Pstat 174 Boston Crime Project

Riley Mault

February 18, 2020

1. Abstract

Crime incident reports are provided by Boston Police Department (BPD) to document the initial details surrounding an incident to which BPD officers respond. This is a dataset containing records capturing the type of incident as well as when and where it occurred. For this time series project, I will be focusing exclusively on when crimes occurred. Records begin in June 14, 2015 and continue to September 3, 2018.

I will be addressing questions such as: Does the frequency of crimes have any pattern? If the frequency of crimes does have a pattern, why is that? Is it possibly to forecast the daily frequency of crimes?

To address these questions, I plotted the data ordered by time, analyzed auto-correlation plots, and developed seasonal ARIMA models to predict/forecast future values.

2.

Introduction

My goal of this daily Boston crime report dataset is to find and observe frequencies at which crimes occur. In observing those frequencies, I will build an ARIMA model to forecast the frequencies of future crime reports.

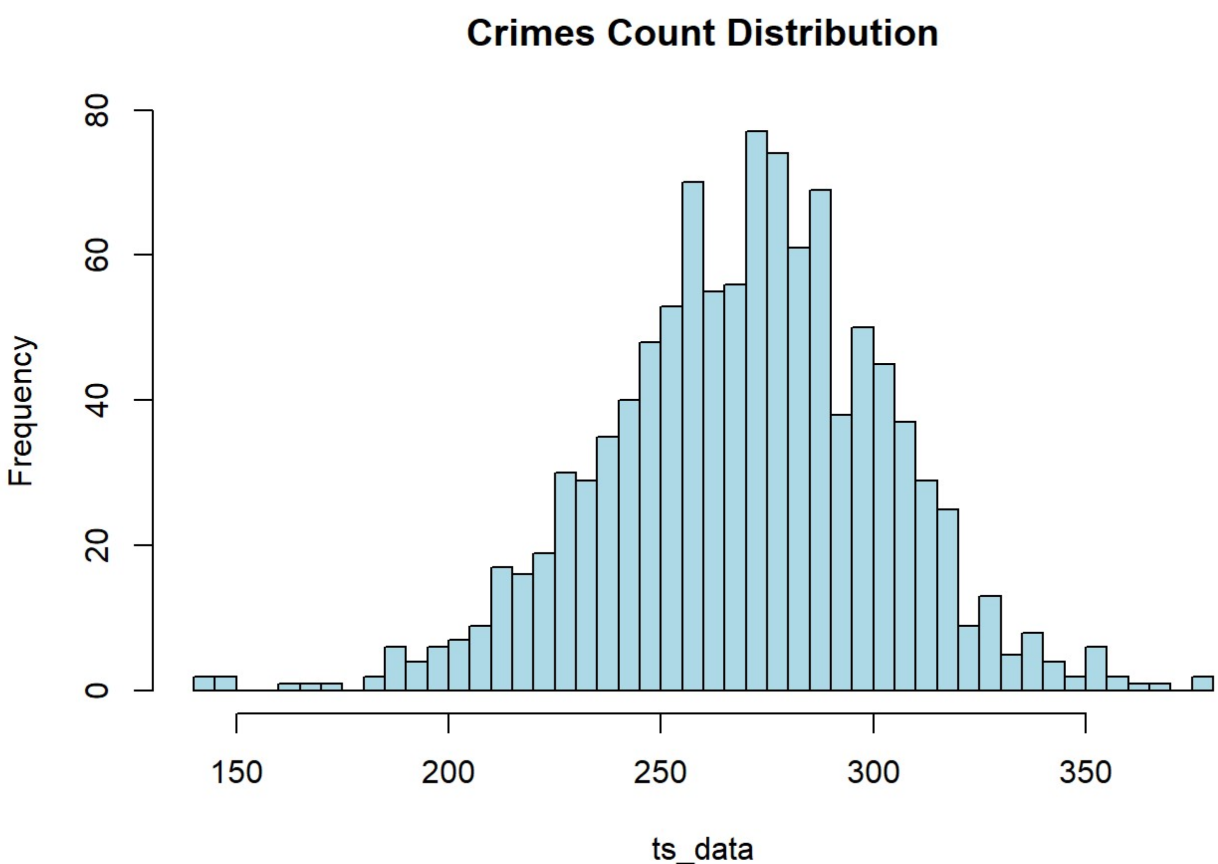
Importing and Cleaning Data

```r
# import dataset
crime_df = read.csv('crime.csv')
attach(crime_df)
```

```r
# sorting data by date of crime
crime_df = crime_df  %>% separate(OCCURRED_ON_DATE, c("Date", "Time"), sep = " ") %>%
mutate(Date = ymd(Date))
crime_df$Date = as.Date(crime_df$Date)
crime_df = crime_df[order(crime_df$Date),]
```

Distribution of the Counts of Crimes by Date

```
ts_data_full = ts(table(crime_df$Date))

# split up data into train/test split
ts_data = ts(ts_data_full[c(1:1067)])
ts_data.test = ts(ts_data_full[c(1068:1077)])

# histogram of our crime data
hist(ts_data, 50, col='light blue', main='Crimes Count Distribution')
```



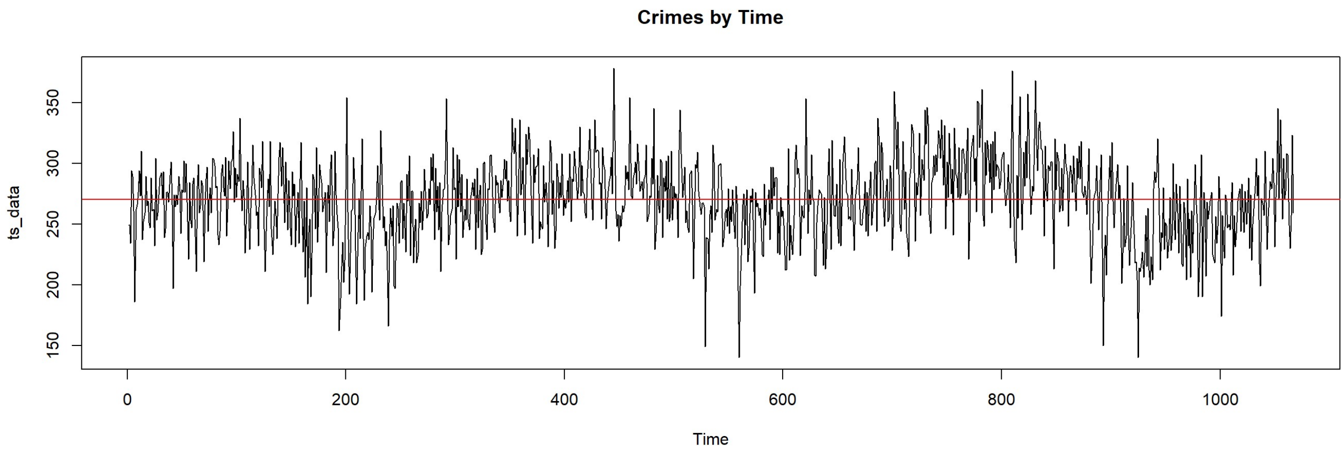

```
paste(c('Skewness is', skewness(ts_data)), collapse = ' ')
```

```
## [1] "Skewness is -0.22456247991086"
```

I separated the data into 50 bins and plotted a histogram. Most days, about 250 - 300 crimes occur. We can see that the data seems pretty normal with a very slight left skew. Therefore, a transformation isn't necessary.

Distribution of Crimes by Time

```
# Ploting timeseries plot
ts.plot(ts_data, main = "Crimes by Time")
abline(h=mean(ts_data), col='red')
```

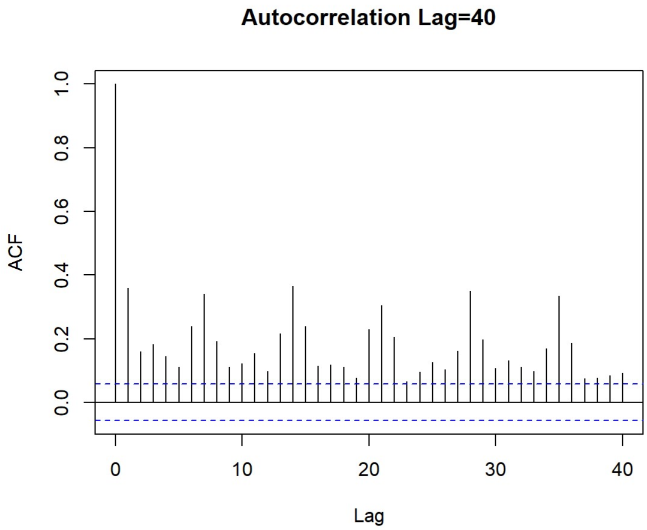


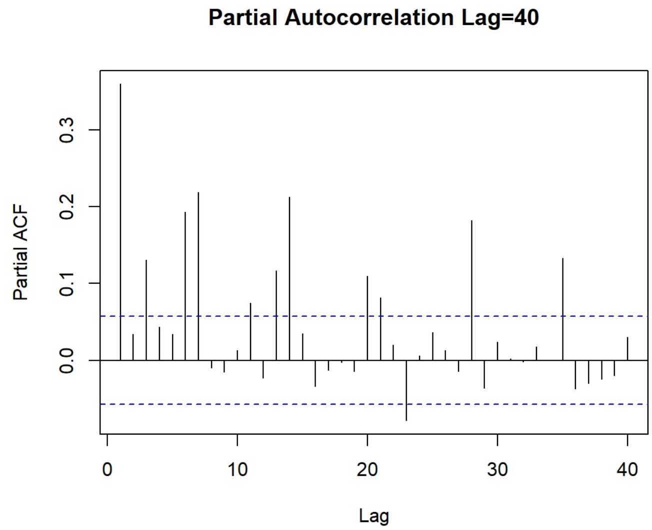

```
# Transformation of data but not needed/performed

# bcTransform<-boxcox(ts_data ~ as.numeric(1:length(ts_data)))
# bcTransform$x[which(bcTransform$y== max(bcTransform$y))]
# lambda = bcTransform$x[which(bcTransform$y== max(bcTransform$y))]
# lambda = bcTransform$x[which(bcTransform$y== max(bcTransform$y))]
# ts.bc = (1/lambda)*(ts_data^lambda-1)
# hist(ts.bc, 50, col='light blue', main='Crimes Count Distribution')
# shapiro.test(ts.bc)
```

From the chart above, there is noticeable seasonality that we will address in order to make our data stationary. We can see a repeating pattern (almost that of a sin wave) every 365 days.

