Pierce test

data:  regarima$residuals
X-squared = 25.482, df = 24, p-value = 0.38
```

Figure 50

complexity. Thus, the ARIMA(2,1,3) x (0,1,1)s=7 appears to be best parsimonious model, and this model can be used for modeling the NYC Collision TS effectively.

	Metric	Training_set	Holdout_set
1	MAPE	8.4386159	9.3083864
2	RMSE	68.3102420	80.6537691
3	MAE	49.5701460	59.3142298
4	R2	0.5282698	0.3435799

Figure 51

Performance metrics for ARIMA(2,1,3)x(0,1,1)s=7 process fit and predictive performance metrics on training and holdout samples (*Figure 53*).

4.3) Analysis and modeling of deterministic time series residuals.

In this section, we analyzed the residuals of the (best) deterministic time series model from SECTION 2.3.

The ANOVA summary of the deterministic model (*Figure 9*) showed that all dummy variables in model (regressors) were statistically significant EXCEPT the dummy variable for February (implying that NYC Collisions in February are not statistically different than those in January). Thus, we dropped it to have a more parsimonious model. Thus, the intercept now effectively represents January, February, and Friday collisions in NYC.

We also decided to remove the trend component from the model because it was not statistically different than 0 (*Figure 9)* and keeping it unnecessarily increases model complexity and risk of overfitting.

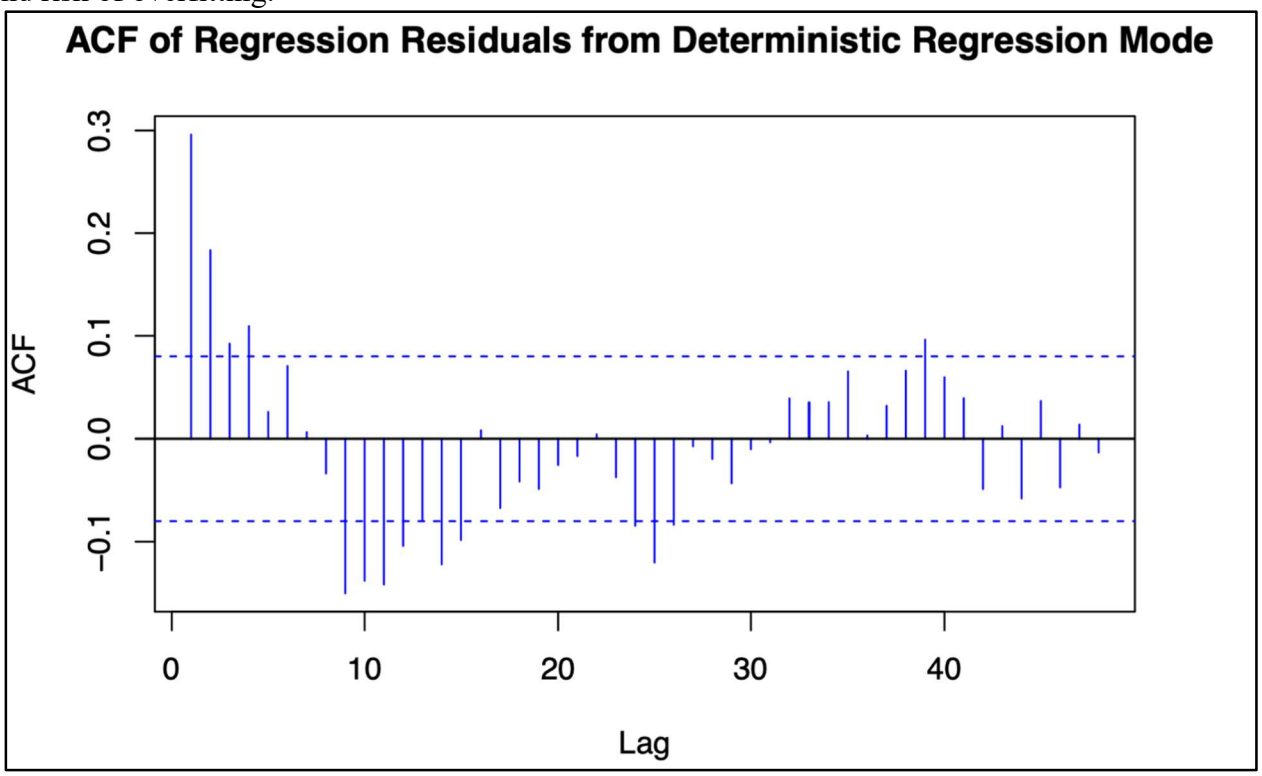


Figure 52

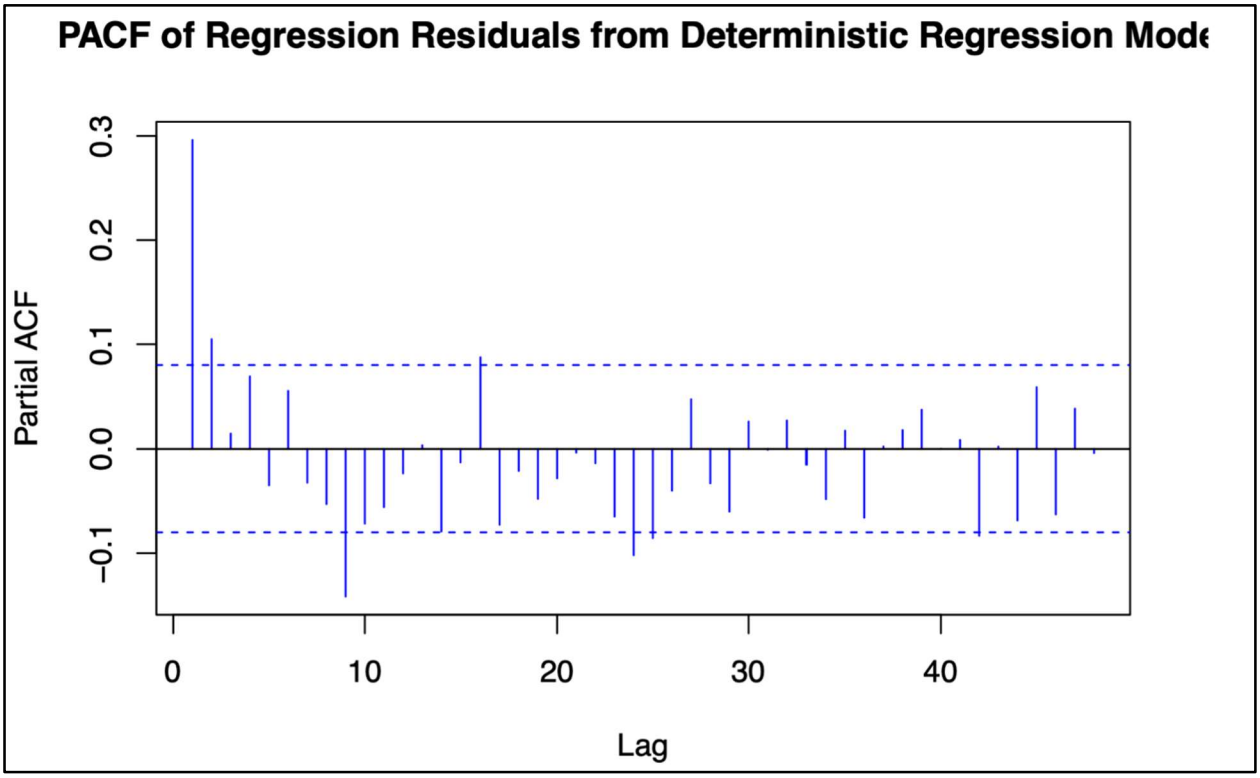


Figure 53

After removing redundant features from the deterministic, we can see that the residuals in the ACF could either be interpreted as stationary or nonstationary (*Figure 54*). To argue for stationarity, we inferred a gradual decay to 0 in the ACF from lags 1-48; thus, we initially concluded autocorrelation of residuals are stationary and can be modeled by an ARIMA process.

In observing the PACF of residuals (*Figure 55*), there seems to be chopping off after lag 1; thus, because (initial interpretation) decay in ACF, will try a AR(1) process. It is worth mentioning though, if arguing for nonstationary residuals that can't be modeled, because the PACF has a handful of spurious lags that lie slightly outside of the SE bounds, the underlying residuals of the TS may be nonstationary, as they do not decay quickly to 0 in the PACF (with somewhat similar behavior appearing in the ACF). Thus, we decided to attempt inducing

stationarity in the residuals, if needed, so the deterministic model could be corrected for forecasting effectively.

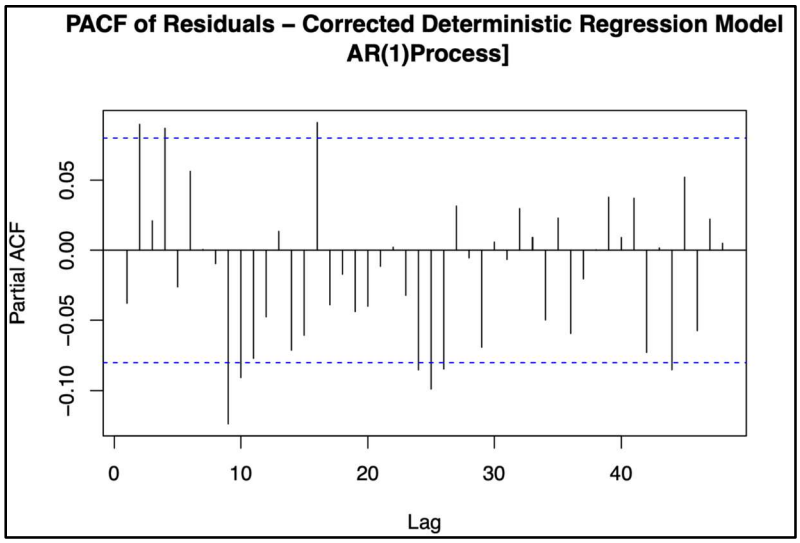


Figure 55

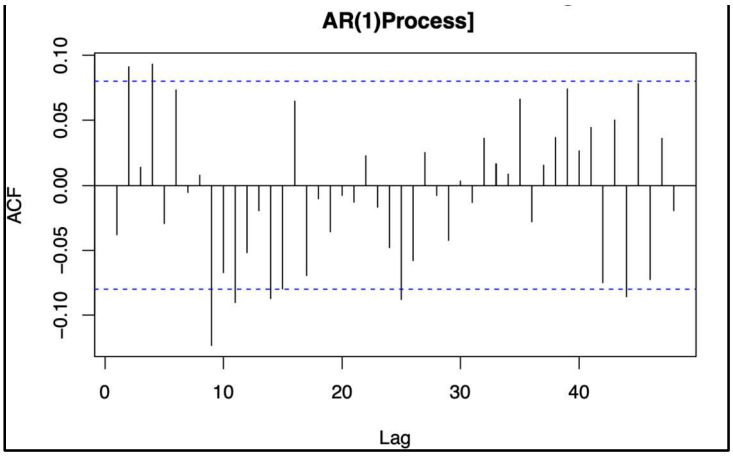


Figure 54

Because the Box Pierce Test returned a small p-value (indicating that there is still underlying correlation between residuals at lags 1-24), with an ACF having a handful of lags lying outside the two standard error bounds (*Figure 56*) and the PACF still exhibits some spurious residuals lying outside of the SE bounds between lags 1-48 (*Figure* 57), the residuals are likely not yet stationary due to the slightly hidden trend/cyclical structure of the underlying

TS. Will try to induce stationarity by differencing to see if ACF/PACF is made easier to interpret and more closely exhibits stationarity so an AR or MA process can be applied to the residuals.

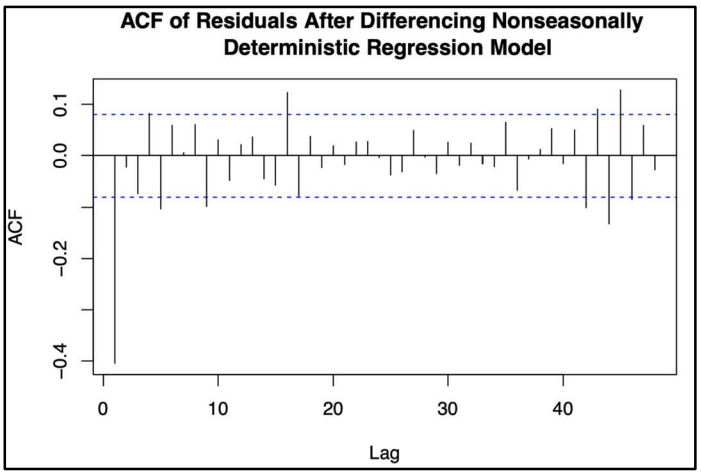


Figure 56

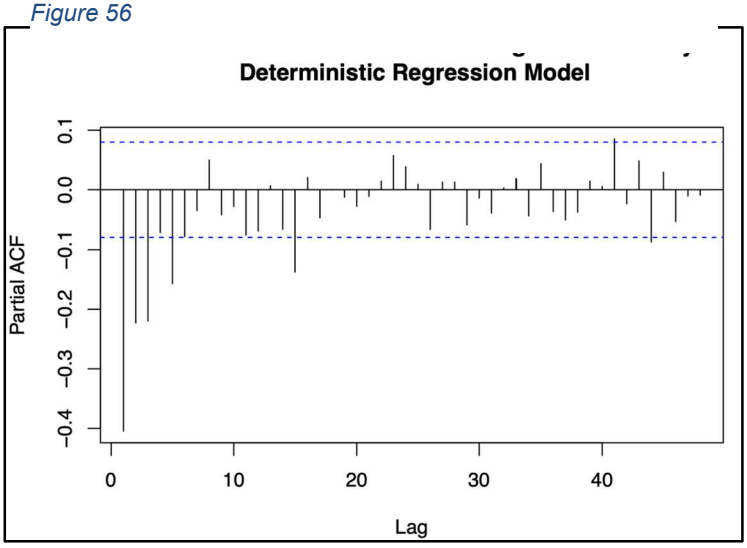


Figure 57

After differencing, ACF appears to chop off at lag 1 (*Figure 58*). PACF gradually decays to 0 (*Figure 59*). Thus, we can infer the differenced series is stationary. Will attempt to add an MA(1) process to differenced corrected deterministic model.

After adding the MA(1) processed to the differenced corrective model, the PACF was interpreted to be chopping off at lag 2 (*not shown)*; thus, we added an AR(2) process to ultimately try an ARMA(1,1,2) Corrected Model.

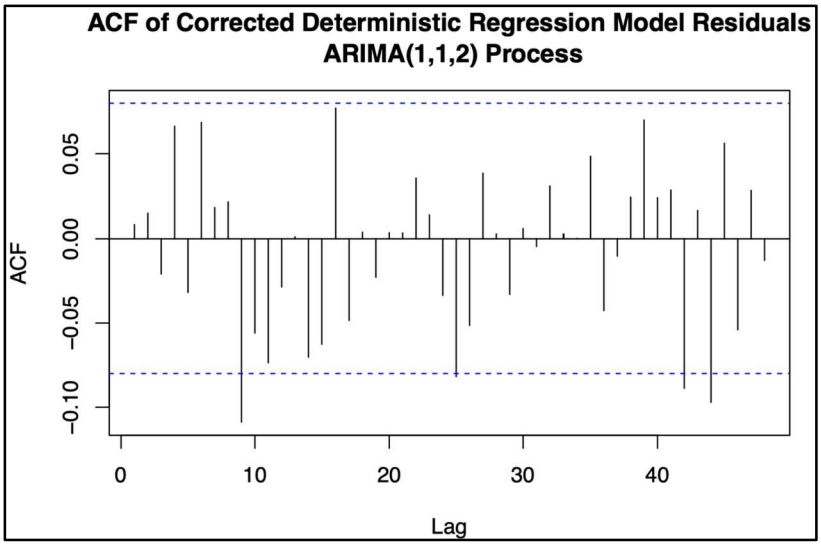


Figure 58

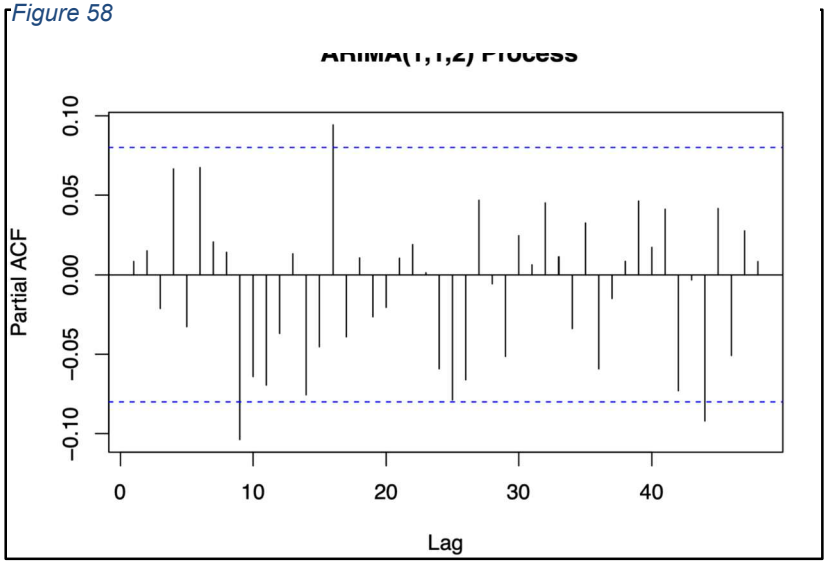


Figure 60

```
 Box-Pierce test

data:  ar_fit$residuals
X-squared = 31.935, df = 24, p-value = 0.1286
```

Figure 59

Because the ACF of the Residuals only show a few lags that lie slightly on or above the two standard error bounds (*Figure 60*), the PACF (*Figure 62*) only has a few lags that lie above the SE bounds (lags 9, 16, and 45), and the p-value of the Box Pierce Test is greater than the level of significance of 5% (*Figure 61)*, we can infer that the autocorrelation and partial

autocorrelation of the residuals have been (mostly) removed by applying the Corrected ARIMA Process to the model. Thus, because we fail to reject the null hypothesis that the residuals are white noise and the regressor and phi(1) coefficients are all statistically significant, the Corrected Deterministic Regression Model can effectively be used for forecasting.

	Metric	Training_set	Holdout_set
1	MAPE	8.2211433	9.1198279
2	RMSE	66.9200207	79.0561726
3	MAE	48.4677212	57.8376256
4	R2	0.5472753	0.3693272

Figure 61

Performance metrics for Corrected Deterministic Regression Model (*Figure 63*).

5.Conclusions

We conclude with a comparison of the models derived across all sections, in terms of training sample fit and hold-out sample performance.

Although the Time Series Regression Model in SECTION 2.2 (using daily and monthly seasonal regressors in modeling) produced the best metrics in terms of training MAPE (8.75%), training RMSE (70.44), training MAE (51.44), holdout MAPE (9.03%), holdout RMSE (77.80), and holdout MAE (56.86) it's results are biased due to the correlated residuals.

As SECTION 4 (ARIMA models) are the only models that do not violate a key assumption of regression modeling (residuals are uncorrelated so that performance metrics are accurate and not misrepresented), we conclude that these are the best models to be used to fit and forecast on the Daily NYC Collision Data.

The model derived from SECTION 4.1 produced the best holdout RMSE (78.43).

The model derived from SECTION 4.2 performed suboptimally compared to the models produced in SECTION 4.1 and SECTION 4.3 across all viable performance metrics.

The model derived from SECTION 4.3 produced the best training MAPE (8.22%), training RMSE (9.92), training MAE (48.47), holdout MAPE (9.12%), and holdout MAE (57.84).

Thus, SECTION 4.3 would ultimately be the model we would move forward with in modeling the Daily NYC Collision Data.

APPENDIX

project_4219

2025-03-24

#Matthew, Jonathan, Jesse, Henrique #setwd("C:/Users/flank/Desktop/4219_FA/Project") —This is the working directory for this

#Importing important packages

```
library(tseries)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
#install.packages("Metrics")
library(Metrics) #For computing evaluation metrics
```

```
## Warning: package 'Metrics' was built under R version 4.4.3
```

```
library(dplyr) #Need for some basic data cleaning
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Introduction) Reading in the data, creating monthly and daily

seasonal dummy variables.

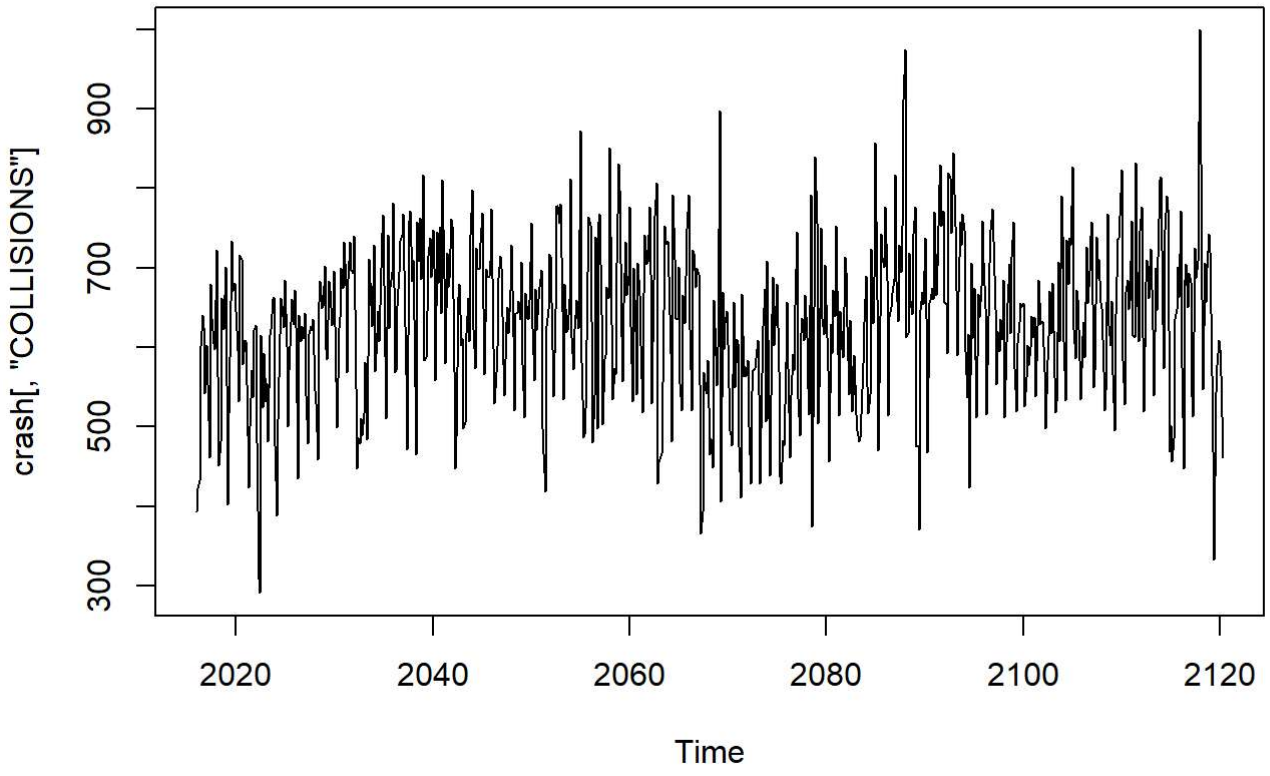
```
#Reading in the data
all= read.csv("NYC_Daily_Collisions (1).csv") #Name of csv file to read in
#JON: below is how I am going about reading in the data, using my absolute directory path
#all = read.csv("C:\\Users\\dell\\OneDrive\\Desktop\\GWU Spring '25\\Forecasting Analytics\\Semester Long Project\\NYC_Daily
_Collisions.csv")
all = data.frame(all) #Combine all columns

#Creating Month Column
all$DATE <- as.Date(all$DATE, format = "%m/%d/%Y") #Convert date data to proper date col
all$Month <- as.factor(format(all$DATE, "%b")) #Have month be a
month_order <- c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec") #Ensure that the order
of months is understood by R, before it was ordering it by alphabetical
all$Month <- factor(all$Month, levels = month_order) #See above

#Create Time Series Object
crash = ts(data = all, start = c(2016,1), frequency = 7) #1/1/16 is a friday, frequency is 7 because seasonal cycle we are o
bserving is daily (7 days in a week)

#Create Column for day of week
crash_day = cycle(crash) #This creates a cycle representing each day of the week
all$WKDY = crash_day #Create column for weekday 1-7

#Create training and holdout samples
train_crash <- crash[,"COLLISIONS"][1:600] #600 observations for the training data
test_crash <- crash[,"COLLISIONS"][601:731] #131 observations for the holdout (test) data

train_day <- crash_day[1:600]#The day of week to be used for the training data
test_day <- crash_day[601:731]#The day of the week to be used for the holdout (test) data

#Create training data for TMAX col
train_TMAX = crash[,"TMAX"][1:600] #Max temperature of the day
test_TMAX = crash[,"TMAX"][601:731]

#Create training data for PRCP col
train_PRCP = crash[,"PRCP"][1:600] #precipitation in (inches or centimeters?) for the day
test_PRCP = crash[,"PRCP"][601:731]
```

```
# Creation of manual dummy variables for month
n=nrow(all) #Use Nrow to get row count of df
m1=rep(0,n)
m2=rep(0,n)
m3=rep(0,n)
m4=rep(0,n)
m5=rep(0,n)
m6=rep(0,n)
m7=rep(0,n)
m8=rep(0,n)
m9=rep(0,n)
m10=rep(0,n)
m11=rep(0,n)
m12=rep(0,n)

for (i in 1:n){
  if (all$Month[i]=="Jan") m1[i]=1 else m1[i]=0 # When month column is ="jan" 1 is assigned to first month dummy indicator
  if (all$Month[i]=="Feb") m2[i]=1 else m2[i]=0
  if (all$Month[i]=="Mar") m3[i]=1 else m3[i]=0
  if (all$Month[i]=="Apr") m4[i]=1 else m4[i]=0
  if (all$Month[i]=="May") m5[i]=1 else m5[i]=0
  if (all$Month[i]=="Jun") m6[i]=1 else m6[i]=0
  if (all$Month[i]=="Jul") m7[i]=1 else m7[i]=0
  if (all$Month[i]=="Aug") m8[i]=1 else m8[i]=0
  if (all$Month[i]=="Sep") m9[i]=1 else m9[i]=0
  if (all$Month[i]=="Oct") m10[i]=1 else m10[i]=0
  if (all$Month[i]=="Nov") m11[i]=1 else m11[i]=0
  if (all$Month[i]=="Dec") m12[i]=1 else m12[i]=0
}
# Monthly dummies for train
m1_train <- m1[1:600]
m2_train <- m2[1:600]
m3_train <- m3[1:600]
m4_train <- m4[1:600]
m5_train <- m5[1:600]
m6_train <- m6[1:600]
m7_train <- m7[1:600]
m8_train <- m8[1:600]
m9_train <- m9[1:600]
```

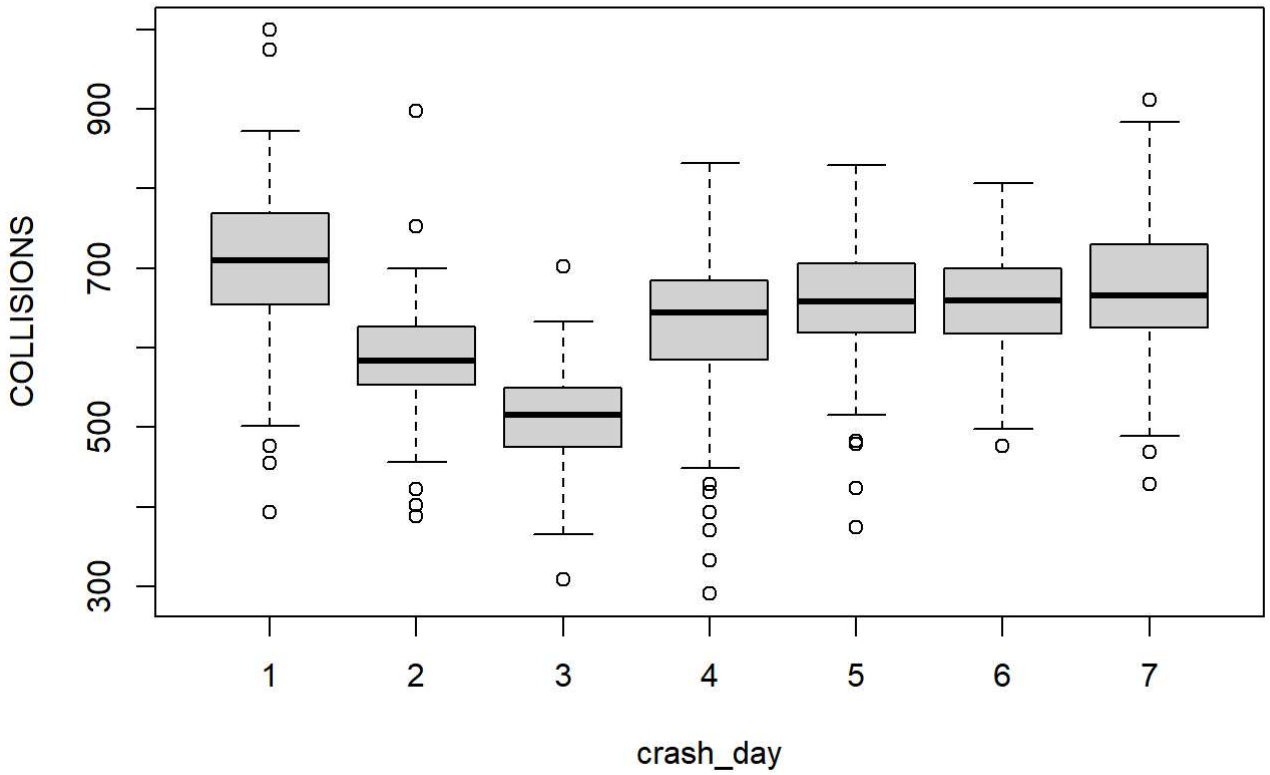
```
m10_train <- m10[1:600]
m11_train <- m11[1:600]
m12_train <- m12[1:600]

# Month dummies for test
m1_test <- m1[601:731]
m2_test <- m2[601:731]
m3_test <- m3[601:731]
m4_test <- m4[601:731]
m5_test <- m5[601:731]
m6_test <- m6[601:731]
m7_test <- m7[601:731]
m8_test <- m8[601:731]
m9_test <- m9[601:731]
m10_test <- m10[601:731]
m11_test <- m11[601:731]
m12_test <- m12[601:731]
```

```
# Creation of manual dummy variables for day of the week
d1=rep(0,n) #Friday
d2=rep(0,n) #Saturday
d3=rep(0,n) #Sunday
d4=rep(0,n) #Monday
d5=rep(0,n) #Tuesday
d6=rep(0,n) #Wednesday
d7=rep(0,n) #Thursday

for (i in 1:n){
  if (all$WKDY[i]== 1) d1[i]=1 else d1[i]=0 # When wkdy column is = 1, 1 is assigned to first wkdy dummy indicator
  if (all$WKDY[i]== 2) d2[i]=1 else d2[i]=0
  if (all$WKDY[i]== 3) d3[i]=1 else d3[i]=0
  if (all$WKDY[i]== 4) d4[i]=1 else d4[i]=0
  if (all$WKDY[i]== 5) d5[i]=1 else d5[i]=0
  if (all$WKDY[i]== 6) d6[i]=1 else d6[i]=0
  if (all$WKDY[i]== 7) d7[i]=1 else d7[i]=0
}

# Weekday dummies for train
d1_train <- d1[1:600]
d2_train <- d2[1:600]
d3_train <- d3[1:600]
d4_train <- d4[1:600]
d5_train <- d5[1:600]
d6_train <- d6[1:600]
d7_train <- d7[1:600]

# Weekday dummies for test
d1_test <- d1[601:731]
d2_test <- d2[601:731]
d3_test <- d3[601:731]
d4_test <- d4[601:731]
d5_test <- d5[601:731]
d6_test <- d6[601:731]
d7_test <- d7[601:731]
```

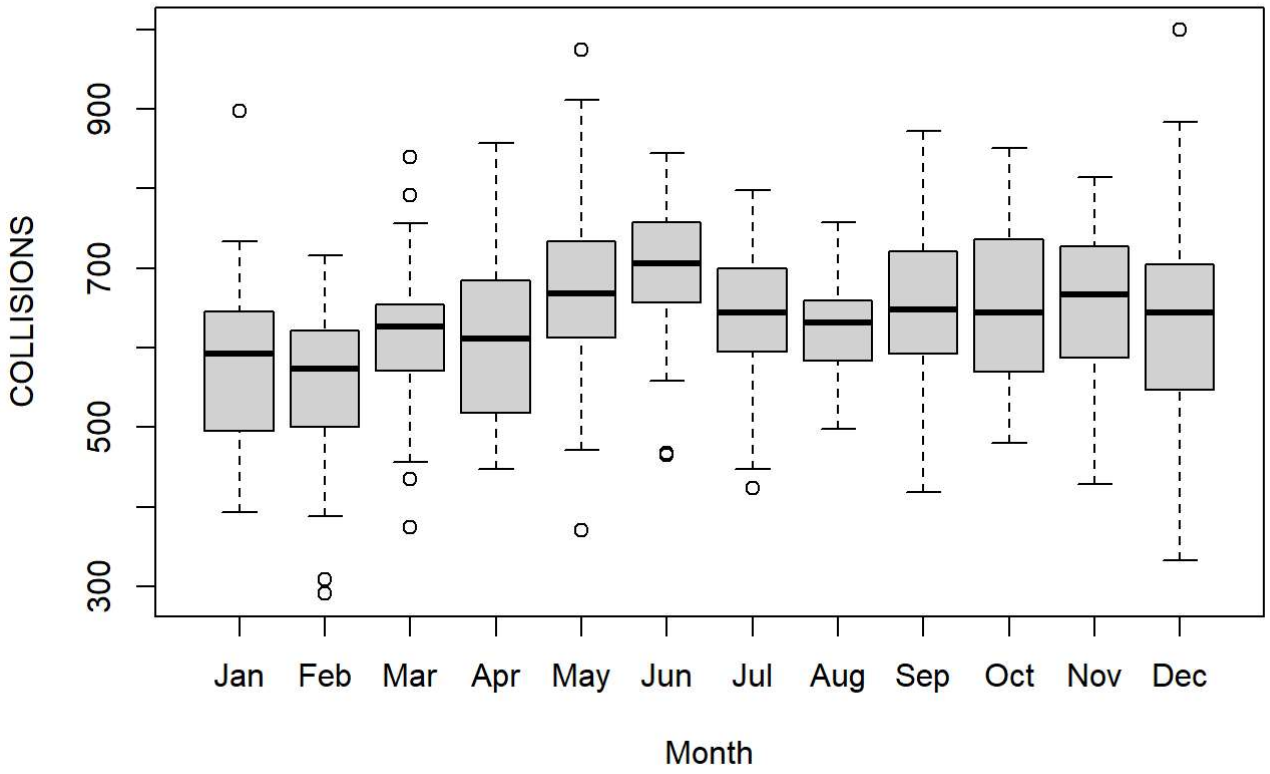
```
#Time Series Plot of variable of interest (Collisions)
ts.plot(crash[,"COLLISIONS"]) #plot of the time series of the data
```



