PSTAT 174 Final Project: Forecasting U.S. Unemployment Rate

Brandon Lee

12/11/2020

Abstract

In this project, we will be examining the U.S. unemployment rates during the past two decades in a time series dataset created by the official U.S. Bureau of Labor Statistics. Our ultimate goal regarding this dataset is to find a seasonal autoregressive integrated moving average (SARIMA) model that is able to successfully forecast future unemployment rates based on the preexisting data that has already been gathered over the years. To accomplish this task, we will split the dataset into a training set and a test set, use transformations and differencing to make our data stationary, conduct ACF/PACF analysis to find candidate models, compare AIC(C) values to identify the "best" model out of the possible candidates, and perform diagnostic checks to determine whether or not the final model is indeed suitable for forecasting use. After employing all of these statistical techniques, we successfully determined a SARIMA model that is able to forecast future U.S. unemployment rates to a satisfactory extent, based on having successful forecasting results on the test set.

Introduction

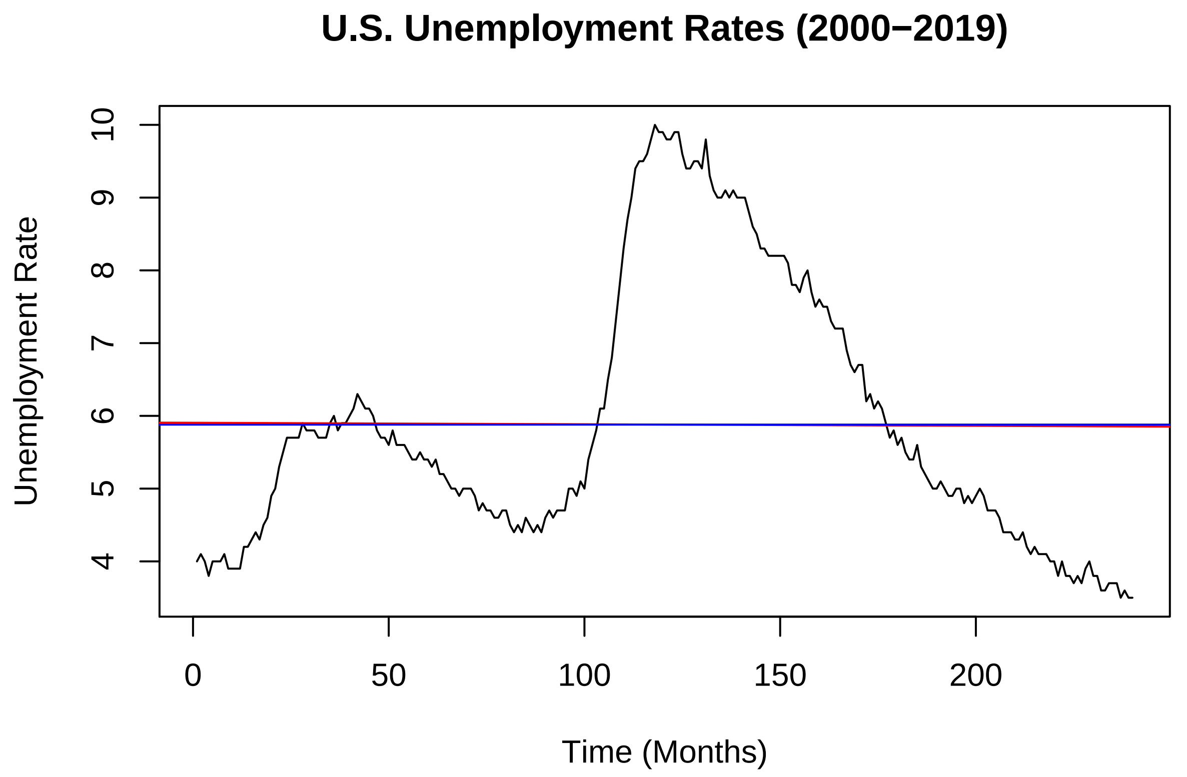
This time series dataset from the official U.S. Bureau of Labor Statistics website lists the unemployment rate of those within the U.S. population that are 16 years and over from the years 2000 to 2019. The U.S. Bureau of Labor Statistics records the unemployment rate in percents from a "Current Population Survey" every month. This U.S. unemployment rate data is interesting due to the fact that the U.S. has suffered from a high unemployment rate in the past, especially during the deep recession that occurred after the housing bubble collapse in 2008, and, as such, there is always a potential for another unemployment crisis occurring, even during times of seemingly stable employment rates. Since the possibility of unemployment rate spikes exists, we are interested in determining whether or not a SARIMA model can be used to adequately forecast future unemployment rates in the hope of the model being usable to predict an onset of high unemployment in the near future. In R Studio, we successfully achieve this goal of forecasting U.S. unemployment rates within a 95 percent prediction interval by splitting the data into a training and test set, using transformations and differencing to make our initially non-stationary data stationary, using ACF/PACF analysis and AIC(C) values to determine the "best" model, and using diagnostic checks to determine whether or not the final model is in fact suitable for forecasting. By the end of this project, we conclude that our final model $SARIMA(5,1,3) \times (3,1,1)_{12}$ for the logarithm transformation of the original data can be used to satisfactorily forecast U.S. unemployment rates.

