PSTAT 174 Final Project

Ron Vechter

6/5/2020

Abstract

Predicting Armed Robberies in Boston

The question I addressed in this project is one of predicting armed robberies in Boston in order to warn it's citizens and law enforcement and consequently make them safer. By analyzing a dataset of monthly boston armed robberies from January 1966 to October 1975, we can infer patterns for for the purpose of forecasting, and create a model to try to predict when robberies are likely to happen.

Some of the techniques I used in included trunkating the data set, one part to be used for model training, and the other for model validation to ensure an accurate model. I used a Box-cox transformation in order to stabilize variance, then differenced at lag 12 to remove seasonality and at lag 1 to remove trend. After ensuring the data was stationary through a series of tests, I analyzed the ACF and PACF of our stationary and invertible transformed data, and tested candidate models to find the one with the lowest AICc, and checked for insignificant coefficients. I discovered that the best model to try was SARIMA(0,1,1)x(2,1,1)$_{12}$ with SAR1 fixed at 0, since the other model I discovered to not be invertible, and tested the residuals for normality. Finally, I forecasted the model on the transformed data, and then on the original data to compare it to the true values.

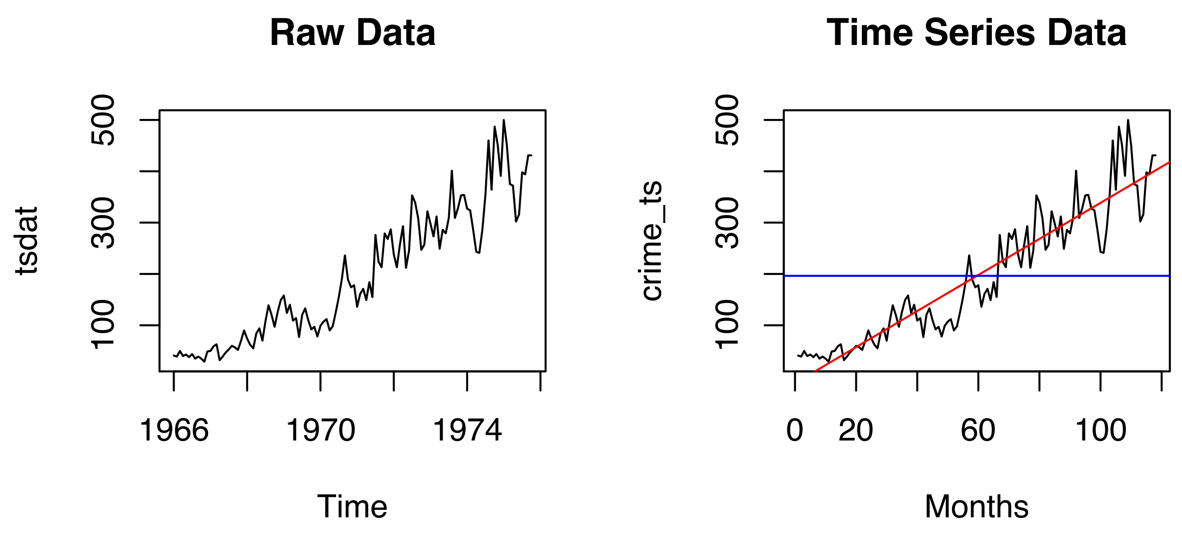
Conclusively, my model could definitely be improved, as the predicted values are still slightly higher than the true values, but finding the model with the lowest AICc proved to be very difficult for me, so perhaps finding a lower AICc model could improve my forecast, but it is satisfactory for me.

Monthly Boston armed robberies: Jan. 1966 - Oct. 1975

Introduction:

The problem I am looking to answer is: Can we predict when armed robberies in Boston are likely to occur and how we can prepare citizens to be on high alert. The dataset is of interest because Boston is known as a great hub of crime and if we can decipher what factors lead to seasonality of armed robberies, or why they are increasing as time goes on, we can try to stop armed robberies way before they happen. However, we must be aware that the dataset spans from January 1966 to October 1975, so it cannot be easily applied to today, as there are lots of unexpected changes and outside factors that would likely make our model not so accurate that far into the future. So lets get into the data:
- Monthly totals in armed robberies in Boston; $\{U_t$; t=1,2,...,118$\}$
- Data Reference: tsdl package

```
library(tsdl)
library(MASS)
BosCrime <- subset(tsdl, 12, "Crime")[[2]]
tsdat <- ts(BosCrime, start = c(1966,1), end = c(1975,10), frequency = 12)
op <- par(mfrow=c(1,2))
plot.ts(tsdat, main = "Raw Data")
crime_ts <- ts(BosCrime)
plot.ts(crime_ts, xlab = "Months", main = "Time Series Data")
fit <- lm(crime_ts ~ as.numeric(1:length(crime_ts)))
abline(fit, col = "Red")
abline(h=mean(crime_ts), col="blue")
```

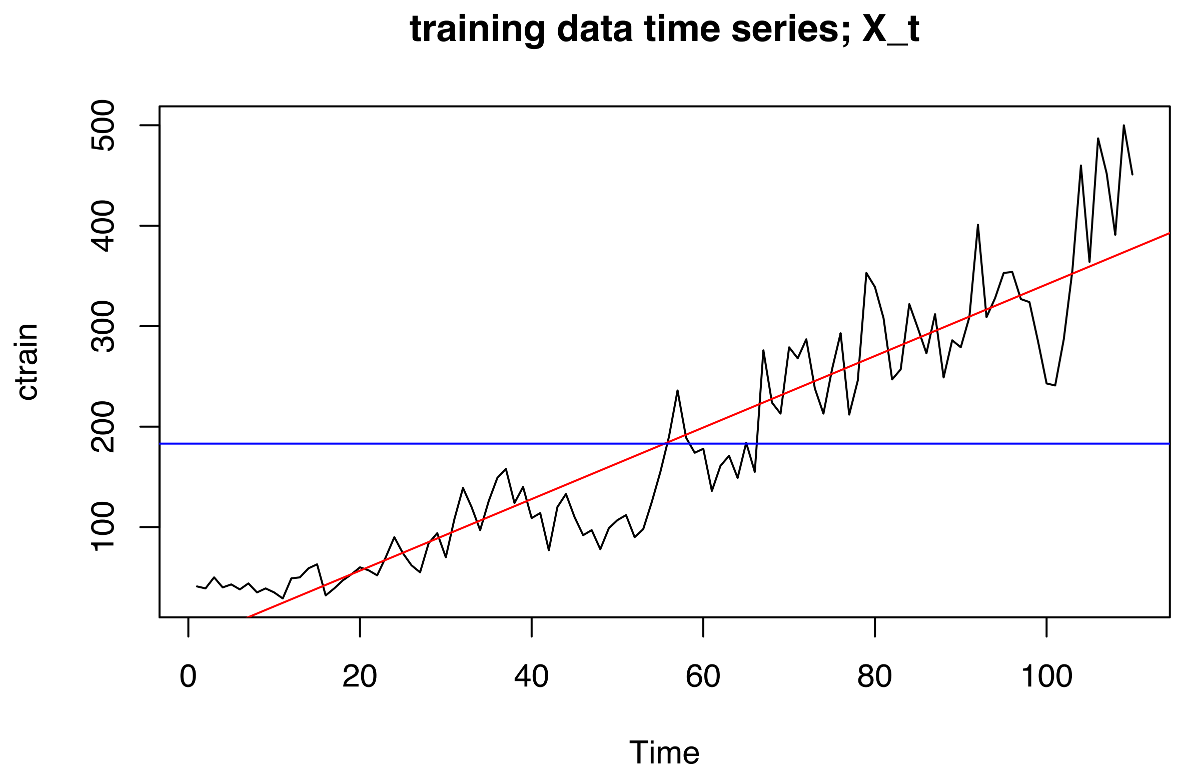


```
par(op)
```

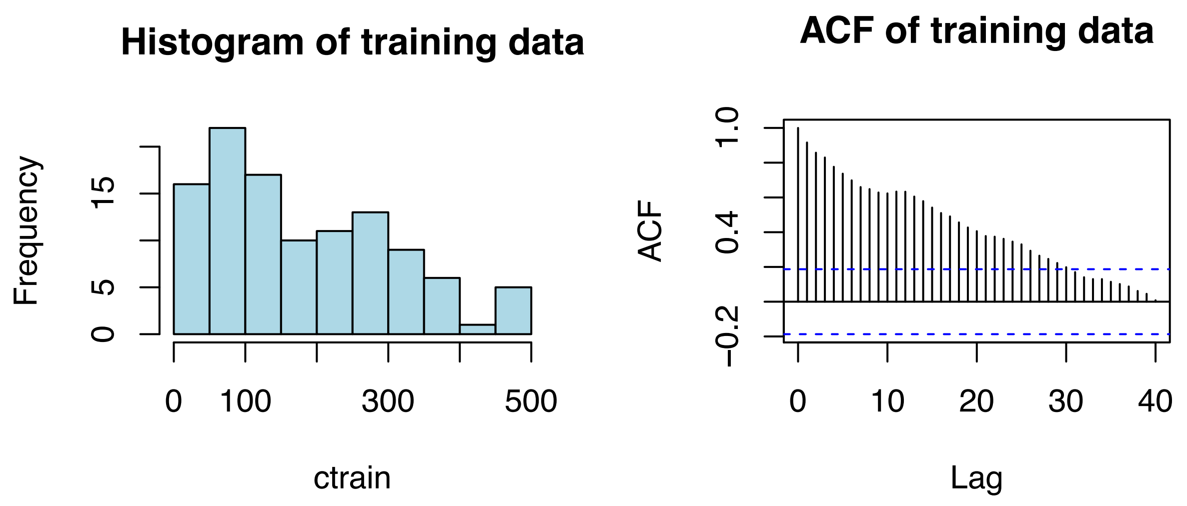
Partition dataset to two parts for model training and model validation

