Time Series Analysis of ‘MLB’ Google Search Popularity

Evan Mouchard

Abstract

In this paper, I analyzed a monthly time series of ‘MLB’ Google Search data. My goal was to create a good-fitting model for the years 2004-2019. I used Box-Jenkins methodology to guide the process. I split the time series into training and testing sets. The training data was then transformed and differenced to make an approximately normal and stationary time series. Autocorrelations were used to identify possible SARIMA models. I ran into a brief complication which caused me to re-difference the data and identify a new list of possible models. I fit various candidates using MLE estimation and compared their Akaike Information Criterion scores to narrow down the list. Various diagnostic tests were run to decide upon a final model. The model provided a fairly accurate forecast of the testing set from 2018 to 2019. After forecasting the test data, I briefly compared my model’s 2020 forecast with actual 2020 data and reflected on the results.

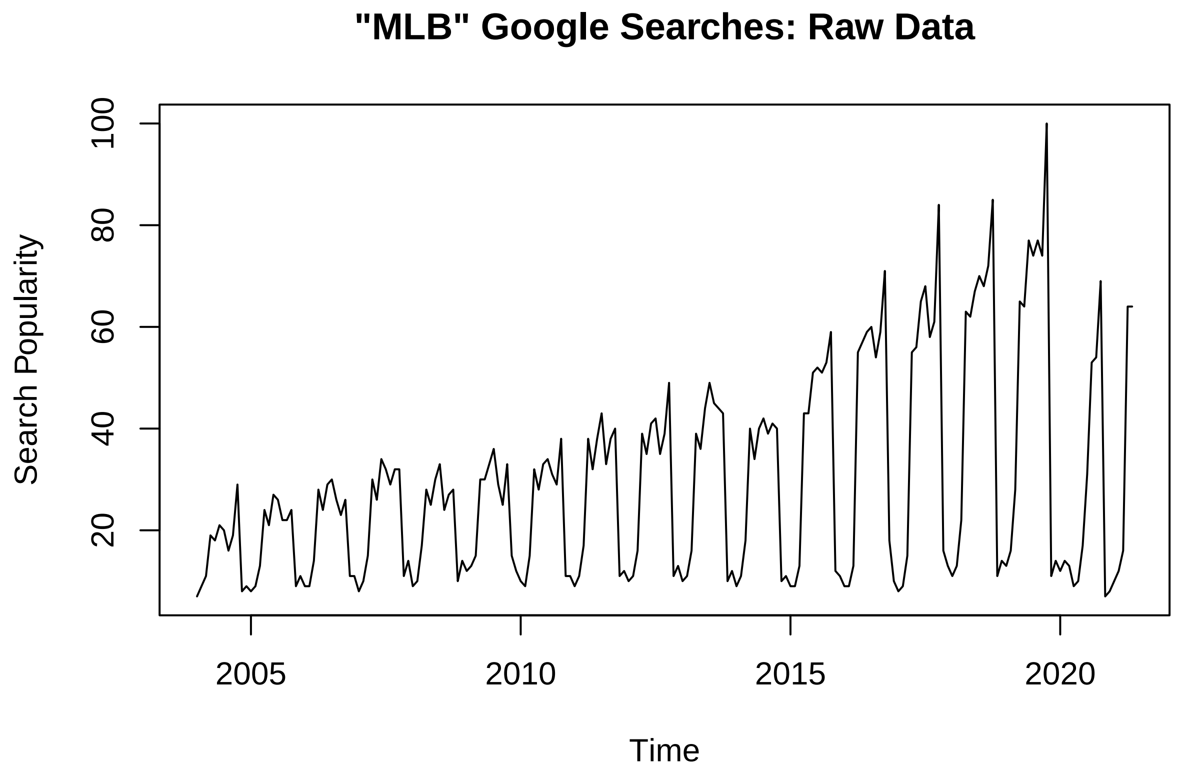
Introduction

‘MLB’ is an acronym for Major League Baseball, the largest professional baseball league in the United States and the world. Using Google Trends, I acquired a monthly record of Google searches containing the term ‘MLB’. The data measures the popularity of ‘MLB’ searches from January 2004 to May 2021, comprising 209 observations. The data are scaled from 0 to 100, such that ‘0’ is the minimum of the dataset and ‘100’ is the maximum. I chose this dataset because it provides interesting, useful information about the online popularity of Major League Baseball. Being able to analyze and forecast this data could be valuable to advertisers, MLB organizations, and sports fanatics such as myself.

The main goal of this project is to produce a good-fitting model that makes accurate forecasts about ‘MLB’ Google searches. I will apply Box-Jenkins methodology to produce a SARIMA model for the years 2004-2019. I will use 2004-2017 as training data and 2018-19 as testing data. I will then use this model to analyze data from 2020, an abnormal year for most major sports due to Covid-19.

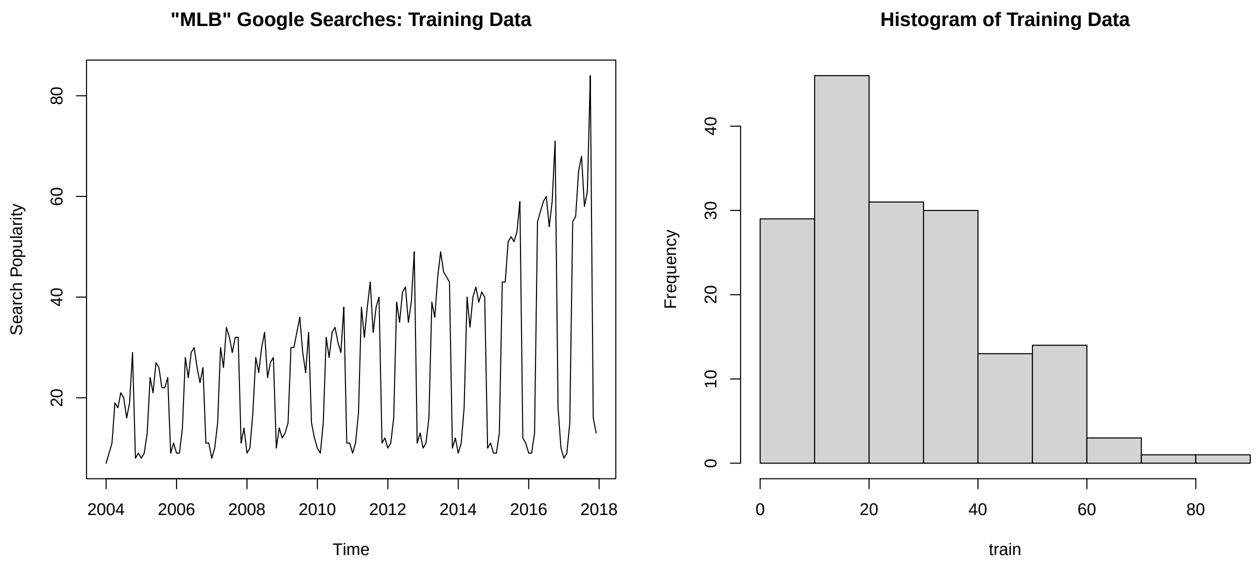
Part 1: Data Transformation and Differencing

```r
mlbdata <- read.csv('MLB.csv')
colnames(mlbdata) <- c("Month", "Searches")
MLB <- ts(mlbdata$Searches, frequency = 12, start = 2004)

train <- ts(MLB[1:168], frequency = 12, start = 2004) # training set: 1/2004-12/2017
test <- ts(MLB[169:192], frequency = 12, start = 2018) # test set: 1/2018-12/2019
traintest <- ts(MLB[1:192], frequency = 12, start = 2004) # combined: 1/2004-12/2019
since_covid <- ts(MLB[193:209], frequency = 12, start = 2020) # pandemic set 1/2020-5/2021

plot.ts(MLB, ylab = 'Search Popularity', main = '"MLB" Google Searches: Raw Data')
```



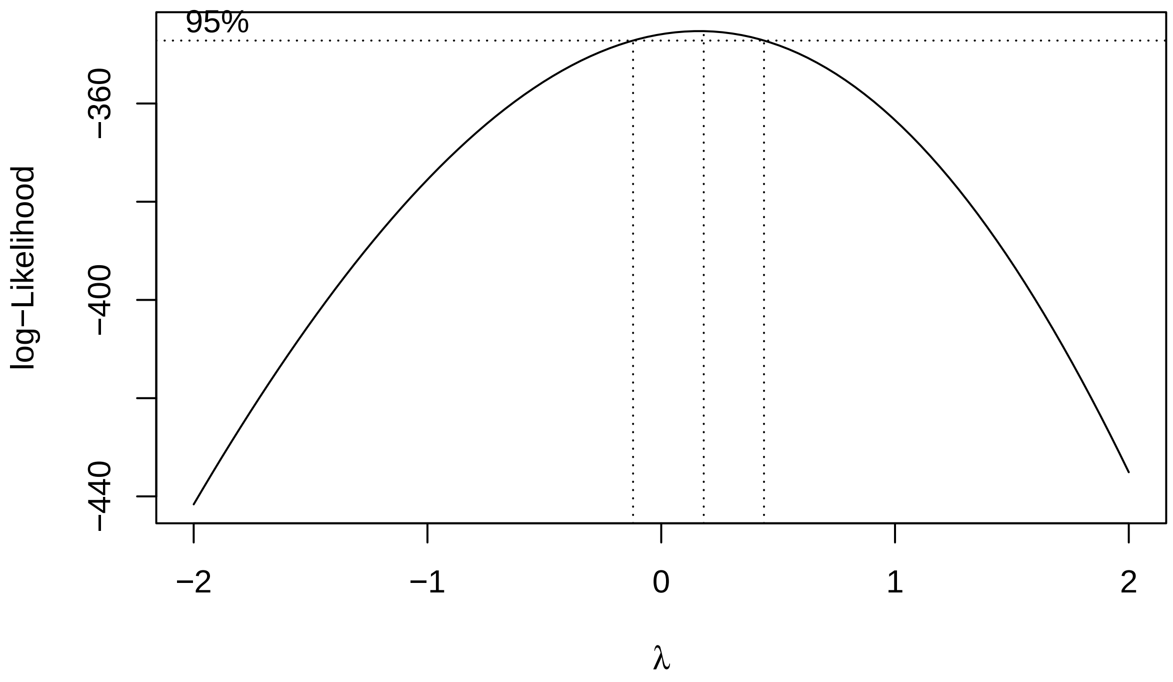
From 2004 to 2019, the data shows seasonality and increasing variance. The values seem to rise and fall in correspondence with the MLB season and off-season. In general, the regular season runs from late March/early April to late September/early October, followed by the playoffs which last until early November. The data for 2020 shows an uncharacteristic dip. Like many major sports, the MLB had a postponed and shortened 2020 season due to Covid-19. This could explain the abnormal behavior of the time series. For now, I will exclude this year from the training and testing sets. I will train the model on 2004-2017 data.

```r
par(mfrow = c(1, 2))
ts.plot(train, ylab = 'Search Popularity', main = '"MLB" Google Searches: Training Data')
hist(train, main = 'Histogram of Training Data')
```