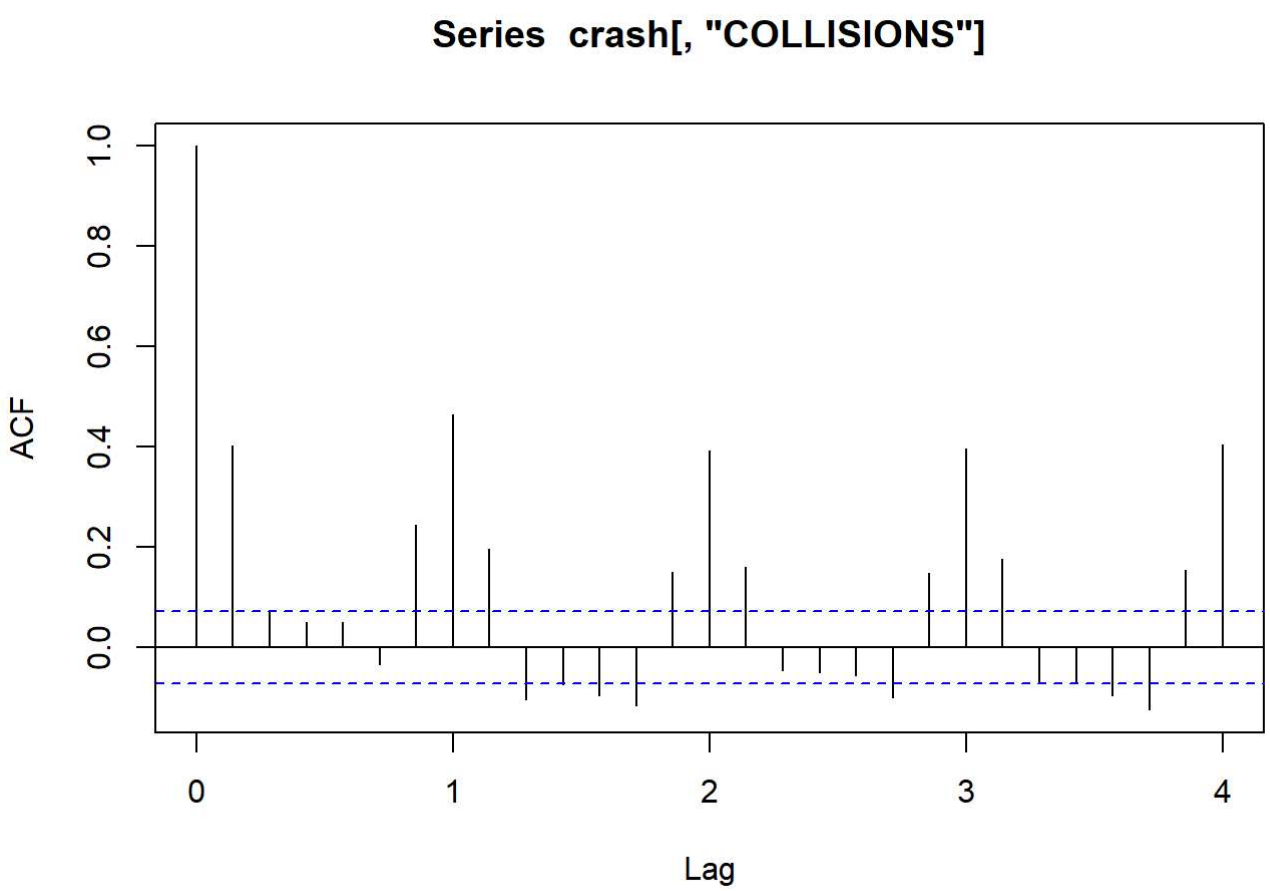

```
#Seasonal Box Plot for day of week
boxplot(COLLISIONS~crash_day, data = all) #plot of the seasonality looking by day of the week, 2 and 3 are Saturday and Sund
ay, so it makes sense that the avg daily crashes are lower than that of the week days
```



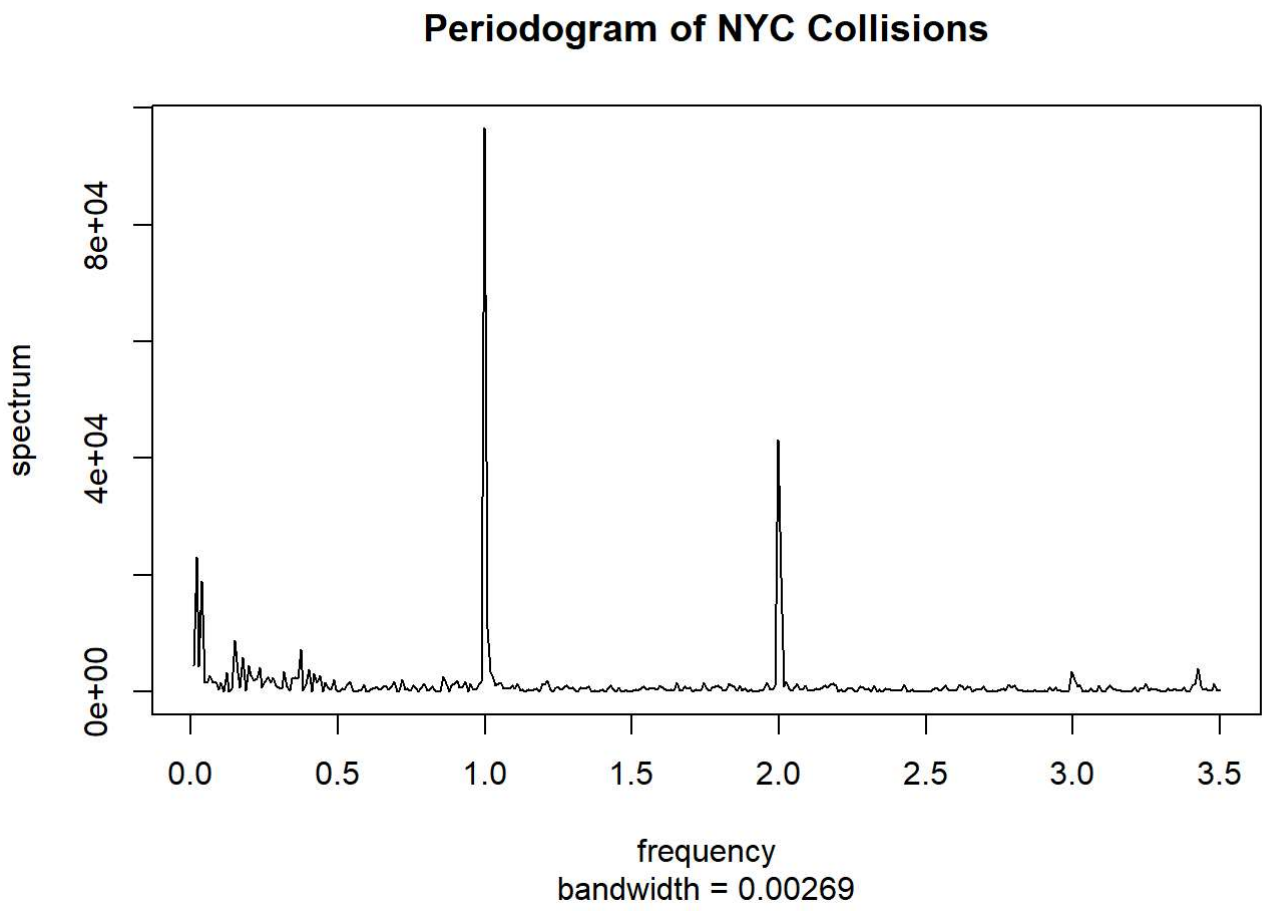

```
#Seasonal Box Plot for month of year
boxplot(COLLISIONS~Month, data = all)
```




```
#ACF Plot of the collision data
acf(crash,"COLLISIONS"], plot = TRUE) #ACF looking at the first 27 lags, it appears that there is a pattern/cycle in the AC
F, which suggests seasonality in the data, additionally multiple lags appear to be surpassing the 2 standard error bounds
```




```
# Periodogram to identify dominant cycles/frequencies
spectrum(crash[,"COLLISIONS"], log = "no", main = "Periodogram of NYC Collisions")
```

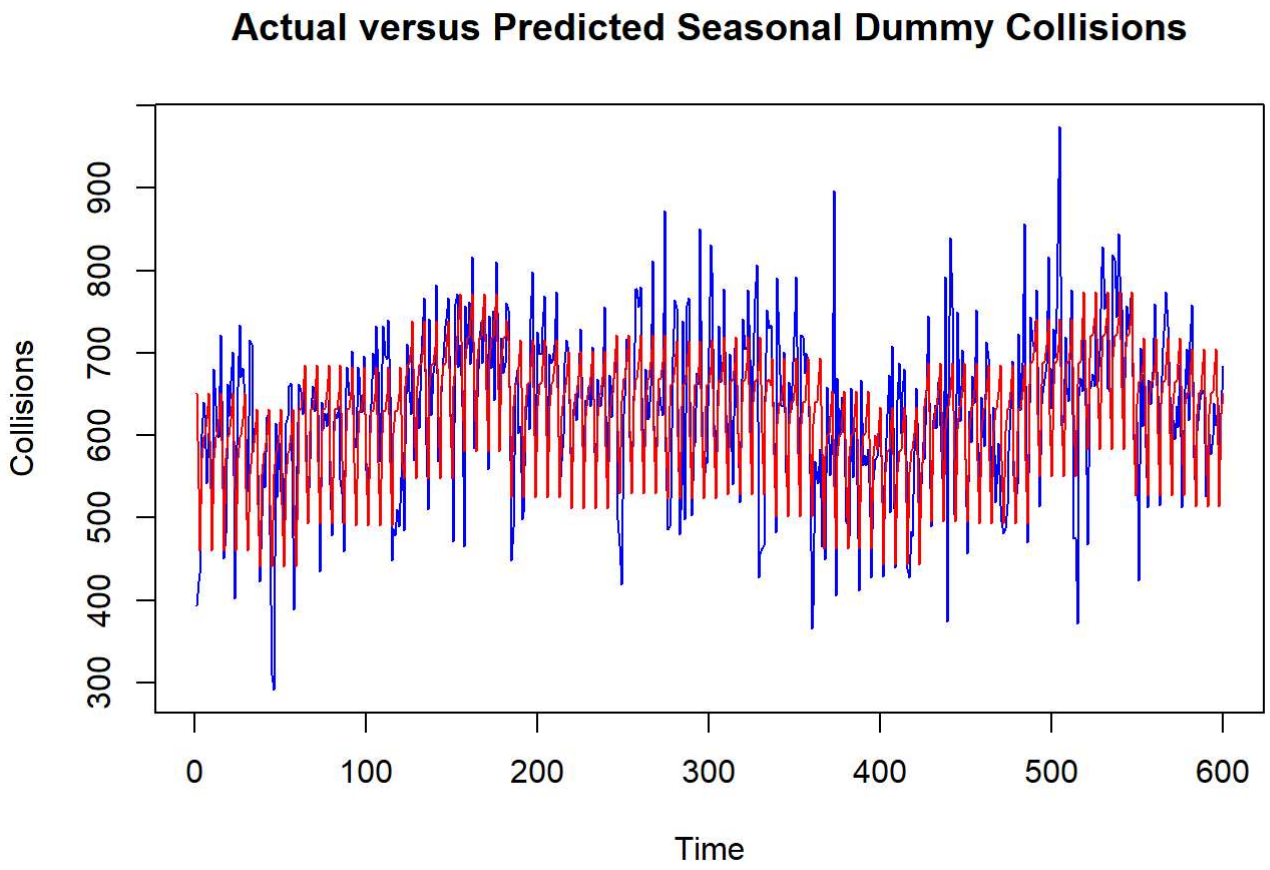


#2.1 Deterministic Time Series Models (Seasonal Dummies)

```
#Seasonal Dummy variables with a trend
time<-seq(1, length(train_crash)) #Create a time variable for the model

#Create Seasonal Dummy variables with a trend
dummy <-lm(train_crash~time+m2_train+m3_train+m4_train+m5_train+m6_train+
         m7_train+m8_train+m9_train+m10_train+m11_train+m12_train+d2_train+d3_train+d4_train+d5_train+d6_train+
         d7_train) #January and Friday are reference month and

#Summarize model
summary(dummy)
```

```
##
## Call:
## lm(formula = train_crash ~ time + m2_train + m3_train + m4_train +
##     m5_train + m6_train + m7_train + m8_train + m9_train + m10_train +
##     m11_train + m12_train + d2_train + d3_train + d4_train +
##     d5_train + d6_train + d7_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -290.69  -34.46    4.11   42.63  358.67
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.515e+02  1.209e+01  53.886  < 2e-16 ***
## time         6.561e-03  1.796e-02   0.365 0.715025
## m2_train    -1.970e+01  1.315e+01  -1.498 0.134548
## m3_train     3.268e+01  1.291e+01   2.531 0.011636 *
## m4_train     3.020e+01  1.307e+01   2.311 0.021187 *
## m5_train     8.667e+01  1.304e+01   6.645 6.97e-11 ***
## m6_train     1.192e+02  1.325e+01   8.994  < 2e-16 ***
## m7_train     6.289e+01  1.327e+01   4.741 2.68e-06 ***
## m8_train     4.966e+01  1.378e+01   3.603 0.000341 ***
## m9_train     6.809e+01  1.597e+01   4.265 2.33e-05 ***
## m10_train    6.148e+01  1.583e+01   3.883 0.000115 ***
## m11_train    6.605e+01  1.608e+01   4.108 4.56e-05 ***
## m12_train    3.937e+01  1.599e+01   2.462 0.014087 *
## d2_train    -1.156e+02  1.092e+01 -10.585  < 2e-16 ***
## d3_train    -1.915e+02  1.092e+01 -17.528  < 2e-16 ***
## d4_train    -7.985e+01  1.093e+01  -7.308 8.99e-13 ***
## d5_train    -5.355e+01  1.093e+01  -4.901 1.24e-06 ***
## d6_train    -5.148e+01  1.096e+01  -4.699 3.27e-06 ***
## d7_train    -3.235e+01  1.095e+01  -2.954 0.003267 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 71.59 on 581 degrees of freedom
## Multiple R-squared:  0.4983, Adjusted R-squared:  0.4827
## F-statistic: 32.05 on 18 and 581 DF,  p-value: < 2.2e-16
```

```
#Plot actual versus predicted of the model
plot.ts(train_crash, main="Actual versus Predicted Seasonal Dummy Collisions", ylab="Collisions",col="blue")
lines(predict(dummy),col="red")
```



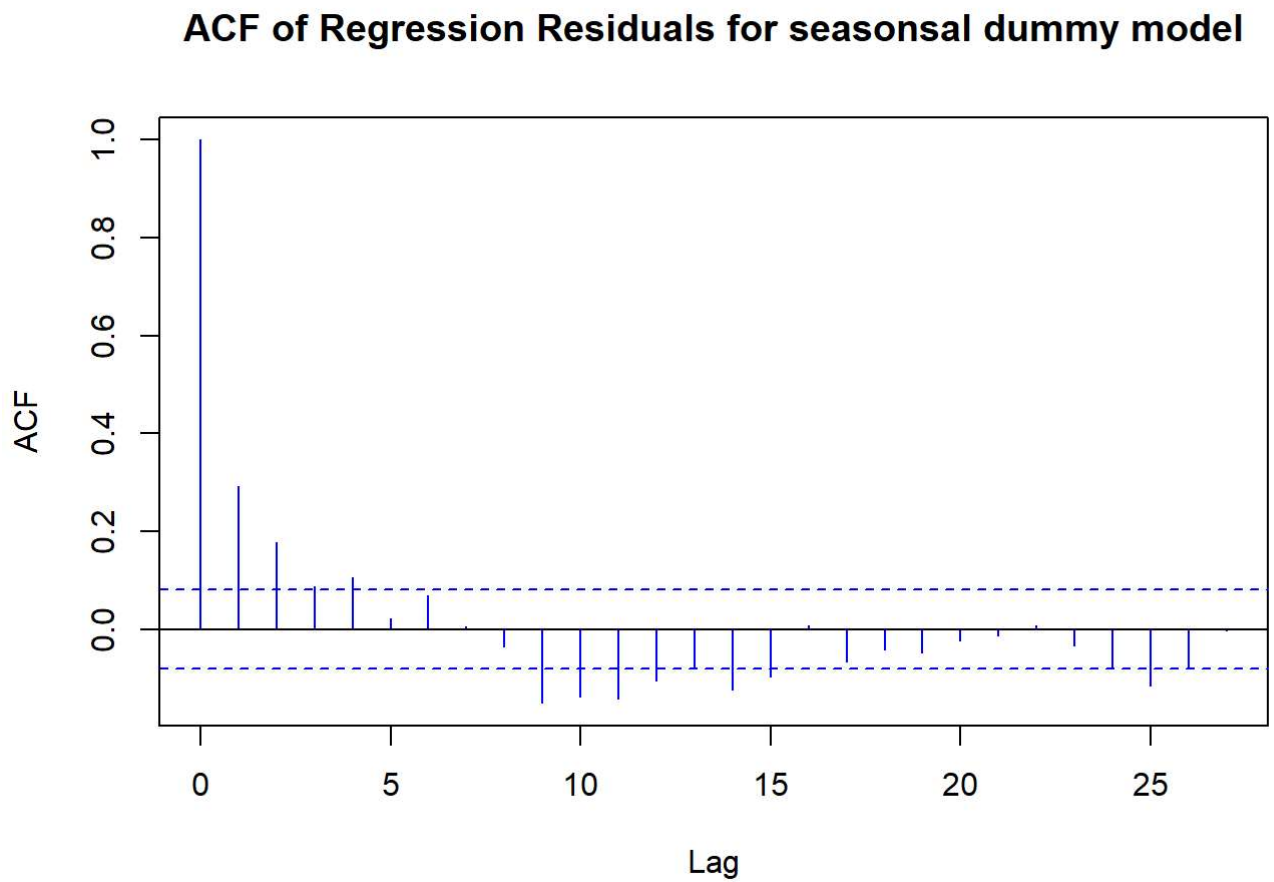
#2.2 Seasonal dummy model in terms of fit (using R-square, MAPE, RMSE and MAE) and hold-out sample (using MAPE, RMSE and MAE).

```
#Evaluation of Seasonal Dummy Variable Model

#Training Data Prediction
pred=predict(dummy, newdata = data.frame(time = time,m2_train = m2_train,m3_train = m3_train, m4_train = m4_train, m5_train=
m5_train,
m6_train= m6_train, m7_train = m7_train, m8_train = m8_train, m9_train = m9_train, m10_train = m10_train, m11_train = m11_tr
ain, m12_train = m12_train, d2_train = d2_train, d3_train = d3_train, d4_train = d4_train, d5_train = d5_train, d6_train = d
6_train, d7_train = d7_train), interval="prediction")

#Calculating Evaluation Metrics for training sample
SD_Train_Mape= mape(train_crash, pred[,1])
SD_Train_Rmse = rmse(train_crash, pred[,1])
SD_Train_Mae = mae(train_crash, pred[,1])
SD_Train_R2 = 1 - sum((train_crash - pred[,1])^2) / sum((train_crash - mean(train_crash))^2)

#Holdout data prediction
time_test <- seq(601, 731)
pred_test <- predict(dummy, newdata = data.frame(time = time_test, m2_train = m2_test,m3_train = m3_test, m4_train = m4_tes
t, m5_train= m5_test, m6_train= m6_test, m7_train = m7_test, m8_train = m8_test, m9_train = m9_test, m10_train = m10_test, m
11_train = m11_test, m12_train = m12_test, d2_train = d2_test, d3_train = d3_test, d4_train = d4_test, d5_train = d5_test, d
6_train = d6_test, d7_train = d7_test), interval = "prediction")

#Calculating Evaluation Metrics for holdout sample
SD_Test_Mape= mape(test_crash, pred_test[,1])
SD_Test_Rmse = rmse(test_crash, pred_test[,1])
SD_Test_Mae = mae(test_crash, pred_test[,1])
SD_Test_R2 = 1 - sum((test_crash - pred_test[,1])^2) / sum((test_crash - mean(test_crash))^2)


#Creating df of metrics for seasonal dummy variable model
SD_Metric_df <- data.frame(Metric = c("MAPE", "RMSE", "MAE", "R2"),
                Training_set = c(SD_Train_Mape, SD_Train_Rmse, SD_Train_Mae, SD_Train_R2),
                Holdout_set = c(SD_Test_Mape, SD_Test_Rmse, SD_Test_Mae, SD_Test_R2))
SD_Metric_df
```

```
##   Metric Training_set Holdout_set
## 1   MAPE   0.08753549  0.09033148
## 2   RMSE  70.44960663 77.80325642
## 3    MAE  51.44189077 56.85556672
## 4     R2   0.49825944  0.38915910
```

#2.3 Seasonal Dummy variable residuals

```
#ACF of residuals for Dummy Variable Model
res_dummy=residuals(dummy) #Residuals of seasonal dummy var model
acf(res_dummy,main="ACF of Regression Residuals for seasonsal dummy model",col="blue") #ACF of SDM residuals
```



```
Box.test(res_dummy, lag = 20) #ask what lag to use for this data
```

```
##
##  Box-Pierce test
##
## data:  res_dummy
## X-squared = 153.53, df = 20, p-value < 2.2e-16
```

```
#The acf of residuals does not appear to be white noise, as the p-val is so small and we can see multiple lags surpassing th
e 2se bounds
```

#2.1 Deterministic Time Series Models (Cyclical Trend)

```
#Creating Periodogram
library(TSA)
```

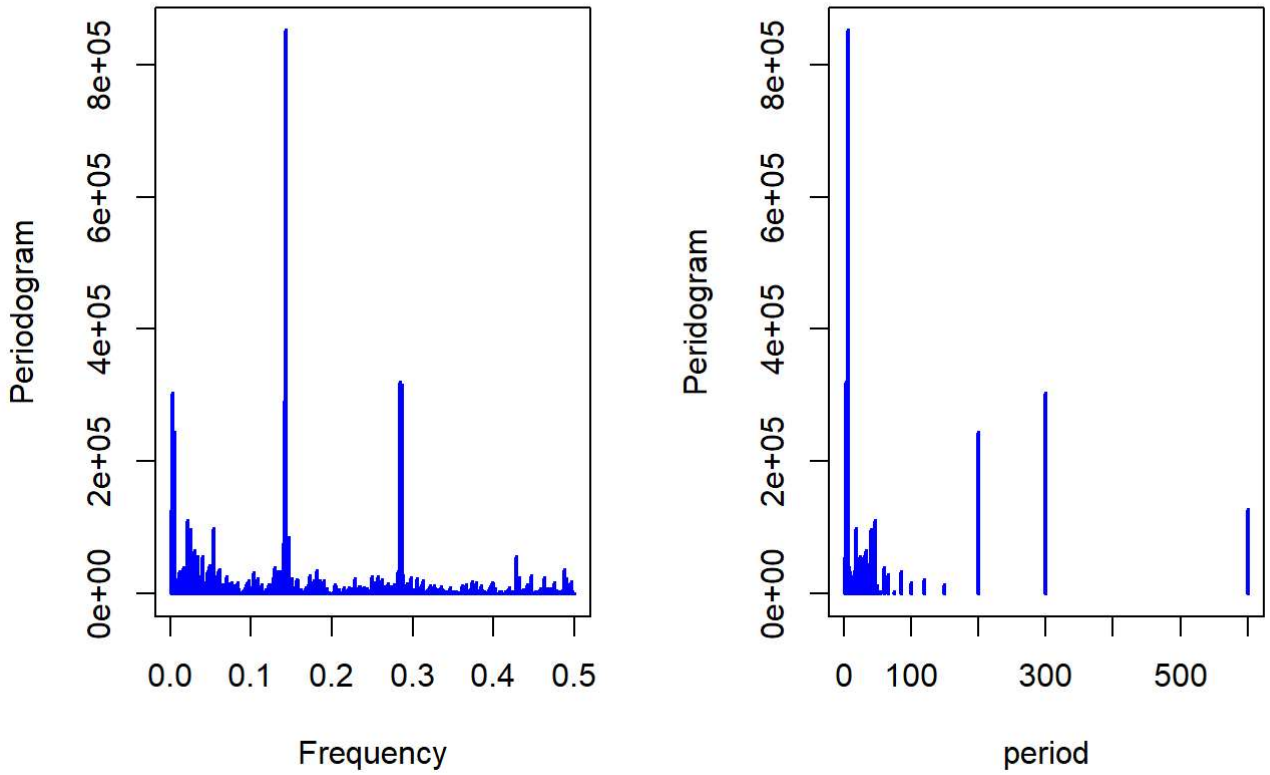
```
## Warning: package 'TSA' was built under R version 4.4.3
```

```
##
## Attaching package: 'TSA'
```

```
## The following objects are masked from 'package:stats':
##
##     acf, arima
```

```
## The following object is masked from 'package:utils':
##
##     tar
```

```
#Creating the trend component
time=seq(1,length(train_crash))
#removing trend
detrend<-lm(train_crash~time)

#Creating Periodogram
prdgrm=periodogram(detrend$residuals,col="blue")
period=1/prdgrm$freq

#Plotting Periodogram
par(mfrow=c(1,2))
periodogram(detrend$residuals,col="blue")
plot(period,prdgrm$spec, type="h",col="blue",ylab="Peridogram",lwd=2)
```




```
#Define frequency and amplitude from the periodogram output
frequency=prdgrm$freq
amplitude=prdgrm$spec

#Periodogram DF to see which periods have the highest amplitudes
periodogram_df = cbind(period,frequency, amplitude) #Create periodogram
periodogram_df = data.frame(periodogram_df) #Make periodogram into a dataframe so it can be properly sorted
periodogram_df = periodogram_df[order(periodogram_df$amplitude, decreasing = TRUE),] #Now sort the periodgram by decreasing amplitude
head(periodogram_df, 15) # Top 15 amplitudes
```

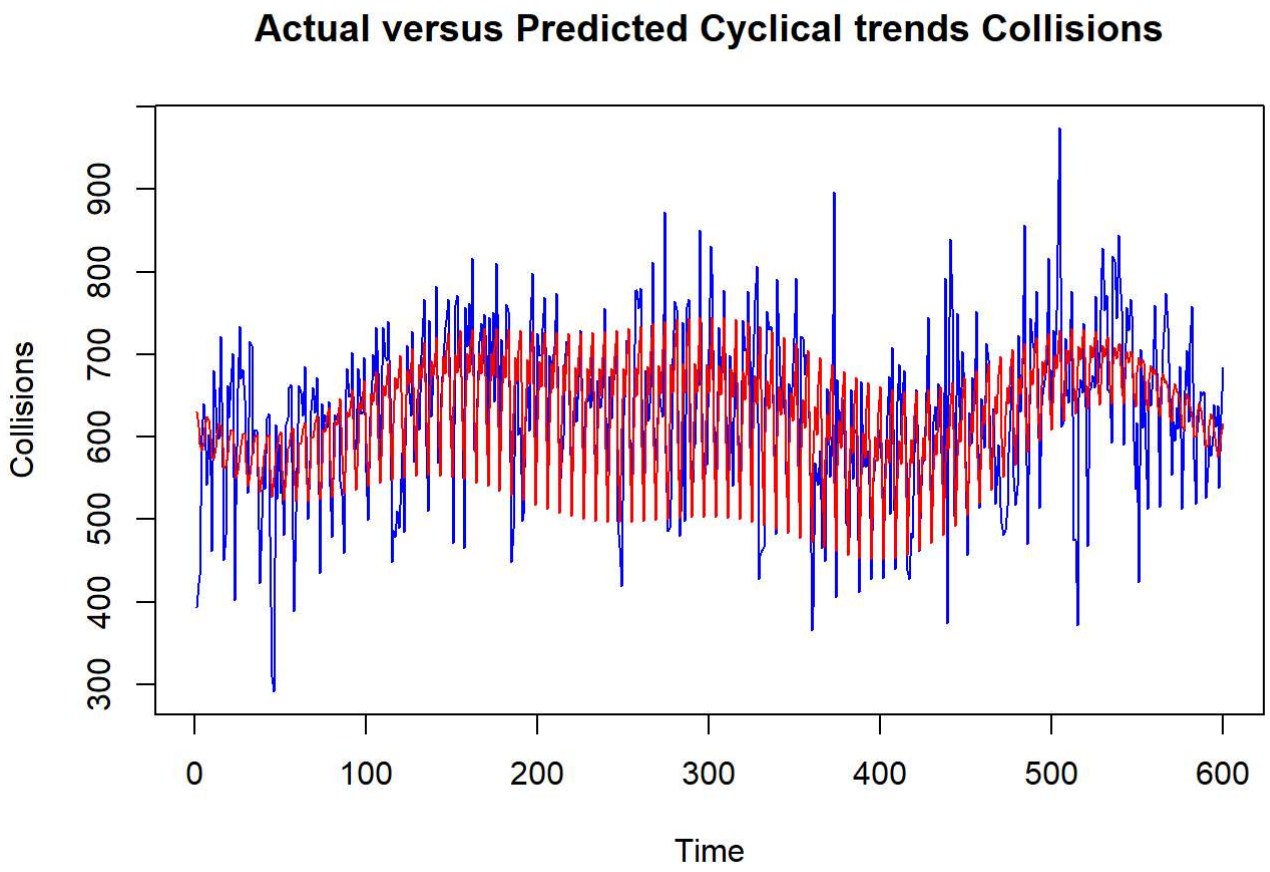
```
##         period   frequency amplitude
## 86    6.976744 0.143333333 852150.75
## 171   3.508772 0.285000000 320903.19
## 172   3.488372 0.286666667 315208.30
## 2   300.000000 0.003333333 303350.86
## 85    7.058824 0.141666667 292366.08
## 3   200.000000 0.005000000 245108.83
## 1   600.000000 0.001666667 128294.86
## 13   46.153846 0.021666667 111376.16
## 32   18.750000 0.053333333  98255.81
## 15   40.000000 0.025000000  96639.39
## 88    6.818182 0.146666667  84771.10
## 84    7.142857 0.140000000  77515.09
## 18   33.333333 0.030000000  65993.34
## 257   2.334630 0.428333333  56895.14
## 24   25.000000 0.040000000  56644.11
```

#After sorting by amplitude, we can observe that the top two periods are 7 and 3.5, while the period 2.33 is 14th.

```
#Define time (trend component) and n
time=seq(1,length(train_crash)) #This time variable when used in the model is insignificant, with a p-val of 0.11, so drop f
rom model
n=length(train_crash)

#Creating the cos and sin pairs for top 6 highest amplitudes
cos1=cos(2*pi*(86/n)*time) #For period 6.976
sin1=sin(2*pi*(86/n)*time)

cos2=cos(2*pi*(171/n)*time) #For period 3.508
sin2=sin(2*pi*(171/n)*time)

cos3=cos(2*pi*(172/n)*time) #For period 3.488
sin3=sin(2*pi*(172/n)*time)

cos4=cos(2*pi*(2/n)*time) #For period 300
sin4=sin(2*pi*(2/n)*time)

cos5=cos(2*pi*(85/n)*time) #for period 7.058
sin5=sin(2*pi*(85/n)*time)

cos6=cos(2*pi*(3/n)*time) #For period 200
sin6=sin(2*pi*(3/n)*time)

#Define cyclical trend model
CM <-lm(train_crash~time+cos1+sin1+cos2+sin2+cos3+sin3+cos4+sin4+cos5+sin5+cos6+sin6)
summary(CM) #Based on the model output, periods 7 and 3.5 are significant, while period 2.33 is insignificant
```

```
##
## Call:
## lm(formula = train_crash ~ time + cos1 + sin1 + cos2 + sin2 +
##     cos3 + sin3 + cos4 + sin4 + cos5 + sin5 + cos6 + sin6)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -313.79  -40.99    8.16   47.15  353.16
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 636.22911    6.88144  92.456  < 2e-16 ***
## time         -0.03186    0.02041  -1.561 0.119008
## cos1         43.92816    4.41546   9.949  < 2e-16 ***
## sin1         30.12364    4.41561   6.822 2.24e-11 ***
## cos2         29.46599    4.41546   6.673 5.80e-11 ***
## sin2         14.30704    4.41544   3.240 0.001262 **
## cos3        -28.53391    4.41546  -6.462 2.17e-10 ***
## sin3        -15.28111    4.41544  -3.461 0.000578 ***
## cos4         -2.73078    4.41546  -0.618 0.536513
## sin4        -40.35086    4.82630  -8.361 4.56e-16 ***
## cos5        -22.67169    4.41546  -5.135 3.85e-07 ***
## sin5        -21.55449    4.41562  -4.881 1.36e-06 ***
## cos6        -20.69855    4.41546  -4.688 3.44e-06 ***
## sin6        -25.40174    4.60254  -5.519 5.12e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 76.48 on 586 degrees of freedom
## Multiple R-squared:  0.4225, Adjusted R-squared:  0.4097
## F-statistic: 32.98 on 13 and 586 DF,  p-value: < 2.2e-16
```

```
#TS plot for Predicted vs Actual
plot.ts(train_crash, main="Actual versus Predicted Cyclical trends Collisions", ylab="Collisions",col="blue")
lines(predict(CM),col="red")
```



#2.2 Seasonal dummy model in terms of fit (using R-square, MAPE, RMSE and MAE) and hold-out sample (using MAPE, RMSE and MAE).