We partition the dataset into two sets, one called ctrain, which will be our training set to find the right model and will contain the first 110 observations (months). The other will be called ctest, which we will contain the last 8 observations (months), used to test our model and compare it to the true values in ctest.

```r
ctrain <- BosCrime[c(1:110)]
ctest <- BosCrime[c(111:118)]
plot.ts(ctrain, main = "training data time series; X_t")
fit <- lm(ctrain ~ as.numeric(1:length(ctrain)))
abline(fit, col="red")
abline(h=mean(ctrain), col="blue")
```



```
op <- par(mfrow=c(1,2))
hist(ctrain, col = "light blue", main = "Histogram of training data")
acf(ctrain, lag.max = 40, main="ACF of training data")
```



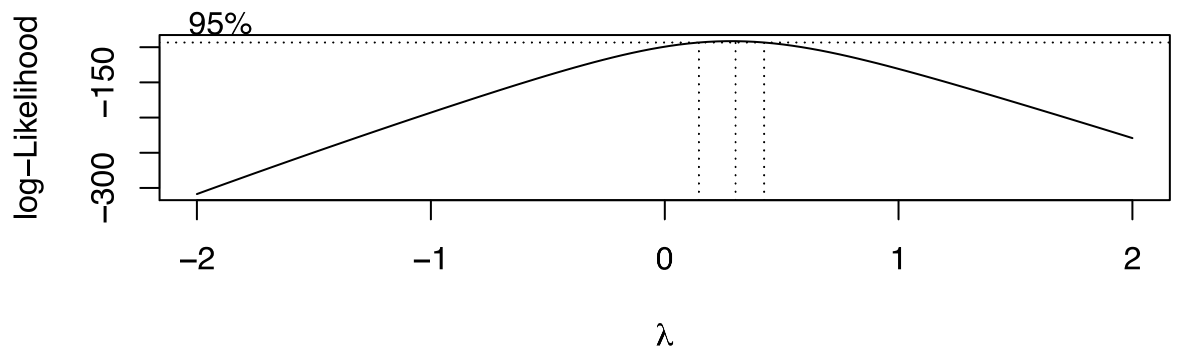
```
par(op)
```

Immediate observations of the data:

Highly non-stationary:
- obvious linear trend, possible seasonality, non-constant variance and mean.
- the histogram is badly skewed, and ACFs remain large at large lags and decay slowly, may be periodic.

We need to perform a transformation to stabilize variance, and difference to remove trend and seasonality.

```
# Start with Box-cox transformation:
bcTransform <- boxcox(ctrain ~ as.numeric(1:length(ctrain)), plotit = T)
```

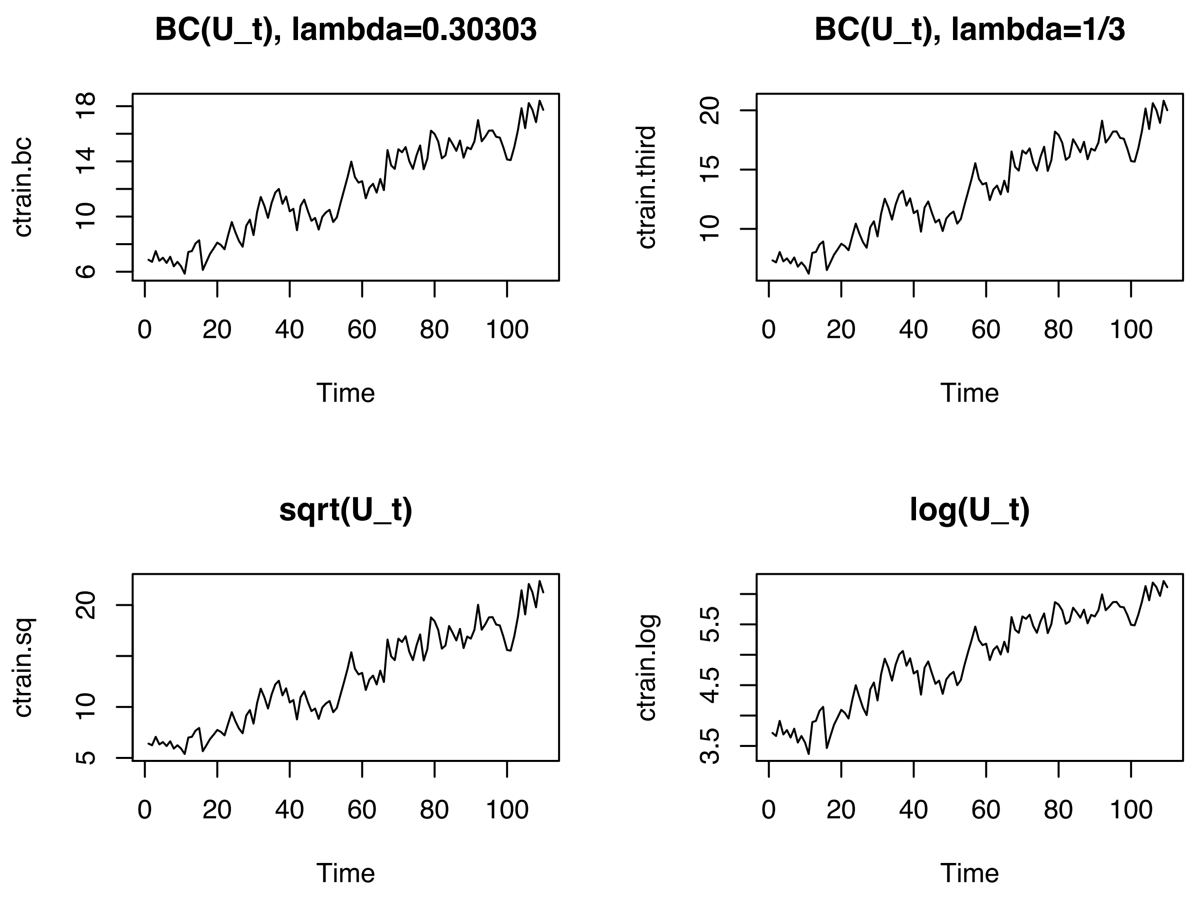


```
lambda = bcTransform$x[which(bcTransform$y == max(bcTransform$y))]
lambda
```

```
## [1] 0.3030303
```

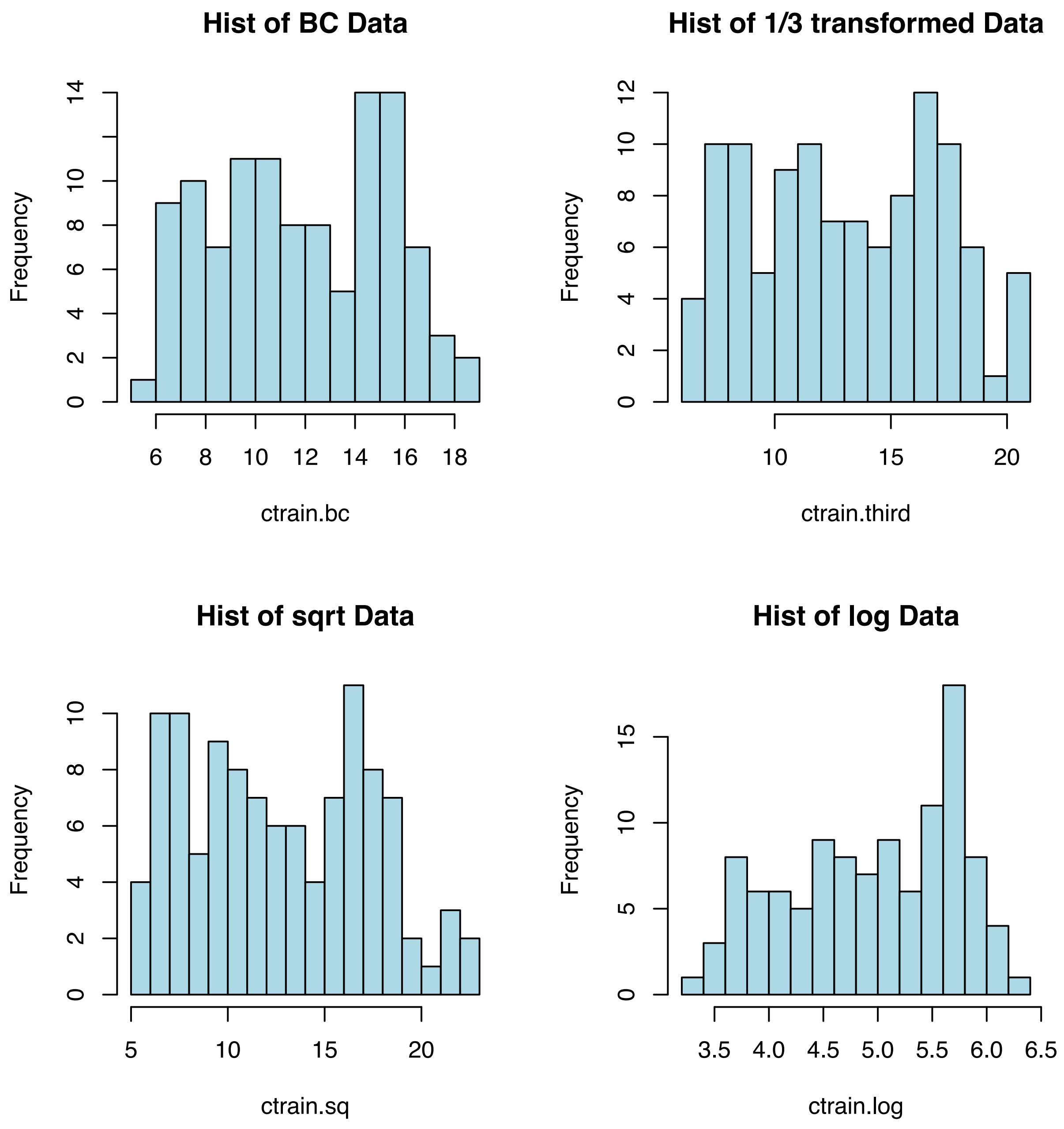
We notice that the lambda value given from Box-Cox transformation is very close to 1/3, so we will try that transformation as well, along with square root and log transformations:

```
l<- 1/3
ctrain.bc <- (1/lambda)*(ctrain^lambda-1)
ctrain.third <- (1/l)*(ctrain^l-1)
ctrain.sq <- sqrt(ctrain)
ctrain.log <- log(ctrain)
op <- par(mfrow=c(2,2))
plot.ts(ctrain.bc, main = "BC(U_t), lambda=0.30303")
plot.ts(ctrain.third, main = "BC(U_t), lambda=1/3")
plot.ts(ctrain.sq, main = "sqrt(U_t)")
plot.ts(ctrain.log, main = "log(U_t)")
```



```
par(op)
```

```r
op <- par(mfrow=c(2,2))
hist(ctrain.bc, breaks = 15, col = "light blue", main = "Hist of BC Data")
hist(ctrain.third, breaks = 15, col = "light blue", main = "Hist of 1/3 transformed Data")
hist(ctrain.sq, breaks = 15, col = "light blue", main = "Hist of sqrt Data")
hist(ctrain.log, breaks = 15, col = "light blue", main = "Hist of log Data")
```