ACF and PACF

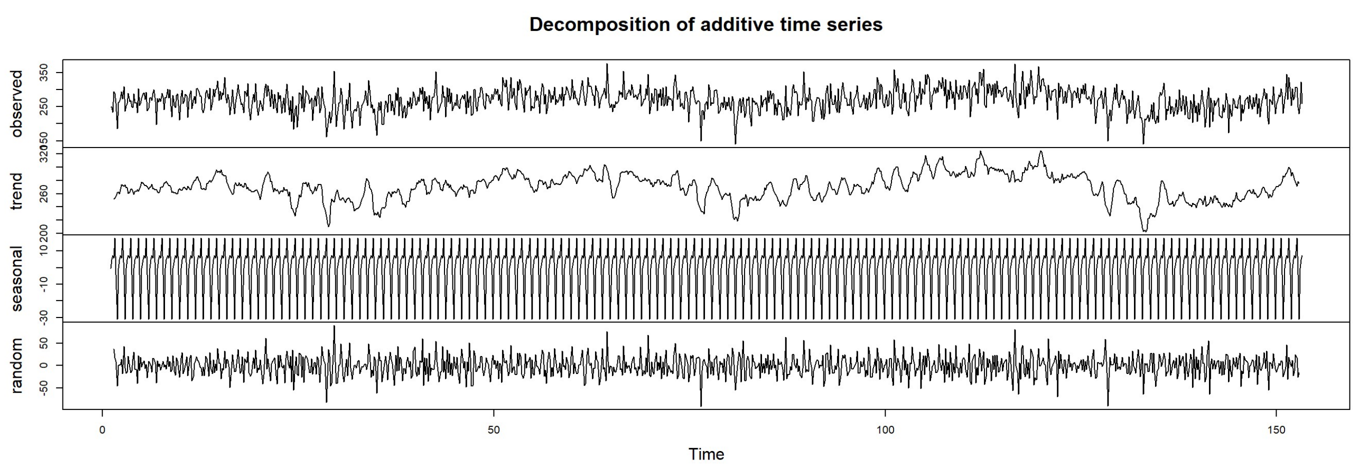
```
par(mfrow=c(1,2))

# ACF and PACF
acf(table(crime_df$Date), 40, main='Autocorrelation Lag=40')
pacf(table(crime_df$Date), 40, main='Partial Autocorrelation Lag=40')
```





In both the ACF and PACF plots, there are lag spikes every 7 lags. To approach this, I will difference the data by 7 lag units. The partial correlation shows many significant lags. We can conclude that crimes are correlated with yesterday and the same day in each previous week.

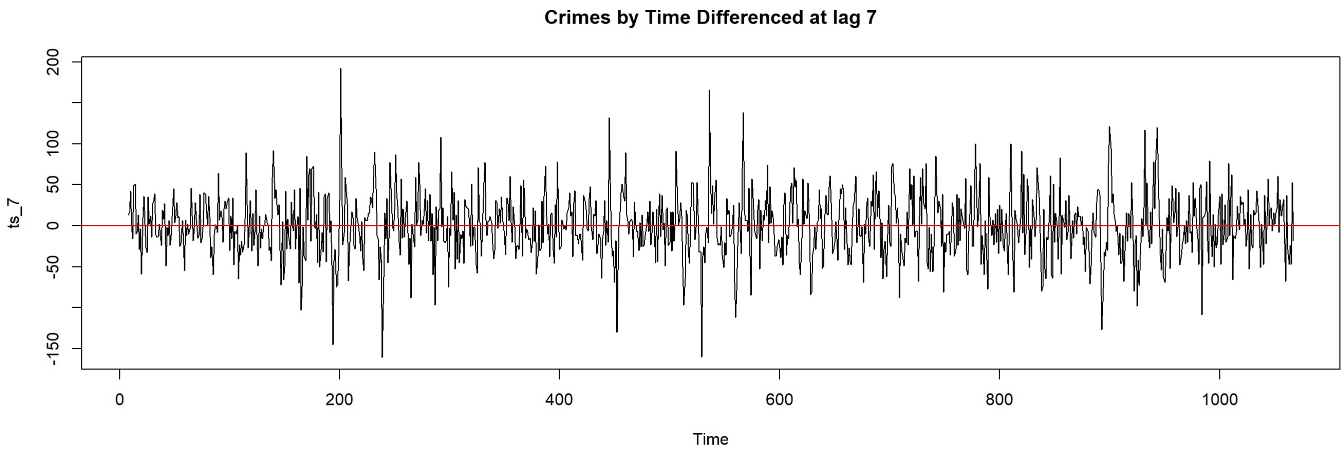
```
# Decompose ts to observe trend/seasonality
y = ts(as.ts(ts_data), frequency=7)
plot(decompose(y))
```



In decomposing the crime data, we notice a small amount of trend and noticeable seasonality. Again, this gives us more reason to difference the data to make it stationary.

Differencing Data

```
# Differencing the model by 7 to remove seasonality/trend
ts_7 = diff(ts_data, lag=7)

plot.ts(ts_7, main="Crimes by Time Differenced at lag 7")
abline(h=mean(ts_7), col='red')
```



Dickey-Fuller Test for Stationary

```
adf.test(ts_7)
```

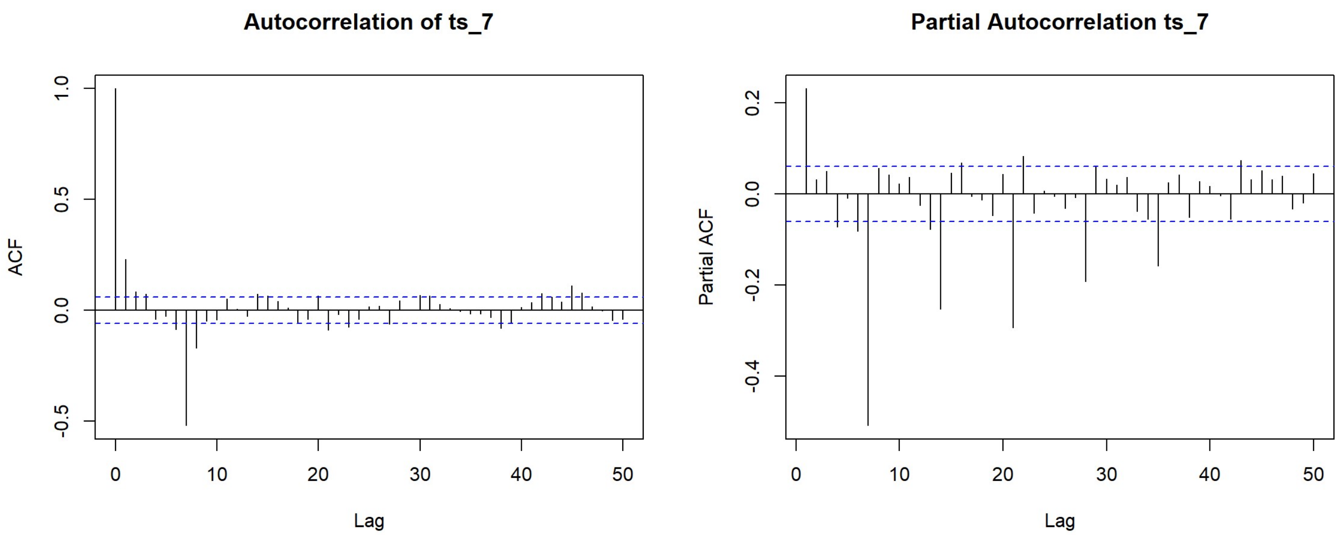
```
## Warning in adf.test(ts_7): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ts_7
## Dickey-Fuller = -11.633, Lag order = 10, p-value = 0.01
## alternative hypothesis: stationary
```

After differencing the data by 7, the time series graph looks much more stationary. The p-value of the Dickey-Fuller Test (smaller than .01) being less than .05 confirms the alternative hypothesis that the data is stationary.