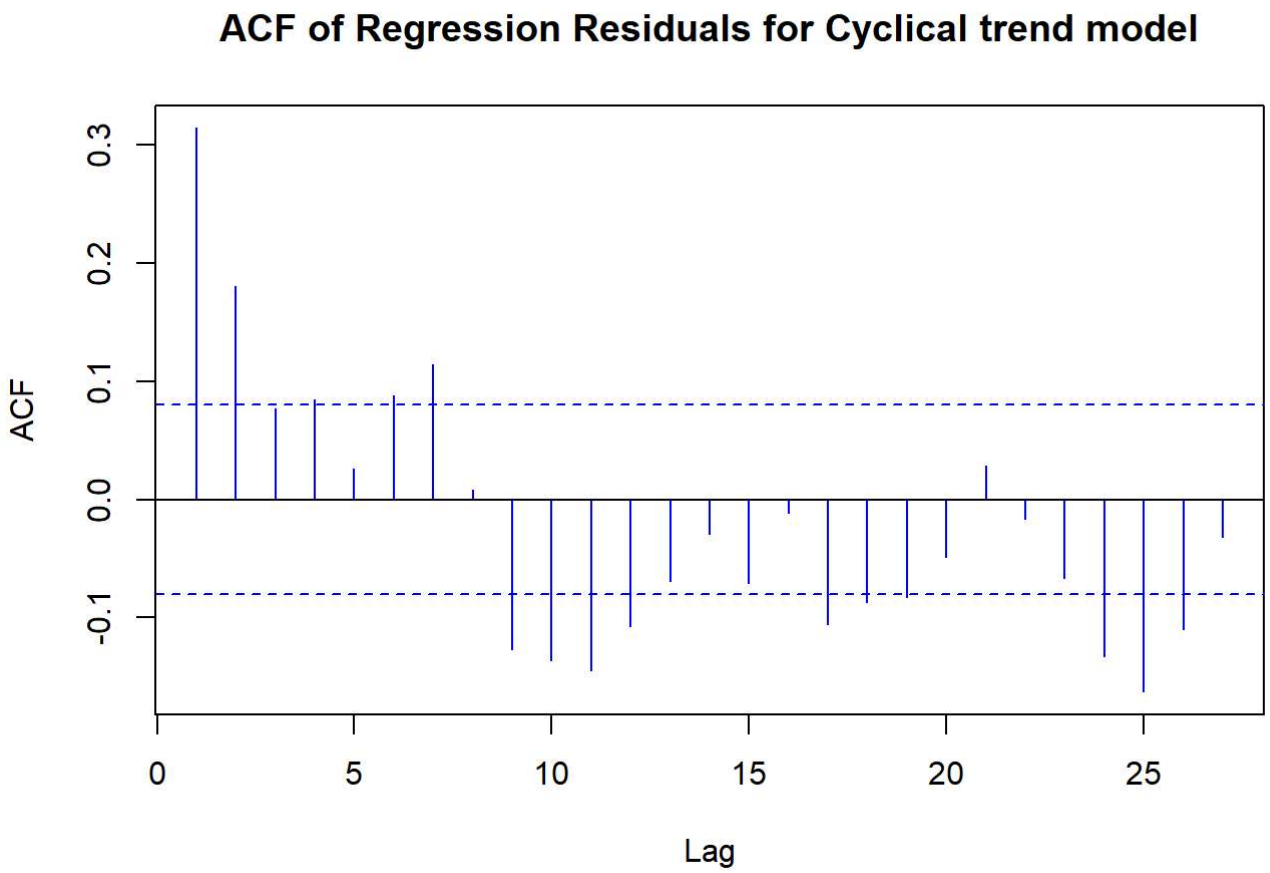
```
#Evaluation of Cyclical trend model

#Define Cos and Sin pairs and time for holdout sample
time_test = c(601:731)

cos1_test=cos(2*pi*(86/n)*time_test)
sin1_test=sin(2*pi*(86/n)*time_test)

cos2_test=cos(2*pi*(171/n)*time_test)
sin2_test=sin(2*pi*(171/n)*time_test)

cos3_test=cos(2*pi*(172/n)*time_test)
sin3_test=sin(2*pi*(172/n)*time_test)

cos4_test=cos(2*pi*(2/n)*time_test)
sin4_test=sin(2*pi*(2/n)*time_test)

cos5_test=cos(2*pi*(85/n)*time_test)
sin5_test=sin(2*pi*(85/n)*time_test)

cos6_test=cos(2*pi*(3/n)*time_test)
sin6_test=sin(2*pi*(3/n)*time_test)

#Training Sample Prediction
pred_train = predict(CM, data.frame(time=time,cos1=cos1,sin1=sin1, cos2=cos2,sin2=sin2, cos3 = cos3, sin3 =sin3, cos4=cos4,s
in4=sin4, cos5=cos5,sin5=sin5, cos6 = cos6, sin6 =sin6))

#Training Sample Metrics
CM_train_mape = mape(train_crash,pred_train)
CM_train_rmse = rmse(train_crash,pred_train)
CM_train_mae = mae(train_crash,pred_train)
CM_Train_R2 = 1 - sum((train_crash - pred_train)^2) / sum((train_crash - mean(train_crash))^2)


#Holdout sample prediction
pred_test=predict(CM,data.frame(time=time_test,cos1=cos1_test,sin1=sin1_test, cos2=cos2_test,sin2=sin2_test, cos3 = cos3_tes
t, sin3 =sin3_test, cos4=cos4_test,sin4=sin4_test, cos5=cos5_test,sin5=sin5_test, cos6 = cos6_test, sin6 =sin6_test))

#Holdout Sample Metrics
```

```
CM_test_mape = mape(test_crash,pred_test)
CM_test_rmse = rmse(test_crash,pred_test)
CM_test_mae = mae(test_crash,pred_test)
CM_Test_R2 = 1 - sum((test_crash - pred_test)^2) / sum((test_crash - mean(test_crash))^2)

#Creating dataframe for cyclical trend model
CM_Metric_df <- data.frame(Metric = c("MAPE", "RMSE", "MAE", "R2"),
                Training_set = c(CM_train_mape, CM_train_rmse, CM_train_mae, CM_Train_R2),
                Holdout_set = c(CM_test_mape, CM_test_rmse, CM_test_mae, CM_Test_R2))
CM_Metric_df
```

```
##   Metric Training_set Holdout_set
## 1   MAPE   0.09673193   0.1571088
## 2   RMSE  75.57964381 133.6968750
## 3    MAE  56.54141130 104.1497154
## 4     R2   0.42252691  -0.8037447
```

#2.3 Cyclical trend residuals

```
#ACF of residuals of seasonal for Cyclical trend model
res_CM=residuals(CM)
acf(res_CM,main="ACF of Regression Residuals for Cyclical trend model",col="blue")
```

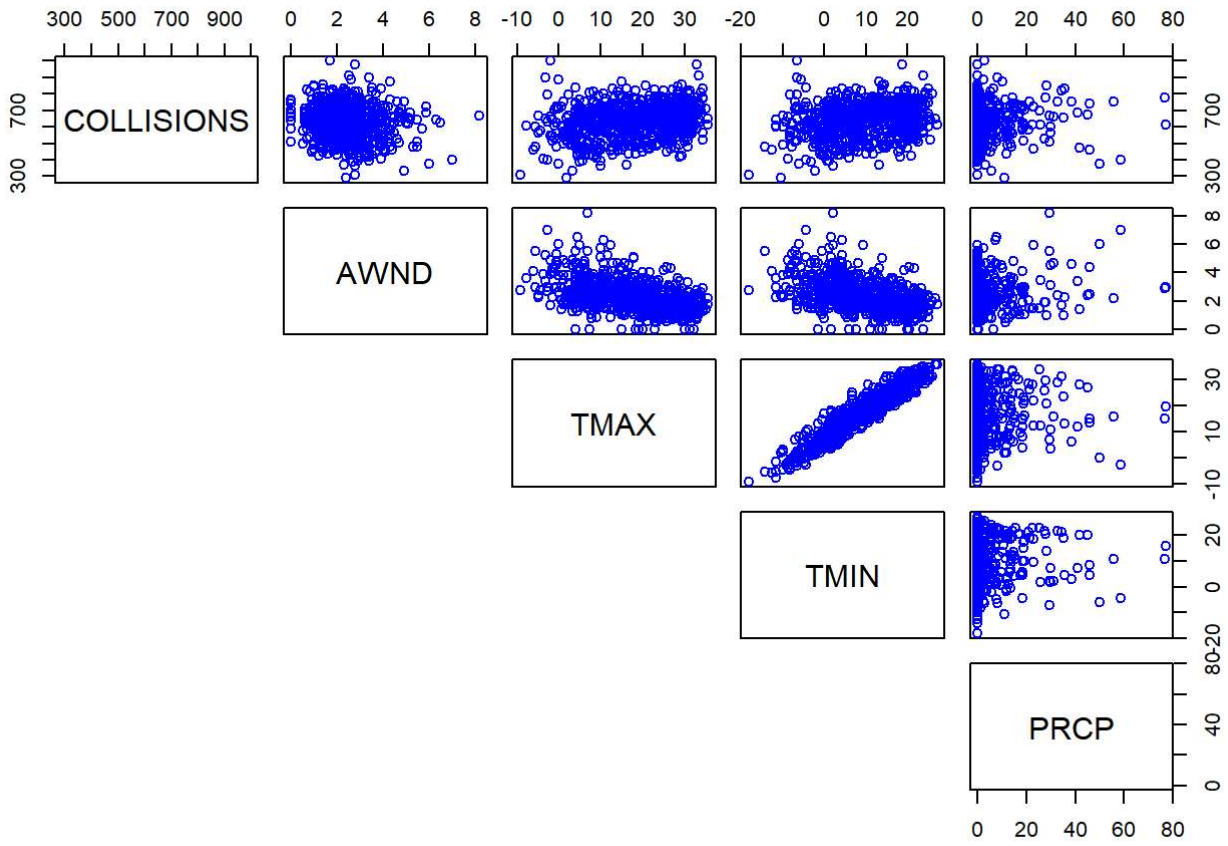



```
Box.test(res_CM, lag = 20)
```

```
##
##  Box-Pierce test
##
## data:  res_CM
## X-squared = 162.52, df = 20, p-value < 2.2e-16
```

#3.1 Discussion of independent variables. Correlation analysis and scatter plots.

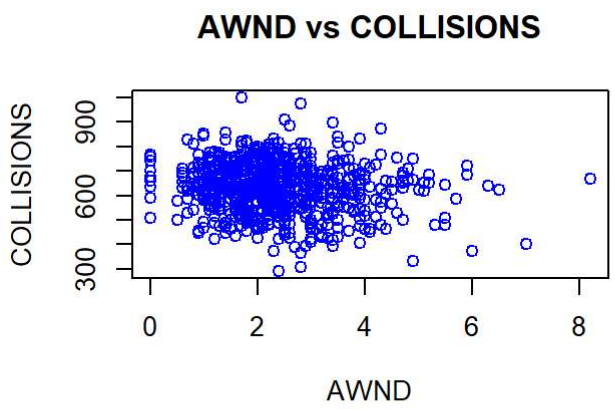
```
#Create dataframe from the TS object for the correlation analysis
reg_data = data.frame(crash)

#Clean NA values present
reg_data[is.na(reg_data)] = 0 #Assigns a value of 0 to rows with an NA (wind speed col)

#Create Plot for Correlation
pairs(reg_data[,2:6],lower.panel = NULL,col="blue")
```

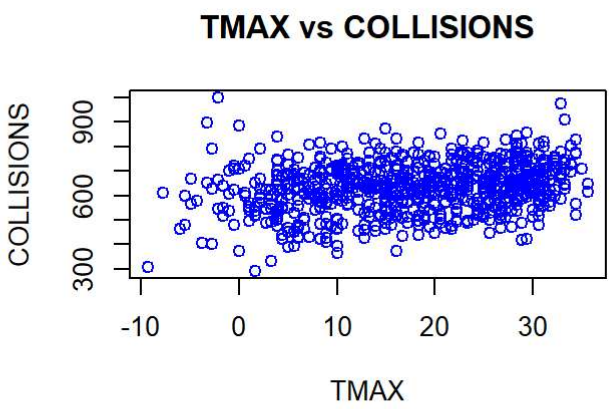



```
#Create Correlation Matrix for variables
cor(reg_data[,2:6])
```

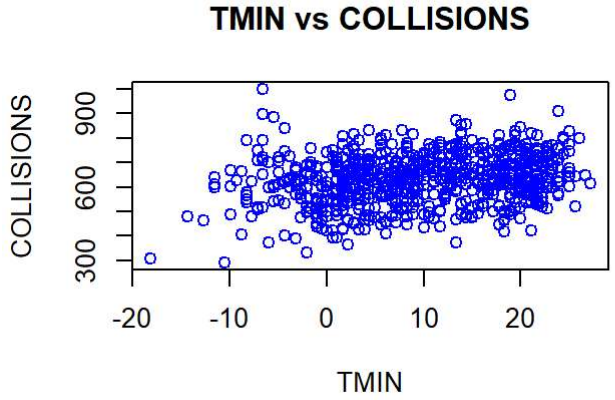
```
##             COLLISIONS       AWND        TMAX        TMIN        PRCP
## COLLISIONS  1.00000000 -0.1299010  0.25712158  0.22875593  0.02533619
## AWND       -0.12990100  1.0000000 -0.50534141 -0.48674360  0.18480279
## TMAX        0.25712158 -0.5053414  1.00000000  0.95597105 -0.03633706
## TMIN        0.22875593 -0.4867436  0.95597105  1.00000000  0.02147704
## PRCP        0.02533619  0.1848028 -0.03633706  0.02147704  1.00000000
```

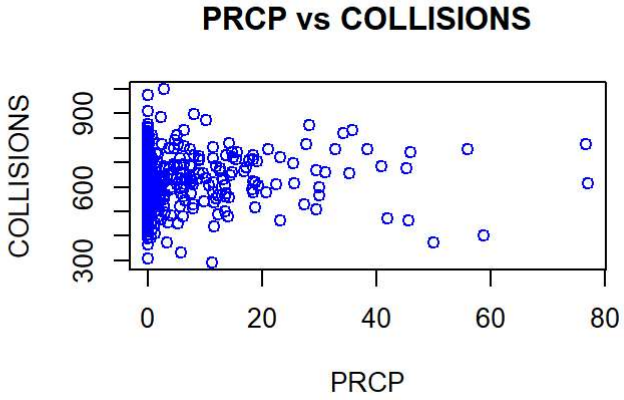
```
# Set up 2x2 plotting space
par(mfrow = c(2, 2))

# Scatter plots vs COLLISIONS for better clarity
plot(reg_data$AWND, reg_data$COLLISIONS, main = "AWND vs COLLISIONS", xlab = "AWND", ylab = "COLLISIONS", col = "blue", pch
= 1)
plot(reg_data$TMAX, reg_data$COLLISIONS, main = "TMAX vs COLLISIONS", xlab = "TMAX", ylab = "COLLISIONS", col = "blue", pch
= 1)
plot(reg_data$TMIN, reg_data$COLLISIONS, main = "TMIN vs COLLISIONS", xlab = "TMIN", ylab = "COLLISIONS", col = "blue", pch
= 1)
plot(reg_data$PRCP, reg_data$COLLISIONS, main = "PRCP vs COLLISIONS", xlab = "PRCP", ylab = "COLLISIONS", col = "blue", pch
= 1)
```





#3.2 Comparison of “candidate”



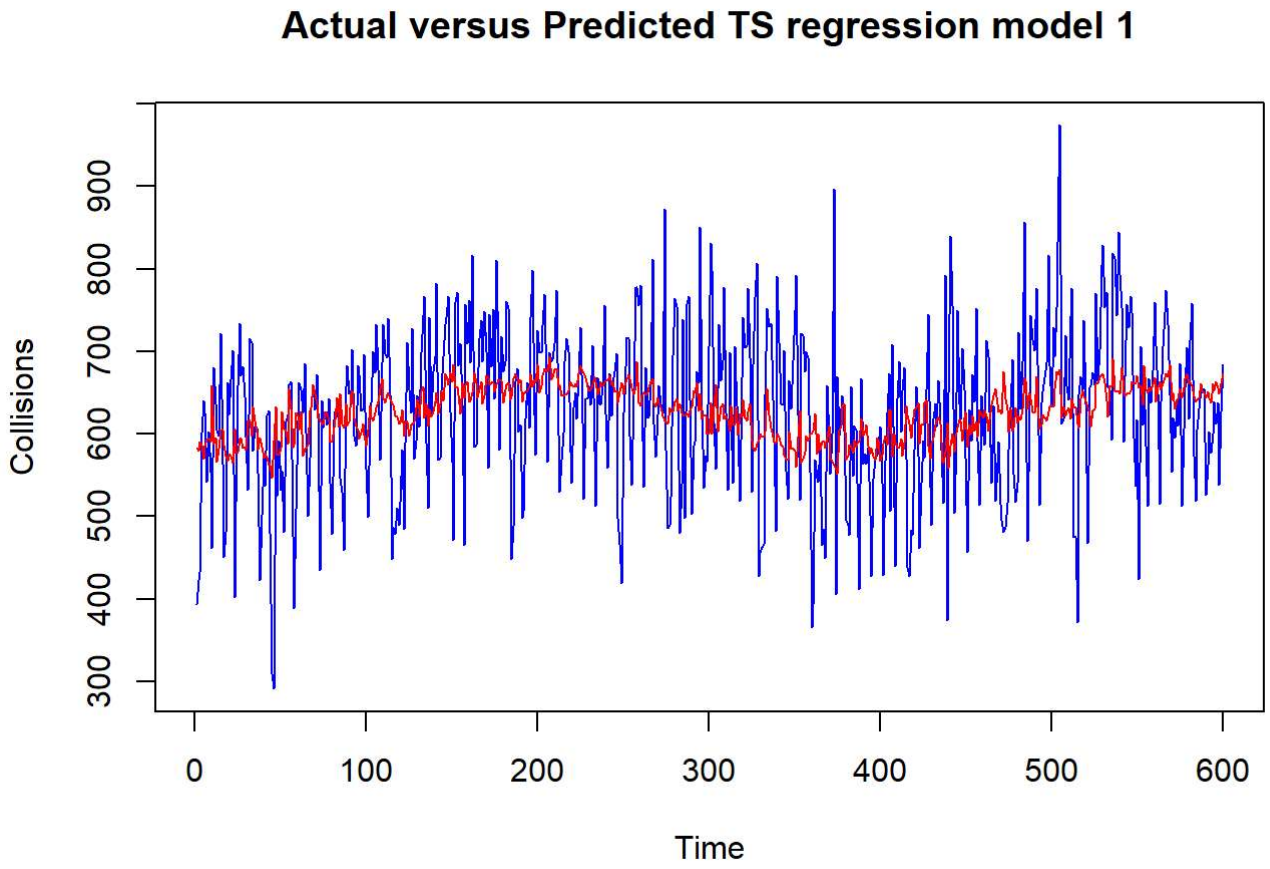


models based on variables: AWND, TMAX, TMIN, PRCP

```
#Creating independent variables for training sample and holdout sample:

#Dependent variable
train_crash = crash[,"COLLISIONS"][1:600] #600 observations for the training data
test_crash = crash[,"COLLISIONS"][601:731] #131 observations for the holdout (test) data

#Dealing with NA values in AWND column
train_AWND = crash[,"AWND"][1:600] #Average wind speed variable
train_AWND[is.na(train_AWND)] = 0 #Impute NA values with 0
test_AWND = crash[,"AWND"][601:731]
test_AWND[is.na(test_AWND)] = 0 #Impute NA values with 0

#Remaining Independent Variables
train_TMAX = crash[,"TMAX"][1:600] #Max temperature of the day
test_TMAX = crash[,"TMAX"][601:731]

train_TMIN = crash[,"TMIN"][1:600] #Min temperature of the day
test_TMIN = crash[,"TMIN"][601:731]

train_PRCP = crash[,"PRCP"][1:600] #precipitation in (inches or centimeters?) for the day
test_PRCP = crash[,"PRCP"][601:731]

#Creating and summarizing model
REG_1<-lm(train_crash~train_AWND+train_TMAX+train_TMIN+train_PRCP)
summary(REG_1)
```

```
##
## Call:
## lm(formula = train_crash ~ train_AWND + train_TMAX + train_TMIN +
##     train_PRCP)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -300.89  -59.91    3.67   60.62  334.84
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 553.6539    17.8243  31.062  < 2e-16 ***
## train_AWND    1.7568     4.0877   0.430 0.667517
## train_TMAX    4.6096     1.3400   3.440 0.000623 ***
## train_TMIN   -1.6664     1.4471  -1.152 0.249941
## train_PRCP    0.7753     0.4649   1.668 0.095910 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 94.84 on 595 degrees of freedom
## Multiple R-squared:  0.0983, Adjusted R-squared:  0.09224
## F-statistic: 16.22 on 4 and 595 DF,  p-value: 1.292e-12
```

```
#Actual vs Predicted values for TS plot
plot.ts(train_crash, main="Actual versus Predicted TS regression model 1", ylab="Collisions",col="blue")
lines(predict(REG_1),col="red")
```



#3.2 Comparison of REG1 model in terms of fit (using R-square, MAPE, RMSE and MAE) and hold-out sample (using MAPE, RMSE and MAE).

```
#Training Data Prediction
train_pred=predict(REG_1, data.frame(train_AWND, train_TMAX, train_TMIN, train_PRCP), interval="prediction")

#Calculating Training Data Metrics
REG_1_Train_Mape = mape(train_crash, train_pred[,1])
REG_1_Train_Rmse = rmse(train_crash, train_pred[,1])
REG_1_Train_Mae = mae(train_crash, train_pred[,1])
REG_1_Train_R2 = 1 - sum((train_crash - train_pred[,1])^2) / sum((train_crash - mean(train_crash))^2)

#Holdout data Prediction:
test_pred=predict(REG_1, newdata = data.frame(train_AWND = test_AWND, train_TMAX = test_TMAX, train_TMIN = test_TMIN, train_
PRCP = test_PRCP),interval="prediction")

#Calculating Holdout Data Metrics
REG_1_Test_Mape =mape(test_crash, test_pred[,1])
REG_1_Test_Rmse = rmse(test_crash, test_pred[,1])
REG_1_Test_Mae = mae(test_crash, test_pred[,1])
REG_1_Test_R2 = 1 - sum((test_crash - test_pred[,1])^2) / sum((test_crash - mean(test_crash))^2)

#Creating dataframe for Regression model
Reg1_Metric_df <- data.frame(Metric = c("MAPE", "RMSE", "MAE", "R2"),
                Training_set = c(REG_1_Train_Mape, REG_1_Train_Rmse, REG_1_Train_Mae, REG_1_Train_R2),
                Holdout_set = c(REG_1_Test_Mape, REG_1_Test_Rmse, REG_1_Test_Mae, REG_1_Test_R2))
Reg1_Metric_df
```

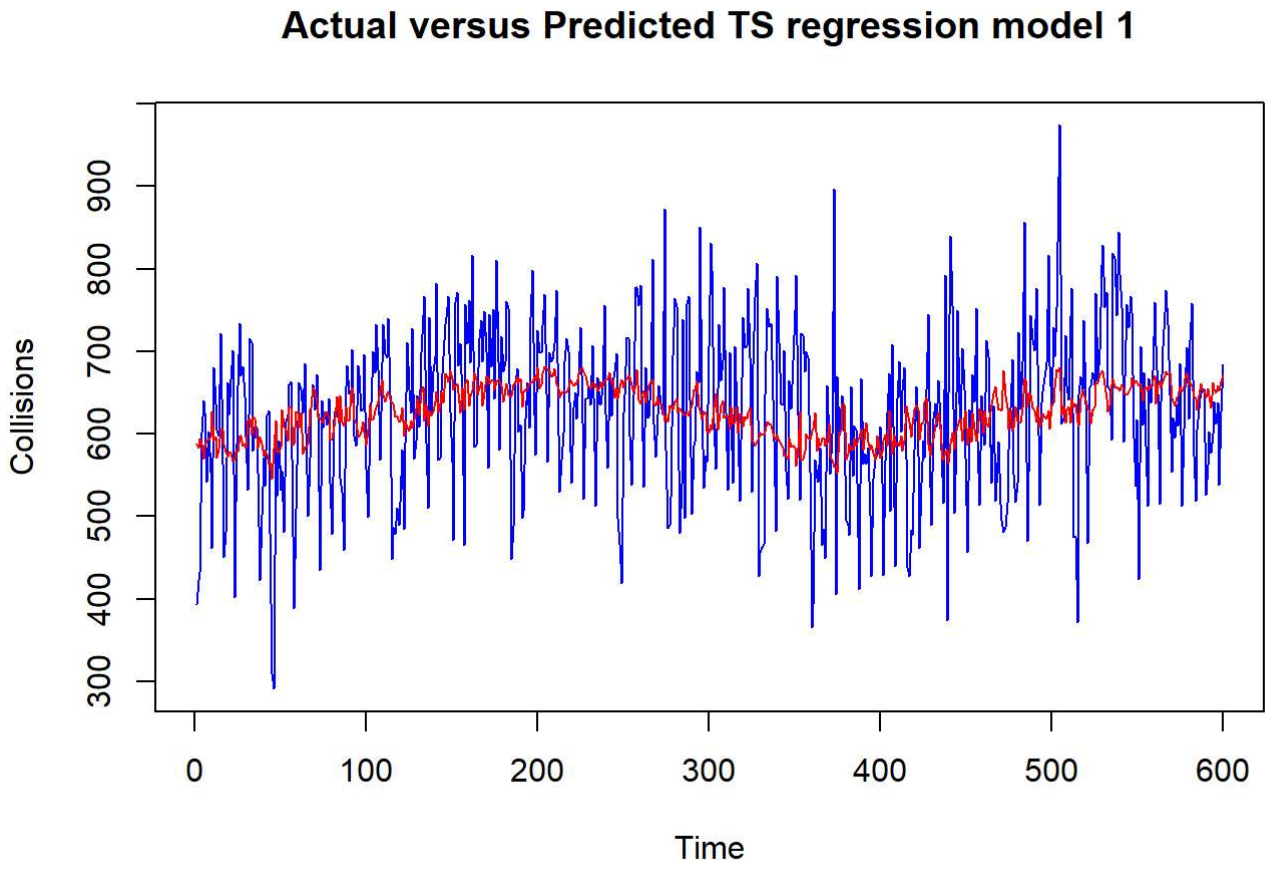
```
##   Metric Training_set Holdout_set
## 1   MAPE   0.12673983   0.1297729
## 2   RMSE  94.44290944 105.1734734
## 3    MAE  74.27738352  83.6655433
## 4     R2   0.09830267  -0.1162076
```

#3.2 Comparison of “candidate” models based on variables: AWND, TMAX, TMIN

```
#Creating and summarizing model without prcp column now
REG_2<-lm(train_crash~train_AWND+train_TMAX+train_TMIN)
summary(REG_2)
```

```
##
## Call:
## lm(formula = train_crash ~ train_AWND + train_TMAX + train_TMIN)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -291.08  -58.55    3.00   60.68  337.29
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  554.907     17.835  31.113  < 2e-16 ***
## train_AWND     3.088      4.015   0.769  0.44220
## train_TMAX     4.244      1.324   3.205  0.00142 **
## train_TMIN    -1.195      1.421  -0.841  0.40088
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 94.98 on 596 degrees of freedom
## Multiple R-squared:  0.09409,    Adjusted R-squared:  0.08953
## F-statistic: 20.63 on 3 and 596 DF,  p-value: 9.893e-13
```

```
#Actual vs Predicted values for TS plot
plot.ts(train_crash, main="Actual versus Predicted TS regression model 1", ylab="Collisions",col="blue")
lines(predict(REG_2),col="red")
```



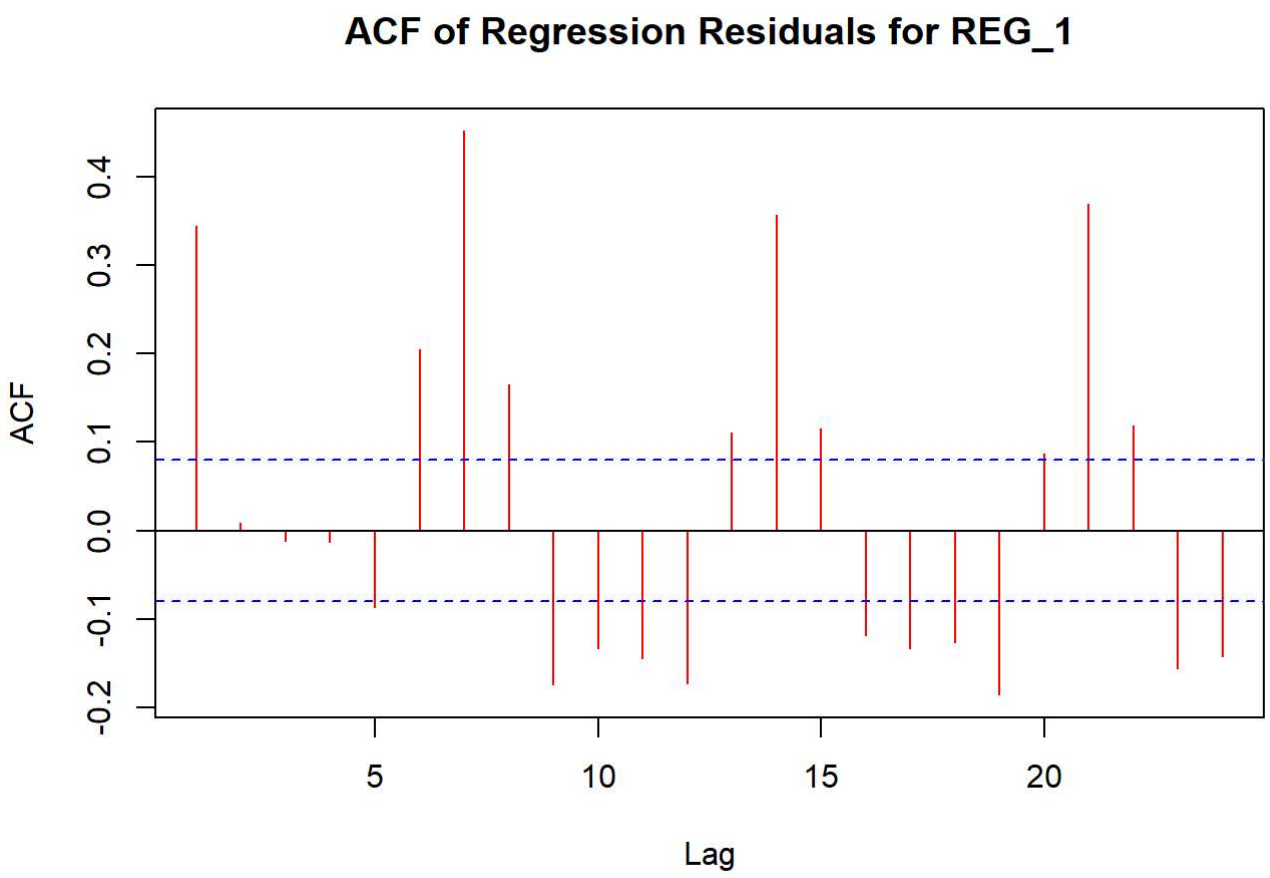
#3.2 Comparison of REG2 (using R-square, MAPE, RMSE and MAE) and hold-out sample (using MAPE, RMSE and MAE).

```
#Training Data Prediction
train_pred=predict(REG_2, data.frame(train_AWND, train_TMAX, train_TMIN, train_PRCP), interval="prediction")

#Calculating Training Data Metrics
REG_2_Train_Mape = mape(train_crash, train_pred[,1])
REG_2_Train_Rmse = rmse(train_crash, train_pred[,1])
REG_2_Train_Mae = mae(train_crash, train_pred[,1])
REG_2_Train_R2 = 1 - sum((train_crash - train_pred[,1])^2) / sum((train_crash - mean(train_crash))^2)

#Holdout data Prediction:
test_pred=predict(REG_2, newdata = data.frame(train_AWND = test_AWND, train_TMAX = test_TMAX, train_TMIN = test_TMIN, train_
PRCP = test_PRCP),interval="prediction")

#Calculating Holdout Data Metrics
REG_2_Test_Mape =mape(test_crash, test_pred[,1])
REG_2_Test_Rmse = rmse(test_crash, test_pred[,1])
REG_2_Test_Mae = mae(test_crash, test_pred[,1])
REG_2_Test_R2 = 1 - sum((test_crash - test_pred[,1])^2) / sum((test_crash - mean(test_crash))^2)

#Creating dataframe for Regression model
Reg2_Metric_df <- data.frame(Metric = c("MAPE", "RMSE", "MAE", "R2"),
                Training_set = c(REG_2_Train_Mape, REG_2_Train_Rmse, REG_2_Train_Mae, REG_2_Train_R2),
                Holdout_set = c(REG_2_Test_Mape, REG_2_Test_Rmse, REG_2_Test_Mae, REG_2_Test_R2))
Reg2_Metric_df
```

```
##   Metric Training_set Holdout_set
## 1   MAPE   0.12710366   0.1288649
## 2   RMSE  94.66336880 104.7523530
## 3    MAE  74.58783389  82.9645013
## 4     R2   0.09408807  -0.1072868
```

#3.3 Residuals/is best model stationary

```
#ACF of regression residuals
res=residuals(REG_1)
acf(res,main="ACF of Regression Residuals for REG_1",col="red",lag=24)
```