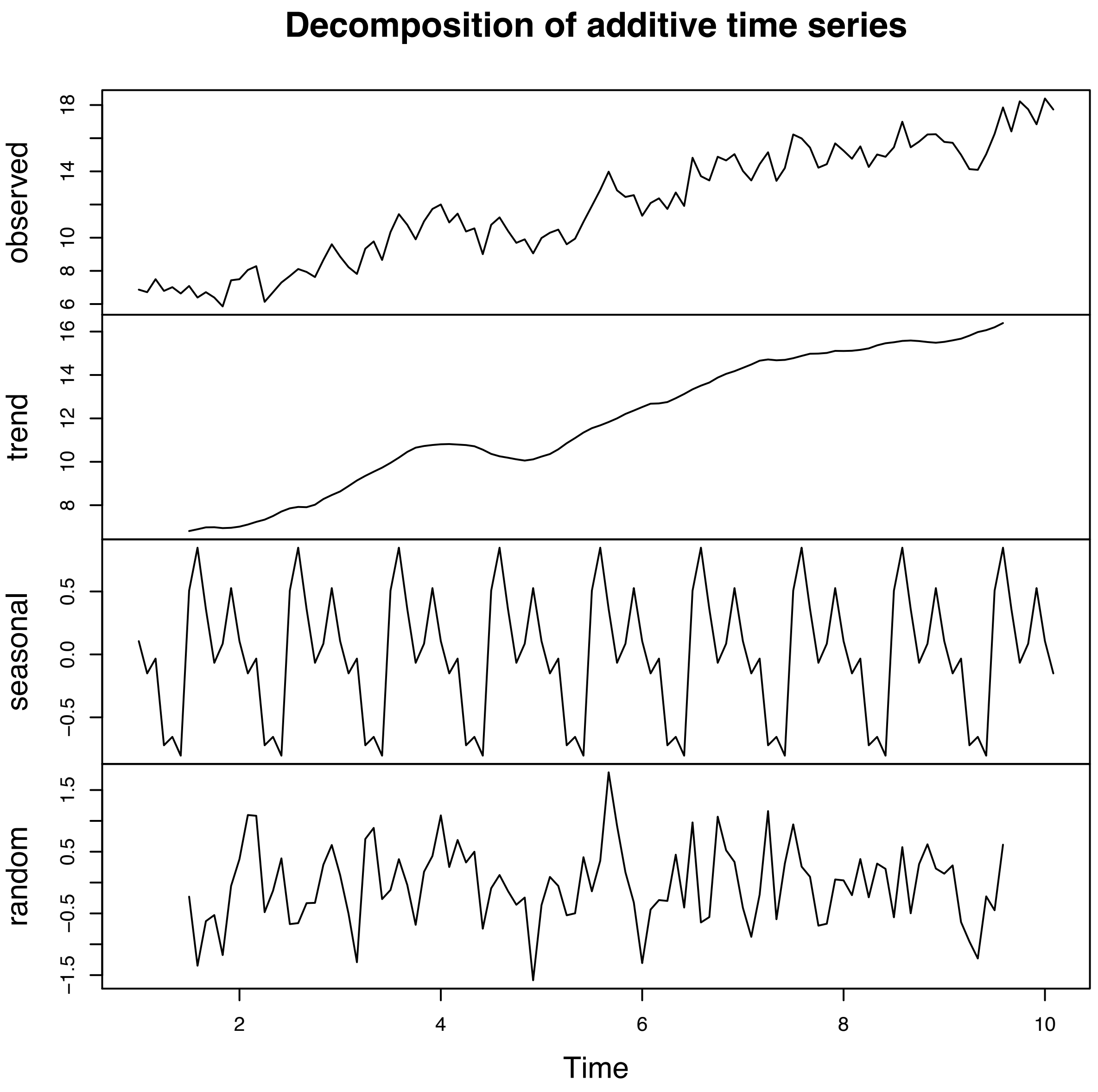


```r
par(op)
```

Our Box-Cox transformation gives λ=0.3030303, and we see that our time series has a much more stable variance after transformation.

To produce a decomposition of BC(U_t)

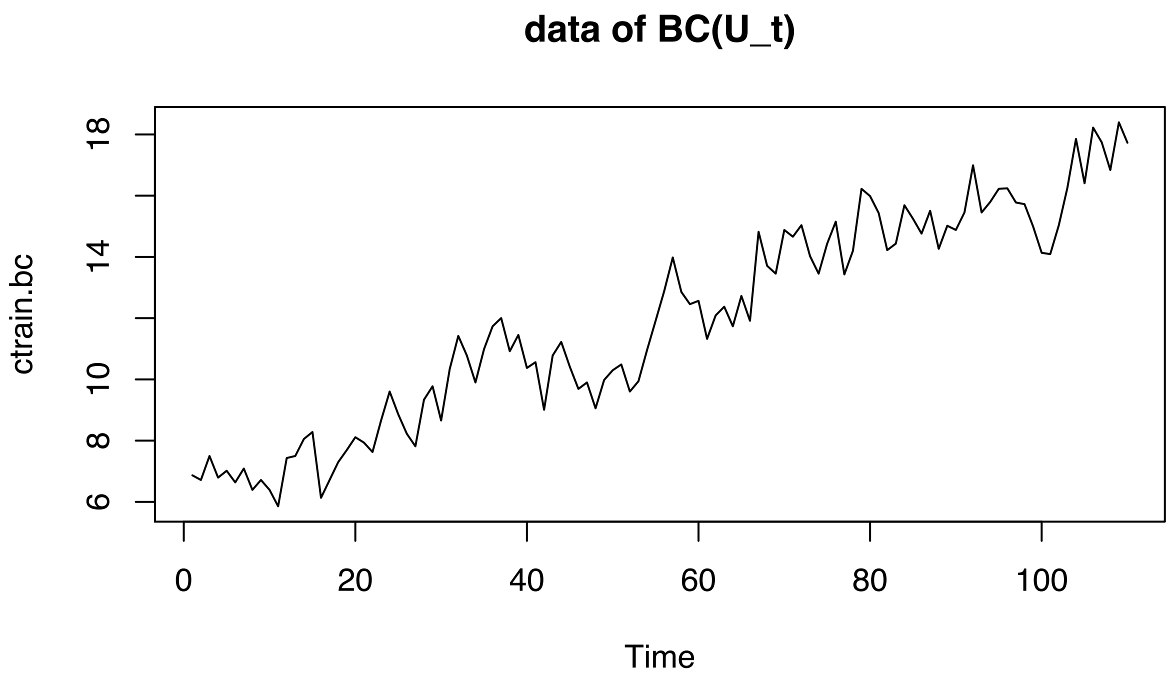
```
library(ggplot2)
library(ggfortify)
y <- ts(as.ts(ctrain.bc), frequency = 12)
decomp <- decompose(y)
plot(decomp)
```



Decomposition of BC(U_t) shows seasonality and almost linear trend, so we are ready to difference:

Differencing BC(U_t)

```r
plot.ts(ctrain.bc, main="data of BC(U_t)")
```



```r
var(ctrain.bc)
```