Differenced ACF and PACF

```
par(mfrow=c(1,2))

acf(ts_7, 50, main='Autocorrelation of ts_7')
pacf(ts_7,50, main='Partial Autocorrelation ts_7')
```



After differencing our data, the ACF and PACF plots look much cleaner and easier to analyze. In the ACF plot the first few lags show significance. From this, we can possibly assign a q value of 2 or 3 for the non-seasonal component. There is another significant spike at 7 due to the seasonal component; so we assign Q=1.

In the PACF plot, there is one significant lag at lag 1. However, we might be better off having a complete MA model and having no AR components to the non-seasonal aspect. There are also continuous decreasing lags every 7 lags. From this, can attribute a seasonal AR component with order of at least 3 or 4.

ARIMA Models

```
# Model A
fit.A = arima(ts_data, order=c(0,1,3), seasonal = list(order = c(4,1,1), period = 7),
method="CSS")
fit.A
```

```
##
## Call:
## arima(x = ts_data, order = c(0, 1, 3), seasonal = list(order = c(4, 1, 1), period =
7),
##     method = "CSS")
##
## Coefficients:
##           ma1      ma2      ma3     sar1     sar2     sar3     sar4
##       -0.7326  -0.1575  -0.0236  -0.065  -0.0066  -0.1432  -0.0179
## s.e.   0.0310   0.0403   0.0307   0.032   0.0274   0.0308   0.0301
##          sma1
##       -0.9644
## s.e.   0.0090
##
## sigma^2 estimated as 734.4:  part log likelihood = -4996.88
```

```
AICc(arima(ts_data, order=c(0,1,3), seasonal = list(order = c(4,1,1), period = 7),  me
thod="ML"))
```

```
## [1] 9997.219
```

```
# Model B
fit.B = arima(ts_data, order=c(0,1,2), seasonal = list(order = c(4,1,1), period = 7),
method="CSS")
fit.B
```

```
##
## Call:
## arima(x = ts_data, order = c(0, 1, 2), seasonal = list(order = c(4, 1, 1), period =
7),
##     method = "CSS")
##
## Coefficients:
##           ma1      ma2     sar1     sar2     sar3     sar4     sma1
##       -0.7347  -0.1771  -0.0677  -0.0090  -0.1462  -0.0183  -0.9642
## s.e.   0.0303   0.0303   0.0319   0.0274   0.0305   0.0301   0.0090
##
## sigma^2 estimated as 734.8:  part log likelihood = -4997.18
```

```
AICc(arima(ts_data, order=c(0,1,2), seasonal = list(order = c(4,1,1), period = 7), met
hod="ML"))
```

```
## [1] 9995.625
```

After testing out many similar models to the ps and qs suggested from the ACF/PACF plots, these two models (Model A and Model B) resulted in the lowest AICc values and best fitting models. Model A has MA order 3 coefficients while Model B has MA order 2. Model B has a slightly lower AICc value by 2. To decide which model is better, we will run diagnostics on both.

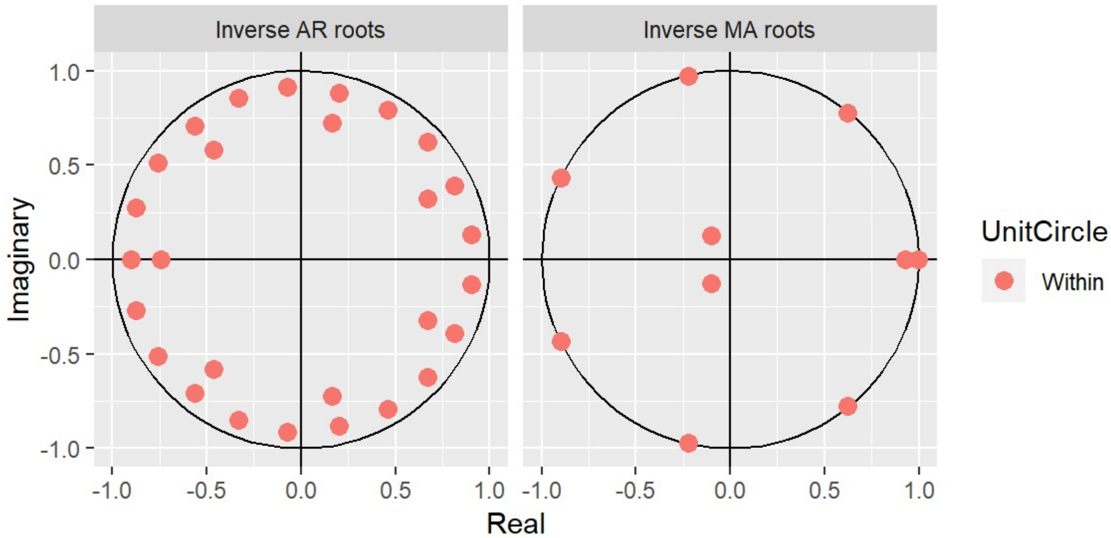
Diagnostic Checking for Model A

$$\Delta_7$$

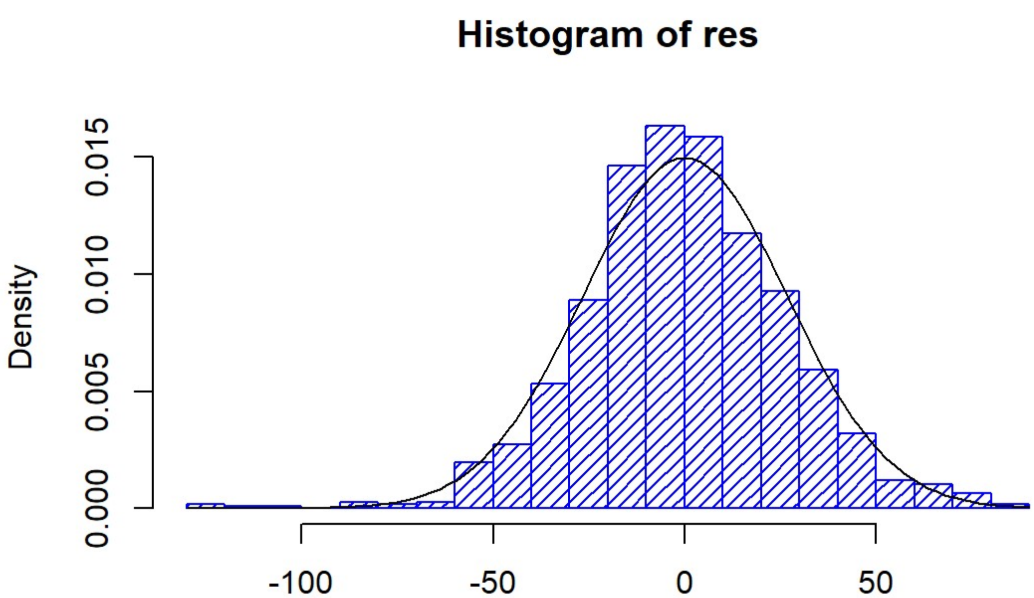
$$U_t$$

= $(1 - 0.7326B - 0.1575B^2 - 0.0236B^3)(1 - 0.9644B^7)Z_t$ +
$(0.065B^7 - 0.0066B^{14} - 0.1432B^{21} - 0.0179B^{28})X_t$

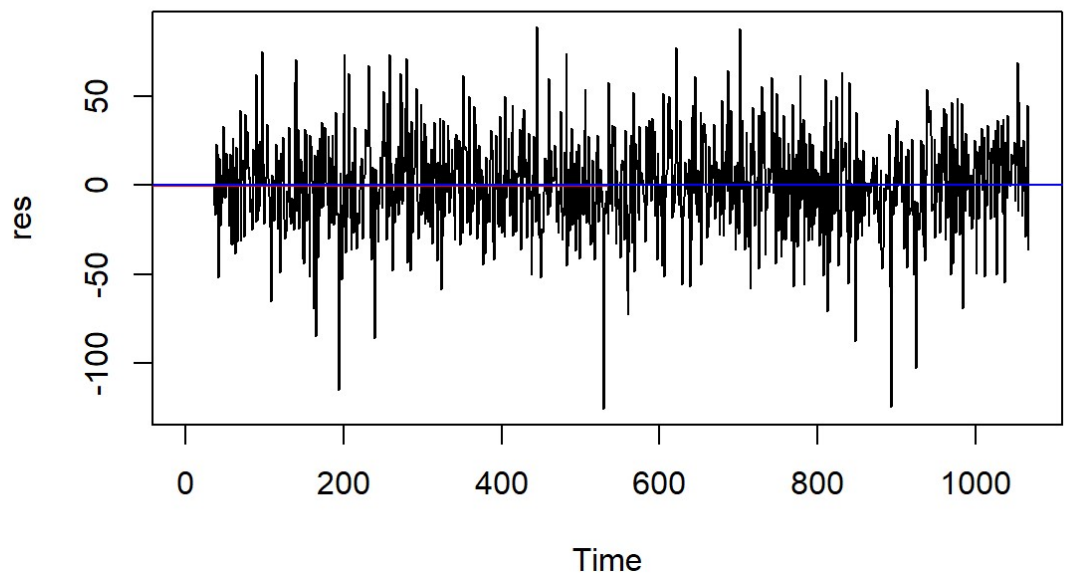
```
# Checking Invertibility of model A
autoplot(fit.A)
```