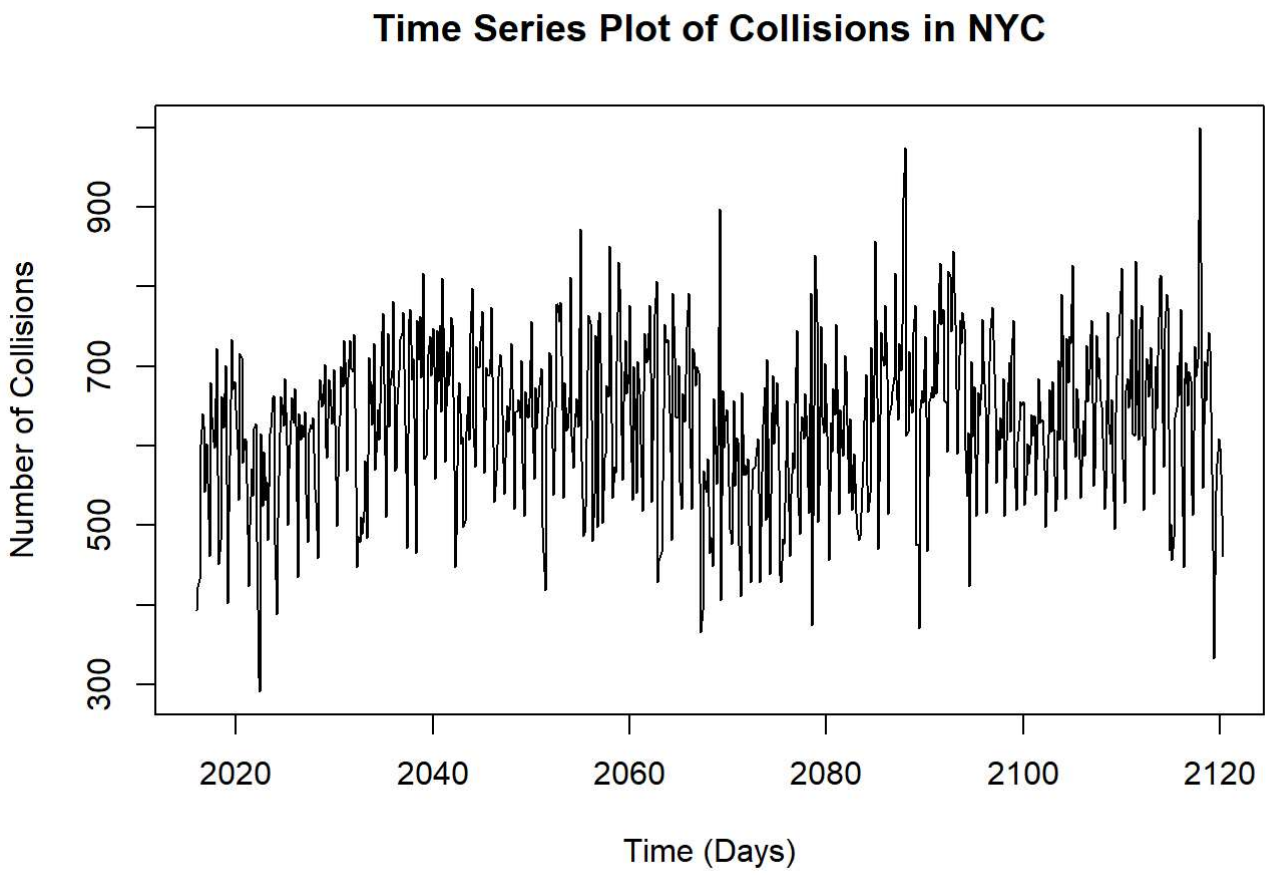



```
Box.test(res, lag=20)
```

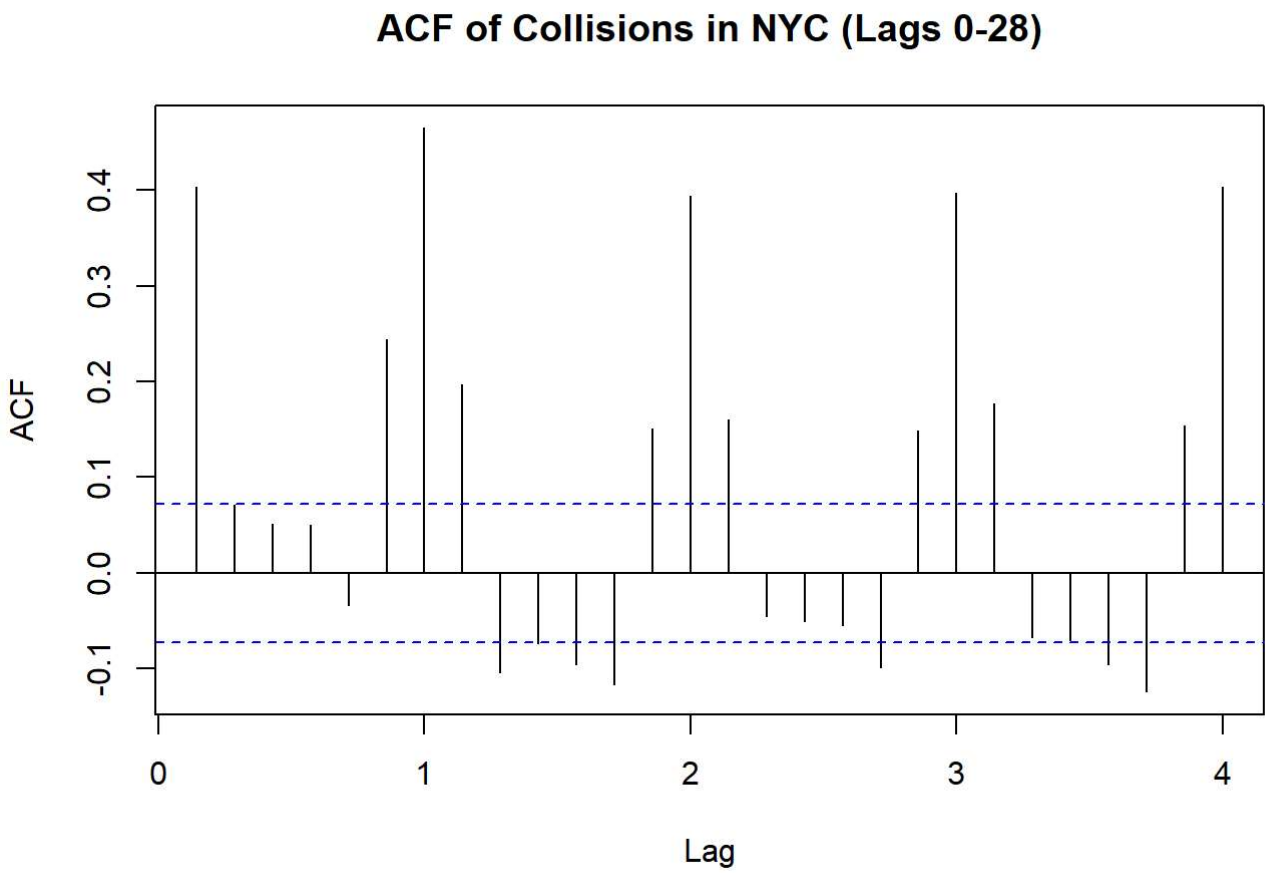
```
##
##  Box-Pierce test
##
## data:  res
## X-squared = 443.81, df = 20, p-value < 2.2e-16
```

SECTION 4.1) You will be using your target (dependent) variable and develop an ARIMA model to describe it. For this you need to transform the target variable time series to a stationary series by differencing it and then identify an AR, MA or ARMA models. If you have seasonal characteristics in the data you may need to use seasonal differencing as discussed in the Johnson and Johnson data example discussed in class (Week 12). Potentially, you may need to include seasonal AR and MA components in the model discussed in Week 13.

```
#Time Series Plot of variable of interest (Collisions)
ts.plot(crash[,"COLLISIONS"], main = "Time Series Plot of Collisions in NYC", xlab = "Time (Days)", ylab = "Number of Collis
ions")
```

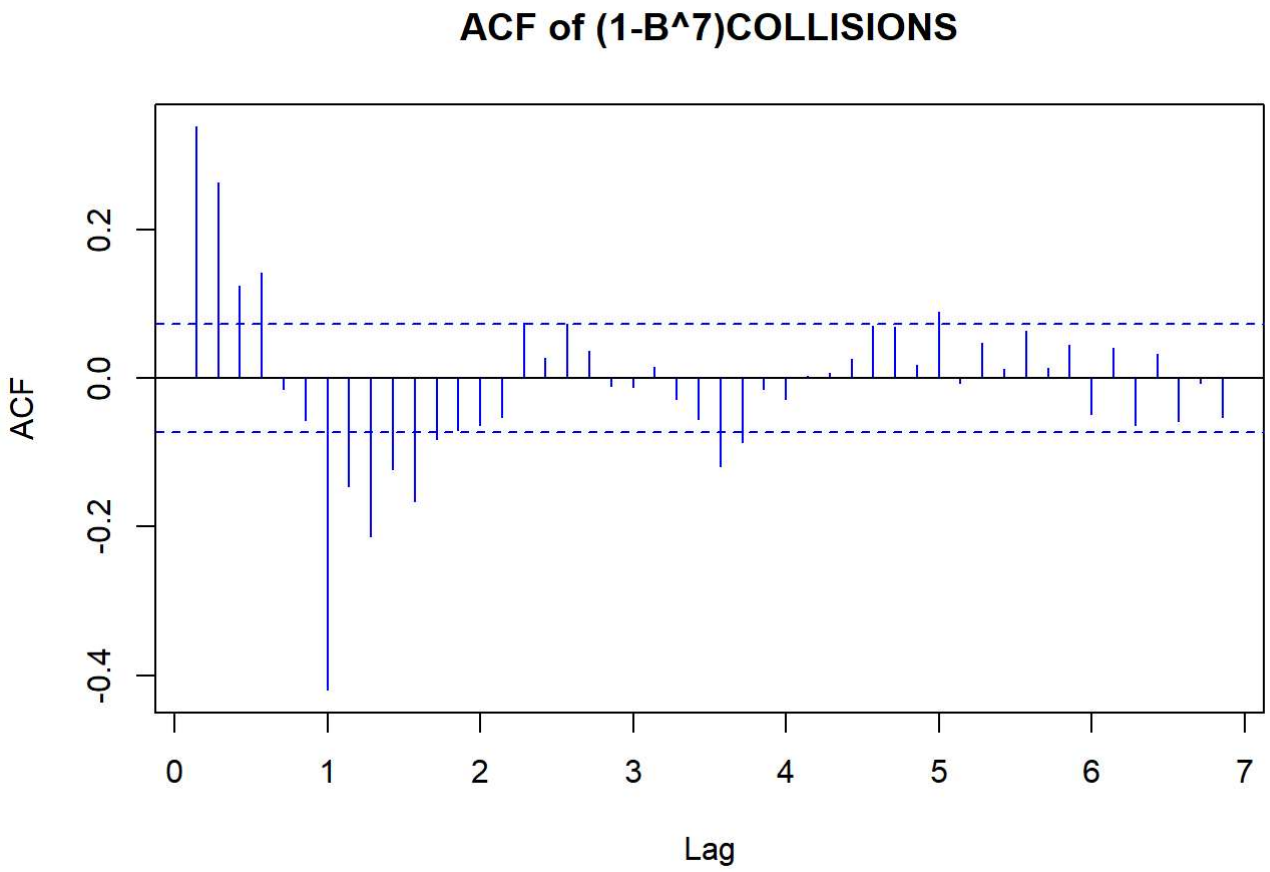



```
#ACF Plot
acf(crash[,"COLLISIONS"], main = "ACF of Collisions in NYC (Lags 0-28)")
```

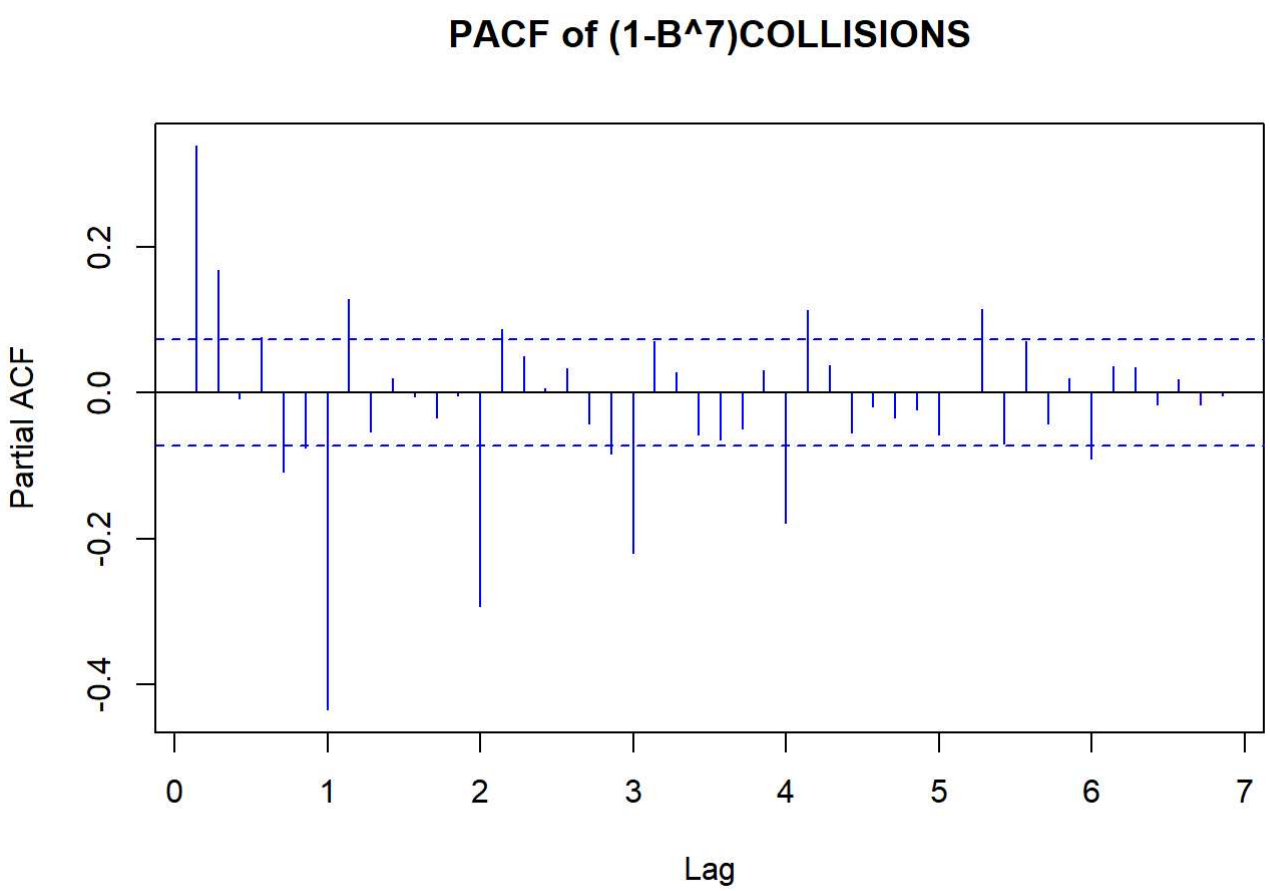


#: The Time Series plot shows that there is no clear Trend or Cyclical component in the series; thus, differencing at lag 1 is unnecessary. However, there seems to be strong autocorrelation between lag 0 (time t) and every lag that is a factor of 7 (lags 7, 14, 21, etc). The time series needs to be differenced at the (daily) seasonal level to remove the seasonal componen t so that the time series is stationary (constant mean, constant variance, and CONSISTENT AUTOCOVARIANCE AT ALL LAGS). IN OT HER WORDS, although there seems to be a constant mean and variance in the time series, which indicates that there is no (cle ar and obvious) trend component, there is still strong residual seasonal autocorrelation at lags 7, 14, 21 (which implies th at there is considerable weekly/daily seasonality in the data). Additionally, because there seems to be nonstationary season al behavior in the data (extremely slow decay at the seasonal lags, at factors of 7), seasonal differencing is necessary at s=7.

```
#Seasonal Differencing (ACF and PACF Plot)

acf(diff(crash[,"COLLISIONS"], lag=7), lag=48, col="blue", main="ACF of (1-B^7)COLLISIONS")
```




```
pacf(diff(crash[,"COLLISIONS"], lag=7),lag = 48, col='blue', lwd=1, main="PACF of (1-B^7)COLLISIONS")
```




```
# COMMENT ON STATIONARITY: After differencing, the series appears stationary.

# SEASONAL COMMENTS: After differencing at the seasonal lags, PACF seems to be decaying slowly at lags 7, 14, 21, 28, before reaching 0 autocorrelation. ACF is chopped off after lag 7; thus, an MA(1)s=7 Process likely best model fit.

 # NONSEASONAL COMMENTS: Perhaps a chopping off after lag 1 (nonseasonal component) in the PACF. ACF at nonseasonal lags seems to decay to 0.
```

```
# Estimating a model using training sample
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 4.4.3
```

```
## Registered S3 methods overwritten by 'forecast':
##   method       from
##   fitted.Arima TSA
##   plot.Arima   TSA
```

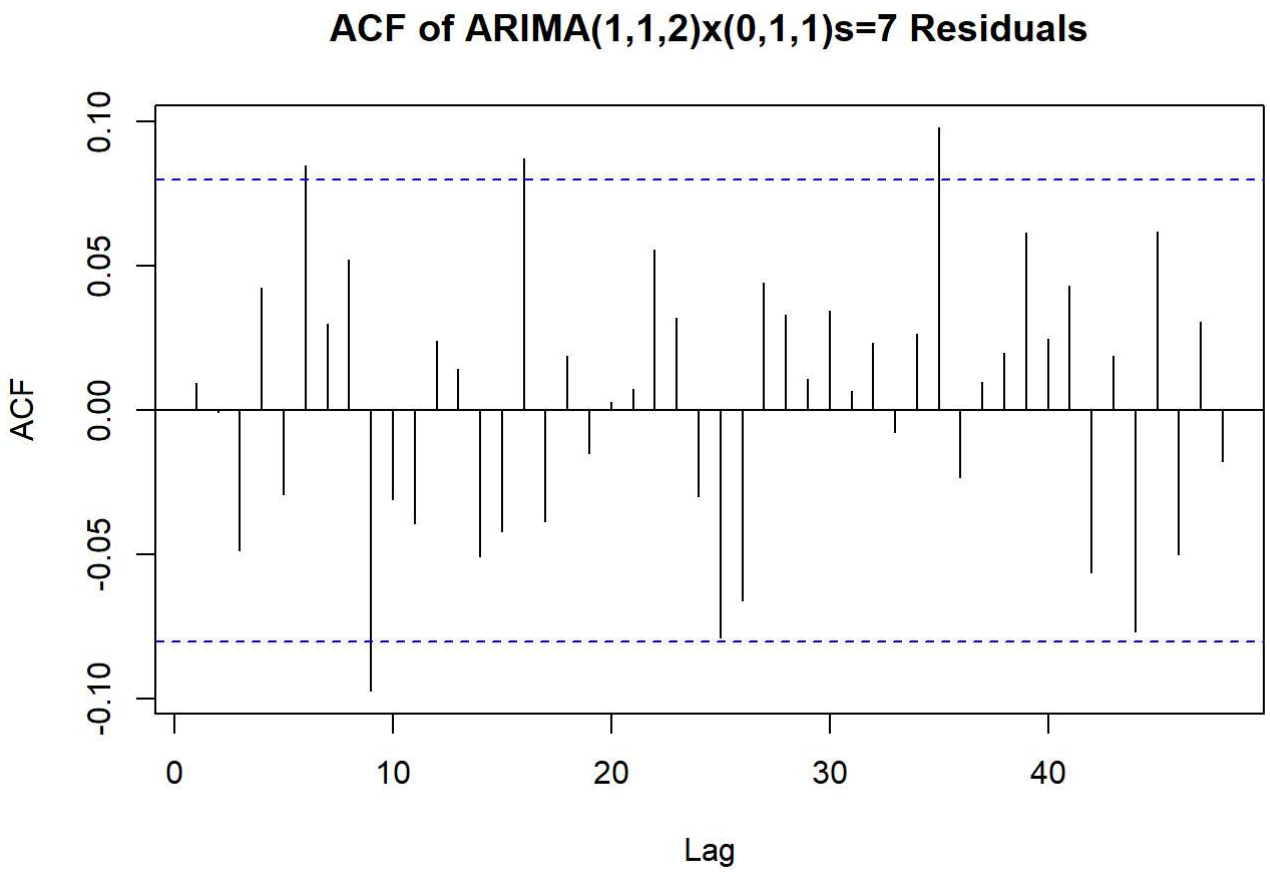
```
##
## Attaching package: 'forecast'
```

```
## The following object is masked from 'package:Metrics':
##
##     accuracy
```

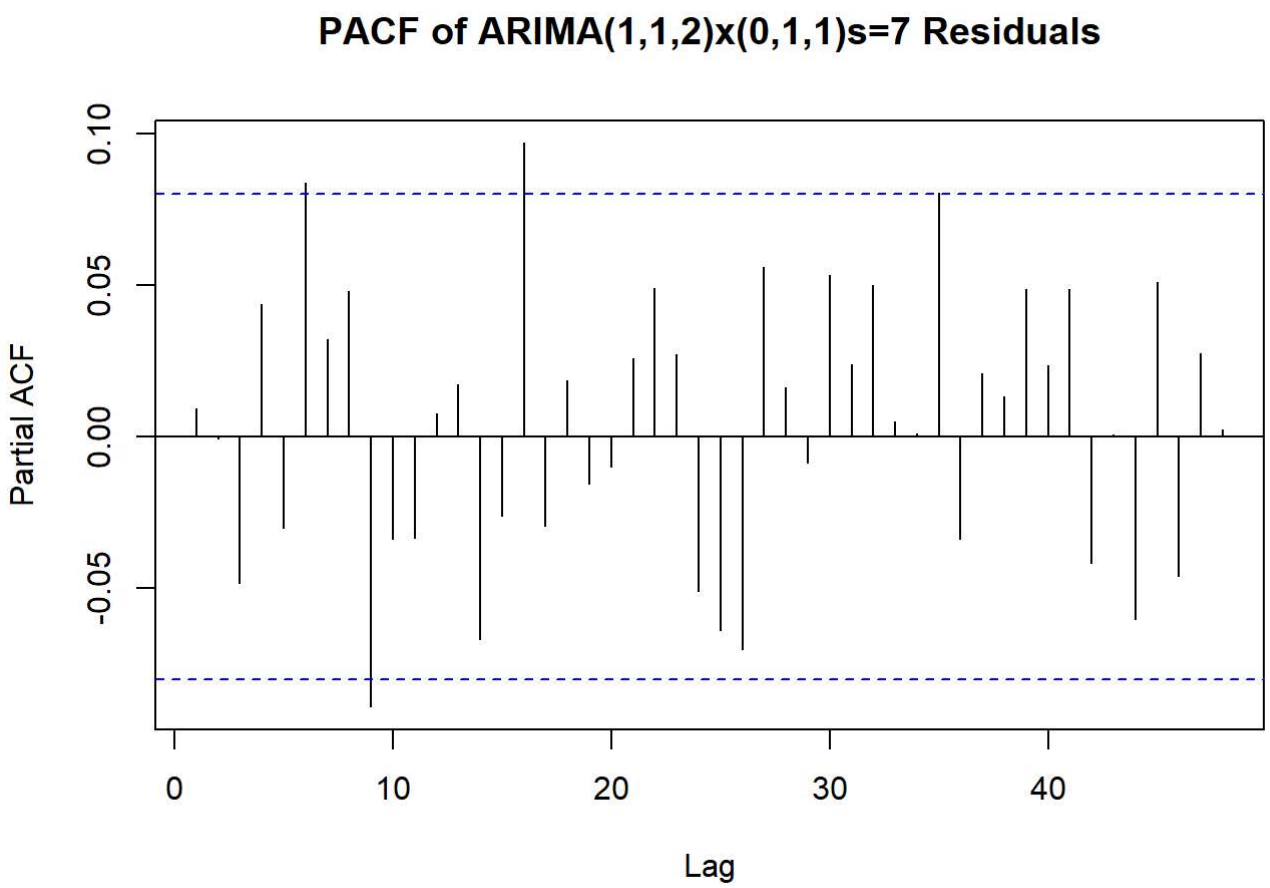
```
fit = Arima(train_crash, order = c(1,1,2),seasonal=list(order=c(0,1,1), period=7), lambda=0)

summary(fit)
```

```
## Series: train_crash
## ARIMA(1,1,2)(0,1,1)[7]
## Box Cox transformation: lambda= 0
##
## Coefficients:
##          ar1      ma1     ma2     sma1
##       0.7208  -1.3843  0.4037  -0.9773
## s.e.  0.0853   0.1072  0.1001   0.0221
##
## sigma^2 = 0.01412:  log likelihood = 409.99
## AIC=-809.99   AICc=-809.89   BIC=-788.07
##
## Training set error measures:
##                      ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -1.643512 69.56998 50.76395 -1.556262 8.579936 0.6144068
##                       ACF1
## Training set -0.001224739
```

```
acf(fit$residuals, main = "ACF of ARIMA(1,1,2)x(0,1,1)s=7 Residuals", lag = 48)
```




```
pacf(fit$residuals, main = "PACF of ARIMA(1,1,2)x(0,1,1)s=7 Residuals", lag = 48)
```




```
Box.test(fit$residuals, lag=24)
```

```
##
##  Box-Pierce test
##
## data:  fit$residuals
## X-squared = 28.597, df = 24, p-value = 0.2357
```

```
#COMMENT: By applying a Moving Average Component to the seasonality, the underlying autocorrelation at the weekly seasonal l
evel [lag (s=7)] was effectively removed. Adding the Moving Average component to the difference of Yt and Yt-7 effectively h
andles and captures most of the residual relationship left over that influences todays number of collisions in NYC with last
weeks; however, there was still correlation at the nonseasonal lags left to be captured. The Box Pierce test returned an ext
remely small p value, indicating that the autocorrelation at lags 1-24 were not yet white noise. Moreover, there appeared to
be extremely slow decay in both the ACF and PACF at the nonseasonal lags, with random lags having autocorrelation and partia
l autocorrelation statistically different than 0. Thus, nonseasonal differencing was applied to remove any hidden trend/cycl
ical component of the series.

# After adding the nonseasonal differencing to the model, the ACF appeared to decay to 0 in a stationary manner and the PACF
appeared to chop off after lag 1. Thus, an AR(1) process was added to the model, to capture the autocorrelations of the nons
easonal component of the series.

# We then argued that the PACF exhibited decay, after adding the AR(1) process to the model, and the ACF chopped off after l
ag 2; thus, we added an MA(2) process to the model, at the nonseasonal component (to get a more favorable p value from the B
ox Pierce Test.)
# This sequential procedure, cumulatively, removed the autocorrelations and partial autocorrelations at all nonseasonal and
seasonal lags of the model, while keeping our model parsimonious to diminish the likelihood of overfitting.

# COMMENT: This step by step procedure, moving from using strictly an MA(1)s=7 process to a Mixed ARIMA (1,1,2)x(0,1,1)s=7 p
rocess to capture correlations at seasonal and nonseasonal lags (after differencing), increased the p value of the box pierc
e test significantly (from 2.2e-16 to 0.2357) . This also increased training fit significantly across the board of all viabl
e error metrics.

#COMMENT: Because the p value of the Box pierce test is now greater than our level of significance (0.05), we fail to reject
the null hypothesis that the residuals are no different than white noise. Thus, because the residuals resemble white noise,
the correlation between errors at lags 1-24 are statistically equal to 0. The model can now effectively be used for forecast
ing at the 5% level of significance. Moreover, all coefficients in the model are statistically significant at the 5% level o
f significance.
```

```
# Test Performance on Holdout Sample Using Section 4.1 Model

preds <- forecast(fit, h = length(test_crash)) # Generate forecasts for the test set

# Display Summary of Predictions
summary(preds)
```

```
##
## Forecast method: ARIMA(1,1,2)(0,1,1)[7]
##
## Model Information:
## Series: train_crash
## ARIMA(1,1,2)(0,1,1)[7]
## Box Cox transformation: lambda= 0
##
## Coefficients:
##          ar1      ma1     ma2     sma1
##       0.7208  -1.3843  0.4037  -0.9773
## s.e.  0.0853   0.1072  0.1001   0.0221
##
## sigma^2 = 0.01412:  log likelihood = 409.99
## AIC=-809.99   AICc=-809.89   BIC=-788.07
##
## Error measures:
##                       ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -1.643512 69.56998 50.76395 -1.556262 8.579936 0.6144068
##                      ACF1
## Training set -0.001224739
##
## Forecasts:
##     Point Forecast    Lo 80    Hi 80    Lo 95     Hi 95
## 601       673.1277 577.9917 783.9230 533.2000  849.7766
## 602       694.6924 591.5176 815.8635 543.2560  888.3428
## 603       723.4075 612.9653 853.7487 561.4996  932.0014
## 604       602.8773 509.2980 713.6510 465.7920  780.3074
## 605       527.0482 444.3629 625.1193 405.9803  684.2199
## 606       634.1765 533.9568 753.2067 487.4838  825.0116
## 607       658.0143 553.4760 782.2974 505.0378  857.3275
## 608       666.5418 560.0080 793.3422 510.6889  869.9582
## 609       690.0397 579.3278 821.9090 528.1035  901.6314
## 610       720.1762 604.2648 858.3219 550.6599  941.8768
## 611       601.1557 504.1391 716.8423 459.2903  786.8406
## 612       526.1562 441.0393 627.7000 401.7056  689.1623
## 613       633.6354 530.9065 756.2421 483.4496  830.4770
## 614       657.8512 550.9776 785.4552 501.6213  862.7390
## 615       666.6676 557.9996 796.4983 507.8398  875.1692
```

```
## 616       690.3872 577.5955 825.2046 525.5503  906.9247
## 617       720.7024 602.7071 861.7984 548.2783  947.3510
## 618       601.6934 502.9843 719.7738 457.4658  791.3924
## 619       526.6889 440.1175 630.2889 400.2079  693.1427
## 620       634.3308 529.8711 759.3839 481.7286  835.2743
## 621       658.6135 549.9564 788.7384 499.8936  867.7283
## 622       667.4696 557.0078 799.8374 506.1378  880.2262
## 623       691.2398 576.5960 828.6780 523.8177  912.1731
## 624       721.6090 601.6823 865.4393 546.4893  952.8447
## 625       602.4602 502.1386 722.8249 455.9824  795.9918
## 626       527.3664 439.3823 632.9689 398.9143  697.1806
## 627       635.1522 528.9885 762.6221 480.1730  840.1521
## 628       659.4705 549.0409 792.1110 498.2783  872.8082
## 629       668.3411 556.0795 803.2662 504.4996  885.3918
## 630       692.1445 575.6324 832.2394 522.1179  917.5400
## 631       722.5551 600.6730 869.1683 544.7100  958.4657
## 632       603.2511 501.2928 725.9468 454.4926  800.6995
## 633       528.0594 438.6388 635.7092 397.6059  701.3143
## 634       635.9874 528.0891 765.9314 478.5918  845.1460
## 635       660.3381 548.1026 795.5561 496.6307  878.0092
## 636       669.2207 555.1239 806.7682 502.8241  890.6818
## 637       693.0556 574.6378 835.8762 520.3762  923.0361
## 638       723.5064 599.6292 872.9754 542.8848  964.2221
## 639       604.0455 500.4170 729.1338 452.9630  805.5203
## 640       528.7548 437.8683 638.5062 396.2621  705.5472
## 641       636.8250 527.1564 769.3087 476.9673  850.2595
## 642       661.2077 547.1294 799.0718 494.9378  883.3345
## 643       670.1021 554.1326 810.3417 501.1024  896.0979
## 644       693.9684 573.6061 839.5869 518.5866  928.6630
## 645       724.4594 598.5466 876.8596 541.0095  970.1149
## 646       604.8411 499.5088 732.3850 451.3917  810.4551
## 647       529.4512 437.0694 641.3594 394.8817  709.8798
## 648       637.6638 526.1897 772.7538 475.2990  855.4933
## 649       662.0787 546.1209 802.6577 493.1995  888.7847
## 650       670.9847 553.1058 813.9862 499.3349  901.6404
## 651       694.8825 572.5376 843.3711 516.7497  934.4207
## 652       725.4136 597.4258 880.8205 539.0851  976.1444
## 653       605.6378 498.5687 735.7002 449.7796  815.5040
## 654       530.1486 436.2428 644.2686 393.4658  714.3125
```

```
## 655       638.5037 525.1896 776.2663 473.5879  860.8474
## 656       662.9507 545.0778 806.3136 491.4170  894.3599
## 657       671.8685 552.0441 817.7015 497.5229  907.3096
## 658       695.7978 571.4331 847.2287 514.8671  940.3097
## 659       726.3691 596.2675 884.8579 537.1131  982.3109
## 660       606.4355 497.5975 739.0793 448.1280  820.6674
## 661       530.8469 435.3890 647.2337 392.0154  718.8452
## 662       639.3447 524.1568 779.8460 471.8356  866.3222
## 663       663.8240 544.0010 810.0394 489.5918  900.0605
## 664       672.7535 550.9483 821.4876 495.6679  913.1058
## 665       696.7143 570.2936 851.1594 512.9401  946.3302
## 666       727.3259 595.0728 888.9717 535.0951  988.6147
## 667       607.2343 496.5960 742.5221 446.4381  825.9453
## 668       531.5461 434.5087 650.2546 390.5317  723.4784
## 669       640.1868 523.0924 783.4929 470.0433  871.9180
## 670       664.6983 542.8914 813.8348 487.7253  905.8867
## 671       673.6396 549.8195 825.3442 493.7714  919.0292
## 672       697.6320 569.1199 855.1631 510.9703  952.4826
## 673       728.2839 593.8426 893.1616 533.0326  995.0561
## 674       608.0341 495.5649 746.0284 444.7112  831.3383
## 675       532.2462 433.6027 653.3310 389.0158  728.2122
## 676       641.0301 521.9970 787.2067 468.2124  877.6350
## 677       665.5739 541.7497 817.6997 485.8190  911.8388
## 678       674.5269 548.6584 829.2711 491.8347  925.0802
## 679       698.5509 567.9129 859.2396 508.9592  958.7670
## 680       729.2432 592.5778 897.4275 530.9272 1001.6356
## 681       608.8350 494.5050 749.5982 442.9487  836.8464
## 682       532.9473 432.6715 656.4630 387.4688  733.0470
## 683       641.8744 520.8715 790.9874 466.3442  883.4736
## 684       666.4505 540.5769 821.6340 483.8741  917.9171
## 685       675.4154 547.4658 833.2683 489.8592  931.2592
## 686       699.4710 566.6736 863.3889 506.9081  965.1841
## 687       730.2037 591.2792 901.7693 528.7804 1008.3534
## 688       609.6369 493.4171 753.2313 441.1518  842.4702
## 689       533.6493 431.7159 659.6503 385.8918  737.9829
## 690       642.7199 519.7166 794.8348 464.4401  889.4341
## 691       667.3284 539.3737 825.6375 481.8921  924.1221
## 692       676.3050 546.2427 837.3356 487.8464  937.5666
## 693       700.3923 565.4026 867.6107 504.8186  971.7340
```

```
## 694       731.1655 589.9479 906.1868 526.5935 1015.2100
## 695       610.4399 492.3020 756.9276 439.3216  848.2099
## 696       534.3522 430.7366 662.8930 384.2860  743.0203
## 697       643.5665 518.5333 798.7487 462.5014  895.5169
## 698       668.2074 538.1411 829.7101 479.8743  930.4542
## 699       677.1958 544.9899 841.4729 485.7976  944.0027
## 700       701.3148 564.1010 871.9050 502.6919  978.4174
## 701       732.1286 588.5847 910.6799 524.3682 1022.2059
## 702       611.2440 491.1603 760.6870 437.4595  854.0659
## 703       535.0560 429.7341 666.1910 382.6522  748.1597
## 704       644.4142 517.3221 802.7292 460.5293  901.7225
## 705       669.0875 536.8797 833.8519 477.8221  936.9138
## 706       678.0878 543.7080 845.6802 483.7140  950.5681
## 707       702.2386 562.7696 876.2717 500.5295  985.2347
## 708       733.0929 587.1905 915.2486 522.1058 1029.3417
## 709       612.0491 489.9928 764.5095 435.5666  860.0387
## 710       535.7608 428.7091 669.5441 380.9917  753.4013
## 711       645.2630 516.0840 806.7762 458.5250  908.0514
## 712       669.9688 535.5904 838.0626 475.7367  943.5015
## 713       678.9810 542.3981 849.9573 481.5971  957.2633
## 714       703.1636 561.4091 880.7108 498.3328  992.1864
## 715       734.0585 585.7660 919.8928 519.8077 1036.6179
## 716       612.8553 488.8002 768.3950 433.6440  866.1289
## 717       536.4665 427.6622 672.9524 379.3054  758.7457
## 718       646.1129 514.8196 810.8896 456.4899  914.5040
## 719       670.8513 534.2739 842.3422 473.6193  950.2177
## 720       679.8753 541.0607 854.3043 479.4480  964.0888
## 721       704.0898 560.0204 885.2221 496.1030  999.2732
## 722       735.0254 584.3123 924.6124 517.4753 1044.0352
## 723       613.6625 487.5832 772.3434 431.6930  872.3368
## 724       537.1731 426.5941 676.4158 377.5943  764.1932
## 725       646.9639 513.5297 815.0693 454.4251  921.0810
## 726       671.7349 532.9311 846.6908 471.4713  957.0631
## 727       680.7709 539.6968 858.7210 477.2681  971.0452
## 728       705.0172 558.6042 889.8057 493.8414 1006.4957
## 729       735.9936 582.8300 929.4075 515.1099 1051.5941
## 730       614.4708 486.3426 776.3549 429.7147  878.6631
## 731       537.8807 425.5053 679.9343 375.8593  769.7444
```

```
# Extract Forecast Values
holdfit <- preds$mean

# Accuracy Measures for Holdout Sample
accuracy(holdfit, test_crash)
```

```
##                 ME     RMSE      MAE       MPE     MAPE
## Test set -1.495009 78.42954 58.11444 -1.864279 9.444852
```