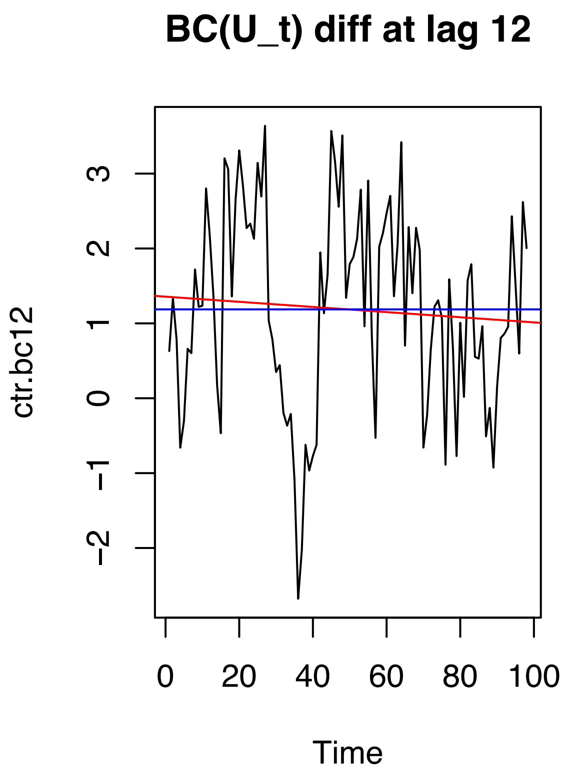
```
## [1] 11.67565
```

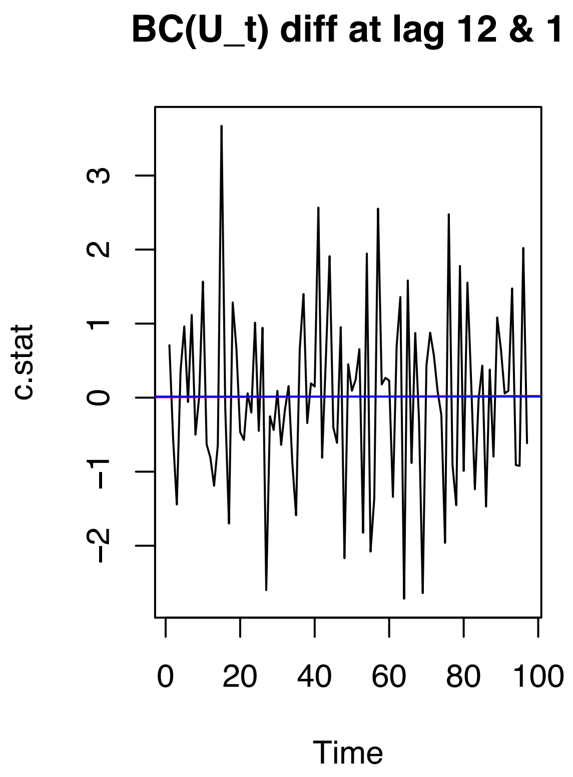
```r
# at lag 12
ctr.bc12 <- diff(ctrain.bc, lag = 12)
var(ctr.bc12) # lower variance !
```

```
## [1] 1.804579
```

```r
op <- par(mfrow=c(1,2))
plot.ts(ctr.bc12, main = "BC(U_t) diff at lag 12") # seasonality no longer apparent
fit <- lm(ctr.bc12 ~ as.numeric(1:length(ctr.bc12)))
abline(fit, col="red")
abline(h=mean(ctr.bc12), col="blue")

# at lags 12 and 1
c.stat <- diff(ctr.bc12, lag = 1)
plot.ts(c.stat, main = "BC(U_t) diff at lag 12 & 1")
fit <- lm(c.stat ~ as.numeric(1:length(c.stat)))
abline(fit, col="red")
abline(h=mean(c.stat), col="blue") # trend is eliminated
```





```
par(op)
var(c.stat) # even lower variance !
```

```
## [1] 1.472102
```

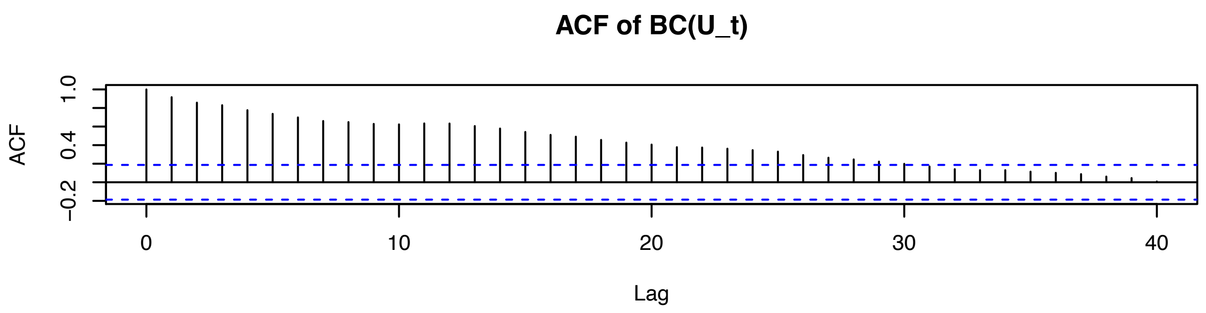
```
mean(c.stat) # mean close to 0 !
```

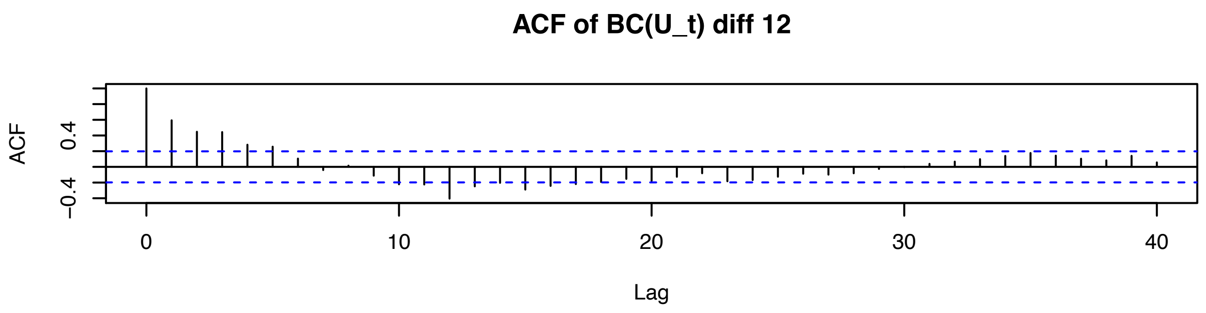
```
## [1] 0.01417604
```

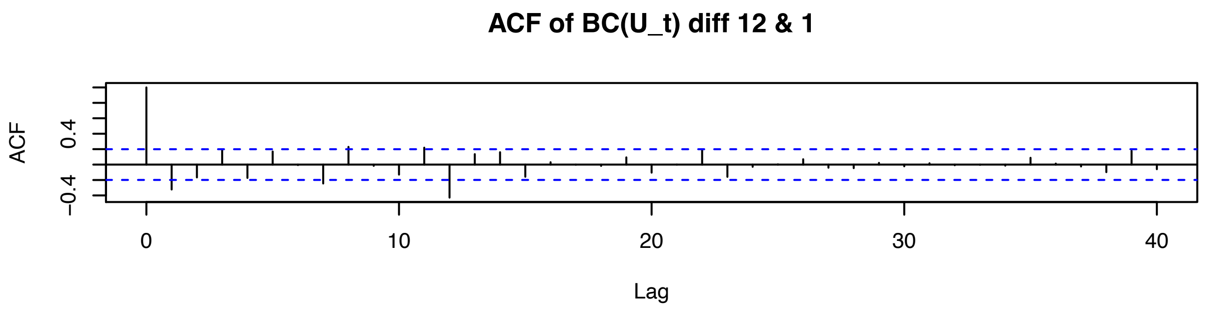
We first difference at lag 12 in attempt to remove seasonality. We see that our variance decreased, which is good, and we no longer seem to have signs of seasonality, but we still have some trend, so we need to difference at lag 1. After differencing at lag 1, we notice the variance decreased again, which is good, and we seem to no longer have any trend. The data looks stationary, but we need to check ACFs to make sure.

ACFs of BC(U_t) and it's differences

```r
op <- par(mfrow=c(3,1))
acf(ctrain, lag.max = 40, main="ACF of BC(U_t)")
acf(ctr.bc12, lag.max = 40, main = "ACF of BC(U_t) diff 12")
acf(c.stat, lag.max = 40, main = "ACF of BC(U_t) diff 12 & 1")
```