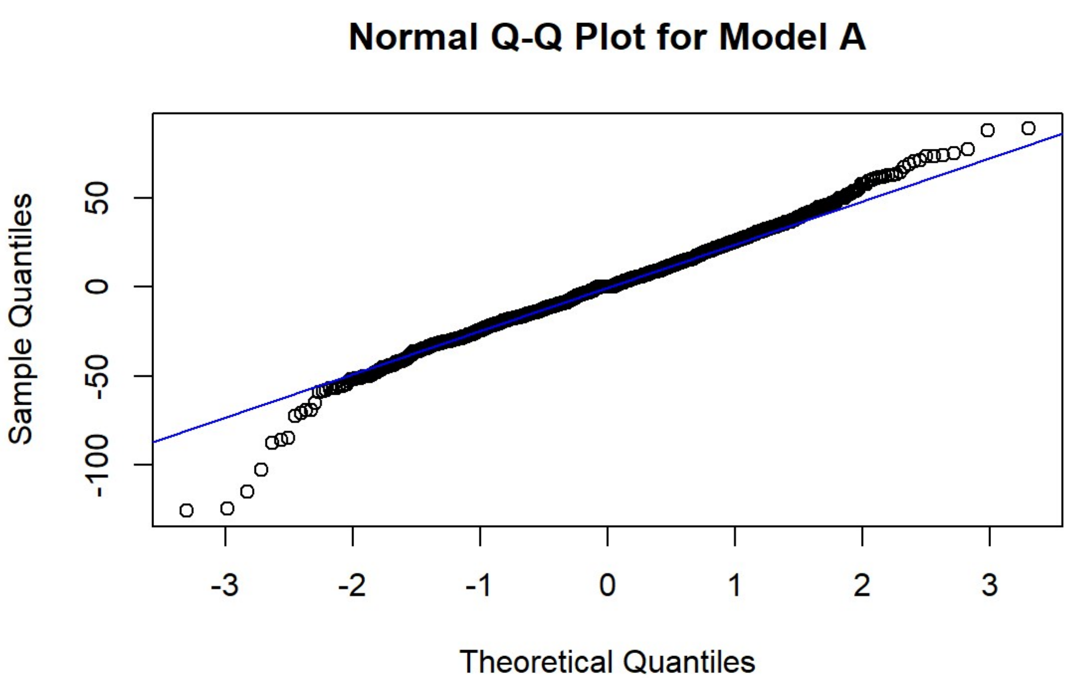

```
# residual hist
res = residuals(fit.A)
hist(res,density=20,breaks=20, col="blue", xlab="", prob=TRUE)
m = mean(res)
std = sqrt(var(res))
curve( dnorm(x,m,std), add=TRUE )
```



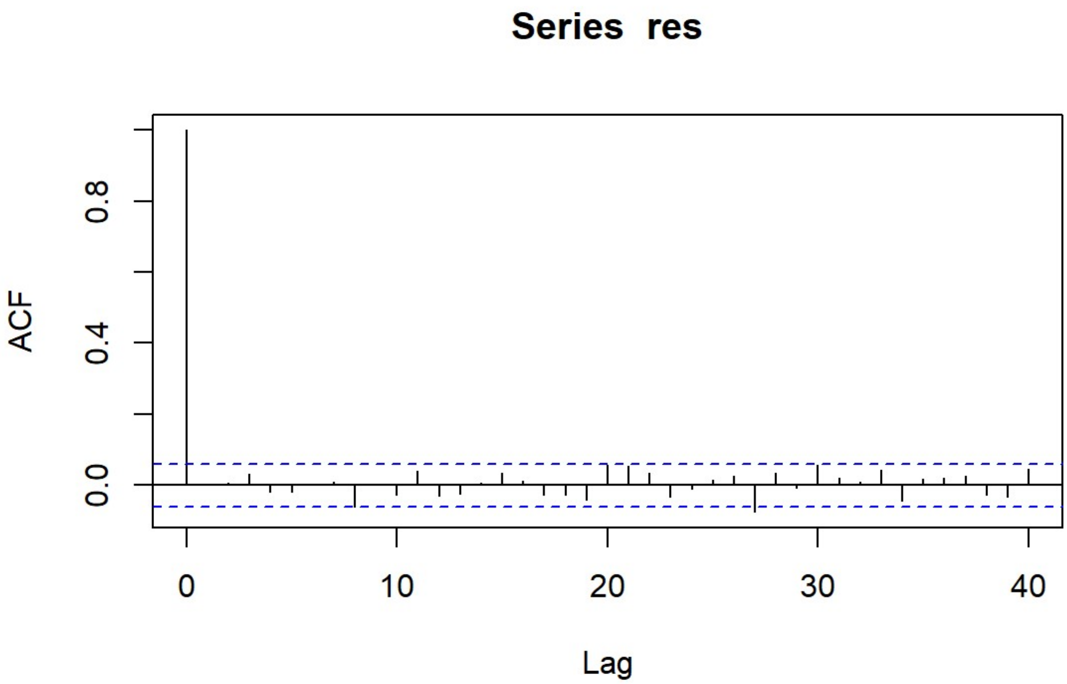
```
plot.ts(res)
fitt = lm(res ~ as.numeric(1:length(res))); abline(fitt, col="red")
abline(h=mean(res), col="blue")
```



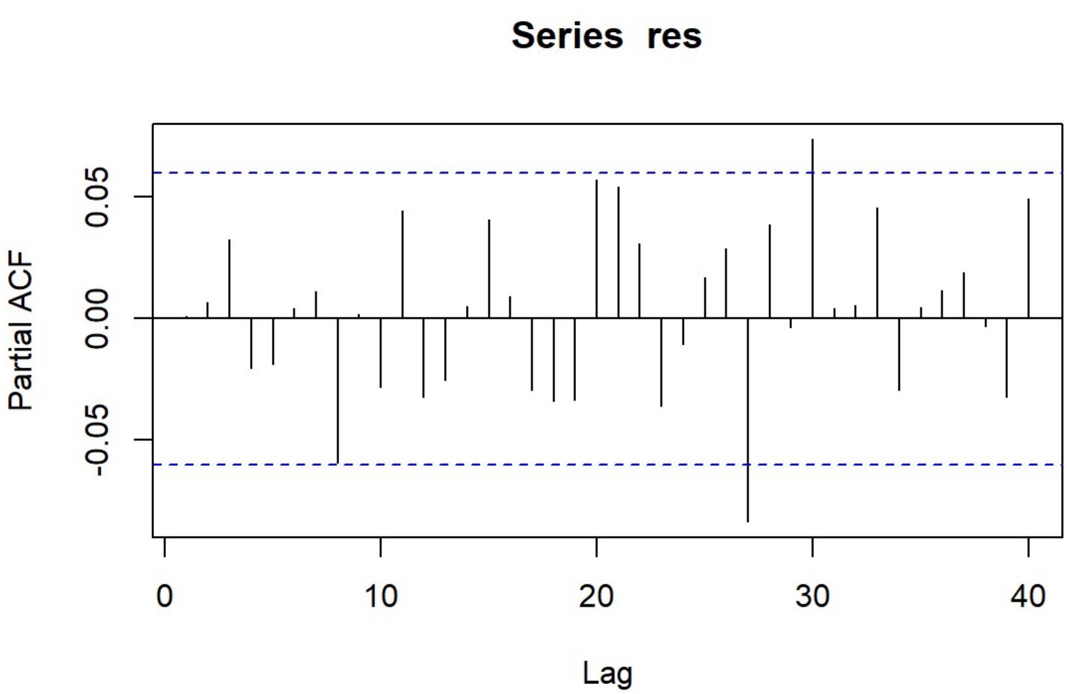

```
# Normal Q-Q to check for normality of res
qqnorm(res,main= "Normal Q-Q Plot for Model A")
qqline(res,col="blue")
```



```
acf(res, lag.max=40)
```




```
pacf(res, lag.max=40)
```




```
# Tests to check fit
df = 3
shapiro.test(res)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res
## W = 0.98705, p-value = 4.116e-08
```

```
Box.test(res, lag = 7, type = c("Box-Pierce"), fitdf= df)
```

```
##
##  Box-Pierce test
##
## data:  res
## X-squared = 2.0956, df = 4, p-value = 0.7182
```

```
Box.test(res, lag = 7, type = c("Ljung-Box"), fitdf= df)
```

```
##
##  Box-Ljung test
##
## data:  res
## X-squared = 2.1069, df = 4, p-value = 0.7161
```

```
Box.test(res^2, lag = 7, type = c("Ljung-Box"), fitdf= 0)
```

```
##
##  Box-Ljung test
##
## data:  res^2
## X-squared = 12.243, df = 7, p-value = 0.09285
```