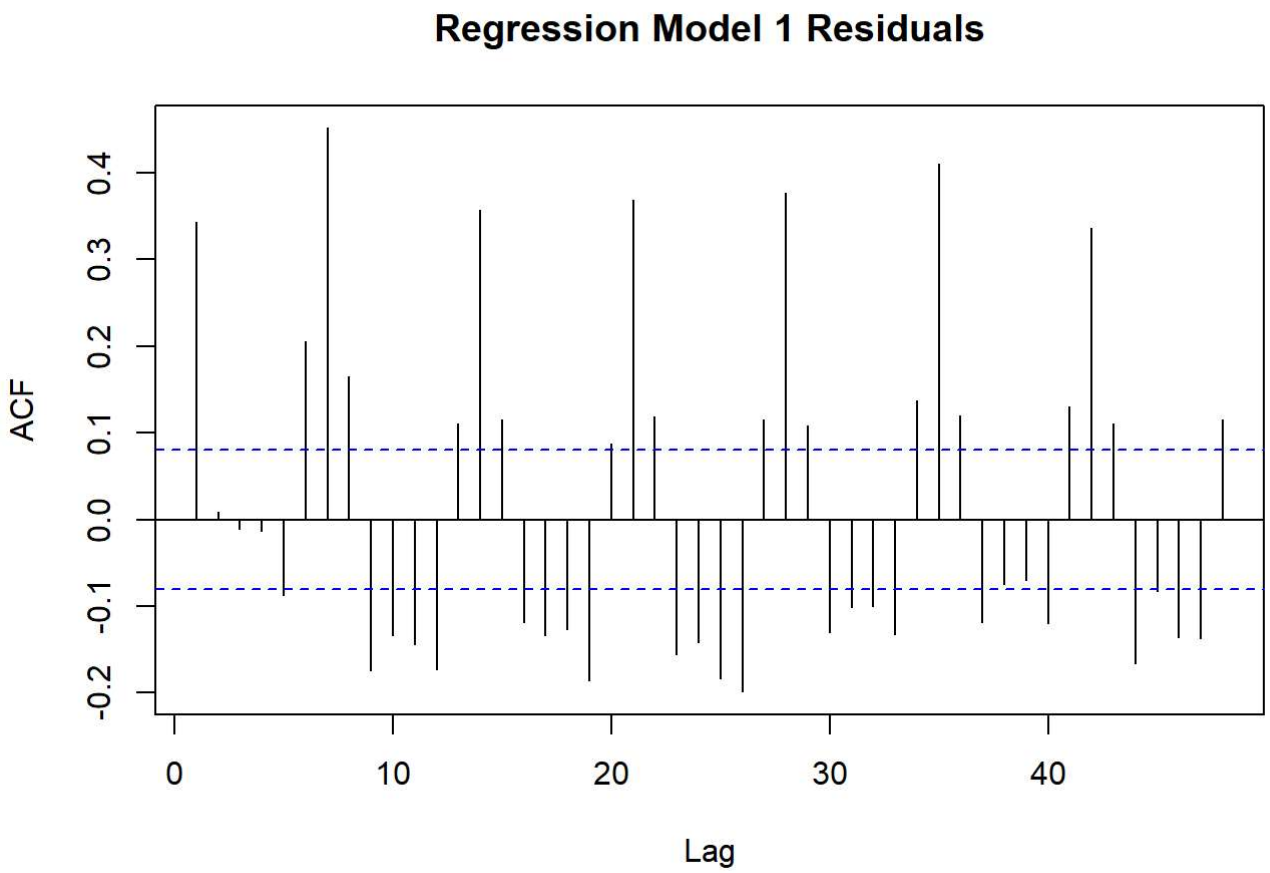
```
# TABLE of Predictive Performance Metrics FOR 4.1

#training metrics
acc_train<- accuracy(forecast(fit, h=length(train_crash)), train_crash)

#manually deriving Training R-Squared Metrics
REG_Train_R2 <- 1 - sum((train_crash - fitted(fit))^2) / sum((train_crash - mean(train_crash))^2)

#test metrics
acc<- accuracy(holdfit, test_crash)

#manually deriving Test R-Squared Metric
REG_Test_R2 <- 1 - sum((test_crash - holdfit)^2) / sum((test_crash - mean(test_crash))^2)


Model_Metric_df <- data.frame(Metric = c("MAPE", "RMSE", "MAE", "R2"),
                Training_set = c(acc_train[1,5], acc_train[1,2], acc_train[1,3], REG_Train_R2),
                Holdout_set = c(acc[,5], acc[,2], acc[,3], REG_Test_R2))
Model_Metric_df
```

```
##   Metric Training_set Holdout_set
## 1   MAPE    8.5799361   9.4448525
## 2   RMSE   69.5699780  78.4295399
## 3    MAE   50.7639476  58.1144394
## 4     R2    0.5107106   0.3792855
```

```
# Best: Best performance on holdout set (for most metrics). Simpler model. <-- Relative to Model derived in 4.2 and 4.3.
# HOWEVER, it is close between this model and 4.3 (on TEST/HOLDOUT sample). Very.
# Simplest: 4.1
# BEST MAPE: 4.3
# BEST RMSE: 4.1
# Best MAE: 4.3
# Best R^2: 4.1
```

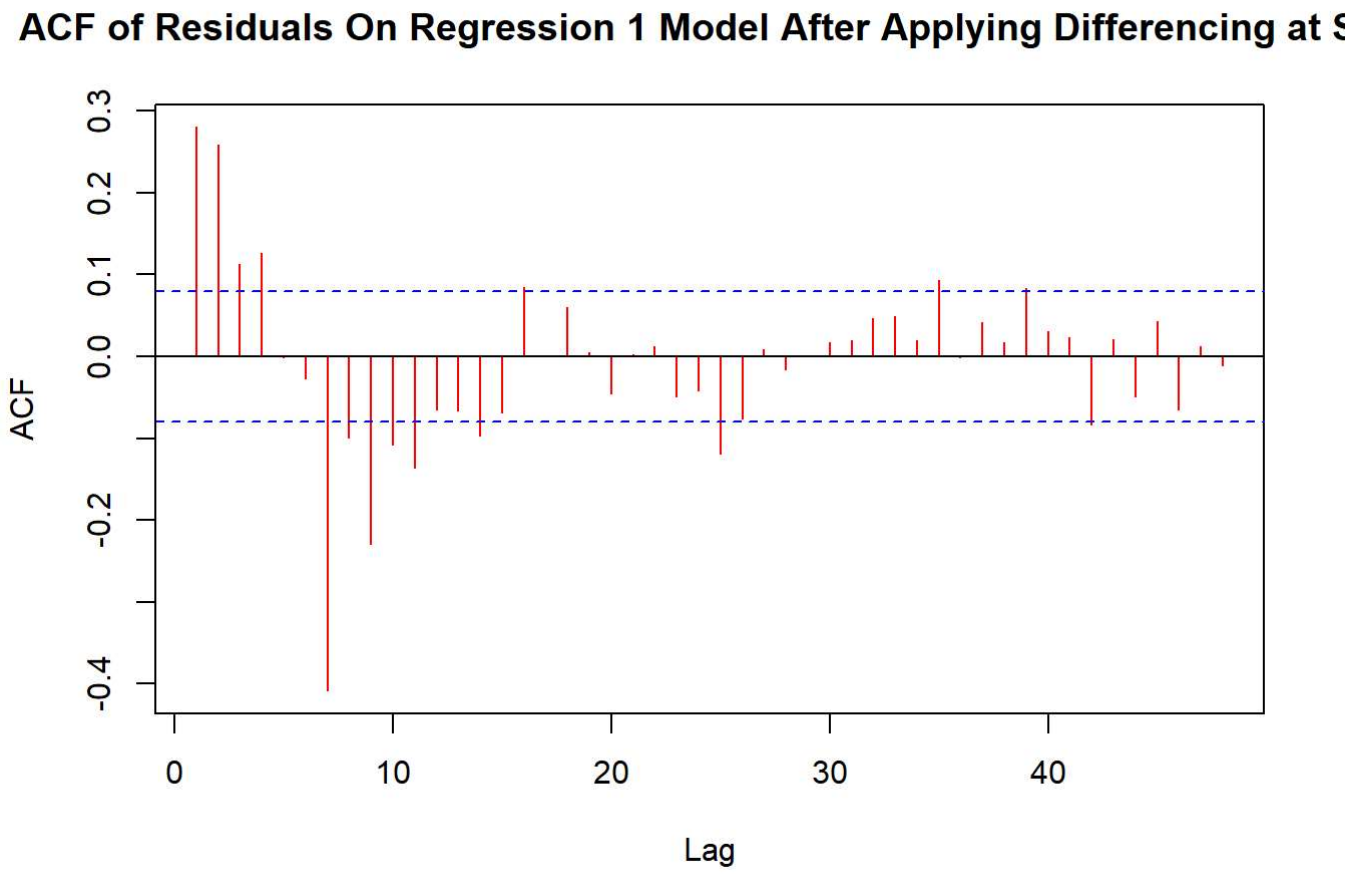
SECTION 4.2) You will be analyzing the residuals of the time series regression models from SECTION 3.3. If these residuals behave like stationary time series then you can model them using an AR, MA, or ARMA process as discussed in Lecture Sets 7 and 8. If the residuals do not exhibit stationary behavior then you need to difference the dependent (target) variable (and your predictors). NOTE: Professor stated that it is best to simply model and analyze the residuals of the "best" time series regression model from 3.3.

```
# Time Series Regression Model 1 - Collisions regressed on the three independent variables (best time series regression mode
l from 3.3)
acf(REG_1$residuals, main = "Regression Model 1 Residuals", lag=48)
```

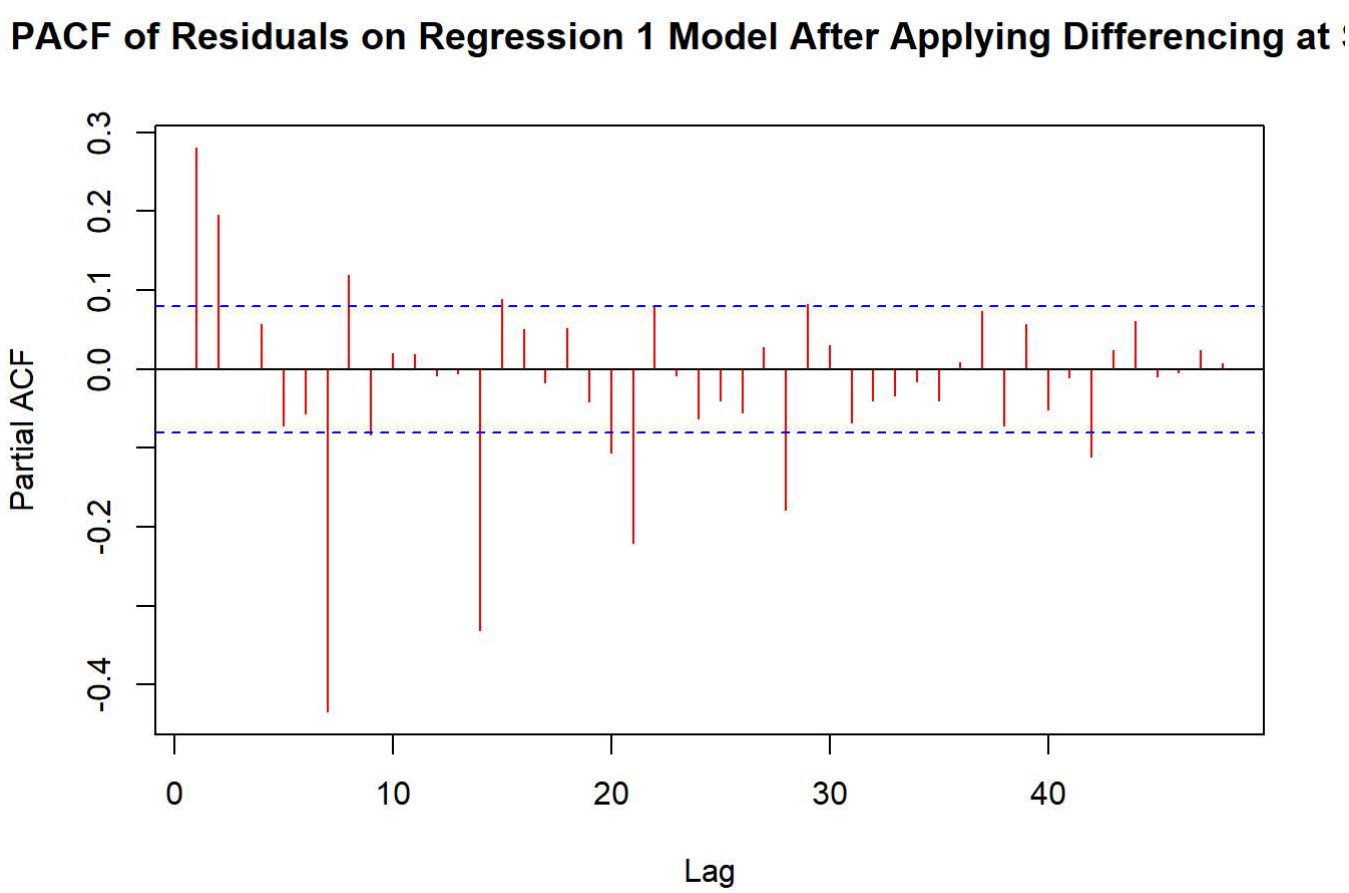



```
# COMMENT: Nonstationary residuals at the seasonal (s=7) lags. Need to difference dependent variable at the seasonal level.
```

```
# ACF AND PACF OF DIFFERENCED RESIDUALS

x = cbind(train_AWND, train_TMAX, train_TMIN)

regdif = arima(train_crash, seasonal = list(order = c(0, 1, 0), period = 7), xreg=x)

acf(regdif$residuals, main = "ACF of Residuals On Regression 1 Model After Applying Differencing at S=7", lag = 48, col='re
d')
```




```
pacf(regdif$residuals, main = "PACF of Residuals on Regression 1 Model After Applying Differencing at S=7", lag = 48, col='r
ed')
```

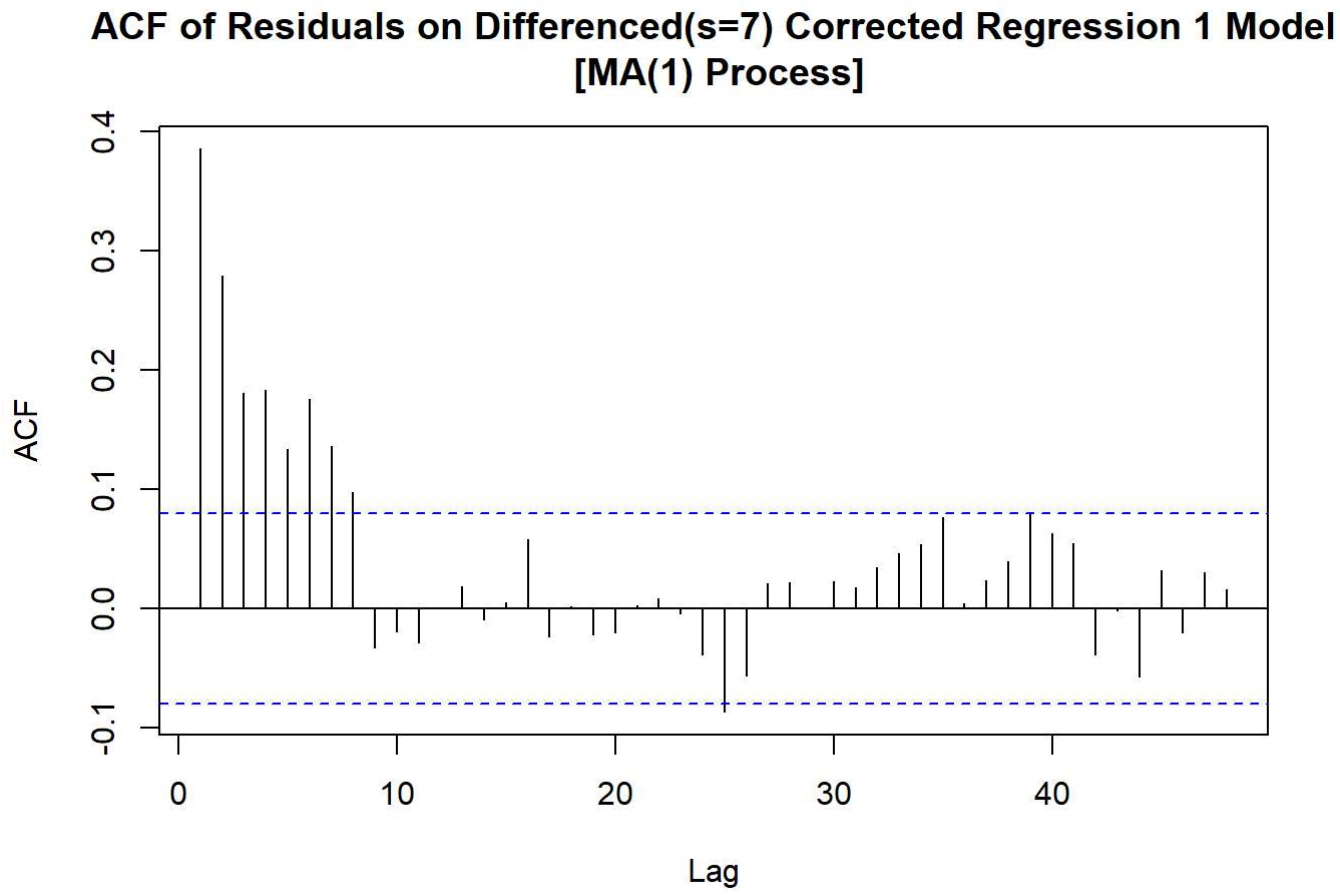



```
Box.test(regdif$residuals)
```

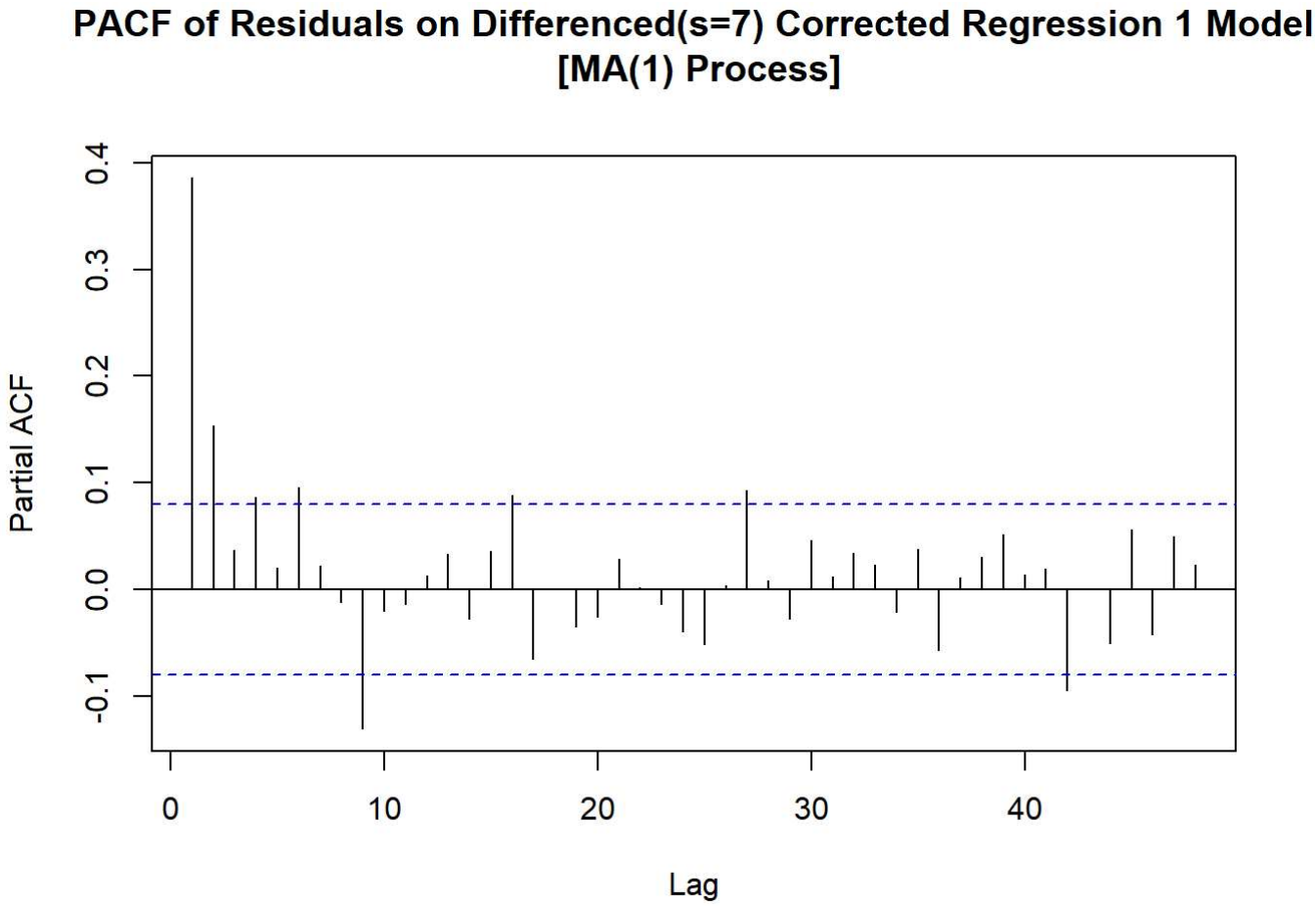
```
##
##  Box-Pierce test
##
## data:  regdif$residuals
## X-squared = 47.063, df = 1, p-value = 6.876e-12
```

```
# COMMENT: After differencing, appears to be gradual decay in PACF at seasonal lags. Chopping off at ACF, lag 7. Therefore,
because differenced series indicates stationarity, will try fitting an MA(1) process at the first seasonal lag (7).
```

```
# ESTIMATING CORRECTED REGRESSION MODEL

regarima <- arima(train_crash, seasonal=list(order = c(0,1,1),period=7), xreg=x)

acf(regarima$residuals, main = "ACF of Residuals on Differenced(s=7) Corrected Regression 1 Model\n [MA(1) Process]", lag =
48)
```




```
pacf(regarima$residuals, main = "PACF of Residuals on Differenced(s=7) Corrected Regression 1 Model\n [MA(1) Process]", lag
= 48)
```




```
Box.test(regarima$residuals)
```

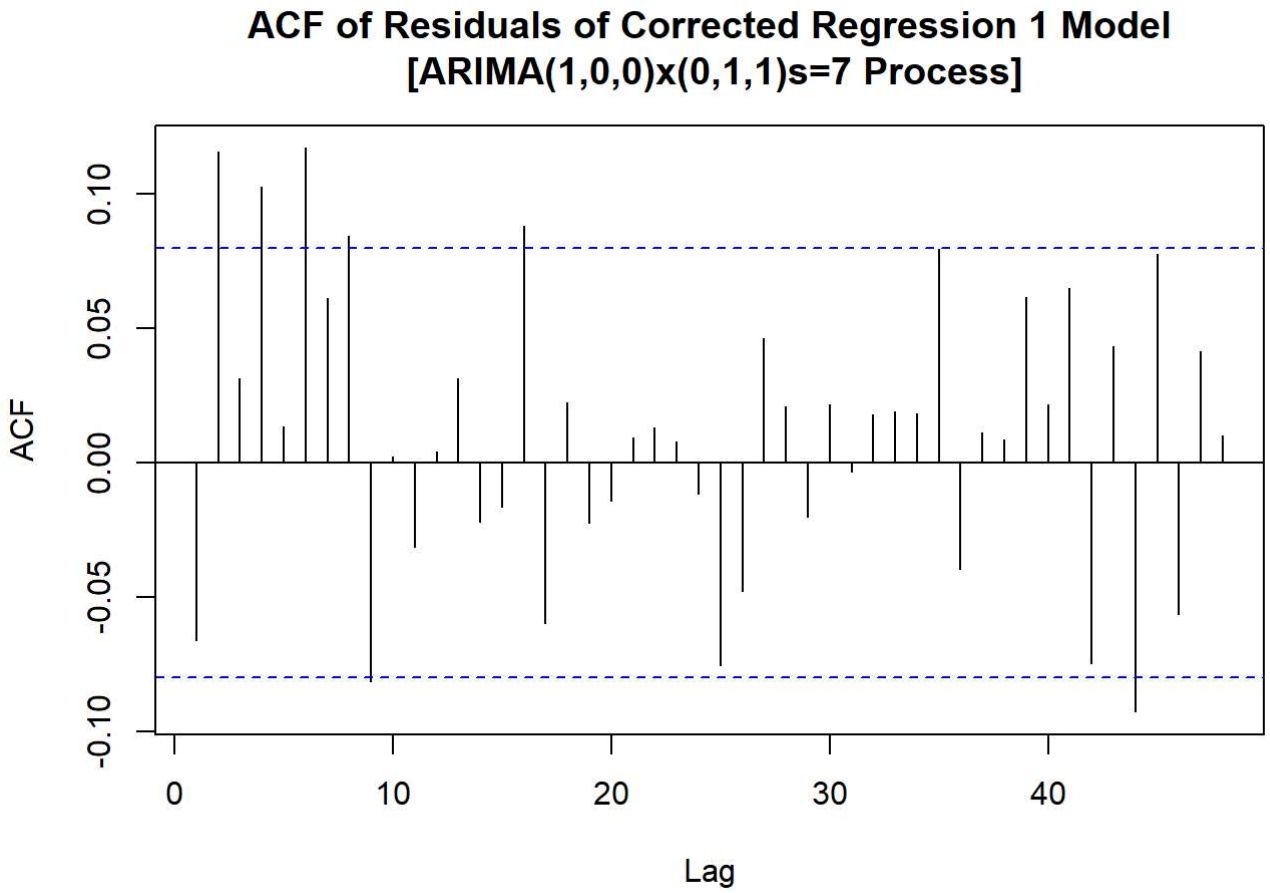
```
##
##  Box-Pierce test
##
## data:  regarima$residuals
## X-squared = 89.213, df = 1, p-value < 2.2e-16
```

 # COMMENT: Autocorrelation and partial autocorrelation seems to be handled; however, the p value returned by the Box Pierce test is still extremely small. Moreover, there seems to be gradual decay in the ACF at the nonseasonal lags, with PACF showing a chopping off at lag 1. Therefore, will try adding an AR(1) process at the nonseasonal lags to the model.

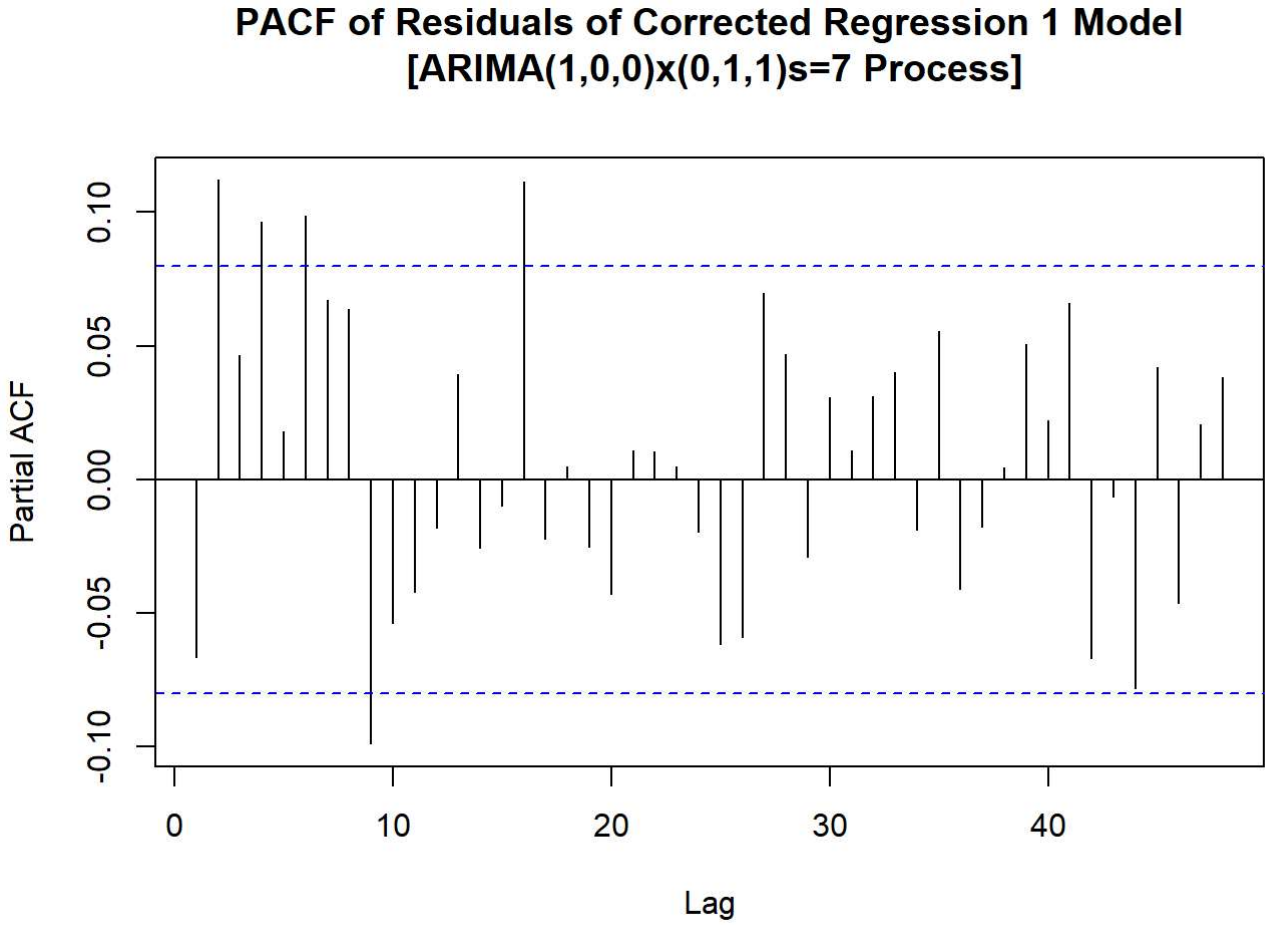
```
# ATTEMPTING TO IMPROVE CORRECTED MODEL


regarima <- Arima(train_crash, order = c(1,0,0), seasonal=list(order = c(0,1,1),period=7), xreg=x)
summary(regarima)
```

```
## Series: train_crash
## Regression with ARIMA(1,0,0)(0,1,1)[7] errors
##
## Coefficients:
##          ar1     sma1  train_AWND  train_TMAX  train_TMIN
##       0.3942  -0.9768      2.0944      4.1983     -1.5587
## s.e.  0.0383   0.0222      3.0388      1.0086      1.1205
##
## sigma^2 = 4911:  log likelihood = -3369.78
## AIC=6751.55   AICc=6751.7   BIC=6777.86
##
## Training set error measures:
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 4.838229 69.37749 50.60893 -0.541593 8.450621 0.6125305
##                      ACF1
## Training set -0.06634741
```

```
acf(regarima$residuals,
    main = "ACF of Residuals of Corrected Regression 1 Model\n [ARIMA(1,0,0)x(0,1,1)s=7 Process]",
    lag = 48)
```




```
pacf(regarima$residuals,
     main = "PACF of Residuals of Corrected Regression 1 Model\n [ARIMA(1,0,0)x(0,1,1)s=7 Process]",
     lag = 48)
```




```
Box.test(regarima$residuals)
```