- U.S. unemployment rate data, from January 2000 to January 2019
- Monthly unemployment rate in percents $\{U_t, t = 1, 2, ..., 240\}$
- Data Reference: U.S. Bureau of Labor Statistics (https://data.bls.gov/cgi-bin/surveymost?bls)

Sections

Plot and analyze time series

We begin by plotting the U.S. unemployment rate time series with both a trend line and a constant mean line to examine the main features of the graph:

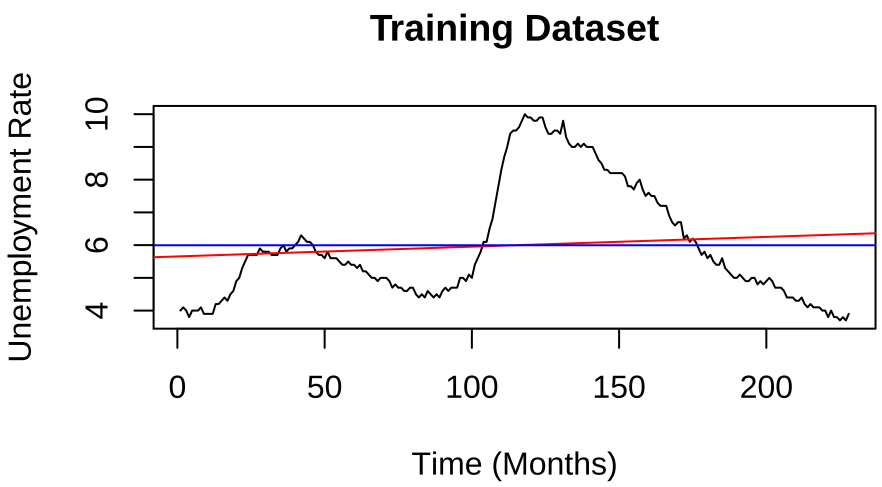


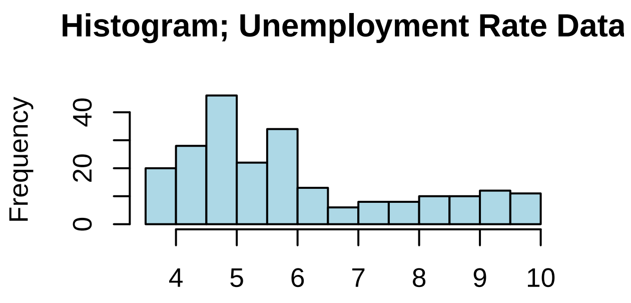
From the graph, we immediately see that the data does not seem to have constant variance or constant mean. We also see that, based on the trend line (in red) being almost perfectly horizontal and parallel to the constant mean line (in blue), there does not seem to be any particular trend in the original dataset. However, we notice that there was a very sharp increase in unemployment rate starting around Month 100 to until the unemployment rate hit approximately 10 percent, where it then started to decrease somewhat steadily. This behavior is unusual based on the previous peak of unemployment rate only reaching to approximately 6.5 percent. Lastly, there also seems to be a seasonal component to this time series due to there being some evidence of recurring increasing and decreasing pattern throughout most of the dataset. Thus, keeping seasonality in mind, we conclude that the dataset does not seem to be stationary.

