Forecasting the NBA's Popularity

Ethan Choi

University of California, Santa Barbara

Abstract

The data I work with in this project is the monthly popularity score of the search term "NBA" in the United States from January 2004 to December 2019, according to Google Trends. In this dataset, each month is assigned a score from 0 to 100 that represents search interest relative to the highest score received during this time period. I used this dataset to create a time series model that is able to accurately forecast the NBA's popularity in each month of 2020.

In order to create my model, I applied Box-Jenkins methodology to this data and determined that the best model was a SARIMA$(0,1,2)(0,1,1)_{12}$. This model passed all diagnostic tests, except the Shapiro-Wilk test, and had a 95% prediction interval within which all test data fell into. This led me to conclude that, although this model makes accurate predictions, a heavy-tailed model or a non-linear model may be even better suited for this data.

Introduction

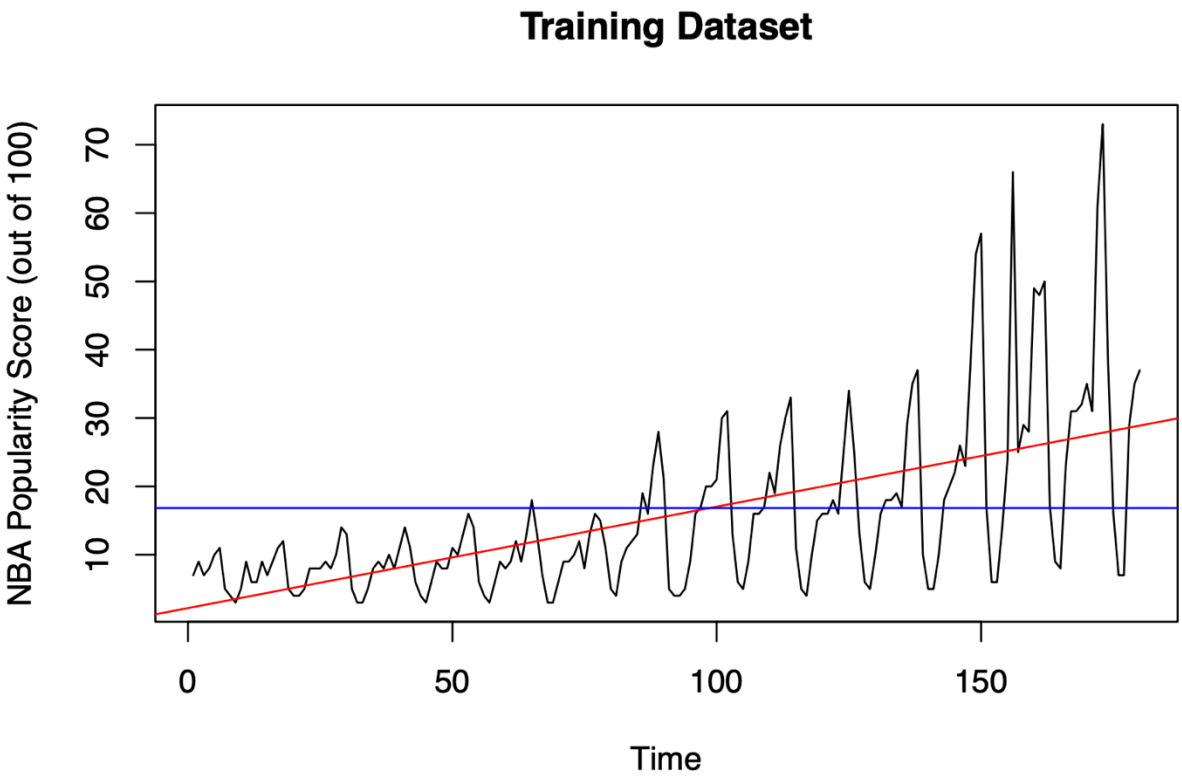
The goal of this project is to forecast the NBA's popularity in 2020 had the COVID-19 pandemic not disrupted the 2020 NBA season. To do this, I used Google Trends data that assigned a relative popularity score (0 to 100) to each month from January 2004 to December 2019 based on the number of Google searches of the term "NBA". I then used R to transform and difference this data in order to make it stationary. Making this data stationary allowed me to compare a series of simple algebraic models based on their AICc and ability to pass diagnostic tests. I compared the diagnostics of four models and determined that the best one was SARIMA$(0,1,2)(0,1,1)_{12}$. This is because it did not have any unit roots, unlike two of the other models I tested, and had the lowest AICc of any of the four models I took to diagnostic checking.

Although SARIMA$(0,1,2)(0,1,1)_{12}$ turned out to be the best model of any of the ones I tested, none of my models passed the Shapiro-Wilk test. This leads me to believe that a non-linear or heavy tailed model may fit this data better and create even more accurate forecasts.

Regardless, this data is important because it is directly tied to the NBA's revenue. If the NBA is able to accurately predict an upward trend in popularity, using a time series model such as mine, it can use this forecast as an important negotiation piece when talking to TV stations and other media entities about how much to charge for rights to broadcast NBA games or run advertisements during games.

Section I: Exploratory Data Analysis

The graph produced below is of my training dataset. This training dataset contains data from January 2004 to December 2019. It is also the data I based my model upon, then validated using a test dataset which contains the NBA Popularity Score from each month of 2020.

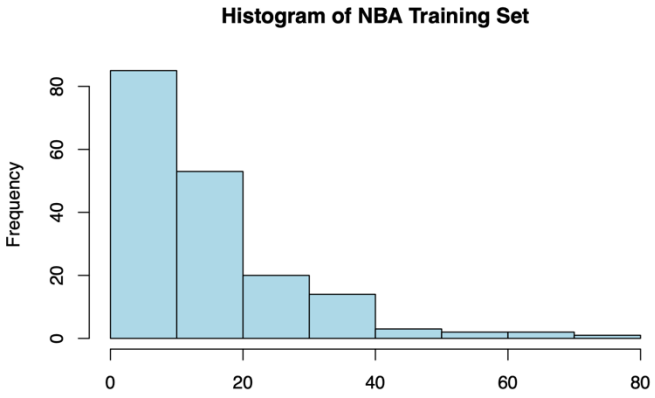


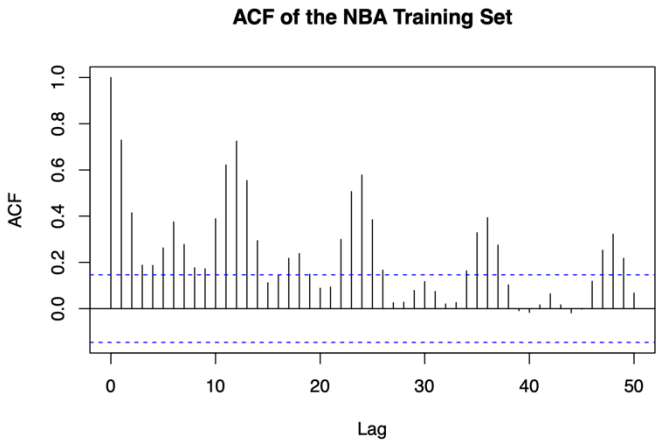
This graph of the time series shows that it is highly nonstationary. It exhibits positive linear trend (shown by the red line), seasonality (rising and falling periodically), and increasing variance (larger and larger changes in behavior as time goes on), which are all signs of nonstationarity.

This plot also shows why I expected the residuals of my model to fail the Shapiro-Wilk test. The graph above is characterized by short rises followed by sudden drops which correspond to the NBA being in season versus when it is not in season. This is non-Gaussian behavior and is actually similar to the data on slide 19 of Lecture 12 which displays a threshold model. So,

because the data displays non-Gaussian behavior, I do not expect the residuals of my model to pass the Shapiro-Wilk test, which tests to see if the residuals are normally distributed.

Additionally, in order to further prove that this time series is nonstationary, I plotted the histogram and ACF of the training set data.

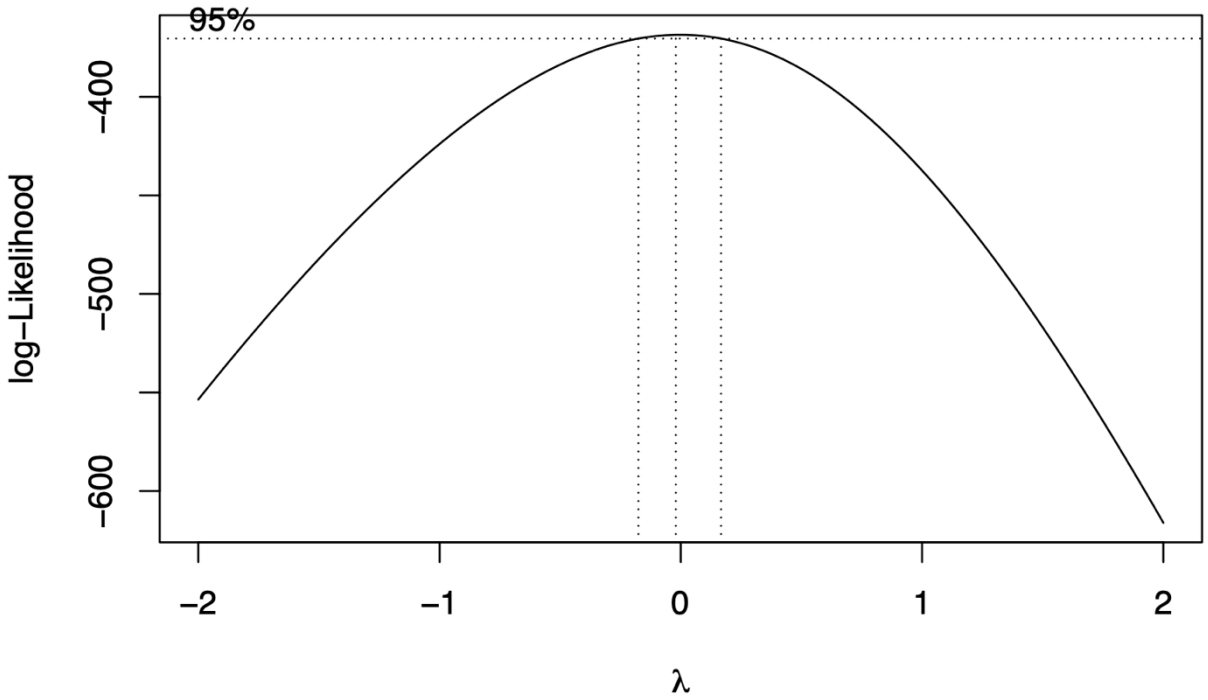




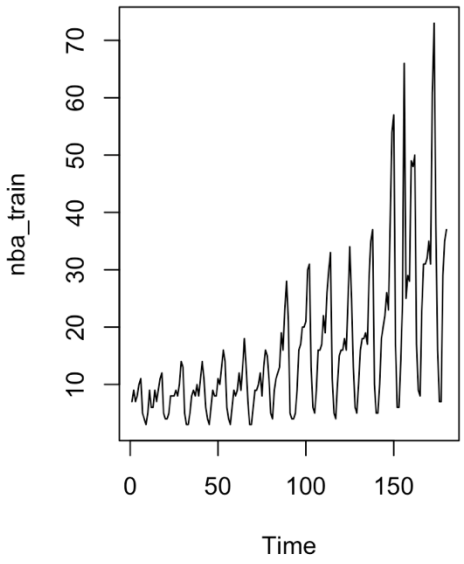
The histogram is very skewed to the right and the ACF remains large and periodic even at high lags. Both of these are indicators of nonstationary data. Therefore, in order to make this time series stationary and suitable for model estimation, I looked into transforming and differencing the data, which is done in the next section.

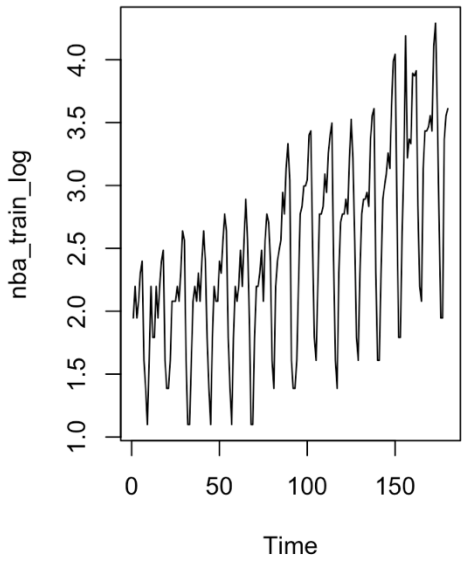
Section II: Transforming and Differencing the Time Series

First, in order to deal with the non-constant variance displayed in my data, I attempted to use a Box-Cox transformation to stabilize the variance.



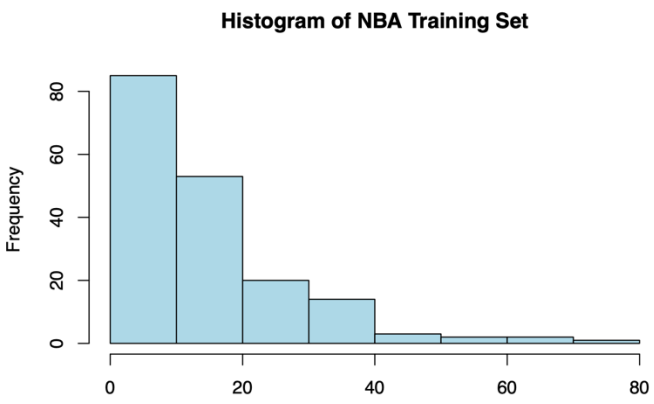
Because the 95% confidence interval for λ does not contain 1, I knew this data required a transformation. Also, because this 95% confidence interval for λ contains 0, I decided to try a log transformation of the data.

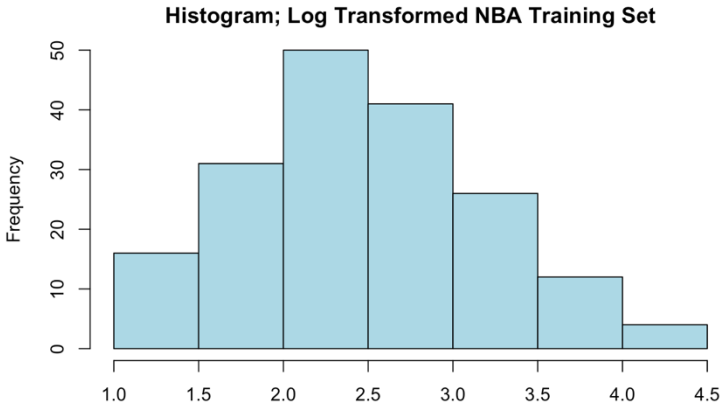





```
> var(nba_train)
[1] 161.7894
> var(nba_train_log)
[1] 0.5363114
```

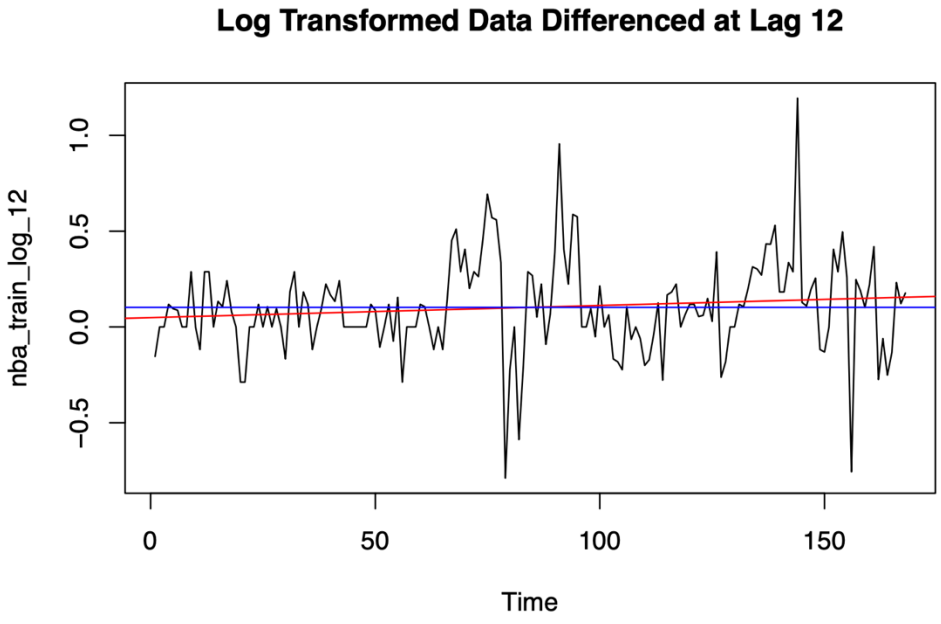
As shown by the graphs and the actual values of the variances, it is easy to see that applying a log transformation to the data significantly reduces the variance (from 161.7894 to 0.5363114). Therefore, I decided to follow through with it.





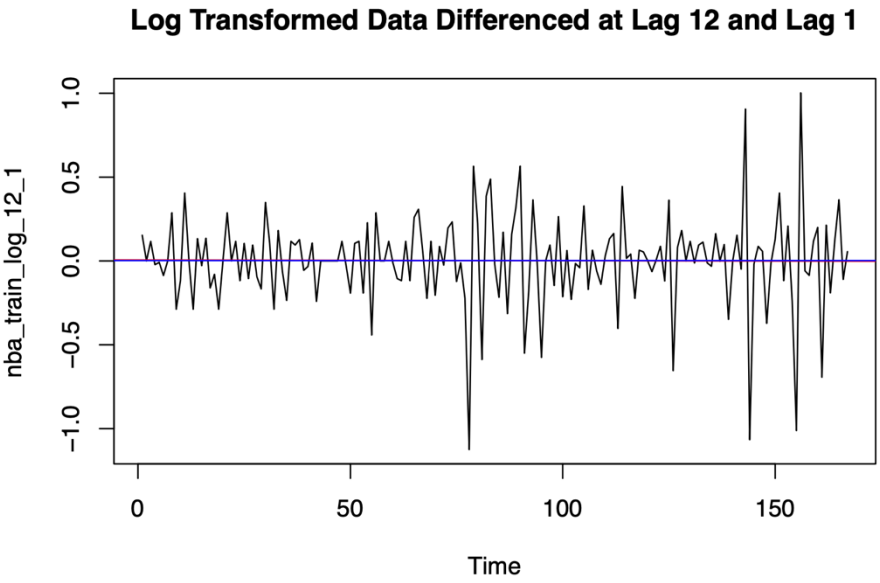
Additionally, the log-transformed data produced a much more symmetric, normal-looking histogram compared to the untransformed data, giving me another reason to use this transformation.

Next, in order to deal with the seasonality shown in the data, I tried differencing the log-transformed data once at lag 12.




```
> var(nba_train_log)
[1] 0.5363114
> var(nba_train_log_12)
[1] 0.06216348
```

Applying this difference lowered the variance (from 0.5363114 to 0.06216348) and produced data that looks much more like white noise. Therefore, I decided to follow through with it. However, there is still a slight linear trend, as shown by the red line on the graph above. Thus, I decided to try differencing this data once at lag 1.




```
> var(nba_train_log_12)
[1] 0.06216348
> var(nba_train_log_12_1)
[1] 0.07753534
```

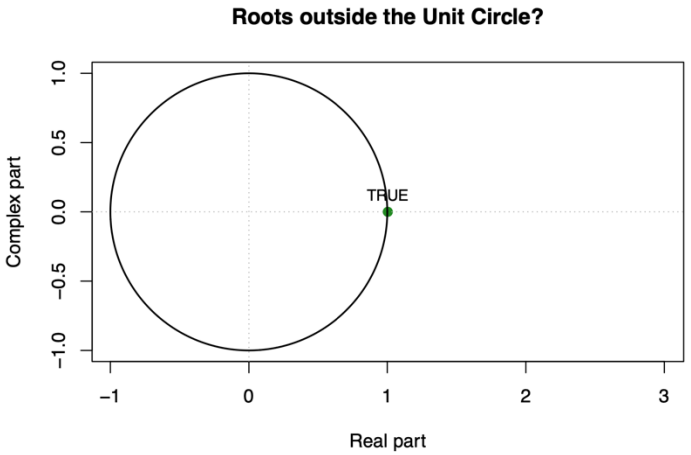
Even though applying this difference seems to produce white noise and eliminate linear trend altogether, I initially decided not to use this differencing at lag 1 because it raised the variance compared to the log transformed data only differenced at lag 12 once (from 0.06216348 to 0.07753534). However, upon fitting and analyzing the two best models created using the log transformed data differenced at lag 12 once (Model A and Model B in my code), I found that both of them contained unit roots in the AR part, which indicated underdifferencing.

```r
# Checking stationarity and inveritibility of Model A
library(UnitCircle)
model_a
```

```
##
## Call:
## arima(x = nba_train_log, order = c(1, 0, 2), seasonal = list(order = c(0, 1,
##     1), period = 12), method = "ML")
##
## Coefficients:
##          ar1      ma1      ma2     sma1
##       0.9963  -0.5344  -0.3184  -0.6790
## s.e.  0.0058   0.0773   0.0821   0.0616
##
## sigma^2 estimated as 0.03614:  log likelihood = 36.52,  aic = -63.03
```

```r
# Checking AR part
uc.check(pol_ = c(1, -0.9963), plot_output = TRUE) # PASS BUT SUPER CLOSE (CONSIDER UNIT ROOT)
```

```
##       real complex outside
## 1 1.003714       0    TRUE
## *Results are rounded to 6 digits.
```

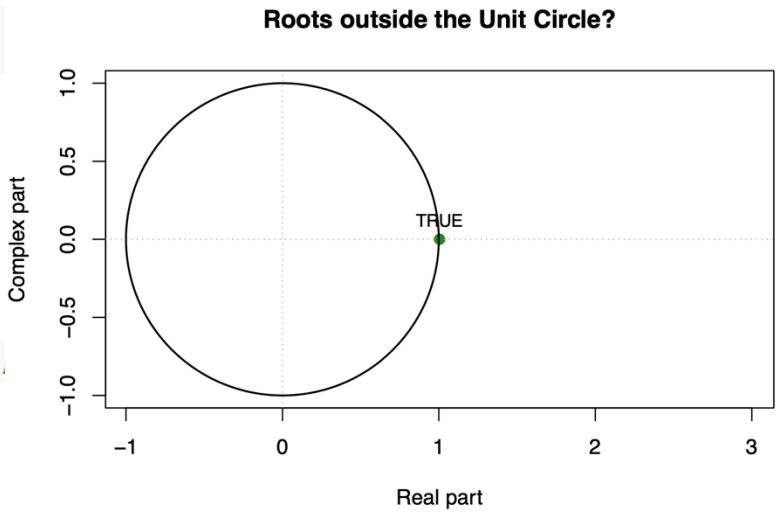



```r
# Checking stationarity and inveritibility of Model B
library(UnitCircle)
model_b
```

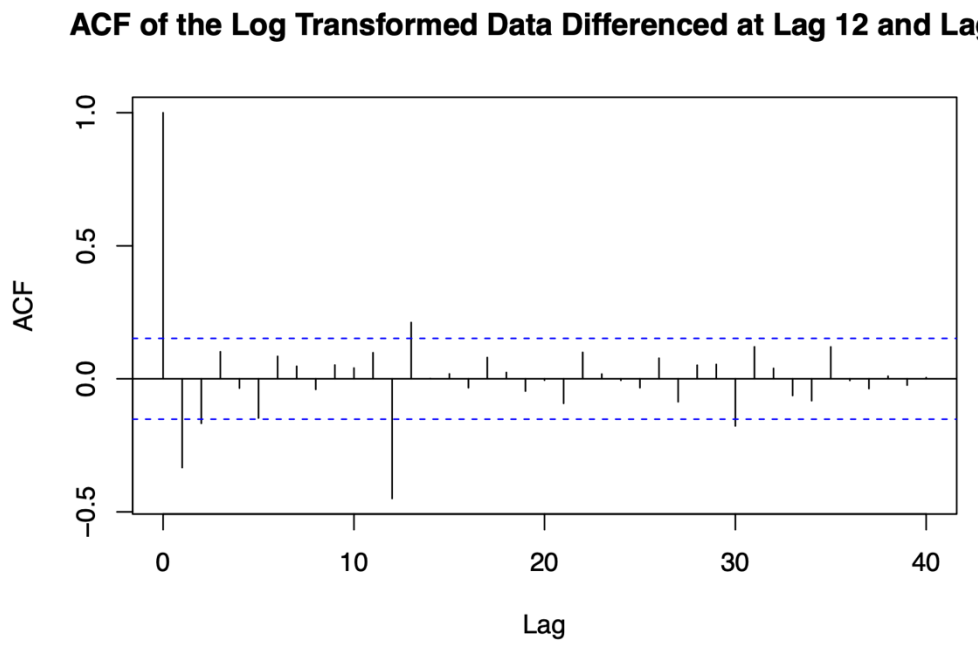
```
##
## Call:
## arima(x = nba_train_log, order = c(1, 0, 2), seasonal = list(order = c(1, 1,
##     1), period = 12), method = "ML")
##
## Coefficients:
##          ar1      ma1      ma2     sar1     sma1
##       0.9950  -0.5268  -0.3119  -0.1310  -0.6095
## s.e.  0.0073   0.0791   0.0836   0.1087   0.0918
##
## sigma^2 estimated as 0.03582:  log likelihood = 37.21,  aic = -62.42
```

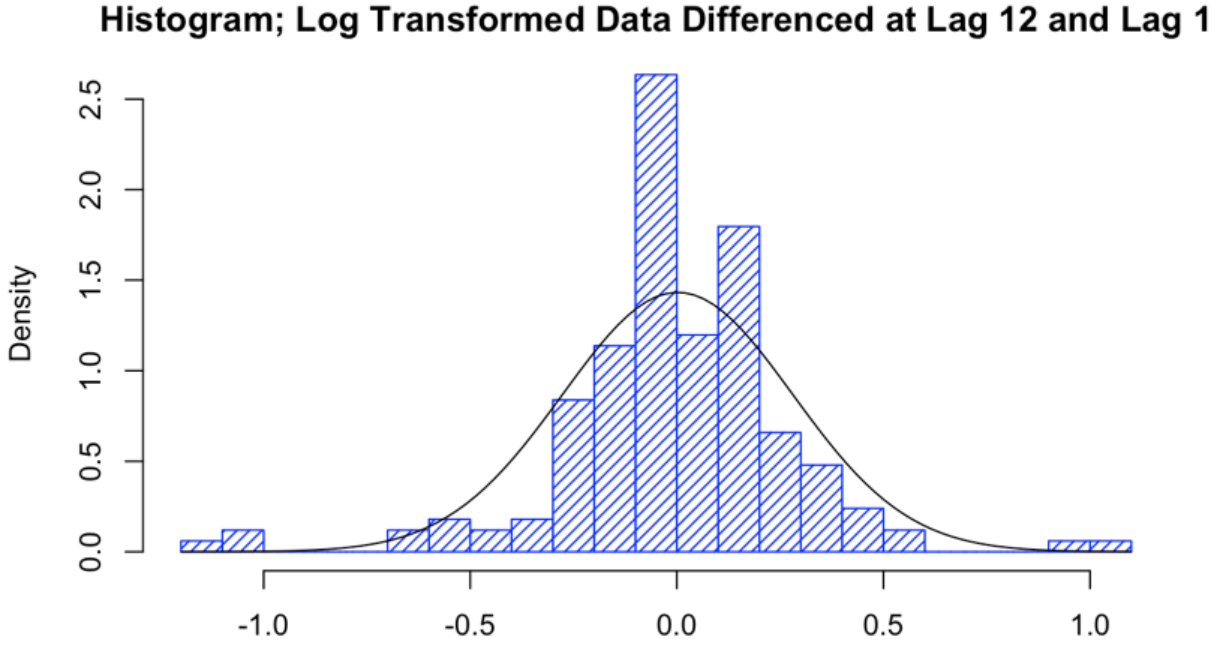
```r
# Checking AR part
uc.check(pol_ = c(1, -0.9950), plot_output = TRUE) # PASS BUT SUPER CLOSE (CONSIDER UNIT
```

```
##       real complex outside
## 1 1.005025       0    TRUE
## *Results are rounded to 6 digits.
```



Therefore, I decided to use the log transformed data differenced at lag 12 and lag 1 even though the variance slightly increased after differencing at lag 1. Additionally, to confirm my decision, I analyzed the ACF and histogram of the log transformed data differenced once at lag 12 and once lag 1.

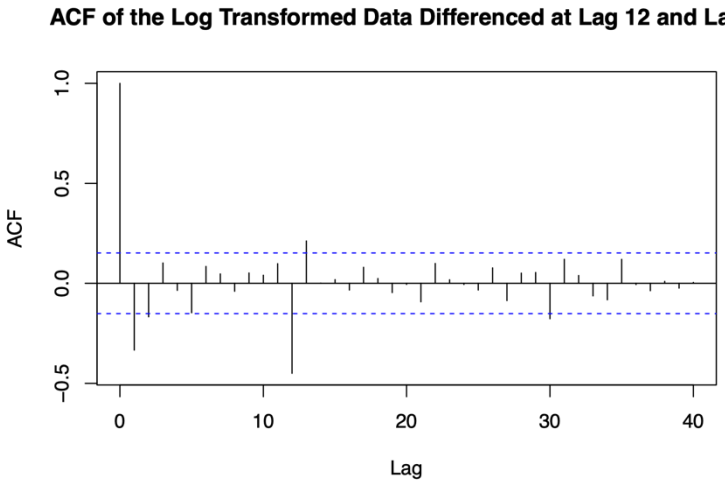


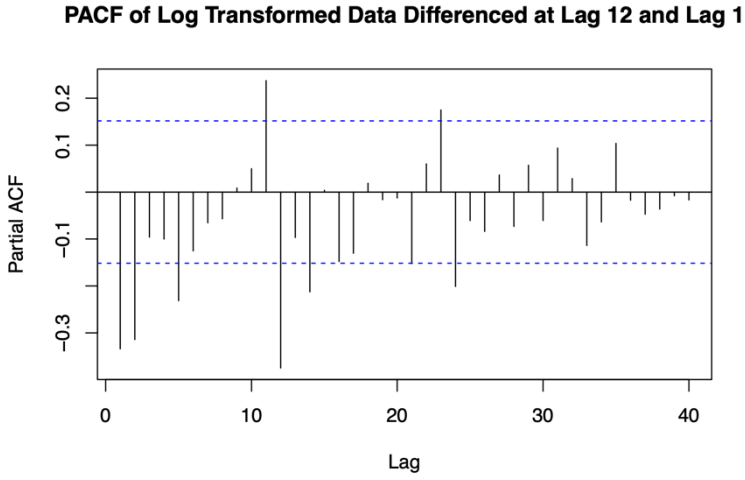


The ACF looks like that of a stationary process and the histogram follows the normal curve decently well, albeit some heavy tailed outliers. And once again, we do not expect the histogram to follow the normal curve exactly because our data has some non-Gaussian characteristics and too few observations (Lecture 11 Slide 18).

So, now that I have a stationary dataset, I can analyze the ACF and PACF of the log transformed data differenced at lag 12 and lag 1 to form preliminary estimates of p, q, P, and Q.

Section III: Analysis of ACF and PACF for Preliminary Model Estimation





Firstly, from my differencing choices, I know that **d=1** (because I differenced at lag 1 once), **D=1** (because I differenced at lag 12 once), and **s=12** (because I am working with monthly data).

Next, in order to determine q and Q, I looked at the ACF and identified lags at which the graph lies outside the confidence interval (represented by the blue lines). The ACF lies outside of the confidence interval at lags 0,1,2,12,13, and 30. Because the ACF lies outside of the confidence interval at lag 12, I suspect **Q = 0 or 1.** Additionally, because the ACF lies outside of the confidence interval at lags 1 and 2, I suspect **q = 0, 1, or 2.** If q = 1 this would also explain why the ACF lies outside the confidence interval at lag 13. Also, just to note, I chose to disregard the ACF lying outside of the confidence interval at lag 30 because of Bartlett's formula.

Finally, in order to determine p and P, I looked at the PACF and identified lags at which the graph lies outside the confidence interval. The PACF lies outside of CI at lags 1, 2, 5, 11, 12, 14, 23, and 24. Because the PACF lies outside of the confidence interval at 12 and 24, I suspect **P = 0, 1, or 2.** Additionally, because the PACF lies outside of the confidence interval at lags 1, 2, and 5, I suspect **p = 0, 1, 2, or 5.**

Section IV: Fitting Models and Diagnostic Checking

Based on my preliminary estimations for p, q, P, and Q, I created a series of candidate models using every possible combination of each of these parameters (72 total) and compared their AICcs.

```
# Candidate models:
p <- c(0, 1, 2, 5)
df_test <- expand.grid(p=p, q=0:2, P=0:2, Q=0:1)
df_test <- cbind(df_test, AICc=NA)
# Testing Models and Computing AICcs:
for (i in 1:nrow(df_test)) {
sarima.obj <- NULL
try(arima.obj <- arima(nba_train_log, order=c(df_test$p[i], 1, df_test$q[i]),
seasonal=list(order=c(df_test$P[i], 1, df_test$Q[i]), period=12),
method="ML"))
if (!is.null(arima.obj)) { df_test$AICc[i] <- AICc(arima.obj) }
# print(df[i, ])
}
```

I then looked at the 6 models with the lowest AICcs in order to determine which ones were the best candidates to move onto diagnostic checking.

```
head(df_test[order(df_test$AICc),]) # sorting models by lowest AICC
```

```
##    p q P Q      AICc
## 45 0 2 0 1 -63.03148
## 42 1 1 0 1 -62.55753
## 57 0 2 1 1 -62.24675
## 54 1 1 1 1 -61.23708
## 69 0 2 2 1 -61.12960
## 46 1 2 0 1 -60.94909
```

From this, I chose to proceed with SARIMA$(0,1,2)(0,1,1)_{12}$ because it has the lowest AICc (-63.03148) and is tied for the least amount of parameters (3) of any of the six models with the lowest AICcs. Throughout the rest of my report and code, I refer to this model as “Model C”.

I also considered SARIMA(1,1,1)(0,1,1)$_{12}$ because it has the second lowest AICc (-62.55753) and is tied for the least amount of parameters (3) of any of the six models with the lowest AICcs. I refer to this model as "Model D" throughout the rest of my report and code.