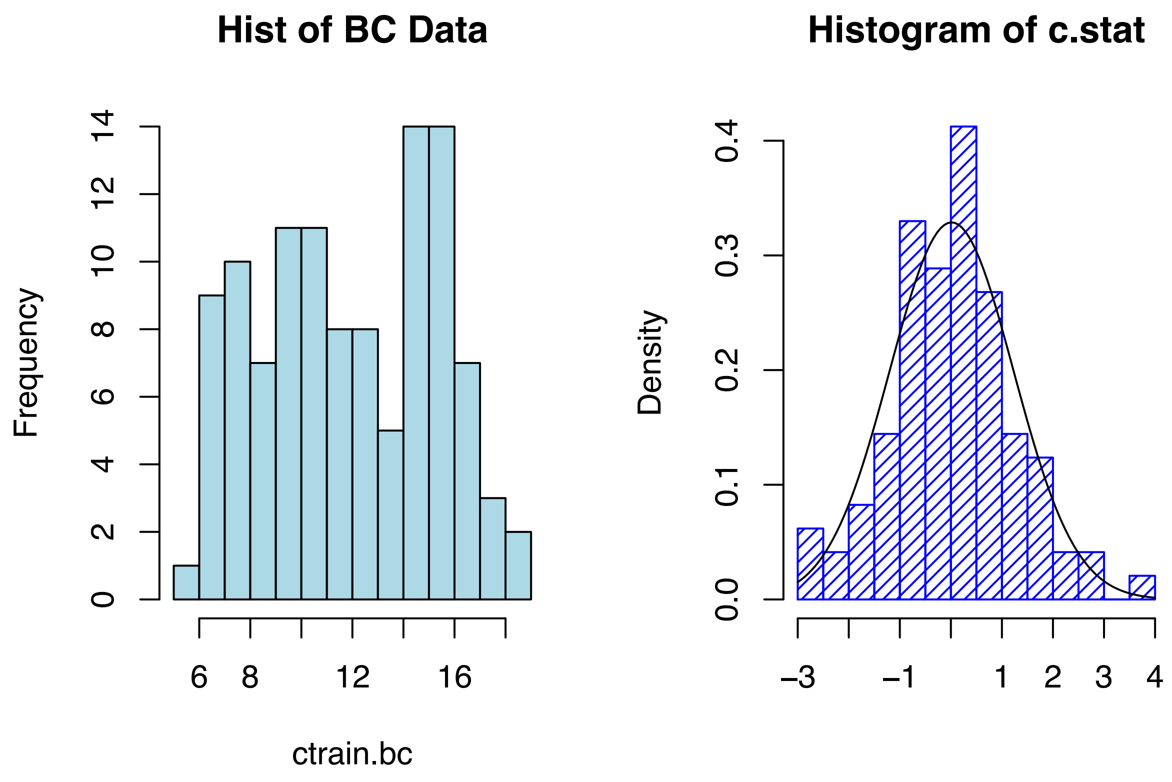


```r
par(op)
```

- For the ACF of BC(U_t), we notice that ACF decays slowly and shows signs of seasonality, so it is clearly not stationary.
- For the ACF of BC(U_t) differenced at lag 12, we no longer see a seasonality component, but the ACF still decays slowly, indicationg non-stationarity.
- For the ACF of BC(U_t) differenced at lags 12 and 1, the ACF decays corresponds to that of a stationary process.
  So we conclude that we will work with $\nabla_1\nabla_{12}\text{BC}(U_t)$, where U_t is the first 110 observations from the original data.

Histogram of transformed and differenced data with normal curve

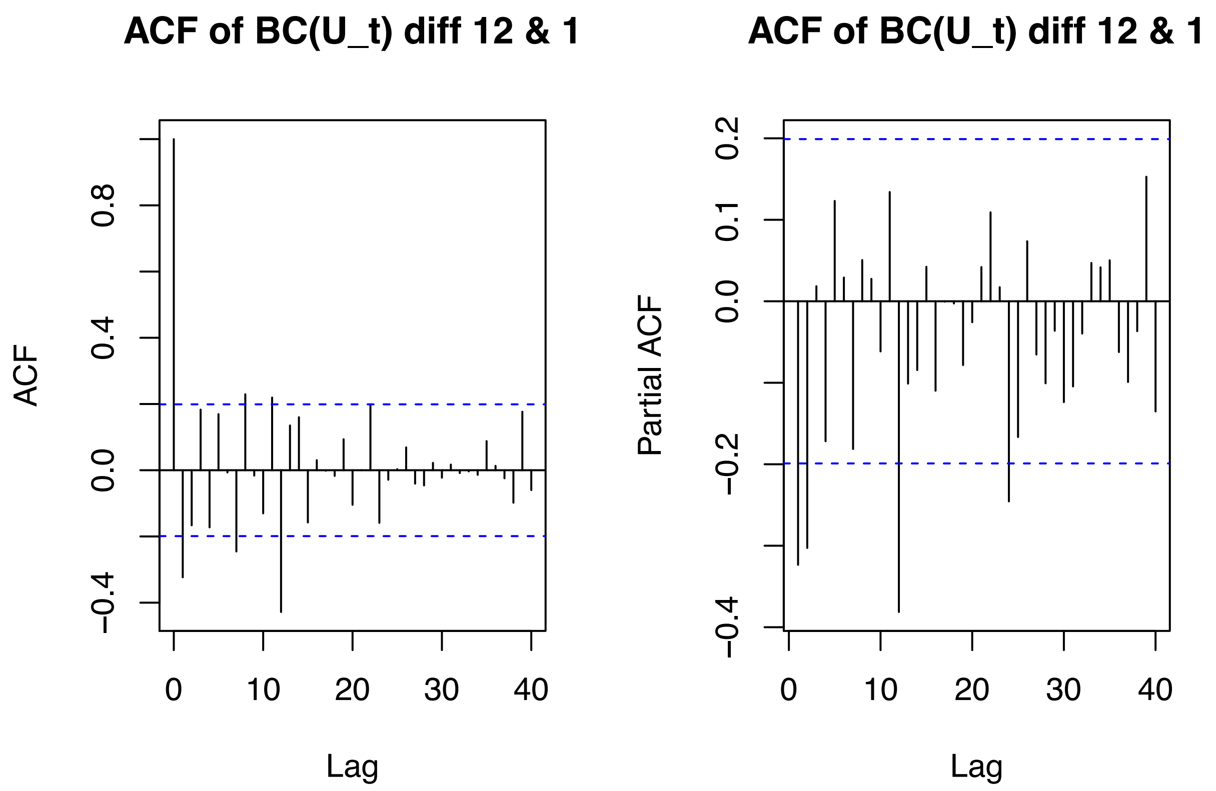
```
op <- par(mfrow=c(1,2))
hist(ctrain.bc, breaks = 15, col = "light blue", main = "Hist of BC Data")
hist(c.stat, density=20,breaks=20, col="blue", xlab="", prob=TRUE)
m <- mean(c.stat)
std <- sqrt(var(c.stat))
curve(dnorm(x,m,std), add = TRUE)
```



```
par(op)
```

ACF and PACF of c.stat

```r
op <- par(mfrow=c(1,2))
acf(c.stat, lag.max = 40, main = "ACF of BC(U_t) diff 12 & 1")
pacf(c.stat, lag.max = 40, main = "ACF of BC(U_t) diff 12 & 1")
```



```r
par(op)
```

We see that for the ACF we are outside the confidence interval at lags 1 and 12, maybe also at lags 7,8,11. From this we can most likely say Q = 0 or 1, q = 0 or 1 or 2(unlikely). For the PACF, we seem to be outside the confidence interval at lags 1,2,12,24. From this we can infer that P = 1 or 2, p = 1 or 2(unlikely). Since we differenced at lags 12 and 1, we know we will have a SARIMA model with S=12, D=1, d=1. Some candidate models to try:

- S=12, D=1, d=1, q=0 or 1 or 2(unlikely), Q=0 or 1, p=1 or 2, P=1 or 2.

Testing Candidate models

We test pure SMA models first, meaning that P=p=0. We already know that S=12, d=D=1, so let's try to find our lowest AICc for q=0 or 1 or 2(unlikely), Q=0 or 1. After testing all these, our lowest two AICc models are below:

```
library(qpcR)
# lowest AICc:
arima(ctrain.bc, order = c(0,1,1), seasonal = list(order=c(0,1,1), period=12),
      method = "ML")
```

```
##
## Call:
## arima(x = ctrain.bc, order = c(0, 1, 1), seasonal = list(order = c(0, 1, 1),
##     period = 12), method = "ML")
##
## Coefficients:
##           ma1     sma1
##       -0.4707  -0.9999
## s.e.   0.0990   0.5524
##
## sigma^2 estimated as 0.618:  log likelihood = -127.65,  aic = 261.3
```

```
AICc(arima(ctrain.bc, order = c(0,1,1), seasonal = list(order=c(0,1,1), period=12),
           method = "ML"))
```

```
## [1] 261.4079
```

```
# 2nd lowest AICc
arima(ctrain.bc, order = c(0,1,2), seasonal = list(order=c(0,1,1), period=12),
      method = "ML")
```

```
##
## Call:
## arima(x = ctrain.bc, order = c(0, 1, 2), seasonal = list(order = c(0, 1, 1),
##     period = 12), method = "ML")
##
## Coefficients:
##           ma1      ma2     sma1
##       -0.4199  -0.1215  -1.0000
## s.e.   0.1060   0.1268   0.8972
##
## sigma^2 estimated as 0.6116:  log likelihood = -127.17,  aic = 262.35
```

```
AICc(arima(ctrain.bc, order = c(0,1,2), seasonal = list(order=c(0,1,1), period=12),
           method = "ML"))
```

```
## [1] 262.5748
```

We see that for the second model, 0 is in the confidence interval for ma2, but when we reduce it, we get a model identical to our first model, so we will keep it as our model with the second lowest AICc. Next, we try to introduce an AR component to our models with the predictions for P and p above, and we see that as we predicted, a new second lowest AICc model is p=0,P=2, where SAR1 is deemed insignificant and is therefore fixed at 0:

```
# new second lowest AICc:
AICc(arima(ctrain.bc, order = c(0,1,1), seasonal = list(order=c(2,1,1), period=12),
           fixed = c(NA,0,NA,NA)))
```

```
## [1] 262.1298
```

```
arima(ctrain.bc, order = c(0,1,1), seasonal = list(order=c(2,1,1), period=12),
      fixed = c(NA,0,NA,NA))
```

```
##
## Call:
## arima(x = ctrain.bc, order = c(0, 1, 1), seasonal = list(order = c(2, 1, 1),
##     period = 12), fixed = c(NA, 0, NA, NA))
##
## Coefficients:
##           ma1  sar1     sar2     sma1
##       -0.4683     0  -0.1513  -0.8956
## s.e.   0.0986     0   0.1184   0.2774
##
## sigma^2 estimated as 0.6449:  log likelihood = -126.87,  aic = 261.75
```

So we have 3 main models, in order from greatest to lowest AICc:

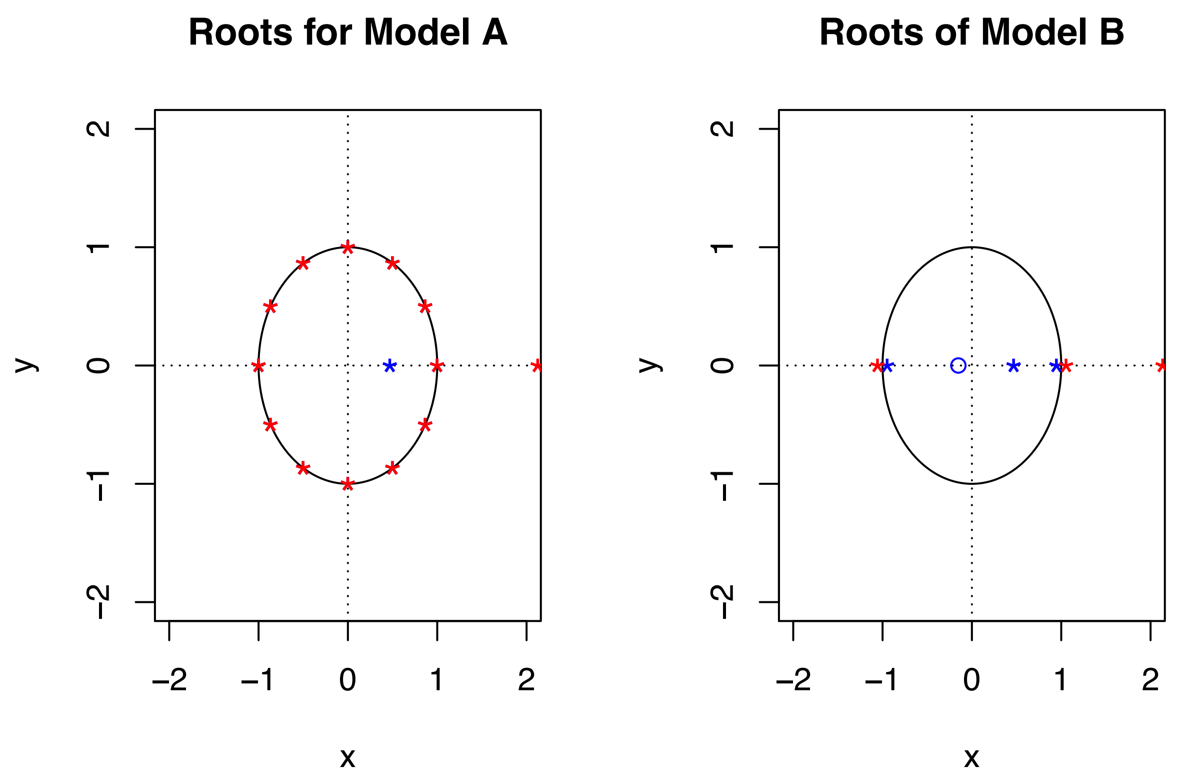
- **Model A** is SMA(0,1,1)x(0,1,1)$_{12}$:
  $\nabla_1\nabla_{12}\text{BC}(U_t)$ = (1 - $0.4704_{0.0990}B$)(1 - $0.9993_{0.5304}B^{12}$)Z_t, which expands to:
  $\nabla_1\nabla_{12}\text{BC}(U_t)$ = (1 - $0.4704B$ - $0.9993B^{12}$ + $0.47007B^{13}$)Z_t
  and we have $\hat{\sigma^2}_Z = 0.5996$
- **Model B** is SARIMA(0,1,1)x(2,1,1)$_{12}$ with SAR1 fixed at 0 (insignificant):
  (1 + $0.1513_{0.1185}B^{12}$)$\nabla_1\nabla_{12}\text{BC}(U_t)$ = (1 - $0.4680_{0.0987}B$)(1 - $0.8979_{0.2834}B^{12}$)Z_t, which expands to:
  (1 + $0.1513_{0.1185}B^{12}$)$\nabla_1\nabla_{12}\text{BC}(U_t)$ = (1 - $0.4680B$ - $0.8979B^{12}$ + $0.4202B^{13}$)Z_t
  and we have $\hat{\sigma^2}_Z = 0.6244$

Checking stationarity and invertibility

A) For **Model A**, it is a SMA process, which we know is always stationary, so we must check invertibility by ensuring that the roots of the characteristic polynomial $\theta(z)$ = 1 - 0.4704z - 0.9993z^{12} + 0.47007z^{13}, executed by the code below.
We immediately see that the roots, represented by red stars, are on the unit circle on many occasions, so therefore our model is not invertible, and we cannot use it.

B) For **Model B**, we have both AR and MA components, so we will check the characteristics of each characteristic polynomial. We already know that MA is stationary, so we check it's characteristic polynomial $\theta(z)$ = 1 - 0.4680z - 0.8979z^{12} + 0.4202z^{13} for invertibility. On the other hand, we know that AR is always invertible by it's composition, so we will check it's characteristic polynomial $\phi(z)$ = 1 + 0.1513z with the code below.
The blue circle represents the inverse of the MA root, and the red stars represent the AR roots, and we see that all the reds are outside of the unit circle, meaning our **Model B is stationary and invertible**.

```
op <- par(mfrow=c(1,2))
plot.roots(NULL,polyroot(c(1,-0.4704,0,0,0,0,0,0,0,0,0,0,-0.9993,0.47007)),
           main = "Roots for Model A")
plot.roots(polyroot(c(1,0.1513)), polyroot(c(1,-0.4680,-0.8979,0.4202)),
           main = "Roots of Model B")
```

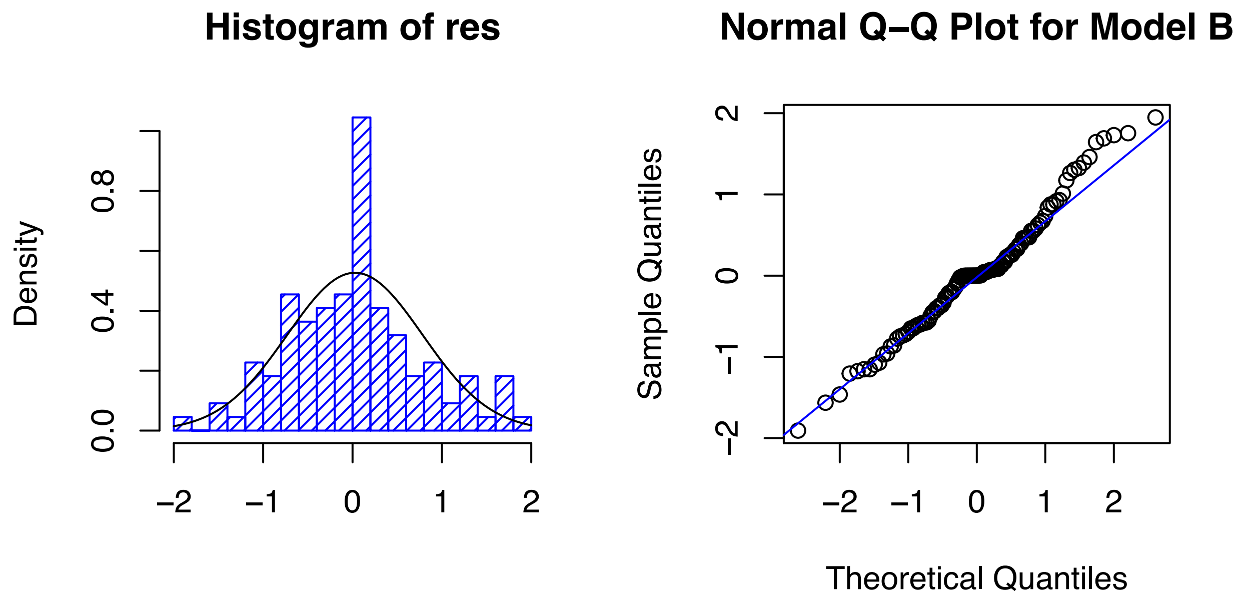


```
par(op)
```

Diagnostic Checking for Model B

First, we look at the residuals of our fitted model to see that they are approximately normal. We plot the histogram and see that our results are very close to the normal curve. Then we we plot normal Q-Q and see that it looks ok, with minimal outliers:

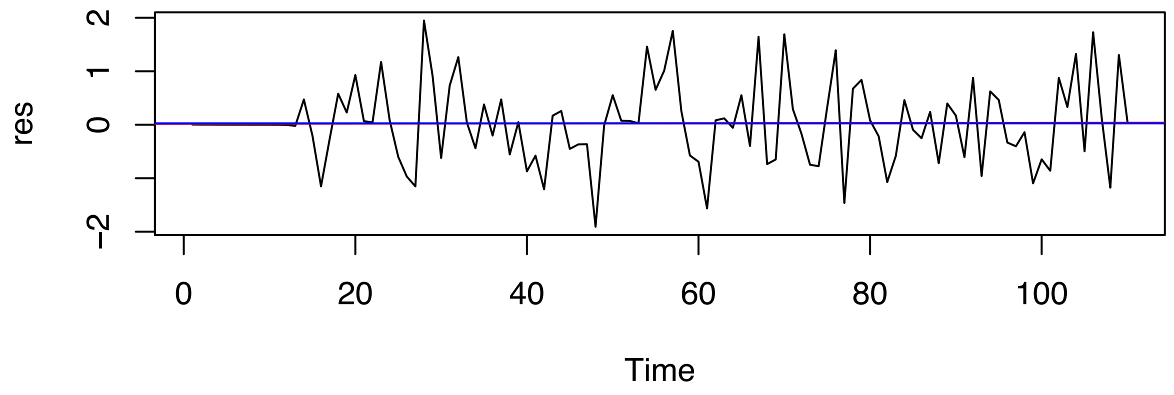
```
fit <- arima(ctrain.bc, order = c(0,1,1), seasonal = list(order=c(2,1,1), period=12),
             fixed = c(NA,0,NA,NA))
res <- residuals(fit)
op <- par(mfrow=c(1,2))
hist(res,density=20,breaks=20, col="blue", xlab="", prob=TRUE)
m <- mean(res)
std <- sqrt(var(res))
curve( dnorm(x,m,std), add=TRUE )
qqnorm(res,main= "Normal Q-Q Plot for Model B")
qqline(res,col="blue")
```



```
par(op)
```