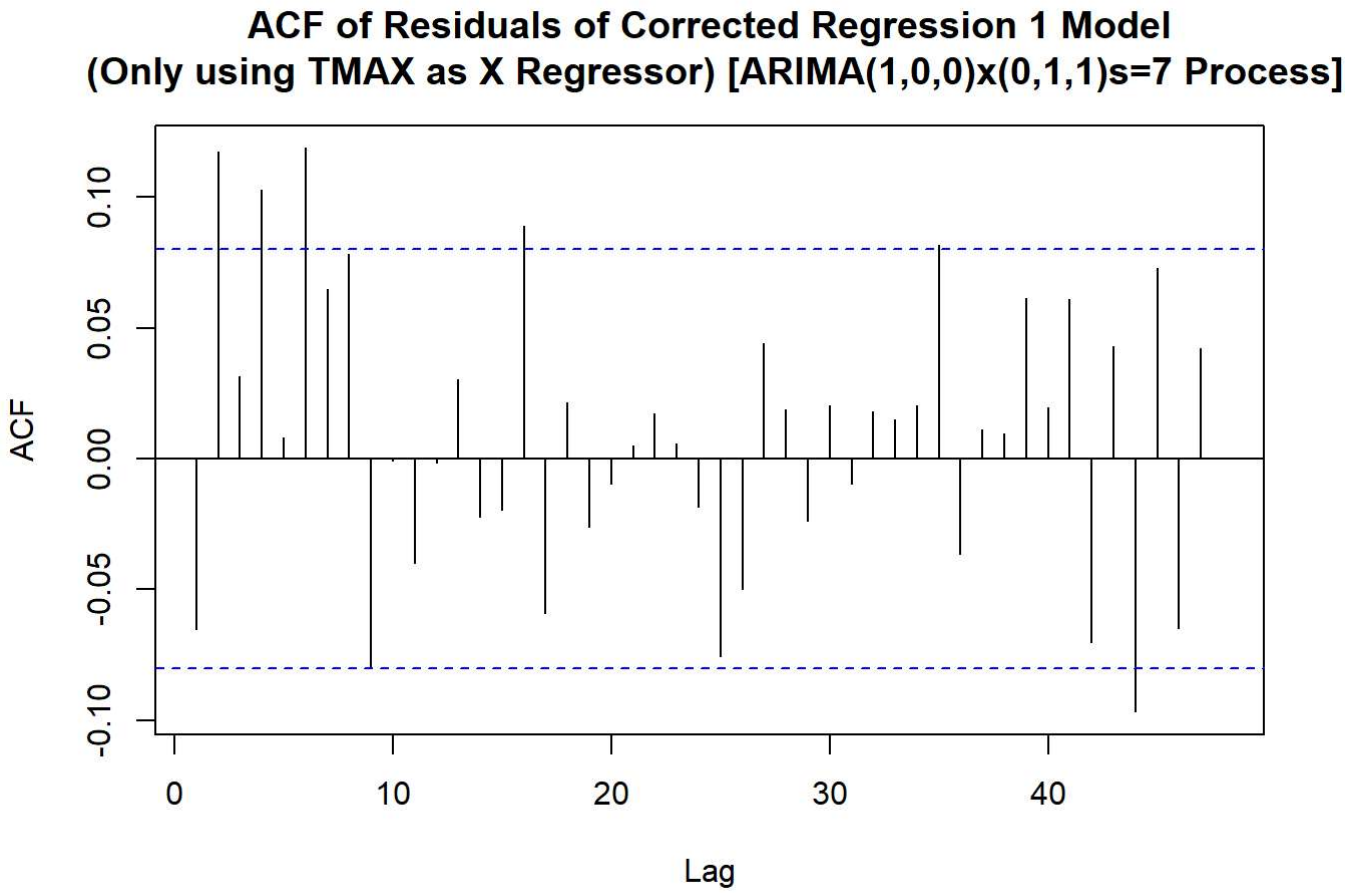
```
##
##  Box-Pierce test
##
## data:  regarima$residuals
## X-squared = 2.6412, df = 1, p-value = 0.1041
```

```
# COMMENT: MULTIPLICATIVE ARMA(1,0,0) x (0,1,1) appears to be best parsimonious model. At the 5% level of significance (usin
g the p-value derived from the Box-Pierce Test, at lag=1), we fail to reject the null hypothesis that the residuals are whit
e noise; thus, this model can be used for modeling the TS. However, it is notable that neither the 'train_AWND' nor the 'tra
in_TMIN' features are significant. They can be removed from the model; thus, will drop these features from the model and inc
rease the lag in the Box Pierce Test from 1 to 24.
```

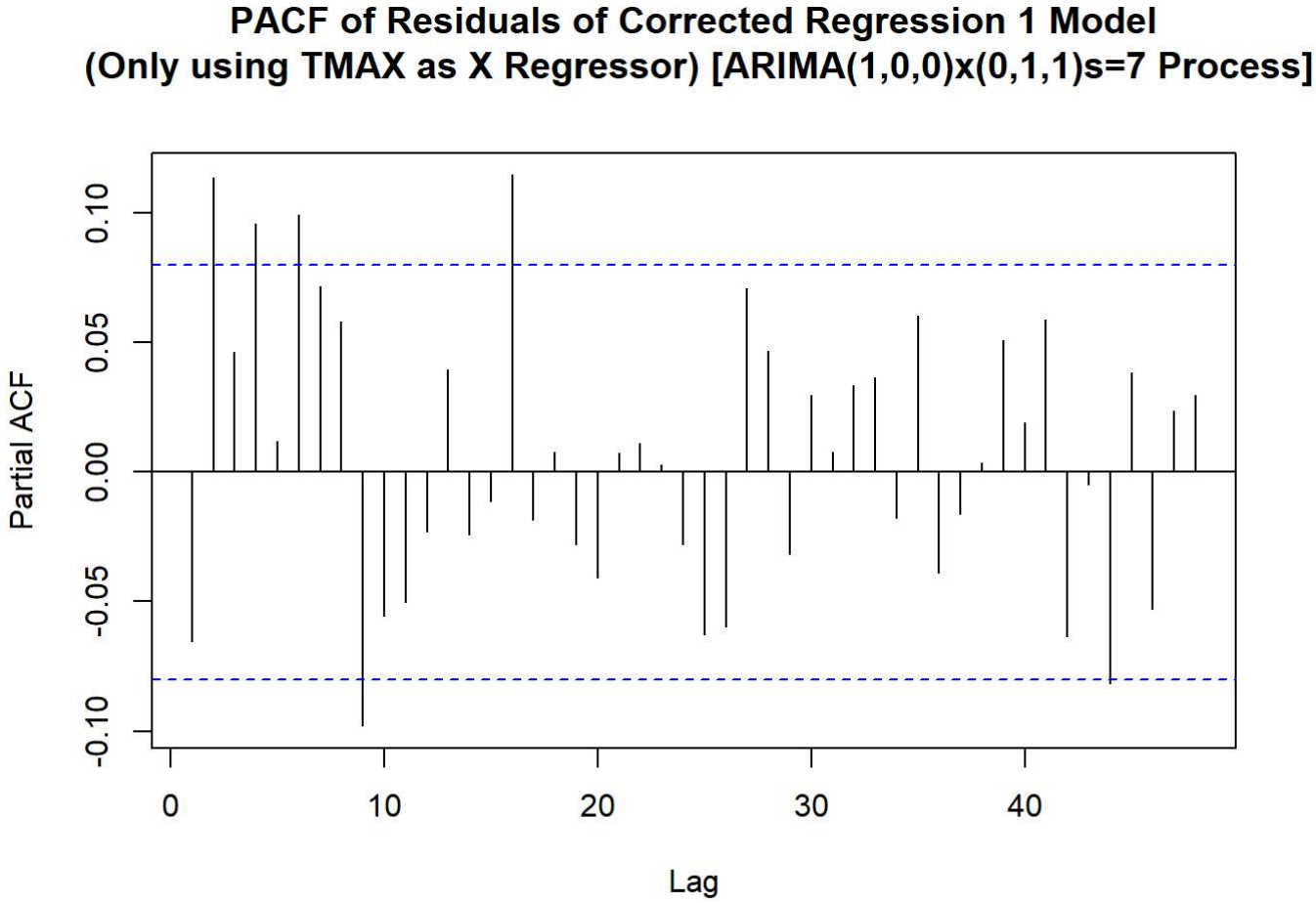
```
# ATTEMPTING TO IMPROVE CORRECTED MODEL (dropping non significant regressors)


regarima <- Arima(train_crash, order = c(1,0,0), seasonal=list(order = c(0,1,1),period=7), xreg=train_TMAX)
summary(regarima)
```

```
## Series: train_crash
## Regression with ARIMA(1,0,0)(0,1,1)[7] errors
##
## Coefficients:
##          ar1     sma1   xreg
##       0.3953  -0.9791  2.834
## s.e.  0.0382   0.0230  0.430
##
## sigma^2 = 4909:  log likelihood = -3370.97
## AIC=6749.93   AICc=6750   BIC=6767.47
##
## Training set error measures:
##                     ME    RMSE      MAE        MPE     MAPE      MASE
## Training set 4.220802 69.4784 50.81898 -0.6450736 8.490716 0.6150729
##                      ACF1
## Training set -0.06528981
```

```
acf(regarima$residuals,
    main = "ACF of Residuals of Corrected Regression 1 Model\n (Only using TMAX as X Regressor) [ARIMA(1,0,0)x(0,1,1)s=7 Pro
cess]",
    lag = 48)
```




```
pacf(regarima$residuals,
     main = "PACF of Residuals of Corrected Regression 1 Model\n (Only using TMAX as X Regressor) [ARIMA(1,0,0)x(0,1,1)s=7 P
rocess]",
     lag = 48)
```




```
Box.test(regarima$residuals, lag=24)
```

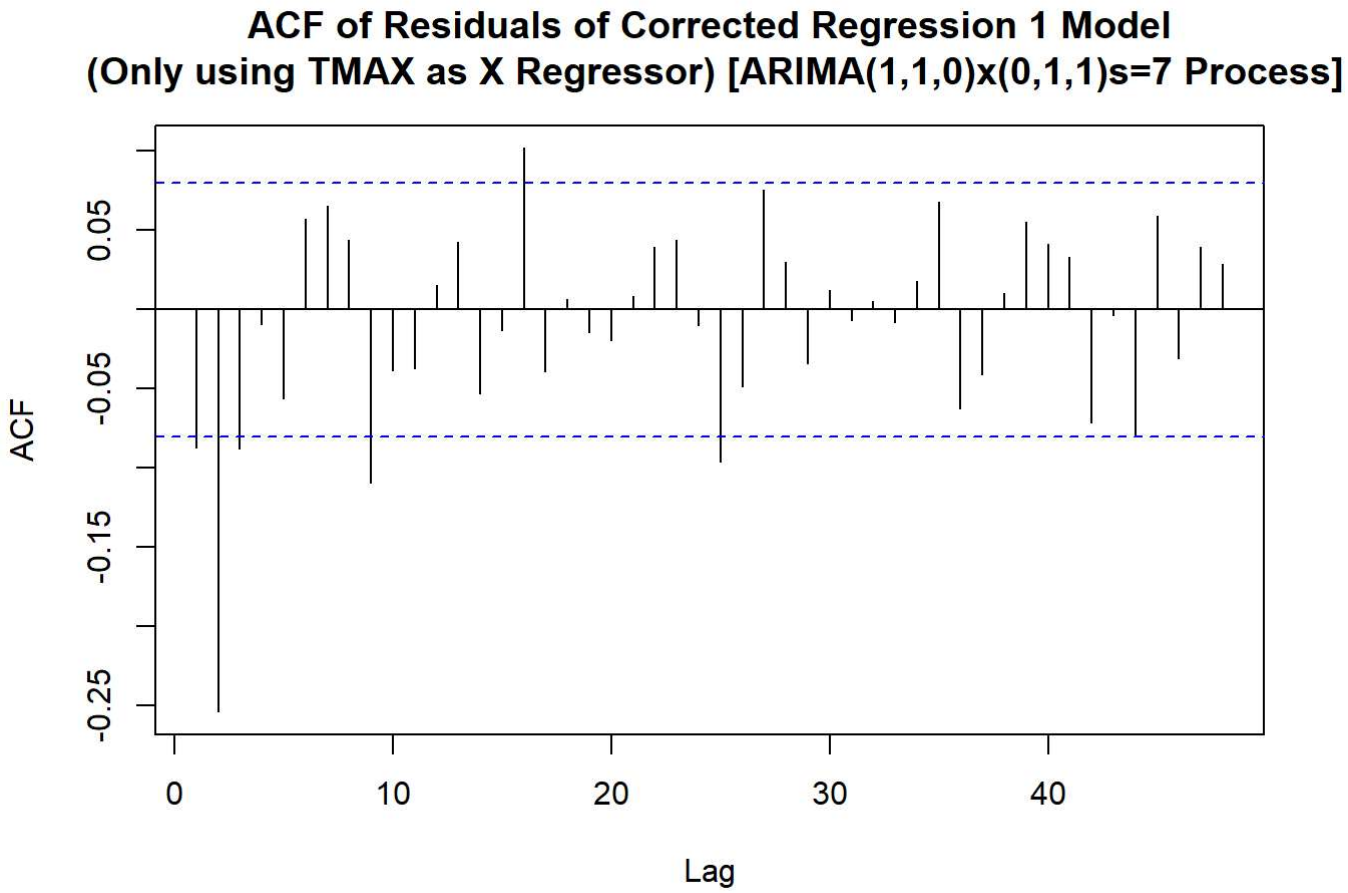
```
##
##  Box-Pierce test
##
## data:  regarima$residuals
## X-squared = 46.254, df = 24, p-value = 0.004126
```

```
# COMMENT: After changing the lag to 24 (thus changing the calculated value of the Box Pierce Test), we noticed that the p v
alue of model for this model was still extremely small, indicating that the residuals of the model were still correlated var
ious lags 1-24 and not yet white noise. Moreover, after more closely inspecting the ACF of the residuals, there appears to b
e extremely slow decay to zero at spurious lags throughout. Thus, differencing at the nonseasonal lags will likely be benefi
cial.

# COMMENT: However, it is worth mentioning that all coefficients in the model (phi, theta, and the x independent regressor)
are now statistically significant after dropping the redundant regressors.
```

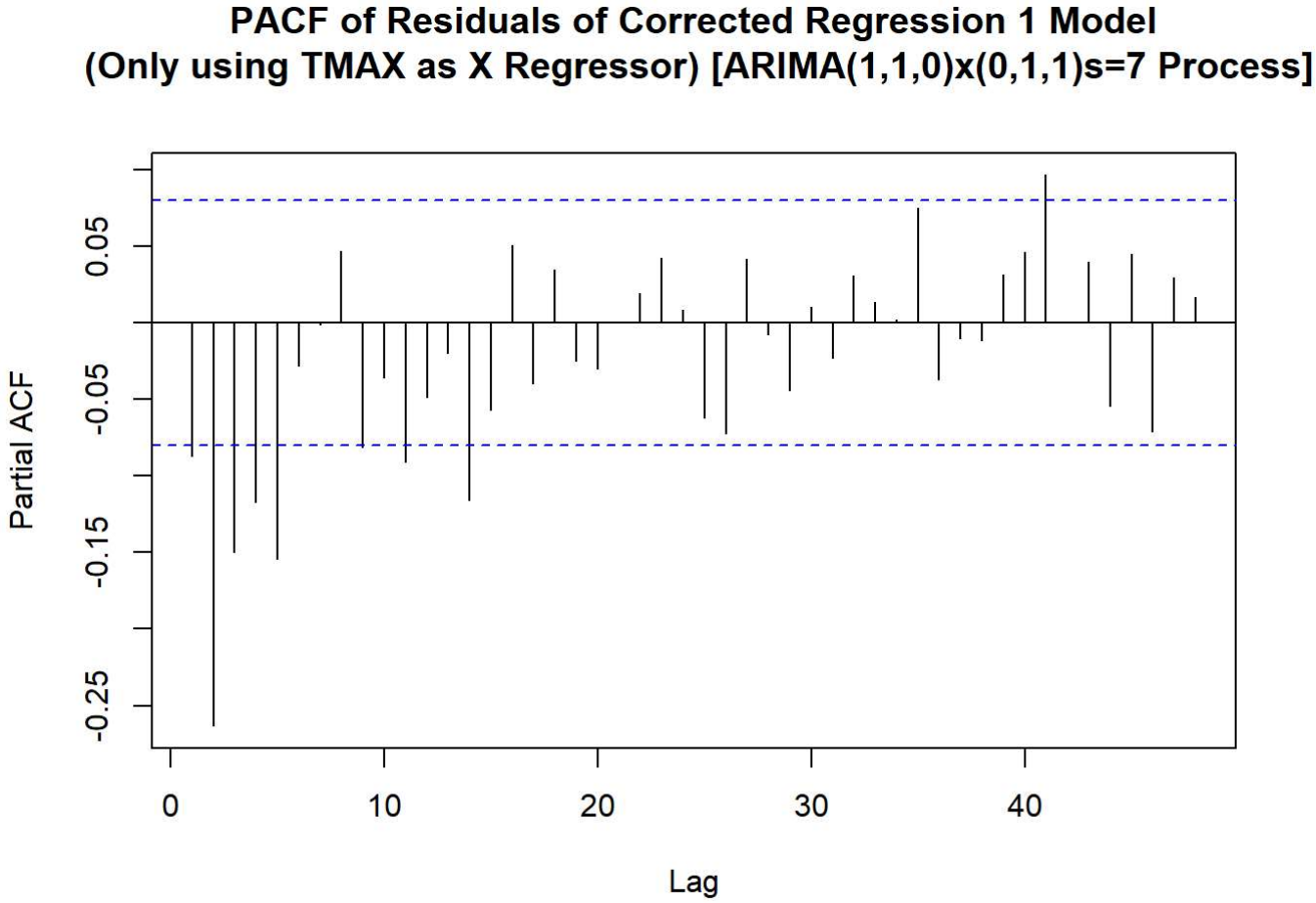
```
# ATTEMPTING TO IMPROVE CORRECTED MODEL (additionally differencing at the nonseasonal level)


regarima <- Arima(train_crash, order = c(1,1,0), seasonal=list(order = c(0,1,1),period=7), xreg=train_TMAX)
summary(regarima)
```

```
## Series: train_crash
## Regression with ARIMA(1,1,0)(0,1,1)[7] errors
##
## Coefficients:
##           ar1     sma1    xreg
##       -0.4171  -0.9714  2.3625
## s.e.   0.0375   0.0200  0.6733
##
## sigma^2 = 5800:  log likelihood = -3413.68
## AIC=6835.37   AICc=6835.44   BIC=6852.9
##
## Training set error measures:
##                    ME     RMSE      MAE       MPE     MAPE      MASE        ACF1
## Training set -2.0761 75.45867 54.42623 -1.464268 9.218818 0.6587322 -0.08683259
```

```
acf(regarima$residuals,
    main = "ACF of Residuals of Corrected Regression 1 Model\n (Only using TMAX as X Regressor) [ARIMA(1,1,0)x(0,1,1)s=7 Pro
cess]",
    lag = 48)
```




```
pacf(regarima$residuals,
     main = "PACF of Residuals of Corrected Regression 1 Model\n (Only using TMAX as X Regressor) [ARIMA(1,1,0)x(0,1,1)s=7 P
rocess]",
     lag = 48)
```




```
Box.test(regarima$residuals, lag = 24)
```

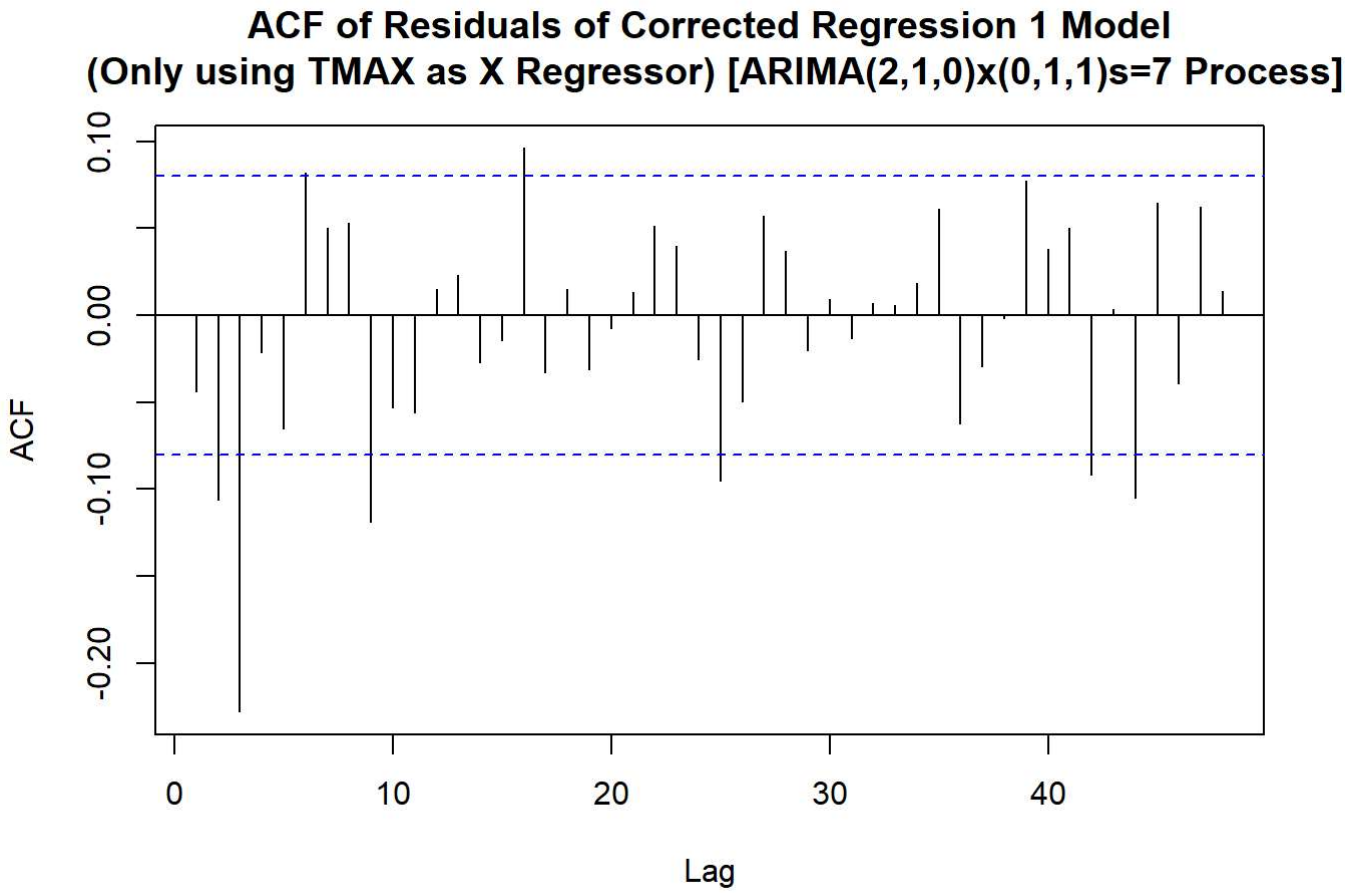
```
##
##  Box-Pierce test
##
## data:  regarima$residuals
## X-squared = 76.937, df = 24, p-value = 1.857e-07
```

```
# COMMENT: Box Pierce Test p value still extremely small. Corrective Model could be improved. ACF shows more gradual decay t
o zero at lags 1-24, indicating differencing at nonseasonal level makes residuals more stationary. However, may also be easi
er to interpret a gradual decay in the PACF and a chopping off at lag 2 in the ACF; thus, will leverage an AR(2) process to
the nonseasonal differenced series, rather than an AR(1) process.
```

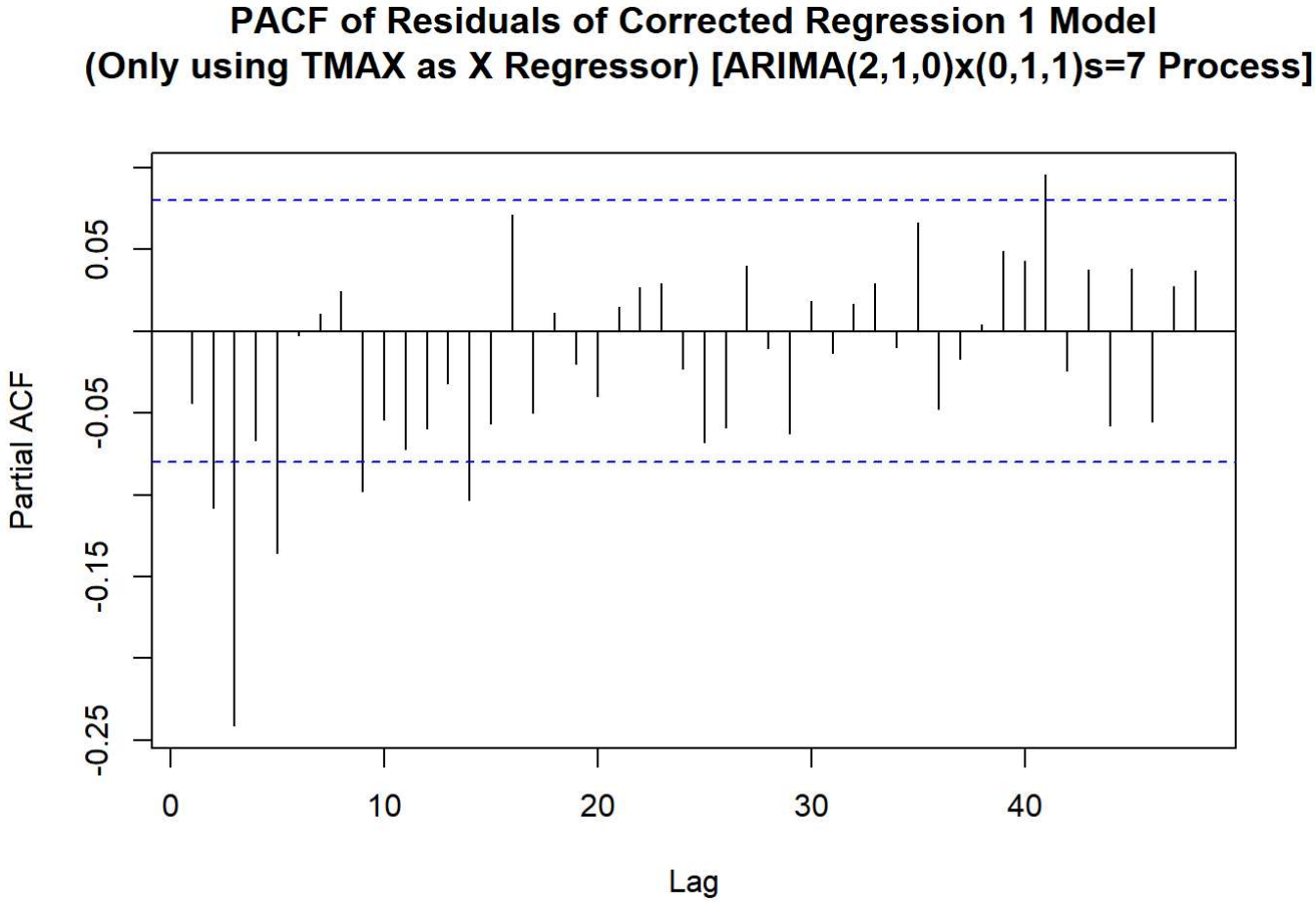
```
# ATTEMPTING TO IMPROVE CORRECTED MODEL (AR(2) at the nonseasonal level, after differencing)


regarima <- Arima(train_crash, order = c(2,1,0), seasonal=list(order = c(0,1,1),period=7), xreg=train_TMAX)
summary(regarima)
```

```
## Series: train_crash
## Regression with ARIMA(2,1,0)(0,1,1)[7] errors
##
## Coefficients:
##           ar1      ar2     sma1    xreg
##       -0.5076  -0.2149  -0.9727  2.0887
## s.e.   0.0404   0.0404   0.0207  0.6632
##
## sigma^2 = 5541:  log likelihood = -3399.88
## AIC=6809.75   AICc=6809.85   BIC=6831.67
##
## Training set error measures:
##                      ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -2.271454 73.69258 53.15925 -1.543334 9.023085 0.6433976
##                      ACF1
## Training set -0.04383061
```

```
acf(regarima$residuals,
    main = "ACF of Residuals of Corrected Regression 1 Model\n (Only using TMAX as X Regressor) [ARIMA(2,1,0)x(0,1,1)s=7 Pro
cess]",
    lag = 48)
```




```
pacf(regarima$residuals,
     main = "PACF of Residuals of Corrected Regression 1 Model\n (Only using TMAX as X Regressor) [ARIMA(2,1,0)x(0,1,1)s=7 P
rocess]",
     lag = 48)
```




```
Box.test(regarima$residuals, lag = 24)
```

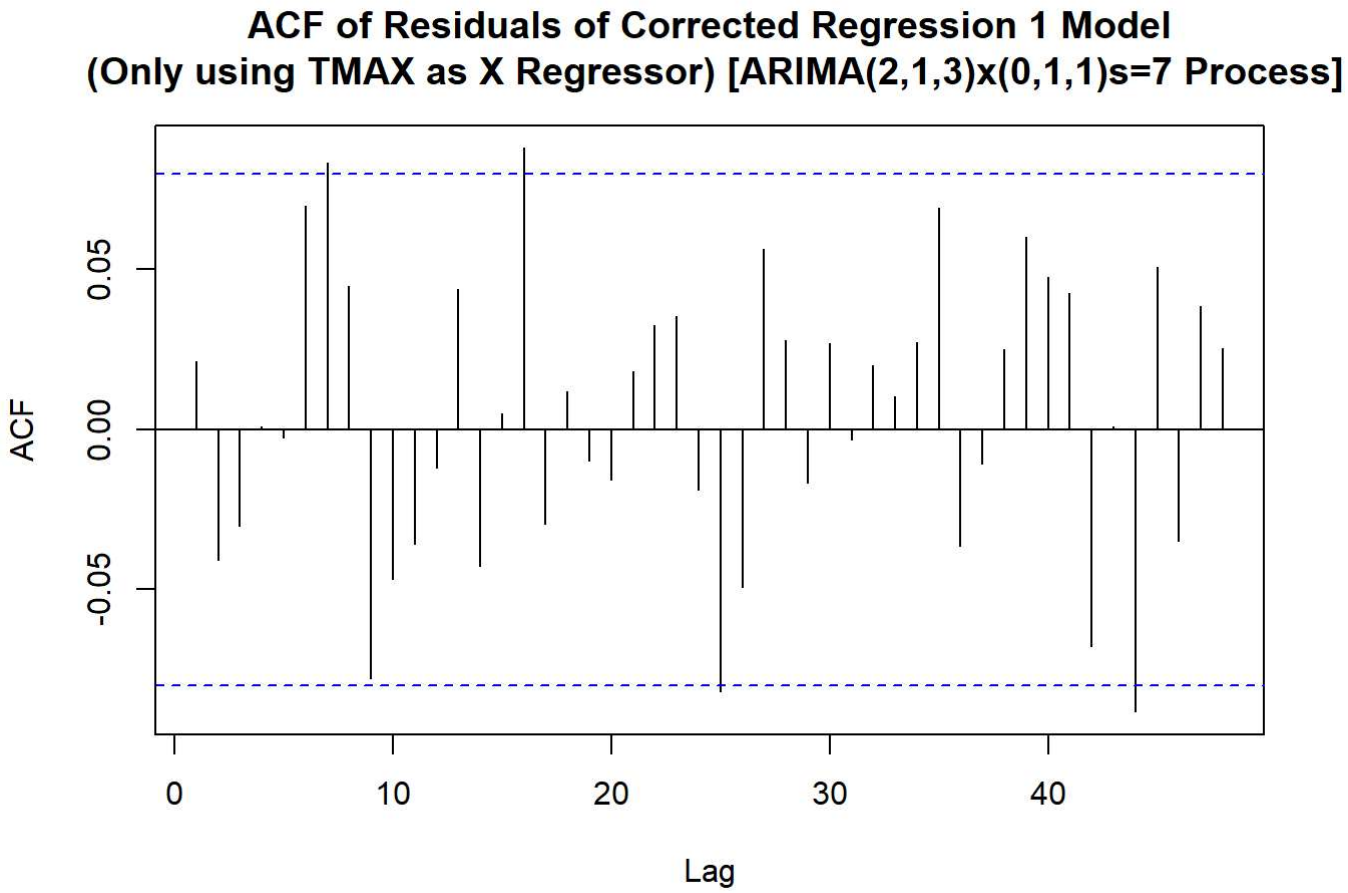
```
##
##  Box-Pierce test
##
## data:  regarima$residuals
## X-squared = 72.164, df = 24, p-value = 1.023e-06
```

```
# COMMENT: Box Pierce P value still incredibly small. Interpreting a gradual decay in residuals (lags 1-24) in ACF and chopp
ing off at lag 3. Thus, will leverage an ARMA(2,1,3) at the nonseasonal level, rather than an AR(2,1,0) process to see how t
his fit changes performance outputs and the calculated value of Box Pierce test.
```