Putting these models into algebraic form, where X_t symbolizes my log transformed training set, I obtained the following equations:

```
# Checking Coefficients of Model C
# Checking coefficients of SARIMA(0,1,2)(0,1,1)12 (Model C)
model_c <- arima(nba_train_log, order=c(0,1,2), seasonal = list(order = c(0,1,1),
                                                                period = 12), method = "ML")
model_c
```

```
##
## Call:
## arima(x = nba_train_log, order = c(0, 1, 2), seasonal = list(order = c(0, 1,
##     1), period = 12), method = "ML")
##
## Coefficients:
##           ma1      ma2     sma1
##       -0.5366  -0.3233  -0.6828
## s.e.   0.0766   0.0809   0.0605
##
## sigma^2 estimated as 0.03619:  log likelihood = 35.58,  aic = -63.17
```

Model C Equation: $(1-B)(1-B^{12})X_t = (1-0.5366B-0.3233B^2)(1-0.6828B^{12})Z_t$

```
# Checking Coefficients of Model D
# Checking coefficients of SARIMA(1,1,1)(0,1,1)12 (Model D)
model_d <- arima(nba_train_log, order=c(1,1,1), seasonal = list(order = c(0,1,1),
                                                                period = 12), method = "ML")
model_d
```

```
##
## Call:
## arima(x = nba_train_log, order = c(1, 1, 1), seasonal = list(order = c(0, 1,
##     1), period = 12), method = "ML")
##
## Coefficients:
##           ar1      ma1     sma1
##       0.3716  -0.9335  -0.6794
## s.e.  0.0901   0.0468   0.0611
##
## sigma^2 estimated as 0.03623:  log likelihood = 35.35,  aic = -62.69
```

Model D Equation: $(1-0.3716B)(1-B)(1-B^{12})X_t = (1-0.9335B)(1-0.6794B^{12})Z_t$

Before taking Model C and Model D to diagnostic checking, though, I created a confidence interval for each coefficient in each model and checked to see if any of them contained 0. If a confidence interval for a coefficient contains 0, a model may improve (obtain a lower AICc) after removing that coefficient from the model.

```
# 95% CIs for Coefficients of Model C
confint(model_c)
```

```
##             2.5 %     97.5 %
## ma1  -0.6867375 -0.3863894
## ma2  -0.4818628 -0.1646390
## sma1 -0.8012891 -0.5642391
```

```
# 95% CIs for Coefficients of Model D
confint(model_d)
```

```
##             2.5 %     97.5 %
## ar1    0.1950018  0.5481501
## ma1   -1.0252150 -0.8417000
## sma1 -0.7991011 -0.5597100
```

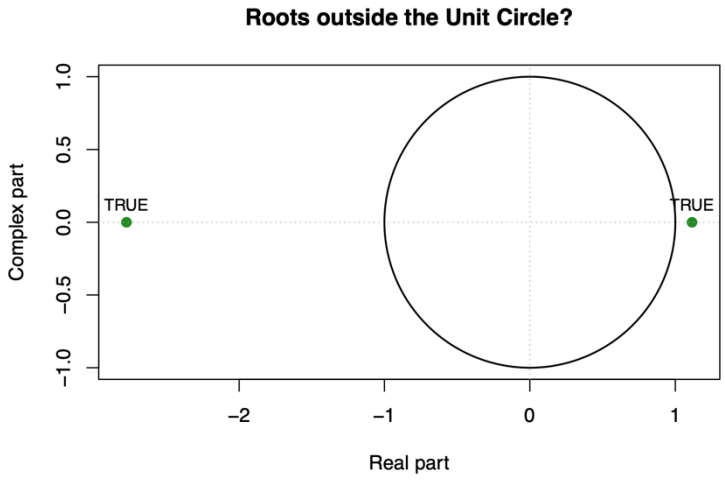
Luckily for me, none of the coefficients' confidence intervals in either model contained 0. Therefore, I did not have to test any variations of either model.

Lastly, in order to confirm that my models that my models were stationary and invertible, I checked each of them for unit roots.

I first completed this process for Model C. Before doing any analysis, however, I already knew Model C was stationary because it is a pure moving average model. I also knew that the seasonal moving average part of my model was stationary and invertible because the absolute value of the coefficient in that part (-0.6828) is less than 1. Therefore, I only had to check for unit roots in the moving average part of my model.

```
# Seasonal part is invertible because the absolute value of the coefficient is less than 1
# Checking MA part
uc.check(pol_ = c(1, -0.5366, -0.3233), plot_output = TRUE) # PASS
```

```
##        real complex outside
## 1  1.114806       0    TRUE
## 2 -2.774565       0    TRUE
## *Results are rounded to 6 digits.
```

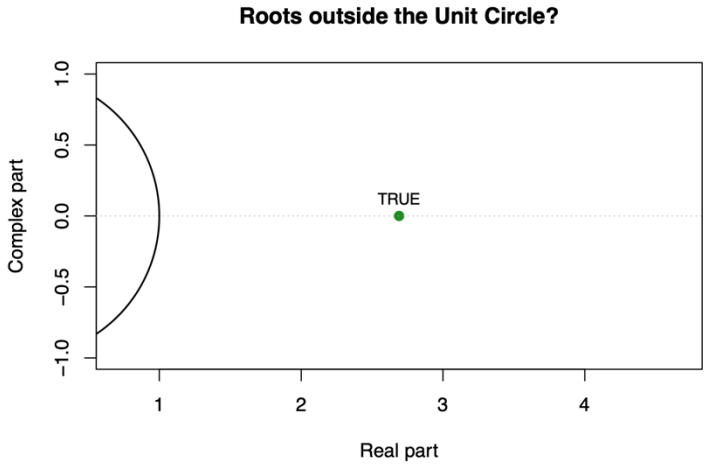


Since the roots of the moving average part in Model C fall outside of the unit circle, I can confirm it is stationary and invertible.

I then repeated this process of checking for unit roots for Model D. And, once again, like Model C, before doing any analysis of unit roots, I am able to determine that the seasonal moving average part of Model D is stationary and invertible because the absolute value of the coefficient in that part (-0.6794) is less than 1. Thus, I only had to check for unit roots in the moving average and autoregressive parts.

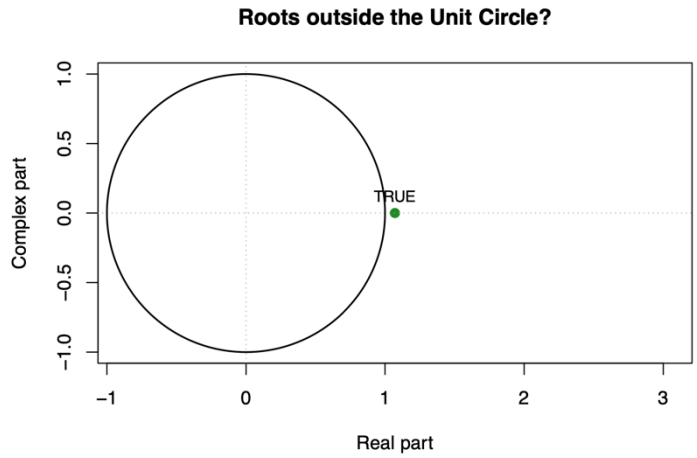
```
# Seasonal part is stationary and invertible
# because the absolute value of the coefficient is less than 1
# Checking AR part
uc.check(pol_ = c(1, -0.3716), plot_output = TRUE) # PASS
```

```
##       real complex outside
## 1 2.691066       0    TRUE
## *Results are rounded to 6 digits.
```




```
# Checking MA part
uc.check(pol_ = c(1, -0.9335), plot_output = TRUE) # PASS
```

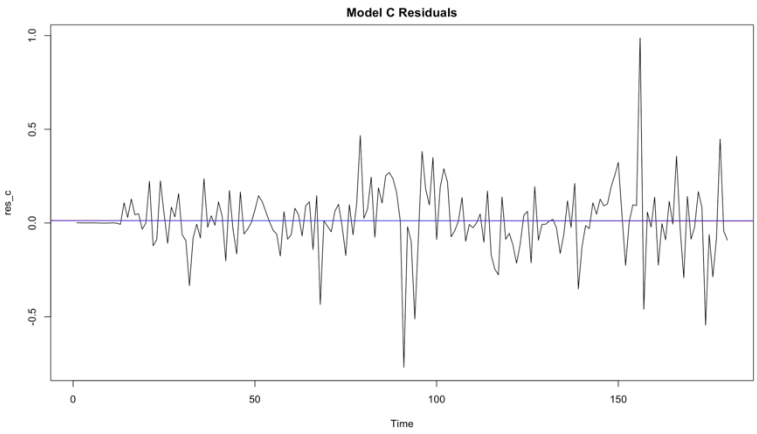
```
##       real complex outside
## 1 1.071237       0    TRUE
## *Results are rounded to 6 digits.
```

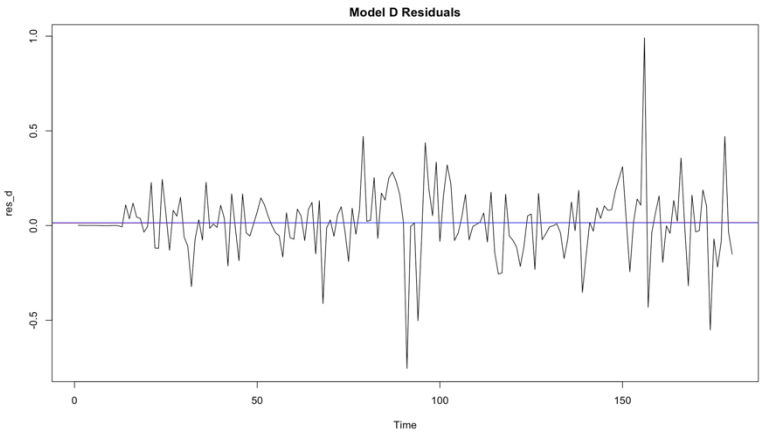


Neither the moving average part nor the autoregressive part of Model D contains unit roots, therefore Model D is stationary and invertible.

Because both Model C and Model D are stationary and invertible, I took both of them to diagnostic checking.

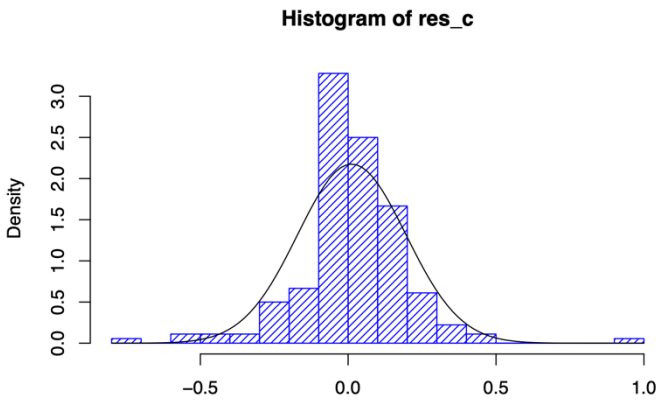
The main idea of diagnostic checking is that if the residuals of a model resemble Gaussian WN(0,1) then that model is a good fit (Lecture 11 Slide 5). Therefore, I first plotted the residuals of both models to see if they resembled white noise.

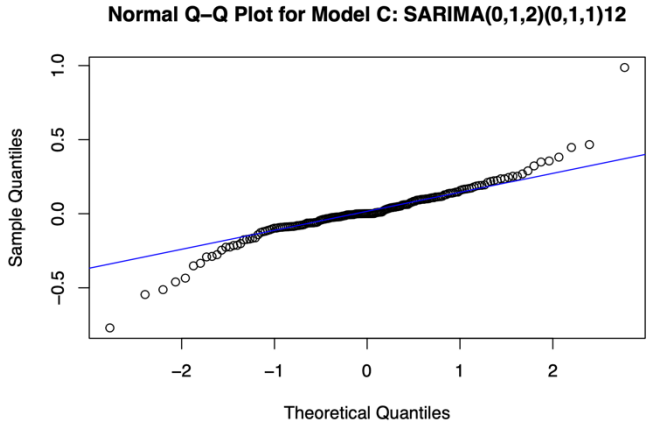




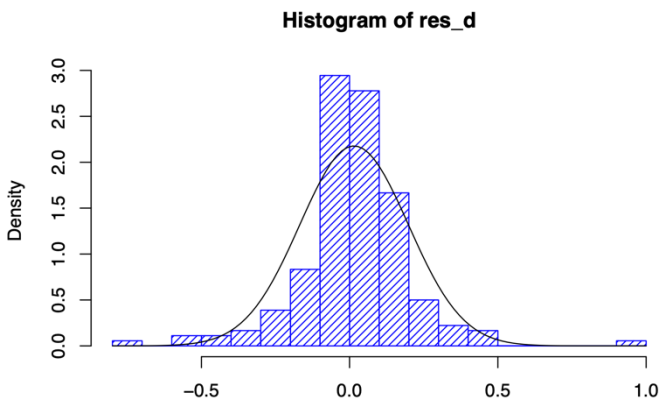
The plot of each model's residuals looks nearly identical. Neither graph has any trend, seasonality, or change of variance. Therefore, both model's residuals resemble white noise.

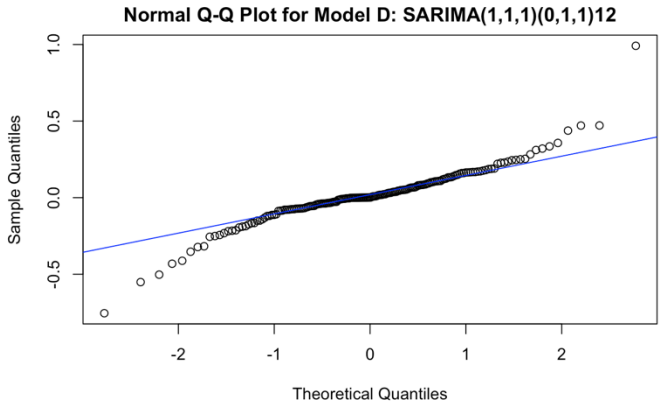
Next, I ran a series of diagnostic tests to see if the models' residuals were normally distributed.






```
##
##  Shapiro-Wilk normality test
##
## data:  res_c
## W = 0.91397, p-value = 8.938e-09
```



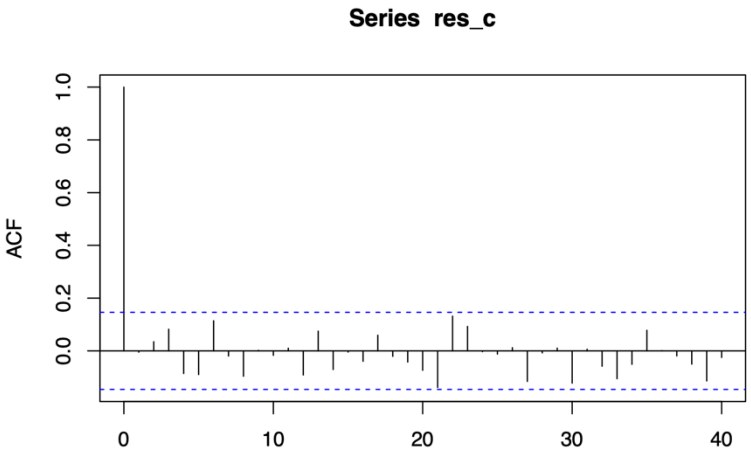


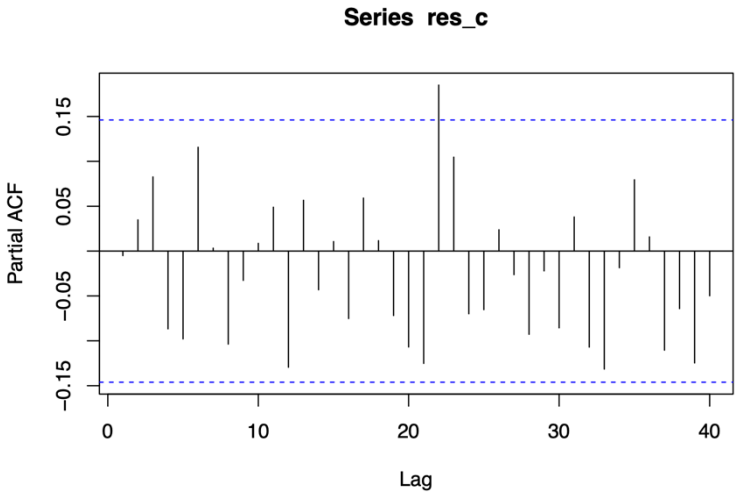

```
##
##  Shapiro-Wilk normality test
##
## data:  res_d
## W = 0.91461, p-value = 9.87e-09
```

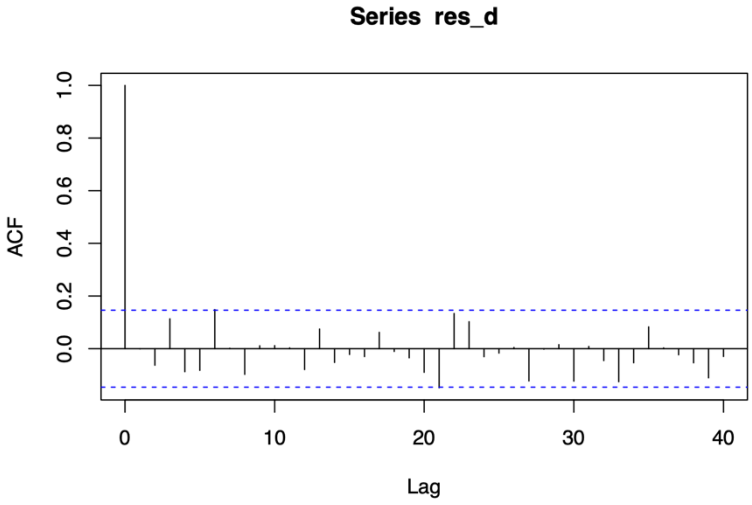
Neither model's residuals passed the Shapiro-Wilk test of normality because both of the models' p-values associated with the test were less than 0.05 (Model C p-value = $8.938e^{-9}$, Model D p-value = $9.87e^{-9}$). Also, neither model's residuals fit the Q-Q line particularly well. If a model's residuals are normally distributed, they should follow the straight blue line in the Q-Q plot. However, both Model C and Model D have large deviations from the line near the ends of the Q-Q plot.

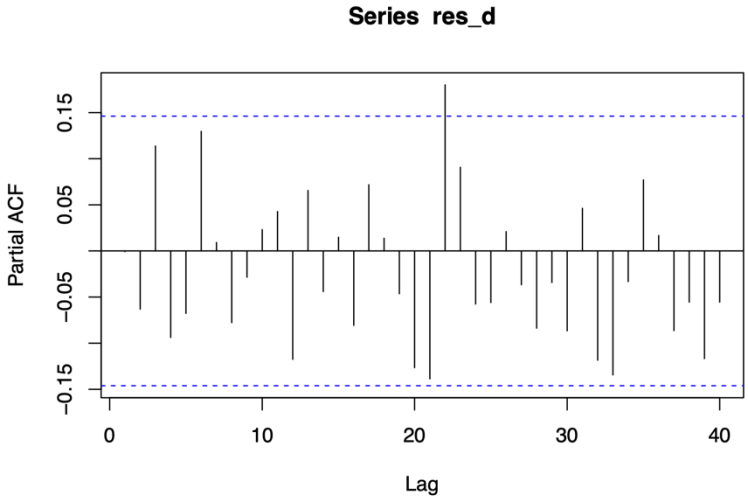
Remember though, I did not expect my models' residuals to pass the Shapiro-Wilk test or be normally distributed because my original data displayed some non-Gaussian behavior (short rises followed by sudden drops) and suggested that a heavy-tailed or non-linear model may be better suited for this data. So, for this reason, and for the sake of this project, I proceeded with diagnostic checking to determine the best possible time series model for this data that can be found using Box-Jenkins methodology.

The next diagnostic test I did was analyzing the ACF and PACF of my models to see if they resembled white noise. For a white noise process, the ACF and PACF should lie within the confidence interval produced by Bartlett's formula at all lags.









The ACFs of Models C and D lie within the confidence interval at all lags. The PACFs for both Model C and Model D lie within the confidence interval at all lags except lag 22. Because the PACF of both models lies outside of the confidence interval at only one lag, I still consider the residuals of both models to be white noise.

I then ran the Portmanteau tests on each model's residuals to see if they were correlated. If the residuals of my models are uncorrelated, then the p-values associated with all the tests should be greater than 0.05.

```
##
##  Box-Pierce test
##
## data:  res_c
## X-squared = 11.801, df = 11, p-value = 0.3788

##
##  Box-Ljung test
##
## data:  res_c
## X-squared = 12.461, df = 11, p-value = 0.33

##
##  Box-Ljung test
##
## data:  (res_c)^2
## X-squared = 6.3164, df = 14, p-value = 0.9579
```

```
##
##  Box-Pierce test
##
## data:  res_d
## X-squared = 14.035, df = 11, p-value = 0.231

##
##  Box-Ljung test
##
## data:  res_d
## X-squared = 14.733, df = 11, p-value = 0.1951

##
##  Box-Ljung test
##
## data:  (res_d)^2
## X-squared = 6.6371, df = 14, p-value = 0.9478
```

The Box-Pierce and Box-Ljung tests test residuals for linear dependence. Since Model C and Model D produce p-values larger than 0.05 for both these tests, I can conclude that both model's residuals are not linearly dependent. The Box-Ljung test using squared residuals, also known as the Mcleod-Li test, tests residuals for non-linear dependence. Because the p-values associated with this test for Model C (0.9579) and Model D (0.9478) are far greater than 0.05 I can conclude that the residuals of both models are not nonlinearly dependent. Further, because both models pass all three of the Portmanteau tests, I can conclude that both model's residuals are uncorrelated.

The final diagnostic check I ran on Model C and Model D was applying Yule-Walker estimation to each model's residuals. This is another test to see if the residuals resemble a white noise process. If they do, the Yule-Walker estimate should fit the models' residuals to AR(0), which is also known as white noise.

```
##
## Call:
## ar(x = res_c, aic = TRUE, order.max = NULL, method = c("yule-walker"))
##
##
## Order selected 0  sigma^2 estimated as  0.03362
```

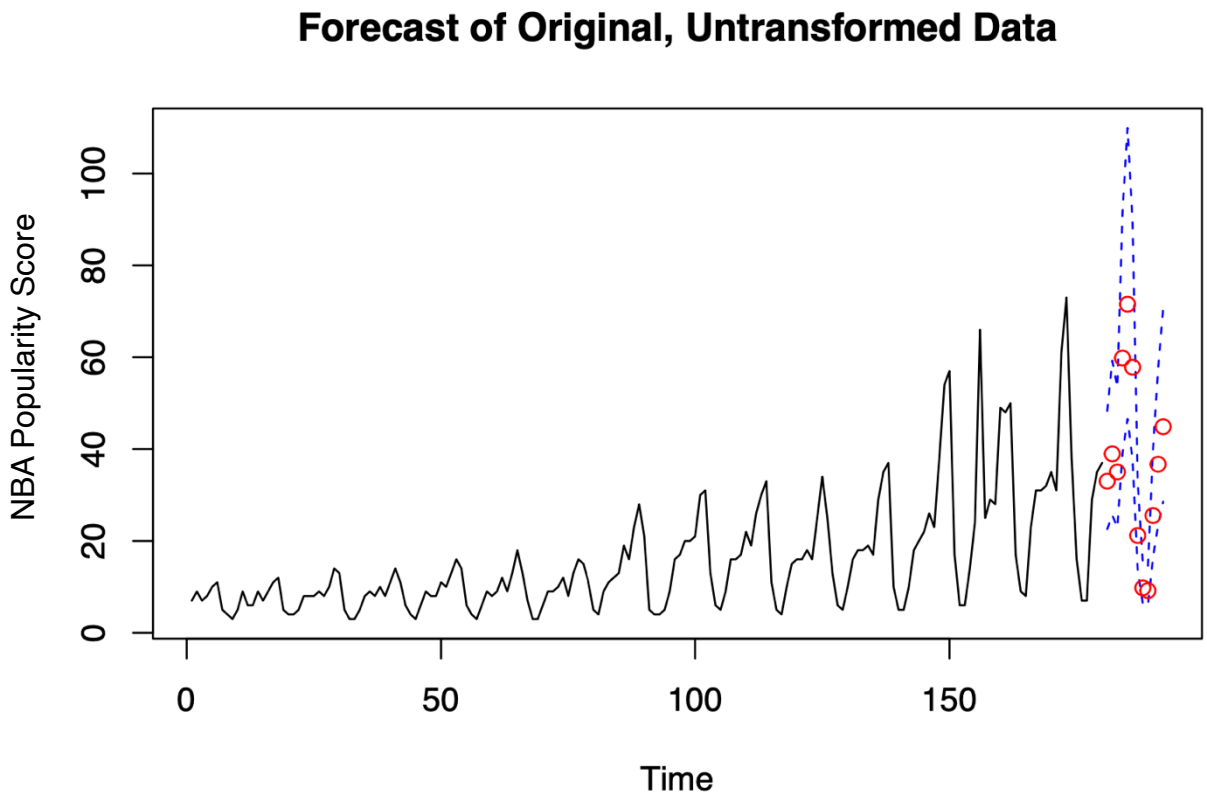
```
##
## Call:
## ar(x = res_d, aic = TRUE, order.max = NULL, method = c("yule-walker"))
##
##
## Order selected 0  sigma^2 estimated as  0.03357
```

This is exactly what happens in the case of both models. Thus, I can further confirm that the residuals resemble a white noise process.

Now, to sum everything up, based on my analysis of AICc and diagnostic checking, I deemed Model C to be the best model. This is because Model C has the lowest AICc of any model I tested and passed all the same diagnostic checks as Model D, the model with the second lowest AICc. Thus, given my dataset and analysis of residuals, I determined that Model C is satisfactory. Further, Model C is also the same as one of the models suggested by the ACF and PACF of my log transformed data differenced at lag 12 once and lag 1 once. For these reasons, I will be using Model C to make my forecasts.

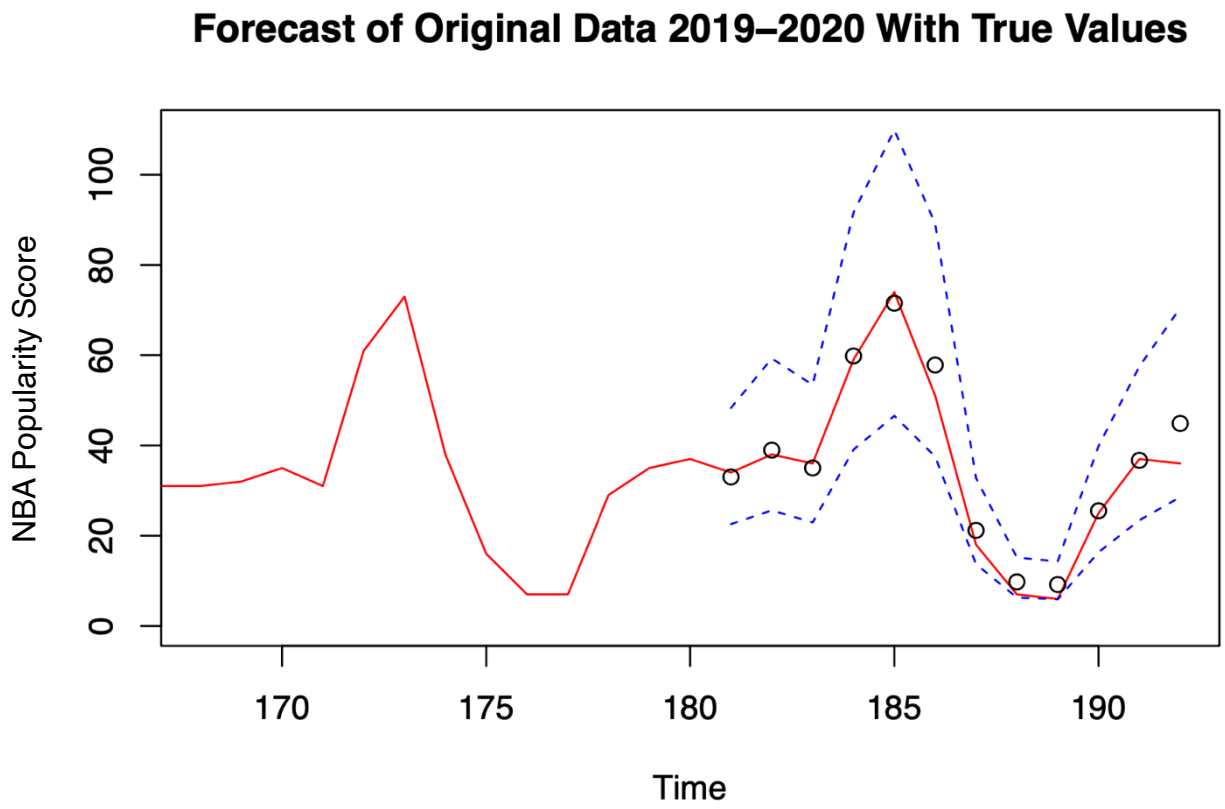
Section V: Forecasting

Going back to the goals of this project, one of them was to accurately forecast the NBA's popularity score in each month of 2020. Now that I have determined the Model C to be the best model, I can use it to accomplish this goal.



The above graph shows a plot of my training set (black lines) and my forecasts for the NBA's popularity score in each month of 2020 (red dots) using Model C. The blue lines on the graph represent the prediction interval of my forecasts. From this graph, it seems that my forecasts are reasonable follow the historical pattern of the data.

In order to get a better idea of just how accurate my forecasts are, though, I zoomed into the last two years of this graph (2019 and 2020) and plotted the true values of the NBA's popularity score in 2020 along with my predictions. This resulted in the following graph:



In this graph, the red plot represents the true values of the NBA's popularity score in each month of 2019 and 2020. The black dots on the graph represent my forecasts, and the blue lines once again represent the prediction interval of my forecasts.

Based on this graph, I would say that the model I chose (Model C) is satisfactory. One reason for this is that the true values of the data in 2020 all fall within the prediction interval. Another reason why I consider my model to be acceptable is that my predictions (black dots) follow the plot of the true values (red line) extremely well. The one pretty significant difference between my predictions and the true values of the data comes in December 2020. My forecast overestimates the NBA's popularity score this month. This can be explained by the fact that in a normal, non-COVID affected year, the NBA would be in the middle of its season in December. However, in 2020, the season had already been over by December.

Conclusion

In conclusion, I achieved all the goals of my project. Using Box-Jenkins methodology I was able to create a time series model that could accurately predict the NBA's Popularity Score in each month of 2020. This model ended up being SARIMA$(0,1,2)(0,1,1)_{12}$ with the equation:

$$\text{Model C Equation: } (1-B)(1-B^{12})X_t = (1-0.5366B-0.3233B^2)(1-0.6828B^{12})Z_t$$

where X_t represents my log-transformed dataset. This model passed all the diagnostic tests I expected it to, given the non-Gaussian nature of my data.

I received help and advice on this project from Thiha Aung, Lihao Xiao, and Raya Feldman.