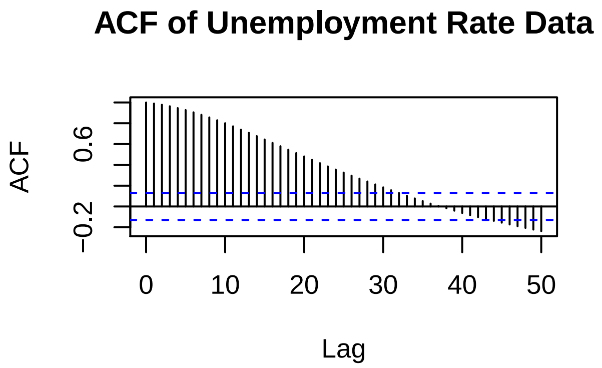
Next, for model validation purposes, we split the data. We create a training dataset (which we will refer to as U_t) that will be used to build a model using the first 228 unemployment rate values, leaving the remaining 12 values for a test dataset that will be used to test the forecasting of our models:

```
#create training dataset
ur.train = ur[c(1:228)]
#create test dataset
ur.test = ur[c(229:240)]
```

We then plot the training set data, the training set histogram, and the training set ACFs to confirm whether or not the training dataset is stationary:



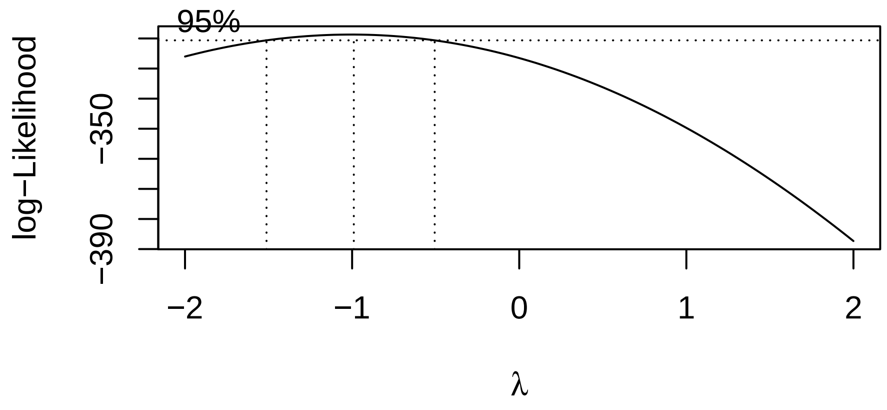


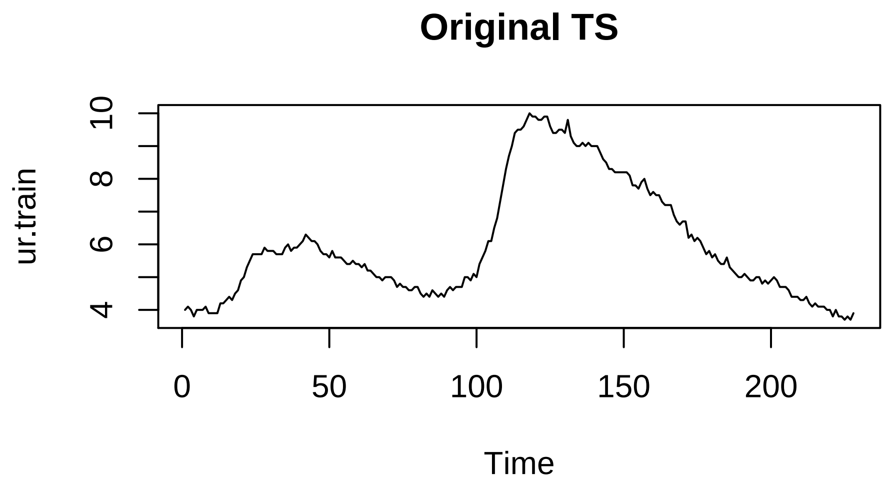


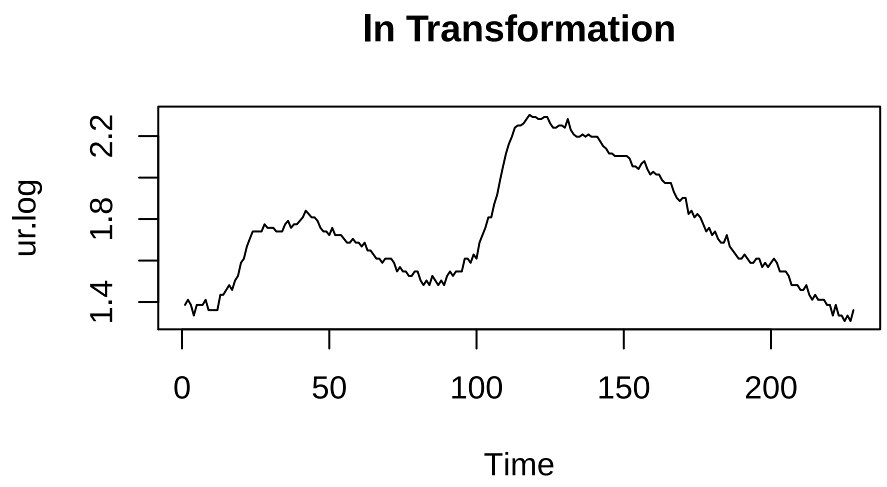
From the graph of the training dataset, we see that the only notable difference between the graph of the original dataset and the training set is that there does seem to be some evidence of a slightly