Forecasting Traffic Congestion

Ryan Valencia

March 22, 2025

1 Abstract

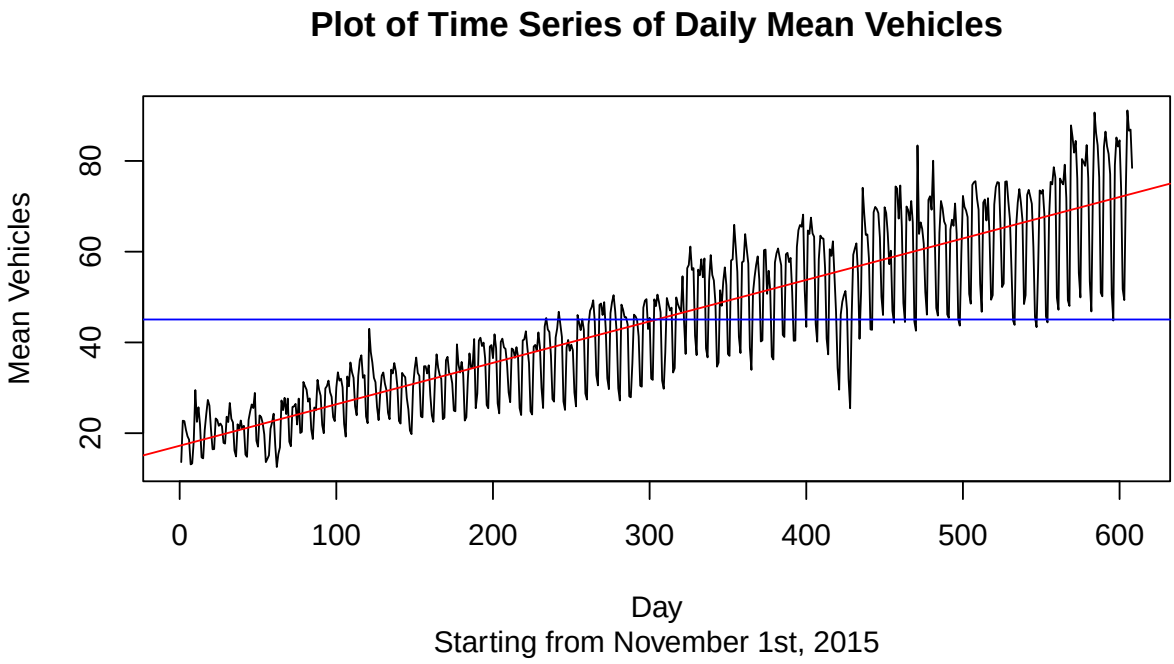
As the human population grows exponentially over time, public goods become overused. This can be seen in our roads; traffic congestion is becoming a problem. As more and more cars go on the road, it becomes harder for people to go from point A to point B, wasting fuel, time, and efficiency. In order to attempt to solve this problem, we must try to forecast traffic congestion. Using a traffic prediction data set from Kaggle, I analyzed the data with time series techniques in order to effectively fit a model that is able to predict congestion over time. This was done by transforming and differencing the data set in order to achieve stationary data to effectively fit time series models. To summarize, I fit a SARIMA model to data of the daily mean number of vehicles in a certain junction, which was able to accurately predict the future data.

2 Introduction

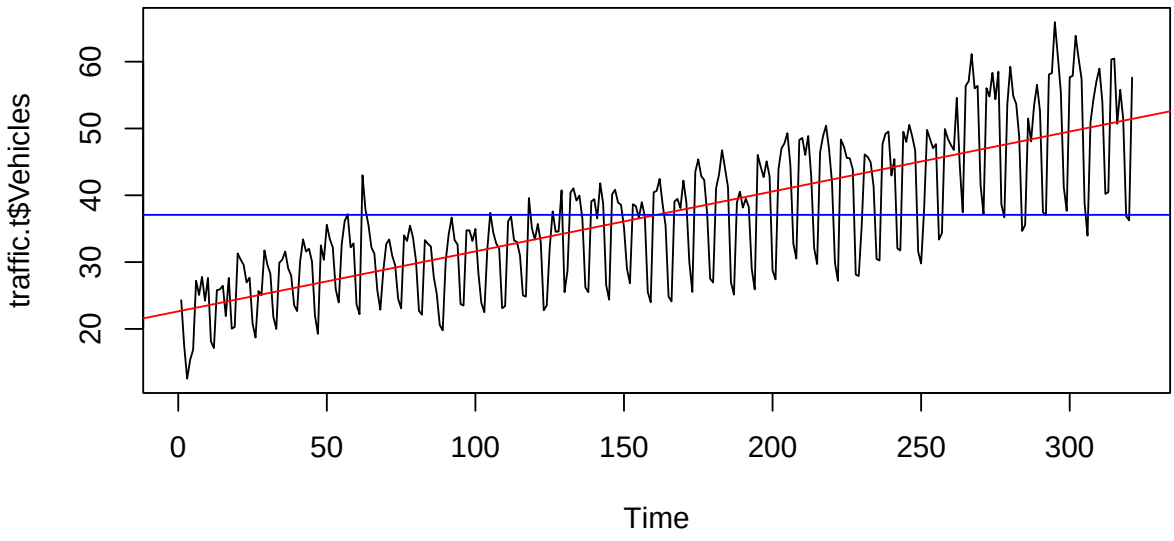
The Kaggle dataset I used contained observations regarding the number of vehicles in each hour in four different highway junctions, which are usually the most congested areas and the cause of traffic. To simplify this problem, I decided to aggregate the data by mean in order to find a trend in the average number of vehicles per day. While the model's forecasts were not completely accurate, it would do a good job providing an idea of the level of traffic on a given day. This model was built by manipulating the data through Box-Cox transformations, as well as differencing to remove trend and seasonality in order analyze the time series. Diagnostics checks were done in order to ensure the model is a good fit, and forecasts were plotted on a validation set in order to test prediction accuracy.

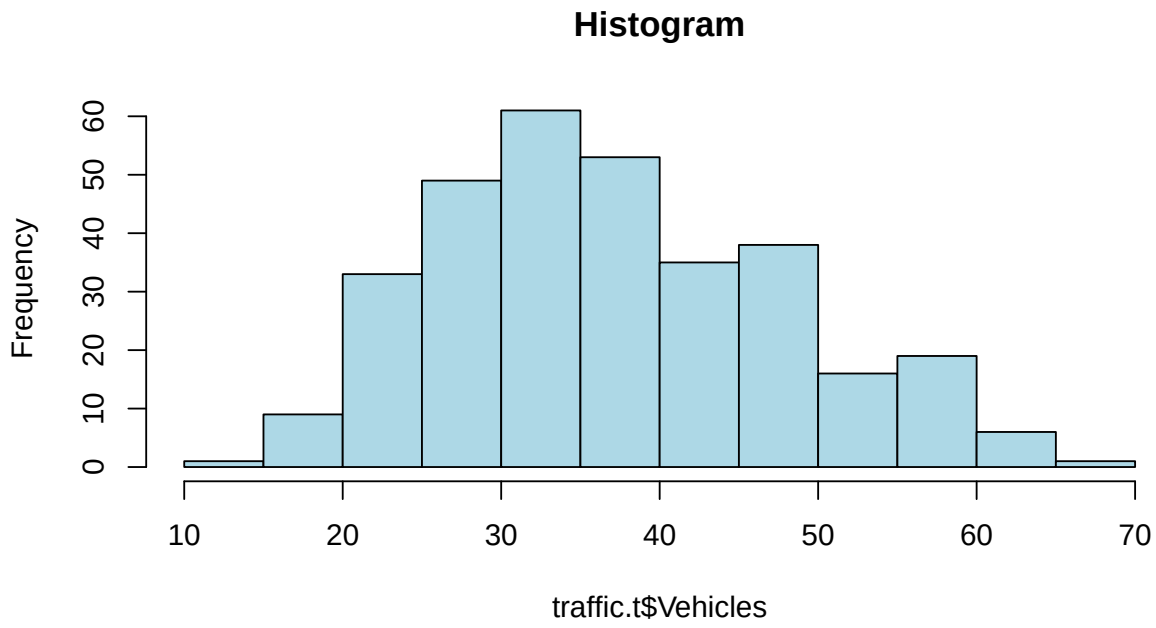
3 Modeling

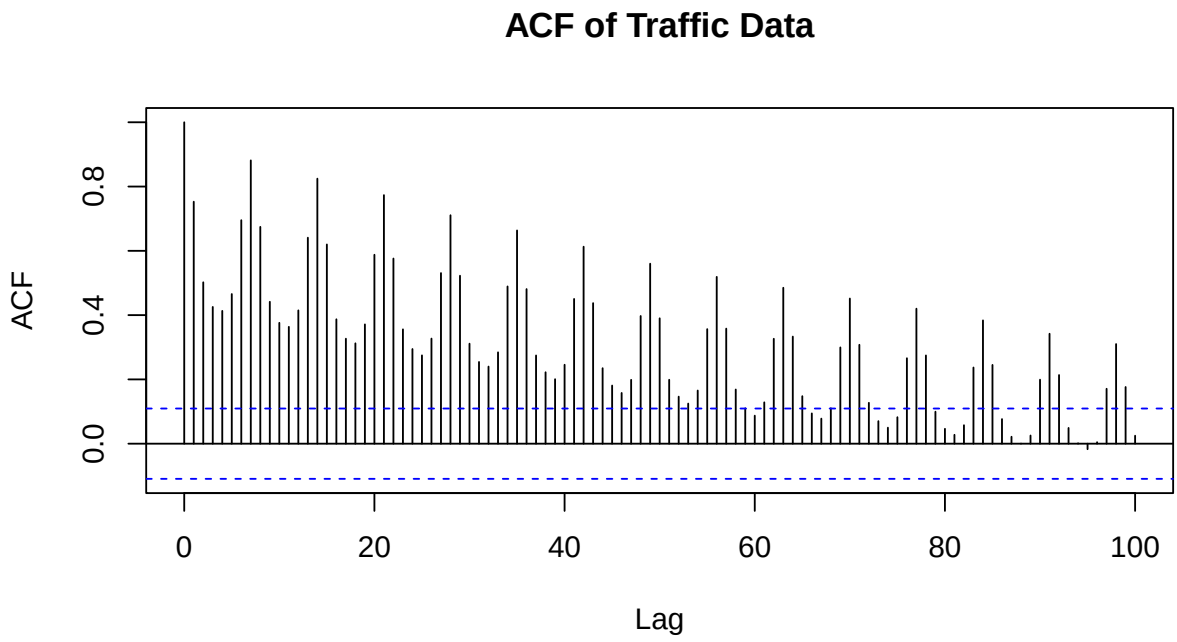
I started by loading in the raw data, in which this plot of number of vehicles versus days after November 15th, 2015 was created.



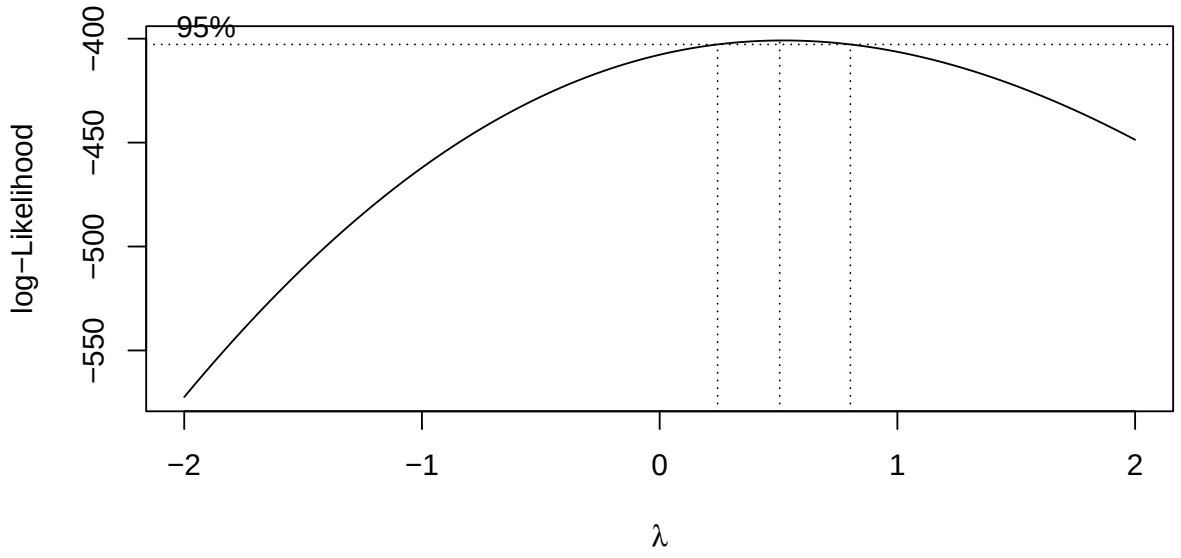
I then partitioned the data into a training set consisting of a subset of observations, and a validation set of the next 30 observations after the training set ended. Plots of the time series, histogram of the distribution of vehicles, and autocorrelations are shown below.