The training data contains 168 observations. The data's increasing peak size and skewed histogram indicate non-constant variance. To stabilize the variance, I will use the Box-Cox function to find an optimal transformation.

```r
t = 1:length(train)
bcTransform = boxcox(train~t, interp = TRUE, plotit=TRUE)
```

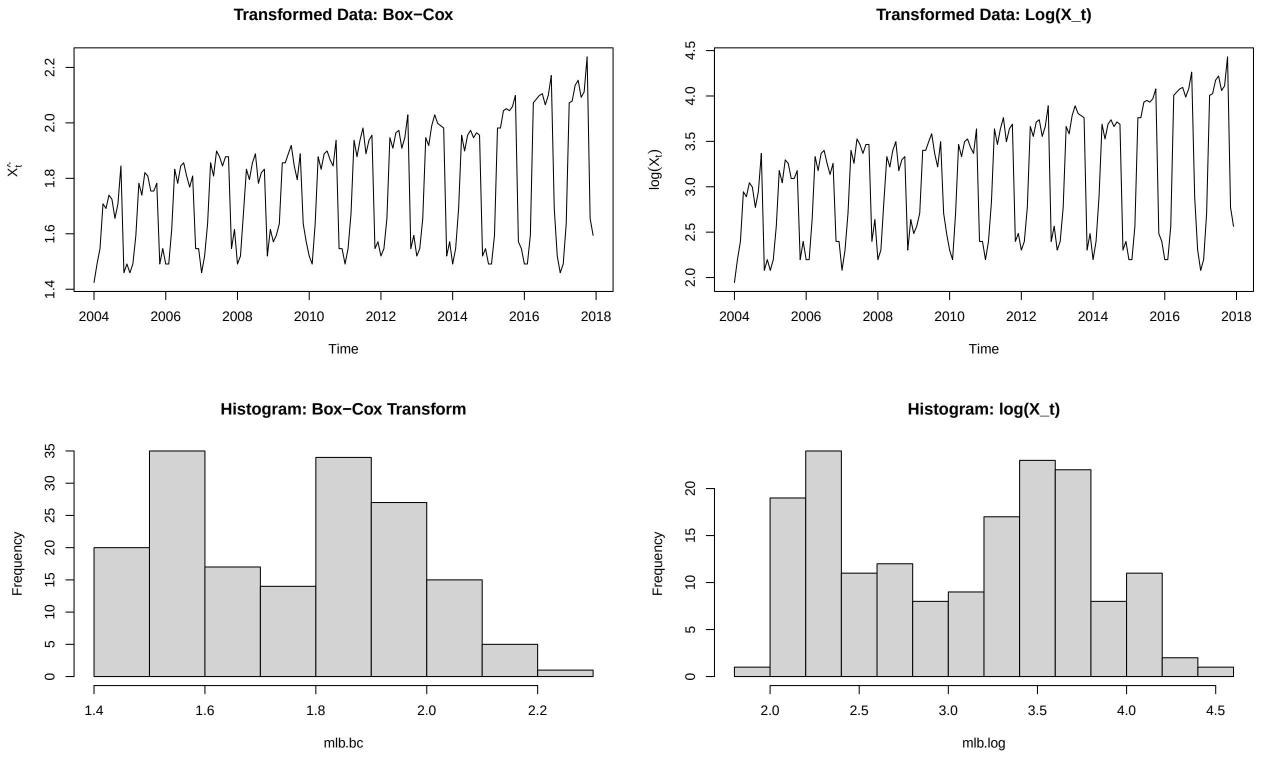


```r
lambda = bcTransform$x[which.max(bcTransform$y)]
mlb.bc <- train^lambda
mlb.log <- log(train)
lambda
```

```
## [1] 0.1818182
```

The Box-Cox function gives value $\lambda = 0.182$. Its 95% confidence interval contains $\lambda = 0$, which corresponds to the natural logarithmic transform. I will compare these two transformations and select the best candidate.

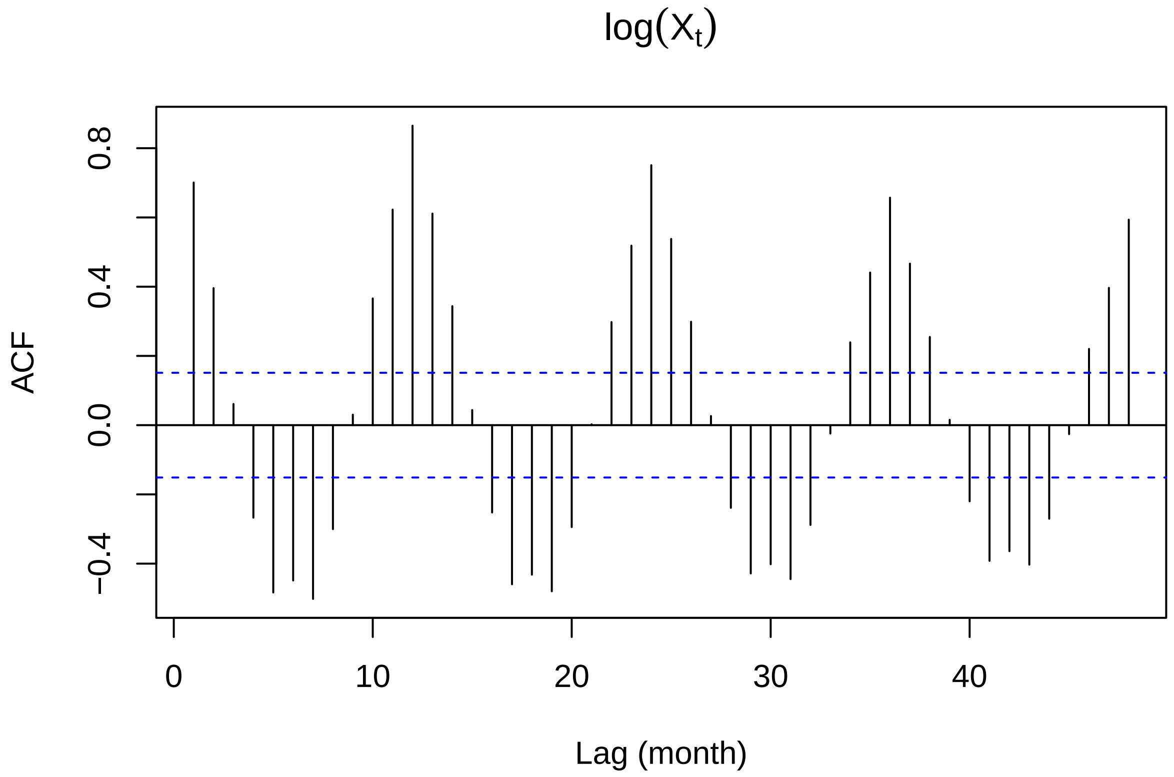
```r
par(mfrow = c(2, 2))
plot.ts(mlb.bc, ylab = expression(X[t]^lambda), main = 'Transformed Data: Box-Cox')
plot.ts(mlb.log, ylab = expression(log(X[t])), main = 'Transformed Data: Log(X_t)')
hist(mlb.bc, main = "Histogram: Box-Cox Transform")
hist(mlb.log, main = "Histogram: log(X_t)")
```



The two candidate transformations are quite similar. Both line plots show approximately constant variance. Both histograms have two, slightly asymmetric peaks. This asymmetry could be caused by seasonality, so I will ignore it for now. I will select the natural logarithm transform because it is more interpretable and more commonly used.

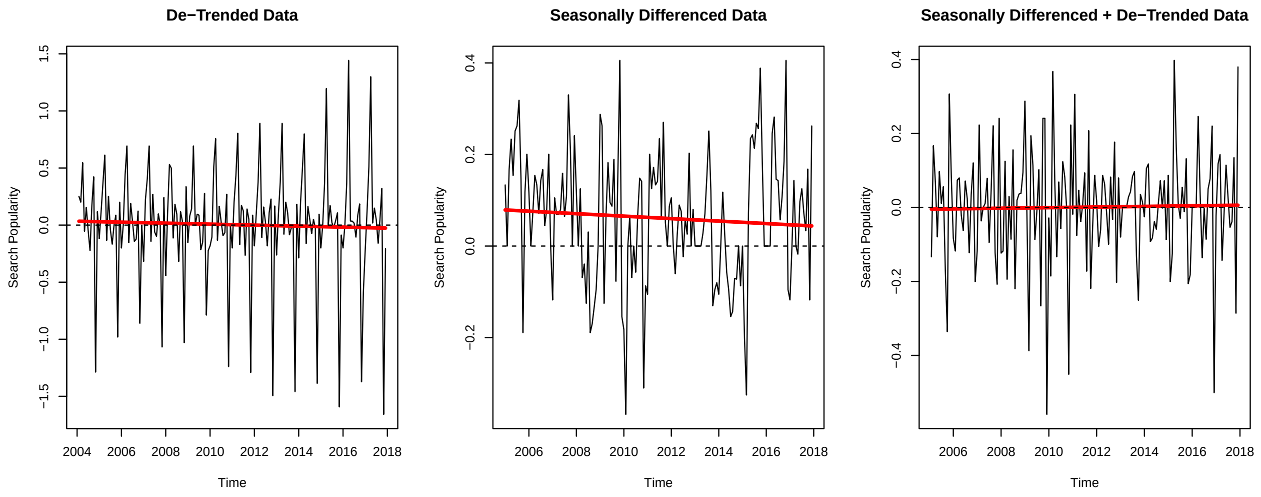
Now I can check the ACFs for seasonality.

```r
acf(ts(mlb.log, freq = 1), lag.max = 48, main = expression(log(X[t])), xlab = 'Lag (month)')
```



The ACFs show oscillating, connected peaks at multiples of 12, indicating seasonality at lag 12. This passes the sanity test: the MLB season has a regular pattern that repeats every 12 months. To remove seasonality, I will try differencing at lag 12. I will also try differencing at lag 1 to remove possible trend.

```r
mlb.dt <- diff(mlb.log, lag = 1)
mlb.ds <- diff(mlb.log, lag = 12)
mlb.dd <- diff(mlb.dt, lag = 12)

par(mfrow = c(1, 3))
ts.plot(mlb.dt, lm(mlb.dt~as.numeric(1:length(mlb.dt)))$fit,
        col = c('black', 'red'), lwd = c(1, 3),
        main = 'De-Trended Data', ylab = 'Search Popularity')
abline(h = 0, lty = 2)
ts.plot(mlb.ds, lm(mlb.ds~as.numeric(1:length(mlb.ds)))$fit,
        col = c('black', 'red'), lwd = c(1, 3),
        main = 'Seasonally Differenced Data', ylab = 'Search Popularity')
abline(h = 0, lty = 2)
ts.plot(mlb.dd, lm(mlb.dd~as.numeric(1:length(mlb.dd)))$fit,
        col = c('black', 'red'), lwd = c(1, 3),
        main = 'Seasonally Differenced + De-Trended Data', ylab = 'Search Popularity')
abline(h = 0, lty = 2)
```