increasing trend in the training dataset, which was not present in the original. From the histogram we notice that that the data is heavily right-skewed, indicating that the data is not Gaussian. Lastly, from the ACF of the unemployment rate data, it is easily determined that the data is non-stationary due its very large and slowly decaying ACFs. Using these three pieces of evidence, we can reasonably conclude that our data is non-stationary. Thus, our next step is to attempt to make the data stationary by using transformations and differencing to stabilize the variance and to remove trend and seasonality.

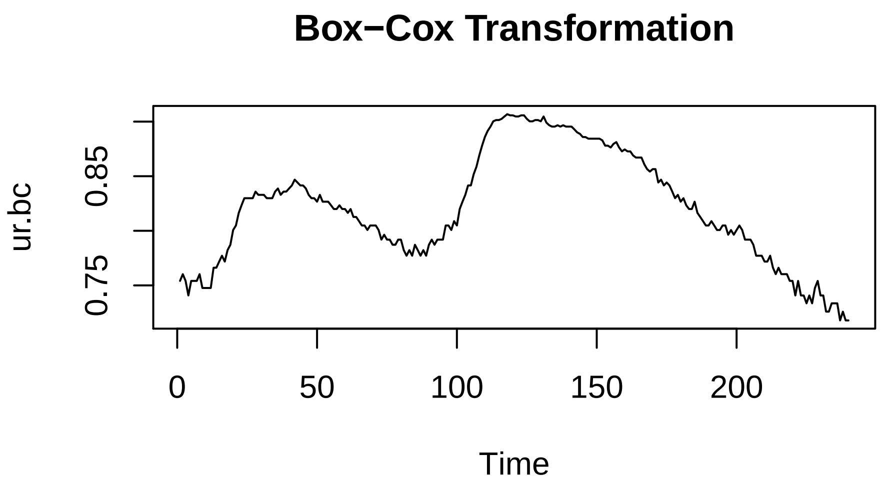
Make data stationary

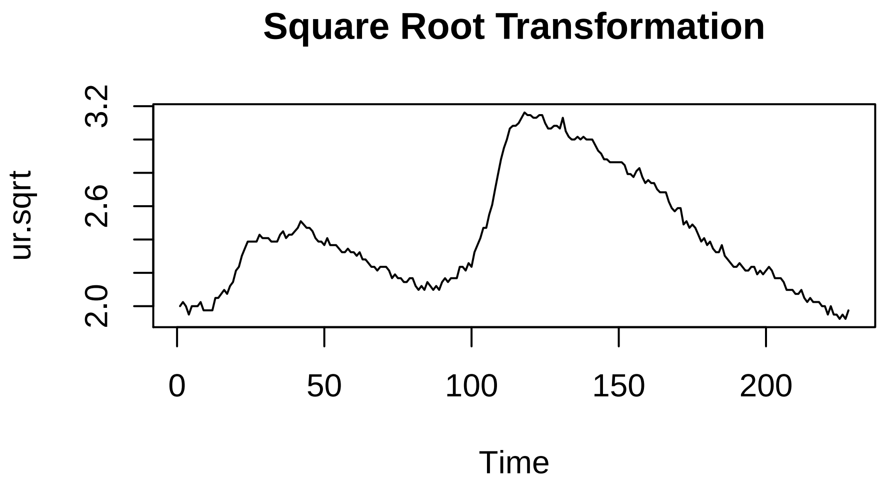
To begin to fix the non-stationarity issue, we first perform various transformations (Box-Cox, log, and square root) on the data to determine whether or not there are transformations that seem to help stabilize the variance:


95%
log−Likelihood
−350
−390
−2
−1
0
1
2
λ

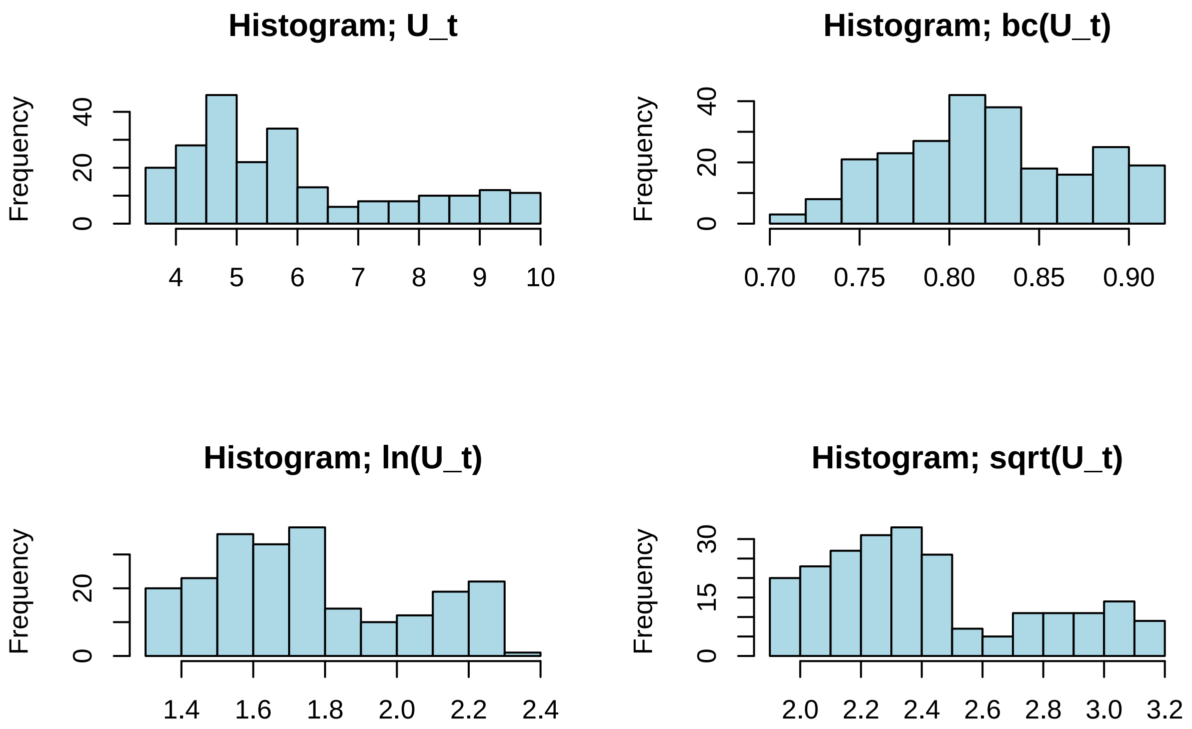

Original TS
ur.train
10
8
6
4
0
50
100
150
200
Time


ln Transformation
ur.log
2.2
1.8
1.4
0
50
100
150
200
Time



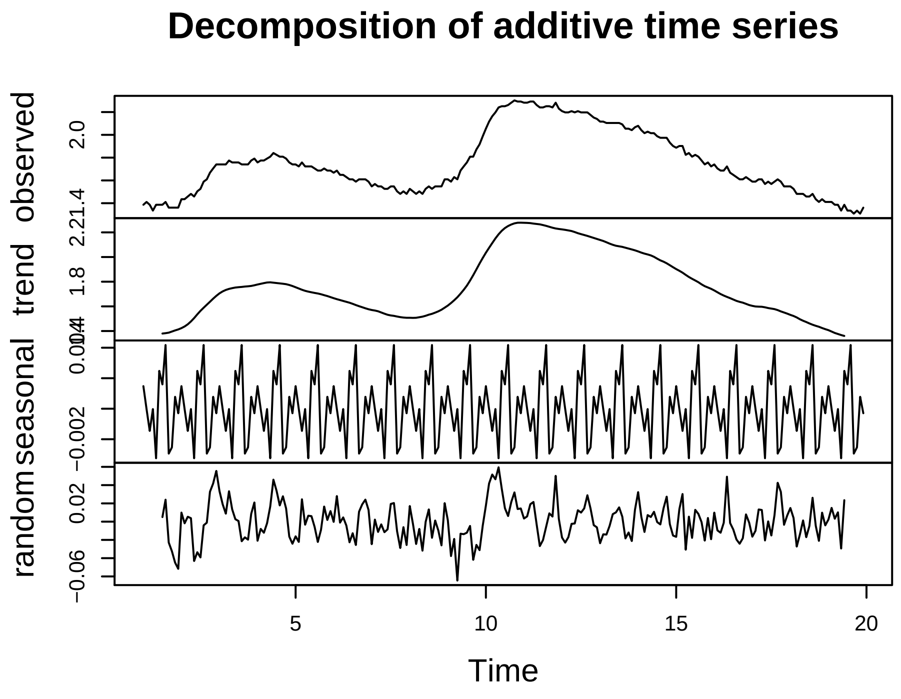


From the 4 graph outputs, we see that the most promising transformations are log and Box-Cox due to these transformations both visibly stabilizing the variance, while the square root transformation only stabilizes the variance to a lesser extent. We then proceed to plot the histograms of the transformations to compare the normality of the transformed data:

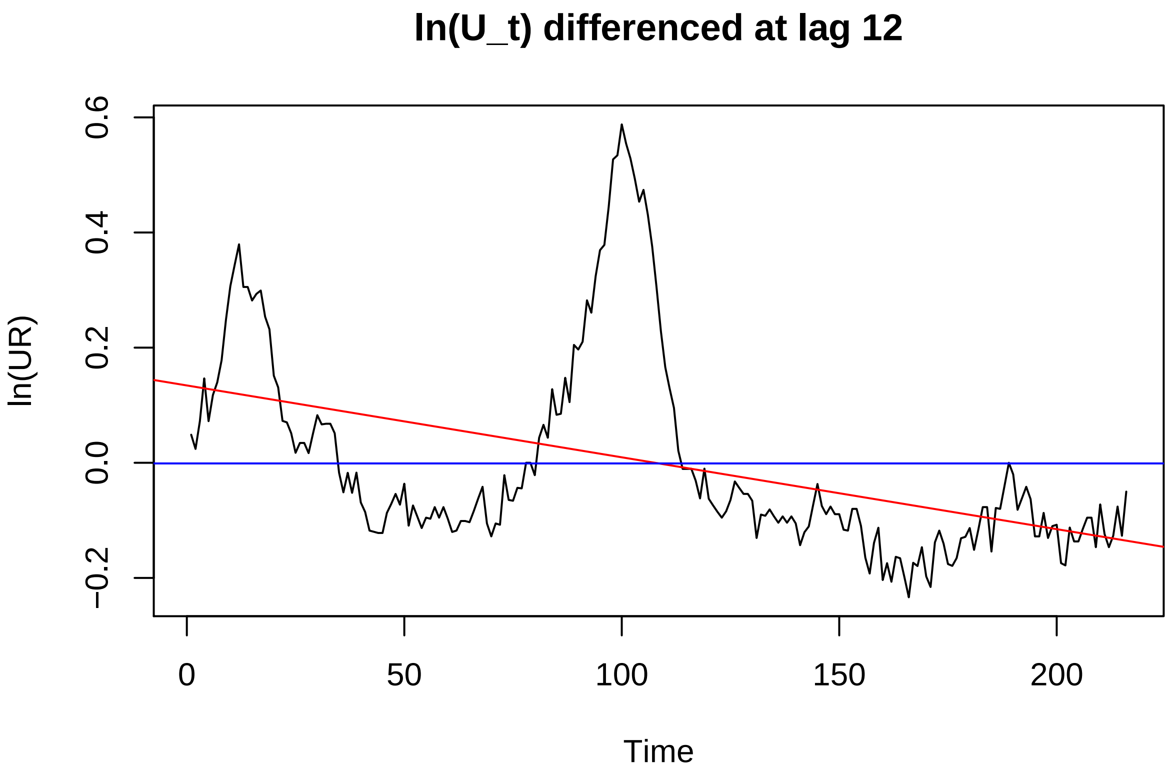


From the 4 histogram outputs, we see that, once again, the most promising transformations are log and Box-Cox due to their transformed data seeming to be the most Gaussian. As of now the Box-Cox transformation seems to be the “best” transformation we can perform. However, we decide to also test the suitability of the log transformation just to be sure. Thus, we proceed to checking the suitability of the two transformations for this data, starting with the log transformation.