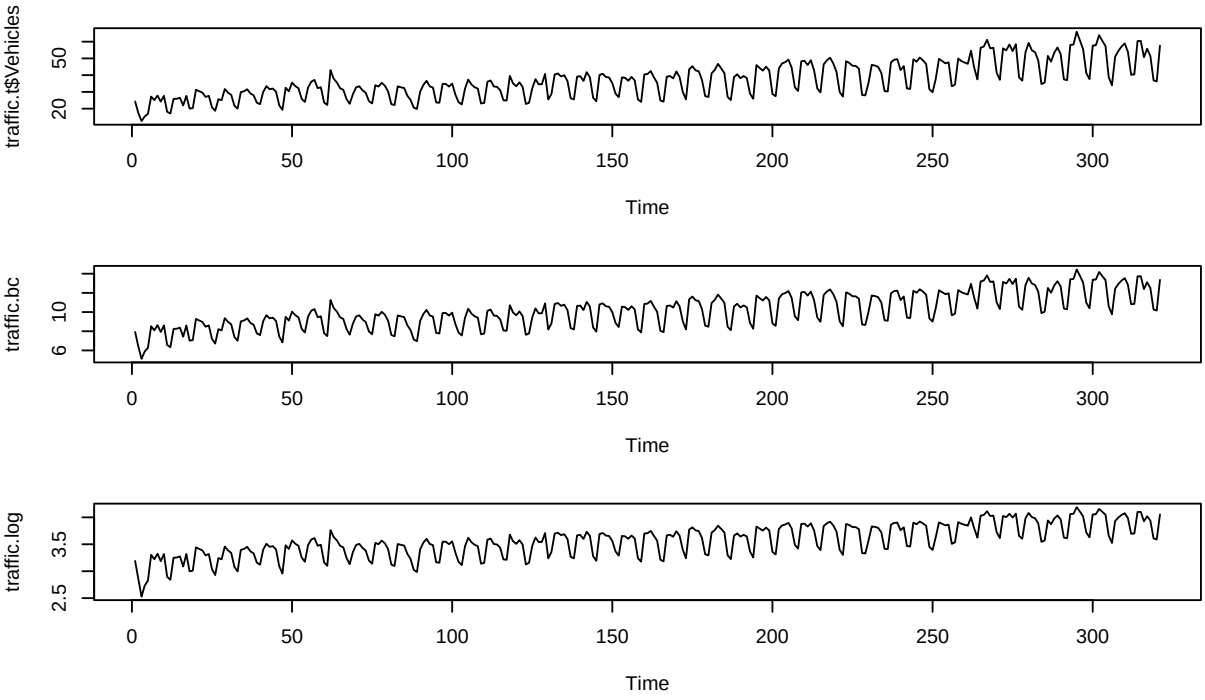




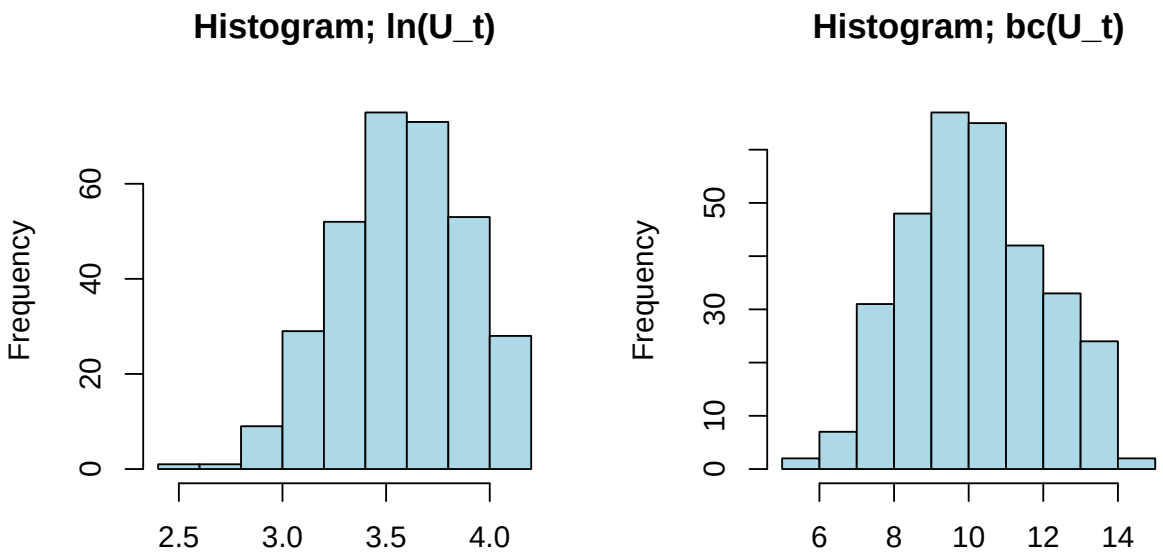
The histogram of the Traffic Data appears to be somewhat normal, with a slight right-skew. Furthermore, the ACFs are very large and periodic. These characteristics are evident of non-stationary data. To stabilize the variance, we are going to transform the data using either Box-Cox or log transformation. In order to choose, I plotted the transformed data into separate histograms to analyze normality and variance.



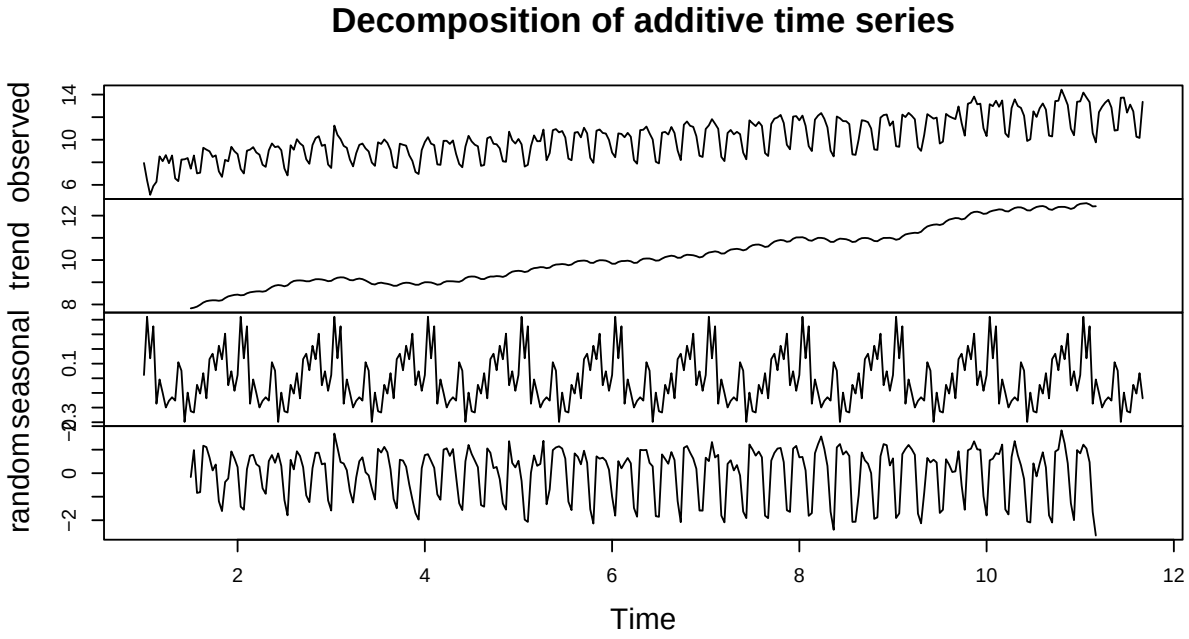
The Box-Cox transformation command gives the value $\lambda = 0.5051$. We will now check if the log or Box-Cox transformation appears to be more normal.



The plots of the time-series transformations shows that their variances are smaller than the original data set, implying that the transformations were necessary in order to achieve constant variance and analyze the data properly.

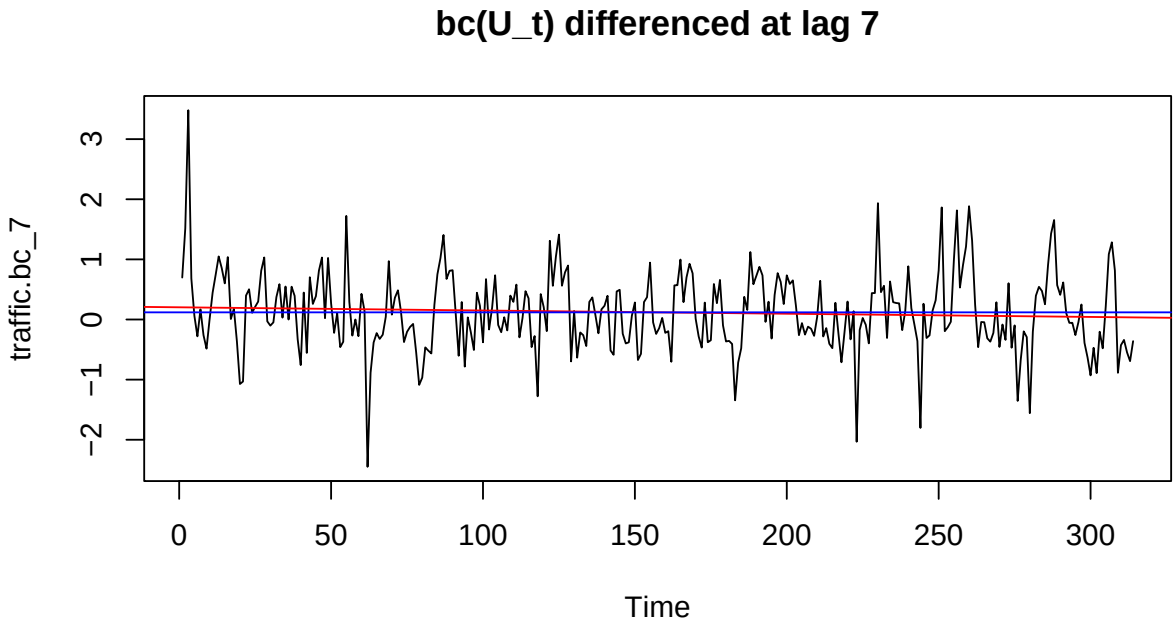


From the histograms of the new data, we can see that the transformations returned histograms that appear to be more normal than the original data. Comparing the two transformations, it seems like the Box-Cox transformation is more normal with a more stabilized variance. For this reason, we will choose to use the Box-Cox transformed data.