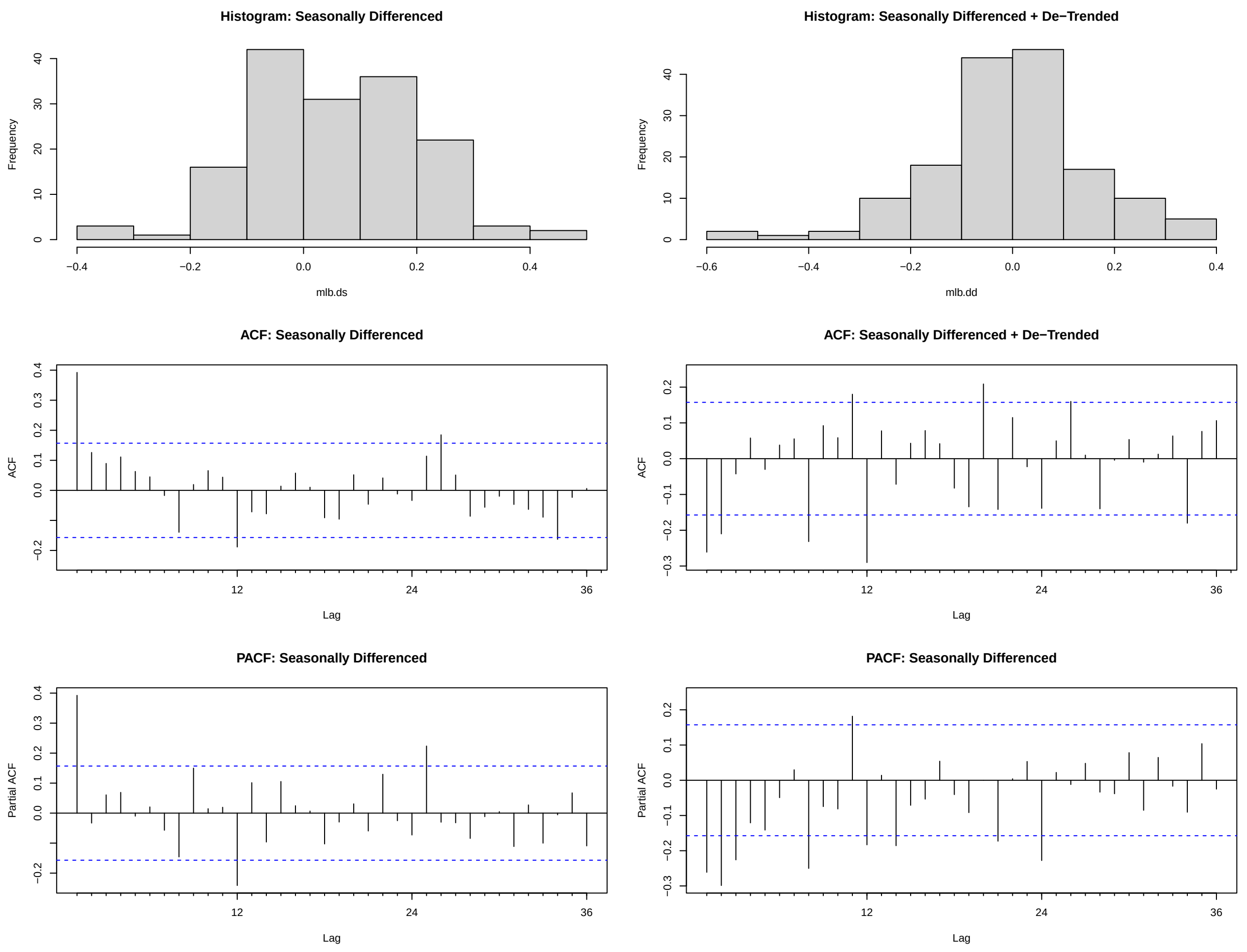


```
variances = data.frame(
  Variance = c(var(mlb.log), var(mlb.ds), var(mlb.dt), var(mlb.dd)),
  row.names = c('Not Differenced', 'Seasonally Differenced',
                'De-Trended', 'Seasonally Differenced + De-trended'))
variances
```

```
##                                       Variance
## Not Differenced                     0.41340647
## Seasonally Differenced              0.02002084
## De-Trended                          0.23946594
## Seasonally Differenced + De-trended 0.02418929
```

Seasonal differencing removes seasonality and results in the lowest variance: 0.02. The combination of seasonal differencing and de-trending gives a slightly higher variance, which is a sign of over-differencing, but it exhibits less of a trend and appears stationary about zero. To decide which differencing level to use, I will examine the normality and ACF/PACF's of both levels. (On the next page)

```
par(mfrow = c(3, 2))
hist(mlb.ds, main = 'Histogram: Seasonally Differenced')
hist(mlb.dd, main = 'Histogram: Seasonally Differenced + De-Trended')
Acf(mlb.ds, main = 'ACF: Seasonally Differenced', lag.max = 36)
Acf(mlb.dd, main = 'ACF: Seasonally Differenced + De-Trended', lag.max = 36)
Pacf(mlb.ds, main = 'PACF: Seasonally Differenced', lag.max = 36)
Pacf(mlb.dd, main = 'PACF: Seasonally Differenced', lag.max = 36)
```



```
shapiro.test(mlb.ds) #Shapiro test for seasonal differenced data
```

```
##
##  Shapiro-Wilk normality test
##
## data:  mlb.ds
## W = 0.99028, p-value = 0.3613
```

```
shapiro.test(mlb.dd) #Shapiro test for seasonal differenced + de-trended data
```

```
##
##  Shapiro-Wilk normality test
##
## data:  mlb.dd
## W = 0.97258, p-value = 0.003486
```

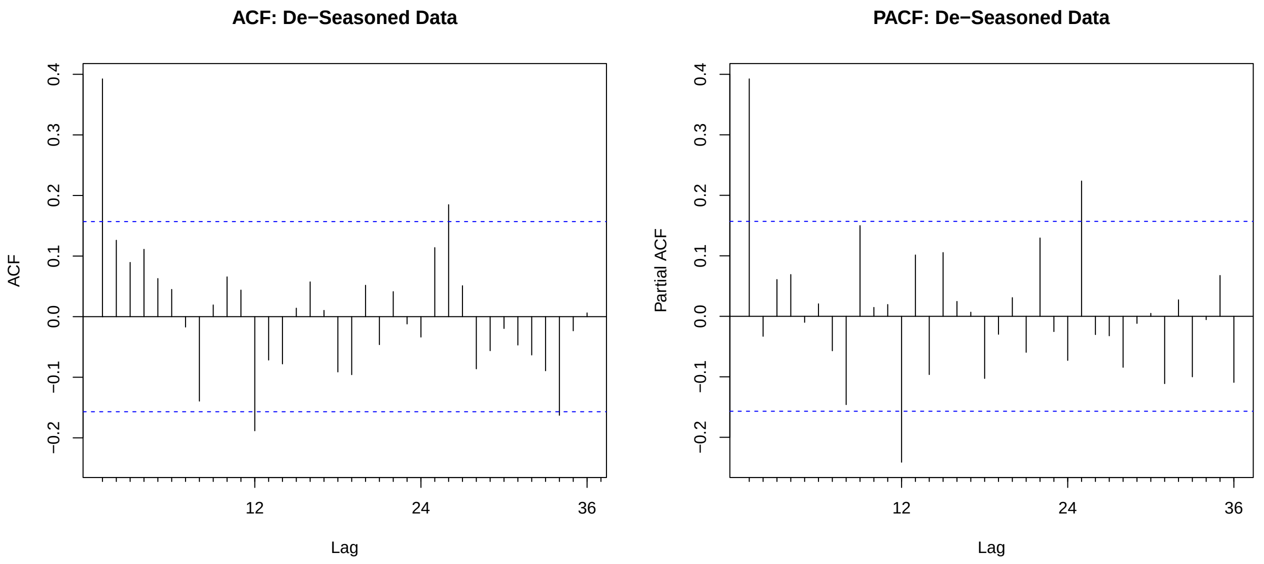
From the histograms and Shapiro-Wilk tests on the previous page, I can conclude that the seasonally differenced data is approximately normal, whereas the seasonally differenced + de-trended data has a more heavy-tailed distribution. Ideally, the transformed and differenced data should be normal and stationary. Unfortunately, neither of my choices fit perfectly: seasonal differencing results in a normal, possibly non-stationary distribution, while the combination of seasonal + regular differencing results in a stationary, but possibly not normal distribution. I will move forward with the seasonally differenced data. If I encounter unit roots or other problems with model fitting, I will switch to the de-trended data.

To summarize Part 1, I have applied the natural logarithmic transformation to the data and seasonally differenced at lag 12. I decided to not de-trend the data, but will revise this decision if I encounter problems with model fitting. As it stands, the model should have the form: SARIMA$(p, 0, q) \times (P, 1, Q)_{12}$.

Part 2: Model Identification

The proposed model is SARIMA$(p, 0, q) \times (P, 1, Q)_{12}$. In this section, I will identify the correct values for p, q, P, and Q. I will form a list of candidate models from the ACF/PACFs. Then, I will use MLE estimation to fit different candidate models and compare them using corrected Akaike Information Criterion (AICc). This should help narrow down the list to a few candidates. The model roots will be checked for stationarity and invertibility before moving on to diagnostic checking.

```r
par(mfrow = c(1, 2))
Acf(mlb.ds, main = 'ACF: De-Seasoned Data', lag.max = 36)
Pacf(mlb.ds, main = 'PACF: De-Seasoned Data', lag.max = 36)
```



Seasonally Differenced Data:

ACF spikes: 1, 12, 26
PACF spikes: 1, 12, and 25

The ACF and PACF have spikes at lag 1, indicating an MA(1) and/or AR(1) component. There might be an MA(2) expressed by the spike at lag 26, equal to 2s+2. The faint ACF spike at lag 12 indicates possible SMA(1). The PACF spike at lag 12 indicates SAR(1). The candidate models include SARIMA$(p, 0, q) \times (P, 1, Q)_{12}$ where p = 0 or 1; q = 0, 1, or 2; P = 0 or 1; and Q = 0 or 1. Checking all possible combinations for the lowest AICc:

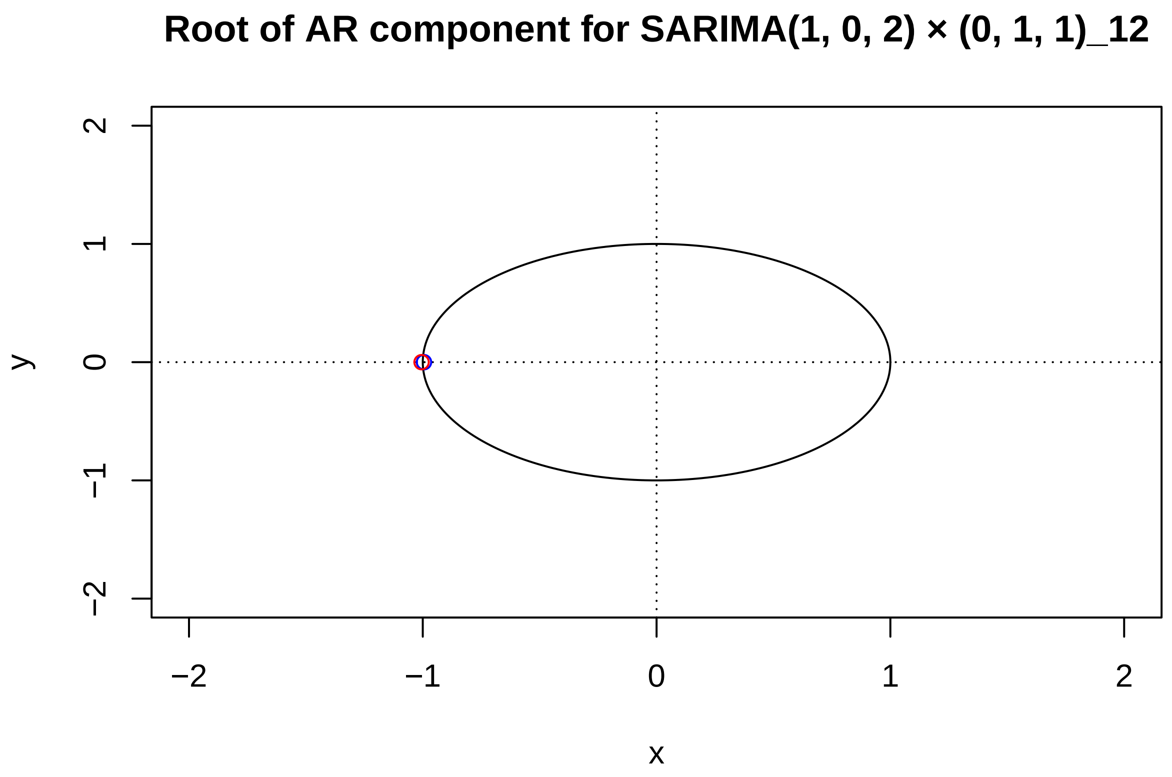
```
     p q P Q      AICc
[1,] 1 2 0 1 -196.3467
[2,] 1 2 1 1 -194.2433
[3,] 1 2 1 0 -192.9049

[See Appendix for code]
```

The candidate with the lowest AICc (-196.35) is SARIMA$(1, 0, 2) \times (0, 1, 1)_{12}$. Checking its fitted coefficients and roots:

```r
m1 <- arima(mlb.log, order = c(1, 0, 2), seasonal = list(order = c(0, 1, 1), s = 12))
m1$coef
```

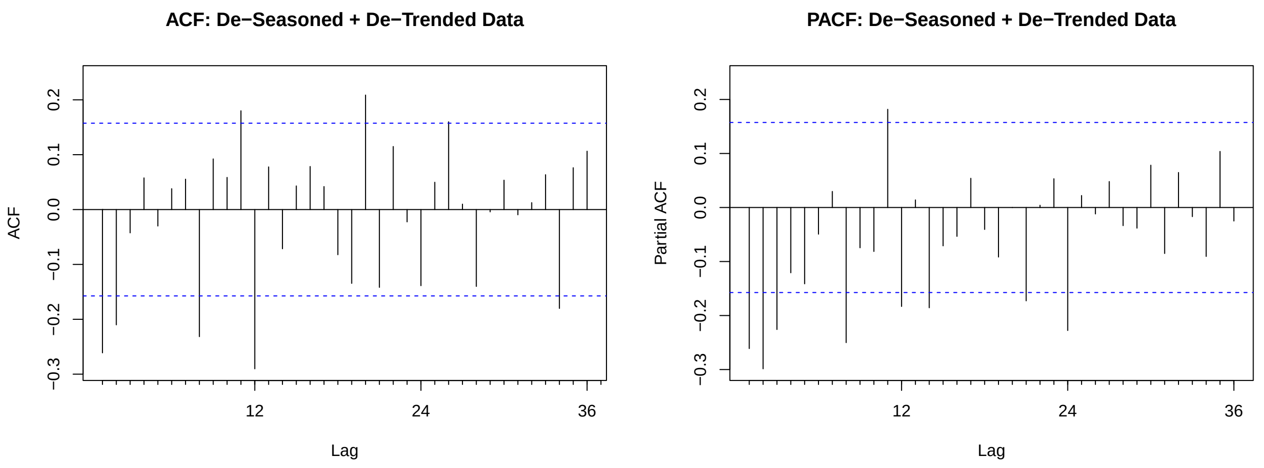
```
##        ar1        ma1        ma2       sma1
##  0.9948153 -0.4847581 -0.4042580 -0.3780563
```

```r
plot.roots(ar.roots = polyroot(c(1, 0.9948153)), ma.roots = NULL,
           main = 'Root of AR component for SARIMA(1, 0, 2) × (0, 1, 1)_12')
```



The coefficient ϕ_1 is very close to 1, indicating that the AR component has a unit root. As I suspected, this model is not stationary. I will alter my approach by differencing the data to eliminate trend. To find the best possible model, I will start again from scratch, forming a new list of candidate models by looking at the ACFs and PACFs of the seasonally differenced + de-trended data. (On the next page)

```
par(mfrow = c(1, 2))
Acf(mlb.dd, main = 'ACF: De-Seasoned + De-Trended Data', lag.max = 36)
Pacf(mlb.dd, main = 'PACF: De-Seasoned + De-Trended Data', lag.max = 36)
```



Seasonally Differenced + De-Trended Data:

ACF spikes: 1, 2, 8, 11, 12, 20
PACF spikes: 1, 2, 3, 8, 11, 14, 22, and 24

Based on the relative magnitudes of the spikes, the candidate non-seasonal components include: ARMA(0, 2), ARMA(2, 1), ARMA(0, 8), ARMA(0, 11). The seasonal AR component is 1. The seasonal MA component is either 0 or 2. The final candidate models include: SARIMA$(0, 1, 2) \times (P, 1, 1)_{12}$, SARIMA$(2, 1, 1) \times (P, 1, 1)_{12}$, SARIMA$(0, 1, 8) \times (P, 1, 1)_{12}$, SARIMA$(0, 1, 11) \times (P, 1, 1)$ where P is either 0 or 2. I will now fit all combinations and identify the models with the lowest AICc.

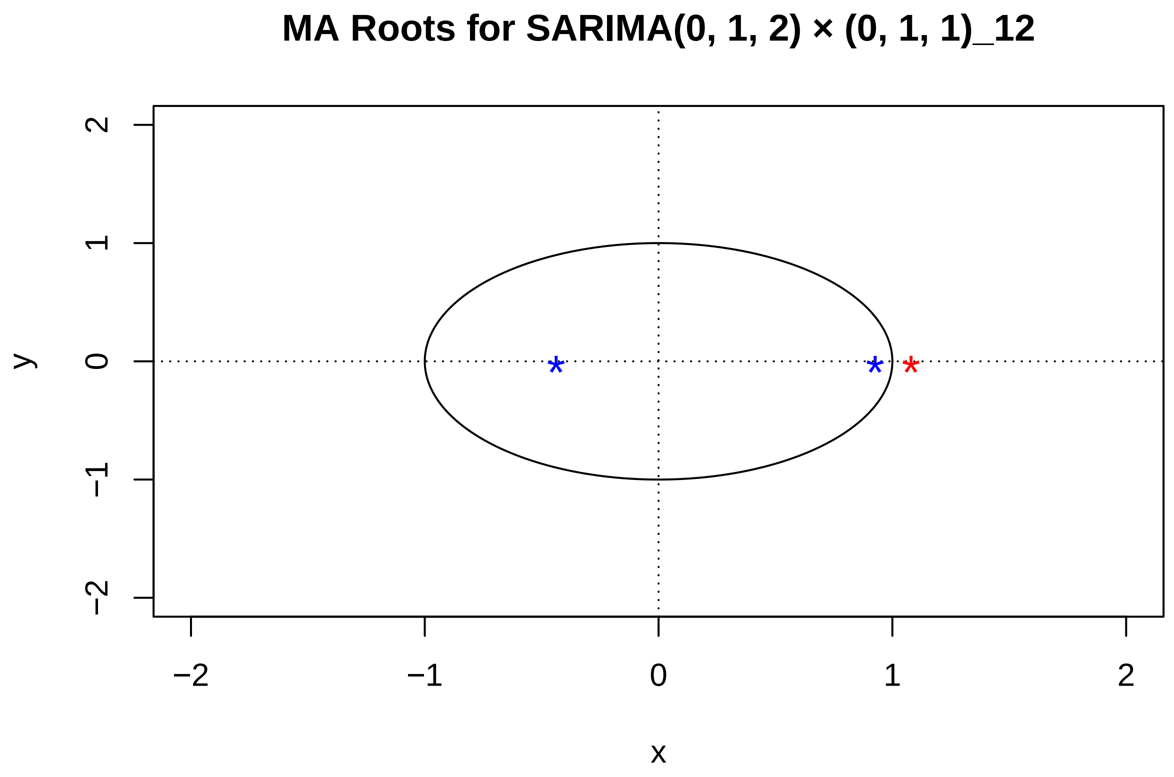
```
     p q P Q      AICc
[1,] 0 2 0 1 -196.1935
[2,] 2 1 0 1 -194.5789
[3,] 0 2 1 1 -194.1092

[See appendix for code]
```

The candidate with the lowest AICc (-196.19) – not including any fixed-component MA models – is SARIMA$(0, 1, 2) \times (0, 1, 1)_{12}$. I will check is fitted coefficients and roots:

```
m2 <- arima(mlb.log, order = c(0, 1, 2), seasonal = list(order = c(0, 1, 1), period = 12),
            method = 'ML')
m2$coef
```

```
##        ma1        ma2       sma1
## -0.4878719 -0.4059991 -0.3773245
```

```
plot.roots(ar.roots = NULL, ma.roots = polyroot(c(1, -0.4878719, -0.4059991)),
           main = 'MA Roots for SARIMA(0, 1, 2) × (0, 1, 1)_12')
```



```
polyroot(c(1, -0.4878719, -0.4059991))
```