References

Google Trends (Data Source):

https://trends.google.com/trends/explore?date=all&geo=US&q=%2Fm%2F05jvx&hl=en

PSTAT 174 Lecture 11

PSTAT 174 Lecture 12

PSTAT 174 Lecture 15

PSTAT 174 Lab 7

Appendix

Note: Beginning of analysis of model used for forecasting (Model C) begins on page 69 (page labeled 45 in RMarkdown page numbering) in the section titled “Testing Models After Data Differenced at Lag 12 and Lag 1”

PSTAT 174 Edited Project

Ethan Choi

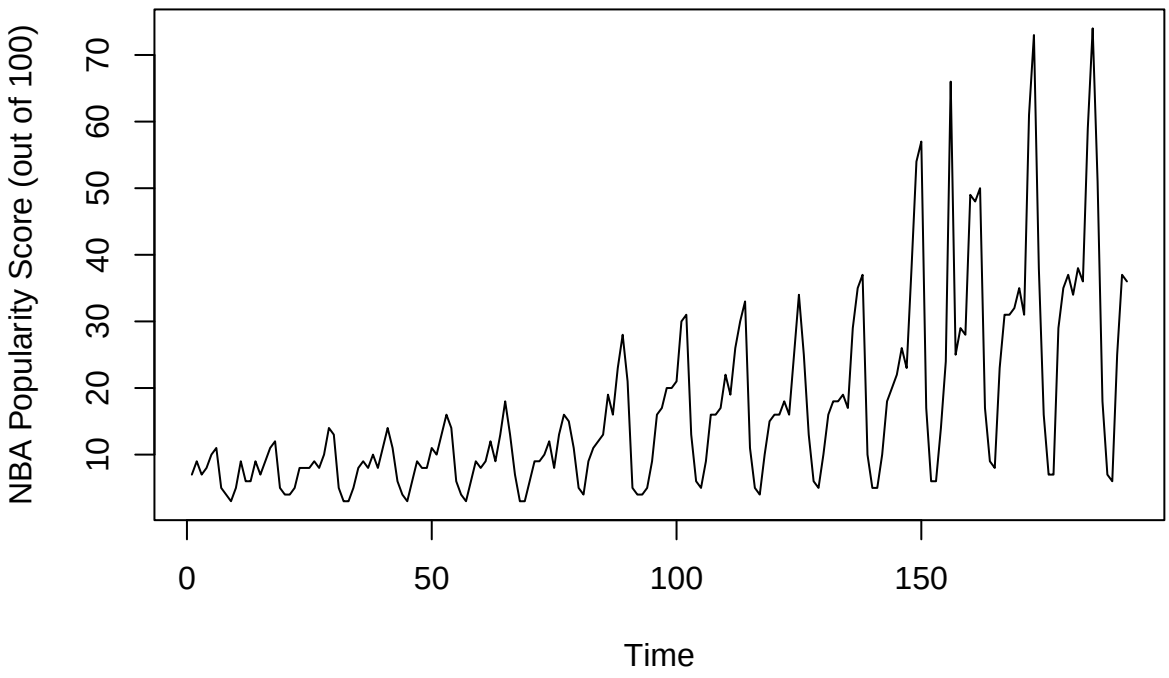
2023-05-31

Exploratory Data Analysis

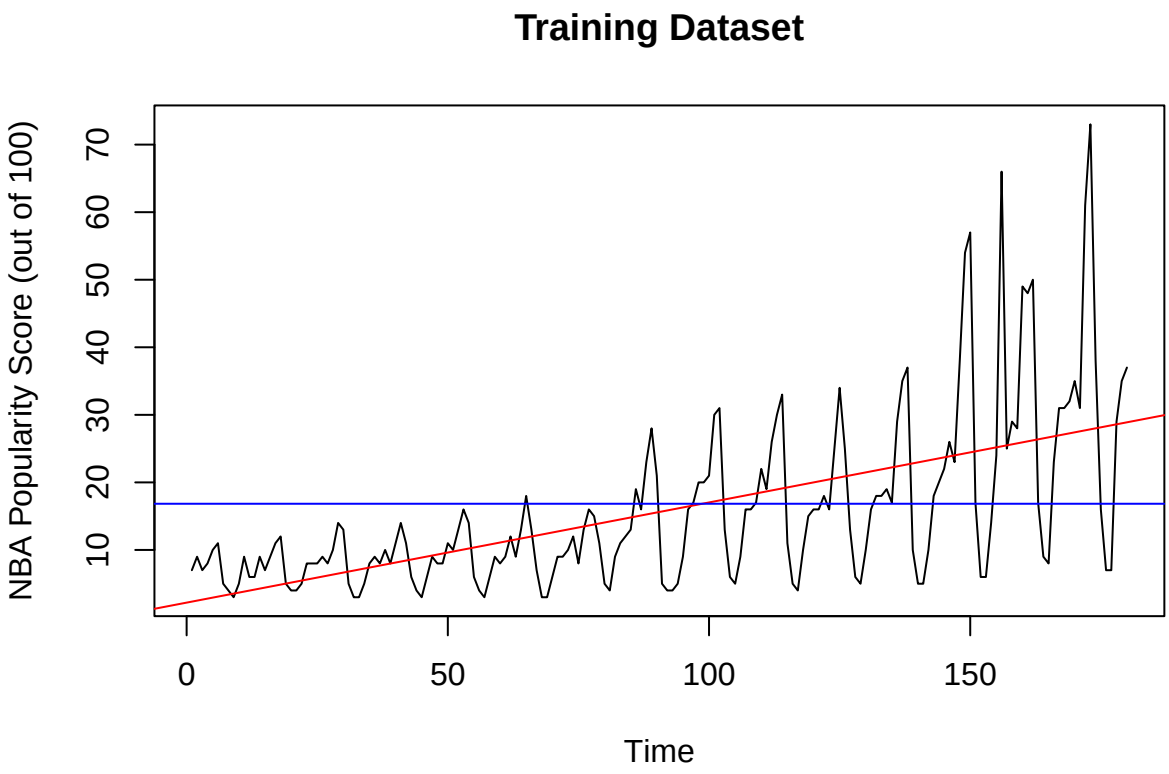
```
# Reading in data
nba.csv <- read.table("nba.csv", sep = ",", header = F, skip = 3)
head(nba.csv)
```

```
##        V1 V2
## 1 2004-01  7
## 2 2004-02  9
## 3 2004-03  7
## 4 2004-04  8
## 5 2004-05 10
## 6 2004-06 11
```

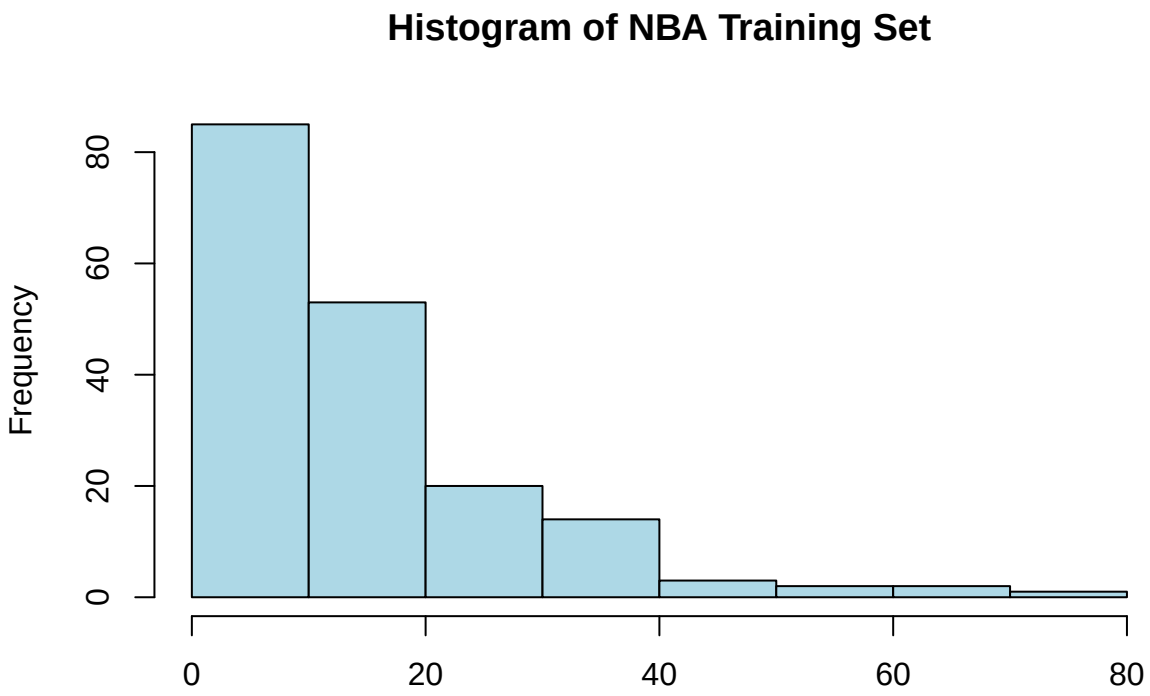
```
# Converting to a time series object
nba <- ts(nba.csv[,2], start = c(2004, 1), frequency = 12)
nba <- ts(nba[1:192])
# Plotting times series
plot.ts(nba, ylab = "NBA Popularity Score (out of 100)")
```



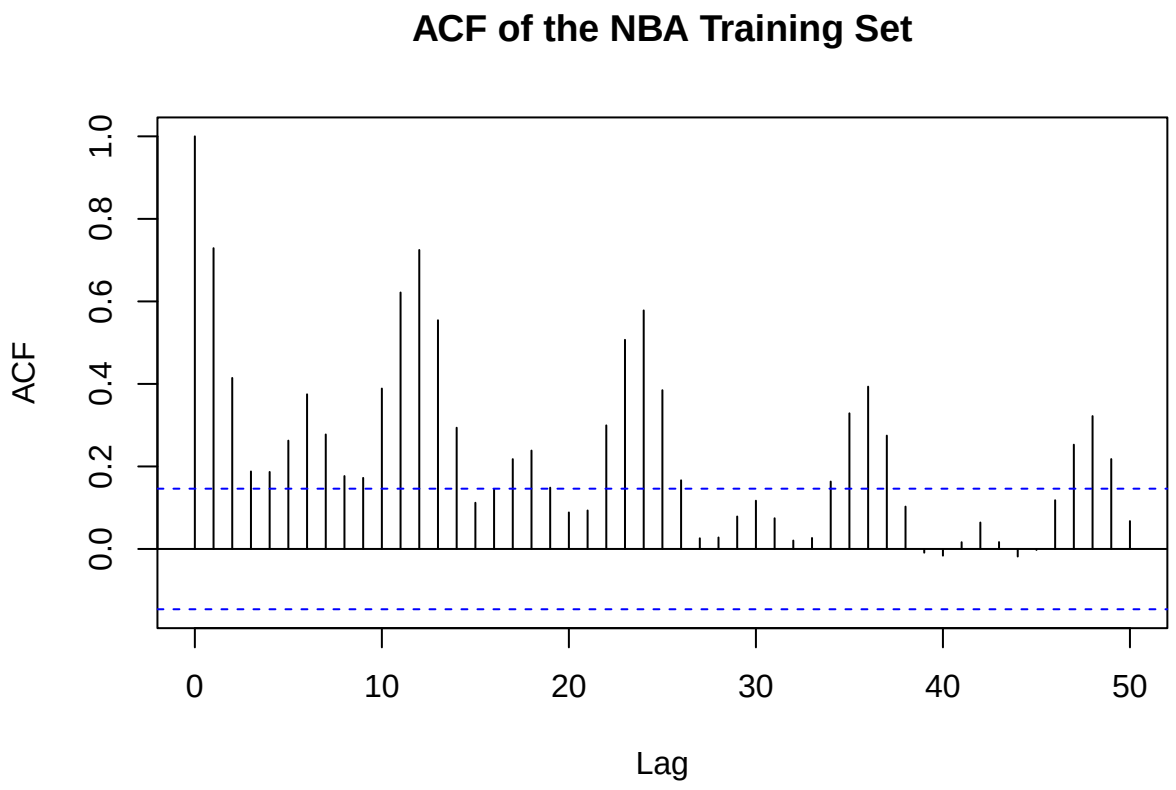

```
# Creating training and test datasets
nba_train <- nba[c(1:180)] # 2004 to 2019
nba_test <- nba[c(181:192)] # 2020

# Plotting training set
plot.ts(nba_train, ylab = "NBA Popularity Score (out of 100)", main = "Training Dataset")
fit1 <- lm(nba_train ~ as.numeric(1:length(nba_train))); abline(fit1, col="red")
abline(h=mean(nba), col="blue")
```




```
# This data is highly non-stationary, has linear trend, seasonality, non-constant variance

# Confirming non-stationarity of original data (training set)
hist(nba_train, col="light blue", xlab="", main="Histogram of NBA Training Set")
```




```
# histogram is very skewed to the right
acf(nba_train,lag.max=50, main="ACF of the NBA Training Set")
```



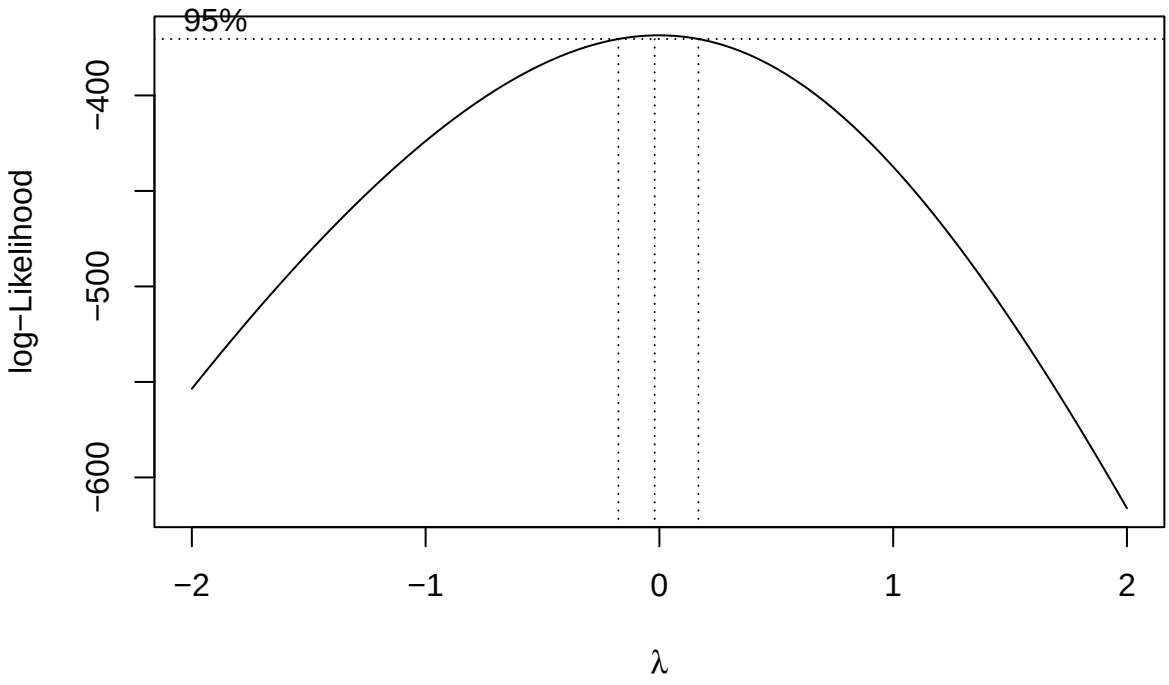

```
# the acf remains large and periodic
```

At first I did my analysis from January 2004 to May 2023. However, the data from 2020 and beyond greatly skewed my plots of data and predictions. Therefore, I decided to only use data from January 2004 to December 2019.

Based on the plot of this data, I can tell this data is highly non-stationary, has linear trend, seasonality, non-constant variance. Therefore, I may have to perform differencing and transformations.

Transforming and Differencing Time Series

```
# Seeing if Box-Cox Transformation is necessary
# (because our data is skewed means non-constant variance)
library(MASS)
bcTransform <- boxcox(nba_train~ as.numeric(1:length(nba_train)))
```




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

```
## [1] -0.02020202
```

```
# because our 95% confidence interval for lambda does not contain 1,
# we follow through with the Box-Cox transformation to stabilize variance
# perhaps choose lambda = 0 because it is in the confidence interval.
# Then we can just use log() as our transformation.
# because our confidence interval for lambda contains zero, we choose the log transformation.

# Comparing Box-Cox vs. Log transformation vs Original Data

# Box-Cox Transformation
nba_train_bc <- (1/lambda)*(nba_train^lambda-1)

# Log Transformation
nba_train_log <- log(nba_train)

# Comparing Variances of all three datasets
# (untransformed training set, box-cox transformed training set, log transformed training set)
var(nba_train)
```

```
## [1] 161.7894
```

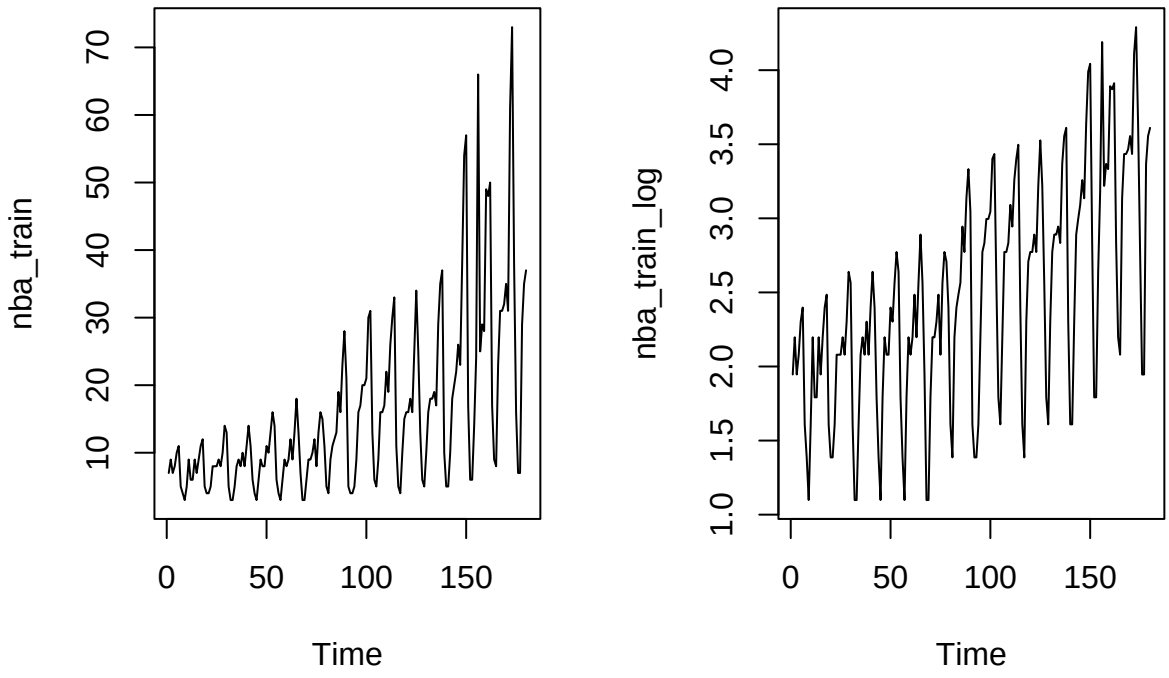
```r
var(nba_train_log)
```

```
## [1] 0.5363114
```

```r
var(nba_train_bc)
```

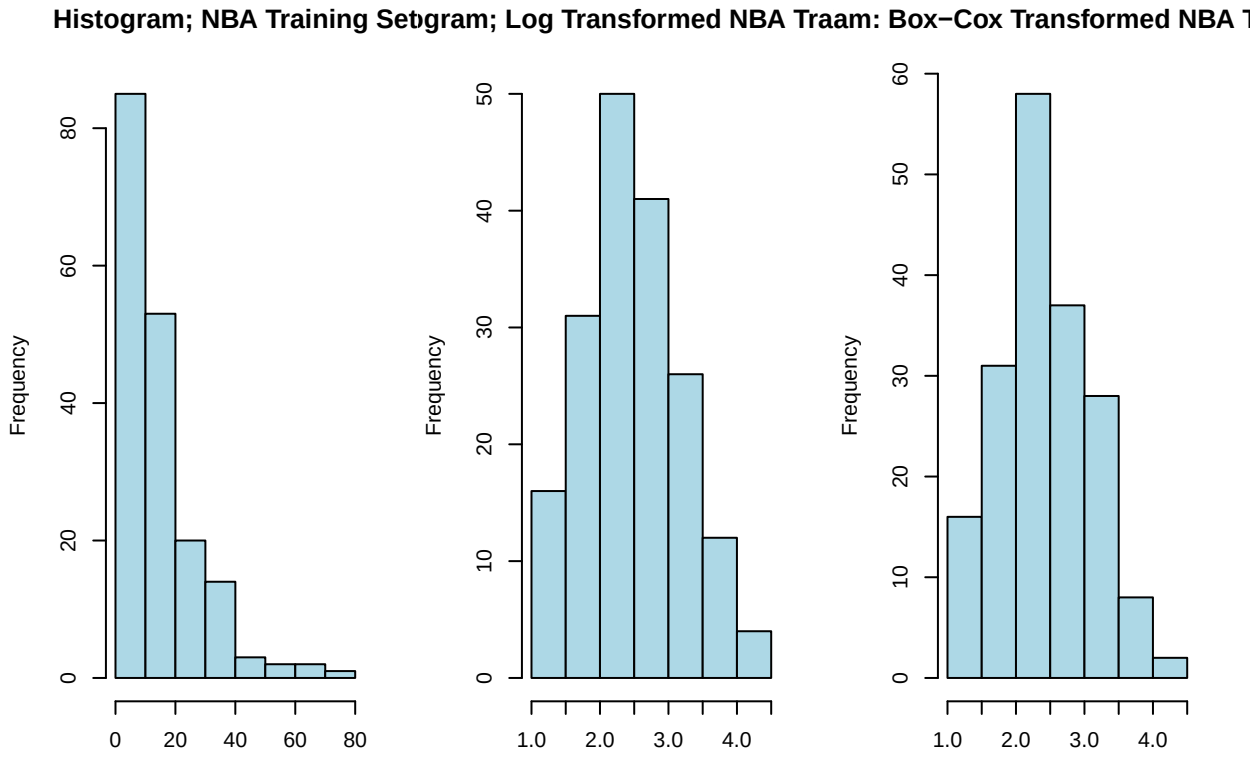
```
## [1] 0.4839666
```

```r
# Comparing Plots of Untransformed Data vs. Log Transformation
op <- par(mfrow = c(1,2))
plot.ts(nba_train)
plot.ts(nba_train_log)
```

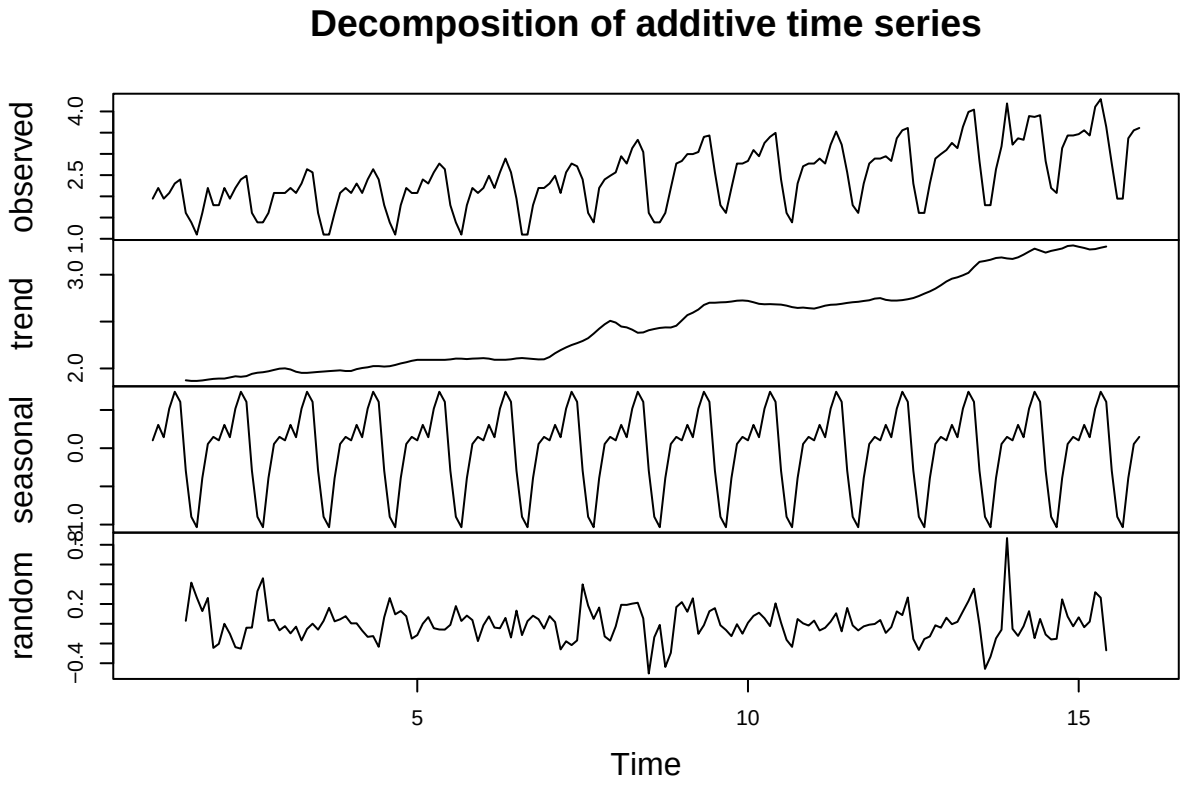



```r
# variance is much more stable after transformations compared to original data

# Comparing histograms of all 3 datasets

# Histogram of Untransformed Training Set
op <- par(mfrow = c(1,3))
hist_train <- hist(nba_train, col="light blue", xlab="", main="Histogram; NBA Training Set")

# Histogram of Log Transformed Training Set
hist_log <- hist(nba_train_log, col="light blue", xlab="",
```

```
                main="Histogram; Log Transformed NBA Training Set")

# Histogram of Box-Cox Transformed Training Set
hist(nba_train_bc, col="light blue", xlab="",
     main="Histogram: Box-Cox Transformed NBA Training Set")
# ultimately choose log transformation because
# it is simpler and there is not much difference between the box-cox and log transformations.
# log transformation also gives most normal looking and symmetric histogram

# Decomopostion of Log Transformed Training Set
library(ggplot2)
```




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




```
# there is linear trend and seasonality that we have to get rid of

# Differencing at lag 12 to remove seasonality
var(nba_train_log)
```

```
## [1] 0.5363114
```

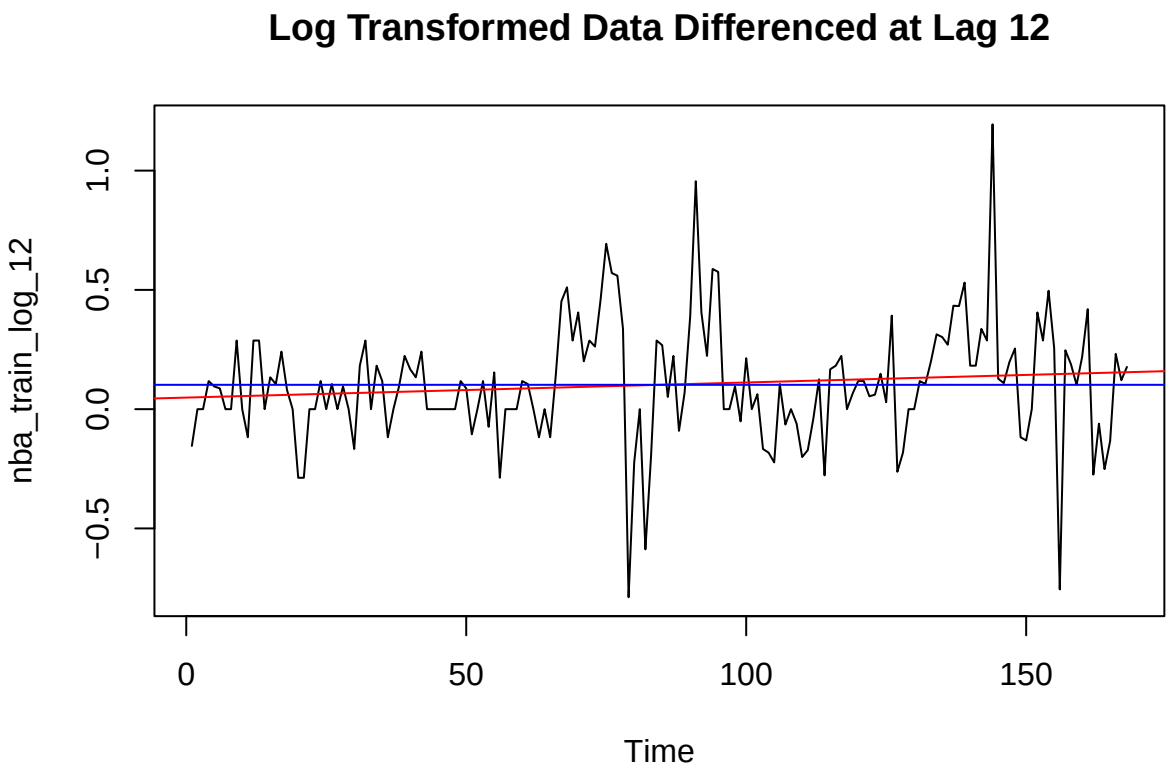
```
nba_train_log_12 <- diff(nba_train_log, lag=12)
var(nba_train_log_12)
```

```
## [1] 0.06216348
```

```
# variance is lower after differencing at lag 12 once
par(mfrow=c(1, 1))
plot.ts(nba_train_log_12, main="Log Transformed Data Differenced at Lag 12")
fit <- lm(nba_train_log_12 ~ as.numeric(1:length(nba_train_log_12))); abline(fit, col="red")
mean(nba_train_log_12)
```

```
## [1] 0.1019656
```

```
abline(h=mean(nba_train_log_12), col="blue")
```



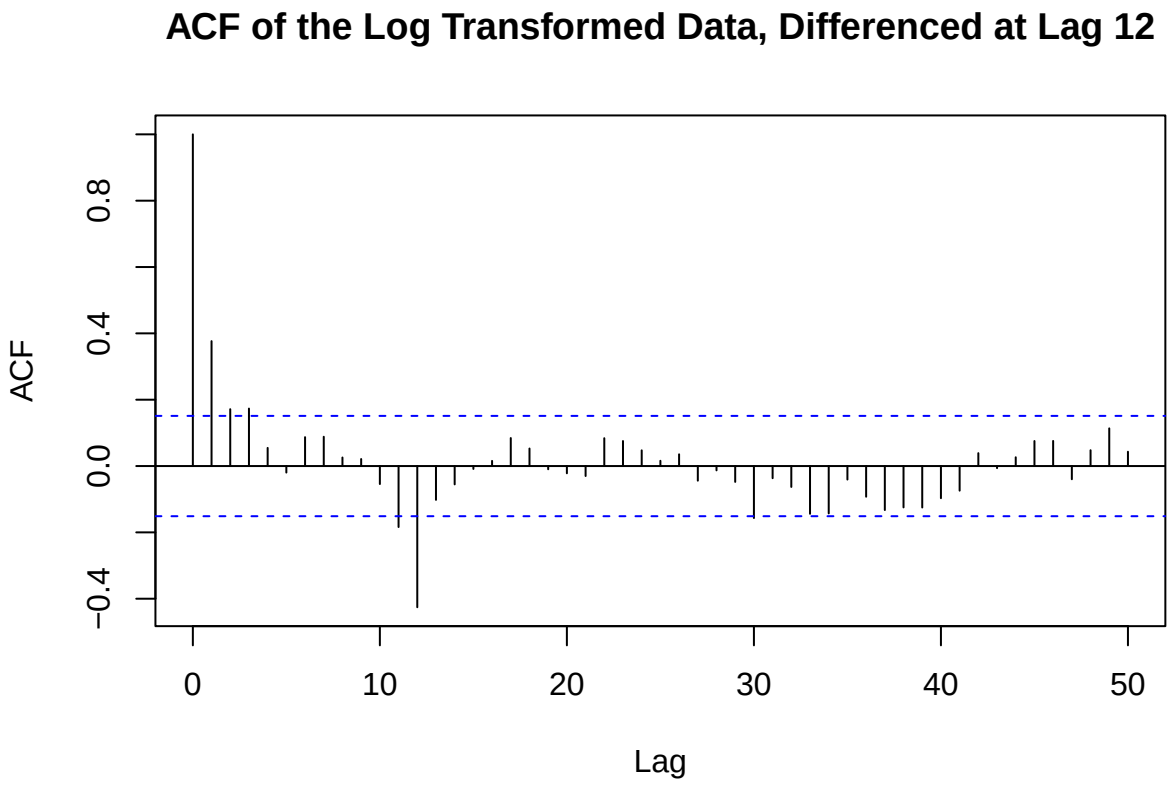

```
# there is still slight linear trend, so try differencing at lag 1

# Differencing at lag 1 to remove trend
nba_train_log_12_1 <- diff(nba_train_log_12, lag=1)
var(nba_train_log_12)
```

```
## [1] 0.06216348
```

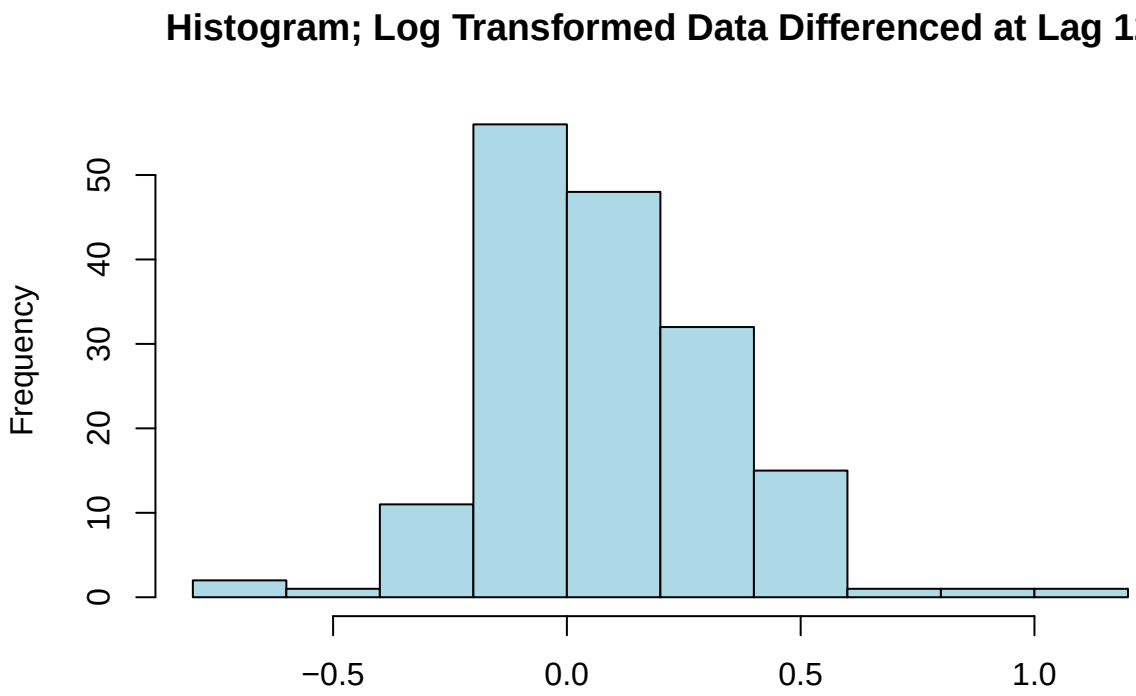
```
var(nba_train_log_12_1)
```

```
## [1] 0.07753534
```

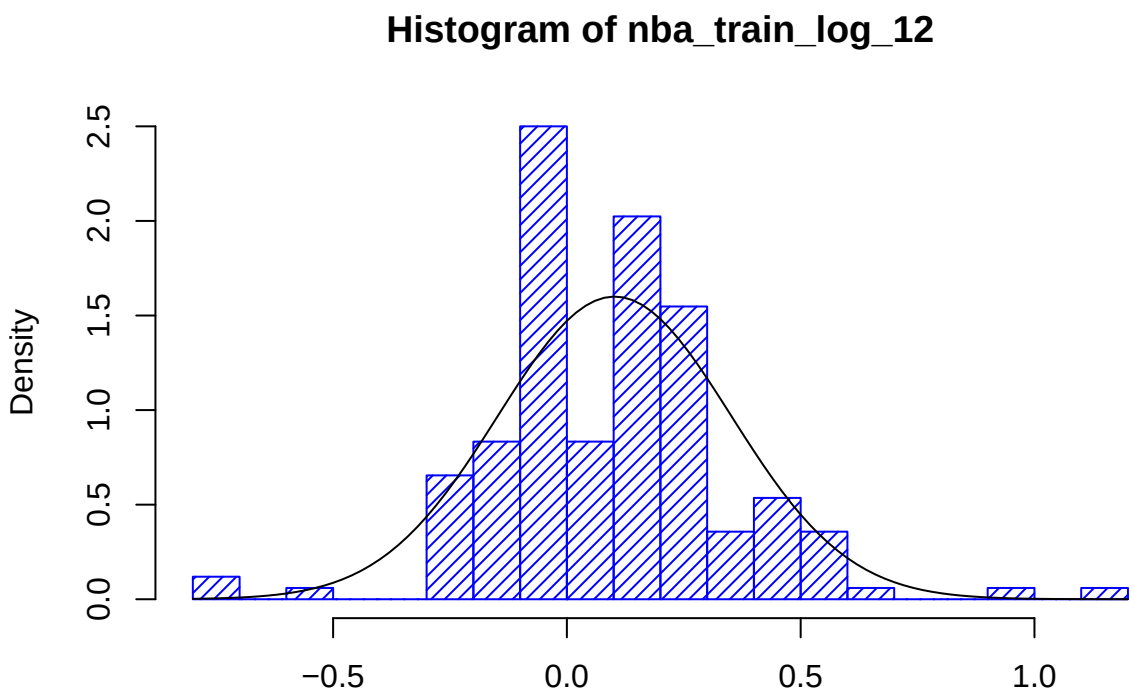
```
# variance is higher so don't follow through with this differencing
# but only slightly so maybe difference if things don't work out.
# END UP USING DATA DIFFERENCED AT LAG 12 AND LAG 1

# Data looks stationary, check ACF to make sure
acf(nba_train_log_12, lag.max=50, main="ACF of the Log Transformed Data, Differenced at Lag 12")
```