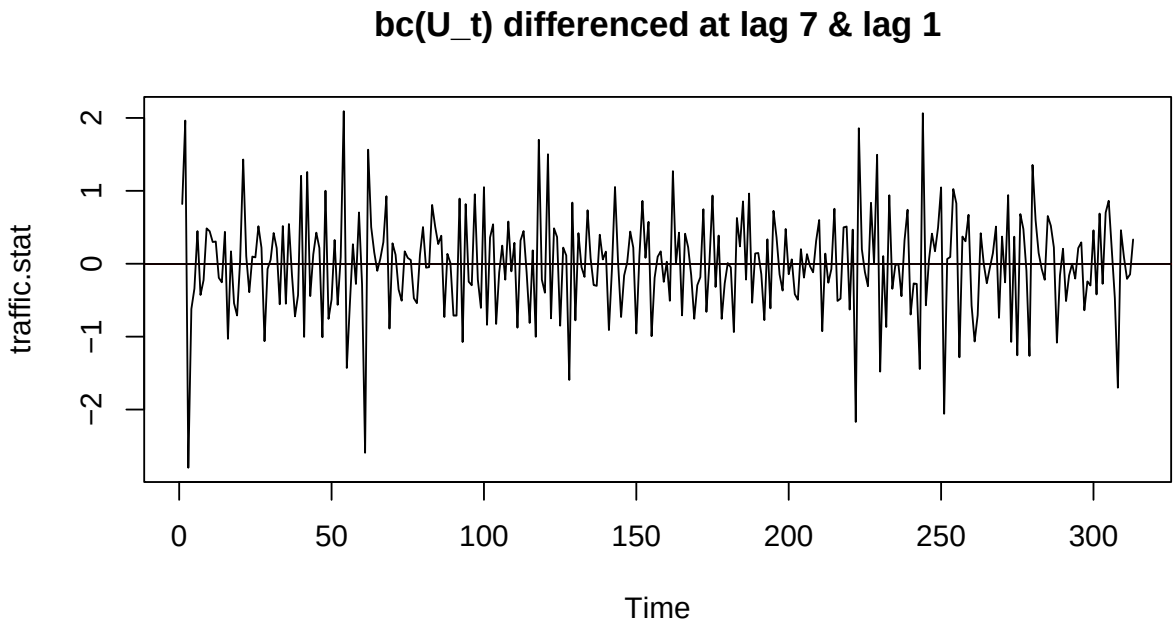


The decomposition of bc(U_t) shows that there is seasonality present and a somewhat linear trend. This means that we must difference the data in order to achieve a stationary training set to fit our model. Furthermore, we can see the random component has a roughly equal variance across all observations, meaning that the transformation was successful in normalizing the data. While differencing the data, we will check the variance to ensure that each difference decreases

is, otherwise there will be overdifferencing present. The pre-differenced data has a variance of 3.3284, so we will try to achieve a variance closer to zero in the process of making the data stationary.

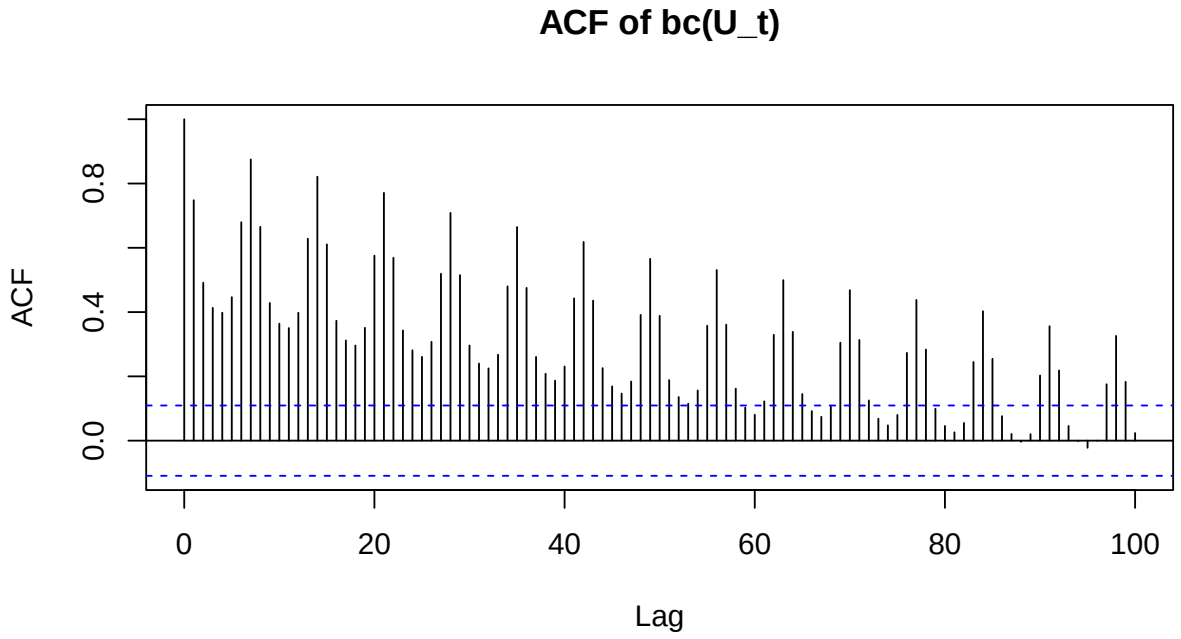


The plot of the Box-Cox transformed data differenced at lag 7 displays the removal of the seasonality. The variance of this differenced data is 0.4318, which is a lot lower than the original variance of 3.3284. From the plot, it is hard to see a visible trend, however the decomposition of the data clearly showed a slightly linear trend, so one more differencing at lag 1 should be done.

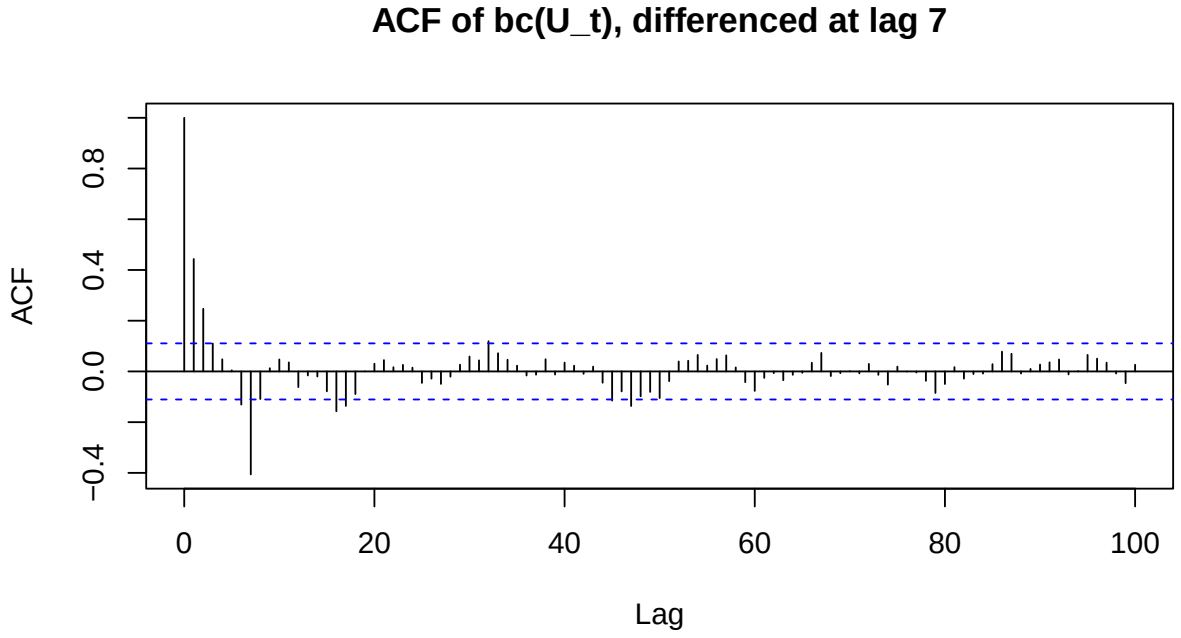


The plot of the Box-Cox transformed data differenced twice at lag 7 and 1 appears to be stationary. There is no

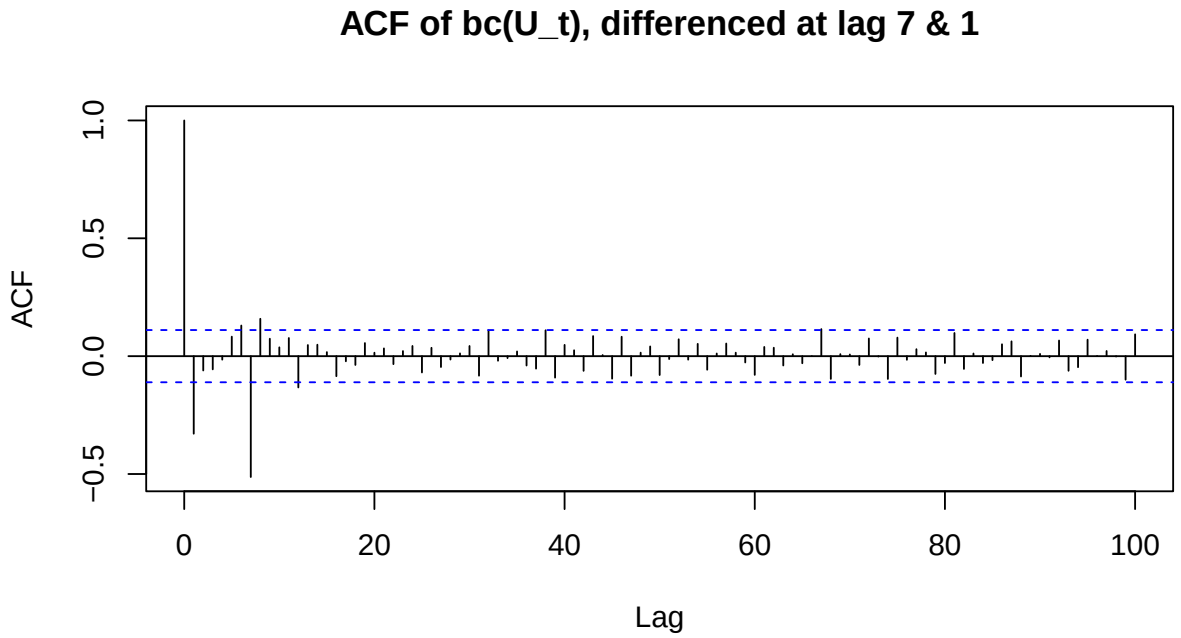
seasonality and there seems to be no trend. The variance of this data is 0.4806, which is slightly lower than 0.4318, the variance of the data only differenced at lag 7. In order to confirm stationarity, we will look at the graph of the ACFs.



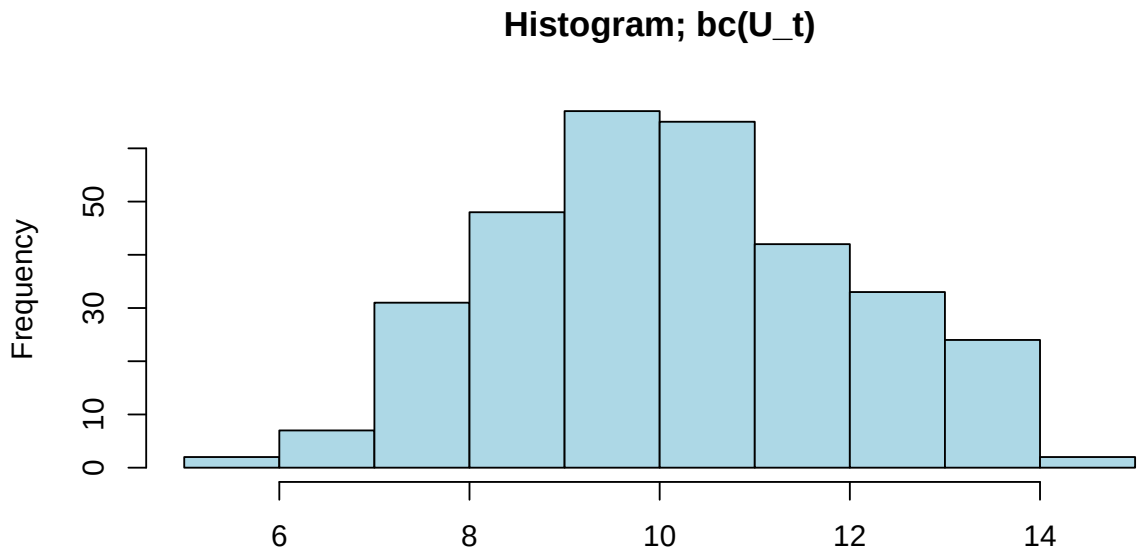
The plot of ACF of bc(U_t) shows slow decay with a seasonal component, which indicates non-stationarity.