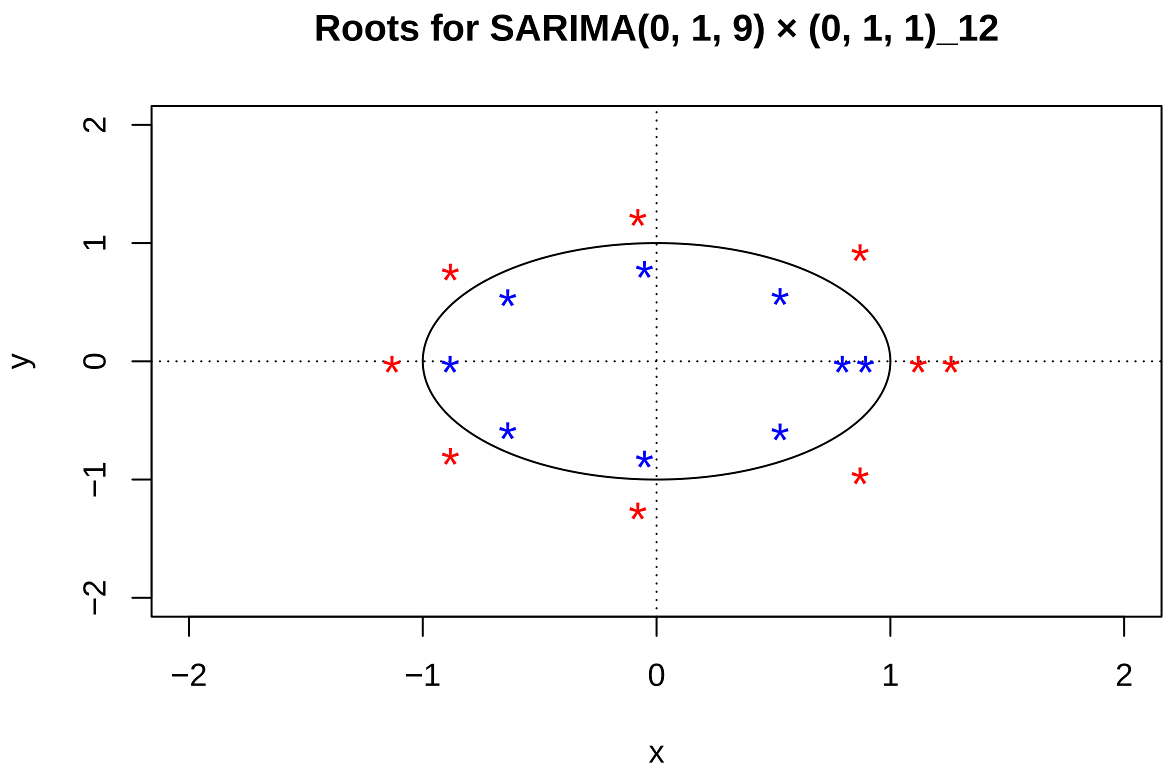
```
## [1]  1.079664-0i -2.281321+0i
```

All MA roots appear outside the unit circle, which means the model is invertible. It is also stationary because there is no AR component. This appears to be a good candidate for the final model. I should also check the MA(11) model, fixing the least significant coefficients to 0 to reduce the AICc:

```
#SEE APPENDIX FOR CODE THAT LEADS TO THIS MODEL
m3 <- arima(mlb.log, order = c(0, 1, 9), seasonal = list(order = c(0, 1, 1), period = 12),
             fixed = c(NA, NA, 0, 0, 0, 0, 0, NA, NA, NA)) # minimized, fixed-coef MA(9) model
m3$coef
```

```
##        ma1        ma2        ma3        ma4        ma5        ma6        ma7
## -0.4826146 -0.3874401  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000
##        ma8        ma9       sma1
## -0.1883567  0.1775041 -0.3576105
```

```
plot.roots(ar.roots = NULL,
           ma.roots = polyroot(c(1, -0.4826, -0.3874, 0, 0, 0, 0, 0, -0.1884, 0.1775)),
           main = 'Roots for SARIMA(0, 1, 9) × (0, 1, 1)_12')
```



```
AICc(m3)
```

```
[1] -197.2853
```

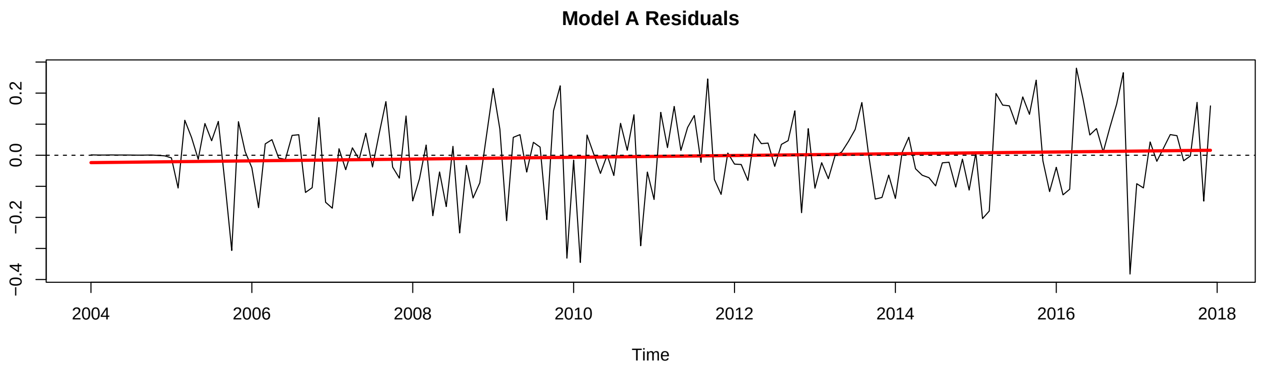
Surprisingly, I arrive at an MA(9) model with the lowest AICc of any model so far. Its roots are all outside the unit circle, indicating that it is invertible. This is a great candidate for the final model. Now I can proceed to diagnostic checking with the two candidate models: $\text{SARIMA}(0,1,2) \times (0,1,1)_{12}$ and $\text{SARIMA}(0,1,9) \times (0,1,1)_{12}$

Part 3: Diagnostic Checking

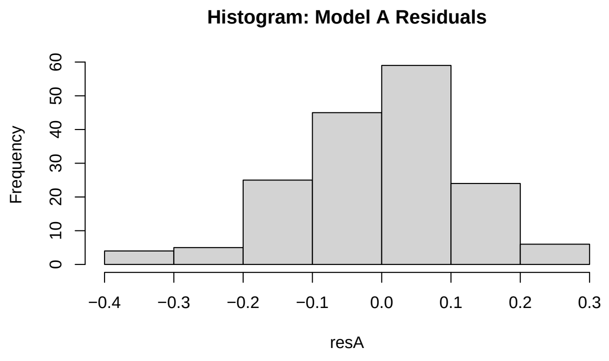
In this section, I will analyze the residuals of the candidate models, **Model A: SARIMA$(0,1,2)\times(0,1,1)_{12}$** and **Model B: SARIMA$(0,1,9)\times(0,1,1)_{12}$**.
Ideally, one or more of the models' residuals should resemble Gaussian white noise by having zero autocorrelations and passing each diagnostic test. I will select the model with the least variance and closest resemblance to Gaussian white noise. If both models perform equally, I will select the simpler model (Model A) by principle of parsimony.

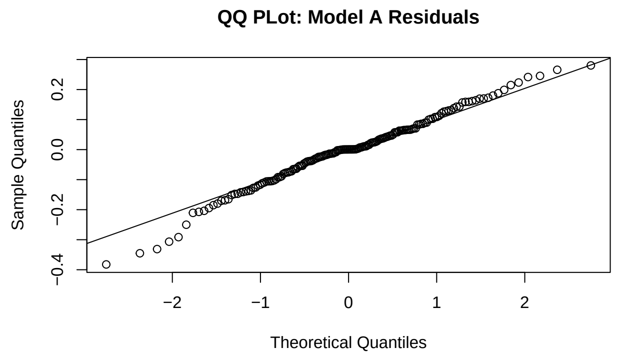
Model A:

```
resA <- m2$residuals
ts.plot(resA, main = 'Model A Residuals', lm(resA~as.numeric(1:length(resA)))$fit,
        col = c('black', 'red'), lwd = c(1, 3))
abline(h = 0, lty = 2)
```



```
par(mfrow = c(1, 2))
hist(resA, main = 'Histogram: Model A Residuals')
qqnorm(resA, main = 'QQ PLot: Model A Residuals')
qqline(resA)
```





```
shapiro.test(resA[13:168])
```

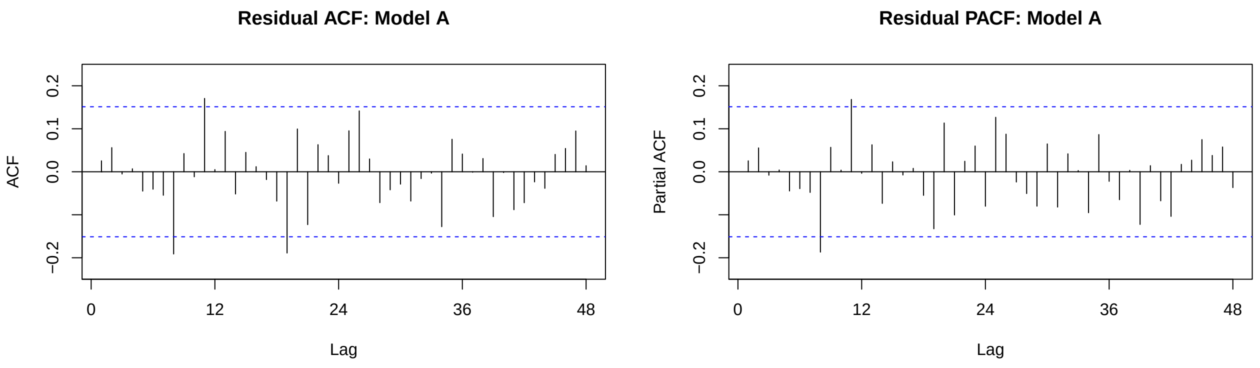
```
##
##  Shapiro-Wilk normality test
##
## data:  resA[13:168]
## W = 0.9893, p-value = 0.2828
```

```r
c(mean = mean(resA), variance = var(resA))
```

```
##         mean     variance
## -0.003893954  0.014187864
```

The residuals for Model A are approximately Gaussian with -0.0039 mean and 0.01419 variance. It passes the Shapiro-Wilks test for normality with p value 0.2828.

```r
par(mfrow = c(1, 2))
Acf(resA, main = 'Residual ACF: Model A', lag.max = 48)
Pacf(resA, main = 'Residual PACF: Model A', lag.max = 48)
```



The ACFs show small spikes at lags 8, 11, and 19. The PACFs show small spikes at lags 8 and 11. The spikes are small, so I still consider this a good model. However, it suggests that the residuals do not perfectly resemble white noise. The other model will be preferred if it shows zero spikes. Before checking Model B, I will run diagnostic tests on Model A.

```r
#lag = h = sqrt(n) = 13 (approx)
Box.test(resA, lag = 13, type = c("Box-Pierce"), fitdf= 2) # fitdf = p + q = 2
```

```
##
##  Box-Pierce test
##
## data:  resA
## X-squared = 14.608, df = 11, p-value = 0.2012
```

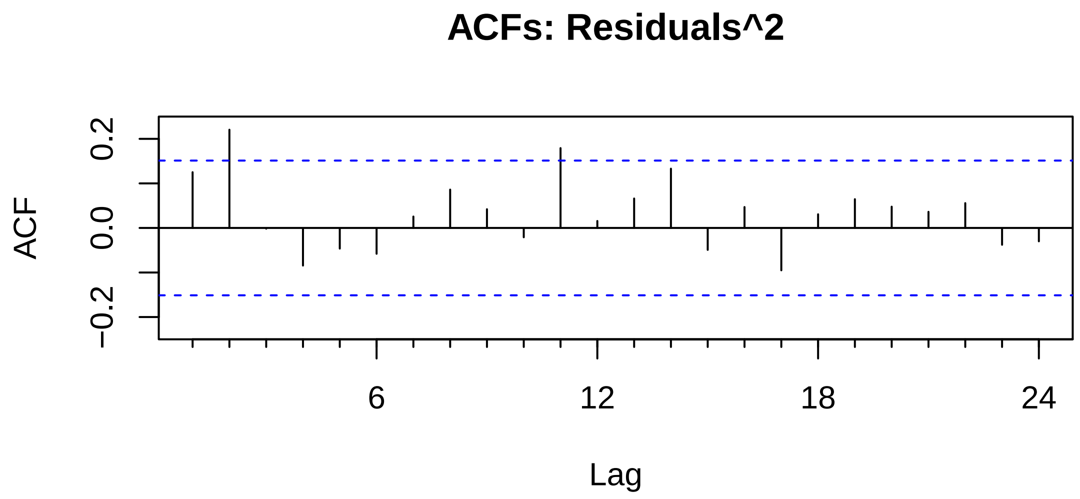
```r
Box.test(resA, lag = 13, type = c("Ljung-Box"), fitdf = 2) # fitdf = p + q = 2
```

```
##
##  Box-Ljung test
##
## data:  resA
## X-squared = 15.634, df = 11, p-value = 0.1553
```

```r
Box.test((resA)^2, lag = 13, type = c("Ljung-Box"), fitdf= 0 ) # fitdf = 0
```

```
##
##  Box-Ljung test
##
## data:  (resA)^2
## X-squared = 21.788, df = 13, p-value = 0.05873
```