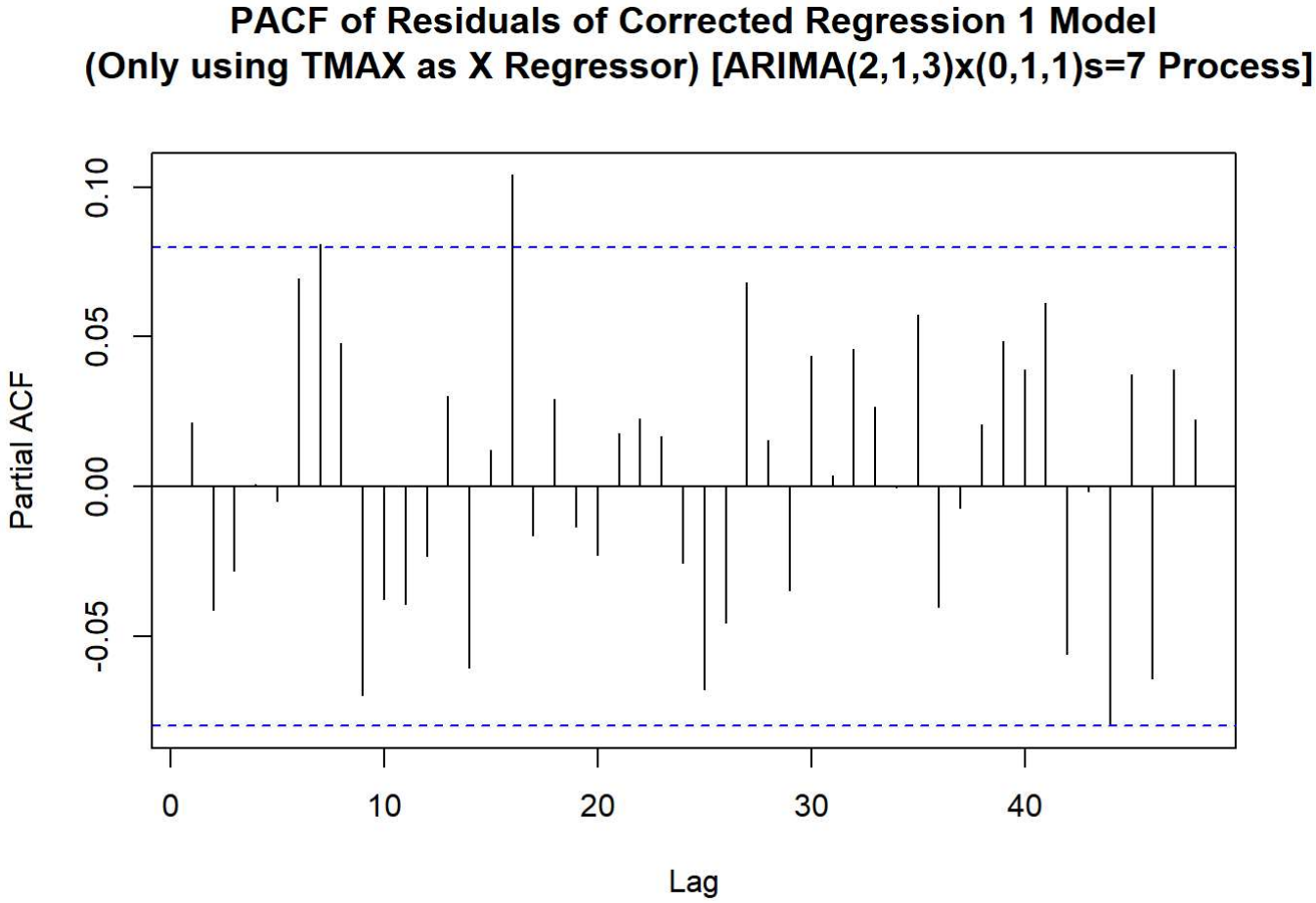
```
# ATTEMPTING TO IMPROVE CORRECTED MODEL (additionally differencing at the nonseasonal level)


regarima <- Arima(train_crash, order = c(2,1,3), seasonal=list(order = c(0,1,1),period=7), xreg=train_TMAX)
summary(regarima)
```

```
## Series: train_crash
## Regression with ARIMA(2,1,3)(0,1,1)[7] errors
##
## Coefficients:
##           ar1     ar2      ma1      ma2     ma3     sma1    xreg
##       -0.1671  0.6982  -0.5292  -0.8512  0.3803  -0.9669  2.5442
## s.e.   0.0867  0.0801   0.1097   0.0944  0.0997   0.0216  0.5133
##
## sigma^2 = 4786:  log likelihood = -3359.53
## AIC=6735.07   AICc=6735.31   BIC=6770.14
##
## Training set error measures:
##                      ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -5.027429 68.31024 49.57015 -2.122212 8.438616 0.5999579
##                     ACF1
## Training set 0.02133642
```

```
acf(regarima$residuals,
    main = "ACF of Residuals of Corrected Regression 1 Model\n (Only using TMAX as X Regressor) [ARIMA(2,1,3)x(0,1,1)s=7 Pro
cess]",
    lag = 48)
```




```
pacf(regarima$residuals,
     main = "PACF of Residuals of Corrected Regression 1 Model\n (Only using TMAX as X Regressor) [ARIMA(2,1,3)x(0,1,1)s=7 P
rocess]",
     lag = 48)
```




```
Box.test(regarima$residuals, lag = 24)
```

```
##
##  Box-Pierce test
##
## data:  regarima$residuals
## X-squared = 25.482, df = 24, p-value = 0.38
```

```
# COMMENT: MULTIPLICATIVE ARMA(2,1,3) x (0,1,1)s=7 appears to be best parsimonious model. At the 5% level of significance (u
sing the p-value derived from the Box-Pierce Test, for lags 1-24), we fail to reject the null hypothesis that the residuals
are white noise; thus, this model can be used for modeling the NYC Collision TS effectively.

# COMMENT: The Box Pierce Test returned a value of .38 and none of the lags of the residuals in ACF appear to lie outside of
the S.E. bounds, indicating that autocorrelations between residuals at the aforementioned lags have been removed. Moreover,
only one lag in the PACF (lag 16) lies outside of the S.E. bounds and trying to remove this partial correlation would uneces
sarily increase model complexity.

# COMMENT: All phi coefficients, theta coefficients, and independent regressor coefficients in model are statistically signi
ficant.
```

```
# PREDICTIVE PERFORMANCE OF MODEL FROM 4.2 ON HOLDOUT SAMPLE

x_test <- test_TMAX
colnames(x_test) <- colnames(train_TMAX) # Matching column names between training and testing regressors

# Use the forecast function on the pre-trained model
preds <- forecast(regarima, xreg = x_test, h = length(test_crash))

# Display Summary of Predictions
summary(preds)
```

```
##
## Forecast method: Regression with ARIMA(2,1,3)(0,1,1)[7] errors
##
## Model Information:
## Series: train_crash
## Regression with ARIMA(2,1,3)(0,1,1)[7] errors
##
## Coefficients:
##          ar1     ar2      ma1      ma2     ma3     sma1    xreg
##       -0.1671  0.6982  -0.5292  -0.8512  0.3803  -0.9669  2.5442
## s.e.   0.0867  0.0801   0.1097   0.0944  0.0997   0.0216  0.5133
##
## sigma^2 = 4786:  log likelihood = -3359.53
## AIC=6735.07   AICc=6735.31   BIC=6770.14
##
## Error measures:
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -5.027429 68.31024 49.57015 -2.122212 8.438616 0.5999579
##                    ACF1
## Training set 0.02133642
##
## Forecasts:
##     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 601       672.2053 583.3845 761.0260 536.3656 808.0449
## 602       697.4081 604.4922 790.3241 555.3054 839.5109
## 603       730.5041 634.5076 826.5006 583.6902 877.3180
## 604       611.8825 514.6674 709.0975 463.2049 760.5601
## 605       542.6064 444.3006 640.9121 392.2607 692.9520
## 606       644.7011 546.0205 743.3818 493.7821 795.6202
## 607       654.5841 555.4864 753.6817 503.0273 806.1408
## 608       670.6920 571.2905 770.0935 518.6705 822.7135
## 609       706.5902 606.9586 806.2218 554.2168 858.9635
## 610       721.1711 621.4777 820.8645 568.7032 873.6391
## 611       603.1848 503.3921 702.9774 450.5651 755.8044
## 612       529.4951 429.6836 629.3067 376.8465 682.1437
## 613       654.6188 554.7599 754.4777 501.8978 807.3398
## 614       677.6737 577.8091 777.5384 524.9440 830.4035
## 615       665.1443 565.1416 765.1471 512.2034 818.0853
## 616       691.6275 591.6115 791.6435 538.6662 844.5887
```

```
## 617       723.6253 623.5752 823.6753 570.6118 876.6387
## 618       605.0222 504.9683 705.0762 452.0029 758.0415
## 619       535.5658 435.4942 635.6374 382.5195 688.6121
## 620       647.8876 547.8151 747.9600 494.8400 800.9352
## 621       669.6748 569.5915 769.7581 516.6106 822.7390
## 622       679.8969 579.7505 780.0434 526.7362 833.0577
## 623       707.0401 606.8695 807.2107 553.8424 860.2379
## 624       736.9715 636.7921 837.1508 583.7603 890.1826
## 625       623.2935 523.1018 723.4852 470.0635 776.5235
## 626       547.7563 447.5618 647.9507 394.5221 700.9904
## 627       644.8866 544.6849 745.0883 491.6413 798.1319
## 628       669.2514 569.0485 769.4543 516.0042 822.4985
## 629       675.2431 574.9558 775.5305 521.8669 828.6194
## 630       705.4405 605.1413 805.7397 552.0462 858.8349
## 631       739.3373 639.0213 839.6533 585.9172 892.7574
## 632       630.1743 529.8535 730.4950 476.7469 783.6016
## 633       562.7564 462.4275 663.0853 409.3166 716.1963
## 634       665.8340 565.5030 766.1651 512.3909 819.2772
## 635       678.3268 577.9906 778.6631 524.8758 831.7779
## 636       686.5860 586.1791 786.9928 533.0270 840.1450
## 637       704.5320 604.1072 804.9568 550.9455 858.1185
## 638       719.4408 619.0041 819.8774 565.8362 873.0453
## 639       600.9728 500.5275 701.4180 447.3550 754.5906
## 640       530.1855 429.7355 630.6355 376.5605 683.8106
## 641       645.2387 544.7838 745.6936 491.6062 798.8713
## 642       660.0437 559.5861 760.5013 506.4071 813.6803
## 643       670.0239 569.4871 770.5607 516.2661 823.7817
## 644       710.1988 609.6474 810.7503 556.4187 863.9790
## 645       739.7380 639.1715 840.3045 585.9349 893.5412
## 646       622.1886 521.6157 722.7616 468.3756 776.0017
## 647       543.3318 442.7520 643.9115 389.5083 697.1553
## 648       650.9598 550.3767 751.5429 497.1312 804.7884
## 649       674.4389 573.8516 775.0262 520.6039 828.2739
## 650       672.9676 572.3049 773.6303 519.0173 826.9179
## 651       683.8667 583.1870 784.5464 529.8904 837.8430
## 652       718.5297 617.8365 819.2230 564.5327 872.5268
## 653       609.8329 509.1317 710.5341 455.8237 763.8421
## 654       539.1853 438.4783 639.8923 385.1672 693.2034
## 655       640.1163 539.4049 740.8277 486.0915 794.1410
```

```
## 656       642.0647 541.3499 742.7795 488.0347 796.0947
## 657       666.1826 565.3891 766.9762 512.0322 820.3330
## 658       693.9924 593.1828 794.8020 539.8174 848.1674
## 659       727.5086 626.6841 828.3331 573.3108 881.7063
## 660       618.6961 517.8643 719.5279 464.4871 772.9051
## 661       542.3036 441.4651 643.1420 388.0845 696.5226
## 662       647.2067 546.3643 748.0491 492.9816 801.4318
## 663       665.0204 564.1740 765.8668 510.7892 819.2516
## 664       659.2868 558.3625 760.2111 504.9364 813.6372
## 665       675.7251 574.7837 776.6665 521.3486 830.1017
## 666       712.2227 611.2667 813.1787 557.8239 866.6216
## 667       606.2751 505.3111 707.2390 451.8641 760.6861
## 668       529.8214 428.8511 630.7917 375.4007 684.2422
## 669       629.1837 528.2090 730.1583 474.7563 783.6110
## 670       641.3480 540.3696 742.3264 486.9148 795.7812
## 671       648.3836 547.3256 749.4416 493.8287 802.9385
## 672       695.8171 594.7421 796.8921 541.2363 850.3979
## 673       729.3026 628.2124 830.3928 574.6985 883.9068
## 674       587.9524 486.8543 689.0504 433.3363 742.5685
## 675       521.7106 420.6058 622.8153 367.0842 676.3370
## 676       636.5603 535.4513 737.6693 481.9275 791.1932
## 677       630.1816 529.0685 731.2947 475.5425 784.8207
## 678       634.3911 533.1982 735.5839 479.6299 789.1522
## 679       672.1819 570.9715 773.3923 517.3940 826.9698
## 680       697.2480 596.0223 798.4737 542.4367 852.0594
## 681       562.7888 461.5550 664.0227 407.9650 717.6126
## 682       500.8522 399.6117 602.0928 346.0182 655.6863
## 683       609.8687 508.6237 711.1137 455.0279 764.7096
## 684       629.1696 527.9206 730.4187 474.3226 784.0167
## 685       637.7200 536.3901 739.0499 482.7493 792.6907
## 686       675.2419 573.8944 776.5894 520.2442 830.2396
## 687       691.6711 590.3078 793.0344 536.6493 846.6929
## 688       588.4935 487.1220 689.8650 433.4592 743.5278
## 689       519.4445 418.0661 620.8229 364.3996 674.4894
## 690       610.1320 508.7491 711.5149 455.0802 765.1838
## 691       643.4359 542.0488 744.8230 488.3777 798.4941
## 692       644.8534 543.3849 746.3220 489.6707 800.0361
## 693       655.6693 554.1827 757.1558 500.4590 810.8795
## 694       696.2609 594.7584 797.7634 541.0262 851.4956
```

```
## 695       590.2917 488.7808 691.8027 435.0442 745.5393
## 696       508.5151 406.9971 610.0330 353.2568 663.7734
## 697       616.2550 514.7324 717.7775 460.9896 771.5203
## 698       638.1043 536.5774 739.6311 482.8324 793.3761
## 699       655.0467 553.4377 756.6558 499.6491 810.4443
## 700       670.1830 568.5557 771.8103 514.7575 825.6085
## 701       699.3300 597.6864 800.9736 543.8795 854.7805
## 702       577.8370 476.1849 679.4891 422.3735 733.3005
## 703       504.4600 402.8007 606.1194 348.9855 659.9345
## 704       616.5216 514.8576 718.1856 461.0399 772.0033
## 705       648.2966 546.6282 749.9650 492.8082 803.7850
## 706       649.7163 547.9651 751.4676 494.1012 805.3315
## 707       657.7315 555.9617 759.5013 502.0880 813.3750
## 708       681.2787 579.4923 783.0651 525.6098 836.9476
## 709       559.7880 457.9929 661.5831 404.1058 715.4702
## 710       492.0062 390.2038 593.8086 336.3128 647.6996
## 711       602.5432 500.7360 704.3505 446.8425 758.2440
## 712       633.0443 531.2326 734.8560 477.3367 788.7518
## 713       610.5500 508.6547 712.4453 454.7145 766.3854
## 714       640.9527 539.0386 742.8669 485.0885 796.8170
## 715       666.2824 564.3513 768.2134 510.3923 822.1725
## 716       565.6530 463.7131 667.5929 409.7494 721.5566
## 717       490.7486 388.8012 592.6959 334.8336 646.6636
## 718       611.2070 509.2547 713.1592 455.2845 767.1294
## 719       640.4369 538.4802 742.3937 484.5075 796.3664
## 720       639.0588 537.0177 741.0999 483.0003 795.1172
## 721       651.1440 549.0838 753.2042 495.0563 807.2316
## 722       697.3354 595.2580 799.4129 541.2215 853.4494
## 723       578.6428 476.5564 680.7293 422.5151 734.7705
## 724       495.3418 393.2478 597.4358 339.2025 651.4811
## 725       598.7545 496.6555 700.8535 442.6076 754.9014
## 726       602.7960 500.6924 704.8996 446.6420 758.9500
## 727       601.4184 499.2297 703.6071 445.1343 757.7025
## 728       621.8990 519.6910 724.1071 465.5853 778.2128
## 729       658.1685 555.9429 760.3940 501.8280 814.5090
## 730       543.8006 441.5659 646.0353 387.4461 700.1551
## 731       467.6238 365.3814 569.8662 311.2575 623.9901
```

```
# Extract Forecast Values (Point Forecasts)
holdfit <- preds$mean

# Accuracy Measures for Holdout Sample
accuracy(holdfit, test_crash)
```

```
##                ME     RMSE      MAE       MPE     MAPE
## Test set 12.82913 80.65377 59.31423 0.4525601 9.308386
```

```
# TABLE of Predictive Performance Metrics FOR 4.2

#training metrics
acc_train<- accuracy(forecast(regarima, xreg = train_TMAX, h=length(train_crash)), train_crash)

#manually deriving Training R-Squared Metrics
REG_Train_R2 <- 1 - sum((train_crash - fitted(regarima))^2) / sum((train_crash - mean(train_crash))^2)

#test metrics
acc<- accuracy(holdfit, test_crash)

#manually deriving Test R-Squared Metric
REG_Test_R2 <- 1 - sum((test_crash - holdfit)^2) / sum((test_crash - mean(test_crash))^2)


Model_Metric_df <- data.frame(Metric = c("MAPE", "RMSE", "MAE", "R2"),
                Training_set = c(acc_train[1,5], acc_train[1,2], acc_train[1,3], REG_Train_R2),
                Holdout_set = c(acc[,5], acc[,2], acc[,3], REG_Test_R2))
Model_Metric_df
```

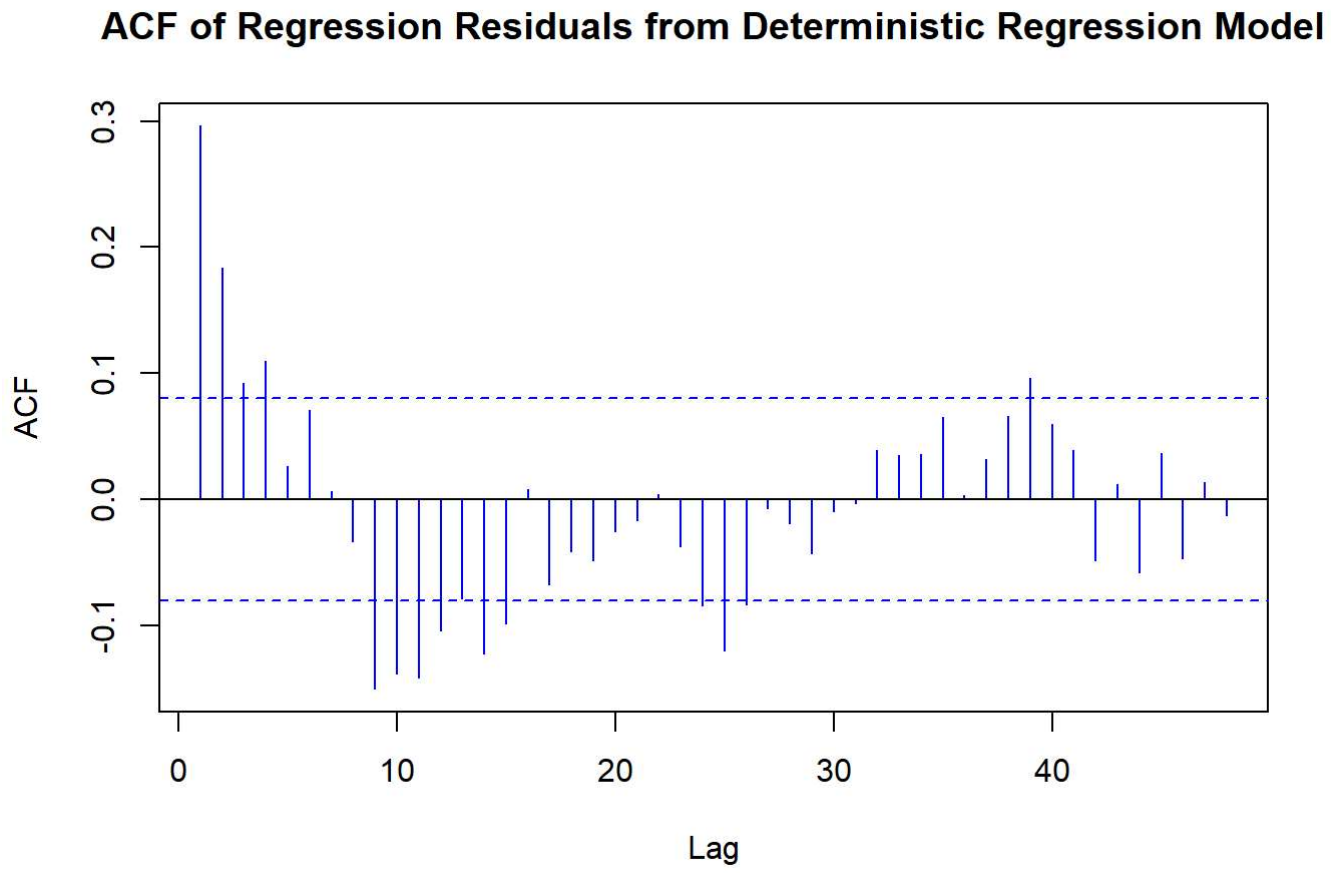
```
##   Metric Training_set Holdout_set
## 1   MAPE    8.4386159   9.3083864
## 2   RMSE   68.3102420  80.6537691
## 3    MAE   49.5701460  59.3142298
## 4     R2    0.5282698   0.3435799
```

SECTION 4.3) You will be analyzing the residuals of the deterministic time series models from SECTION 2.3. If these residuals behave like stationary time series then you can model them using an AR, MA, or ARMA process. If they do not exhibit stationary behavior then you can justify that. In this case you will not be able to model these nonstationary residuals.

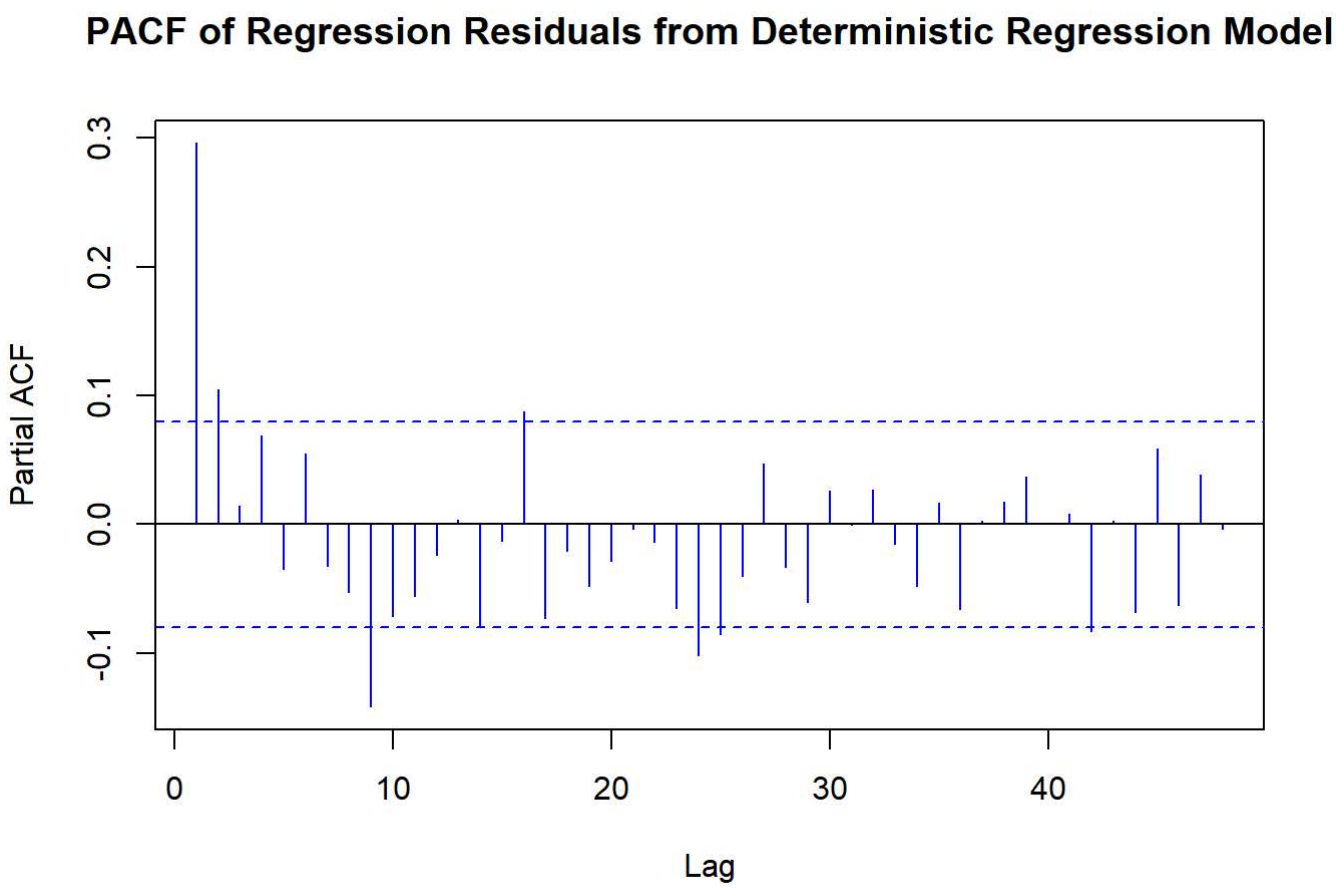
```
# ANALYZING RESIDUALS OF DETERMINISTIC TS MODEL CONTAINING MONTHLY AND DAILY SEASONAL DUMMY VARIABLES

deterministic_model <-lm(train_crash~m3_train+m4_train+m5_train+m6_train+
         m7_train+m8_train+m9_train+m10_train+m11_train+m12_train+d2_train+d3_train+d4_train+d5_train+d6_train+
         d7_train) #Seasonal dummy model with each day of week and month as dummy. Reference month = January. Reference day
= Friday.

summary(deterministic_model) #Show what the model looks like, we see that all variables are significant EXCEPT the dummy var
iable for February (NYC Collisions in February are not statistically different than those in January). Thus, we dropped it t
o have a more parsimonious model. Thus the intercept now effectively represents January, February, and Friday collisions in
NYC.
```

```
##
## Call:
## lm(formula = train_crash ~ m3_train + m4_train + m5_train + m6_train +
##     m7_train + m8_train + m9_train + m10_train + m11_train +
##     m12_train + d2_train + d3_train + d4_train + d5_train + d6_train +
##     d7_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -289.33  -34.32    3.56   41.86  369.14
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  643.463      9.711  66.264  < 2e-16 ***
## m3_train      42.429     11.224   3.780 0.000173 ***
## m4_train      40.141     11.342   3.539 0.000433 ***
## m5_train      96.821     11.219   8.630  < 2e-16 ***
## m6_train     129.540     11.345  11.419  < 2e-16 ***
## m7_train      73.433     11.219   6.546 1.30e-10 ***
## m8_train      60.194     11.830   5.088 4.88e-07 ***
## m9_train      77.847     14.635   5.319 1.49e-07 ***
## m10_train     71.432     14.443   4.946 9.93e-07 ***
## m11_train     76.202     14.635   5.207 2.67e-07 ***
## m12_train     49.727     14.446   3.442 0.000618 ***
## d2_train    -115.607     10.925 -10.582  < 2e-16 ***
## d3_train    -191.339     10.926 -17.512  < 2e-16 ***
## d4_train     -79.949     10.929  -7.316 8.52e-13 ***
## d5_train     -53.530     10.930  -4.897 1.26e-06 ***
## d6_train     -51.589     10.959  -4.707 3.14e-06 ***
## d7_train     -32.446     10.955  -2.962 0.003184 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 71.61 on 583 degrees of freedom
## Multiple R-squared:  0.4962, Adjusted R-squared:  0.4824
## F-statistic: 35.89 on 16 and 583 DF,  p-value: < 2.2e-16
```

```
# NOTE: Tried to add trend component but was NOT significant; thus, it was removed.

deterministic_res=residuals(deterministic_model) #Residuals of seasonal dummy variable model

acf(deterministic_res,main="ACF of Regression Residuals from Deterministic Regression Model",col="blue", lag=48) #We can see that the residuals still illustrate rather gradual decay to 0 in the ACF from lags 1-48; thus, autocorrelation of residuals do appear stationary.
```




```
pacf(deterministic_res, main = "PACF of Regression Residuals from Deterministic Regression Model", col="blue", lag = 48)
```

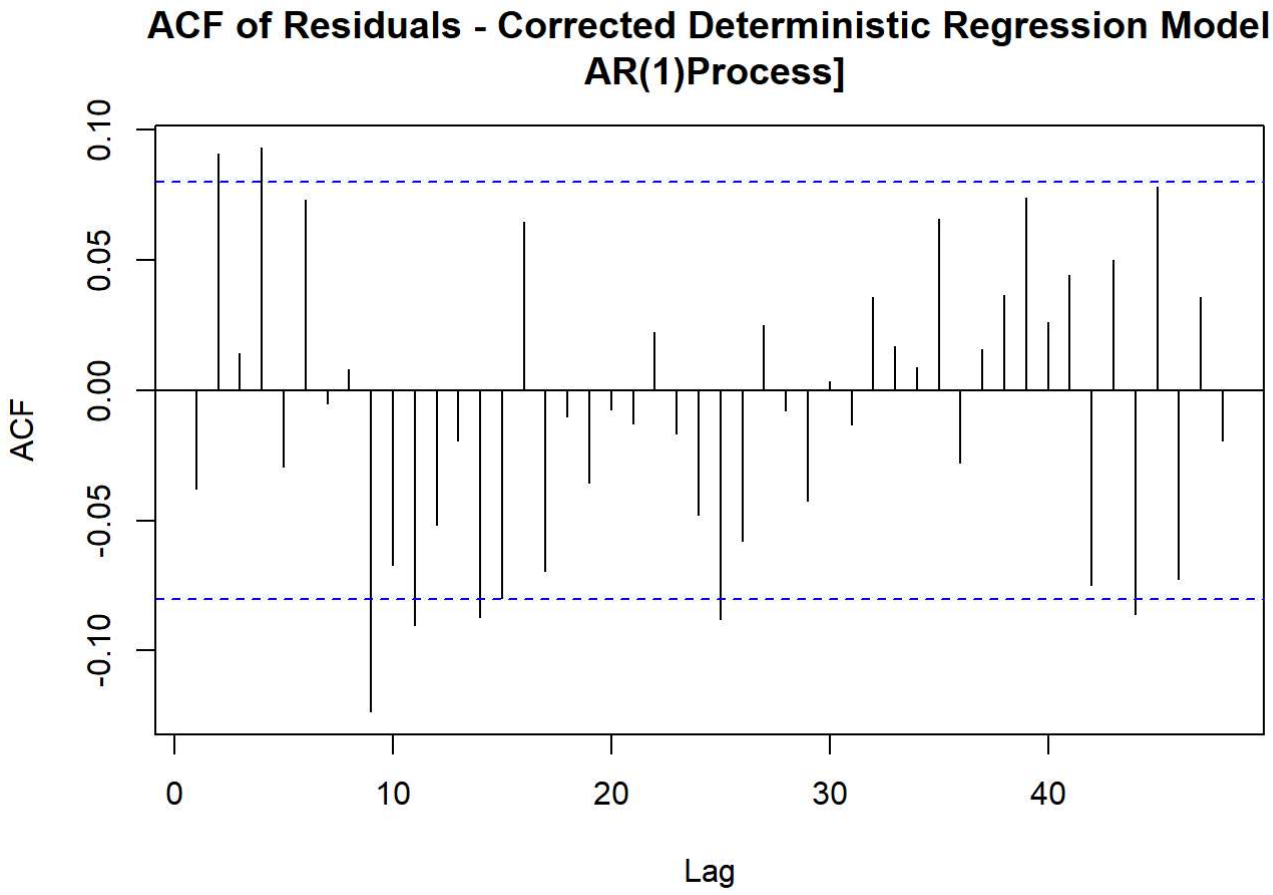


#COMMENT: Seems to be chopping off after lag 1 in PACF; thus, because (initial interpretation) decay in ACF, will try a AR (1) process. HOWEVER, because the PACF has a handfdul of spurious lags that lie slightly outside of the SE bounds, the under lying residuals of the TS may be nonstationary, as they do not decay quickly to 0 in the PACF. Will still attempt to model t hem by inducing stationarity if needed.