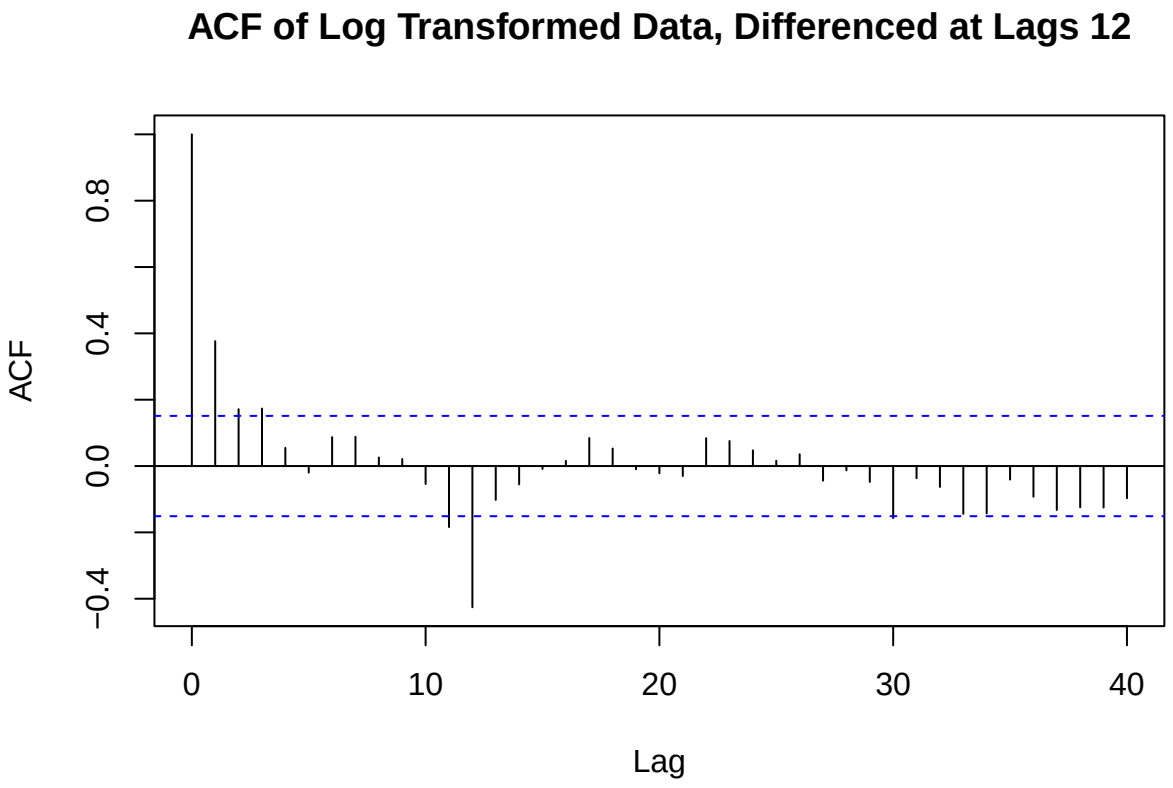

```
# ACF checks out/looks stationary

# Checking histograms to see if data is symmetric and Gaussian
hist(nba_train_log_12, col="light blue", xlab="",
     main="Histogram; Log Transformed Data Differenced at Lag 12")
```



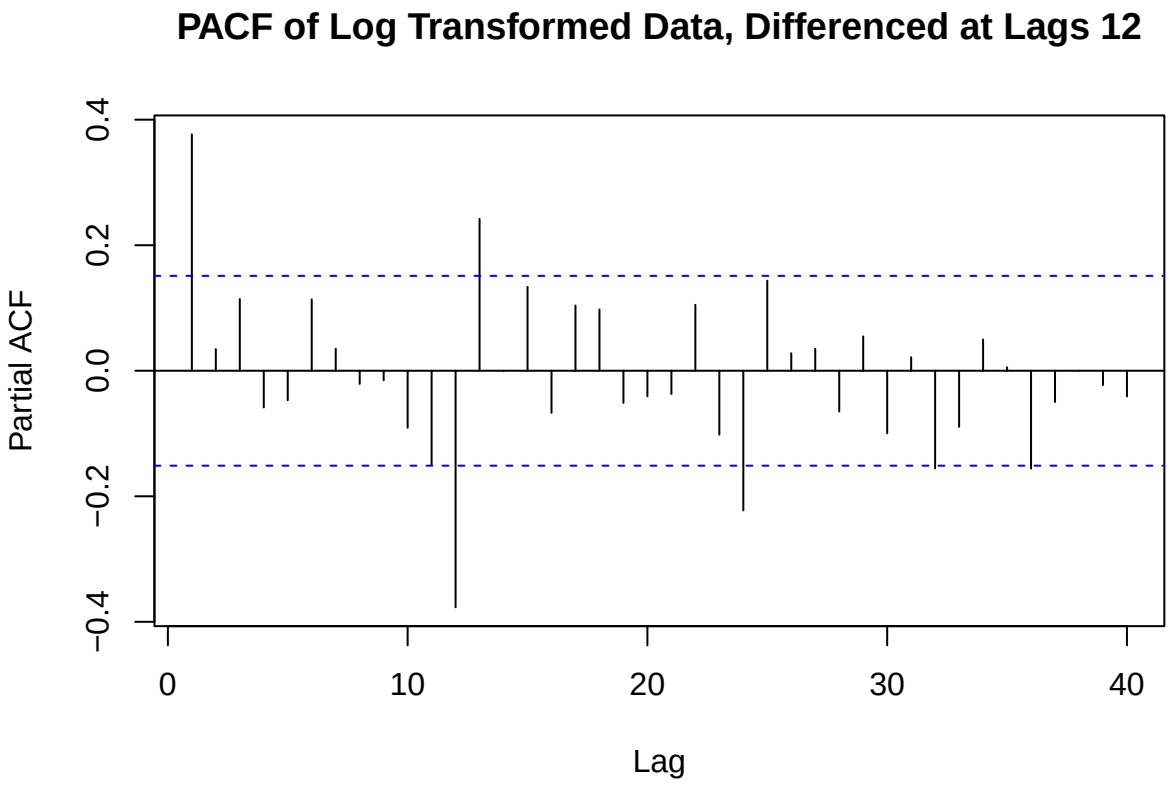

```
hist(nba_train_log_12, density=20,breaks=20, col="blue", xlab="", prob=TRUE)
m<-mean(nba_train_log_12)
std<- sqrt(var(nba_train_log_12))
curve( dnorm(x,m,std), add=TRUE )
```




```
# Although this histogram does not follow the normal curve exactly
# (heavy tailed outliers and slight dispersion close to the center)
# it is ok because we do not have an extremely large amount of observations
# (Reference Lecture 11 Slide 18)

# Analysis of ACF & PACF
acf(nba_train_log_12, lag.max=40, main="ACF of Log Transformed Data, Differenced at Lags 12")
```




```
pacf(nba_train_log_12, lag.max=40, main="PACF of Log Transformed Data, Differenced at Lags 12")
```



Because our 95% confidence interval for lambda does not contain 1, we follow through with the Box-Cox transformation to stabilize variance.

Because our confidence interval for lambda contains 0, we choose the log transformation.

Difference at lag 12 to remove seasonality. Tried differencing at lag 1 but increased variance (only slightly though so maybe can use difference if there's unit roots in AR part of model) END UP USING THIS TRANSFORMATION

From our differencing choices, we know **d=0, D=1, s=12**

ACF lies outside of CI at lags 1,2,3,11,12,30 (used to determine q and Q) (disregard 30 due to Bartlett's formula)

Because ACF lies outside CI at lag 12, suspect **Q=1**

When looking for q, look between lags 1 and 12, suspect **q = 1, 2,3** (q=11 is covered by q =1) (most likely q =1)

PACF lies outside of CI at lags 1, 12, 13, 24, 32, 36 (used to determine p and P)

Because PACF lies outside of CI at lags 12 and 24 suspect **P=1**, **2**, maybe **P=3** because PACF lies outside CI at lag 36 (but may disregard because of Bartlett's formula) (most likely P=2)

When looking for p, look between lags 1 and 12, suspect **p=1**

Model Estimation

```
library(qpcR)
# Candidate models:
df <- expand.grid(p=0:1, q=0:3, P=0:3, Q=0:1)
df <- cbind(df, AICc=NA)
# Testing Models and Computing AICcs:
for (i in 1:nrow(df)) {
sarima.obj <- NULL
try(arima.obj <- arima(nba_train_log, order=c(df$p[i], 0, df$q[i]),
seasonal=list(order=c(df$P[i], 1, df$Q[i]), period=12),
method="ML"))
if (!is.null(arima.obj)) { df$AICc[i] <- AICc(arima.obj) }
# print(df[i, ])
}
df[which.min(df$AICc), ]
```

```
##    p q P Q      AICc
## 38 1 2 0 1 -62.80332
```

```
head(df[order(df$AICc),]) # sorting models by lowest AICC
```

```
##    p q P Q      AICc
## 38 1 2 0 1 -62.80332
## 46 1 2 1 1 -62.07411
## 30 1 2 3 0 -61.92047
## 62 1 2 3 1 -60.98490
## 54 1 2 2 1 -60.91552
## 40 1 3 0 1 -60.68741
```

```
# Code from lab 7 page 7
df[order(df$AICc),]
```

```
##    p q P Q        AICc
## 38 1 2 0 1 -62.8033219
## 46 1 2 1 1 -62.0741100
## 30 1 2 3 0 -61.9204668
## 62 1 2 3 1 -60.9848985
## 54 1 2 2 1 -60.9155169
## 40 1 3 0 1 -60.6874053
## 48 1 3 1 1 -60.0101150
## 32 1 3 3 0 -59.8297404
## 64 1 3 3 1 -58.8017510
## 56 1 3 2 1 -58.7867935
## 22 1 2 2 0 -55.8861017
## 24 1 3 2 0 -54.3156691
## 28 1 1 3 0 -52.4387832
## 44 1 1 1 1 -51.9049936
## 36 1 1 0 1 -51.8935803
## 52 1 1 2 1 -51.6337831
## 60 1 1 3 1 -50.5857151
## 20 1 1 2 0 -50.2695917
## 14 1 2 1 0 -45.2247886
```

```
## 16 1 3 1 0 -44.3018098
## 12 1 1 1 0 -39.5405560
## 63 0 3 3 1 -35.9582400
## 34 1 0 0 1 -34.8298989
## 26 1 0 3 0 -34.7931917
## 42 1 0 1 1 -34.2601027
## 18 1 0 2 0 -33.9020908
## 50 1 0 2 1 -33.4849614
## 58 1 0 3 1 -32.7014805
## 10 1 0 1 0 -29.5372377
## 23 0 3 2 0 -28.7410250
## 47 0 3 1 1 -27.7760161
## 15 0 3 1 0 -27.4196401
## 39 0 3 0 1 -27.2032003
## 31 0 3 3 0 -26.6044984
## 55 0 3 2 1 -26.6017866
## 61 0 2 3 1 -23.2382405
## 59 0 1 3 1 -23.2003718
## 51 0 1 2 1 -19.6104256
## 13 0 2 1 0 -15.0346914
## 21 0 2 2 0 -13.3487316
## 45 0 2 1 1 -13.2967035
## 37 0 2 0 1 -13.2126291
## 11 0 1 1 0 -12.7374792
## 29 0 2 3 0 -11.2325226
## 53 0 2 2 1 -11.2324826
## 19 0 1 2 0 -10.6795232
## 43 0 1 1 1 -10.6775805
## 35 0 1 0 1  -9.0922738
## 27 0 1 3 0  -8.8830982
## 6  1 2 0 0  -3.6978108
## 4  1 1 0 0  -3.2798308
## 2  1 0 0 0  -2.1060671
## 8  1 3 0 0  -1.8521060
## 7  0 3 0 0  -0.1522479
## 5  0 2 0 0   5.1613595
## 3  0 1 0 0   6.2935139
## 57 0 0 3 1  10.8512166
## 49 0 0 2 1  12.0332015
## 17 0 0 2 0  29.3742793
## 25 0 0 3 0  29.6648150
## 9  0 0 1 0  30.6165703
## 41 0 0 1 1  30.7878500
## 33 0 0 0 1  32.8020472
## 1  0 0 0 0  37.1841759
```

Choose $SARIMA(1,0,2)(0,1,1)_{12}$ for diagnostic checking because it has the lowest AICc (-62.80332) and the least amount of parameters (4) of any of the six models with the lowest AICcs.

Also, for the sake of completeness, test $SARIMA(1,0,2)(1,1,1)_{12}$ because it has the second lowest AICc (-62.07411) and the second least amount of parameters (4) of any of the six models with the lowest AICcs.

```
# Checking coefficients of SARIMA(1,0,2)(0,1,1)12 (Model A)
model_a <- arima(nba_train_log, order=c(1,0,2), seasonal = list(order = c(0,1,1),
```

```
                                                                period = 12), method = "ML")
model_a
```

```
##
## Call:
## arima(x = nba_train_log, order = c(1, 0, 2), seasonal = list(order = c(0, 1,
##     1), period = 12), method = "ML")
##
## Coefficients:
##          ar1      ma1      ma2     sma1
##       0.9963  -0.5344  -0.3184  -0.6790
## s.e.  0.0058   0.0773   0.0821   0.0616
##
## sigma^2 estimated as 0.03614:  log likelihood = 36.52,  aic = -63.03
```

```
AICc(model_a)
```

```
## [1] -62.80332
```

```
# 95% CIs for Coefficients
confint(model_a)
```

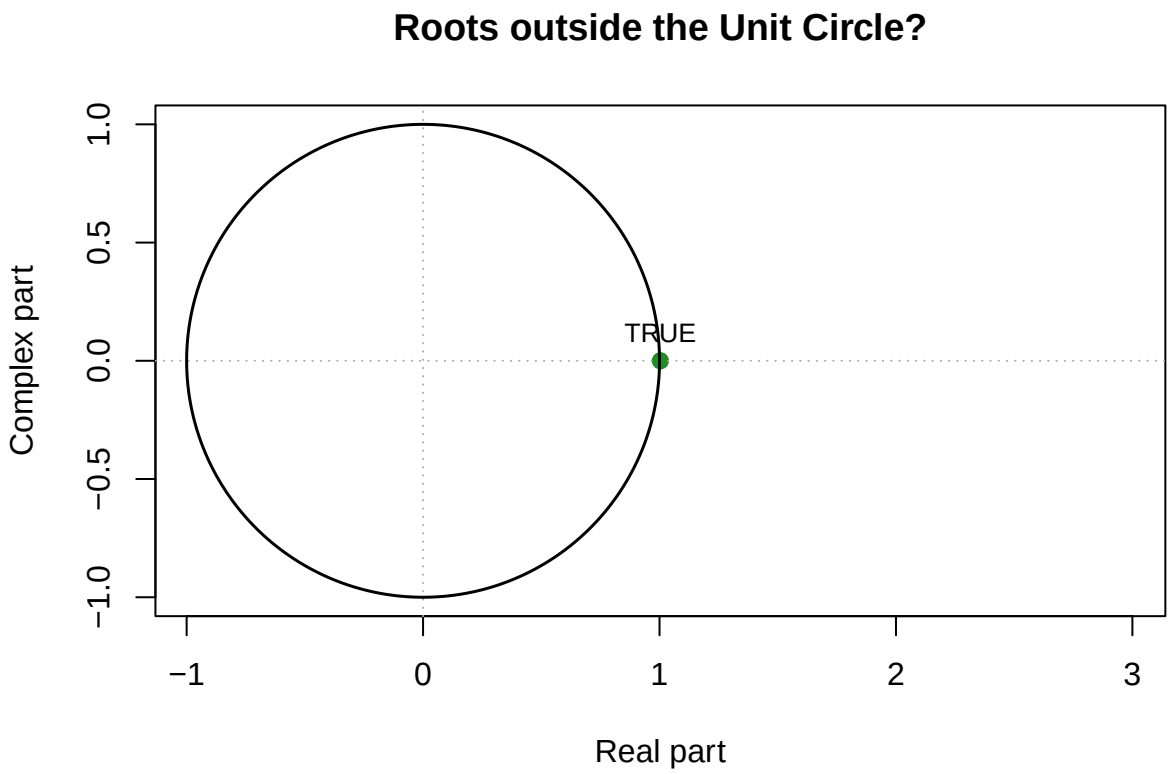
```
##           2.5 %     97.5 %
## ar1   0.9848919  1.0077537
## ma1  -0.6859631 -0.3828595
## ma2  -0.4793466 -0.1575394
## sma1 -0.7997793 -0.5583146
```

```
# none of the confidence intervals contain 0 so no need to test variations of this model.

# Checking stationarity and inveritibility of Model A
library(UnitCircle)
model_a
```

```
##
## Call:
## arima(x = nba_train_log, order = c(1, 0, 2), seasonal = list(order = c(0, 1,
##     1), period = 12), method = "ML")
##
## Coefficients:
##          ar1      ma1      ma2     sma1
##       0.9963  -0.5344  -0.3184  -0.6790
## s.e.  0.0058   0.0773   0.0821   0.0616
##
## sigma^2 estimated as 0.03614:  log likelihood = 36.52,  aic = -63.03
```

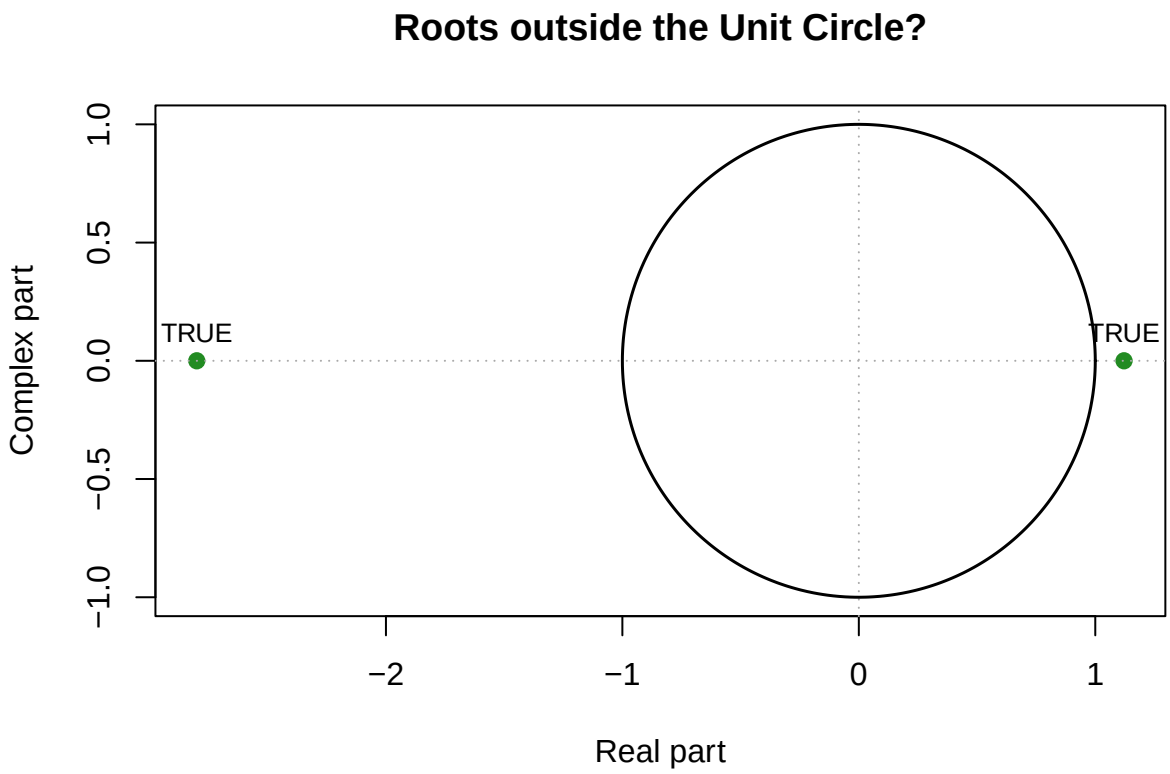
```
# Checking AR part
uc.check(pol_ = c(1, -0.9963), plot_output = TRUE) # PASS BUT SUPER CLOSE (CONSIDER UNIT ROOT)
```

```
##       real complex outside
## 1 1.003714       0    TRUE
## *Results are rounded to 6 digits.
```




```
# Checking MA part
uc.check(pol_ = c(1, -0.5344, -0.3184), plot_output = TRUE) # PASS
```

```
##        real complex outside
## 1  1.121659       0    TRUE
## 2 -2.800051       0    TRUE
## *Results are rounded to 6 digits.
```



Model A Equation: $(1 - 0.9963B)(1 - B^{12})X_t = (1 - 0.5344B - 0.3184B^2)(1 - 0.6790B^{12})Z_t$

```
# Checking coefficients of SARIMA(1,0,2)(1,1,1)12 (Model B)
model_b <- arima(nba_train_log, order=c(1,0,2), seasonal = list(order = c(1,1,1),
                                                                period = 12), method = "ML")
model_b
```

```
##
## Call:
## arima(x = nba_train_log, order = c(1, 0, 2), seasonal = list(order = c(1, 1,
##     1), period = 12), method = "ML")
##
## Coefficients:
##          ar1      ma1      ma2     sar1     sma1
##       0.9950  -0.5268  -0.3119  -0.1310  -0.6095
## s.e.  0.0073   0.0791   0.0836   0.1087   0.0918
##
## sigma^2 estimated as 0.03582:  log likelihood = 37.21,  aic = -62.42
```

```
AICc(model_b)
```

```
## [1] -62.07411
```

```r
# 95% CIs for Coefficients
confint(model_b)
```

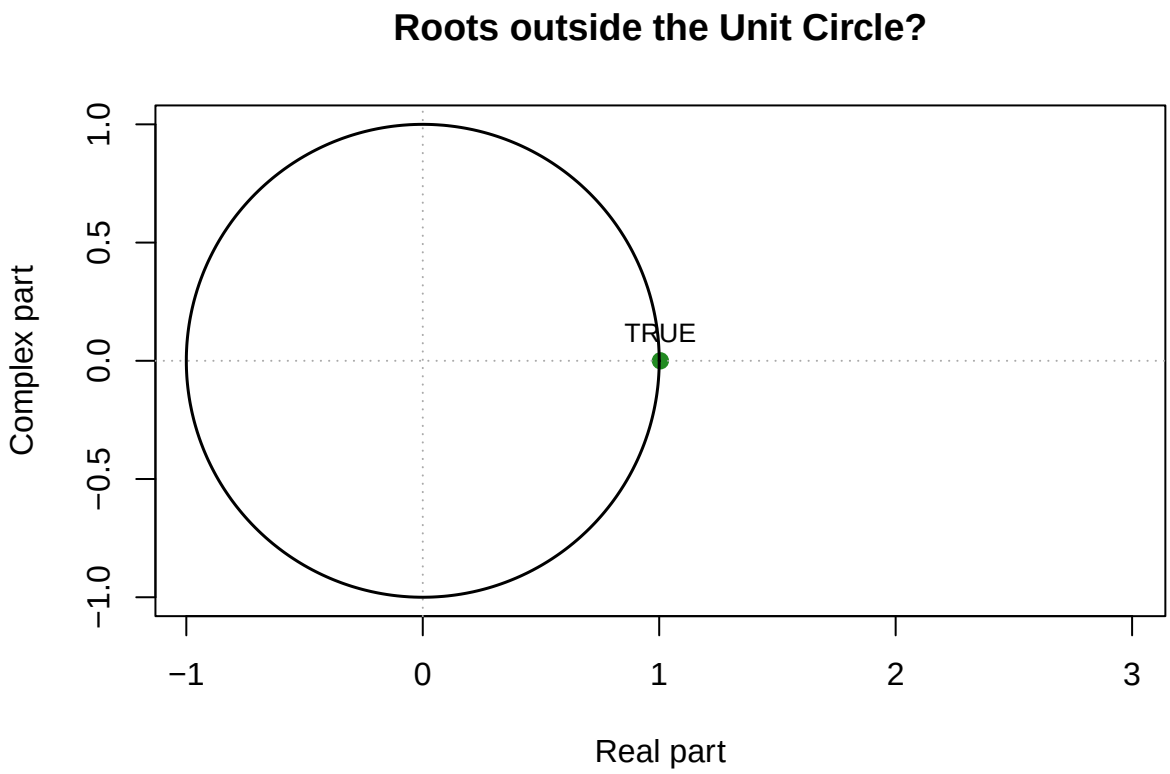
```
##           2.5 %      97.5 %
## ar1   0.9806682  1.00938479
## ma1  -0.6818017 -0.37179690
## ma2  -0.4756758 -0.14805172
## sar1 -0.3440997  0.08213476
## sma1 -0.7893739 -0.42963264
```

```r
# the sar1 coefficient CI contains 0 whcih means P = 0 (this case was tested in Model A)

# Checking stationarity and inveritibility of Model B
library(UnitCircle)
model_b
```

```
##
## Call:
## arima(x = nba_train_log, order = c(1, 0, 2), seasonal = list(order = c(1, 1,
##     1), period = 12), method = "ML")
##
## Coefficients:
##          ar1      ma1      ma2     sar1     sma1
##       0.9950  -0.5268  -0.3119  -0.1310  -0.6095
## s.e.  0.0073   0.0791   0.0836   0.1087   0.0918
##
## sigma^2 estimated as 0.03582:  log likelihood = 37.21,  aic = -62.42
```

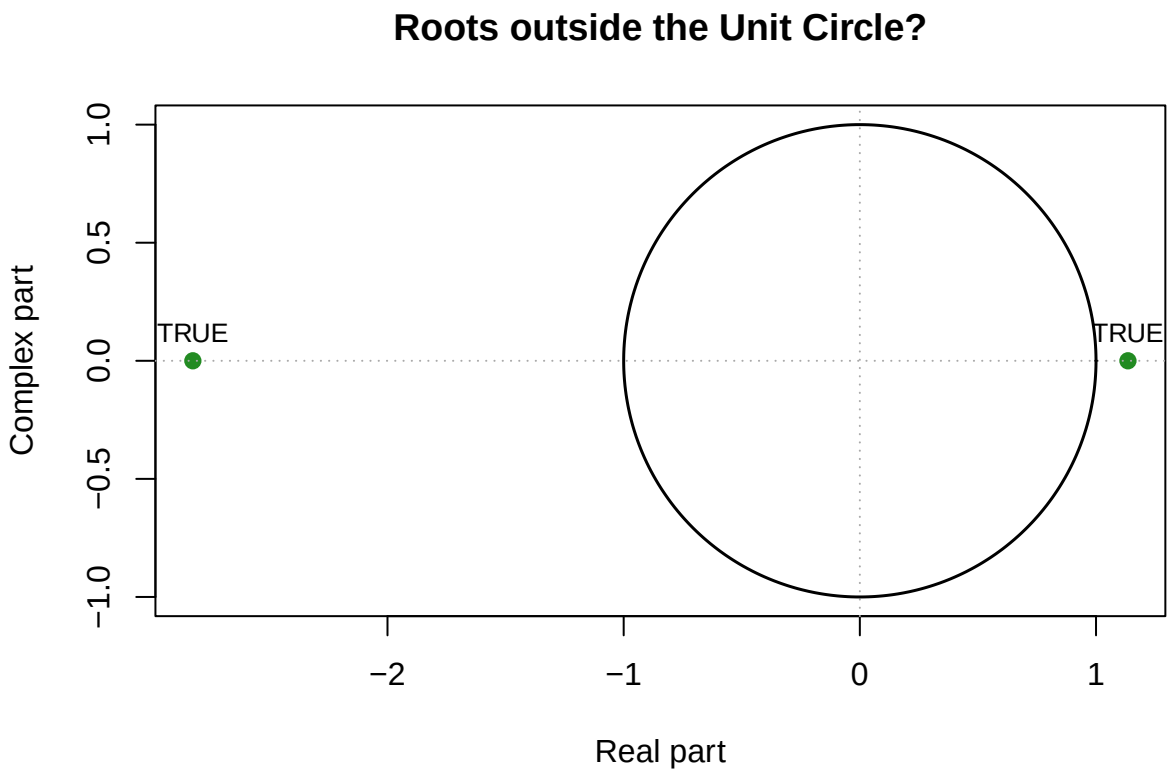
```r
# Checking AR part
uc.check(pol_ = c(1, -0.9950), plot_output = TRUE) # PASS BUT SUPER CLOSE (CONSIDER UNIT ROOT)
```

```
##       real complex outside
## 1 1.005025       0    TRUE
## *Results are rounded to 6 digits.
```




```
# Checking MA part
uc.check(pol_ = c(1, -0.5268, -0.3119), plot_output = TRUE) # PASS
```

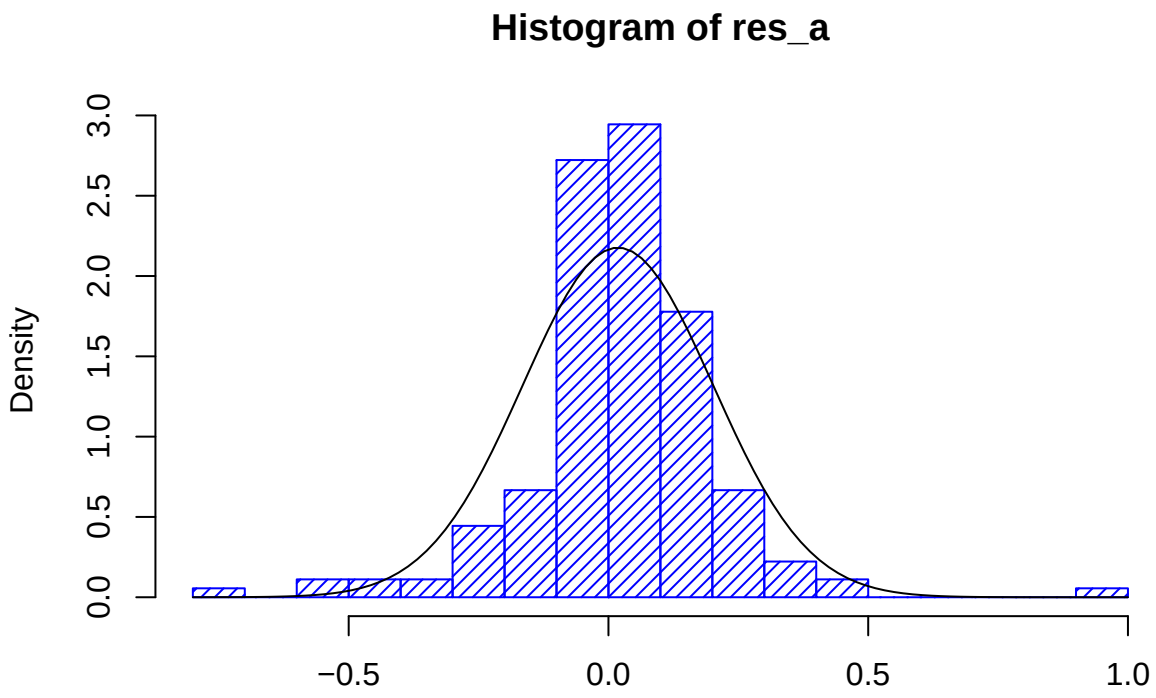
```
##        real complex outside
## 1  1.135230       0    TRUE
## 2 -2.824233       0    TRUE
## *Results are rounded to 6 digits.
```



Model B Equation: $(1-0.995B)(1+0.131B^{12})(1-B^{12})X_t = (1-0.5268B-0.3119B^2)(1-0.6095B^{12})Z_t$

Diagnostic Checking

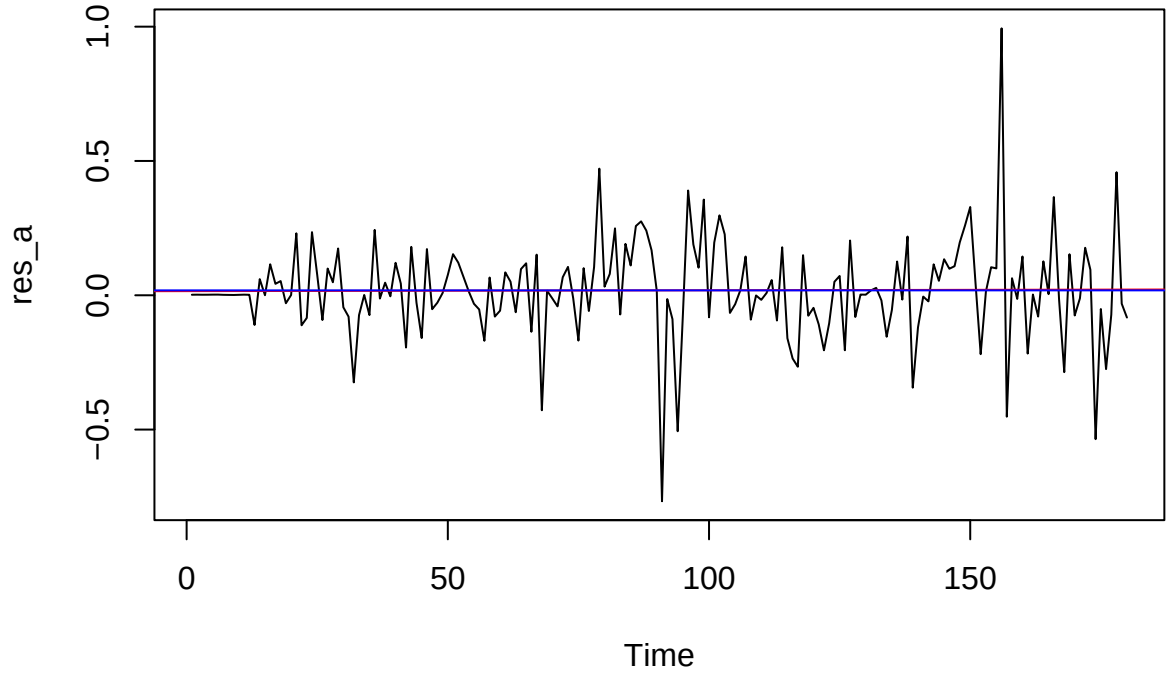
```
# Residuals of SARIMA(1,0,2)(0,1,1)12 (Model A)
res_a <- residuals(model_a)

# Histogram of Residuals
hist(res_a,density=20,breaks=20, col="blue", xlab="", prob=TRUE)
m <- mean(res_a)
std <- sqrt(var(res_a))
curve( dnorm(x,m,std), add=TRUE )
```