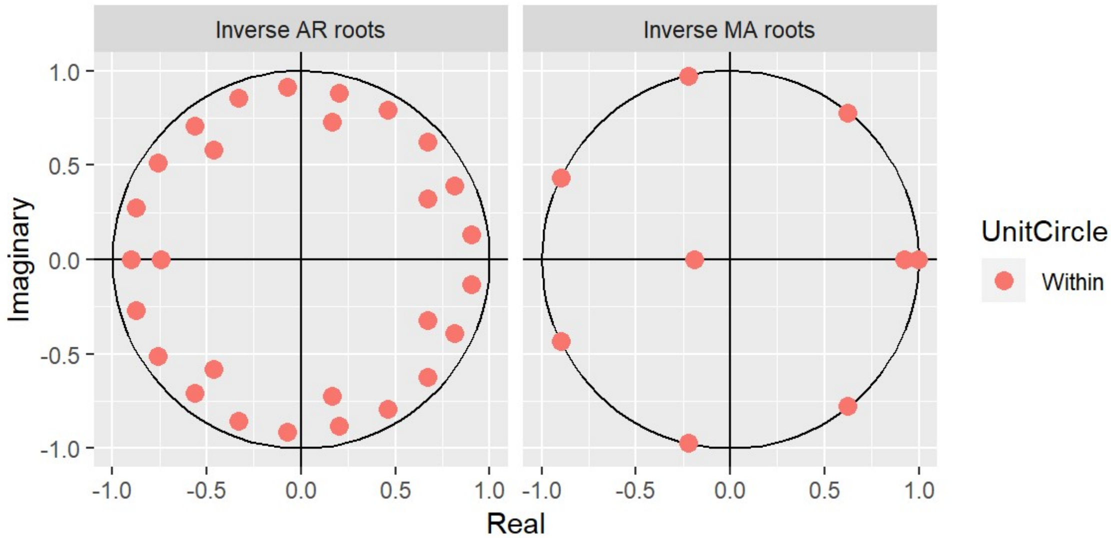
Diagnostic Checking for Model B

$$\Delta_7$$

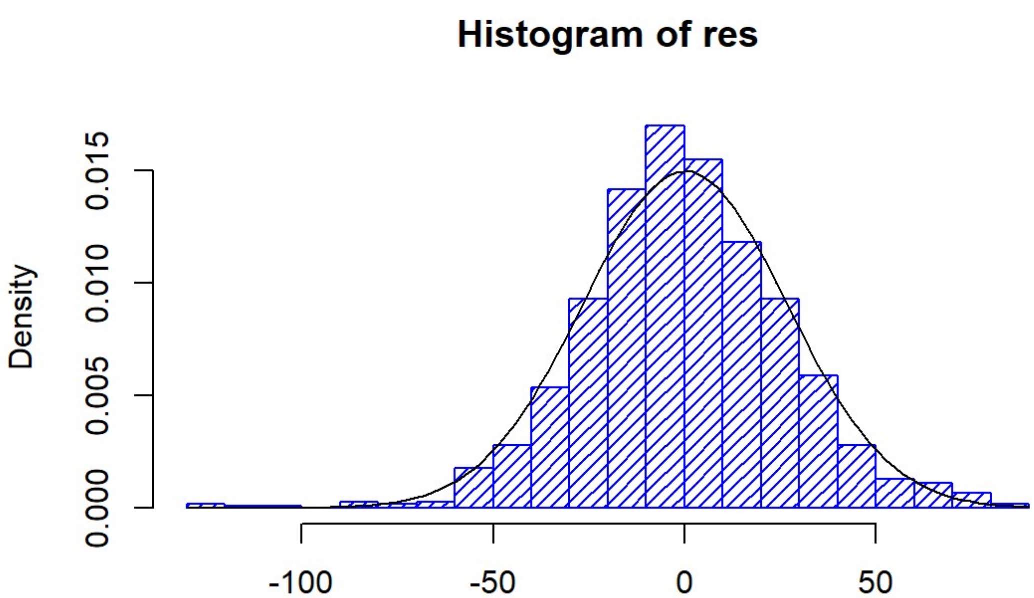
$$U_t$$

= $(1 - 0.7347B - 0.1771B^2)(1 - 0.9642B^7)Z_t$ + $(-0.0677B^7 - 0.0090B^{14} - 0.1462B^{21} - 0.0183B^{28})X_t$

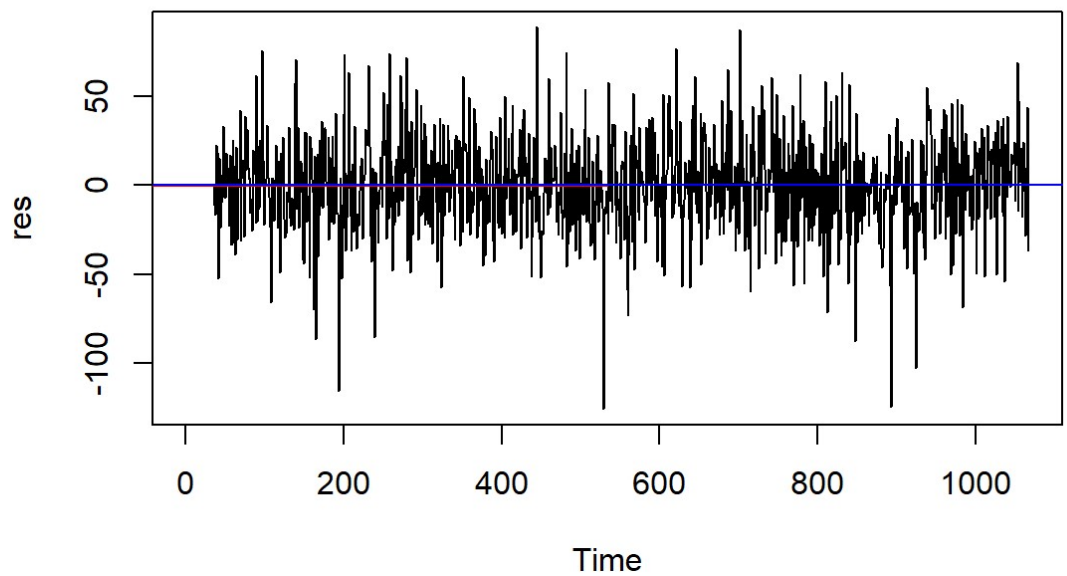
```
#Checking Invertibility of model B
autoplot(fit.B)
```



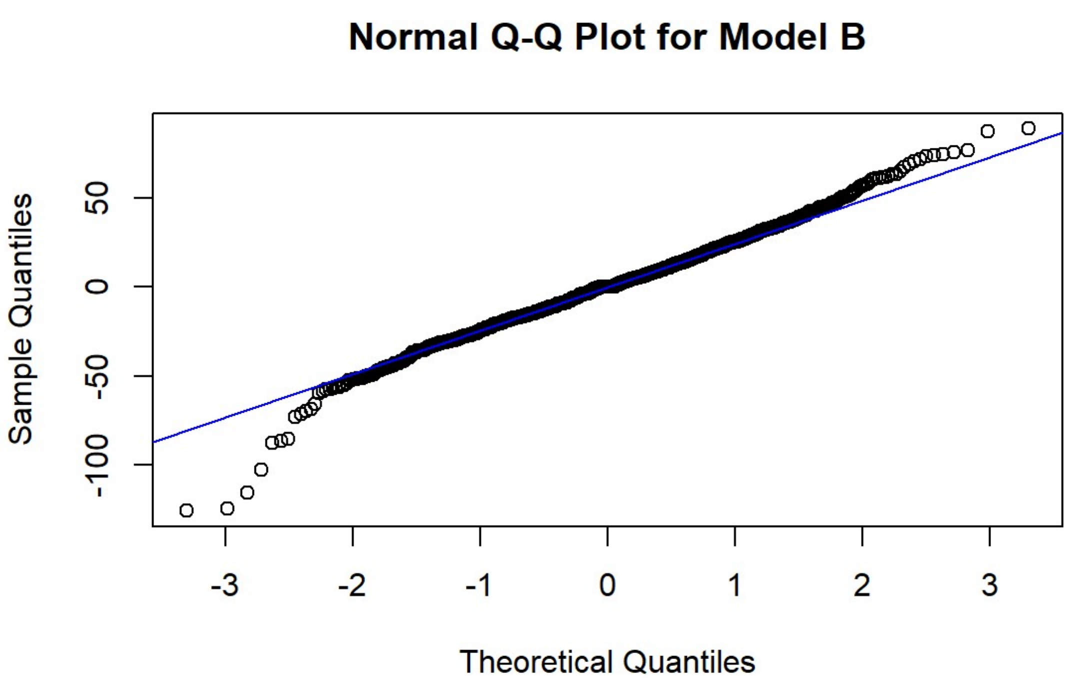

```r
# residual hist
res = residuals(fit.B)
hist(res,density=20,breaks=20, col="blue", xlab="", prob=TRUE)
m = mean(res)
std = sqrt(var(res))
curve( dnorm(x,m,std), add=TRUE )
```