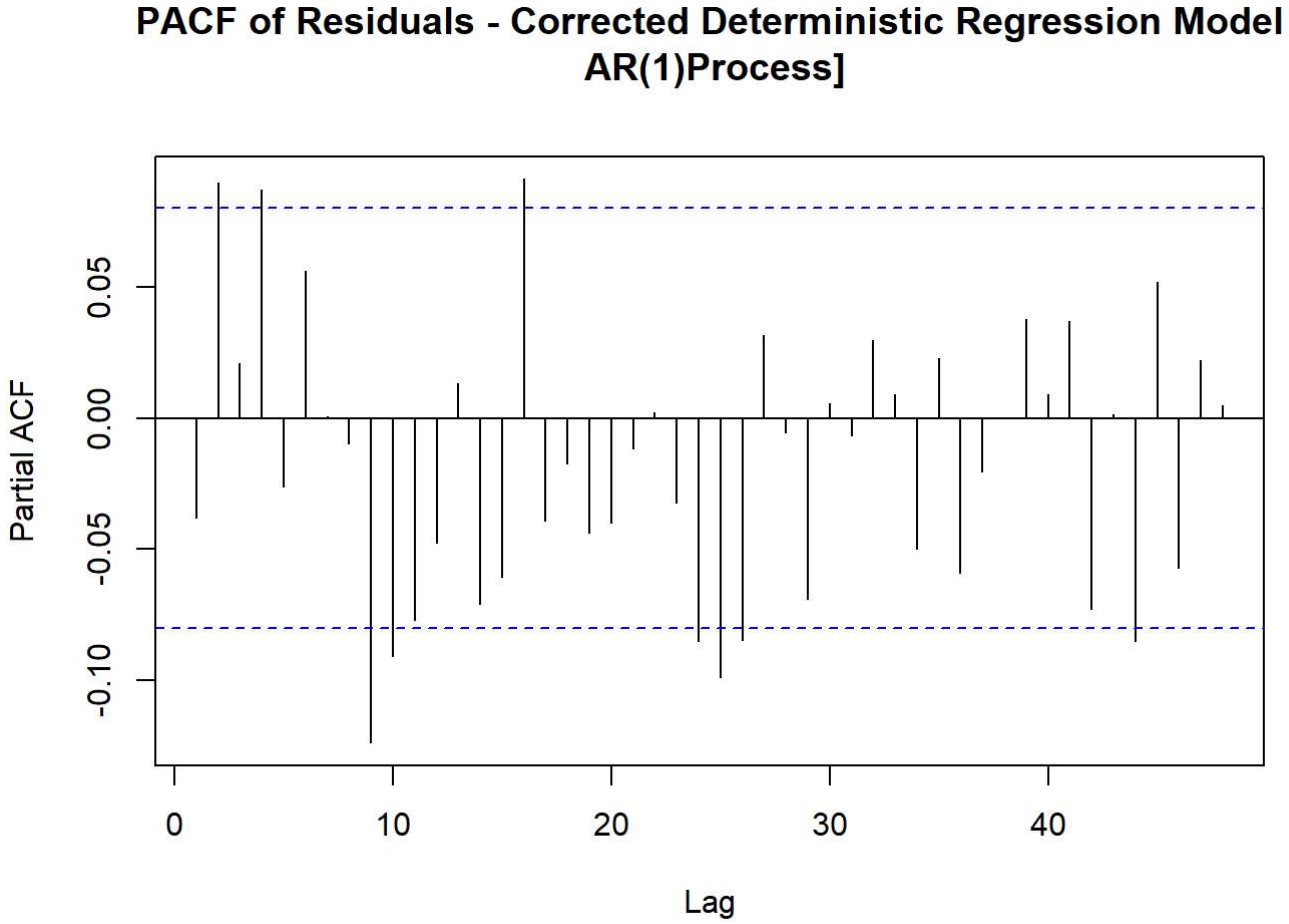
```
# Deterministic TS Corrected Model

x <- cbind(m3_train,m4_train,m5_train,m6_train,
         m7_train,m8_train,m9_train,m10_train,m11_train,m12_train,d2_train,d3_train,d4_train,d5_train,d6_train,
         d7_train)



ar_fit <- Arima(train_crash, order = c(1,0,0), xreg=x)
summary(ar_fit)
```

```
## Series: train_crash
## Regression with ARIMA(1,0,0) errors
##
## Coefficients:
##          ar1  intercept  m3_train  m4_train  m5_train  m6_train  m7_train
##       0.3029   643.8808   41.5841   40.3710   96.6261  128.0821   73.6622
## s.e.  0.0394    10.9461   14.8067   15.0527   14.8988   15.0632   14.8980
##       m8_train  m9_train  m10_train  m11_train  m12_train   d2_train   d3_train
##        60.2201   80.4448    67.0454    76.4678    46.9022  -115.5606  -191.3358
## s.e.   15.6892   19.4046    19.1831    19.3985    19.0924     9.0038    10.2685
##       d4_train  d5_train  d6_train  d7_train
##       -79.9288  -53.6088  -51.7358  -33.2712
## s.e.   10.5962   10.6002   10.3008    9.0346
##
## sigma^2 = 4676:  log likelihood = -3377.33
## AIC=6792.66   AICc=6793.97   BIC=6876.2
##
## Training set error measures:
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 0.1443457 67.34715 48.65946 -1.297419 8.277892 0.5889357
##                     ACF1
## Training set -0.03787381
```

```
acf(ar_fit$residuals,
    main = "ACF of Residuals - Corrected Deterministic Regression Model\n AR(1)Process]",
    lag=48)
```




```
pacf(ar_fit$residuals,
     main = "PACF of Residuals - Corrected Deterministic Regression Model\n AR(1)Process]",
     lag=48)
```




```
Box.test(ar_fit$residuals, lag=24)
```

```
##
##  Box-Pierce test
##
## data:  ar_fit$residuals
## X-squared = 50.07, df = 24, p-value = 0.001387
```

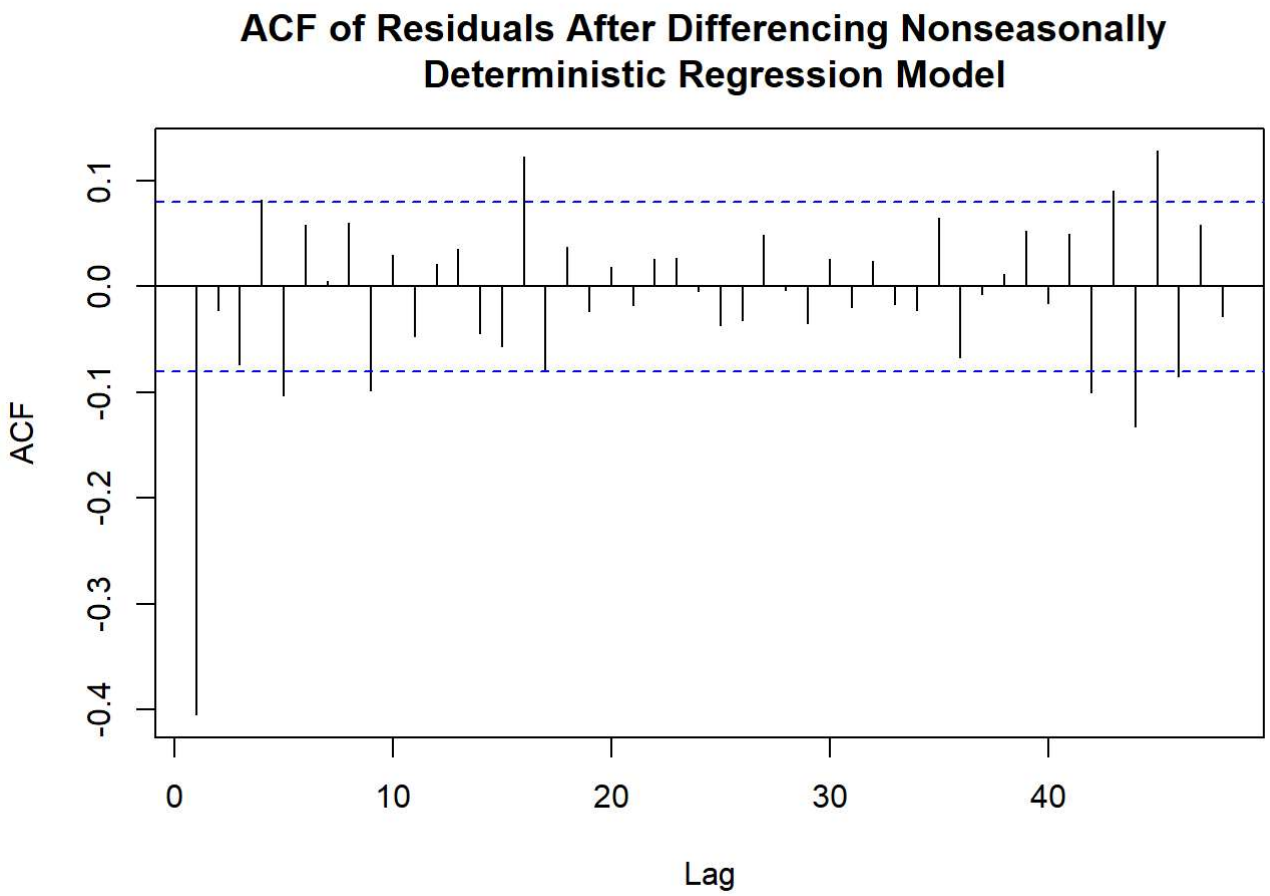
#COMMENT: All x independent regressors (including the intercept), as well as the phi(1) coefficient, are statistically significant and useful in model. Box Pierce Test p value still incredibly small, indicating that there is still underlying correlation between residuals at lags 1-24.

 # COMMENT: Because the PACF still exhibits some spurious residuals lying outside of the SE bounds between lags 1-48 (such as that at lag 44), the residuals are likely not yet stationary due to the slightly hidden trend/cyclical structure of the underlying TS. Will try to induce stationarity by differencing to see if ACF/PACF is made easier to interpret and more closely exhibits stationarity.

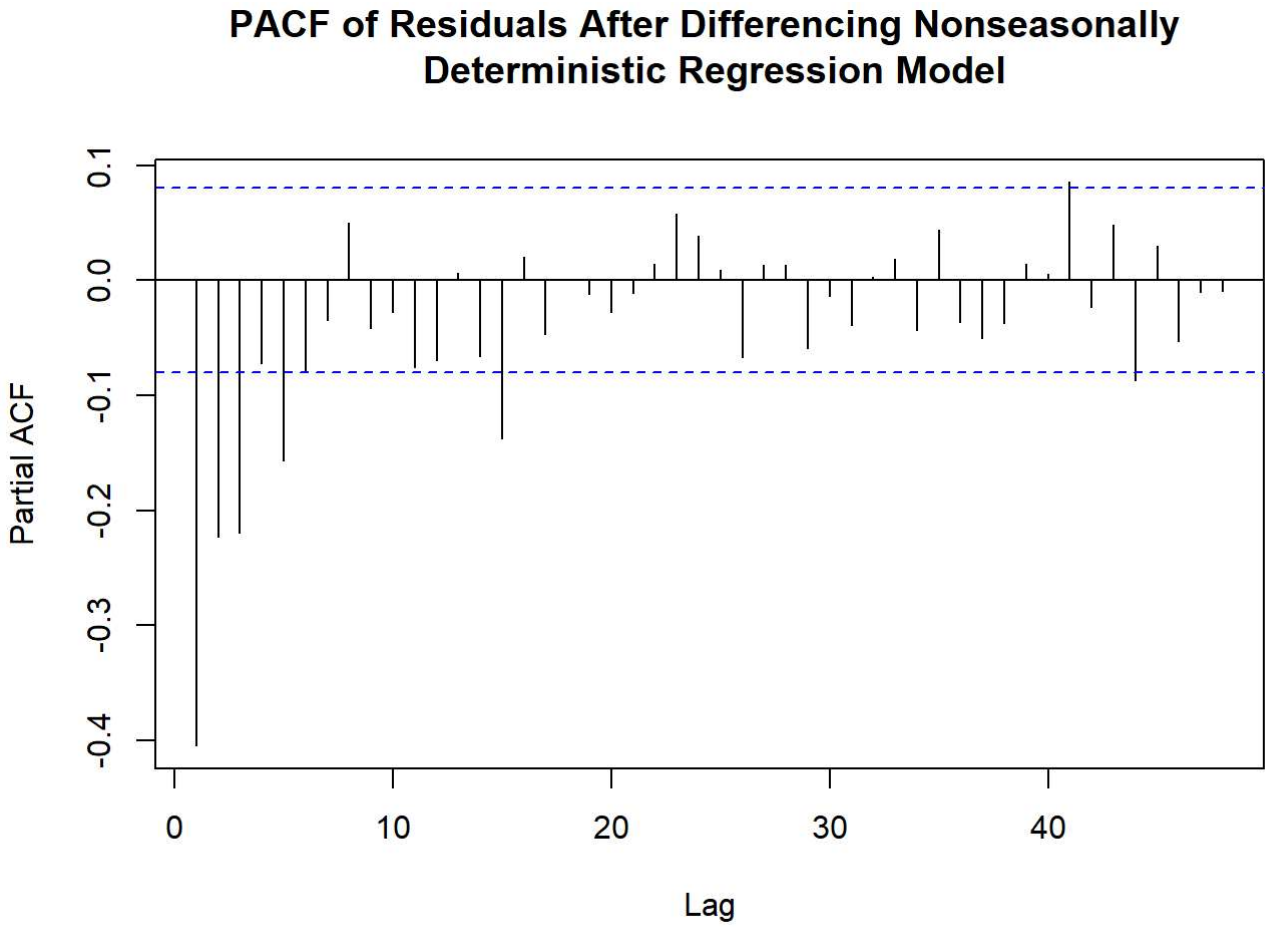
```
# Deterministic TS Corrected Model (inducing stationarity before interpretation)

x <- cbind(m3_train,m4_train,m5_train,m6_train,
         m7_train,m8_train,m9_train,m10_train,m11_train,m12_train,d2_train,d3_train,d4_train,d5_train,d6_train,
         d7_train)



ar_fit <- Arima(train_crash, order = c(0,1,0), xreg=x)
summary(ar_fit)
```

```
## Series: train_crash
## Regression with ARIMA(0,1,0) errors
##
## Coefficients:
##       m3_train  m4_train  m5_train  m6_train  m7_train  m8_train  m9_train
##         6.4204   -1.5726   25.2723   21.1002  -11.8928  -28.2643  -18.0418
## s.e.   56.7873   77.4812   91.3415  101.2998  108.5487  113.3907  116.9329
##       m10_train  m11_train  m12_train   d2_train   d3_train  d4_train  d5_train
##       -179.4701  -113.0628  -112.8403  -114.6299  -192.5284  -81.3780  -55.8436
## s.e.   113.4152   101.5588    77.6931     8.3287    10.7394   11.7828   11.8005
##       d6_train  d7_train
##       -53.2769  -34.5576
## s.e.   10.7777    8.3276
##
## sigma^2 = 7001:  log likelihood = -3493.57
## AIC=7021.14   AICc=7022.2   BIC=7095.86
##
## Training set error measures:
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 0.6258348 82.48102 57.82677 -0.8769879 9.729534 0.6998896
##                     ACF1
## Training set -0.4045769
```

```
acf(ar_fit$residuals,
    main = "ACF of Residuals After Differencing Nonseasonally \n Deterministic Regression Model",
    lag=48)
```




```
pacf(ar_fit$residuals,
     main = "PACF of Residuals After Differencing Nonseasonally \n Deterministic Regression Model",
     lag=48)
```




```
Box.test(ar_fit$residuals, lag=24)
```

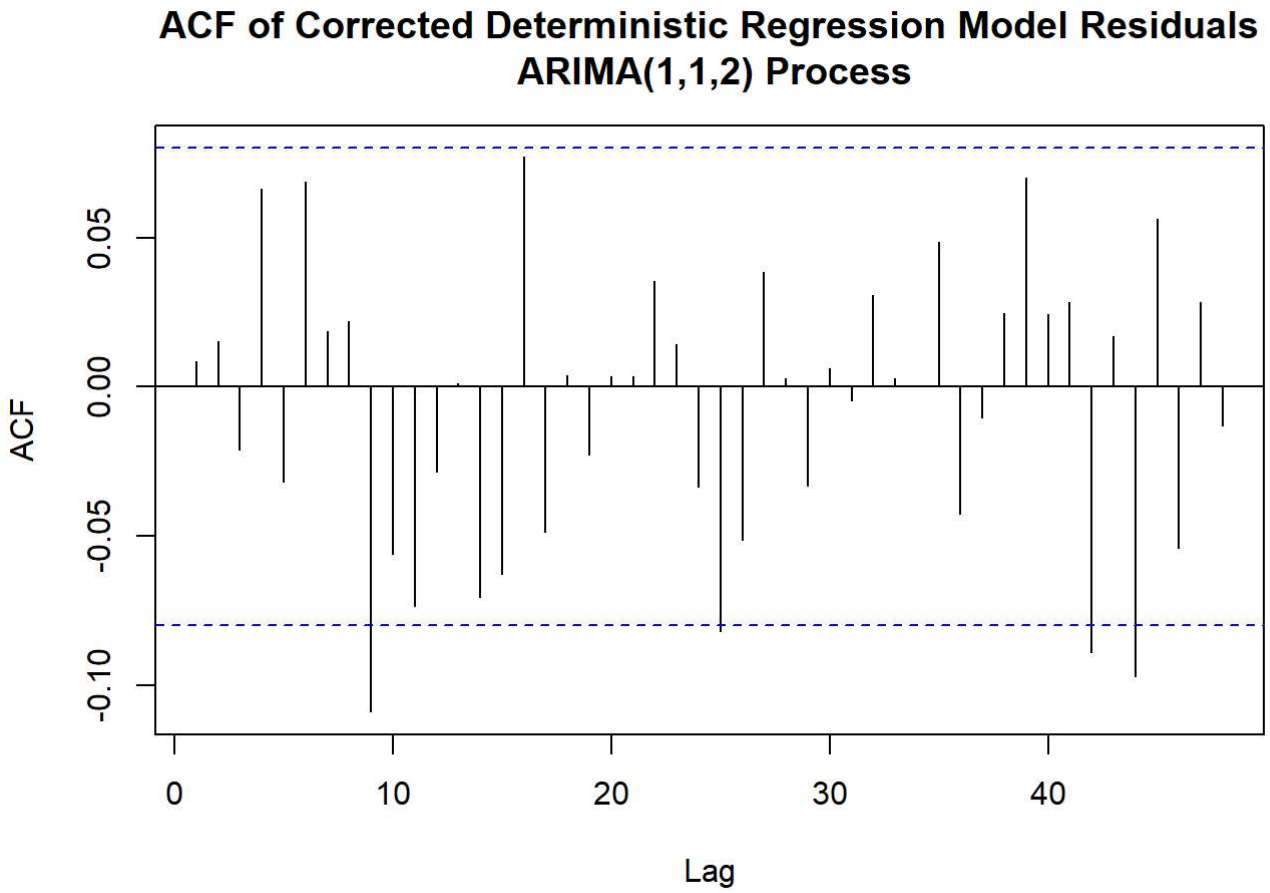
```
##
##  Box-Pierce test
##
## data:  ar_fit$residuals
## X-squared = 143.3, df = 24, p-value < 2.2e-16
```

#COMMENT: After differencing, ACF appears to chop off at lag 1. PACF gradual decay to 0. Differenced series is stationary. W ill attempt to add an MA(1) process to differenced corrected deterministic model.

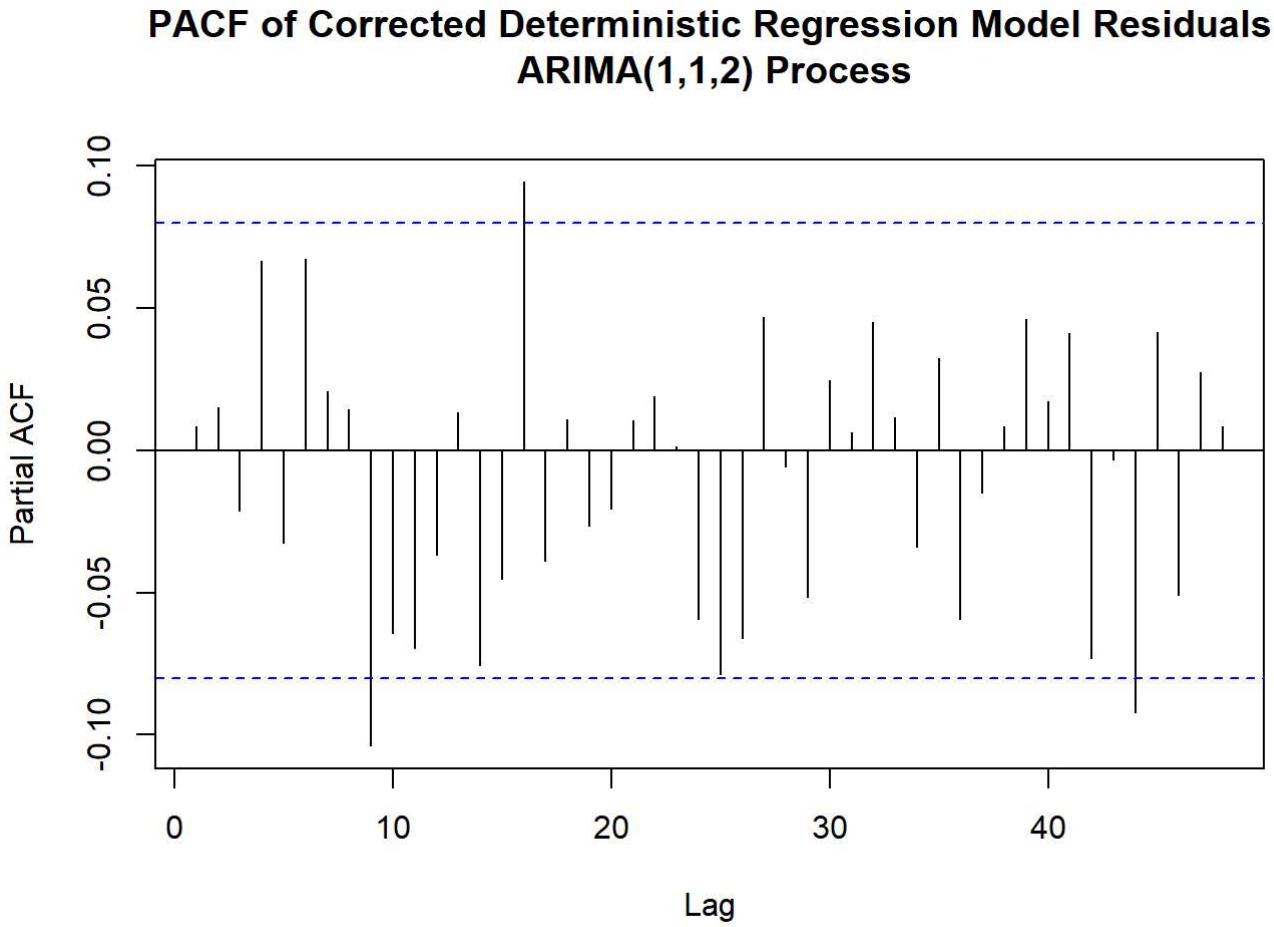
```
# Deterministic TS Corrected Model (adding an AR(1) Process to Differenced Series to induce stationarity in residuals)

x <- cbind(m3_train,m4_train,m5_train,m6_train,
         m7_train,m8_train,m9_train,m10_train,m11_train,m12_train,d2_train,d3_train,d4_train,d5_train,d6_train,
         d7_train)



ar_fit <- Arima(train_crash, order = c(1,1,2), xreg=x)
summary(ar_fit)
```

```
## Series: train_crash
## Regression with ARIMA(1,1,2) errors
##
## Coefficients:
##          ar1      ma1     ma2  m3_train  m4_train  m5_train  m6_train  m7_train
##       0.6317  -1.3629  0.3629   40.9869   40.5137   96.0507  128.9258   74.0156
## s.e.  0.0940   0.1121  0.1120   17.1030   17.5599   17.4107   17.5998   17.4105
##       m8_train  m9_train  m10_train  m11_train  m12_train   d2_train   d3_train
##        60.3435   81.4034    64.5477    76.4802    43.4128  -115.5671  -191.4503
## s.e.   18.2986   22.5975    22.4265    22.5969    22.2024     8.9777     9.6064
##       d4_train  d5_train  d6_train  d7_train
##       -80.1097  -53.8970  -52.0485  -33.2723
## s.e.    9.8821    9.8884    9.6399    9.0061
##
## sigma^2 = 4633:  log likelihood = -3371.06
## AIC=6782.11   AICc=6783.56   BIC=6870.02
##
## Training set error measures:
##                     ME  RMSE      MAE        MPE     MAPE      MASE        ACF1
## Training set 2.297454 66.92 48.46768 -0.8563325 8.221137 0.5866146 0.008479291
```

```
acf(ar_fit$residuals,
    main = "ACF of Corrected Deterministic Regression Model Residuals\n ARIMA(1,1,2) Process",
    lag=48)
```




```
pacf(ar_fit$residuals,
     main = "PACF of Corrected Deterministic Regression Model Residuals\n ARIMA(1,1,2) Process",
     lag=48)
```




```
Box.test(ar_fit$residuals, lag=24)
```

```
##
##  Box-Pierce test
##
## data:  ar_fit$residuals
## X-squared = 31.934, df = 24, p-value = 0.1286
```

```
#COMMENT: PACF can be interpreted as a chopping off at lag 2; thus, adding an AR(2) process to ultimately try an ARMA(1,1,2) Corrected Model.

# COMMENT: Because the ACF of the Residuals only show a few lags that lie slightly on or above the SE bounds, the PACF only has a few lags that lie above the SE bounds (lags 9, 16, and 45), and the p-value of the Box Pierce Test is greater than the level of significance of 5%, we can infer that the autocorrelation and partial autocorrelation of the residuals have been (mostly) removed by applying the Corrected Mix ARMA Process to the model. Thus, because we fail to reject the null hypothesis that the residuals are white noise and the regressor and phi(1) coefficients are all statistically significant, the Corrected Regression Model can effectively be used for forecasting.
```

```
# PREDICTIVE PERFORMANCE OF Corrected Deterministic Model FROM 4.3 ON HOLDOUT SAMPLE

x_test <- cbind(m3_test,m4_test,m5_test,m6_test,
         m7_test,m8_test,m9_test,m10_test,m11_test,m12_test,d2_test,d3_test,d4_test,d5_test,d6_test,
         d7_test)

colnames(x_test) <- colnames(x) # Matching column names between training and testing regressors

# Using the forecast function on the pre-trained model
preds <- forecast(ar_fit, xreg = x_test, h = length(test_crash))

# Display Summary of Predictions
summary(preds)
```

```
##
## Forecast method: Regression with ARIMA(1,1,2) errors
##
## Model Information:
## Series: train_crash
## Regression with ARIMA(1,1,2) errors
##
## Coefficients:
##          ar1      ma1     ma2  m3_train  m4_train  m5_train  m6_train  m7_train
##       0.6317  -1.3629  0.3629   40.9869   40.5137   96.0507  128.9258   74.0156
## s.e.  0.0940   0.1121  0.1120   17.1030   17.5599   17.4107   17.5998   17.4105
##       m8_train  m9_train  m10_train  m11_train  m12_train   d2_train   d3_train
##        60.3435   81.4034    64.5477    76.4802    43.4128  -115.5671  -191.4503
## s.e.   18.2986   22.5975    22.4265    22.5969    22.2024     8.9777     9.6064
##       d4_train  d5_train  d6_train  d7_train
##       -80.1097  -53.8970  -52.0485  -33.2723
## s.e.    9.8821    9.8884    9.6399    9.0061
##
## sigma^2 = 4633:  log likelihood = -3371.06
## AIC=6782.11   AICc=6783.56   BIC=6870.02
##
## Error measures:
##                     ME  RMSE      MAE        MPE     MAPE      MASE        ACF1
## Training set 2.297454 66.92 48.46768 -0.8563325 8.221137 0.5866146 0.008479291
##
## Forecasts:
##     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 601       662.6241 575.3244 749.9239 529.1107 796.1376
## 602       677.6307 587.1951 768.0662 539.3214 815.9399
## 603       708.5217 616.8499 800.1935 568.3218 848.7216
## 604       591.4504 499.2805 683.6202 450.4888 732.4119
## 605       514.6170 422.2433 606.9907 373.3437 655.8904
## 606       625.3573 532.8987 717.8160 483.9541 766.7606
## 607       651.1909 558.6961 743.6858 509.7323 792.6496
## 608       652.7999 560.2891 745.3107 511.3169 794.2830
## 609       671.4249 578.9068 763.9430 529.9307 812.9191
## 610       725.6614 633.1398 818.1830 584.1618 867.1610
## 611       610.0339 517.5105 702.5573 468.5316 751.5362
## 612       534.1126 441.5883 626.6369 392.6089 675.6164
```

```
## 613       645.4291 552.9042 737.9539 503.9245 786.9336
## 614       671.6266 579.1015 764.1518 530.1216 813.1316
## 615       673.4655 580.9402 765.9908 531.9602 814.9708
## 616       692.2356 599.7102 784.7611 550.7302 833.7411
## 617       725.5041 632.9786 818.0296 583.9985 867.0096
## 618       609.9345 517.4090 702.4601 468.4289 751.4401
## 619       534.0498 441.5243 626.5754 392.5442 675.5555
## 620       645.3894 552.8638 737.9150 503.8837 786.8951
## 621       671.6016 579.0760 764.1272 530.0959 813.1072
## 622       673.4497 580.9241 765.9753 531.9440 814.9554
## 623       692.2256 599.7000 784.7513 550.7199 833.7314
## 624       725.4978 632.9721 818.0234 583.9920 867.0035
## 625       609.9305 517.4049 702.4562 468.4248 751.4363
## 626       534.0473 441.5217 626.5729 392.5416 675.5530
## 627       645.3878 552.8622 737.9134 503.8821 786.8935
## 628       671.6005 579.0749 764.1262 530.0948 813.1063
## 629       673.4490 580.9234 765.9747 531.9433 814.9548
## 630       692.2252 599.6996 784.7509 550.7195 833.7310
## 631       725.4975 632.9719 818.0231 583.9918 867.0032
## 632       609.9304 517.4048 702.4560 468.4247 751.4361
## 633       534.0472 441.5216 626.5728 392.5415 675.5529
## 634       645.3878 552.8621 737.9134 503.8820 786.8935
## 635       671.6005 579.0749 764.1261 530.0948 813.1062
## 636       673.4490 580.9234 765.9746 531.9433 814.9547
## 637       692.2252 599.6996 784.7508 550.7195 833.7309
## 638       725.4975 632.9719 818.0231 583.9918 867.0032
## 639       609.9304 517.4048 702.4560 468.4247 751.4361
## 640       517.1916 424.6660 609.7172 375.6858 658.6973
## 641       628.5321 536.0065 721.0577 487.0264 770.0378
## 642       654.7449 562.2192 747.2705 513.2391 796.2506
## 643       656.5934 564.0677 749.1190 515.0876 798.0991
## 644       675.3696 582.8440 767.8952 533.8638 816.8753
## 645       708.6418 616.1162 801.1675 567.1361 850.1476
## 646       593.0747 500.5491 685.6003 451.5690 734.5804
## 647       517.1916 424.6660 609.7172 375.6858 658.6973
## 648       628.5321 536.0065 721.0577 487.0264 770.0378
## 649       654.7449 562.2192 747.2705 513.2391 796.2506
## 650       656.5934 564.0677 749.1190 515.0876 798.0991
## 651       675.3696 582.8440 767.8952 533.8638 816.8753
```

```
## 652       708.6418 616.1162 801.1675 567.1361 850.1476
## 653       593.0747 500.5491 685.6003 451.5690 734.5804
## 654       517.1916 424.6660 609.7172 375.6858 658.6973
## 655       628.5321 536.0065 721.0577 487.0264 770.0378
## 656       654.7449 562.2192 747.2705 513.2391 796.2506
## 657       656.5934 564.0677 749.1190 515.0876 798.0991
## 658       675.3696 582.8440 767.8952 533.8638 816.8753
## 659       708.6418 616.1162 801.1675 567.1361 850.1476
## 660       593.0747 500.5491 685.6003 451.5690 734.5804
## 661       517.1916 424.6660 609.7172 375.6858 658.6973
## 662       628.5321 536.0065 721.0577 487.0264 770.0378
## 663       654.7449 562.2192 747.2705 513.2391 796.2506
## 664       656.5934 564.0677 749.1190 515.0876 798.0991
## 665       675.3696 582.8440 767.8952 533.8638 816.8753
## 666       708.6418 616.1162 801.1675 567.1361 850.1476
## 667       593.0747 500.5491 685.6003 451.5690 734.5804
## 668       517.1916 424.6660 609.7172 375.6858 658.6973
## 669       628.5321 536.0065 721.0577 487.0264 770.0378
## 670       654.7449 562.2192 747.2705 513.2391 796.2506
## 671       668.5258 576.0002 761.0515 527.0201 810.0316
## 672       687.3021 594.7764 779.8277 545.7963 828.8078
## 673       720.5743 628.0487 813.0999 579.0686 862.0800
## 674       605.0072 512.4816 697.5328 463.5015 746.5129
## 675       529.1240 436.5984 621.6497 387.6183 670.6298
## 676       640.4646 547.9390 732.9902 498.9589 781.9703
## 677       666.6773 574.1517 759.2030 525.1716 808.1831
## 678       668.5258 576.0002 761.0515 527.0201 810.0316
## 679       687.3021 594.7764 779.8277 545.7963 828.8078
## 680       720.5743 628.0487 813.0999 579.0686 862.0800
## 681       605.0072 512.4816 697.5328 463.5015 746.5129
## 682       529.1240 436.5984 621.6497 387.6183 670.6298
## 683       640.4646 547.9390 732.9902 498.9589 781.9703
## 684       666.6773 574.1517 759.2030 525.1716 808.1831
## 685       668.5258 576.0002 761.0515 527.0201 810.0316
## 686       687.3021 594.7764 779.8277 545.7963 828.8078
## 687       720.5743 628.0487 813.0999 579.0686 862.0800
## 688       605.0072 512.4816 697.5328 463.5015 746.5129
## 689       529.1240 436.5984 621.6497 387.6183 670.6298
## 690       640.4646 547.9390 732.9902 498.9589 781.9703
```

```
## 691       666.6773 574.1517 759.2030 525.1716 808.1831
## 692       668.5258 576.0002 761.0515 527.0201 810.0316
## 693       687.3021 594.7764 779.8277 545.7963 828.8078
## 694       720.5743 628.0487 813.0999 579.0686 862.0800
## 695       605.0072 512.4816 697.5328 463.5015 746.5129
## 696       529.1240 436.5984 621.6497 387.6183 670.6298
## 697       640.4646 547.9390 732.9902 498.9589 781.9703
## 698       666.6773 574.1517 759.2030 525.1716 808.1831
## 699       668.5258 576.0002 761.0515 527.0201 810.0316
## 700       687.3021 594.7764 779.8277 545.7963 828.8078
## 701       687.5070 594.9813 780.0326 546.0012 829.0127
## 702       571.9398 479.4142 664.4655 430.4341 713.4456
## 703       496.0567 403.5311 588.5823 354.5510 637.5624
## 704       607.3972 514.8716 699.9228 465.8915 748.9029
## 705       633.6100 541.0844 726.1356 492.1043 775.1157
## 706       635.4585 542.9329 727.9841 493.9528 776.9642
## 707       654.2347 561.7091 746.7603 512.7290 795.7404
## 708       687.5070 594.9813 780.0326 546.0012 829.0127
## 709       571.9398 479.4142 664.4655 430.4341 713.4456
## 710       496.0567 403.5311 588.5823 354.5510 637.5624
## 711       607.3972 514.8716 699.9228 465.8915 748.9029
## 712       633.6100 541.0844 726.1356 492.1043 775.1157
## 713       635.4585 542.9329 727.9841 493.9528 776.9642
## 714       654.2347 561.7091 746.7603 512.7290 795.7404
## 715       687.5070 594.9813 780.0326 546.0012 829.0127
## 716       571.9398 479.4142 664.4655 430.4341 713.4456
## 717       496.0567 403.5311 588.5823 354.5510 637.5624
## 718       607.3972 514.8716 699.9228 465.8915 748.9029
## 719       633.6100 541.0844 726.1356 492.1043 775.1157
## 720       635.4585 542.9329 727.9841 493.9528 776.9642
## 721       654.2347 561.7091 746.7603 512.7290 795.7404
## 722       687.5070 594.9813 780.0326 546.0012 829.0127
## 723       571.9398 479.4142 664.4655 430.4341 713.4456
## 724       496.0567 403.5311 588.5823 354.5510 637.5624
## 725       607.3972 514.8716 699.9228 465.8915 748.9029
## 726       633.6100 541.0844 726.1356 492.1043 775.1157
## 727       635.4585 542.9329 727.9841 493.9528 776.9642
## 728       654.2347 561.7091 746.7603 512.7290 795.7404
## 729       687.5070 594.9813 780.0326 546.0012 829.0127
```

```
## 730       571.9398 479.4142 664.4655 430.4341 713.4456
## 731       496.0567 403.5311 588.5823 354.5510 637.5624
```

```
# Extract Forecast Values (Point Forecasts)
holdfit <- preds$mean

# Accuracy Measures for Holdout Sample
accuracy(holdfit, test_crash)
```

```
##                ME     RMSE      MAE       MPE     MAPE
## Test set 12.3668 79.05619 57.83768 0.3732759 9.119832
```

```
# TABLE of Predictive Performance Metrics FOR 4.3 Model

#training metrics
acc_train<- accuracy(forecast(ar_fit, xreg = x, h=length(train_crash)), train_crash)

#manually deriving Training R-Squared Metrics
REG_Train_R2 <- 1 - sum((train_crash - fitted(ar_fit))^2) / sum((train_crash - mean(train_crash))^2)

#test metrics
acc<- accuracy(holdfit, test_crash)

#manually deriving Test R-Squared Metric
REG_Test_R2 <- 1 - sum((test_crash - holdfit)^2) / sum((test_crash - mean(test_crash))^2)


Model_Metric_df <- data.frame(Metric = c("MAPE", "RMSE", "MAE", "R2"),
                Training_set = c(acc_train[1,5], acc_train[1,2], acc_train[1,3], REG_Train_R2),
                Holdout_set = c(acc[,5], acc[,2], acc[,3], REG_Test_R2))
Model_Metric_df
```

```
##   Metric Training_set Holdout_set
## 1   MAPE    8.2211374   9.1198316
## 2   RMSE   66.9200046  79.0561926
## 3    MAE   48.4676843  57.8376798
## 4     R2    0.5472755   0.3693268
```