We start by producing a decomposition of the log transformed data to determine whether or not there is seasonality or trend in the data:



The decomposition of $\ln(U_t)$ shows us that there is seasonality but no linear trend in the data. As such, we decide to perform differencing at lag=12 to remove the seasonal component, plot the results, and calculate the new variance:



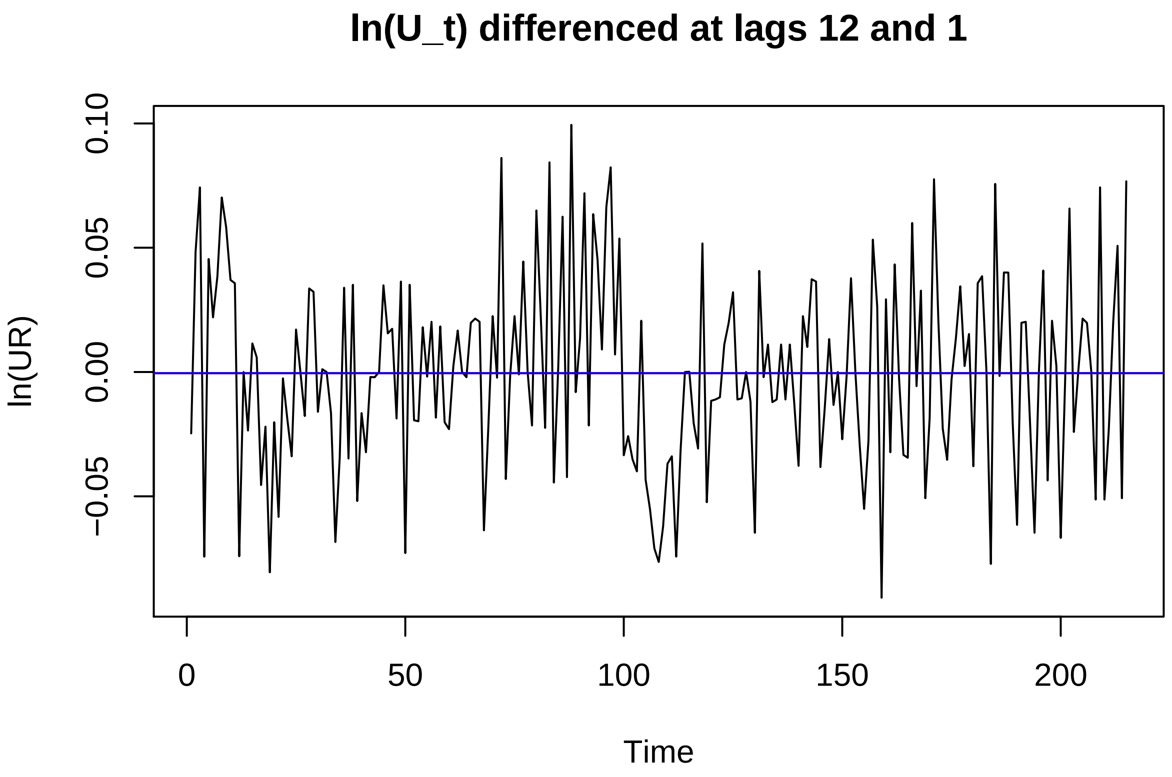
```
#calculate variance before differencing
var(ur.log)
```

```
## [1] 0.07791598
```

```
#calculate variance after differencing at lag=12
var(ur.log_12) #variance is lower!
```

```
## [1] 0.03044997
```