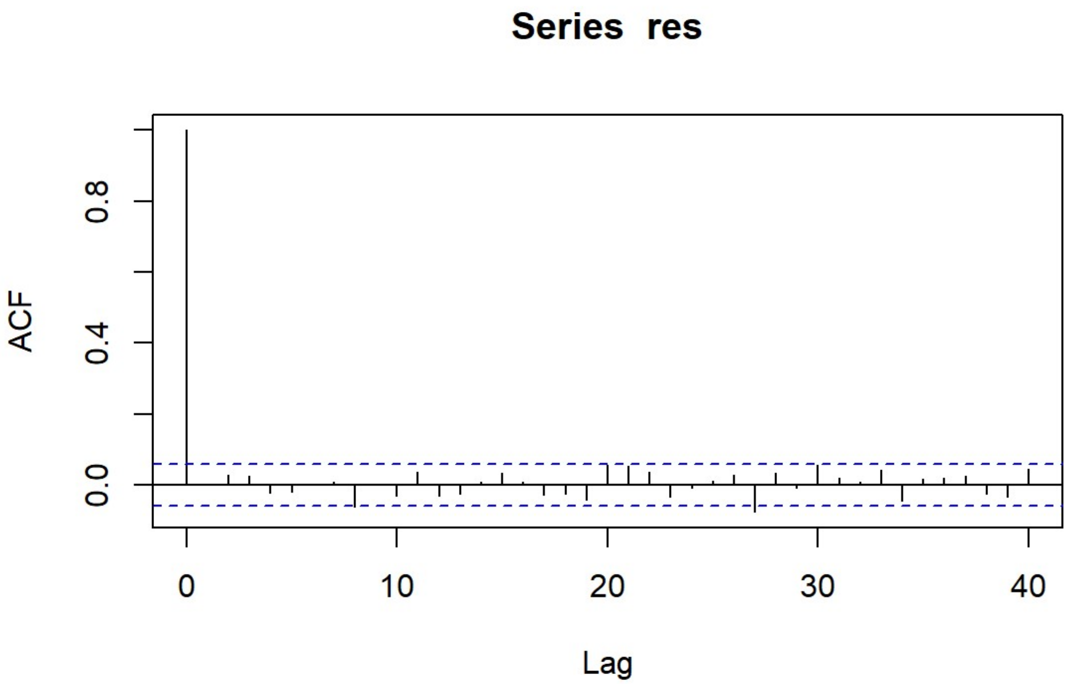
```
plot.ts(res)
fitt = lm(res ~ as.numeric(1:length(res))); abline(fitt, col="red")
abline(h=mean(res), col="blue")
```



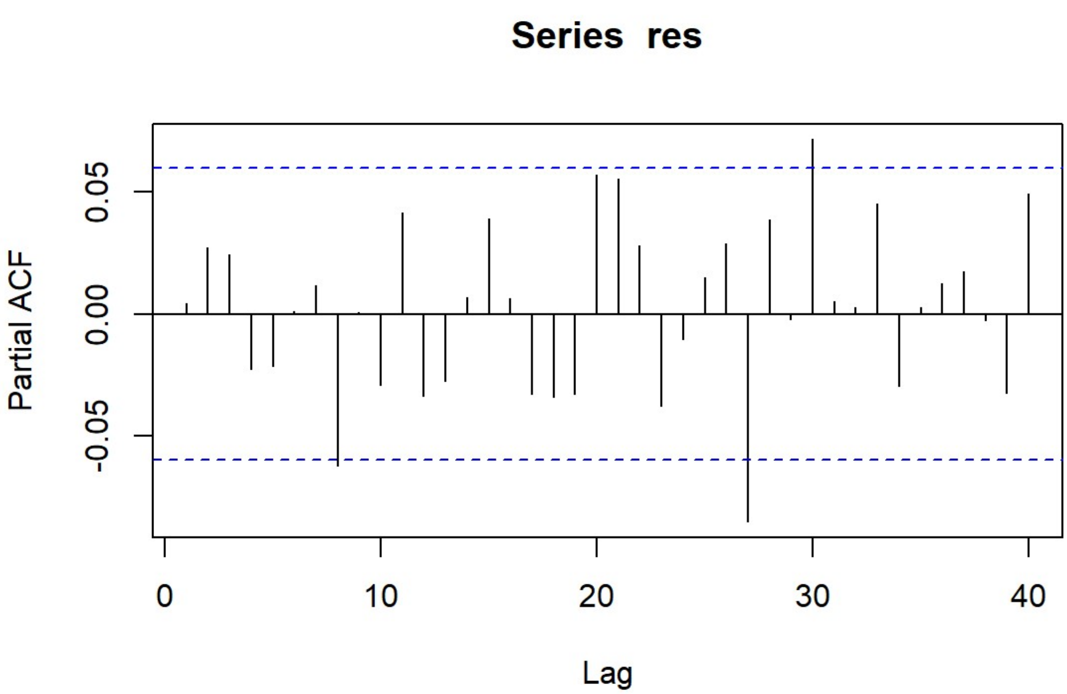

```
# Normal Q-Q to check for normality of res
qqnorm(res,main= "Normal Q-Q Plot for Model B")
qqline(res,col="blue")
```



```
acf(res, lag.max=40)
```




```
pacf(res, lag.max=40)
```




```
# Tests to check fit
df = 2
shapiro.test(res)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res
## W = 0.98707, p-value = 4.212e-08
```

```
Box.test(res, lag = 7, type = c("Box-Pierce"), fitdf= df)
```

```
##
##  Box-Pierce test
##
## data:  res
## X-squared = 2.4736, df = 5, p-value = 0.7805
```

```
Box.test(res, lag = 7, type = c("Ljung-Box"), fitdf= df)
```

```
##
##  Box-Ljung test
##
## data:  res
## X-squared = 2.4861, df = 5, p-value = 0.7786
```

```
Box.test(res^2, lag = 7, type = c("Ljung-Box"), fitdf= 0)
```

```
##
##  Box-Ljung test
##
## data:  res^2
## X-squared = 12.421, df = 7, p-value = 0.08754
```

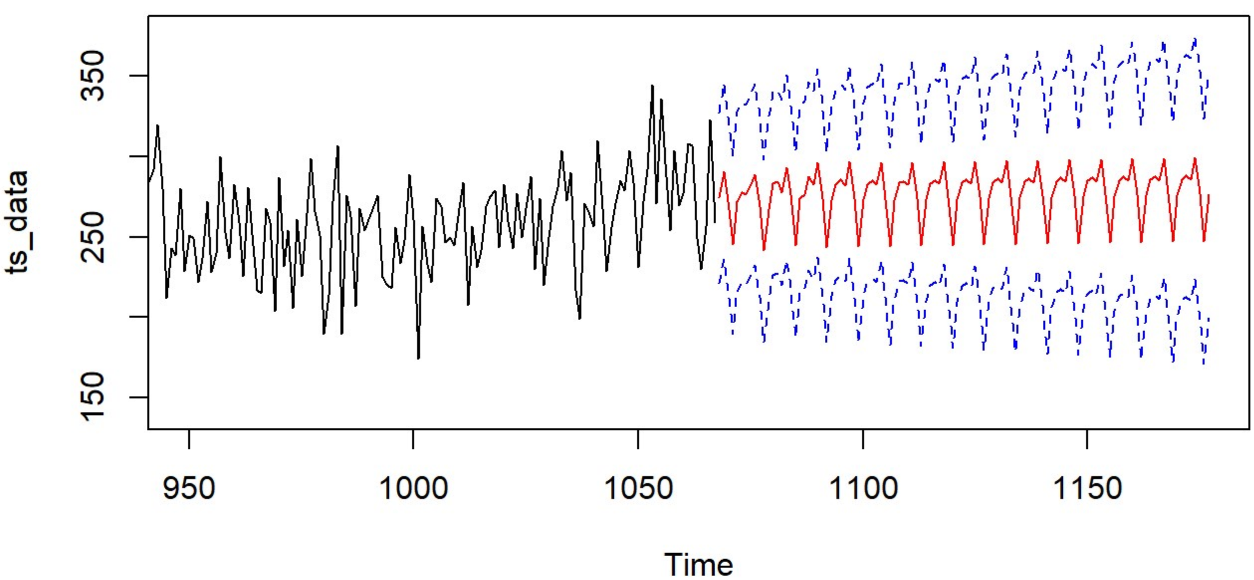
Our diagnostics for both models both pretty solid. Both models pass all the necessary tests of fit (since the p-values are all greater than 0.05), except the Shapiro-Wilk normality test. Both models fail the normality test. However, when looking at the histogram and normal Q-Q plot of the residuals, we see that they all look pretty good and normal. Even after attempting to make a model using box-cox transformed data, it still fails the residual normality test. So even though we have this test error issue, the histogram and normal Q-Q plots give us solid evidence that our model is normally distributed. All the roots for both models are within the unit circle. And lastly, the ACF/PACF plots of the residuals look great. Only 2 at max protrude out of the 95% interval boundary.