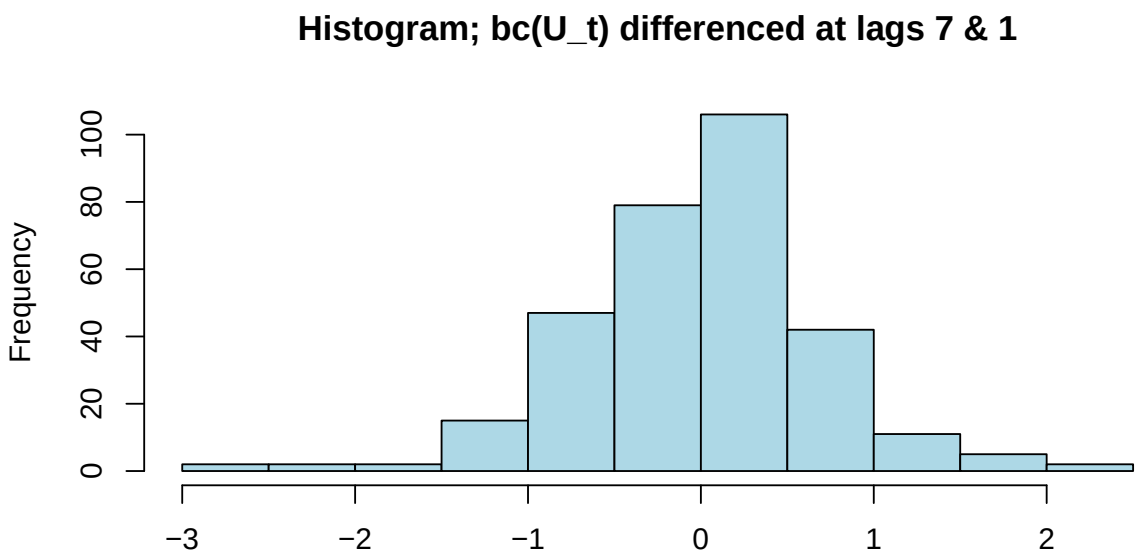


The plot of ACF of bc(U_t) differenced at lag 7 has no seasonality apparent, but there is still a slow decay, meaning the data is still not stationary.

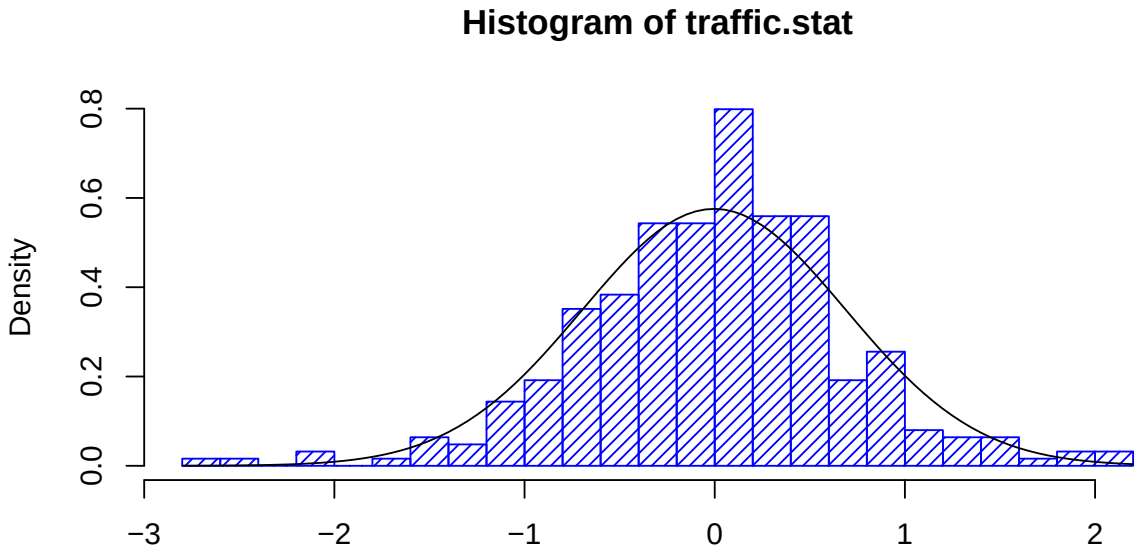


The plot of ACF of bc(U_t) differenced at lag 7 & 1 has fast decay, which indicates stationarity. This means we should work with $\nabla_1 \nabla_7$ bc(U_t).



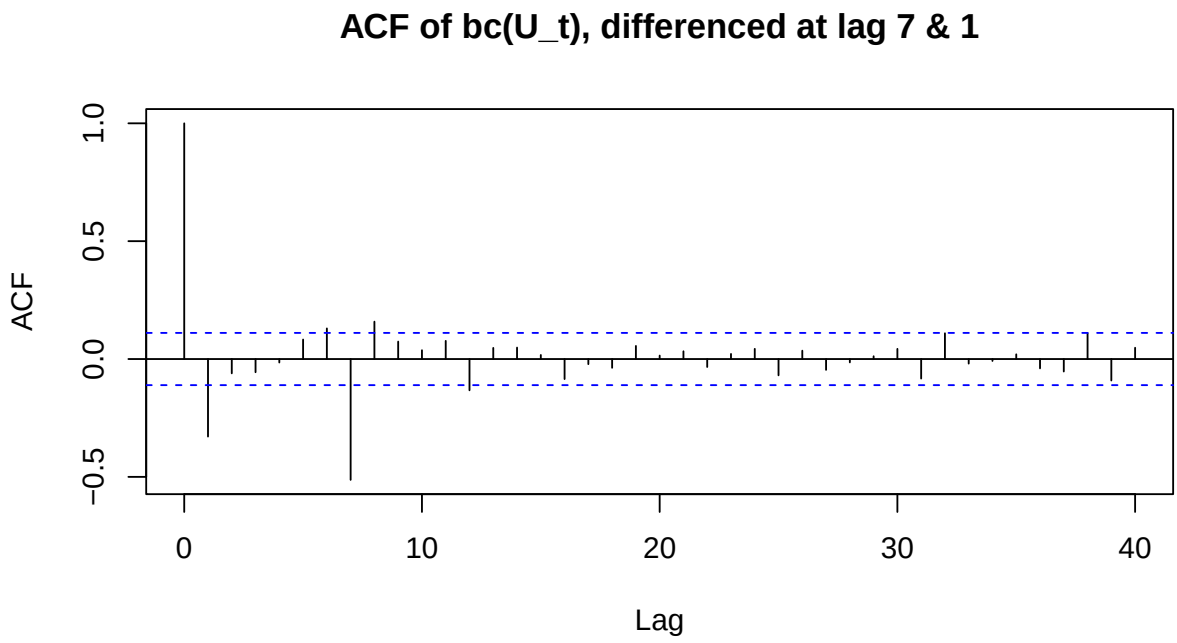


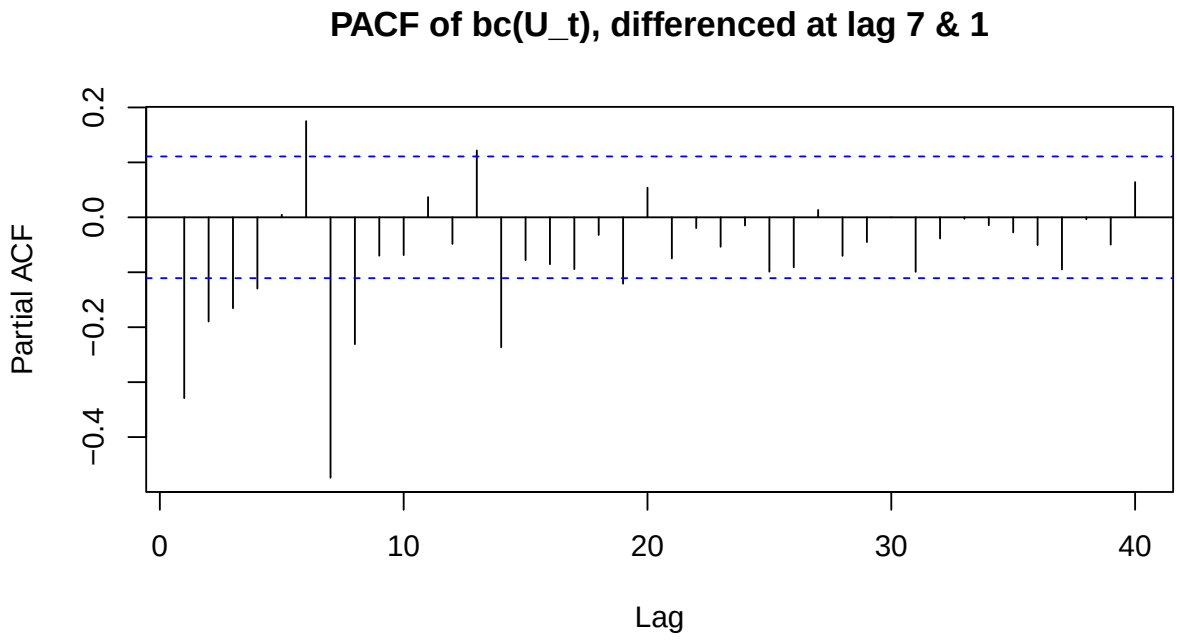
The histogram of $\nabla_1 \nabla_7$ bc(U_t) looks symmetric and almost Gaussian, which is what we were trying to achieve.



The histogram with the curve shows that the stationary traffic data resembles a normal curve. We can now begin fitting different time-series models to this stationary data set. To do so, we must analyze the ACF and PACF graphs to determine candidate models to try.

3.1 Fitting Models





ACF outside confidence intervals: Lags 1, 6, 7, 8

PACF outside confidence intervals: Lags 1, 2, 3, 4, 6, 7, 8, 13, 14

Because we differenced once at lag 1 and once at lag 7, we have that D = d = 1. The seasonal period is s = 7. From the ACF plot, we can see that there is a significant spike at lag 1 and 7, with small spikes at around 6 and 8 most likely due to the seasonal period. This means that q = 1 and Q = 1. From the PACF graph, we see significant spikes at lag 1,

2, 3, 4, 6, 7, 8, 13 and 14. Because there are spikes at and around s and 2s, P can be either 0, 1, or 2. Because of the early spikes, p can be 1, 2, 3, and maybe 4. Therefore we test these model to determine which model has the lowest AICc value.

List of candidate models: SARIMA for bc(U_t): s = 7, d = 1, D = 1,
Q = 1 or 2, q = 1, P = 0 or 1 or 2, p = 1:3

We tried SARIMA with P = 0:2, p = 1:3, Q = 1 and 2, and q = 1. The lowest AICc was SARIMA with p = 0, Q = 2, q = 1, with an AICc value of 410.5. The second lowest AICc was SARIMA with P = p = 1, Q = q = 1, with an AIC of 411.

Therefore, we have:

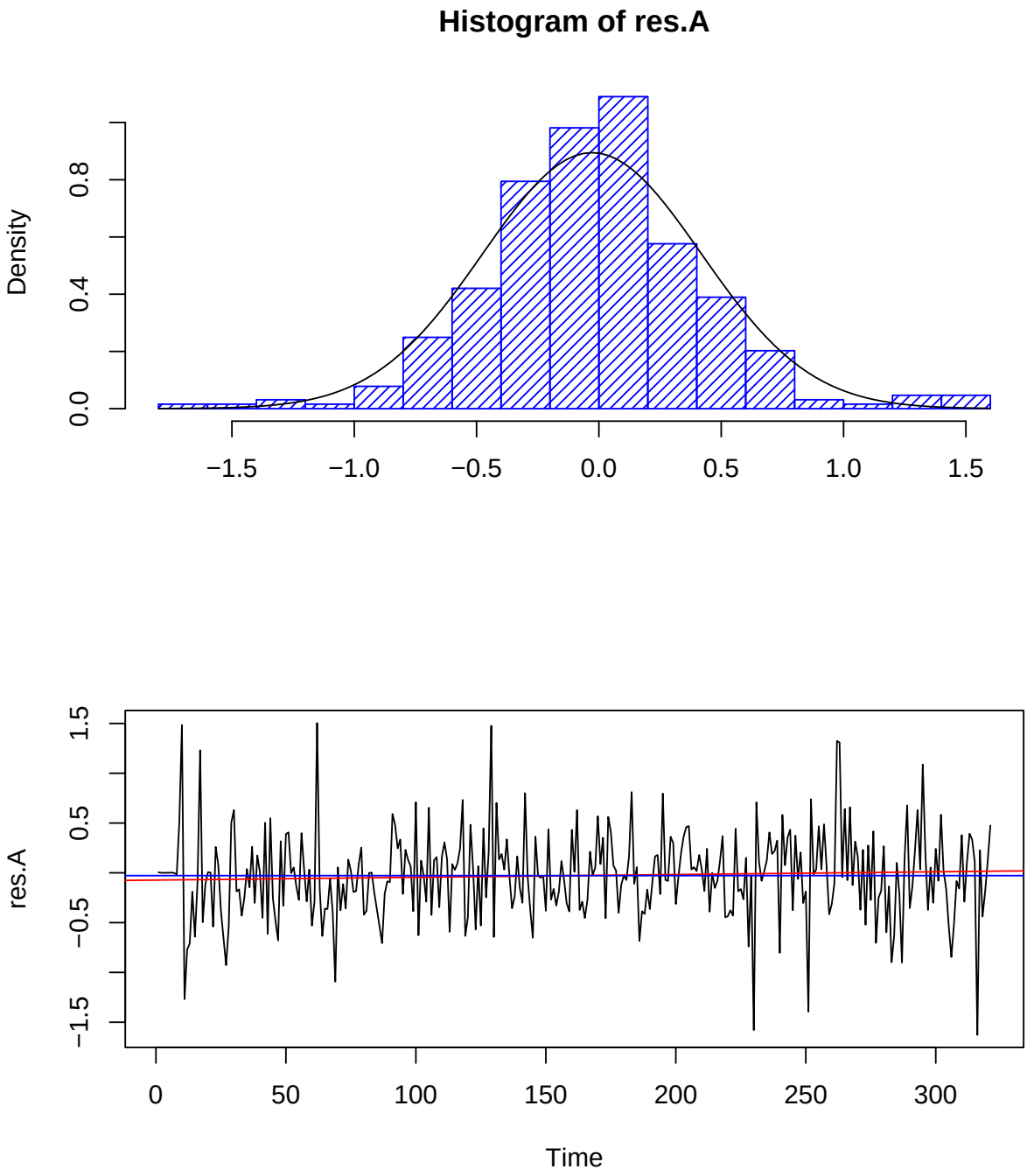
Model A:
$(1 - 0.319_{(0.098)}B)(1 - B)Y_t = (1 - 0.812_{(0.068)}B)(1 - 0.975_{(0.068)}B^7)(1 + 0.161_{(0.067)}B^{14})Z_t$