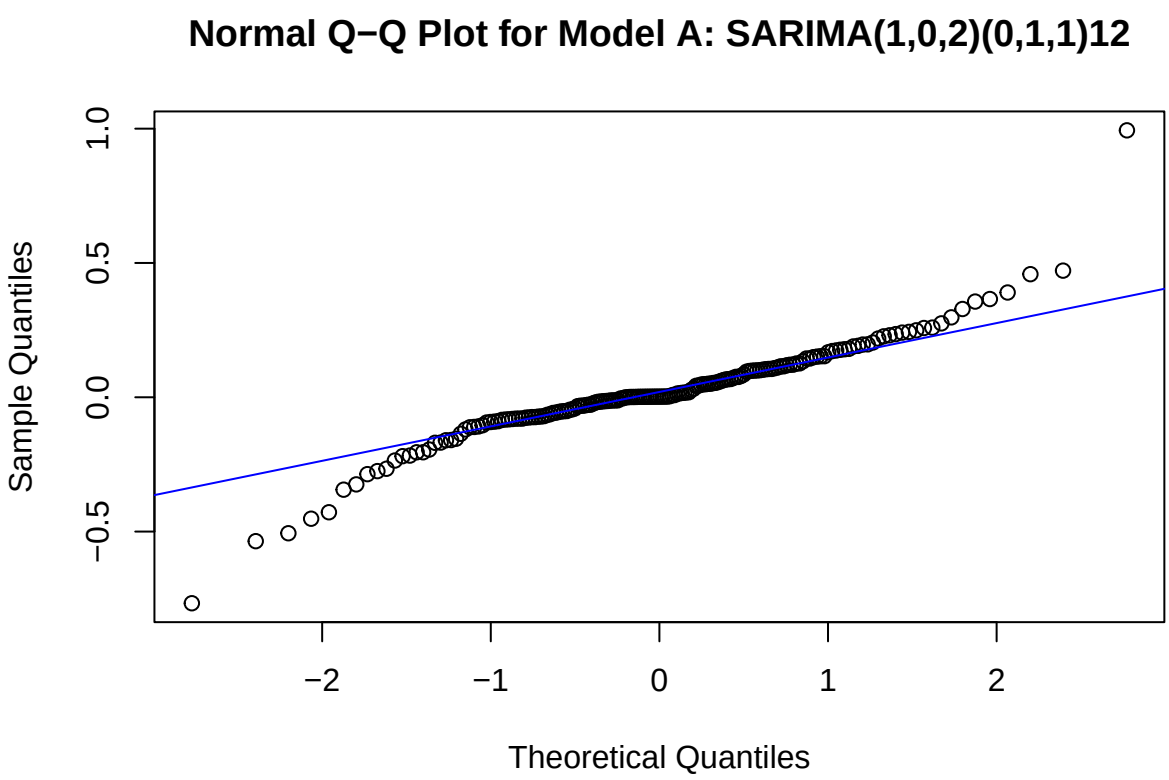

```
# Plot of residuals
mean(res_a) # = 0.01827164 close to 0
```

```
## [1] 0.01827164
```

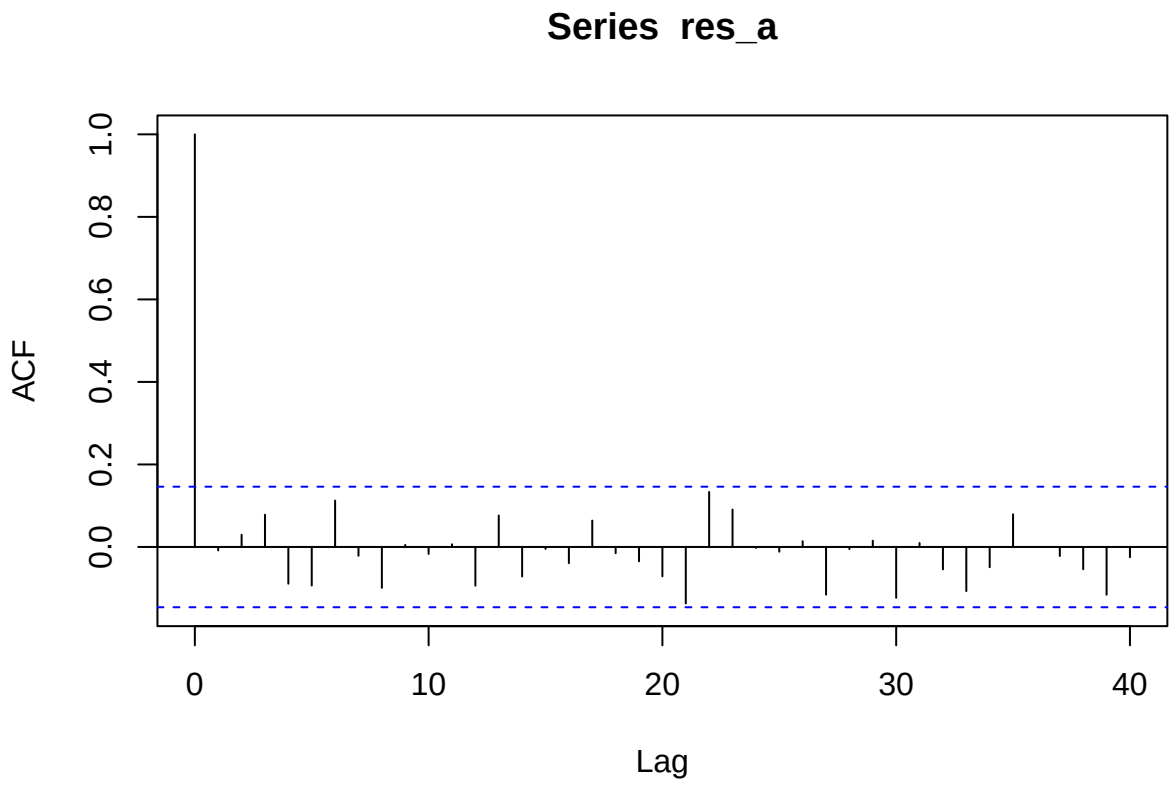
```
plot.ts(res_a)
fitt <- lm(res_a ~ as.numeric(1:length(res_a))); abline(fitt, col="red")
abline(h=mean(res_a), col="blue")
```



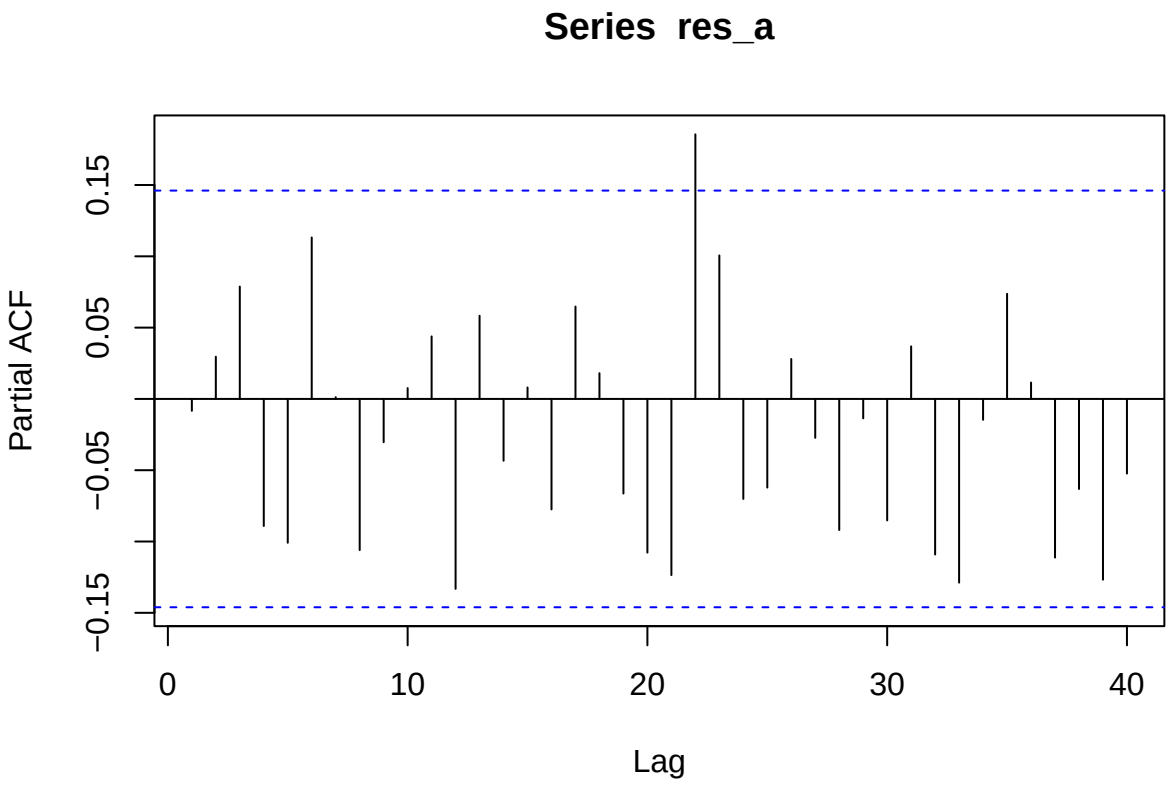

```
# no trend, seasonality, or change of variance

# Q-Q Plot
qqnorm(res_a,main= "Normal Q-Q Plot for Model A: SARIMA(1,0,2)(0,1,1)12")
qqline(res_a,col="blue")
```




```
# fits OK, some deviation at the ends

# ACF and PACF of residuals
acf(res_a, lag.max=40) # within CI at all lags
```




```
pacf(res_a, lag.max=40) # within CI at all lags except lag 22 which is OK
```




```
# LOOK FOR P-VALUES GREATER THAN 0.05
# lag = sqrt(n)
# fitdf = number of coefficients estimated
# Shapiro test
shapiro.test(res_a) # p-value = 8.045e^-9 FAIL
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res_a
## W = 0.91328, p-value = 8.045e-09
```

```
# Box-Pierce test
Box.test(res_a, lag = 14, type = c("Box-Pierce"), fitdf = 4) # p-value = 0.2838 PASS
```

```
##
##  Box-Pierce test
##
## data:  res_a
## X-squared = 12.018, df = 10, p-value = 0.2838
```

```
# Ljung-Box Test
Box.test(res_a, lag = 14, type = c("Ljung-Box"), fitdf = 4) # p-value = 0.2412 PASS
```

```
##
##  Box-Ljung test
##
## data:  res_a
## X-squared = 12.696, df = 10, p-value = 0.2412
```

```
# Mcleod-Li Test
Box.test((res_a)^2, lag = 14, type = c("Ljung-Box"), fitdf = 0) # p-value = 0.9472 PASS
```

```
##
##  Box-Ljung test
##
## data:  (res_a)^2
## X-squared = 6.6552, df = 14, p-value = 0.9472
```

```
# Yule-Walker Check
ar(res_a, aic = TRUE, order.max = NULL, method = c("yule-walker"))
```

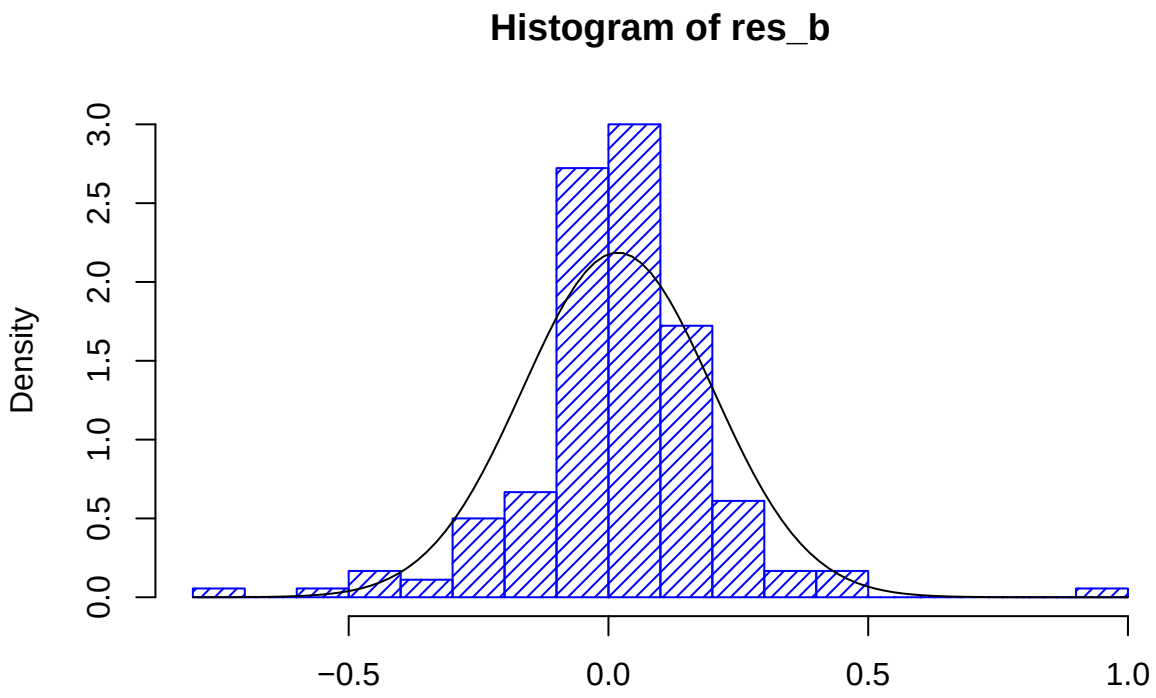
```
##
## Call:
## ar(x = res_a, aic = TRUE, order.max = NULL, method = c("yule-walker"))
##
##
## Order selected 0  sigma^2 estimated as  0.03359
```

```
# Fitted residuals to white noise/AR(0) which is good
```

Model A's residuals fit normal curve pretty well, albeit there are some heavy tailed outliers.

Model A fails the Shapiro-Wilk Test but passes all others tests

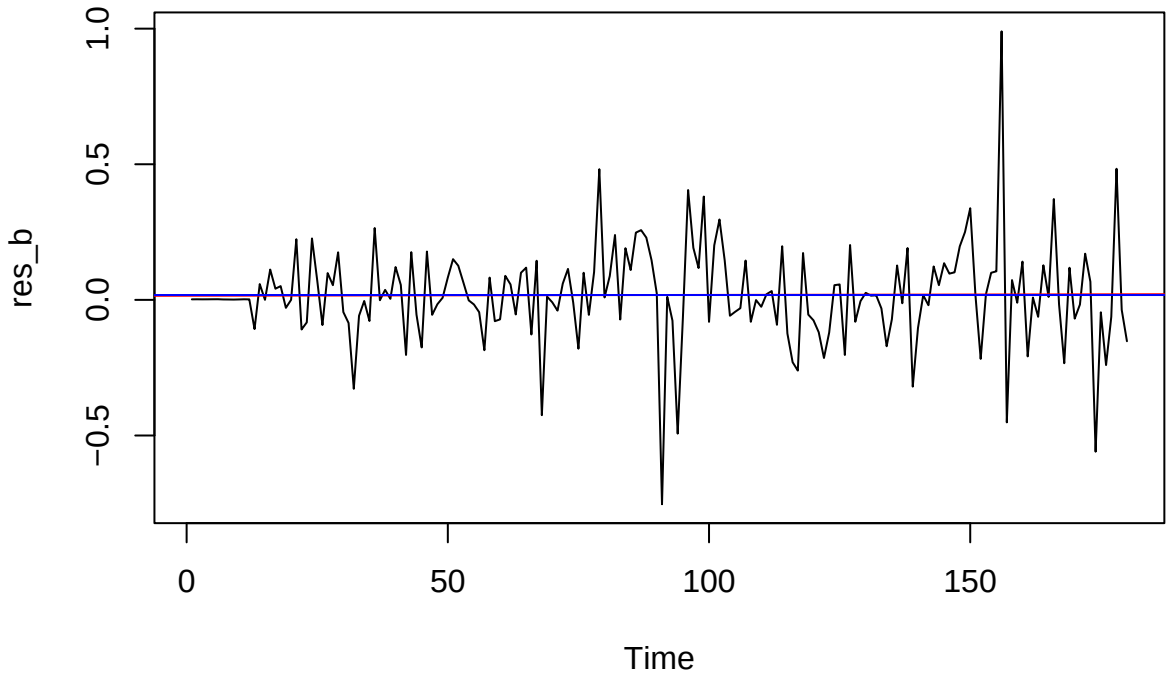
```
# Residuals of SARIMA(1,0,2)(1,1,1)12 (Model B)
res_b <- residuals(model_b)

# Histogram of Residuals
hist(res_b,density=20,breaks=20, col="blue", xlab="", prob=TRUE)
m <- mean(res_b)
std <- sqrt(var(res_b))
curve( dnorm(x,m,std), add=TRUE )
```



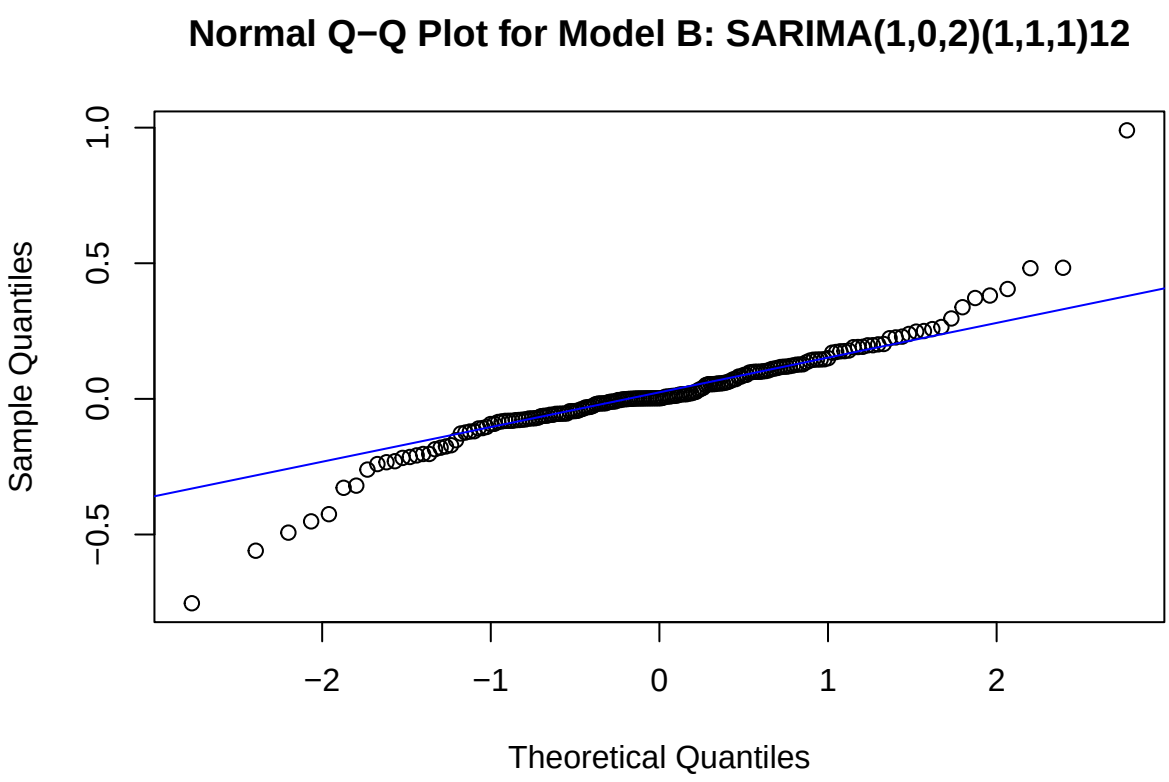

```
# residuals follow normal curve pretty well, just some heavy tailed outliers

# Plot of residuals
mean(res_b) # = 0.01792038 close to 0 and slightly lower than Model A
```

```
## [1] 0.01792038
```

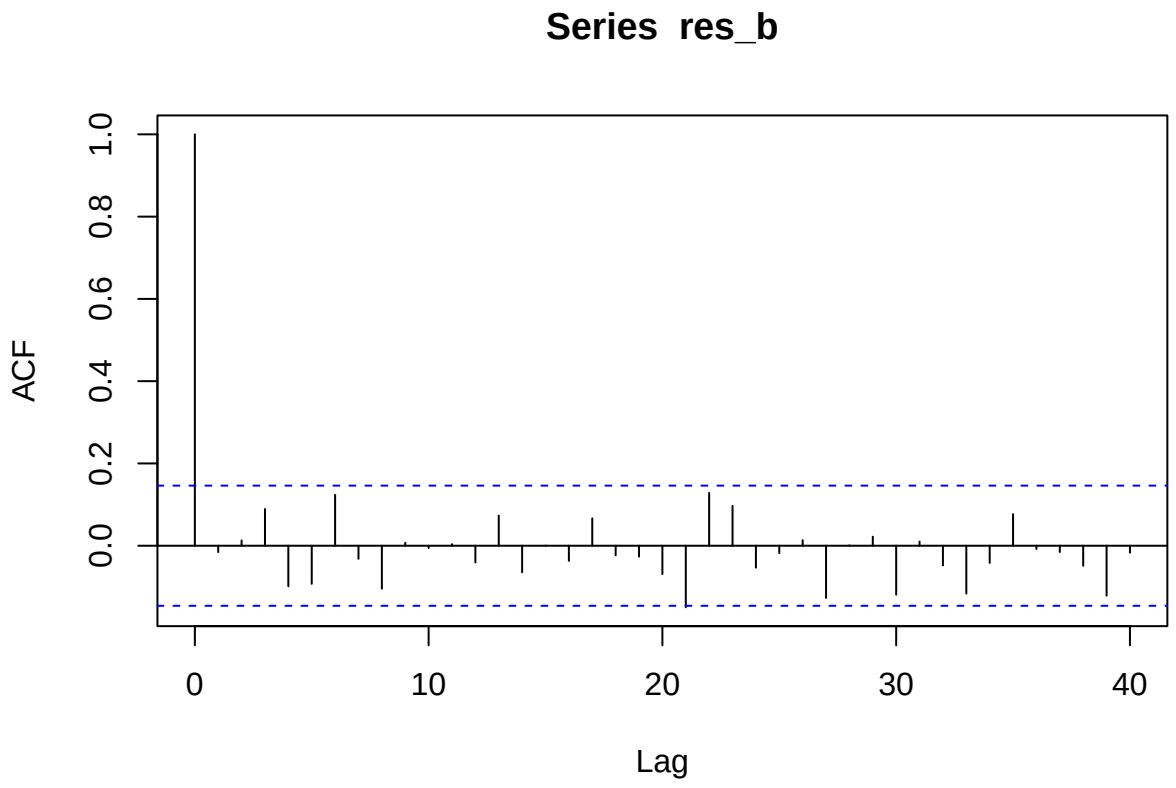
```
plot.ts(res_b)
fitt <- lm(res_b ~ as.numeric(1:length(res_b))); abline(fitt, col="red")
abline(h=mean(res_b), col="blue")
```



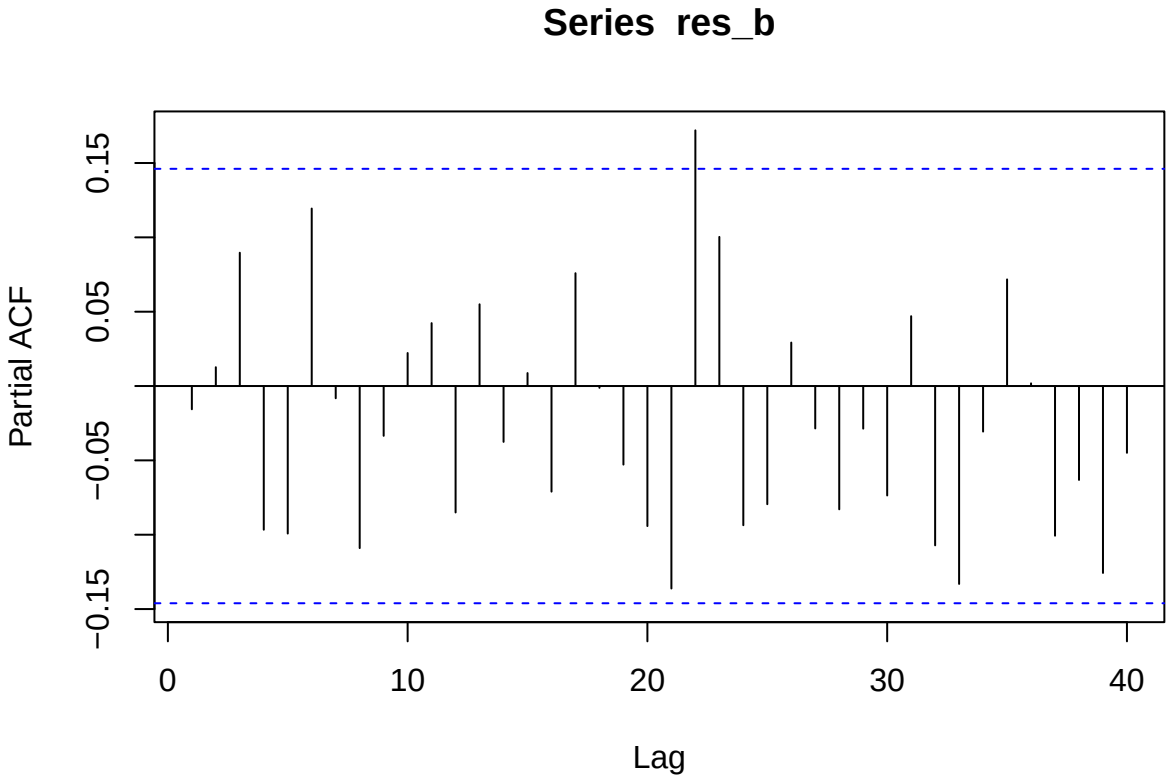

```
# no trend, seasonality, or change of variance

# Q-Q Plot
qqnorm(res_b,main= "Normal Q-Q Plot for Model B: SARIMA(1,0,2)(1,1,1)12")
qqline(res_b,col="blue")
```




```
# fits OK, some deviation at the ends

# ACF and PACF of residuals
acf(res_b, lag.max=40) # within CI at all lags
```




```
pacf(res_b, lag.max=40) # within CI at all lags except lag 22 which is OK
```




```
# LOOK FOR P-VALUES GREATER THAN 0.05
# lag = sqrt(n)
# fitdf = number of coefficients estimated
# Shapiro test
shapiro.test(res_b) # p-value = 5.93e^-9 FAIL
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res_b
## W = 0.91128, p-value = 5.93e-09
```

```
# Box-Pierce test
Box.test(res_b, lag = 14, type = c("Box-Pierce"), fitdf = 5) # p-value = 0.2292 PASS
```

```
##
##  Box-Pierce test
##
## data:  res_b
## X-squared = 11.727, df = 9, p-value = 0.2292
```

```
# Ljung-Box Test
Box.test(res_b, lag = 14, type = c("Ljung-Box"), fitdf = 5) # p-value = 0.1956 PASS
```

```
##
##  Box-Ljung test
##
## data:  res_b
## X-squared = 12.325, df = 9, p-value = 0.1956
```

```
# Mcleod-Li Test
Box.test((res_b)^2, lag = 14, type = c("Ljung-Box"), fitdf = 0) # p-value = 0.9704 PASS
```

```
##
##  Box-Ljung test
##
## data:  (res_b)^2
## X-squared = 5.8367, df = 14, p-value = 0.9704
```

```
# Yule-Walker Check
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  0.03329
```

```
# Fitted residuals to white noise/AR(0) which is good
```

Both models passed all diagnostic checking except Shapiro-Wilk test. To see which one is better, I take both of them to the forecasting step to see which model makes better predictions of the test set.

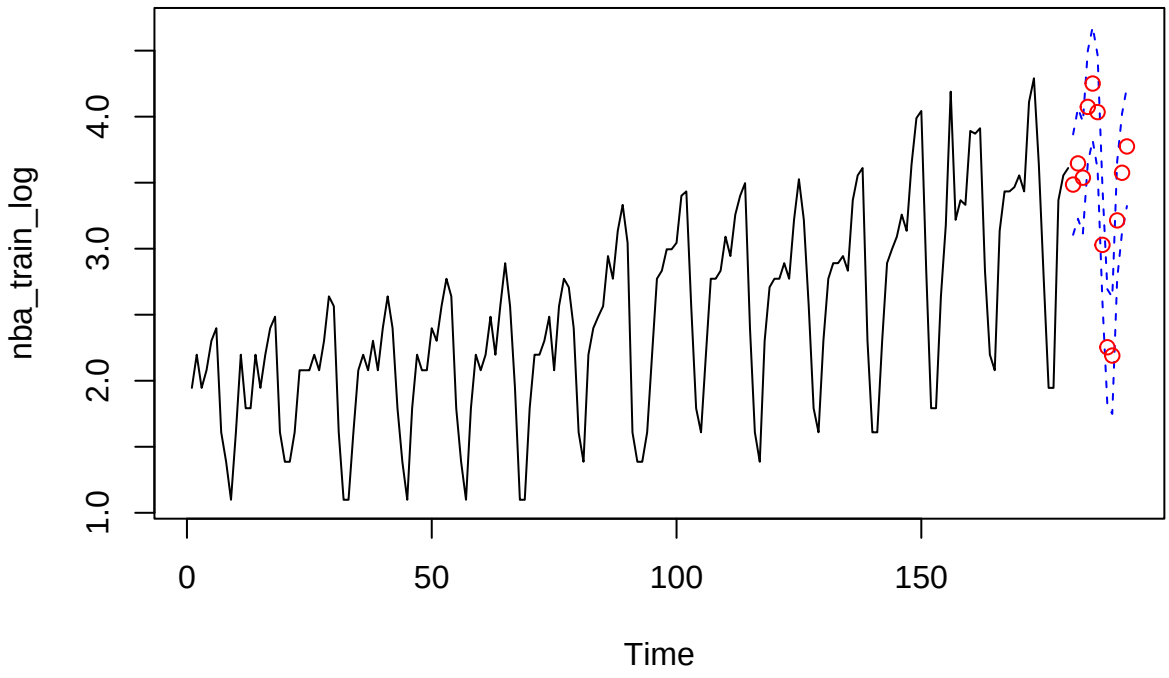
Forecasting

```
library(forecast)
fit.A <- arima(nba_train_log, order=c(1,0,2), seasonal = list(order = c(0,1,1),
                                                              period = 12), method = "ML")
forecast(fit.A)
```

```
##     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 181       3.485487 3.241841 3.729132 3.112863 3.858111
## 182       3.646019 3.377637 3.914401 3.235564 4.056474
## 183       3.537902 3.267307 3.808498 3.124062 3.951743
## 184       4.073930 3.801154 4.346705 3.656756 4.491104
## 185       4.250802 3.975880 4.525724 3.830345 4.671259
## 186       4.034547 3.757510 4.311583 3.610856 4.458237
## 187       3.028946 2.749826 3.308065 2.602069 3.455822
## 188       2.253616 1.972444 2.534788 1.823601 2.683631
## 189       2.190686 1.907492 2.473881 1.757578 2.623795
## 190       3.214389 2.929201 3.499577 2.778232 3.650546
## 191       3.574975 3.287822 3.862128 3.135812 4.014138
## 192       3.774468 3.485377 4.063559 3.332342 4.216595
```

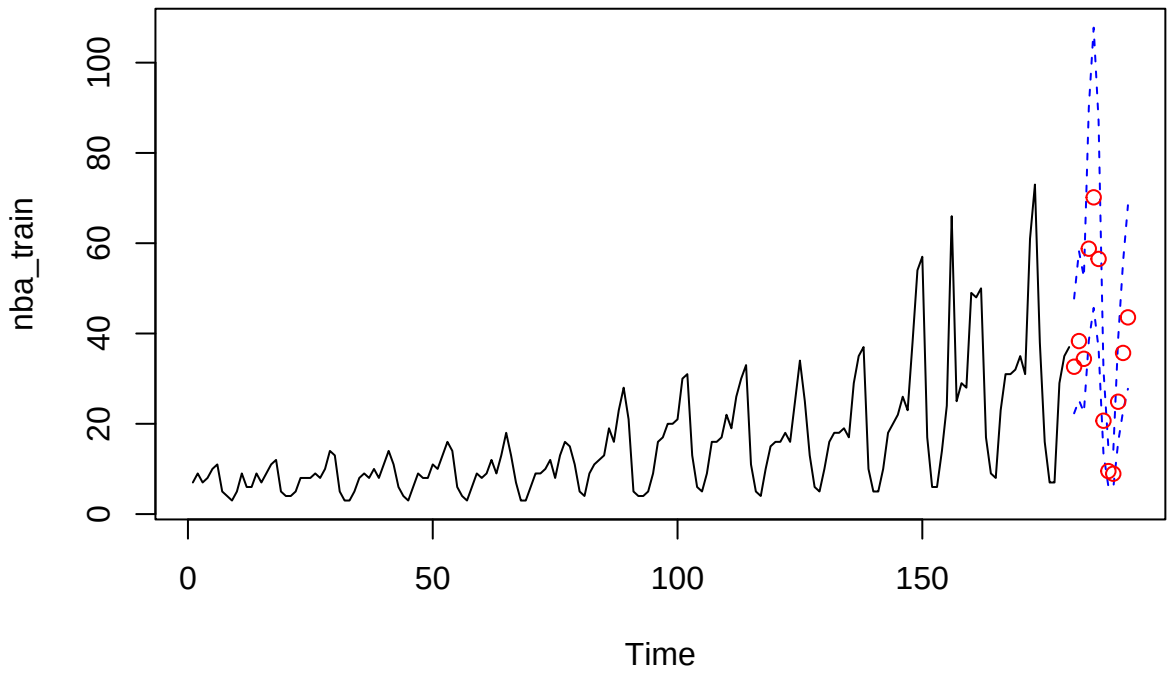
```
## 193       3.611559 3.301713 3.921404 3.137691 4.085426
## 194       3.753697 3.436198 4.071196 3.268125 4.239269
## 195       3.645185 3.324633 3.965736 3.154943 4.135426
## 196       4.180817 3.857264 4.504371 3.685985 4.675649
## 197       4.357297 4.030791 4.683802 3.857949 4.856644
## 198       4.140650 3.811239 4.470060 3.636860 4.644439
## 199       3.134658 2.802390 3.466927 2.626497 3.642819
## 200       2.358940 2.023858 2.694022 1.846477 2.871404
## 201       2.295623 1.957772 2.633474 1.778925 2.812322
## 202       3.318940 2.978362 3.659518 2.798071 3.839809
## 203       3.679142 3.335879 4.022405 3.154166 4.204117
## 204       3.878252 3.532344 4.224160 3.349231 4.407273
```

```r
# Graph with 12 forecasts on transformed data
pred.tr <- predict(fit.A, n.ahead = 12)
U.tr= pred.tr$pred + 2*pred.tr$se # upper bound of prediction interval
L.tr= pred.tr$pred - 2*pred.tr$se # lower bound
ts.plot(nba_train_log, xlim=c(1,length(nba_train_log)+12), ylim = c(min(nba_train_log),max(U.tr)))
lines(U.tr, col="blue", lty="dashed")
lines(L.tr, col="blue", lty="dashed")
points((length(nba_train_log)+1):(length(nba_train_log)+12), pred.tr$pred, col="red")
```