Forecasting Model A

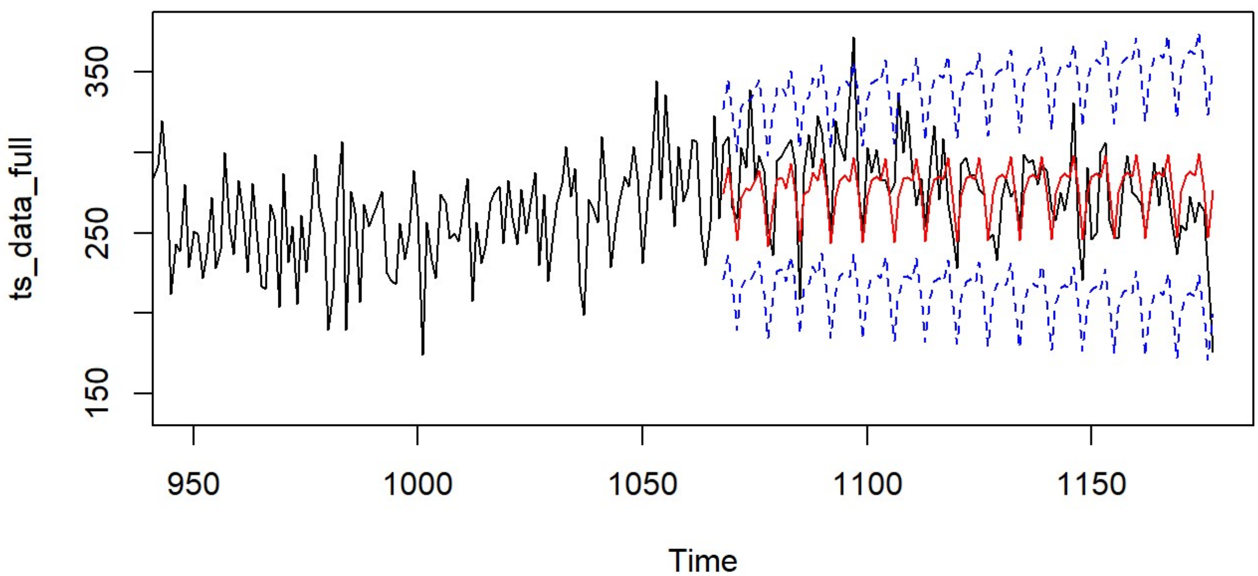
```
# Predicting future values
forecast(fit.A)
```

```
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 1068       274.4687 239.7380 309.1994 221.3526 327.5848
## 1069       291.1030 255.1523 327.0537 236.1211 346.0849
## 1070       272.1628 236.0101 308.3156 216.8720 327.4537
## 1071       245.4002 209.1233 281.6770 189.9195 300.8809
## 1072       272.3563 235.9558 308.7569 216.6865 328.0262
## 1073       277.5209 240.9971 314.0448 221.6625 333.3793
## 1074       276.4096 239.7629 313.0563 220.3633 332.4559
## 1075       282.8314 246.1315 319.5313 226.7037 338.9590
## 1076       289.0594 252.2584 325.8603 232.7772 345.3415
## 1077       269.3490 232.4351 306.2629 212.8941 325.8039
## 1078       241.6142 204.5858 278.6426 184.9841 298.2443
## 1079       267.1865 230.0439 304.3291 210.3818 323.9912
## 1080       283.2895 246.0330 320.5459 226.3107 340.2682
## 1081       284.4475 247.0776 321.8174 227.2952 341.5998
```

```
pred.A = predict(fit.A, n.ahead= 110)
U.A= pred.A$pred+ 1.96*pred.A$se
L.A= pred.A$pred-1.96*pred.A$se
ts.plot(ts_data, xlim=c(950,length(ts_data)+110))
lines(U.A, col="blue", lty=2)
lines(L.A, col="blue", lty=2)
lines(1068:1177, pred.A$pred, col="red")
```



```
# Adding original data to predictions
ts.plot(ts_data_full, xlim= c(950,length(ts_data)+110), col="black")
lines(U.A, col="blue", lty="dashed")
lines(L.A, col="blue", lty="dashed")
lines((length(ts_data)+1):(length(ts_data)+110), pred.A$pred, col="red")
lines((length(ts_data)+1):(length(ts_data)+110), pred.A$pred, col="red")
```

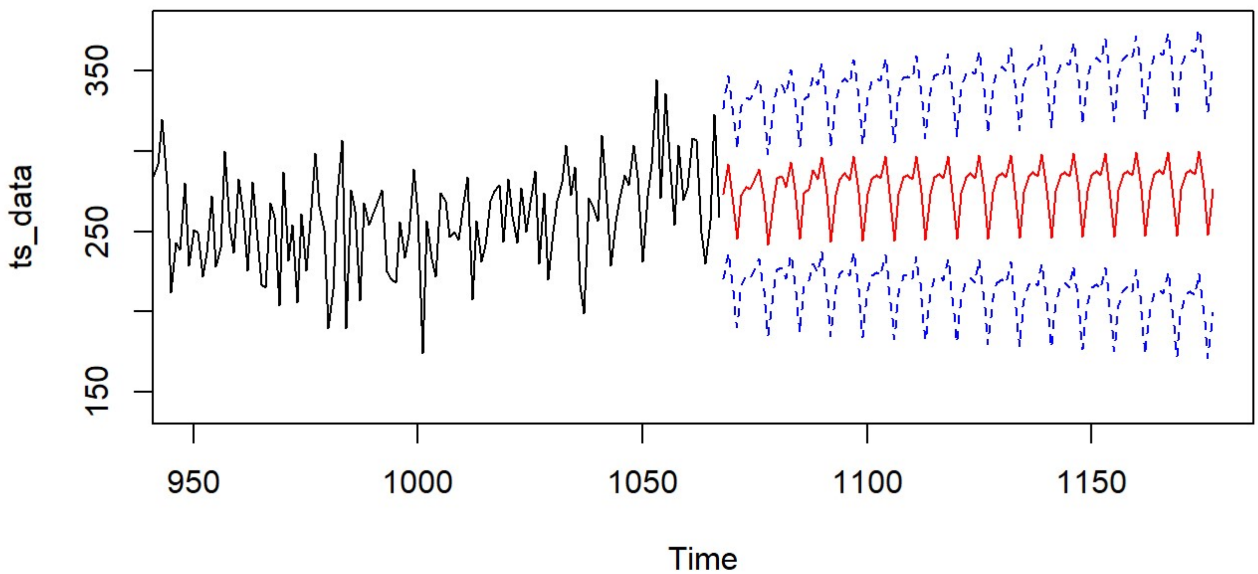


Forecasting Model B

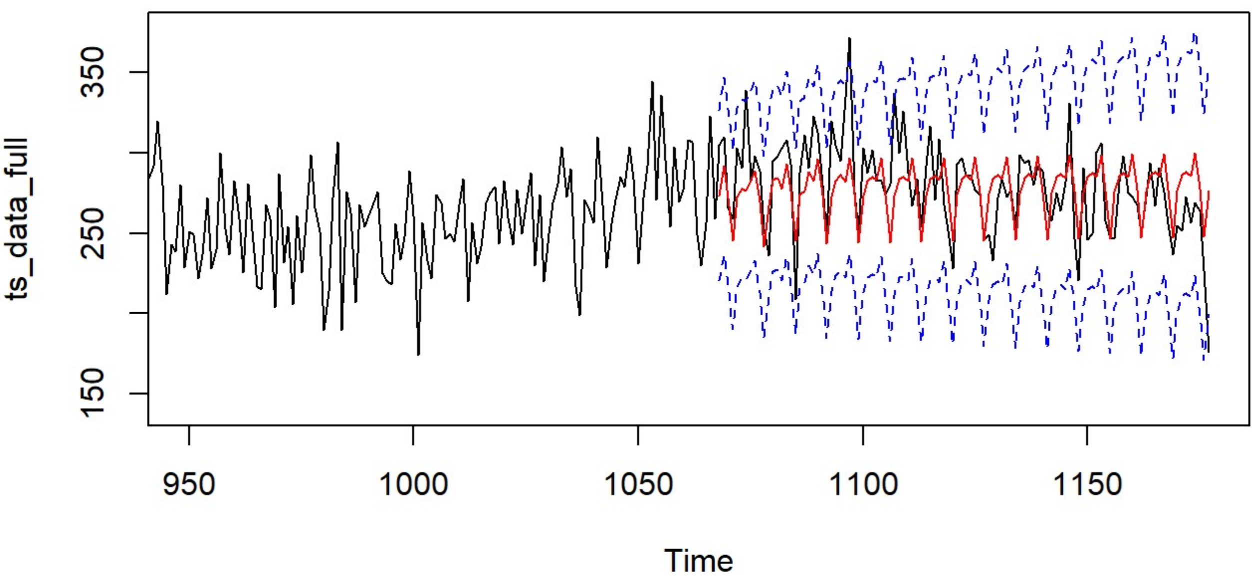
```
# Predicting future values
forecast(fit.B)
```

```
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 1068       273.7099 238.9695 308.4503 220.5791 326.8407
## 1069       292.0265 256.0847 327.9682 237.0583 346.9946
## 1070       272.3912 236.3193 308.4630 217.2240 327.5583
## 1071       245.6800 209.4785 281.8815 190.3146 301.0454
## 1072       272.5438 236.2131 308.8745 216.9808 328.1068
## 1073       277.6559 241.1965 314.1153 221.8960 333.4157
## 1074       276.5554 239.9678 313.1431 220.5994 332.5114
## 1075       283.0766 246.4369 319.7163 227.0410 339.1122
## 1076       289.1334 252.3894 325.8774 232.9382 345.3286
## 1077       269.5850 232.7216 306.4483 213.2073 325.9626
## 1078       241.8360 204.8537 278.8182 185.2765 298.3955
## 1079       267.3441 230.2434 304.4449 210.6034 324.0849
## 1080       283.4828 246.2638 320.7017 226.5613 340.4042
## 1081       284.7771 247.4404 322.1139 227.6756 341.8787
```

```
pred.B = predict(fit.B, n.ahead= 110)
U.B= pred.B$pred+ 1.96*pred.B$se
L.B= pred.B$pred-1.96*pred.B$se
ts.plot(ts_data, xlim=c(950,length(ts_data)+110))
lines(U.B, col="blue", lty=2)
lines(L.B, col="blue", lty=2)
lines(1068:1177, pred.B$pred, col="red")
```




```
# Adding original data to predictions
ts.plot(ts_data_full, xlim= c(950,length(ts_data)+110), col="black")
lines(U.B, col="blue", lty="dashed")
lines(L.B, col="blue", lty="dashed")
lines((length(ts_data)+1):(length(ts_data)+110), pred.B$pred, col="red")
lines((length(ts_data)+1):(length(ts_data)+110), pred.B$pred, col="red")
```



After examining the forecast predictions for both models, we can see that they are extremely similar. They both show somewhat accurate predictions and do a good job articulating the weekly seasonality. As it attempts to predict more, it very slightly begins to over/under predict possibly due to the changing yearly seasonality (Not uncommon to see in models such as these). Because it is very difficult to notice differences, we will choose Model B being better based on its lower AICc value.