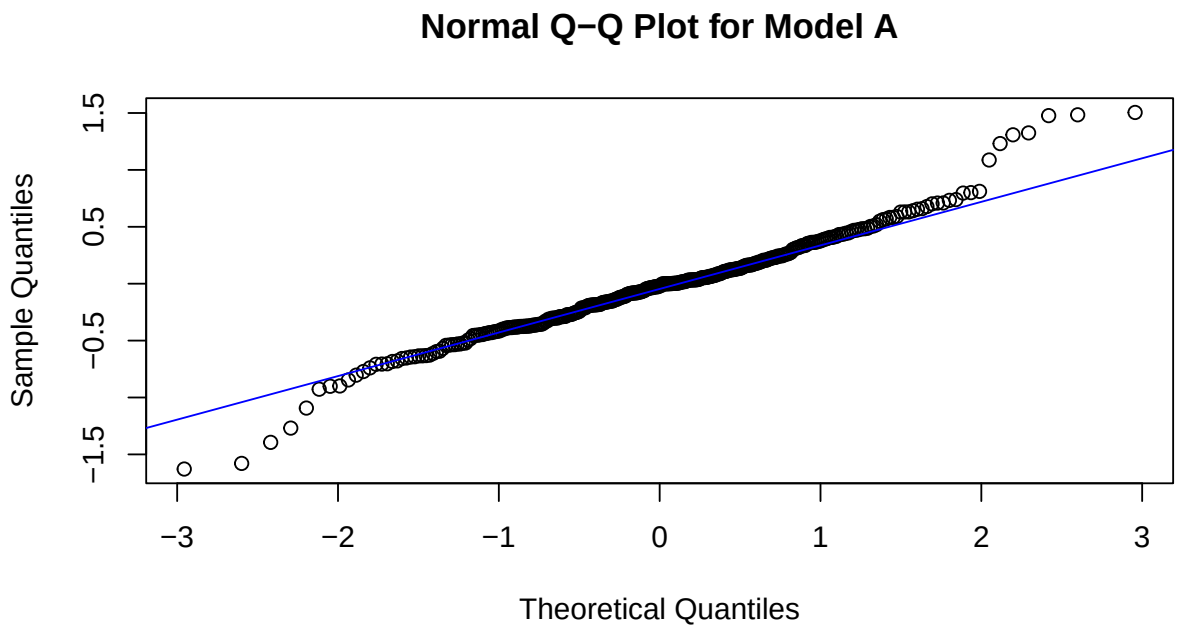
Model B:
$(1 - 0.325_{(0.100)}B)(1 - B)(1 + 0.172_{(0.075)}B^7)Y_t = (1 - 0.817_{(0.070)}B)(1 - 0.795_{(0.047)}B^7)Z_t$

Model A and B are stationary because $|\phi_1| < 1$ for both. They are both also invertible because $|\theta_1| < 1$ for both models.

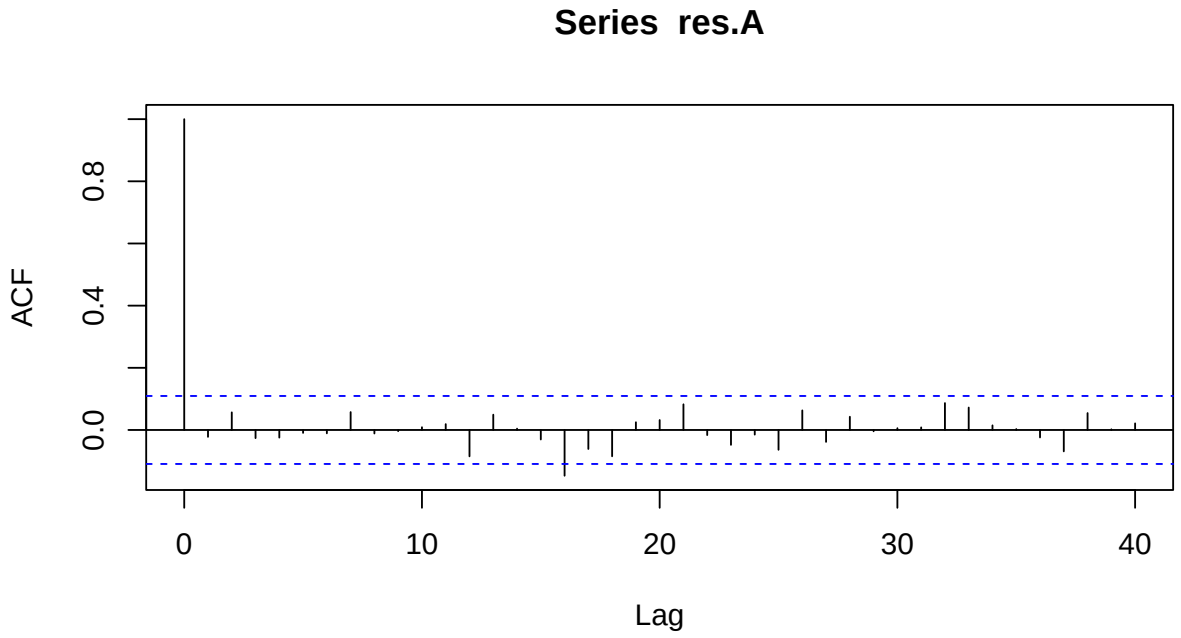
3.2 Diagnostic Checking

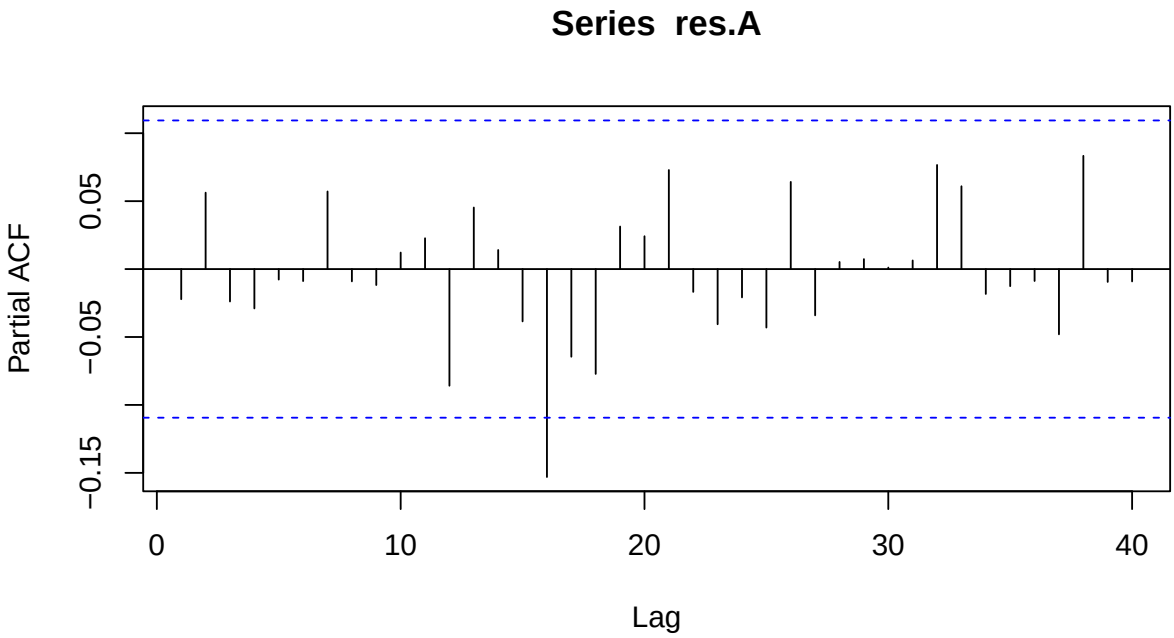
3.2.1 Model A:

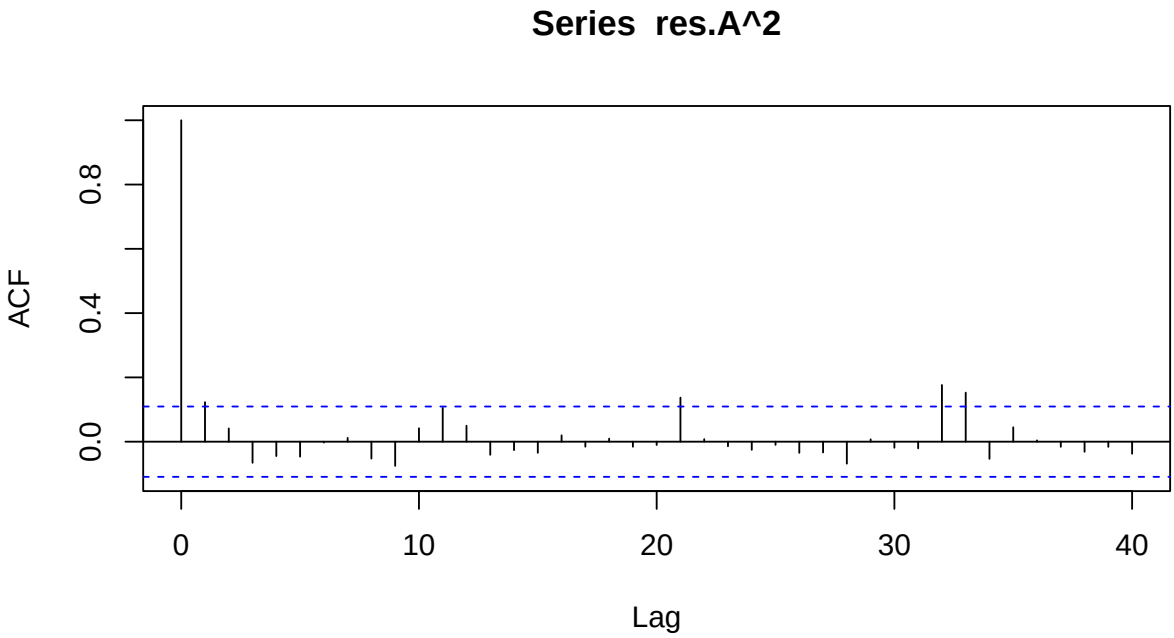




From the plots of the residuals, we can see that they appear to be normally distributed, resembling white noise. The Q-Q plot follows a straight line, however it deviates from the line at the end points. This could imply that there are outliers in the data.