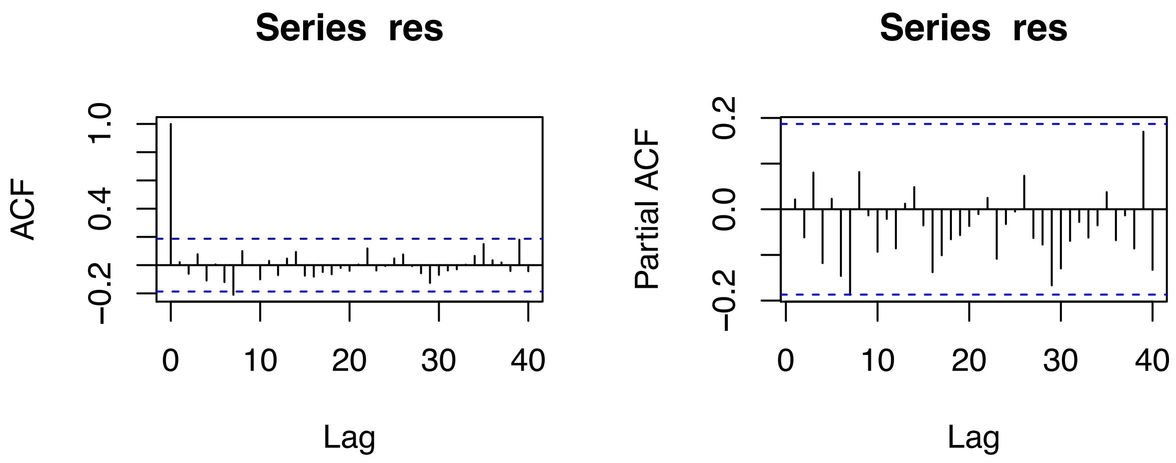
Next, we plot the residuals as a time series and see that they show no signs of trend, no visible change of variance, and no seasonality, and we see that our sample mean is close to 0:

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



Lastly, we check ACF/PACF of residuals, to ensure they within the confidence intervals for all lags:

```r
op <- par(mfrow=c(1,2))
acf(res, lag.max = 40)
pacf(res, lag.max = 40)
```



```r
par(op)
```

The ACF and PACF of the residuals looks pretty good. We may have one outlier at lag 7/8, but we will say this looks good enough.

Tests

Next, we will use the Shapiro test, Box-Pierce test, Ljung-Box test, and Mcleod-Li test, and check for p-values greater than 0.05 to ensure that our residuals pass the normality and linear dependence tests. Lastly, we plug our residuals into Yule-Walker method, setting AIC=TRUE to rank them by AICc, and order.max=NULL to automatically select an order, so if 0 is selected, our residuals are AR(0), or white noise. This is done in the code below:

```r
shapiro.test(res)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res
## W = 0.981, p-value = 0.1186
```

```r
Box.test(res, lag = 11, type = c("Box-Pierce"), fitdf = 3)
```

```
##
##  Box-Pierce test
##
## data:  res
## X-squared = 11.174, df = 8, p-value = 0.192
```

```r
Box.test(res, lag = 11, type = c("Ljung-Box"), fitdf = 3)
```

```
##
##  Box-Ljung test
##
## data:  res
## X-squared = 12.096, df = 8, p-value = 0.147
```

```r
Box.test((res)^2, lag = 11, type = c("Box-Pierce"), fitdf = 0)
```

```
##
##  Box-Pierce test
##
## data:  (res)^2
## X-squared = 8.3323, df = 11, p-value = 0.6833
```

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

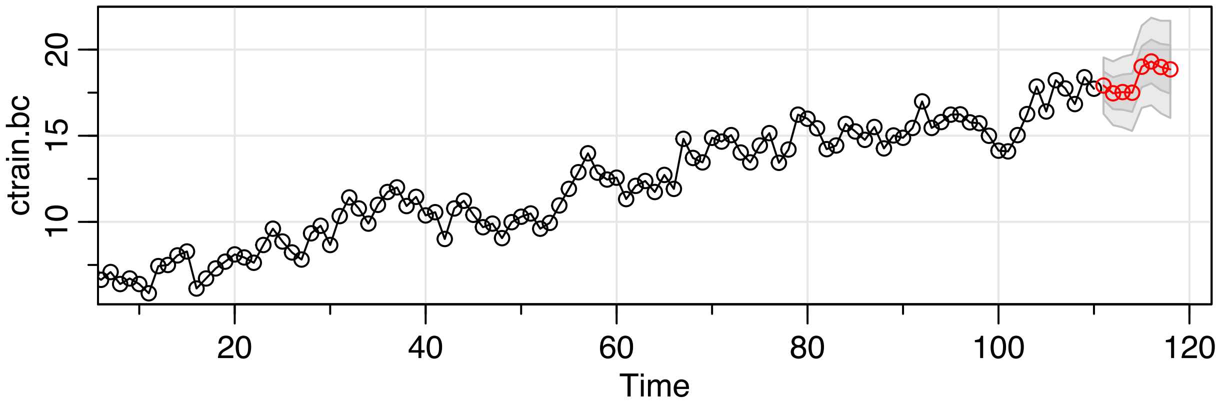
```
##
## Call:
## ar(x = res, aic = TRUE, order.max = NULL, method = c("yule-walker"))
##
##
## Order selected 0  sigma^2 estimated as  0.5732
```

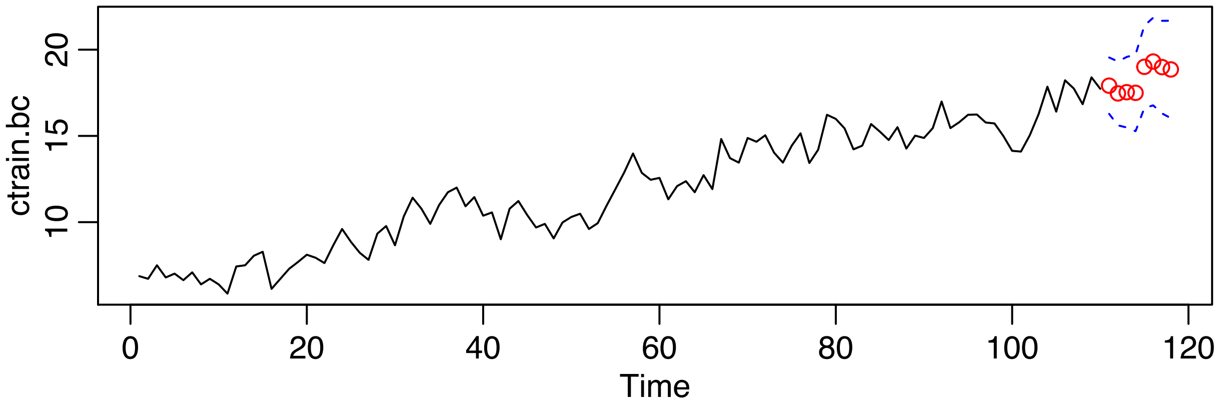
We see that from all our tests, we get a p-value greater than 0.05, so all the tests pass. Finally, from the Yule-Walker method, we obtain an order of 0, meaning our residuals fit AR(0), or white noise. So we have passed diagnostic checking and are ready for forecasting.

Forecasting

First, we forecast on our Box-Cox transformed data BC(U_t):

```
op <- par(mfrow=c(2,1))
library(forecast)
library(astsa)
pred.tr <- sarima.for(xdata = ctrain.bc, n.ahead = 8, p=0, d=1, q=1, P=2, D=1, Q=1, S=12, fixed = c(NA,
U.tr <- pred.tr$pred + 2*pred.tr$se
L.tr <- pred.tr$pred - 2*pred.tr$se
ts.plot(ctrain.bc, xlim=c(1,length(ctrain.bc)+8), ylim = c(min(ctrain.bc),max(U.tr)))
lines(U.tr, col="blue", lty="dashed")
lines(L.tr, col="blue", lty="dashed")
points((length(ctrain.bc)+1):(length(ctrain.bc)+8), pred.tr$pred, col="red")
```