3. Conclusion

After analyzing the time series and acf plots of the crime data, it is apparent that there is a pattern in the frequency of crimes. It seems that there is a yearly pattern as well as a weekly pattern. Friday appears to have he highest number of crimes per week, while Sunday the least. From the data, it is not quite clear why most crimes occur on Friday. Maybe we can attribute it to Friday night being the most popular night of the week to leave the house; And Sunday the least likely night of the week to leave the house. However, there are many parameters that can attribute to this, and we need more outside data to identify the reason.

Due to the strong seasonality of the data, it was necessary to use a SARIMA model in order to forecast future data. From understanding the weekly seasonal pattern, I was able to make a couple SARIMA models that accurately forecast the daily frequency of crimes. Although the residuals of both models appeared normal in the diagnostic plots, it is important to note that they did not pass the Shapiro-Wilk normality test.

4. References

Dataset https://www.kaggle.com/AnalyzeBoston/crimes-in-boston (https://www.kaggle.com/AnalyzeBoston/crimes-in-boston)

Skewness https://www.r-bloggers.com/measures-of-skewness-and-kurtosis/ (https://www.r-bloggers.com/measures-of-skewness-and-kurtosis/)

Decompose http://r-statistics.co/Time-Series-Analysis-With-R.html (http://r-statistics.co/Time-Series-Analysis-

With-R.html)

Adf Stationary Testing https://nwfsc-timeseries.github.io/atsa-labs/sec-boxjenkins-aug-dickey-fuller.html (https://nwfsc-timeseries.github.io/atsa-labs/sec-boxjenkins-aug-dickey-fuller.html)

Forecast https://robjhyndman.com/hyndsight/forecast7-ggplot2/ (https://robjhyndman.com/hyndsight/forecast7-ggplot2/)

5. Appendix

```
# import dataset
crime_df = read.csv('crime.csv')
attach(crime_df)

# sorting data by date of crime
crime_df = crime_df  %>% separate(OCCURRED_ON_DATE, c("Date", "Time"), sep = " ") %>%
mutate(Date = ymd(Date))
crime_df$Date = as.Date(crime_df$Date)
crime_df = crime_df[order(crime_df$Date),]

ts_data_full = ts(table(crime_df$Date))

# split up data into train/test split
ts_data = ts(ts_data_full[c(1:1067)])
ts_data.test = ts(ts_data_full[c(1068:1077)])

# histogram of our crime data
hist(ts_data, 50, col='light blue', main='Crimes Count Distribution')
paste(c('Skewness is', skewness(ts_data)), collapse = ' ')

# Ploting timeseries plot
ts.plot(ts_data, main = "Crimes by Time")
abline(h=mean(ts_data), col='red')


# Transformation of data but not needed/performed

# bcTransform<-boxcox(ts_data ~ as.numeric(1:length(ts_data)))
# bcTransform$x[which(bcTransform$y== max(bcTransform$y))]
# lambda = bcTransform$x[which(bcTransform$y== max(bcTransform$y))]
# lambda = bcTransform$x[which(bcTransform$y== max(bcTransform$y))]
# ts.bc = (1/lambda)*(ts_data^lambda-1)
# hist(ts.bc, 50, col='light blue', main='Crimes Count Distribution')
# shapiro.test(ts.bc)

# ACF and PACF
acf(table(crime_df$Date), 40, main='Autocorrelation Lag=40')
pacf(table(crime_df$Date), 40, main='Partial Autocorrelation Lag=40')


# Decomposing to examine seasonality/trend
y = ts(as.ts(ts_data), frequency=7)
plot(decompose(y))

# Differencing the model by 7 to remove seasonality/trend
ts_7 = diff(ts_data, lag=7)

plot.ts(ts_7, main="Crimes by Time Differenced at lag 7")
abline(h=mean(ts_7), col='red')
```

```
adf.test(ts_7)

acf(ts_7, 50, main='Autocorrelation of ts_7')
pacf(ts_7,50, main='Partial Autocorrelation ts_7')

# Model A
fit.A = arima(ts_data, order=c(0,1,3), seasonal = list(order = c(4,1,1), period = 7),
method="CSS")
fit.A
AICc(arima(ts_data, order=c(0,1,3), seasonal = list(order = c(4,1,1), period = 7),  me
thod="ML"))

# Model B
fit.B = arima(ts_data, order=c(0,1,2), seasonal = list(order = c(4,1,1), period = 7),
method="CSS")
fit.B
AICc(arima(ts_data, order=c(0,1,2), seasonal = list(order = c(4,1,1), period = 7), met
hod="ML"))

# Checking Invertibility of model A
autoplot(fit.A)

# residual hist
res = residuals(fit.A)
hist(res,density=20,breaks=20, col="blue", xlab="", prob=TRUE)
m = mean(res)
std = sqrt(var(res))
curve( dnorm(x,m,std), add=TRUE )
plot.ts(res)
fitt = lm(res ~ as.numeric(1:length(res))); abline(fitt, col="red")
abline(h=mean(res), col="blue")

# Normal Q-Q to check for normality of res
qqnorm(res,main= "Normal Q-Q Plot for Model A")
qqline(res,col="blue")

acf(res, lag.max=40)
pacf(res, lag.max=40)

# Tests to check fit
df = 3
shapiro.test(res)
Box.test(res, lag = 7, type = c("Box-Pierce"), fitdf= df)
Box.test(res, lag = 7, type = c("Ljung-Box"), fitdf= df)
Box.test(res^2, lag = 7, type = c("Ljung-Box"), fitdf= 0)

#Checking Invertibility of model B
autoplot(fit.B)

# residual hist
res = residuals(fit.B)
```

```r
hist(res,density=20,breaks=20, col="blue", xlab="", prob=TRUE)
m = mean(res)
std = sqrt(var(res))
curve( dnorm(x,m,std), add=TRUE )
plot.ts(res)
fitt = lm(res ~ as.numeric(1:length(res))); abline(fitt, col="red")
abline(h=mean(res), col="blue")

# Normal Q-Q to check for normality of res
qqnorm(res,main= "Normal Q-Q Plot for Model B")
qqline(res,col="blue")

acf(res, lag.max=40)
pacf(res, lag.max=40)

# Tests to check fit
df = 2
shapiro.test(res)
Box.test(res, lag = 7, type = c("Box-Pierce"), fitdf= df)
Box.test(res, lag = 7, type = c("Ljung-Box"), fitdf= df)
Box.test(res^2, lag = 7, type = c("Ljung-Box"), fitdf= 0)

# Predicting future values
forecast(fit.A)
pred.A = predict(fit.A, n.ahead= 110)
U.A= pred.A$pred+ 1.96*pred.A$se
L.A= pred.A$pred-1.96*pred.A$se
ts.plot(ts_data, xlim=c(950,length(ts_data)+110))
lines(U.A, col="blue", lty=2)
lines(L.A, col="blue", lty=2)
lines(1068:1177, pred.A$pred, col="red")

# Adding original data to predictions
ts.plot(ts_data_full, xlim= c(950,length(ts_data)+110), col="black")
lines(U.A, col="blue", lty="dashed")
lines(L.A, col="blue", lty="dashed")
lines((length(ts_data)+1):(length(ts_data)+110), pred.A$pred, col="red")
lines((length(ts_data)+1):(length(ts_data)+110), pred.A$pred, col="red")

# Predicting future values
forecast(fit.B)
pred.B = predict(fit.B, n.ahead= 110)
U.B= pred.B$pred+ 1.96*pred.B$se
L.B= pred.B$pred-1.96*pred.B$se
ts.plot(ts_data, xlim=c(950,length(ts_data)+110))
lines(U.B, col="blue", lty=2)
lines(L.B, col="blue", lty=2)
lines(1068:1177, pred.B$pred, col="red")

# Adding original data to predictions
ts.plot(ts_data_full, xlim= c(950,length(ts_data)+110), col="black")
```

```
lines(U.B, col="blue", lty="dashed")
lines(L.B, col="blue", lty="dashed")
lines((length(ts_data)+1):(length(ts_data)+110), pred.B$pred, col="red")
lines((length(ts_data)+1):(length(ts_data)+110), pred.B$pred, col="red")
```