The ACF and PACF of model A's residuals are contained within the confidence interval for the most part. This is also somewhat true for the ACF of the squared residuals. Therefore, we move onto the formal diagnostics tests.

```
# diagnostics checks
shapiro.test(res.A)
```

```
##
##  Shapiro-Wilk normality test
```

```
##
## data:  res.A
## W = 0.97, p-value = 2e-05
```

```
Box.test(res.A, lag = 18, type = c("Box-Pierce"), fitdf = 4)
```

```
##
##  Box-Pierce test
##
## data:  res.A
## X-squared = 17, df = 14, p-value = 0.3
```

```
Box.test(res.A, lag = 18, type = c("Ljung-Box"), fitdf = 4)
```

```
##
##  Box-Ljung test
##
## data:  res.A
## X-squared = 18, df = 14, p-value = 0.2
```

```
Box.test((res.A)^2, lag = 18, type = c("Ljung-Box"), fitdf = 0)
```

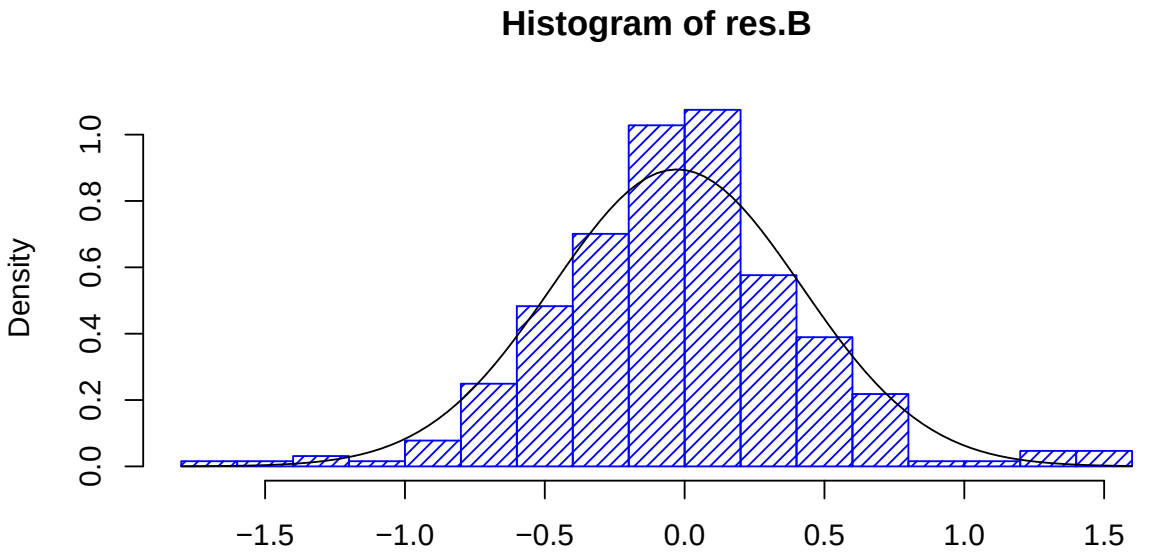
```
##
##  Box-Ljung test
##
## data:  (res.A)^2
## X-squared = 18, df = 18, p-value = 0.5
```

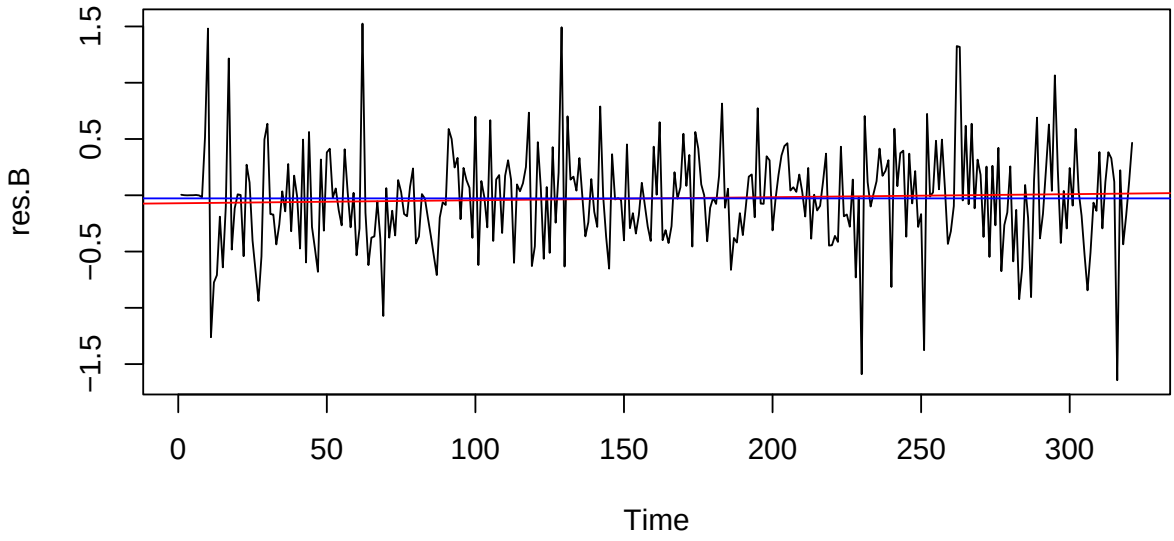
```
ar(res.A, aic = TRUE, order.max = NULL, method = c("yule-walker"))
```

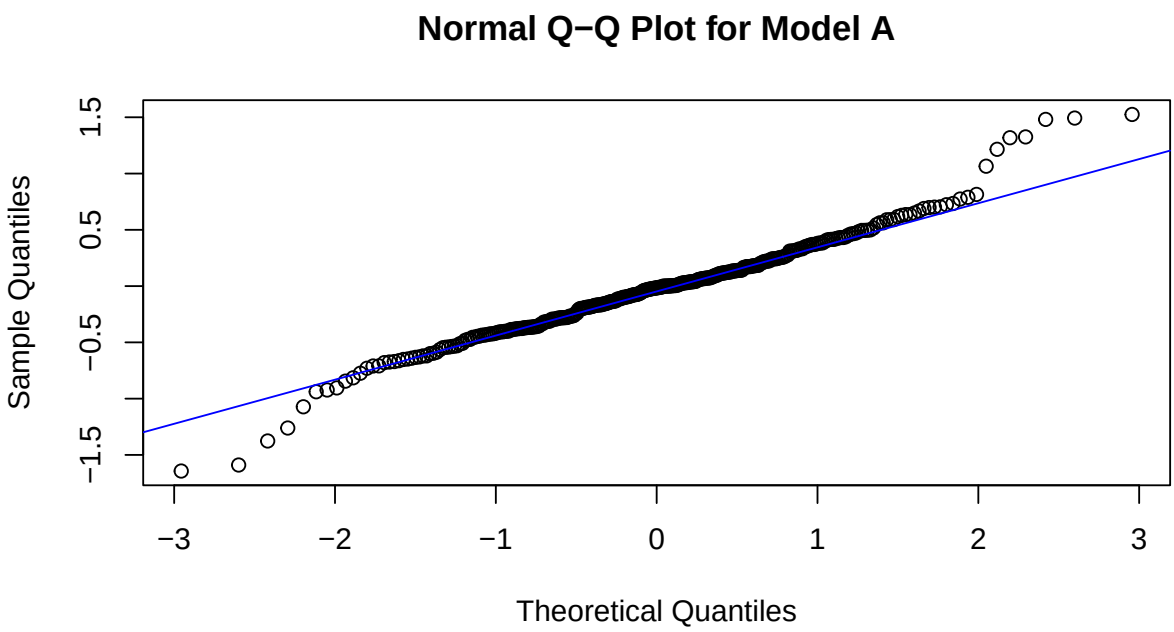
```
##
## Call:
## ar(x = res.A, aic = TRUE, order.max = NULL, method = c("yule-walker"))
##
##
## Order selected 0  sigma^2 estimated as  0.199
```

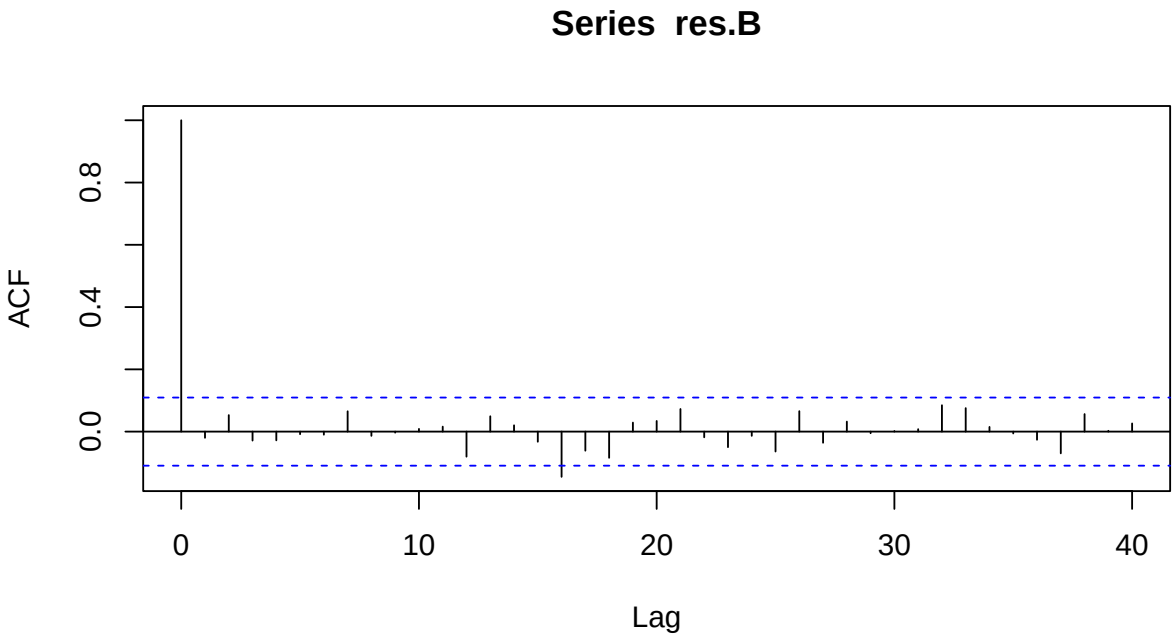
Model A passed all tests, except for the Shapiro Wilk test for normality. This may be due to the presence of outliers in the traffic data set. We repeat the same tests for Model B.

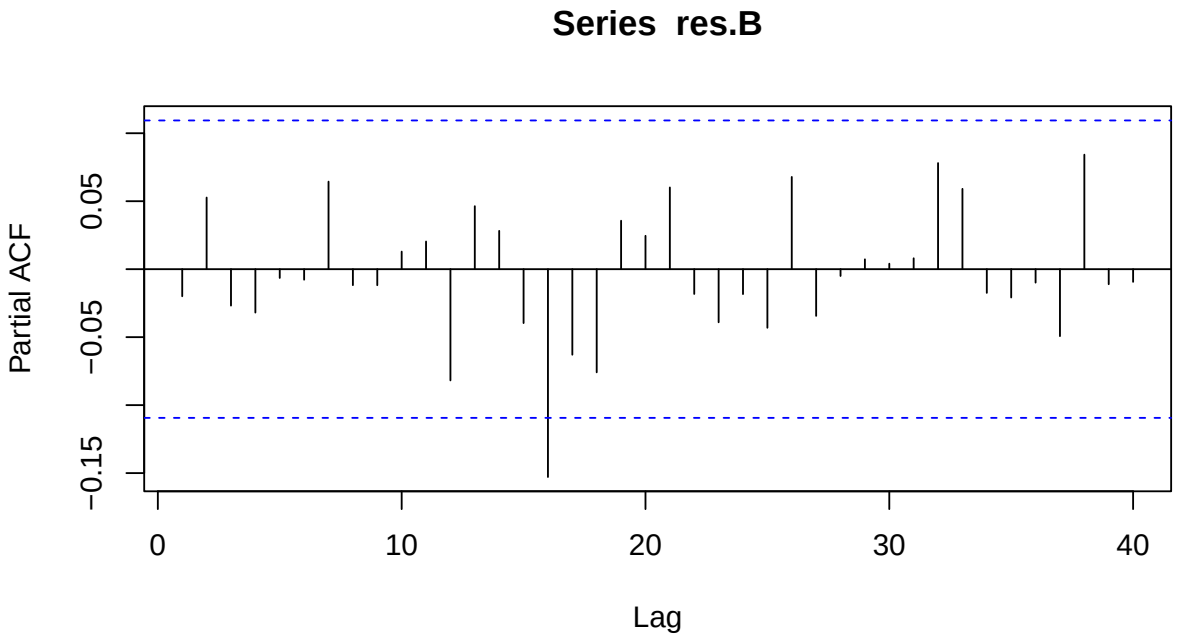
3.2.2 Model B:

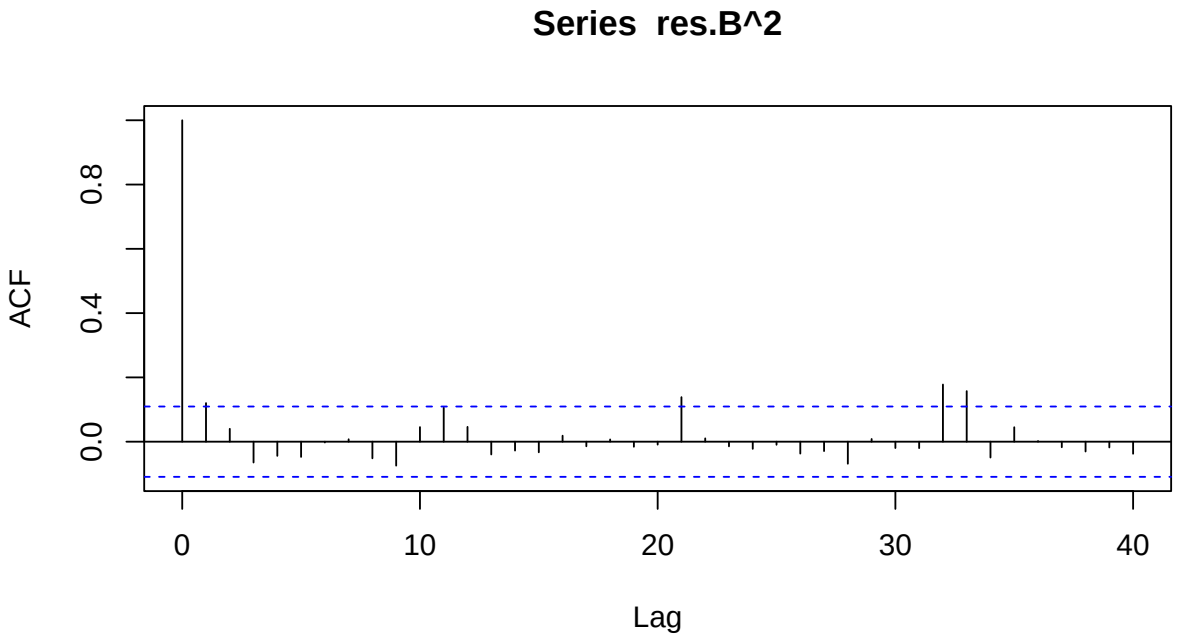

Histogram of res.B
Density
0.0
0.2
0.4
0.6
0.8
1.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


res.B
1.5
0.5
−0.5
−1.5
0
50
100
150
200
250
300
Time


Normal Q−Q Plot for Model A
Sample Quantiles
Theoretical Quantiles
−1.5
−0.5
0.5
1.5
−3
−2
−1
0
1
2
3


Series res.B
ACF
Lag
0.0
0.4
0.8
0
10
20
30
40





Model B had similar plots to Model A. In order to formally check if it is a good model, we move on to diagnostics tests.

```
# diagnostics checks
shapiro.test(res.B)
```

```
##
##  Shapiro-Wilk normality test
```

```
##
## data:  res.B
## W = 0.97, p-value = 2e-05
```

```
Box.test(res.B, lag = 18, type = c("Box-Pierce"), fitdf = 4)
```

```
##
##  Box-Pierce test
##
## data:  res.B
## X-squared = 17, df = 14, p-value = 0.3
```

```
Box.test(res.B, lag = 18, type = c("Ljung-Box"), fitdf = 4)
```

```
##
##  Box-Ljung test
##
## data:  res.B
## X-squared = 18, df = 14, p-value = 0.2
```

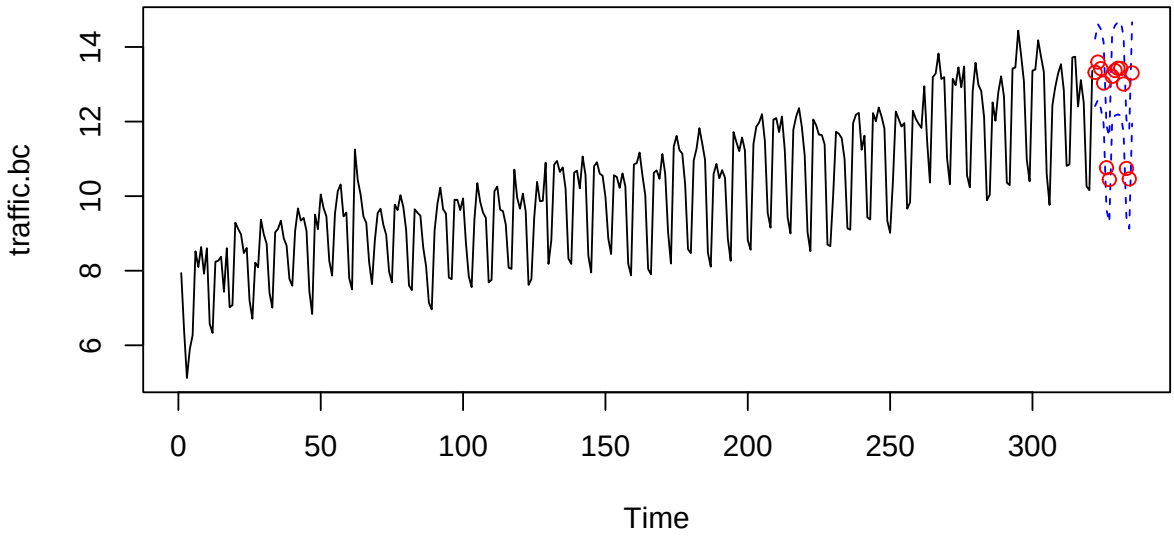
```
Box.test((res.B)^2, lag = 18, type = c("Ljung-Box"), fitdf = 0)
```

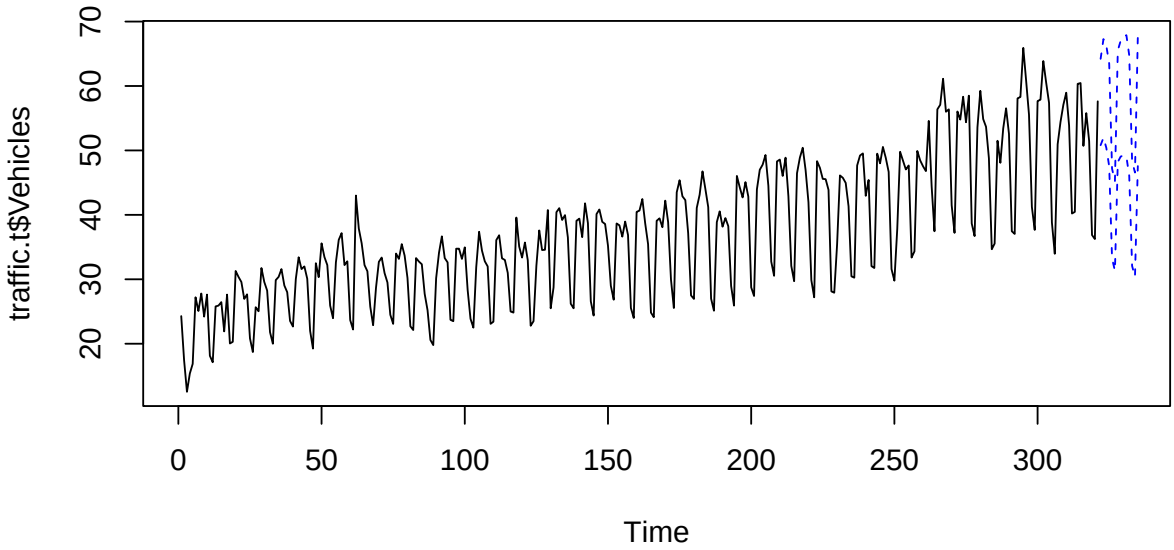
```
##
##  Box-Ljung test
##
## data:  (res.B)^2
## X-squared = 17, df = 18, p-value = 0.5
```

```
ar(res.B, aic = TRUE, order.max = NULL, method = c("yule-walker"))
```

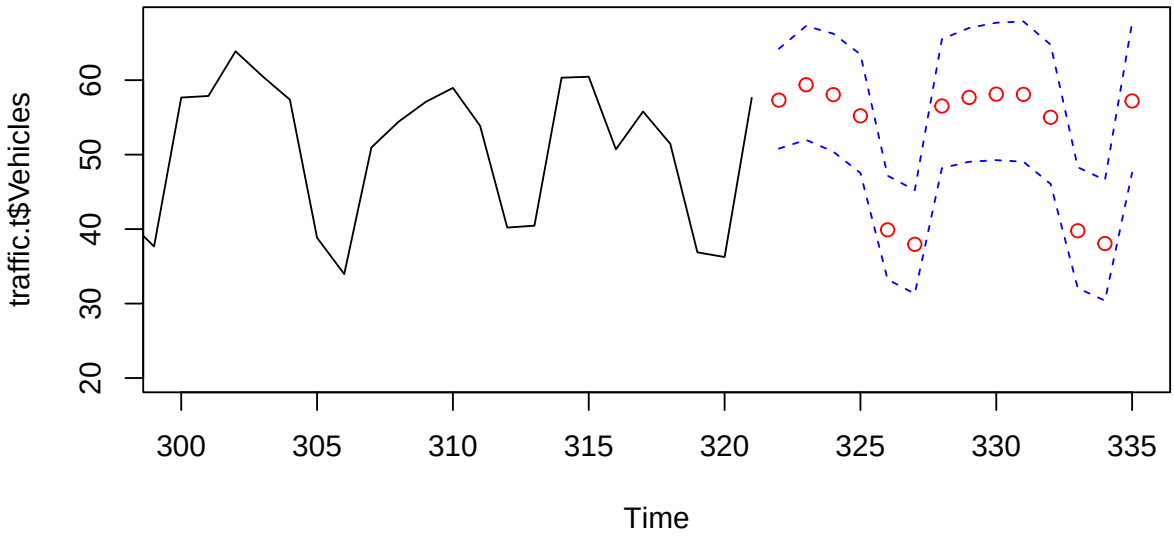
```
##
## Call:
## ar(x = res.B, aic = TRUE, order.max = NULL, method = c("yule-walker"))
##
##
## Order selected 0  sigma^2 estimated as  0.199
```

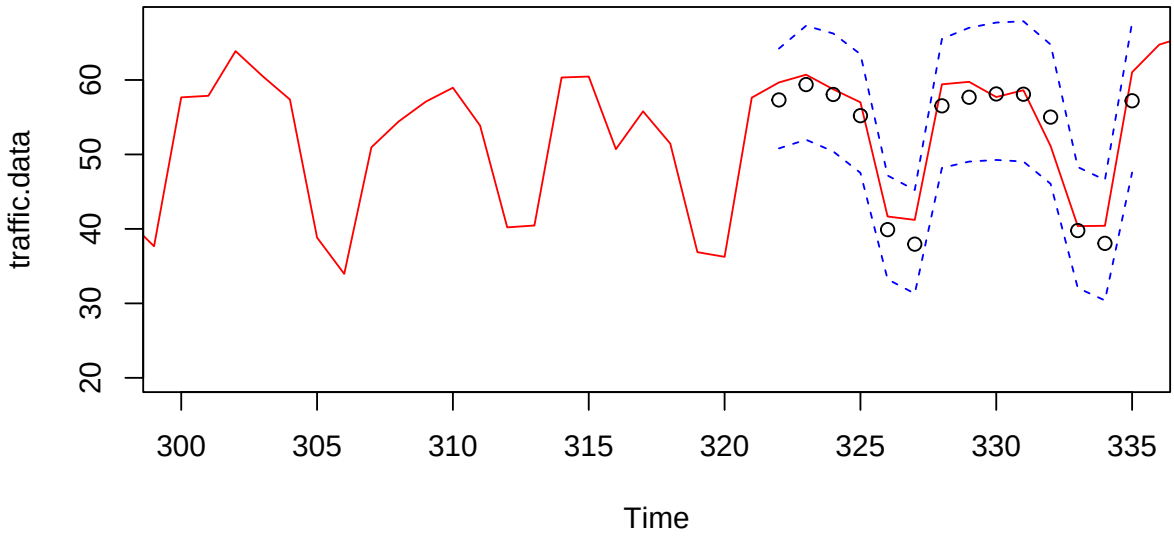
Like model A, model B passed all tests except for the test for normality. Under the assumption that the residuals are non-normal due to outliers, I decided to move on to forecasting using model A since it passed all other diagnostics checks. Using the model, we plot the forecasts for daily vehicle average onto the graph of the Box-Cox transformed data.