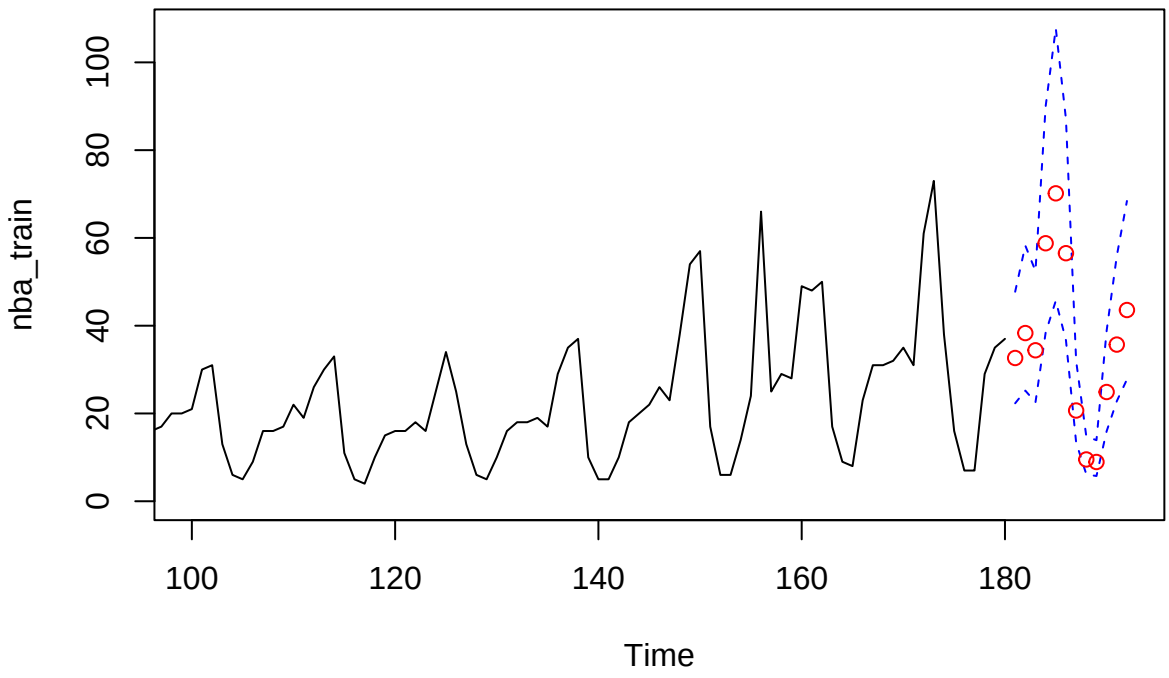



```r
# Graph with forecasts on original data
pred.orig <- exp(pred.tr$pred)
U= exp(U.tr)
```

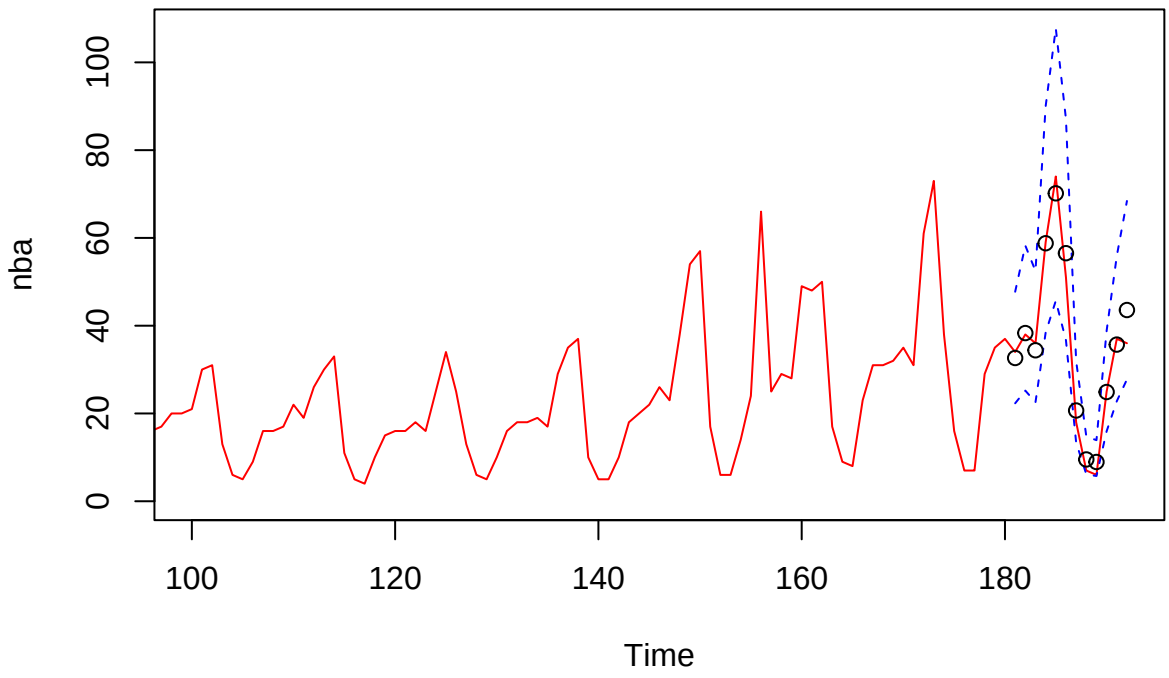
```
L= exp(L.tr)
ts.plot(nba_train, xlim=c(1,length(nba_train)+12), ylim = c(min(nba_train),max(U)))
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points((length(nba_train)+1):(length(nba_train)+12), pred.orig, col="red")
```




```
# Zoom into graph starting from entry 100
ts.plot(nba_train, xlim = c(100,length(nba_train)+12), ylim = c(0,max(U)))
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points((length(nba_train)+1):(length(nba_train)+12), pred.orig, col="red")
```




```
# All predictions fall within prediciton interval

# Zoom into graph starting from entry 100 with forecasts and true values
ts.plot(nba, xlim = c(100,length(nba_train)+12), ylim = c(0,max(U)), col="red")
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points((length(nba_train)+1):(length(nba_train)+12), pred.orig, col="black")
```




```
# Comparing predictions and actual values
pred.orig
```

```
## Time Series:
## Start = 181
## End = 192
## Frequency = 1
##  [1] 32.638312 38.321794 34.394695 58.787520 70.161661 56.517294 20.675420
##  [8]  9.522107  8.941349 24.888077 35.693732 43.574331
```

```
nba_test
```

```
##  [1] 34 38 36 59 74 51 18  7  6 25 37 36
```

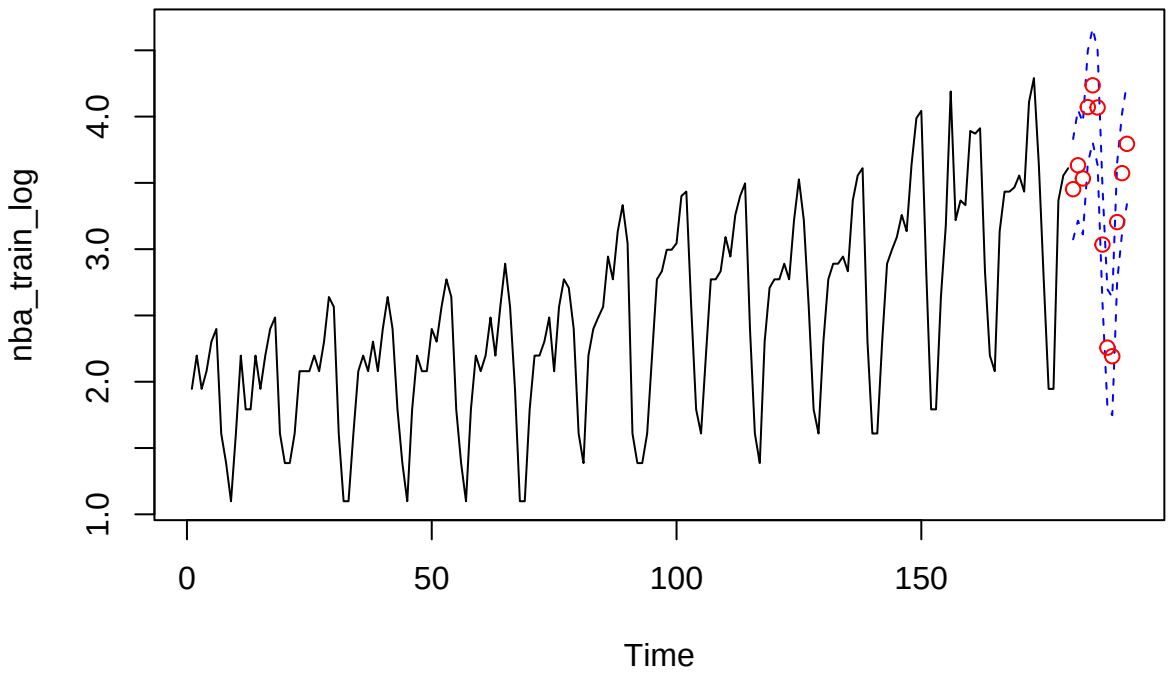
All my predictions using Model A are within the confidence interval and seem to be pretty close to actual values in test set. Good!

```
fit.B <- arima(nba_train_log, order=c(1,0,2), seasonal = list(order = c(1,1,1),
                                                              period = 12), method = "ML")
forecast(fit.B)
```

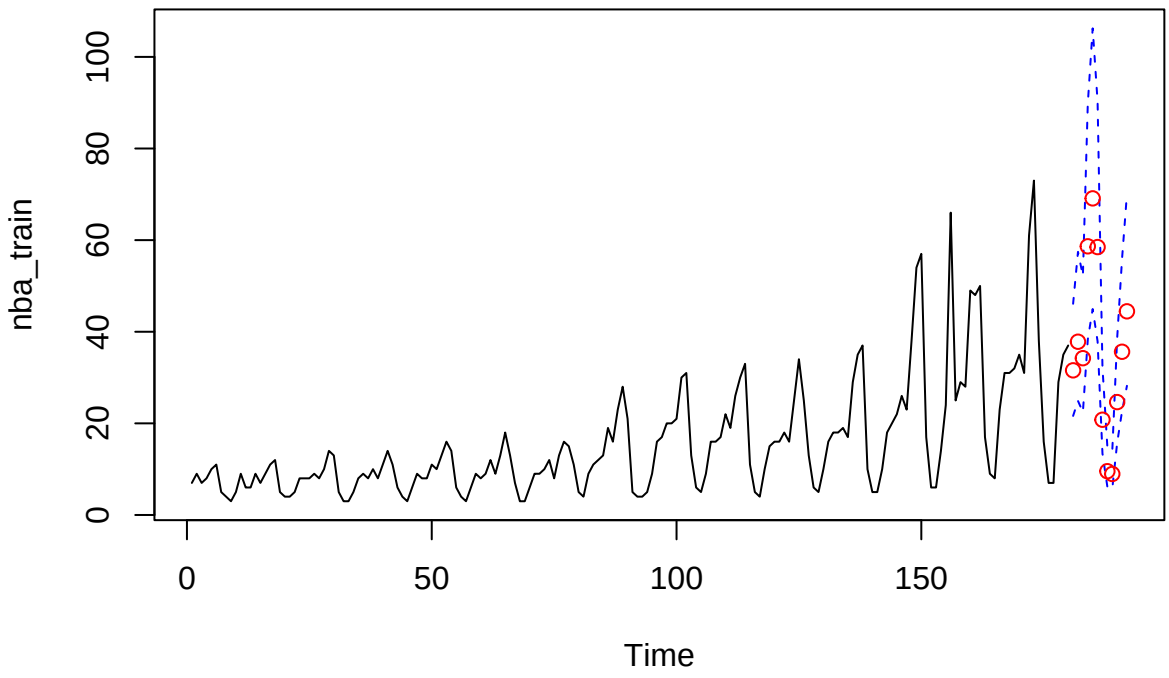
```
##     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 181       3.452362 3.209824 3.694900 3.081433 3.823292
## 182       3.632871 3.365063 3.900679 3.223294 4.042448
```

```
## 183       3.533131 3.262730 3.803532 3.119588 3.946674
## 184       4.071627 3.798683 4.344572 3.654195 4.489060
## 185       4.235765 3.960326 4.511204 3.814517 4.657013
## 186       4.068766 3.790879 4.346653 3.643774 4.493758
## 187       3.034634 2.754344 3.314925 2.605968 3.463301
## 188       2.257096 1.974447 2.539745 1.824822 2.689370
## 189       2.192210 1.907245 2.477174 1.756393 2.628026
## 190       3.204328 2.917088 3.491568 2.765033 3.643623
## 191       3.573318 3.283844 3.862792 3.130605 4.016030
## 192       3.794474 3.502804 4.086144 3.348404 4.240544
## 193       3.603881 3.296033 3.911729 3.133068 4.074694
## 194       3.742757 3.428155 4.057359 3.261615 4.223899
## 195       3.639588 3.321801 3.957376 3.153574 4.125602
## 196       4.195617 3.874707 4.516528 3.704827 4.686407
## 197       4.361187 4.037214 4.685160 3.865714 4.856661
## 198       4.129959 3.802983 4.456935 3.629892 4.630026
## 199       3.117396 2.787472 3.447319 2.612821 3.621970
## 200       2.332839 2.000023 2.665654 1.823841 2.841836
## 201       2.275872 1.940217 2.611526 1.762533 2.789211
## 202       3.341021 3.002579 3.679462 2.823419 3.858622
## 203       3.685737 3.344558 4.026915 3.163949 4.207524
## 204       3.884633 3.540766 4.228500 3.358734 4.410532
```

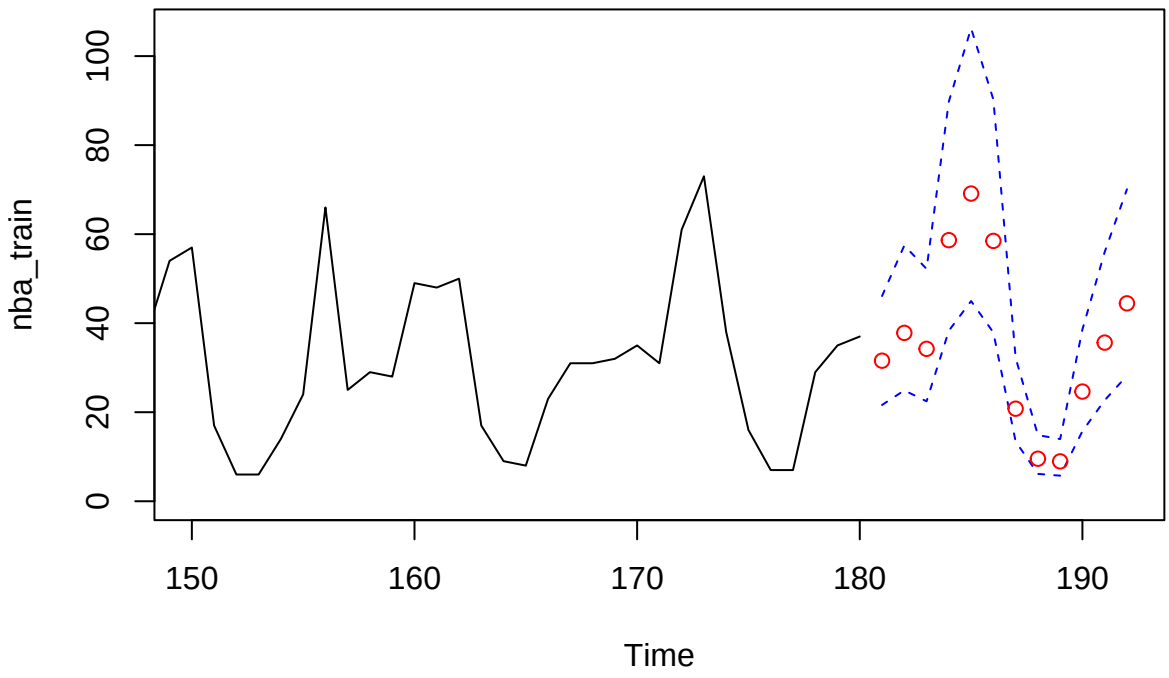
```r
# Graph with 12 forecasts on transformed data
pred.tr_b <- predict(fit.B, n.ahead = 12)
U.tr_b= pred.tr_b$pred + 2*pred.tr_b$se # upper bound of prediction interval
L.tr_b= pred.tr_b$pred - 2*pred.tr_b$se # lower bound
ts.plot(nba_train_log, xlim=c(1,length(nba_train_log)+12), ylim = c(min(nba_train_log),max(U.tr_b)))
lines(U.tr_b, col="blue", lty="dashed")
lines(L.tr_b, col="blue", lty="dashed")
points((length(nba_train_log)+1):(length(nba_train_log)+12), pred.tr_b$pred, col="red")
```



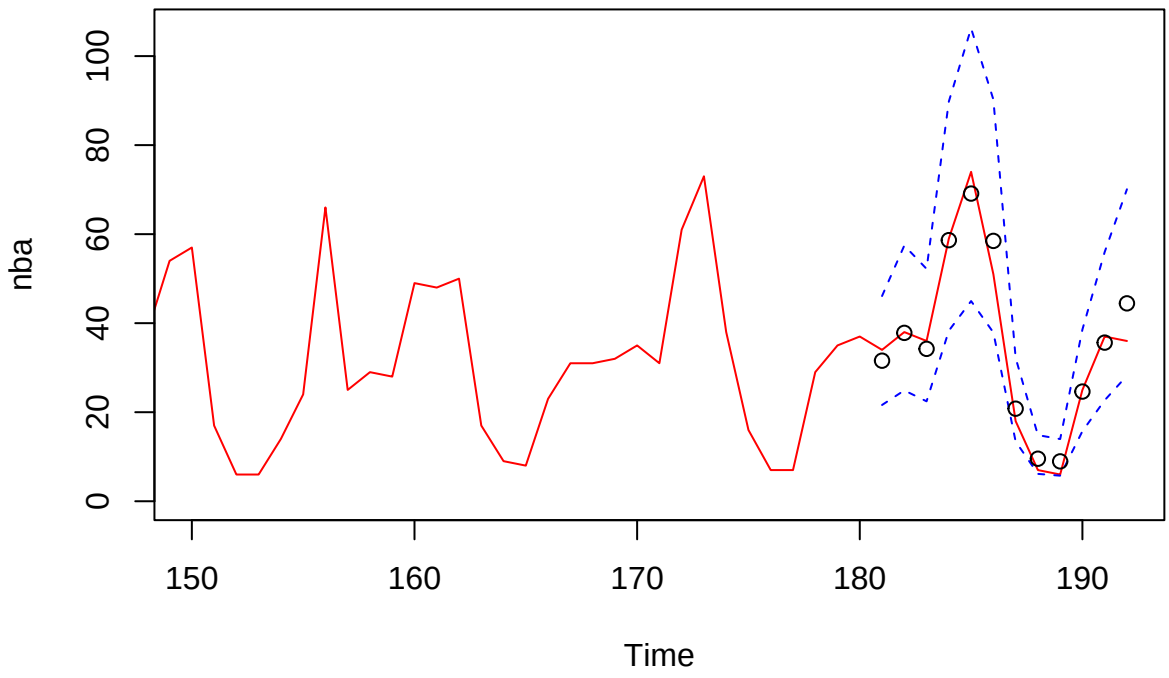

```
# Graph with forecasts on original data
pred.orig_b <- exp(pred.tr_b$pred)
U_b= exp(U.tr_b)
L_b= exp(L.tr_b)
ts.plot(nba_train, xlim=c(1,length(nba_train)+12), ylim = c(min(nba_train),max(U_b)))
lines(U_b, col="blue", lty="dashed")
lines(L_b, col="blue", lty="dashed")
points((length(nba_train)+1):(length(nba_train)+12), pred.orig_b, col="red")
```




```
# Zoom into graph starting from entry 150
ts.plot(nba_train, xlim = c(150,length(nba_train)+12), ylim = c(0,max(U_b)))
lines(U_b, col="blue", lty="dashed")
lines(L_b, col="blue", lty="dashed")
points((length(nba_train)+1):(length(nba_train)+12), pred.orig_b, col="red")
```




```
# All predictions fall within prediciton interval

# Zoom into graph starting from entry 150 with forecasts and true values
ts.plot(nba, xlim = c(150,length(nba_train)+12), ylim = c(0,max(U_b)), col="red")
lines(U_b, col="blue", lty="dashed")
lines(L_b, col="blue", lty="dashed")
points((length(nba_train)+1):(length(nba_train)+12), pred.orig_b, col="black")
```




```
# Comparing predictions and actual values
pred.orig
```

```
## Time Series:
## Start = 181
## End = 192
## Frequency = 1
##  [1] 32.638312 38.321794 34.394695 58.787520 70.161661 56.517294 20.675420
##  [8]  9.522107  8.941349 24.888077 35.693732 43.574331
```

```
pred.orig_b
```

```
## Time Series:
## Start = 181
## End = 192
## Frequency = 1
##  [1] 31.574889 37.821237 34.230979 58.652329 69.114535 58.484745 20.793375
##  [8]  9.555302  8.954977 24.638941 35.634627 44.454842
```

```
nba_test
```

```
##  [1] 34 38 36 59 74 51 18  7  6 25 37 36
```

```
sum(pred.orig-nba_test)
```

```
## [1] 13.11629
```

```
sum(pred.orig_b-nba_test)
```

```
## [1] 12.91078
```

All my predictions using Model B are within the confidence interval and seem to be pretty close to actual values in test set. Good!

I ultimately choose Model A because it is the simpler model and the predictions made by Models A and B are nearly identical and about equally as good at predicting the values of the test set.

Testing Models After Data Differenced at Lag 12 and Lag 1

```
# Comparing Variance
var(nba_train_log_12)
```

```
## [1] 0.06216348
```

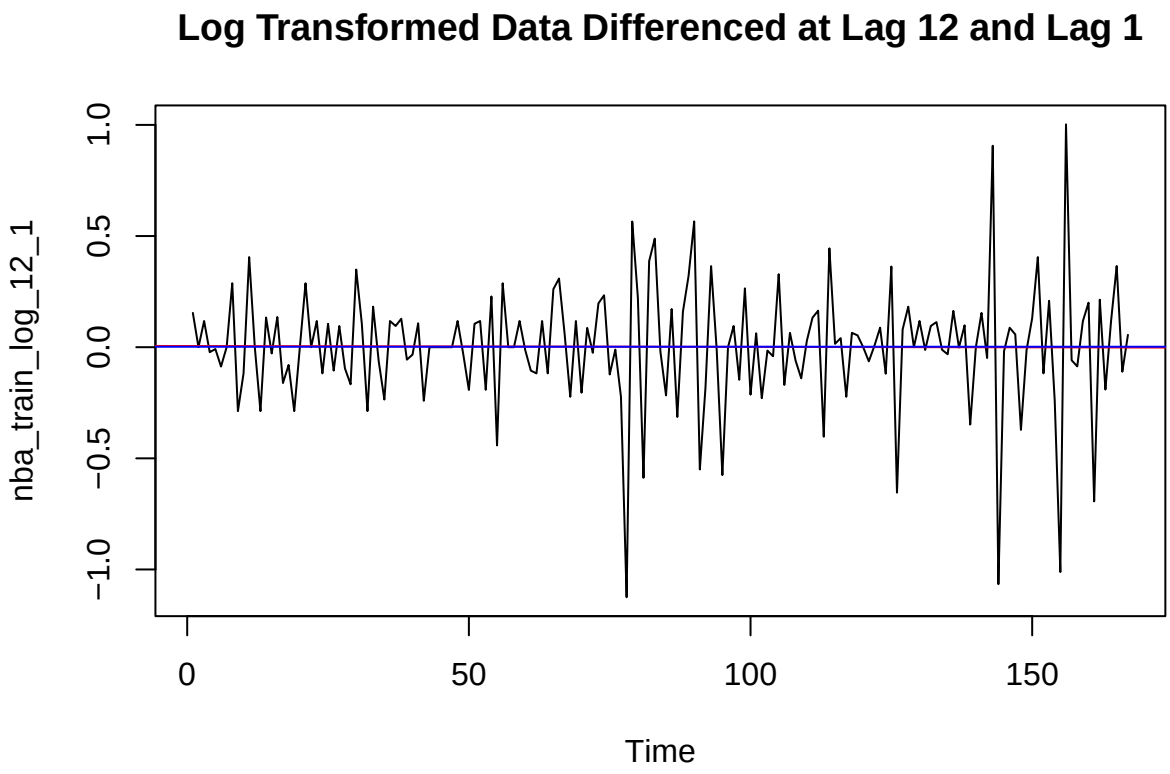
```
var(nba_train_log_12_1)
```

```
## [1] 0.07753534
```

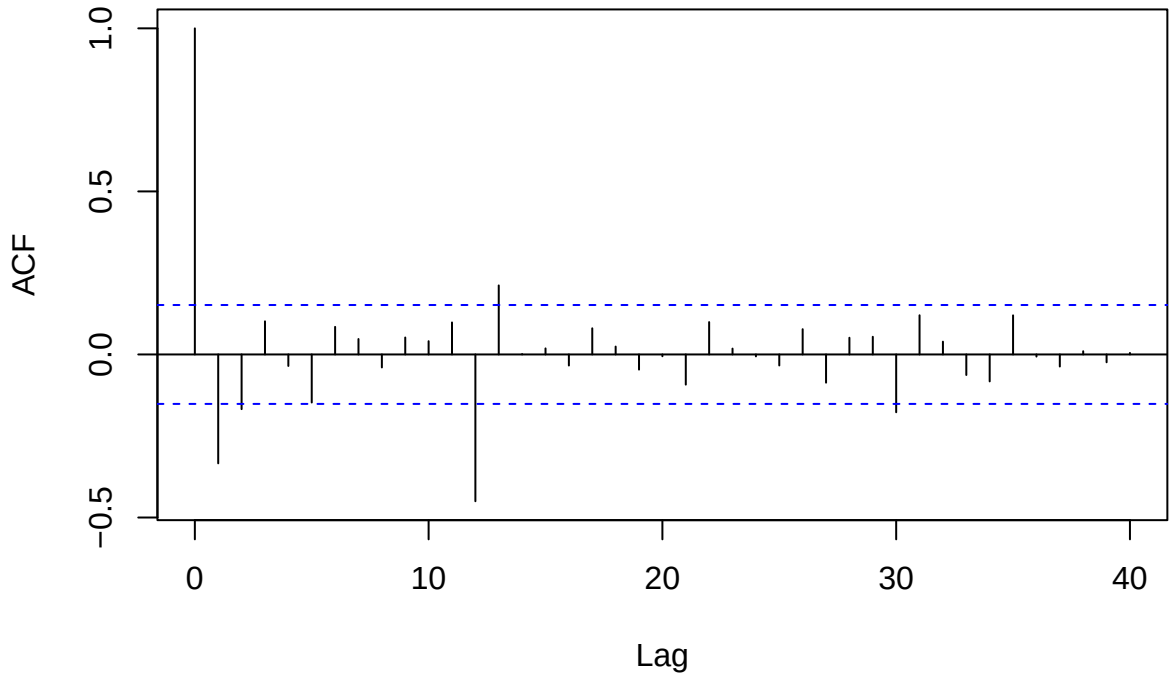
```
# Plot of data after differencing at lag 12 and lag 1
par(mfrow=c(1, 1))
plot.ts(nba_train_log_12_1, main="Log Transformed Data Differenced at Lag 12 and Lag 1")
fit <- lm(nba_train_log_12_1 ~ as.numeric(1:length(nba_train_log_12_1))); abline(fit, col="red")
mean(nba_train_log_12_1)
```

```
## [1] 0.001982523
```

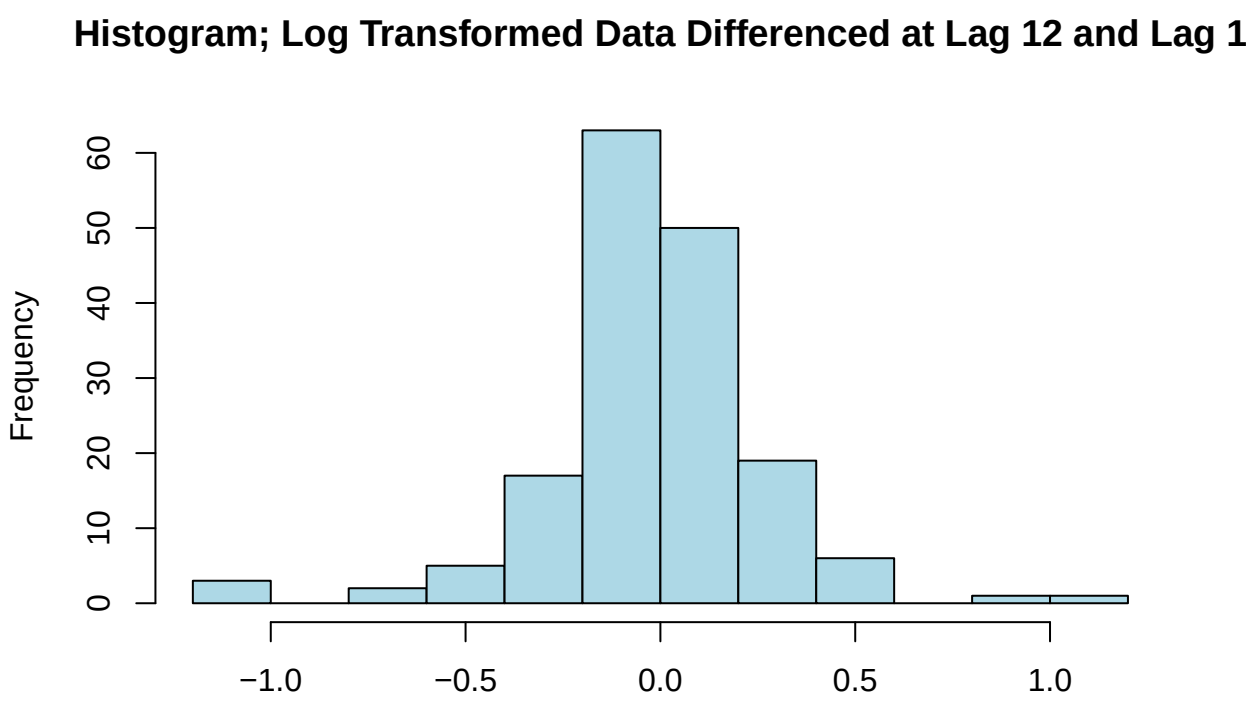
```
abline(h=mean(nba_train_log_12_1), col="blue")
```



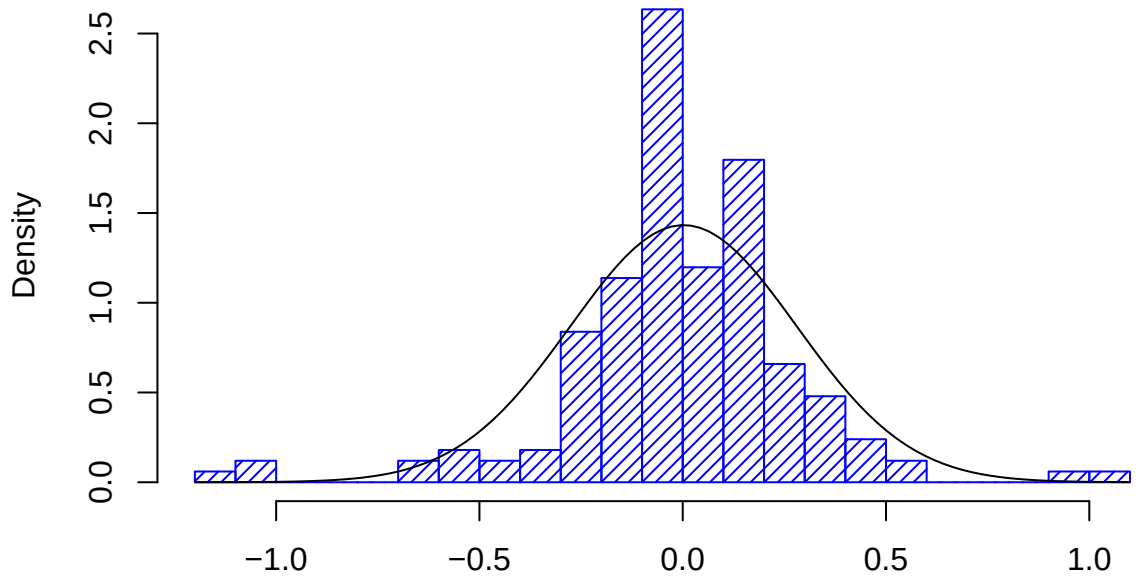

```
# Checking ACF to confirm stationarity
acf(nba_train_log_12_1, lag.max=40,
    main="ACF of the Log Transformed Data Differenced at Lag 12 and Lag 1")
```