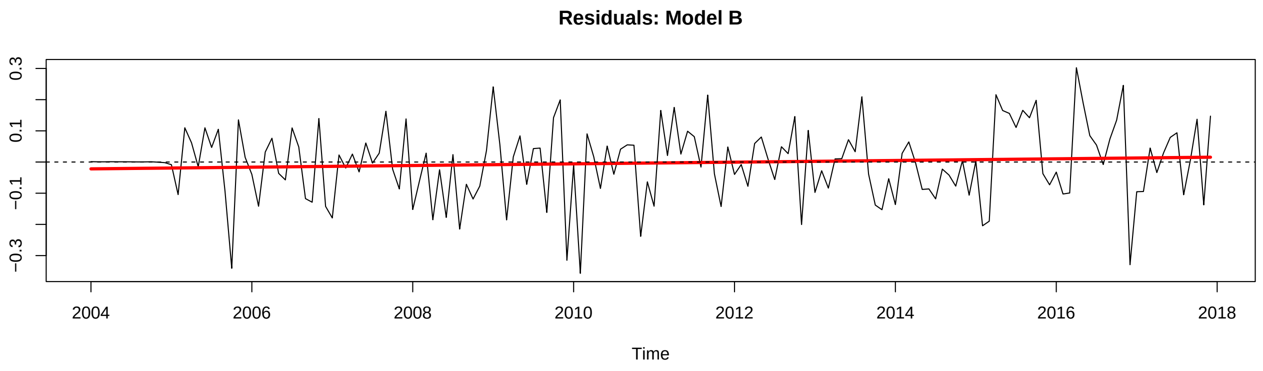
```r
Acf(resA^2, main = 'ACFs: Residuals^2')
```



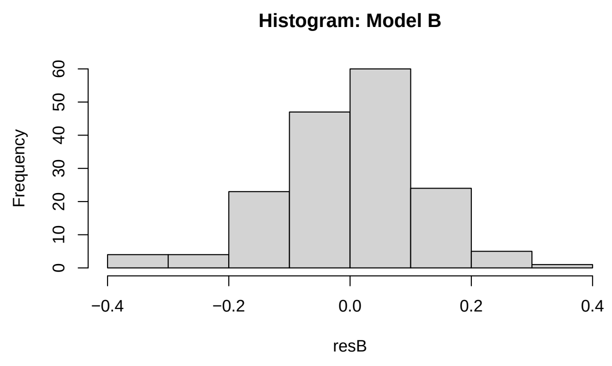
The residuals for Model A pass the Box-Pierce and Ljung-Box tests. They barely pass the Mcleod-Li test (p = 0.059) indicating that there might be nonlinear dependence. The ACFs of the squared residuals show a spike at lag 2. Overall, these are decent results since the residuals passed all diagnostic tests, although the p-value of the McLeod-Li test shows that it might have non-linear dependence

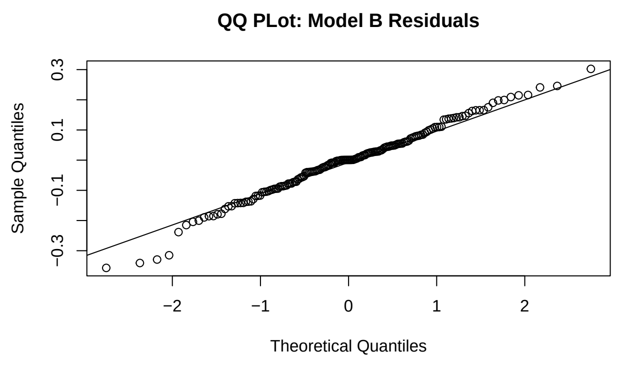
Model B

```r
resB <- m3$residuals
ts.plot(resB, main = 'Residuals: Model B', lm(resB~as.numeric(1:length(resB)))$fit,
        col = c('black', 'red'), lwd = c(1, 3))
abline(h = 0, lty = 2)
```



```r
par(mfrow = c(1, 2))
hist(resB, main = 'Histogram: Model B')
qqnorm(resB, main = 'QQ PLot: Model B Residuals')
qqline(resB)
```





```
shapiro.test(resB[13:168])
```

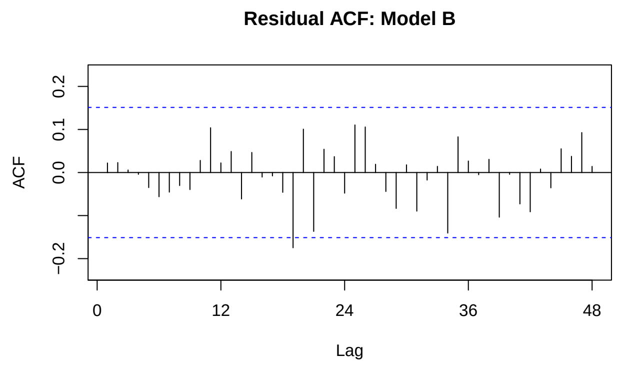
```
##
##  Shapiro-Wilk normality test
##
## data:  resB[13:168]
## W = 0.99081, p-value = 0.4099
```

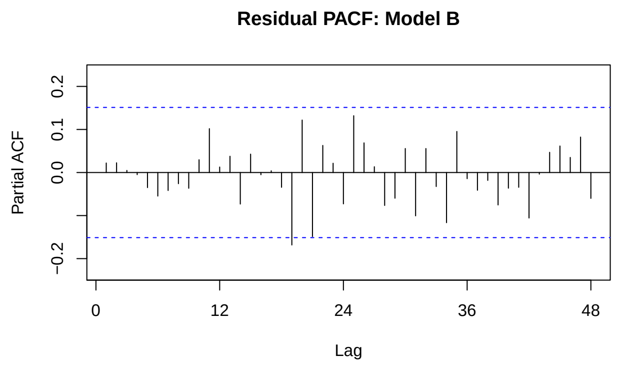
```
c(mean = mean(resB), variance = var(resB))
```

```
##         mean     variance
## -0.003250594  0.013618890
```

The residuals for Model B are approximately Gaussian with -0.0033 mean and 0.01362 variance (both slightly smaller than Model A). They pass the Shapiro-Wilks test for normality with p value 0.4099.

```
par(mfrow = c(1, 2))
Acf(resB, main = 'Residual ACF: Model B', lag.max = 48)
Pacf(resB, main = 'Residual PACF: Model B', lag.max = 48)
```





The ACF/PACFs for Model B residuals look better than Model A, with essentially no autocorrelations. There is a very small spike at lag 19, but this may be ignored due to the high lag (Bartlett's formula implies wider confidence intervals at high lags, not expressed in the R plot).

```
#lag = h = sqrt(n) = 13 (approx)
Box.test(resB, lag = 13, type = c("Box-Pierce"), fitdf= 9) # fitdf = p + q = 9
```

```
##
##  Box-Pierce test
##
```

```
## data:  resB
## X-squared = 4.1133, df = 4, p-value = 0.3909
```

```
Box.test(resB, lag = 13, type = c("Ljung-Box"), fitdf = 9) # fitdf = p + q = 2
```

```
##
##  Box-Ljung test
##
## data:  resB
## X-squared = 4.4062, df = 4, p-value = 0.3538
```

```
Box.test((resB)^2, lag = 13, type = c("Ljung-Box"), fitdf= 0 ) # fitdf = 0
```

```
##
##  Box-Ljung test
##
## data:  (resB)^2
## X-squared = 19.156, df = 13, p-value = 0.1183
```

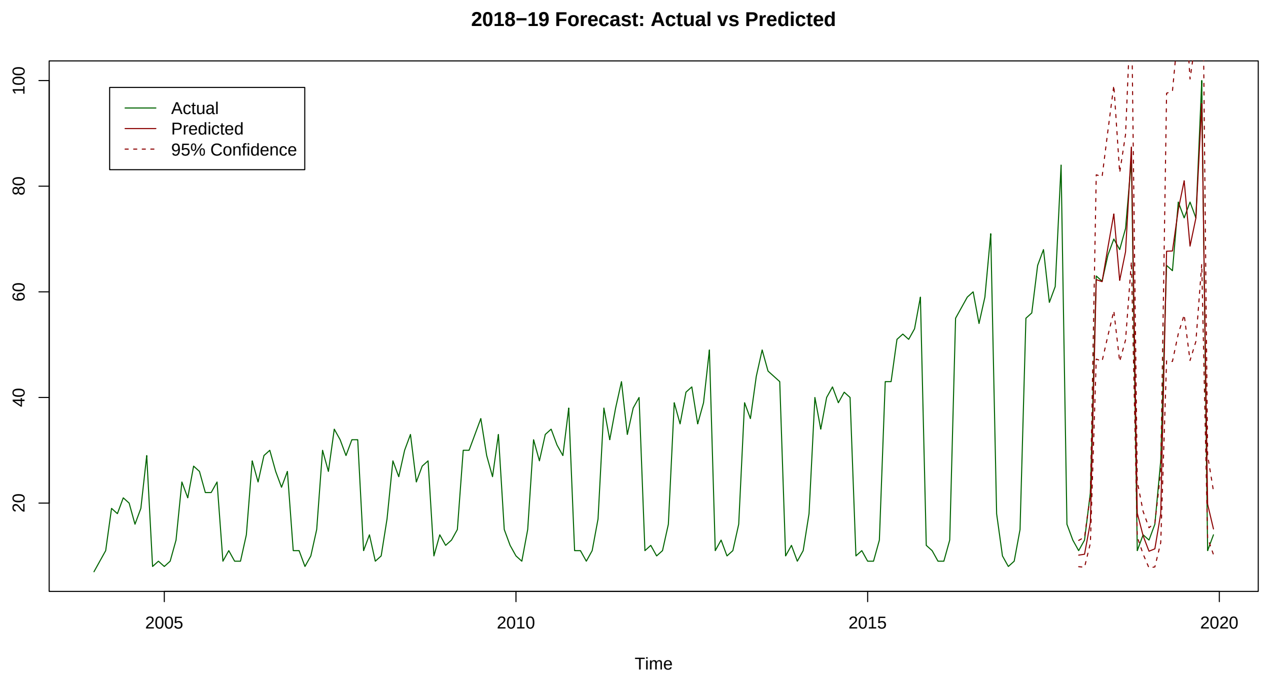
Model B passes all diagnostic checks, indicating that it resembles white noise. In spite of having more degrees of freedom, Model B has higher p-values than Model A on each test. It passes Mcleod-Li test for non-linear dependence more convincingly than Model B, with a p-value of 0.1183.