The confidence interval of the forecasts are given by the blue dotted line. The model's predictions are given by the red circles. From the plot, we can see that it seems to follow the trend of the data accurately. We then plot the predictions onto the validation set, zooming in for a clearer picture.





As shown in the forecast plots, the model is somewhat accurate in the prediction of mean vehicle congestions in this respective junction over the next few days after the training set ended.

4 Conclusions

The goal of this project was to forecast daily mean vehicle congestion in a highway junction. These goals were achieved using model A: $(1-0.319_{(0.098)}B)(1-B)Y_t = (1-0.812_{(0.068)}B)(1-0.975_{(0.068)}B^7)(1+0.161_{(0.067)}B^{14})Z_t$.

My results could have been more accurate if a better model with normal residuals was fit, however it was precise nonetheless. Thank you to Professor Feldman for teaching the material for this project, and for including a sample project which made the process very easy to understand.

5 References

Feldman, Raya. Lecture 15: Let's Do a Time Series Project! UC Santa Barbara, 2025.

6 Appendix

```
# read in traffic data set
traffic <- read.csv("traffic.csv")

# filter by Junction == 1
traffic_1 <- subset(traffic, Junction == 1)

# split DateTime variable into Date and Time
traffic_1$Date <- as.Date((substr(traffic_1$DateTime, 1, 10)))
traffic_1$Hour <- as.numeric(substr(traffic_1$DateTime, 12, 13))

# aggregate data by daily mean
daily_traffic <- aggregate(Vehicles ~ Date, data = traffic_1, FUN = mean)

# convert subsetted data into time series object
traffic_ts <- ts(daily_traffic$Vehicles, frequency = 1)

# plot time series
plot.ts(traffic_ts,
        main = "Plot of Time Series of Daily Mean Vehicles",
        sub = "Starting from November 1st, 2015",
        xlab = "Day",
        ylab = "Mean Vehicles")

nt = length(daily_traffic$Vehicles)
fit <- lm(daily_traffic$Vehicles ~ as.numeric(1:nt))

# add trend line
abline(fit, col = "red")
# add constant mean line
abline(h=mean(daily_traffic$Vehicles), col = "blue")

# partition data into model training and model validation set
traffic.t = daily_traffic[60:380, ]
traffic.test = daily_traffic[381:410, ]
traffic.data = daily_traffic[60:410, ]$Vehicles

# plot the training model
plot.ts(traffic.t$Vehicles)
fit.t <- lm(traffic.t$Vehicles ~ as.numeric(1:length(traffic.t$Vehicles)))
abline(fit.t, col = "red")
```

```r
abline(h=mean(traffic.t$Vehicles), col = "blue")

# histogram of Traffic Data in training set
hist(traffic.t$Vehicles, col = "light blue", main = "Histogram")

# acf of Traffic Data in training set
acf(traffic.t$Vehicles, lag.max = 100, main = "ACF of Traffic Data")

# perform box-cox transformation and extract lambda
bcTransform <- boxcox(traffic.t$Vehicles ~ as.numeric(1:length(traffic.t$Vehicles)))

lambda = bcTransform$x[which(bcTransform$y == max(bcTransform$y))]

traffic.bc <- (1/lambda) * (traffic.t$Vehicles^lambda-1)
traffic.log <- log(traffic.t$Vehicles)

# time series plots of log and box-cox
par(mfrow=c(3,1), mar = c(4,4,2,2))
plot.ts(traffic.t$Vehicles)
plot.ts(traffic.bc)
plot.ts(traffic.log)

# plot histograms of log and box-cox
par(mfrow = c(1,2))
hist(traffic.log,
     col = "light blue", xlab = "",
     main = "Histogram; ln(U_t)")
hist(traffic.bc,
     col = "light blue", xlab = "",
     main = "Histogram; bc(U_t)")

# produce decomposition of transformed data
traffic.t$bc <- traffic.bc
y <- ts(traffic.t$bc, frequency = 30)
decomp <- decompose(y)
plot(decomp)

# variance of original bc data
var.bc <- var(traffic.bc)

# differencing at lag 7
traffic.bc_7 <- diff(traffic.bc, lag = 7)
plot.ts(traffic.bc_7, main = "bc(U_t) differenced at lag 7")
fit.7 <- lm(traffic.bc_7 ~ as.numeric(1:length(traffic.bc_7)))
abline(fit.7, col = "red")
abline(h = mean(traffic.bc_7), col = "blue")

# variance of bc data differenced at 7
var.bc_7 <- var(traffic.bc_7)

# differencing at lag 1 to remove trend
traffic.stat <- diff(traffic.bc_7, lag = 1)
plot.ts(traffic.stat, main = "bc(U_t) differenced at lag 7 & lag 1")
```

```r
fit.stat <- lm(traffic.stat ~ as.numeric(1:length(traffic.stat)))
abline(fit.stat, col = "red")
abline(h = mean(traffic.stat, col = "blue"))

# variance of bc data differenced at 7 and 1
var.stat <- var(traffic.stat)

# acf of Box-Cox Transformed data
acf(traffic.bc, lag.max = 100, main = "ACF of bc(U_t)")

# acf of Box-Cox Transformed data differenced at lag 7
acf(traffic.bc_7, lag.max = 100, main = "ACF of bc(U_t), differenced at lag 7")

# acf of Box-Cox Transformed data differenced at lag 7 & 1
acf(traffic.stat, lag.max = 100, main = "ACF of bc(U_t), differenced at lag 7 & 1")

# histogram of Box-Cox transformed data
hist(traffic.bc,
     col = "light blue", xlab = "",
     main = "Histogram; bc(U_t)")

# histogram of Box-Cox transformed data differenced at lag 7 and 1
hist(traffic.stat,
     col = "light blue", xlab = "",
     main = "Histogram; bc(U_t) differenced at lags 7 & 1")

# histogram of Box-Cox transformed data differenced at lag 7 and 1, with normal curve
hist(traffic.stat, density = 20, breaks = 20,
     col = "blue", xlab = "", prob = TRUE)
curve(dnorm(x,mean(traffic.stat), sqrt(var(traffic.stat))), add = TRUE)

# acf and pacf of stationary data for candidate models
acf(traffic.stat, lag.max = 40, main = "ACF of bc(U_t), differenced at lag 7 & 1")
pacf(traffic.stat, lag.max = 40, main = "PACF of bc(U_t), differenced at lag 7 & 1")

# candidate model testing
arima(traffic.bc, order = c(1,1,1), seasonal = list(order = c(1,1,1), period = 7), method = "ML")
arima(traffic.bc, order = c(2,1,1), seasonal = list(order = c(1,1,1), period = 7), fixed = c(NA, 0, NA,
arima(traffic.bc, order = c(3,1,1), seasonal = list(order = c(1,1,1), period = 7), fixed = c(NA, 0, 0,
arima(traffic.bc, order = c(1,1,1), seasonal = list(order = c(2,1,1), period = 7), fixed = c(NA, NA, NA
arima(traffic.bc, order = c(2,1,1), seasonal = list(order = c(2,1,1), period = 7), fixed = c(NA, 0, NA,
arima(traffic.bc, order = c(1,1,1), seasonal = list(order = c(0,1,1), period = 7), method = "ML")
arima(traffic.bc, order = c(1,1,1), seasonal = list(order = c(0,1,2), period = 7), method = "ML")

# store models into variables
fit.A <- arima(traffic.bc, order = c(1,1,1), seasonal = list(order = c(1,1,1), period = 7), method = "M
fit.B <- arima(traffic.bc, order = c(1,1,1), seasonal = list(order = c(0,1,2), period = 7), method = "M

# plot residuals for diagnostic checking to check normality
res.A <- residuals(fit.A)
hist(res.A, density = 20, breaks = 20, col = "blue", xlab = "", prob = TRUE)
curve(dnorm(x, mean(res.A), sqrt(var(res.A))), add = TRUE)
plot.ts(res.A)
```

```r
fit.res.A <- lm(res.A ~ as.numeric(1:length(res.A)))
abline(fit.res.A, col = "red")
abline(h = mean(res.A), col = "blue")

# qq norm plots
qqnorm(res.A, main = "Normal Q-Q Plot for Model A")
qqline(res.A, col = "blue")

# acf and pacf of residual
acf(res.A, lag.max = 40)
pacf(res.A, lag.max = 40)

# acf of residual^2
acf(res.A^2, lag.max = 40)

# diagnostics checks
shapiro.test(res.A)
Box.test(res.A, lag = 18, type = c("Box-Pierce"), fitdf = 4)
Box.test(res.A, lag = 18, type = c("Ljung-Box"), fitdf = 4)
Box.test((res.A)^2, lag = 18, type = c("Ljung-Box"), fitdf = 0)
ar(res.A, aic = TRUE, order.max = NULL, method = c("yule-walker"))

# plot residuals for diagnostic checking to check normaility
res.B <- residuals(fit.B)
hist(res.B, density = 20, breaks = 20, col = "blue", xlab = "", prob = TRUE)
curve(dnorm(x, mean(res.B), sqrt(var(res.B))), add = TRUE)
plot.ts(res.B)
fit.res.B <- lm(res.B ~ as.numeric(1:length(res.B)))
abline(fit.res.B, col = "red")
abline(h = mean(res.B), col = "blue")

# qq norm plots
qqnorm(res.B, main = "Normal Q-Q Plot for Model A")
qqline(res.B, col = "blue")

# acf and pacf of residual
acf(res.B, lag.max = 40)
pacf(res.B, lag.max = 40)

# acf of residual^2
acf(res.B^2, lag.max = 40)

# diagnostics checks
shapiro.test(res.B)
Box.test(res.B, lag = 18, type = c("Box-Pierce"), fitdf = 4)
Box.test(res.B, lag = 18, type = c("Ljung-Box"), fitdf = 4)
Box.test((res.B)^2, lag = 18, type = c("Ljung-Box"), fitdf = 0)
ar(res.B, aic = TRUE, order.max = NULL, method = c("yule-walker"))

# forecast using model A
forecast(fit.A)

# plot forecasts onto transformed data
```

```r
pred.tr <- predict(fit.A, n.ahead = 14)
Utr = pred.tr$pred + 2*pred.tr$se
Ltr = pred.tr$pred - 2*pred.tr$se
ts.plot(traffic.bc, xlim = c(1, length(traffic.bc)+ 14), ylim = c(min(traffic.bc), max(Utr)))
lines(Utr, col = "blue", lty = "dashed")
lines(Ltr, col = "blue", lty = "dashed")
points((length(traffic.bc)+1):(length(traffic.bc)+14), pred.tr$pred, col = "red")

# produce graph with forecasts on original data, invert transformation
pred.orig <- (pred.tr$pred * lambda + 1) ^ (1/lambda)
U = (Utr * lambda + 1) ^ (1/lambda)
L = (Ltr * lambda + 1) ^ (1/lambda)
ts.plot(traffic.t$Vehicles, xlim = c(1, length(traffic.t$Vehicles)+12), ylim = c(min(traffic.t$Vehicles)
lines(U, col = "blue", lty = "dashed")
lines(L, col = "blue", lty = "dashed")
points((length(daily_traffic$Vehicles)+1):(length(daily_traffic$Vehicles)+14), pred.orig, col = "red")

# zoom in on graph starting at 300
ts.plot(traffic.t$Vehicles, xlim = c(300, length(traffic.t$Vehicles)+14), ylim = c(20, max(U)))
lines(U, col = "blue", lty = "dashed")
lines(L, col = "blue", lty = "dashed")
points((length(traffic.t$Vehicles)+1):(length(traffic.t$Vehicles)+14),pred.orig,col="red")

# plot zoomed forecasts and true values
ts.plot(traffic.data, xlim = c(300, length(traffic.t$Vehicles)+14), ylim = c(20, max(U)), col = "red")
lines(U, col = "blue", lty = "dashed")
lines(L, col = "blue", lty = "dashed")
points((length(traffic.t$Vehicles)+1):(length(traffic.t$Vehicles)+14),pred.orig,col="black")
```