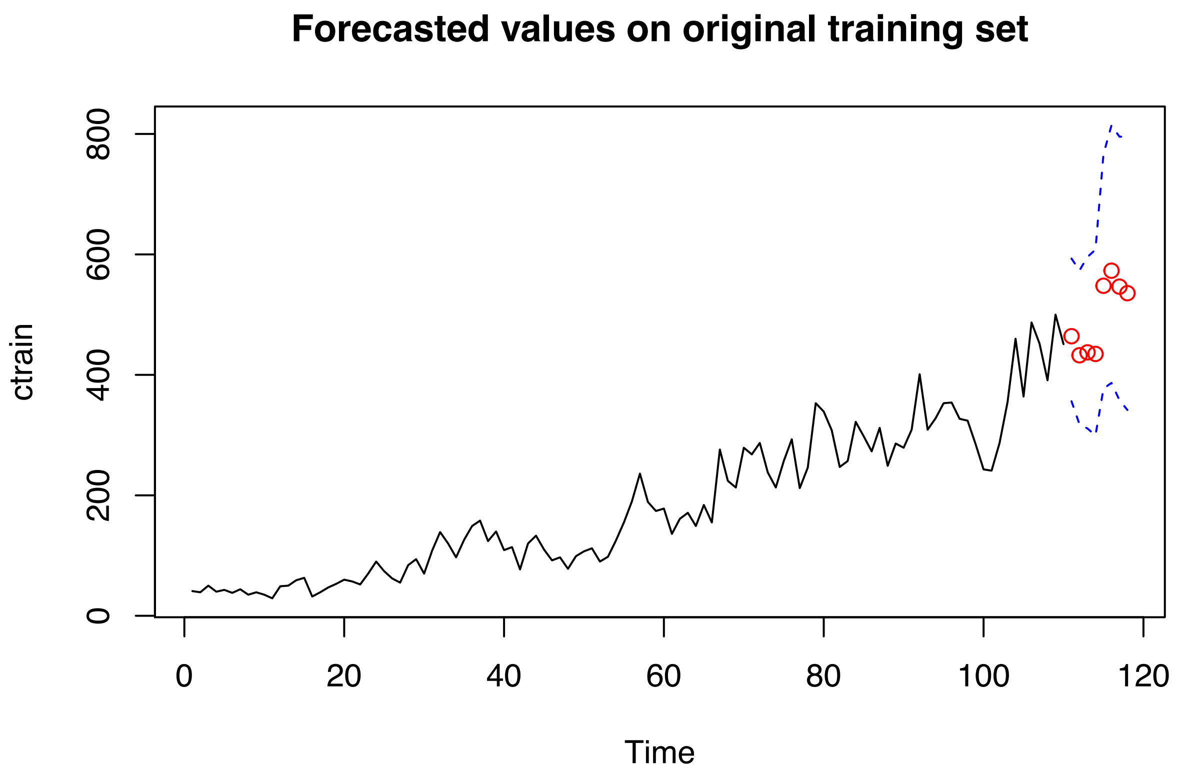




```
par(op)
```

Then, we invert the box-cox transformation and forecast on original data:

```
pred.orig <- (lambda*pred.tr$pred+1)^(1/lambda)
U <- (lambda*U.tr+1)^(1/lambda)
L <- (lambda*L.tr+1)^(1/lambda)
plot.ts(ctrain, xlim=c(1,length(ctrain)+8), ylim = c(min(ctrain),max(U)),
        main = "Forecasted values on original training set")
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points((length(ctrain)+1):(length(ctrain)+8), pred.orig, col="red")
```

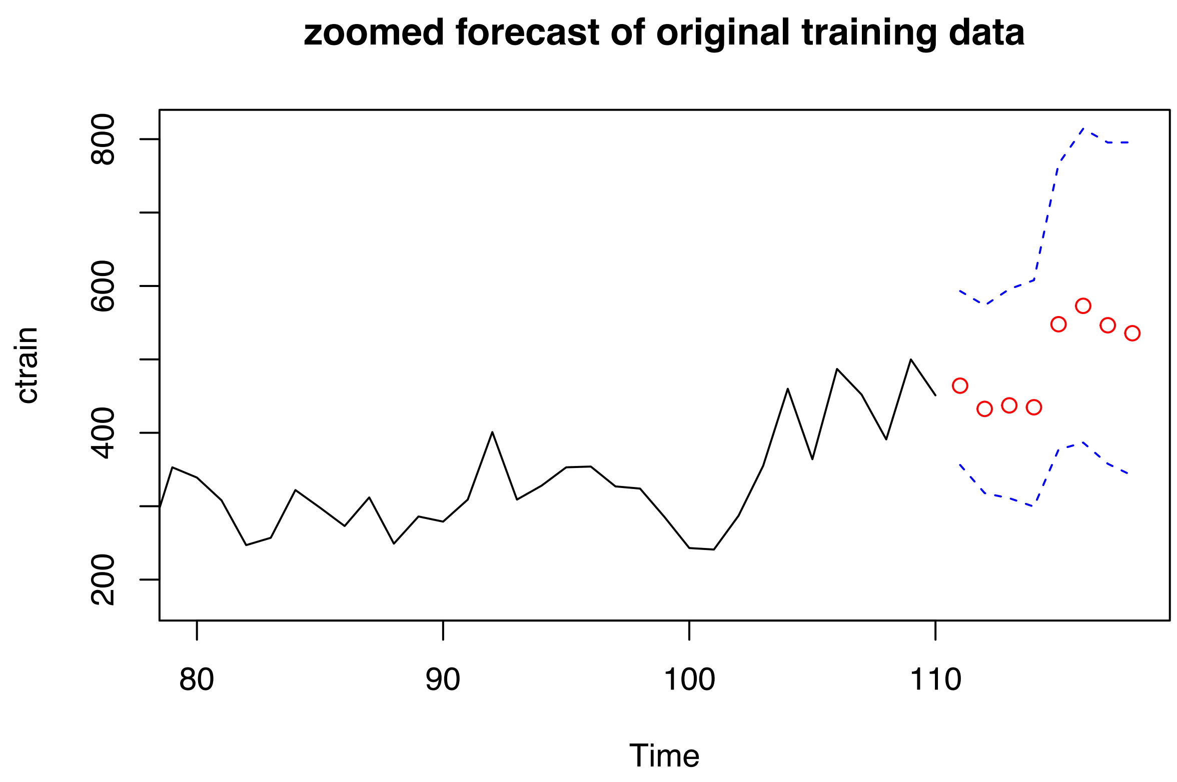


Our forecast looks good! We see that it predicts the seasonality and increases well, and seems to follow the general patterns of our data.

Zoom in

We will zoom in to get a better look at the results from the previous plot, starting from month 80:

```r
ts.plot(ctrain, xlim = c(80,length(ctrain)+8), ylim = c(170,max(U)),
        main="zoomed forecast of original training data")
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points((length(ctrain)+1):(length(ctrain)+8), pred.orig, col="red")
```

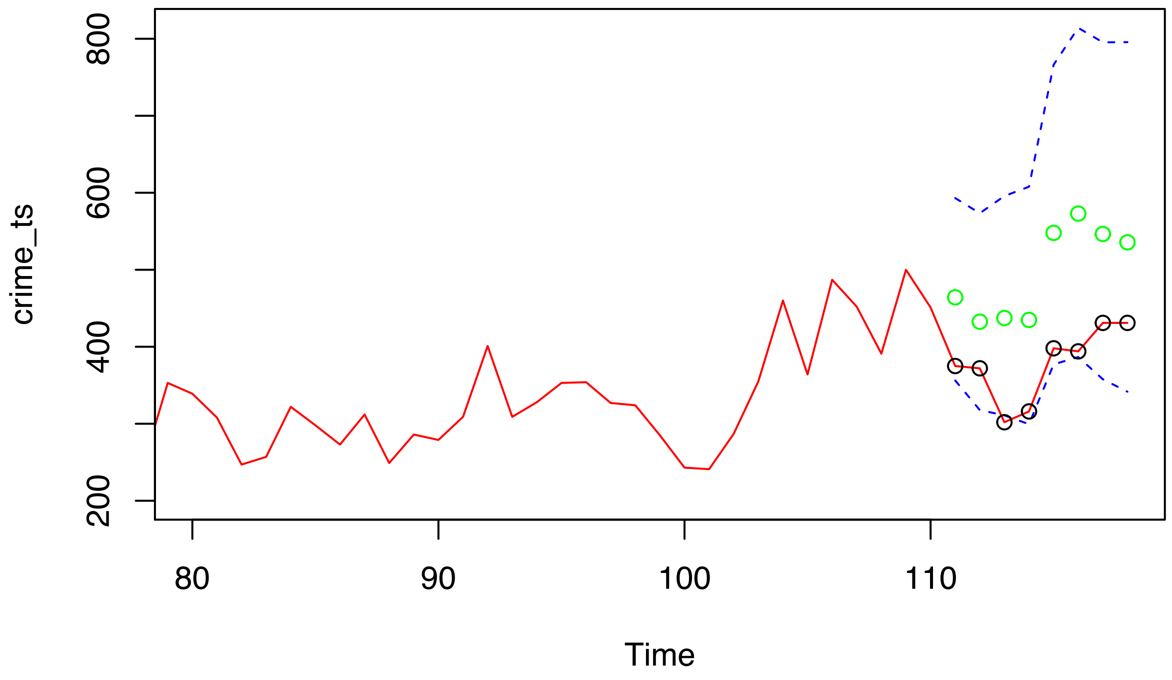


Once again, my forecast looks pretty good!

Zoomed forecast and true values

Finally, we will plot a zoomed version of our original time series, plot our inverted predicted values, as well as the true values from our testing set ctest, and compare them to conclude how accurate our model was at truly predicting armed robberies in Boston:

```r
ts.plot(crime_ts, xlim = c(80,length(ctrain)+8), ylim = c(200,max(U)), col="red")
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points((length(ctrain)+1):(length(ctrain)+8), pred.orig, col="green")
points((length(ctrain)+1):(length(ctrain)+8), ctest, col="black")
```



Conclusion

- The main goal of the project was to analyze an existing data set of Boston armed robberies from January 1966 to October 1975, fit it to a model, and forecast the data in order to try to predict armed robberies in Boston to alert abiding citizens and law enforcement, making them better prepared and ultimately safer.

- The goals were not quite achieved, as the data seemed to get lost in translation from the box-cox forecast to the forecasting on the original data. We see that the model predicts the patterns of the data well, but are slightly too high to be considered overtly accurate. Finding the model with the lowest AICc proved to be the most difficult part for me, but I ended up finding a reliable model that passed all tests of linear dependence and normality, which is SARIMA(0,1,1)x(2,1,1)$_{12}$ with SAR1 fixed at 0 (insignificant):
  $(1 + 0.1513_{0.1185}B^{12})\nabla_1\nabla_{12}\text{BC}(U_t) = (1 - 0.4680_{0.0987}B)(1 - 0.8979_{0.2834}B^{12})Z_t$.

- Professor Raya Feldman aided me in this project in helping try to invert my data back to original for forecasting in the last few steps. Thanks to her for her guidance on this project and throughout the course.