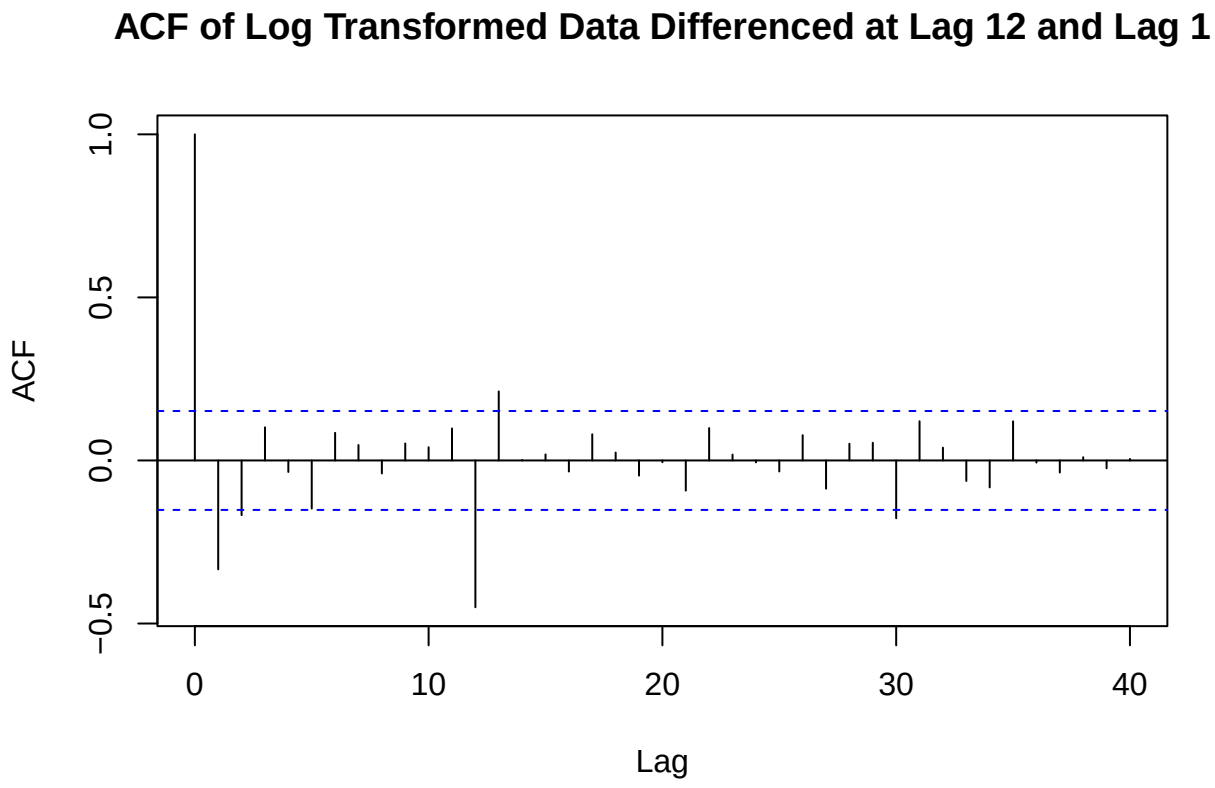

```
# looks stationary

# Checking histograms to see if data is symmetric and Gaussian
hist(nba_train_log_12_1, col="light blue", xlab="",
     main="Histogram; Log Transformed Data Differenced at Lag 12 and Lag 1")
```



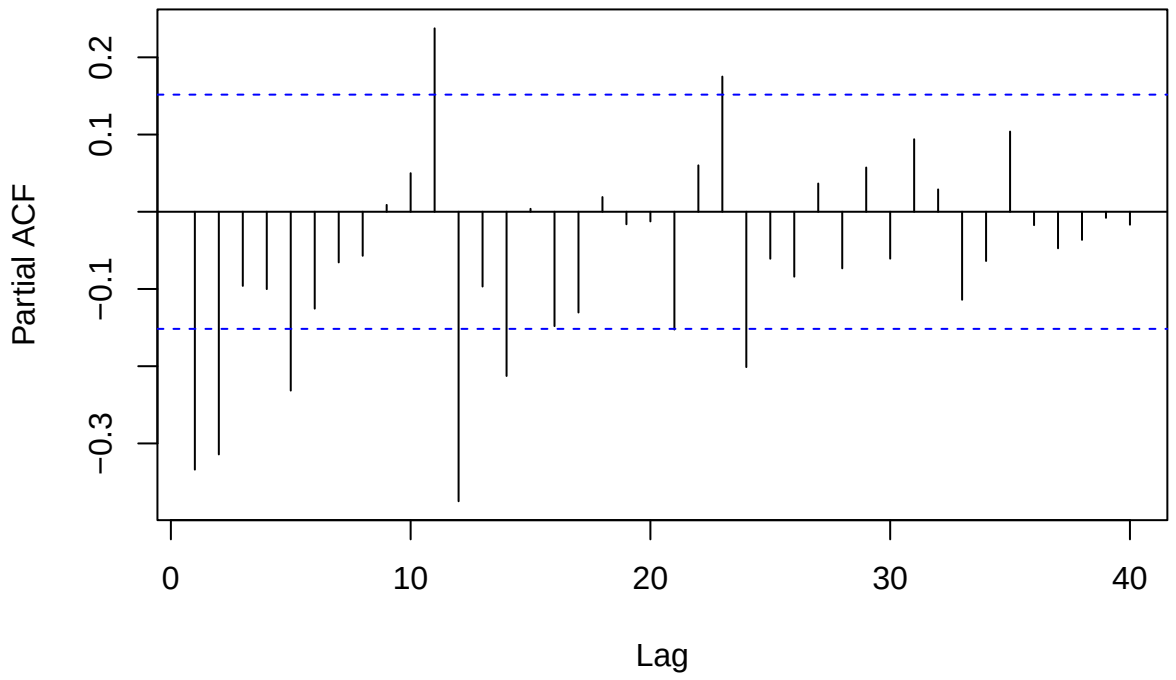

```
hist(nba_train_log_12_1, density=20,breaks=20, col="blue", xlab="",
     main ="Histogram; Log Transformed Data Differenced at Lag 12 and Lag 1", prob=TRUE)
m<-mean(nba_train_log_12_1)
std<- sqrt(var(nba_train_log_12_1))
curve( dnorm(x,m,std), add=TRUE )
```




```
# Analysis of ACF & PACF
acf(nba_train_log_12_1, lag.max=40,
    main="ACF of Log Transformed Data Differenced at Lag 12 and Lag 1")
```




```
pacf(nba_train_log_12_1, lag.max=40,
     main="PACF of Log Transformed Data Differenced at Lag 12 and Lag 1")
```




```
# Candidate models:
p <- c(0, 1, 2, 5)
df_test <- expand.grid(p=p, q=0:2, P=0:2, Q=0:1)
df_test <- cbind(df_test, AICc=NA)
# Testing Models and Computing AICcs:
for (i in 1:nrow(df_test)) {
sarima.obj <- NULL
try(arima.obj <- arima(nba_train_log, order=c(df_test$p[i], 1, df_test$q[i]),
seasonal=list(order=c(df_test$P[i], 1, df_test$Q[i]), period=12),
method="ML"))
if (!is.null(arima.obj)) { df_test$AICc[i] <- AICc(arima.obj) }
# print(df[i, ])
}
```

```
## Error in optim(init[mask], armafn, method = optim.method, hessian = TRUE,  :
##   non-finite finite-difference value [1]
## Error in optim(init[mask], armafn, method = optim.method, hessian = TRUE,  :
##   non-finite finite-difference value [1]
## Error in optim(init[mask], armafn, method = optim.method, hessian = TRUE,  :
##   non-finite finite-difference value [2]
```

```
df_test[which.min(df_test$AICc), ]
```

```
##    p q P Q      AICc
## 45 0 2 0 1 -63.03148
```

```r
head(df_test[order(df_test$AICc),]) # sorting models by lowest AICC
```

```
##    p q P Q      AICc
## 45 0 2 0 1 -63.03148
## 42 1 1 0 1 -62.55753
## 57 0 2 1 1 -62.24675
## 54 1 1 1 1 -61.23708
## 69 0 2 2 1 -61.12960
## 46 1 2 0 1 -60.94909
```

```r
# Code from lab 7 page 7
df_test[order(df_test$AICc),]
```

```
##    p q P Q       AICc
## 45 0 2 0 1 -63.031481
## 42 1 1 0 1 -62.557527
## 57 0 2 1 1 -62.246754
## 54 1 1 1 1 -61.237084
## 69 0 2 2 1 -61.129602
## 46 1 2 0 1 -60.949087
## 43 2 1 0 1 -60.585746
## 58 1 2 1 1 -60.239235
## 47 2 2 0 1 -59.791949
## 55 2 1 1 1 -59.325117
## 70 1 2 2 1 -59.037694
## 59 2 2 1 1 -58.844662
## 44 5 1 0 1 -58.473337
## 52 5 0 1 1 -58.360606
## 64 5 0 2 1 -57.929406
## 71 2 2 2 1 -57.778194
## 40 5 0 0 1 -57.555953
## 48 5 2 0 1 -57.119147
## 60 5 2 1 1 -56.992358
## 56 5 1 1 1 -56.198548
## 72 5 2 2 1 -56.042542
## 68 5 1 2 1 -55.746251
## 33 0 2 2 0 -55.389597
## 28 5 0 2 0 -54.753915
## 30 1 1 2 0 -54.465278
## 31 2 1 2 0 -54.465278
## 34 1 2 2 0 -54.190123
## 32 5 1 2 0 -52.686390
## 35 2 2 2 0 -52.430992
## 36 5 2 2 0 -52.395984
## 41 0 1 0 1 -50.973745
## 53 0 1 1 1 -50.925018
## 65 0 1 2 1 -50.609126
## 66 1 1 2 1 -50.609126
## 67 2 1 2 1 -50.609126
## 29 0 1 2 0 -48.227092
## 51 2 0 1 1 -45.566101
## 63 2 0 2 1 -45.447125
## 39 2 0 0 1 -45.053518
```

```
## 27 2 0 2 0 -44.641065
## 21 0 2 1 0 -43.800572
## 22 1 2 1 0 -42.994038
## 18 1 1 1 0 -42.047899
## 16 5 0 1 0 -41.005455
## 23 2 2 1 0 -40.937476
## 20 5 1 1 0 -40.719501
## 19 2 1 1 0 -40.353818
## 24 5 2 1 0 -39.695262
## 17 0 1 1 0 -35.972456
## 15 2 0 1 0 -31.757439
## 38 1 0 0 1 -29.174471
## 62 1 0 2 1 -28.951035
## 50 1 0 1 1 -28.027778
## 26 1 0 2 0 -24.589721
## 37 0 0 0 1 -16.448652
## 49 0 0 1 1 -15.266987
## 61 0 0 2 1 -15.224794
## 14 1 0 1 0  -8.307734
## 25 0 0 2 0  -7.758432
## 11 2 2 0 0  -7.051297
## 6  1 1 0 0  -5.344547
## 10 1 2 0 0  -3.854190
## 7  2 1 0 0  -3.549125
## 9  0 2 0 0  -2.379068
## 12 5 2 0 0   1.019805
## 13 0 0 1 0   6.962576
## 8  5 1 0 0   6.983247
## 5  0 1 0 0   7.821272
## 4  5 0 0 0   8.660672
## 3  2 0 0 0  15.094112
## 2  1 0 0 0  30.267589
## 1  0 0 0 0  47.908401
```

Variance after differencing at lag 1 and lag 12 is slightly higher but try using this data because both of the best models created using the data only differencing at lag 12 have roots in the AR part that are very close to the unit circle, which indicates under-differencing.

The plot of the data after differencing at lag 1 and lag 12 looks like white noise

After differencing at lag 12 and lag 1, the histogram of the data fits the normal curve decently. There are some heavy-tailed outliers. Won't fit normal curve perfectly or be exactly symmetrical because we don't have a lot of observations (Lec. 11 Slide 18) and our data displays short rises and sudden drops (Non-Gaussian behavior) corresponding to when the NBA is in season versus when it is not. Because of this behavior, later on when doing diagnostic checking, we don't expect the model's residuals to pass the Shapiro-Wilk Test.

From our differencing choices we know **d=1, D=1, s=12**

ACF lies outside of CI at lags 0,1,2,12,13,30 (used to determine q and Q) (disregard 30 due to Bartlett's formula)

Because ACF lies outside CI at lag 12, suspect **Q=1**

When looking for q, look between lags 1 and 12, suspect **q = 0,1, 2** (q=13 is covered by q =1) (most likely q =1)

PACF lies outside of CI at lags 1, 2, 5, 11, 12, 14, 23, 24 (used to determine p and P)

Because PACF lies outside of CI at lags 12 and 24 suspect **P=0, 1, 2**, (most likely P=2)

When looking for p, look between lags 1 and 12, suspect **p=1, 2, 5**

Choose $SARIMA(0,1,2)(0,1,1)_{12}$ for diagnostic checking because it has the lowest AICc (-63.03148) and is tied for the least amount of parameters (3) of any of the six models with the lowest AICcs. (Call this Model C)

Also consider $SARIMA(1,1,1)(0,1,1)_{12}$ for diagnostic checking because it has the second lowest AICc (-62.55753) and is tied for the least amount of parameters (3) of any of the six models with the lowest AICcs. (Call this Model D)

Model C

```
# Checking Coefficients of Model C
# Checking coefficients of SARIMA(0,1,2)(0,1,1)12 (Model C)
model_c <- arima(nba_train_log, order=c(0,1,2), seasonal = list(order = c(0,1,1),
                                                                period = 12), method = "ML")
model_c
```

```
##
## Call:
## arima(x = nba_train_log, order = c(0, 1, 2), seasonal = list(order = c(0, 1,
##     1), period = 12), method = "ML")
##
## Coefficients:
##           ma1      ma2     sma1
##       -0.5366  -0.3233  -0.6828
## s.e.   0.0766   0.0809   0.0605
##
## sigma^2 estimated as 0.03619:  log likelihood = 35.58,  aic = -63.17
```

```
AICc(model_c)
```

```
## [1] -63.03148
```

```
# 95% CIs for Coefficients of Model C
confint(model_c)
```

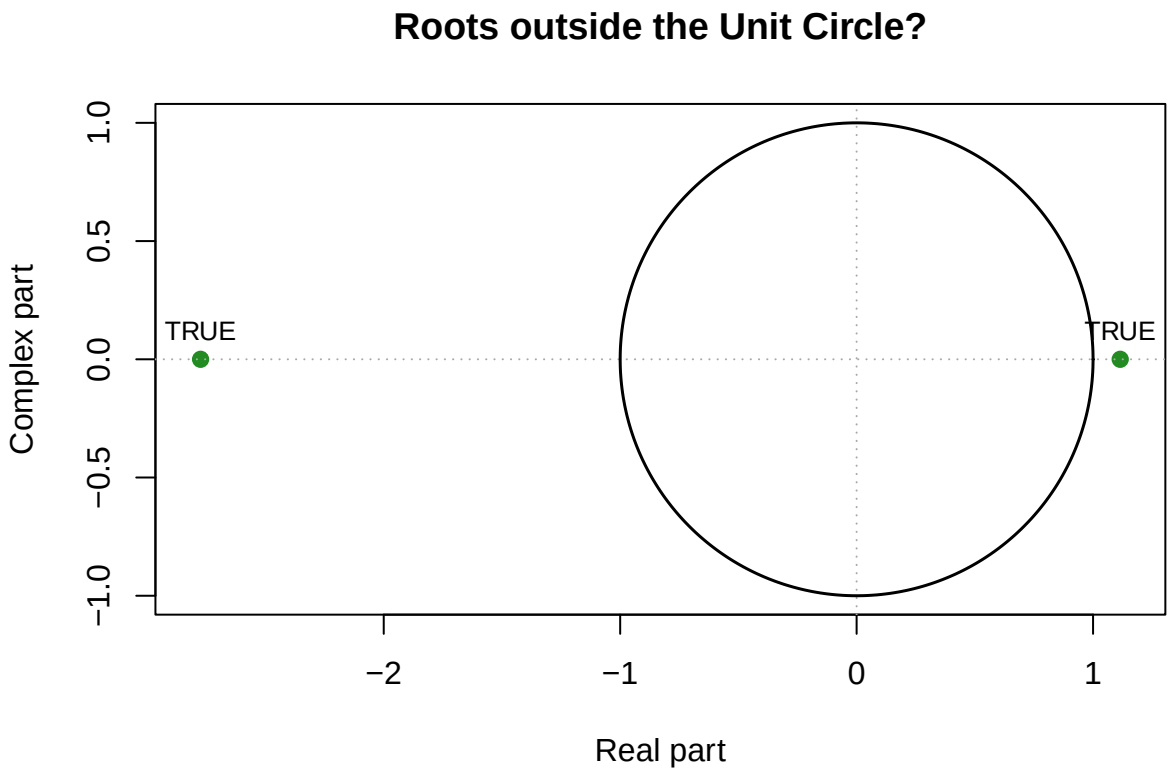
```
##           2.5 %     97.5 %
## ma1  -0.6867375 -0.3863894
## ma2  -0.4818628 -0.1646390
## sma1 -0.8012891 -0.5642391
```

```
# none of the confidence intervals contain 0 so no need to test variations of this model.

# Checking stationarity and inveritibility of Model C
# This is a pure MA model so it is automatically stationary
library(UnitCircle)
model_c
```

```
##
## Call:
## arima(x = nba_train_log, order = c(0, 1, 2), seasonal = list(order = c(0, 1,
##     1), period = 12), method = "ML")
##
## Coefficients:
##           ma1      ma2     sma1
##       -0.5366  -0.3233  -0.6828
## s.e.   0.0766   0.0809   0.0605
##
## sigma^2 estimated as 0.03619:  log likelihood = 35.58,  aic = -63.17
```

```
# Seasonal part is invertible because the absolute value of the coefficient is less than 1
# Checking MA part
uc.check(pol_ = c(1, -0.5366, -0.3233), plot_output = TRUE) # PASS
```

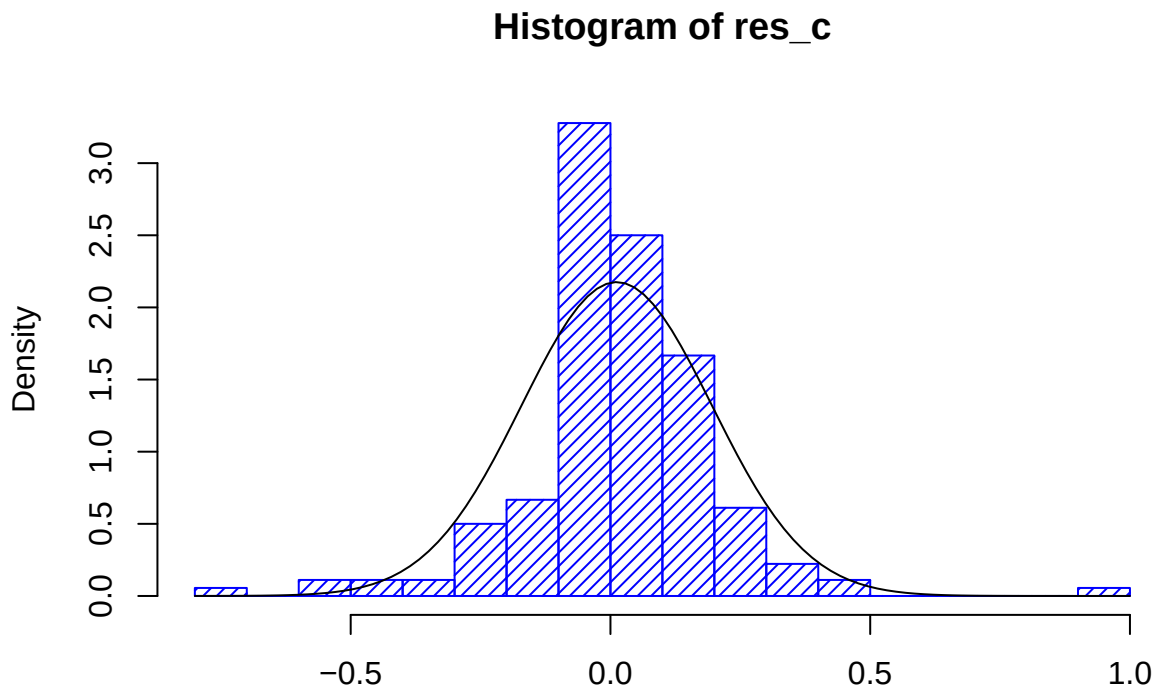
```
##        real complex outside
## 1  1.114806       0    TRUE
## 2 -2.774565       0    TRUE
## *Results are rounded to 6 digits.
```



Model C Equation: $(1-B)(1-B^{12})X_t = (1-0.5366B-0.3233B^2)(1-0.6828B^{12})Z_t$

This model is better than the ones using data after only differencing at lag 12 because it has no unit roots (or roots that are pretty much unit roots like AR part in Models A and B)

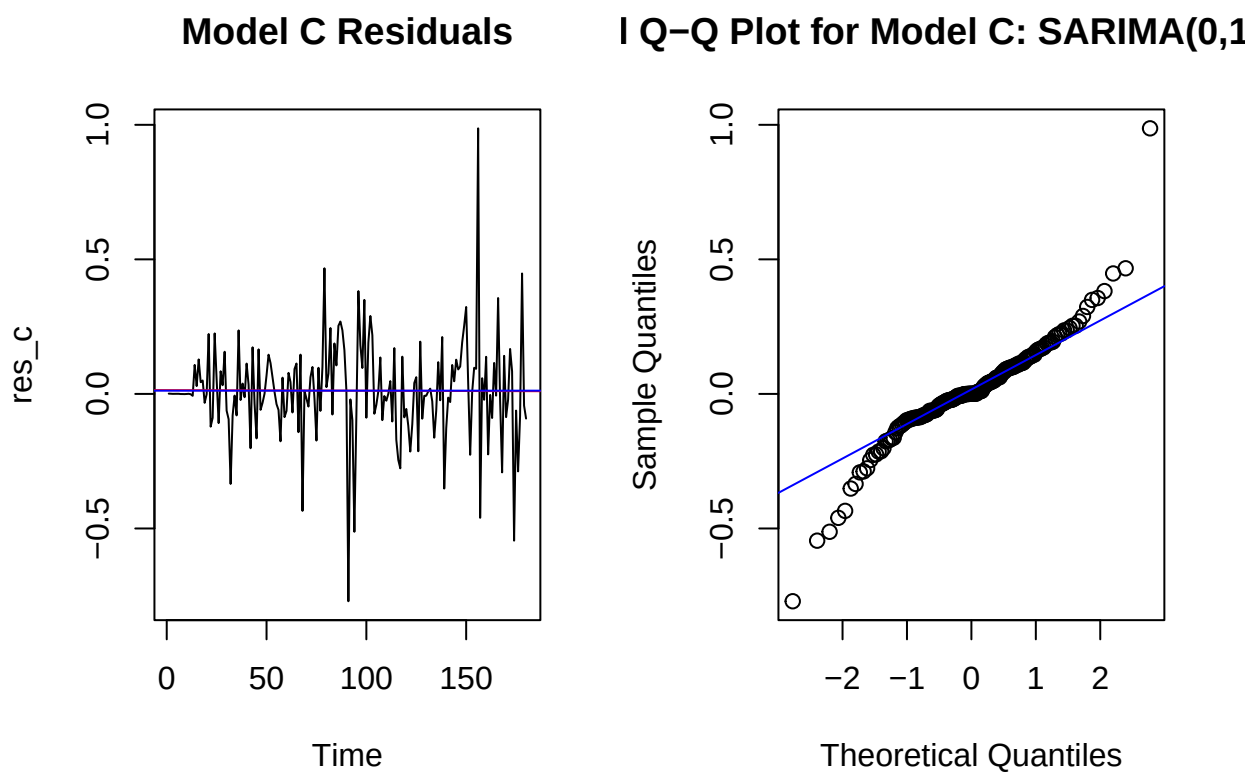
```r
# Residuals of SARIMA(0,1,2)(0,1,1)12 (Model C)
res_c <- residuals(model_c)

# Histogram of Residuals
hist(res_c,density=20,breaks=20, col="blue", xlab="", prob=TRUE)
m <- mean(res_c)
std <- sqrt(var(res_c))
curve(dnorm(x,m,std), add=TRUE )
```



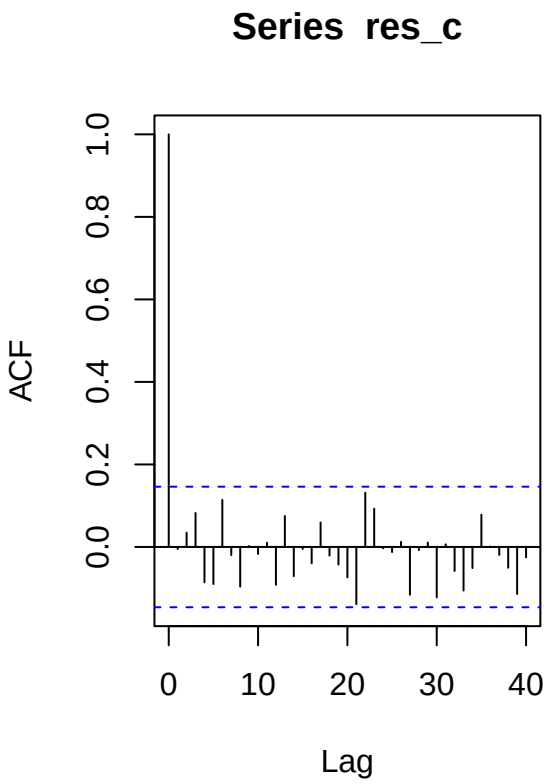

```r
# Plot of residuals
op <- par(mfrow = c(1,2))
mean(res_c) # = 0.01210592 close to 0
```

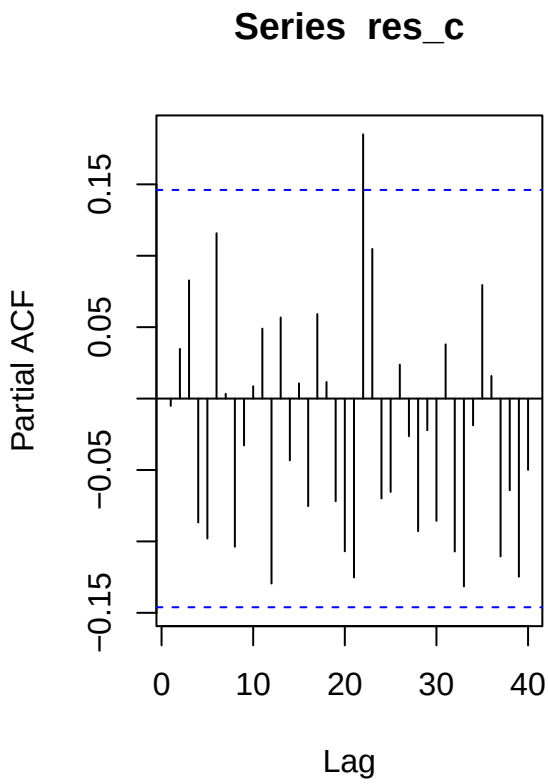
```
## [1] 0.01210592
```

```r
plot.ts(res_c, main = "Model C Residuals")
fitt <- lm(res_c ~ as.numeric(1:length(res_c))); abline(fitt, col="red")
abline(h=mean(res_c), col="blue")
# no trend, seasonality, or change of variance

# Q-Q Plot
qqnorm(res_c,main= "Normal Q-Q Plot for Model C: SARIMA(0,1,2)(0,1,1)12")
qqline(res_c,col="blue")
```




```
# fits OK, some deviation at the ends

# ACF and PACF of residuals
acf(res_c, lag.max=40) # within CI at all lags
pacf(res_c, lag.max=40) # within CI at all lags except lag 22 which is OK
```






```
# LOOK FOR P-VALUES GREATER THAN 0.05
# lag = sqrt(n) rounded to closest integer
# fitdf = number of coefficients estimated
# Shapiro test
shapiro.test(res_c) # p-value = 0.000000008938 FAIL
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res_c
## W = 0.91397, p-value = 8.938e-09
```

```
# Box-Pierce test
Box.test(res_c, lag = 14, type = c("Box-Pierce"), fitdf = 3) # p-value = 0.3788 PASS
```

```
##
##  Box-Pierce test
##
## data:  res_c
## X-squared = 11.801, df = 11, p-value = 0.3788
```

```
# Ljung-Box Test
Box.test(res_c, lag = 14, type = c("Ljung-Box"), fitdf = 3) # p-value = 0.33 PASS
```

```
##
##  Box-Ljung test
##
## data:  res_c
## X-squared = 12.461, df = 11, p-value = 0.33
```

```
# Mcleod-Li Test
Box.test((res_c)^2, lag = 14, type = c("Ljung-Box"), fitdf = 0) # p-value = 0.9579 PASS
```

```
##
##  Box-Ljung test
##
## data:  (res_c)^2
## X-squared = 6.3164, df = 14, p-value = 0.9579
```

```
# Yule-Walker Check
ar(res_c, aic = TRUE, order.max = NULL, method = c("yule-walker"))
```

```
##
## Call:
## ar(x = res_c, aic = TRUE, order.max = NULL, method = c("yule-walker"))
##
##
## Order selected 0  sigma^2 estimated as  0.03362
```

```
# Fitted residuals to white noise/AR(0) which is good
```

Model C's residuals fit normal curve pretty well, albeit there are some heavy tailed outliers.

Fails Shapiro-Wilk test (explained earlier why we expect this), but passes all other tests

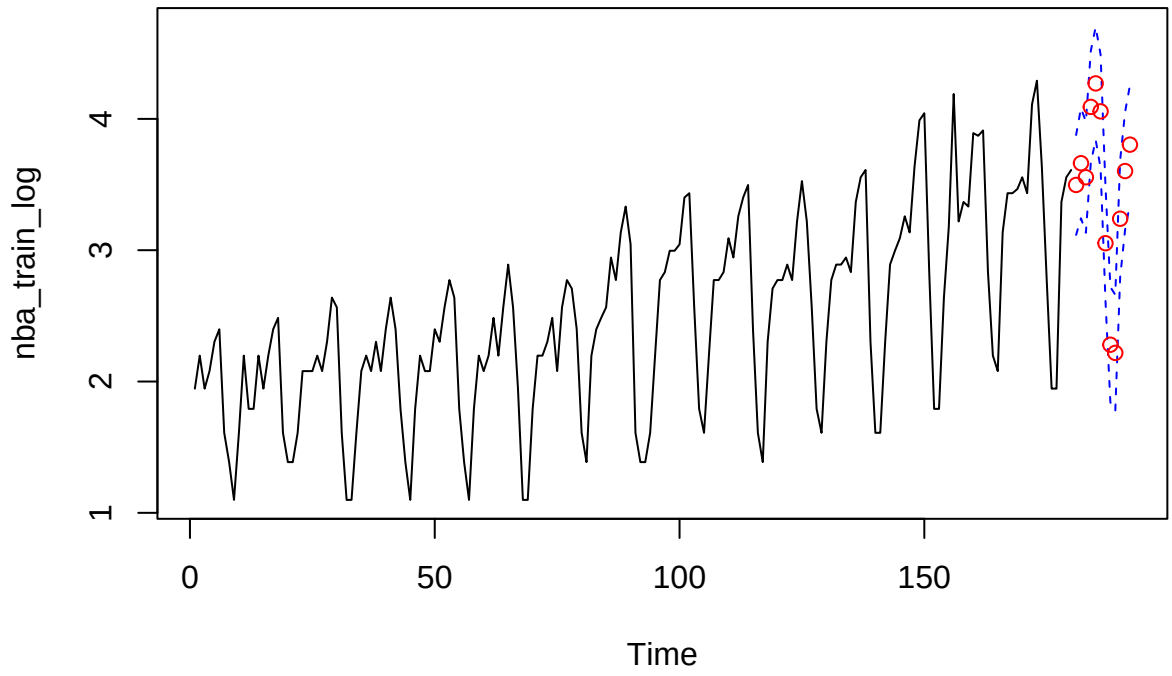
Mention that this data is similar to Influenza data in Lec.12 Slide 19 and that a heavy tailed or non-linear model may be a better fit for this data because it is not symmetric

```
fit.C <- arima(nba_train_log, order=c(0,1,2), seasonal = list(order = c(0,1,1),
                                                              period = 12), method = "ML")
forecast(fit.C)
```

```
##     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 181       3.496900 3.253084 3.740716 3.124016 3.869784
## 182       3.662552 3.393826 3.931277 3.251571 4.073532
## 183       3.555526 3.284635 3.826417 3.141235 3.969818
## 184       4.091354 3.818316 4.364393 3.673778 4.508931
## 185       4.270231 3.995062 4.545401 3.849396 4.691067
## 186       4.057245 3.779961 4.334530 3.633176 4.481315
## 187       3.053233 2.773850 3.332616 2.625954 3.480512
## 188       2.280113 1.998648 2.561579 1.849649 2.710578
## 189       2.218007 1.934473 2.501540 1.784380 2.651634
## 190       3.239615 2.954029 3.525201 2.802849 3.676381
## 191       3.602836 3.315212 3.890461 3.162953 4.042720
## 192       3.803852 3.514204 4.093500 3.360874 4.246830
## 193       3.646696 3.336318 3.957074 3.172013 4.121379
```

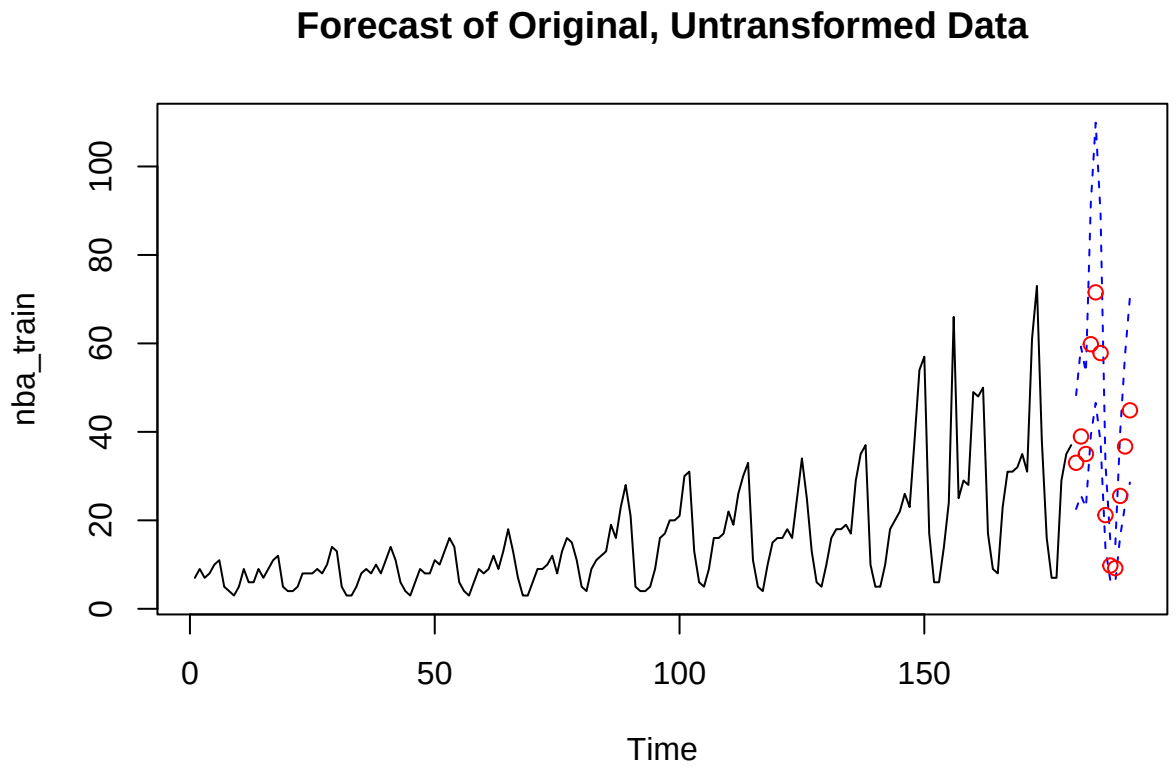
```
## 194       3.792129 3.473949 4.110308 3.305515 4.278742
## 195       3.685103 3.363754 4.006453 3.193642 4.176565
## 196       4.220931 3.896444 4.545419 3.724670 4.717193
## 197       4.399809 4.072212 4.727405 3.898793 4.900824
## 198       4.186823 3.856147 4.517498 3.681098 4.692547
## 199       3.182810 2.849083 3.516536 2.672419 3.693201
## 200       2.409690 2.072941 2.746440 1.894676 2.924705
## 201       2.347584 2.007838 2.687330 1.827987 2.867181
## 202       3.369192 3.026476 3.711908 2.845053 3.893331
## 203       3.732413 3.386753 4.078074 3.203771 4.261056
## 204       3.933429 3.584848 4.282010 3.400321 4.466537
```

```r
# Graph with 12 forecasts on transformed data
pred.tr_c <- predict(fit.C, n.ahead = 12)
U.tr_c= pred.tr_c$pred + 2*pred.tr_c$se # upper bound of prediction interval
L.tr_c= pred.tr_c$pred - 2*pred.tr_c$se # lower bound
ts.plot(nba_train_log, xlim=c(1,length(nba_train_log)+12), ylim = c(min(nba_train_log),max(U.tr_c)))
lines(U.tr_c, col="blue", lty="dashed")
lines(L.tr_c, col="blue", lty="dashed")
points((length(nba_train_log)+1):(length(nba_train_log)+12), pred.tr_c$pred, col="red")
```