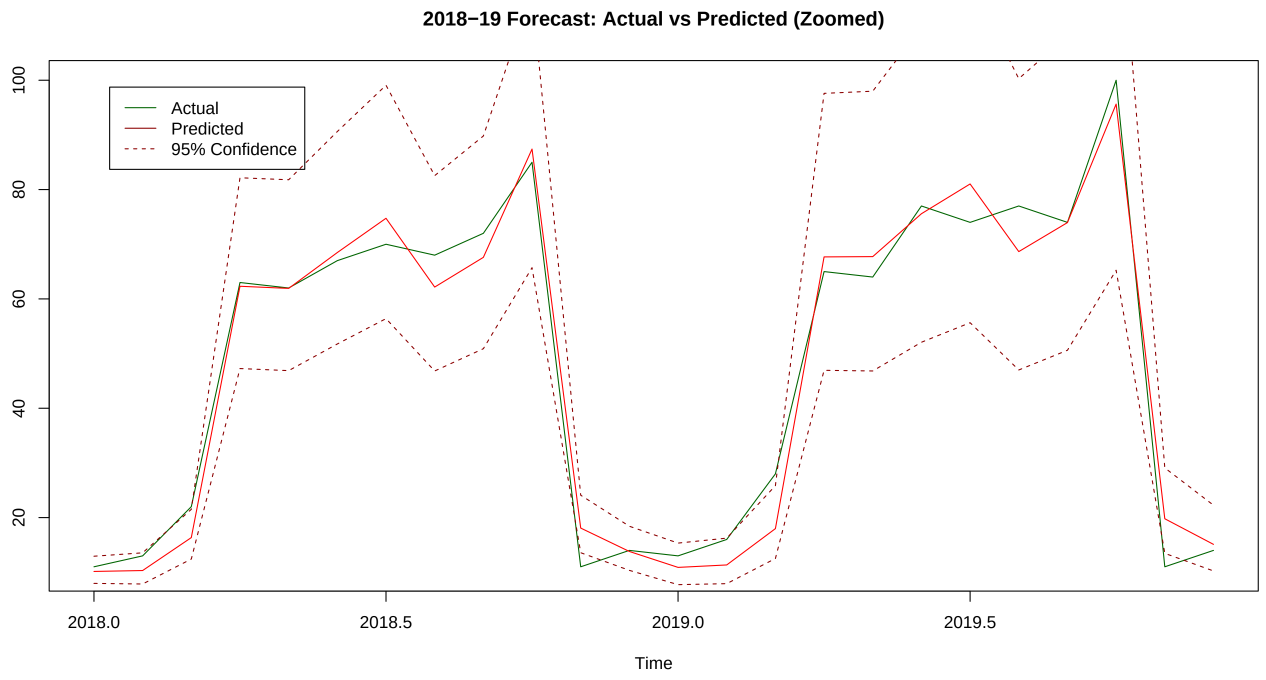
Based on the checks for normality and diagnostic Portmanteau tests for white noise, I will select Model B. It performs better than Model A on the tests, with uncorrelated residuals that resemble Gaussian white noise. It also has the lowest AICc (-197.28) of any model. Therefore, the **final model is $\mathbf{SARIMA(0,1,9)\times(0,1,1)_{12}}$**. It can be written mathematically as:

$$\nabla_1\nabla_{12}\ln(X_t) = (1-0.4826B)*(1+0.3875B^2)*(1-0.1884B^8)*(1+0.1175B^9)*(1-0.3756B^{12})*Z_t$$

Part 4: Forecasting

I will produce a two-year forecast for 2018-2019 and compare with the actual data from those years. The original data was transformed logarithmically, so I will perform an exponential transformation to my predictions so that they might resemble the original data.

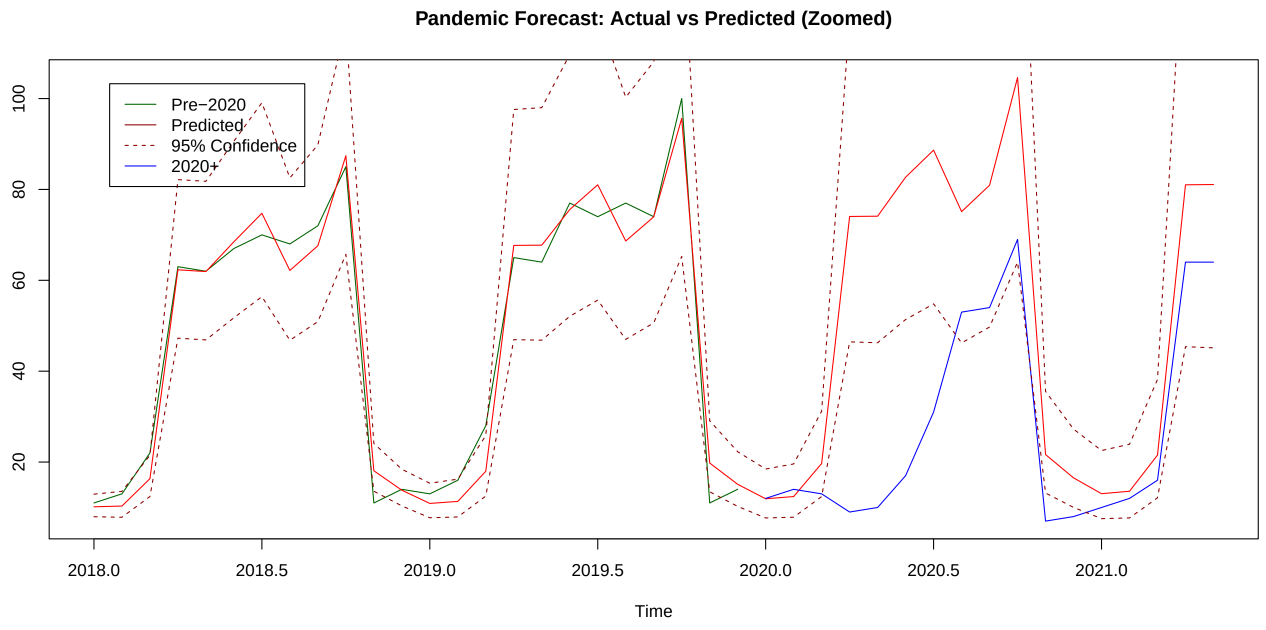




[See Appendix for code]

The 2018-19 forecast is fairly accurate (shown on the previous page). The overall height and shape of each peak is well-estimated, with most points falling near the center of the confidence interval. The forecast is less accurate around the valley of the seasonal periods, with more points falling near the edge of the confidence interval. Due to the exponential transformation applied to the predictions, the confidence interval is smaller around low predictions and larger around high predictions. Observations for November 2018 and November 2019 are slightly outside of the confidence interval. They are both exceptionally low values which mark the first month of each MLB off-season.

I have done my best to achieve the main goal of this project, to create a good-fitting model that produces accurate forecasts about 'MLB' Google Searches. Now, I will compare my model's 2020 forecast with actual 2020 data.



[See Appendix for code]
Clearly, the model does not adequately fit the time series for 2020. I speculate that this is due to the many changes in the world brought on by Covid-19. Quantifying and interpreting the causes of this exact shift is beyond the scope of this paper. Still, the shift is interesting to observe. Most points fall outside or near the edge of the 95% confidence interval. There remains a seasonal peak in the data, but it is less wide and tall than the other seasonal peaks. It will be interesting to see if this time series returns to its normal behavior or if 2020 marked the start of a long-term shift. Monthly observations for 1/2021-5/2021 indicate that it may be returning to 'normal' levels, with values mostly inside the confidence interval, but it is too early to tell if the data can still be correctly modeled by this SARIMA model.

Conclusion

I did my best to achieve the goal of creating a good-fitting model than can produce accurate 'MLB' Google Search forecasts. Using Box-Jenkins methodology, the chosen model was SARIMA$(0, 1, 9) \times (0, 1, 1)_{12}$. Written mathematically with fitted coefficients:

$$\nabla_1 \nabla_{12} \ln(X_t) = (1 - 0.4826B) * (1 + 0.3875B^2) * (1 - 0.1884B^8) * (1 + 0.1175B^9) * (1 - 0.3756B^{12}) * Z_t$$

The model is stationary and invertible. It requires a natural logarithmic transformation to the raw data. It should be noted that after transforming and differencing, the training data was not strictly Gaussian. It was stationary and symmetric, but its distribution had slightly heavier tails than a Gaussian distribution. Still, the model performed quite well at fitting and forecasting. Its residuals resembled Gaussian white noise. The forecasted data generally resembled the test data. The year 2020 was excluded from the model due to abnormal shifts possibly related to Covid-19. In the future, it will be interesting to see how the time series behaves in the wake of this strange period.

References

Miller, Ryan & Schwarz, Harrison & Talke, Ismael. (2017). Forecasting Sports Popularity: Application of Time Series Analysis. Academic Journal of Interdisciplinary Studies. 6. 10.1515/ajis-2017-0009.

"Explore: 'MLB'." Google Trends, Google, May 2021,
trends.google.com/trends/explore?date=all&geo=US&q=%2Fm%2F09p14.