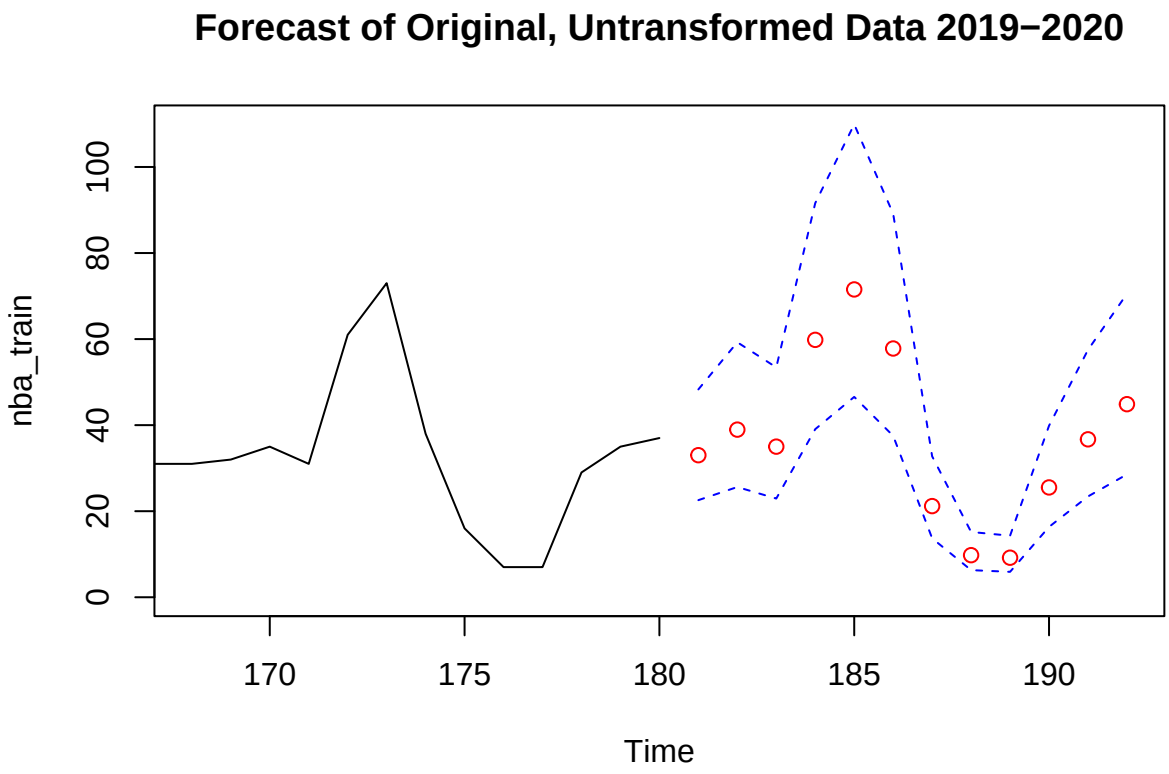



```r
# Graph with forecasts on original data
pred.orig_c <- exp(pred.tr_c$pred)
U_c= exp(U.tr_c)
L_c= exp(L.tr_c)
```

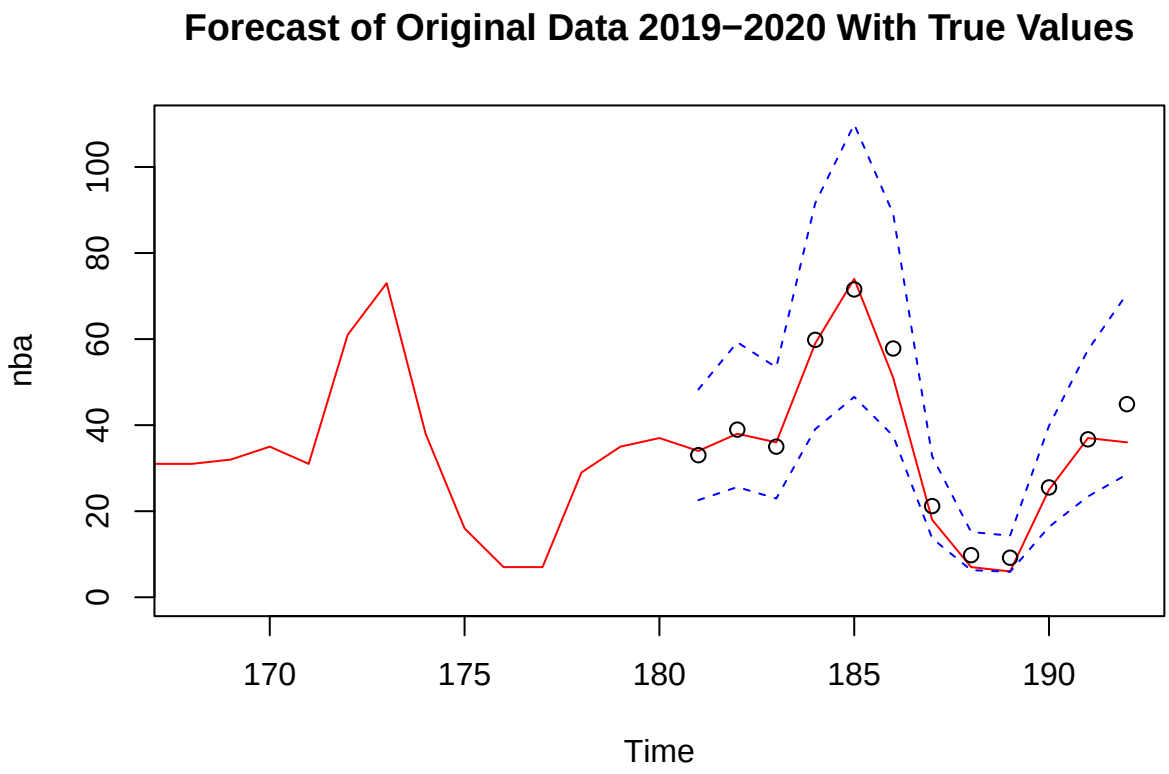
```
ts.plot(nba_train, xlim=c(1,length(nba_train)+12), ylim = c(min(nba_train),max(U_c)),
        main = "Forecast of Original, Untransformed Data")
lines(U_c, col="blue", lty="dashed")
lines(L_c, col="blue", lty="dashed")
points((length(nba_train)+1):(length(nba_train)+12), pred.orig_c, col="red")
```




```
# Zoom into graph starting from entry 168 (last two years)
ts.plot(nba_train, xlim = c(168,length(nba_train)+12), ylim = c(0,max(U_c)),
        main = "Forecast of Original, Untransformed Data 2019-2020")
lines(U_c, col="blue", lty="dashed")
lines(L_c, col="blue", lty="dashed")
points((length(nba_train)+1):(length(nba_train)+12), pred.orig_c, col="red")
```




```
# All predictions fall within prediciton interval

# Zoom into graph starting from entry 168 with forecasts and true values (last two years)
ts.plot(nba, xlim = c(168,length(nba_train)+12), ylim = c(0,max(U_c)), col="red",
        main = "Forecast of Original Data 2019-2020 With True Values")
lines(U_c, col="blue", lty="dashed")
lines(L_c, col="blue", lty="dashed")
points((length(nba_train)+1):(length(nba_train)+12), pred.orig_c, col="black")
```




```
# Comparing predictions and actual values
pred.orig_c
```

```
## Time Series:
## Start = 181
## End = 192
## Frequency = 1
##  [1] 33.012950 38.960627 35.006239 59.820859 71.538194 57.814837 21.183716
##  [8]  9.777788  9.188996 25.523891 36.702188 44.873707
```

```
nba_test
```

```
##  [1] 34 38 36 59 74 51 18  7  6 25 37 36
```

All my predictions using Model C are within the confidence interval and seem to be pretty close to actual values in test set. Good!

Model D

```
# Checking Coefficients of Model D
# Checking coefficients of SARIMA(1,1,1)(0,1,1)12 (Model D)
model_d <- arima(nba_train_log, order=c(1,1,1), seasonal = list(order = c(0,1,1),
                                                                period = 12), method = "ML")
model_d
```

```
##
## Call:
## arima(x = nba_train_log, order = c(1, 1, 1), seasonal = list(order = c(0, 1,
##     1), period = 12), method = "ML")
##
## Coefficients:
##          ar1      ma1     sma1
##       0.3716  -0.9335  -0.6794
## s.e.  0.0901   0.0468   0.0611
##
## sigma^2 estimated as 0.03623:  log likelihood = 35.35,  aic = -62.69
```

```
AICc(model_d)
```

```
## [1] -62.55753
```

```
# 95% CIs for Coefficients of Model D
confint(model_d)
```

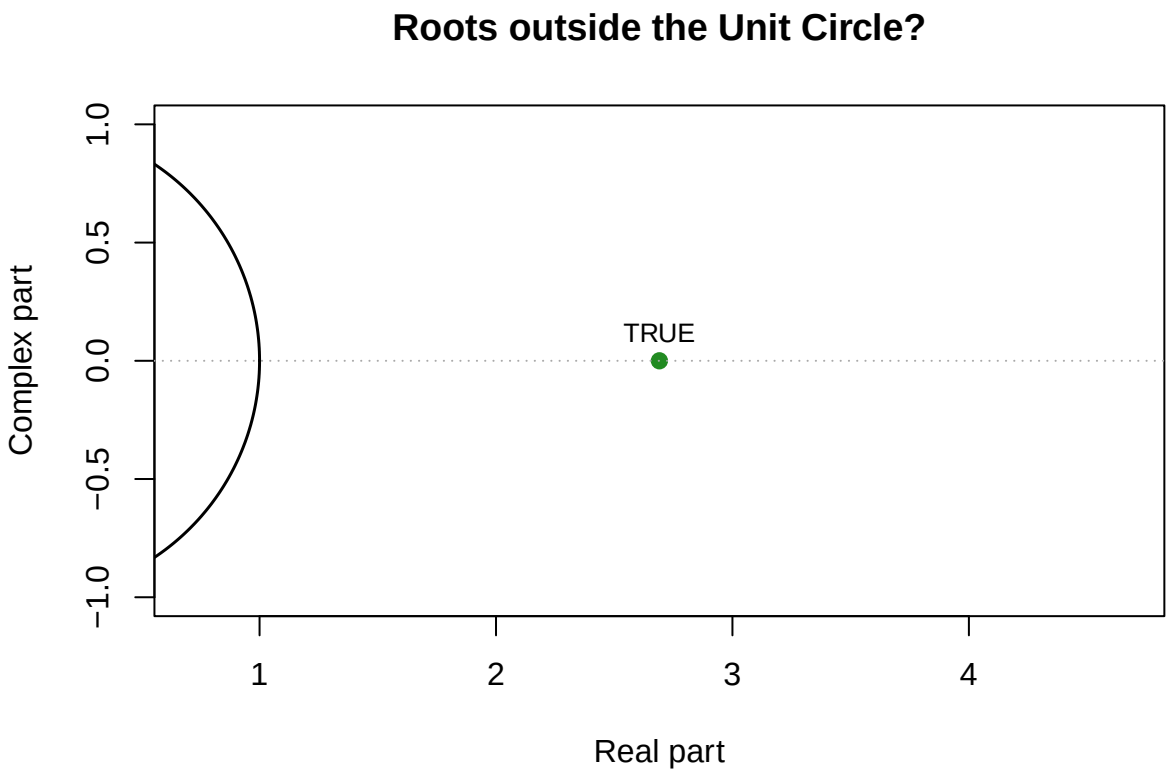
```
##           2.5 %     97.5 %
## ar1   0.1950018  0.5481501
## ma1  -1.0252150 -0.8417000
## sma1 -0.7991011 -0.5597100
```

```
# none of the confidence intervals contain 0 so no need to test variations of this model.

# Checking stationarity and inveritibility of Model D
library(UnitCircle)
model_d
```

```
##
## Call:
## arima(x = nba_train_log, order = c(1, 1, 1), seasonal = list(order = c(0, 1,
##     1), period = 12), method = "ML")
##
## Coefficients:
##          ar1      ma1     sma1
##       0.3716  -0.9335  -0.6794
## s.e.  0.0901   0.0468   0.0611
##
## sigma^2 estimated as 0.03623:  log likelihood = 35.35,  aic = -62.69
```

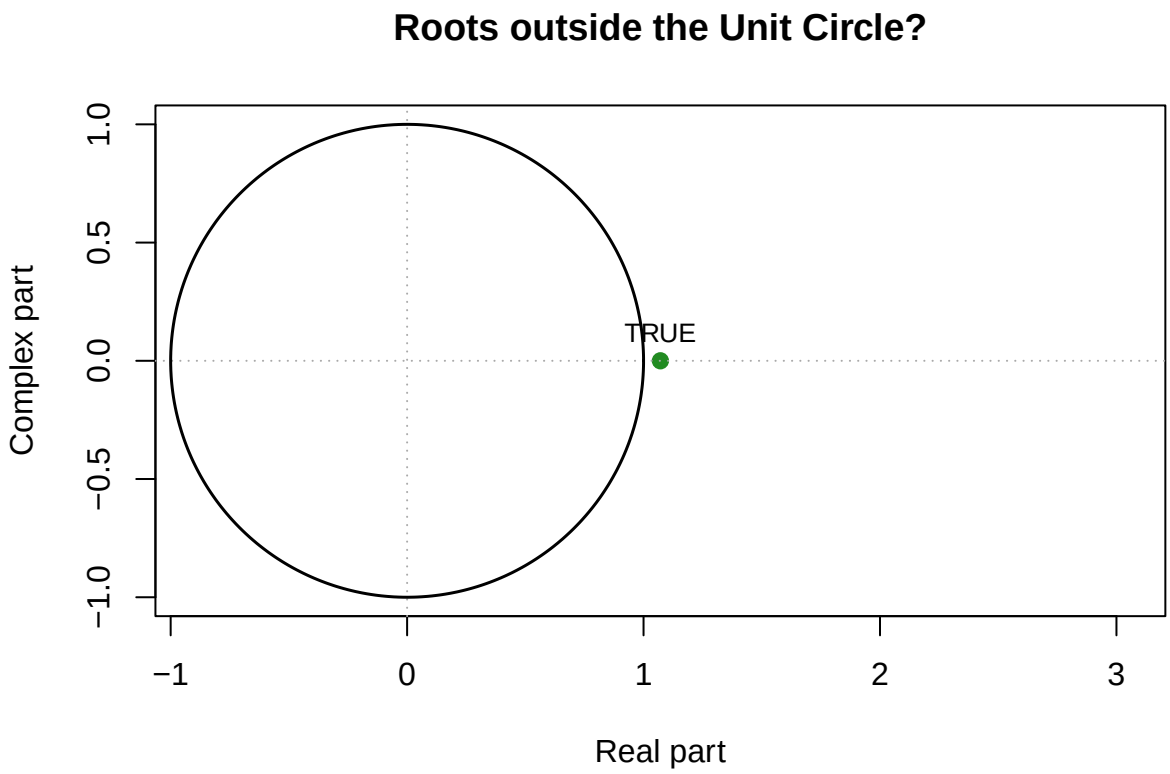
```
# Seasonal part is stationary and invertible
# because the absolute value of the coefficient is less than 1
# Checking AR part
uc.check(pol_ = c(1, -0.3716), plot_output = TRUE) # PASS
```

```
##       real complex outside
## 1 2.691066       0    TRUE
## *Results are rounded to 6 digits.
```




```
# Checking MA part
uc.check(pol_ = c(1, -0.9335), plot_output = TRUE) # PASS
```

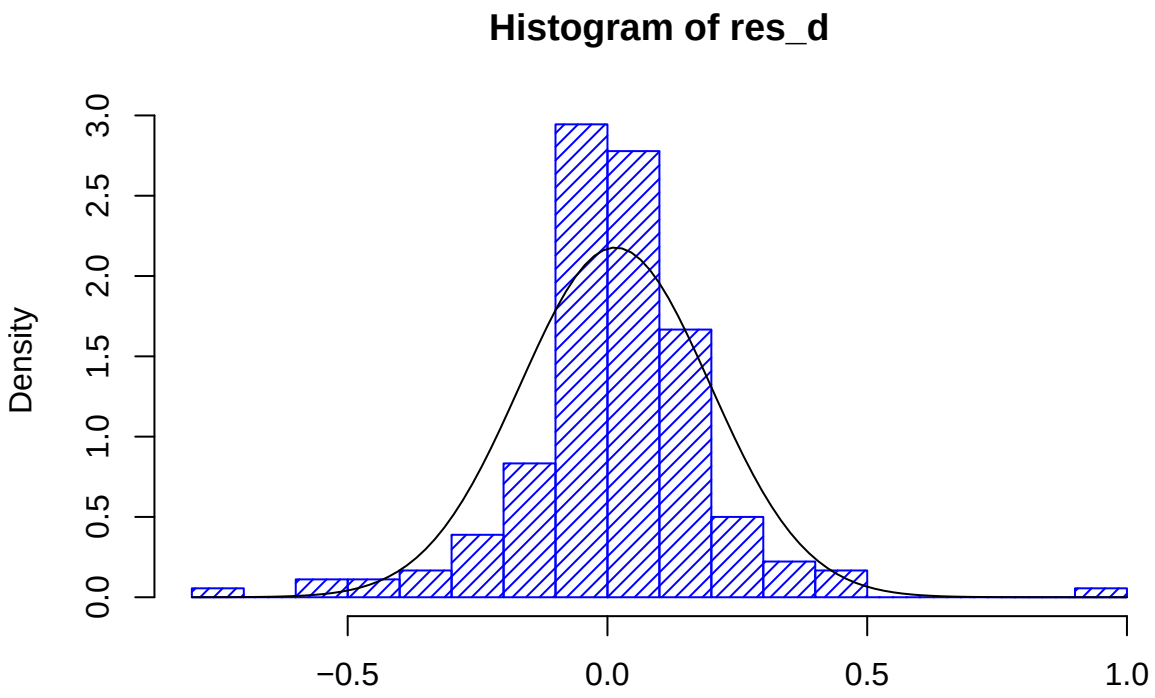
```
##       real complex outside
## 1 1.071237       0    TRUE
## *Results are rounded to 6 digits.
```



Model D Equation: $(1-0.3716B)(1-B)(1-B^{12})X_t = (1-0.9335B)(1-0.6794B^{12})Z_t$

This model is better than the ones using data after only differencing at lag 12 because it has no unit roots (or roots that are pretty much unit roots like AR part in Models A and B)

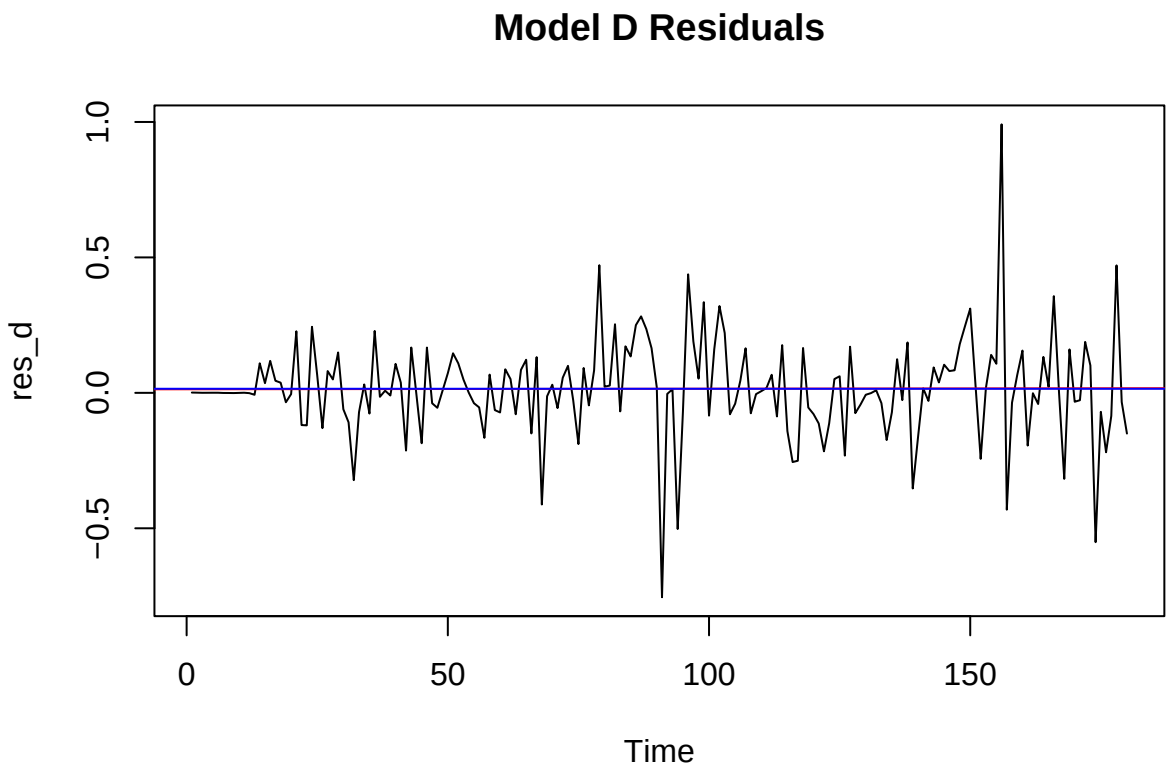
```
# Residuals of SARIMA(0,1,2)(0,1,1)12 (Model C)
res_d <- residuals(model_d)

# Histogram of Residuals
hist(res_d,density=20,breaks=20, col="blue", xlab="", prob=TRUE)
m <- mean(res_d)
std <- sqrt(var(res_d))
curve(dnorm(x,m,std), add=TRUE )
```



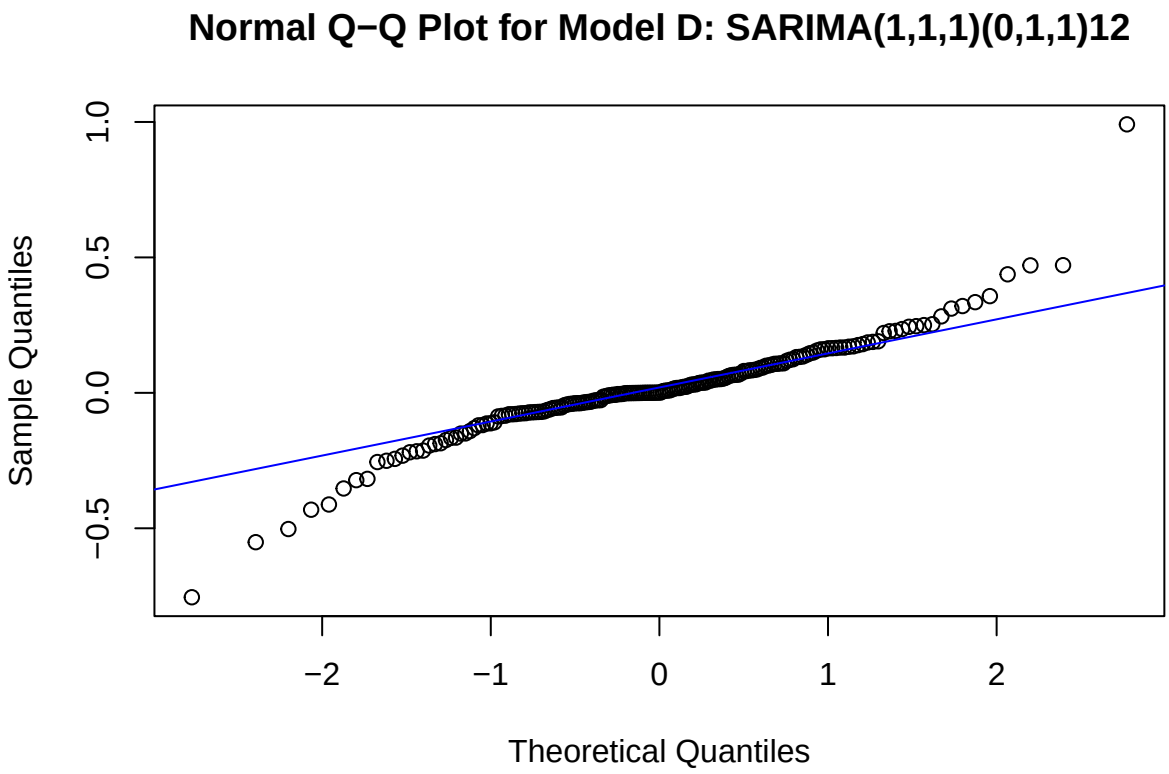

```
# Plot of residuals
mean(res_d) # = 0.01506132 close to 0
```

```
## [1] 0.01506132
```

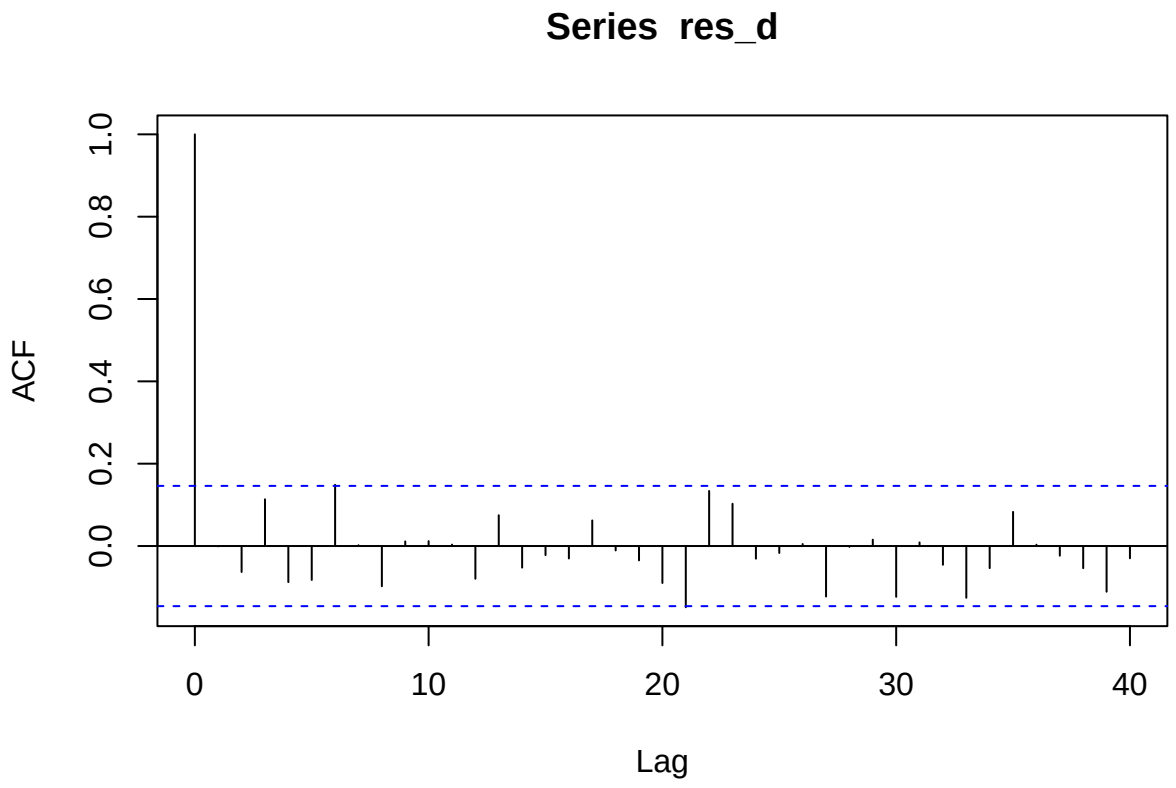
```
plot.ts(res_d, main = "Model D Residuals")
fitt <- lm(res_d ~ as.numeric(1:length(res_d))); abline(fitt, col="red")
abline(h=mean(res_d), col="blue")
```



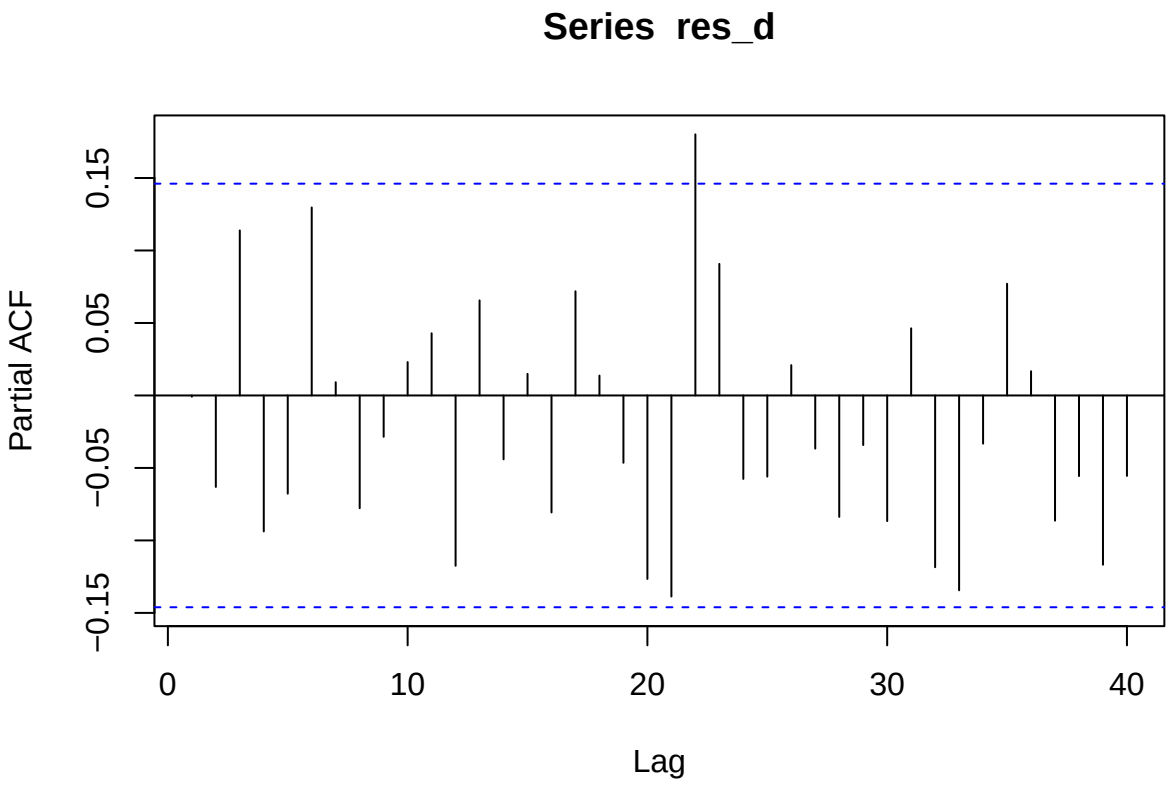

```
# no trend, seasonality, or change of variance

# Q-Q Plot
qqnorm(res_d,main= "Normal Q-Q Plot for Model D: SARIMA(1,1,1)(0,1,1)12")
qqline(res_d,col="blue")
```




```
# fits OK, some deviation at the ends

# ACF and PACF of residuals
acf(res_d, lag.max=40) # within CI at all lags (Bartlett's formula)
```




```
pacf(res_d, lag.max=40) # within CI at all lags except lag 22 which is OK
```




```
# LOOK FOR P-VALUES GREATER THAN 0.05
# lag = sqrt(n) rounded to closest integer
# fitdf = number of coefficients estimated
# Shapiro test
shapiro.test(res_d) # p-value = 0.00000000987 FAIL
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res_d
## W = 0.91461, p-value = 9.87e-09
```

```
# Box-Pierce test
Box.test(res_d, lag = 14, type = c("Box-Pierce"), fitdf = 3) # p-value = 0.231 PASS
```

```
##
##  Box-Pierce test
##
## data:  res_d
## X-squared = 14.035, df = 11, p-value = 0.231
```

```
# Ljung-Box Test
Box.test(res_d, lag = 14, type = c("Ljung-Box"), fitdf = 3) # p-value = 0.1951 PASS
```

```
##
##  Box-Ljung test
##
## data:  res_d
## X-squared = 14.733, df = 11, p-value = 0.1951
```

```
# Mcleod-Li Test
Box.test((res_d)^2, lag = 14, type = c("Ljung-Box"), fitdf = 0) # p-value = 0.9478 PASS
```

```
##
##  Box-Ljung test
##
## data:  (res_d)^2
## X-squared = 6.6371, df = 14, p-value = 0.9478
```

```
# Yule-Walker Check
ar(res_d, aic = TRUE, order.max = NULL, method = c("yule-walker"))
```

```
##
## Call:
## ar(x = res_d, aic = TRUE, order.max = NULL, method = c("yule-walker"))
##
##
## Order selected 0  sigma^2 estimated as  0.03357
```

```
# Fitted residuals to white noise/AR(0) which is good
```

Histogram of residuals follows normal curve decently well, albeit there are some heavy tailed outliers.

Fails Shapiro-Wilk test (explained earlier why we expect this), but passes all other tests

Ultimately, go onto forecasting with Model C because it has the lower AICc and both models have the same amount of parameters and pretty much the same diagnostics.