Appendix

```r
setwd('~/Documents/Spring2021/PSTAT174')
knitr::opts_chunk$set(echo=T,
                      eval=T,
                      cache=T,
                      results='markup',
                      message=F,
                      warning=F,
                      fig.align='center')
library(MASS)
library(forecast)
library(astsa)
library(qpcR)
library(tseries)
source("~/Documents/Spring2021/PSTAT174/plot.roots.R")
source("~/Documents/Spring2021/PSTAT174/spec.arma.R")
mlbdata <- read.csv('MLB.csv')
colnames(mlbdata) <- c("Month", "Searches")
MLB <- ts(mlbdata$Searches, frequency = 12, start = 2004)

train <- ts(MLB[1:168], frequency = 12, start = 2004) # training set: 1/2004-12/2017
test <- ts(MLB[169:192], frequency = 12, start = 2018) # test set: 1/2018-12/2019
traintest <- ts(MLB[1:192], frequency = 12, start = 2004) # combined: 1/2004-12/2019
since_covid <- ts(MLB[193:209], frequency = 12, start = 2020) # pandemic set 1/2020-5/2021

plot.ts(MLB, ylab = 'Search Popularity', main = '"MLB" Google Searches: Raw Data')
par(mfrow = c(1, 2))
ts.plot(train, ylab = 'Search Popularity', main = '"MLB" Google Searches: Training Data')
hist(train, main = 'Histogram of Training Data')
```

```r
t = 1:length(train)
bcTransform = boxcox(train~t, interp = TRUE, plotit=TRUE)
lambda = bcTransform$x[which.max(bcTransform$y)]
mlb.bc <- train^lambda
mlb.log <- log(train)
lambda
par(mfrow = c(2, 2))
plot.ts(mlb.bc, ylab = expression(X[t]^lambda), main = 'Transformed Data: Box-Cox')
plot.ts(mlb.log, ylab = expression(log(X[t])), main = 'Transformed Data: Log(X_t)')
hist(mlb.bc, main = "Histogram: Box-Cox Transform")
hist(mlb.log, main = "Histogram: log(X_t)")
acf(ts(mlb.log, freq = 1), lag.max = 48, main = expression(log(X[t])), xlab = 'Lag (month)')
mlb.dt <- diff(mlb.log, lag = 1)
mlb.ds <- diff(mlb.log, lag = 12)
mlb.dd <- diff(mlb.dt, lag = 12)

par(mfrow = c(1, 3))
ts.plot(mlb.dt, lm(mlb.dt~as.numeric(1:length(mlb.dt)))$fit,
        col = c('black', 'red'), lwd = c(1, 3),
        main = 'De-Trended Data', ylab = 'Search Popularity')
abline(h = 0, lty = 2)
ts.plot(mlb.ds, lm(mlb.ds~as.numeric(1:length(mlb.ds)))$fit,
        col = c('black', 'red'), lwd = c(1, 3),
        main = 'Seasonally Differenced Data', ylab = 'Search Popularity')
abline(h = 0, lty = 2)
ts.plot(mlb.dd, lm(mlb.dd~as.numeric(1:length(mlb.dd)))$fit,
        col = c('black', 'red'), lwd = c(1, 3),
        main = 'Seasonally Differenced + De-Trended Data', ylab = 'Search Popularity')
abline(h = 0, lty = 2)

variances = data.frame(
  Variance = c(var(mlb.log), var(mlb.ds), var(mlb.dt), var(mlb.dd)),
  row.names = c('Not Differenced', 'Seasonally Differenced',
                'De-Trended', 'Seasonally Differenced + De-trended'))
variances
par(mfrow = c(3, 2))
hist(mlb.ds, main = 'Histogram: Seasonally Differenced')
hist(mlb.dd, main = 'Histogram: Seasonally Differenced + De-Trended')
Acf(mlb.ds, main = 'ACF: Seasonally Differenced', lag.max = 36)
Acf(mlb.dd, main = 'ACF: Seasonally Differenced + De-Trended', lag.max = 36)
Pacf(mlb.ds, main = 'PACF: Seasonally Differenced', lag.max = 36)
Pacf(mlb.dd, main = 'PACF: Seasonally Differenced', lag.max = 36)

shapiro.test(mlb.ds) #Shapiro test for seasonal differenced data
shapiro.test(mlb.dd) #Shapiro test for seasonal differenced + de-trended data
par(mfrow = c(1, 2))
Acf(mlb.ds, main = 'ACF: De-Seasoned Data', lag.max = 36)
Pacf(mlb.ds, main = 'PACF: De-Seasoned Data', lag.max = 36)
# CODE FOR MATRIX OF CANDIDATE AICc's AT d = 0
aiccs_d0 = matrix(NA, nr = 24, nc = 5)
colnames(aiccs_d0) = c('p', 'q', 'P', 'Q', 'AICc')
i = 0
for (p in 0:1) {
  for (q in 0:2) {
    for (P in 0:1) {
      for (Q in 0:1) {
        aiccs_d0[i+1, 1] = p
        aiccs_d0[i+1, 2] = q
        aiccs_d0[i+1, 3] = P
        aiccs_d0[i+1, 4] = Q
        aiccs_d0[i+1, 5] = AICc(arima(mlb.log,
                                      order = c(p, 0, q),
                                      seasonal = list(order = c(P, 1, Q)),
                                      method = 'ML'))
        i= i+1
      }
    }
  }
}
aiccs_d0[order(aiccs_d0[,5])[1:3],]
m1 <- arima(mlb.log, order = c(1, 0, 2), seasonal = list(order = c(0, 1, 1), s = 12))
m1$coef
plot.roots(ar.roots = polyroot(c(1, 0.9948153)), ma.roots = NULL,
           main = 'Root of AR component for SARIMA(1, 0, 2) × (0, 1, 1)_12')
par(mfrow = c(1, 2))
Acf(mlb.dd, main = 'ACF: De-Seasoned + De-Trended Data', lag.max = 36)
Pacf(mlb.dd, main = 'PACF: De-Seasoned + De-Trended Data', lag.max = 36)
# CODE FOR FINDING MATRIX OF AICc's AT d=1
aiccs_d1 = matrix(NA, nr = 24, nc = 5)
colnames(aiccs_d1) = c('p', 'q', 'P', 'Q', 'AICc')
i = 0
for (p in c(0, 2)) {
  for (q in c(1, 2)) {
    for (P in 0:2) {
      aiccs_d1[i+1, 1] = p
      aiccs_d1[i+1, 2] = q
      aiccs_d1[i+1, 3] = P
      aiccs_d1[i+1, 4] = 1
      aiccs_d1[i+1, 5] = AICc(arima(mlb.log,
                                    order = c(p, 1, q),
                                    seasonal = list(order = c(P, 1, Q), period = 12),
                                    method = 'ML'))
      i= i+1
    }
  }
}
aiccs_d1[order(aiccs_d1[,5])[1:3],]
m2 <- arima(mlb.log, order = c(0, 1, 2), seasonal = list(order = c(0, 1, 1), period = 12),
            method = 'ML')
m2$coef
plot.roots(ar.roots = NULL, ma.roots = polyroot(c(1, -0.4878719, -0.4059991)),
           main = 'MA Roots for SARIMA(0, 1, 2) × (0, 1, 1)_12')
polyroot(c(1, -0.4878719, -0.4059991))
#MODEL-FITTING THAT LEADS TO MA(9) MODEL
arima(mlb.log, order = c(0, 1, 11), seasonal = list(order = c(2, 1, 1), period = 12))
arima(mlb.log, order = c(0, 1, 11), seasonal = list(order = c(0, 1, 1), period = 12),
             fixed = c(NA, NA, 0, 0, 0, 0, 0, NA, NA, NA, NA, NA))
#fix smallest coefficients to zero, remove SAR
arima(mlb.log, order = c(0, 1, 9), seasonal = list(order = c(0, 1, 1), period = 12),
             fixed = c(NA, NA, 0, 0, 0, 0, 0, NA, NA, NA)) #reduce size of q
#SEE APPENDIX FOR CODE THAT LEADS TO THIS MODEL
m3 <- arima(mlb.log, order = c(0, 1, 9), seasonal = list(order = c(0, 1, 1), period = 12),
             fixed = c(NA, NA, 0, 0, 0, 0, 0, NA, NA, NA)) # minimized, fixed-coef MA(9) model
m3$coef
plot.roots(ar.roots = NULL,
           ma.roots = polyroot(c(1, -0.4826, -0.3874, 0, 0, 0, 0, 0, -0.1884, 0.1775)),
           main = 'Roots for SARIMA(0, 1, 9) × (0, 1, 1)_12')
AICc(m3)
resA <- m2$residuals
ts.plot(resA, main = 'Model A Residuals', lm(resA~as.numeric(1:length(resA)))$fit,
        col = c('black', 'red'), lwd = c(1, 3))
abline(h = 0, lty = 2)
par(mfrow = c(1, 2))
hist(resA, main = 'Histogram: Model A Residuals')
qqnorm(resA, main = 'QQ PLot: Model A Residuals')
qqline(resA)
shapiro.test(resA[13:168])
c(mean = mean(resA), variance = var(resA))
par(mfrow = c(1, 2))
Acf(resA, main = 'Residual ACF: Model A', lag.max = 48)
Pacf(resA, main = 'Residual PACF: Model A', lag.max = 48)
#lag = h = sqrt(n) = 13 (approx)
Box.test(resA, lag = 13, type = c("Box-Pierce"), fitdf= 2) # fitdf = p + q = 2
Box.test(resA, lag = 13, type = c("Ljung-Box"), fitdf = 2) # fitdf = p + q = 2
Box.test((resA)^2, lag = 13, type = c("Ljung-Box"), fitdf= 0 ) # fitdf = 0
Acf(resA^2, main = 'ACFs: Residuals^2')
resB <- m3$residuals
ts.plot(resB, main = 'Residuals: Model B', lm(resB~as.numeric(1:length(resB)))$fit,
        col = c('black', 'red'), lwd = c(1, 3))
abline(h = 0, lty = 2)
par(mfrow = c(1, 2))
hist(resB, main = 'Histogram: Model B')
qqnorm(resB, main = 'QQ PLot: Model B Residuals')
qqline(resB)
shapiro.test(resB[13:168])
c(mean = mean(resB), variance = var(resB))
par(mfrow = c(1, 2))
Acf(resB, main = 'Residual ACF: Model B', lag.max = 48)
Pacf(resB, main = 'Residual PACF: Model B', lag.max = 48)
#lag = h = sqrt(n) = 13 (approx)
Box.test(resB, lag = 13, type = c("Box-Pierce"), fitdf= 9) # fitdf = p + q = 9
Box.test(resB, lag = 13, type = c("Ljung-Box"), fitdf = 9) # fitdf = p + q = 2
Box.test((resB)^2, lag = 13, type = c("Ljung-Box"), fitdf= 0 ) # fitdf = 0

#FORECAST PLOTS
model = m3
preds <- predict(model, n.ahead = 24) # forecast 24 months beginning Jan. 2018
upper <- preds$pred + 2*preds$se # upper bound
lower <- preds$pred - 2*preds$se #lower bound
```

```r
ts.plot(traintest, exp(preds$pred),
        col = c('darkgreen', 'darkred'), main = "2018-19 Forecast: Actual vs Predicted")
lines(exp(upper), col='darkred', lty=2)
lines(exp(lower), col='darkred', lty=2)
legend("topleft", c("Actual", "Predicted", "95% Confidence"),
       col = c('darkgreen', 'darkred', 'darkred'), lty=c(1, 1, 2), inset = .05)

ts.plot(test, exp(preds$pred),
        col = c('darkgreen', 'red'), main = "2018-19 Forecast: Actual vs Predicted (Zoomed)")
lines(exp(upper), col='darkred', lty=2)
lines(exp(lower), col='darkred', lty=2)
legend("topleft", c("Actual", "Predicted", "95% Confidence"),
       col = c('darkgreen', 'darkred', 'darkred'), lty=c(1, 1, 2), inset = .05)

# PANDEMIC FORECAST PLOT
preds2 <- predict(model, n.ahead = 41)
upper2 <- preds2$pred + 2*preds2$se
lower2 <- preds2$pred - 2*preds2$se
zoomed <- ts(MLB[121:192], start = 2014, freq = 12)

ts.plot(test, exp(preds2$pred), since_covid,
        col = c('darkgreen', 'red', 'blue'), main = "Pandemic Forecast: Actual vs Predicted (Zoomed)")
lines(exp(upper2), col='darkred', lty=2)
lines(exp(lower2), col='darkred', lty=2)
legend("topleft", c("Pre-2020", "Predicted", "95% Confidence", '2020+'),
       col = c('darkgreen', 'darkred', 'darkred', 'blue'), lty=c(1, 1, 2, 1), inset = .05)
```