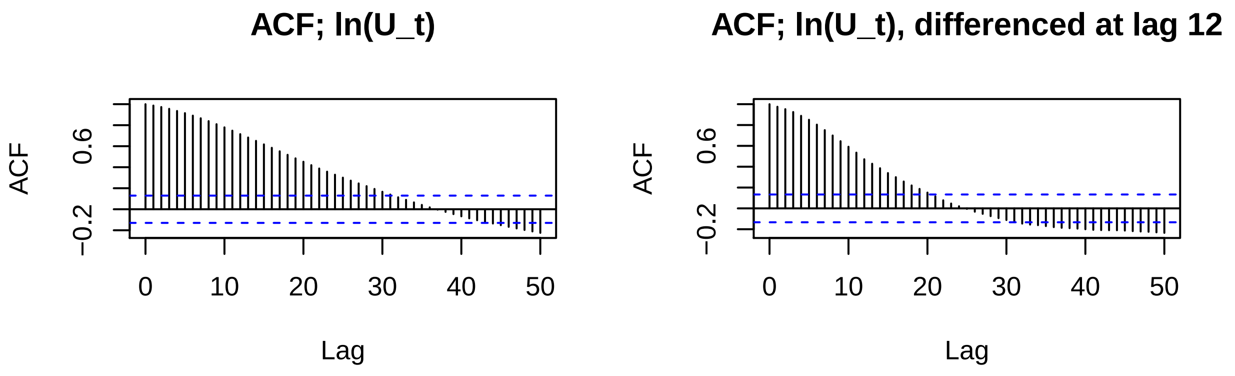
In the graph of $\ln(U_t)$ differenced at lag=12, we see that seasonality is now much less apparent after the differencing, and that the differenced data's variance of 0.03044997 is lower than the variance of the data before differencing, which was 0.07791598. However, this differencing has caused a very apparent decreasing trend that we must resolve. Thus, we perform differencing at lag=1 to remove the trend component and plot the results:

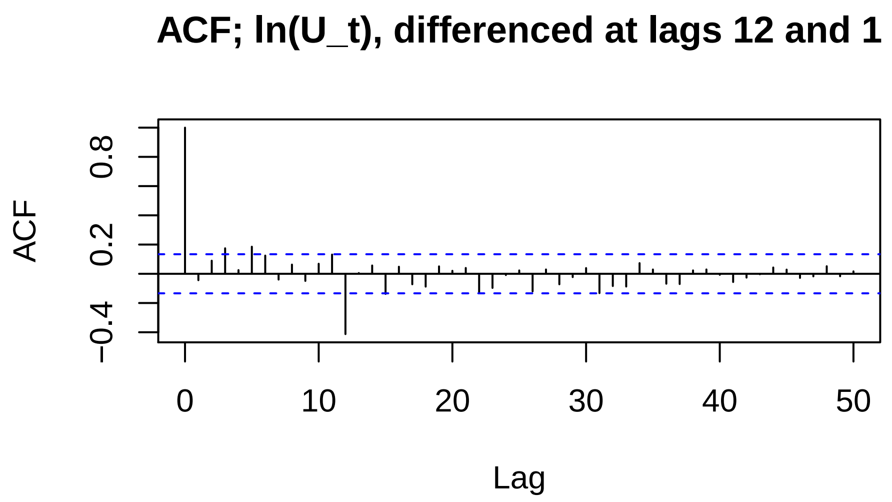


```
#calculate variance after differencing at lag=12 and lag=1
var(diff.ur.log) #variance is lower than when only differenced at lag=12
```

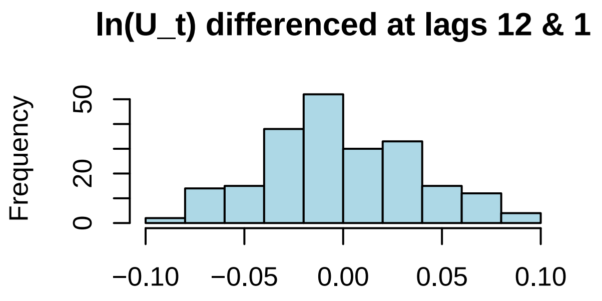
```
## [1] 0.001487522
```

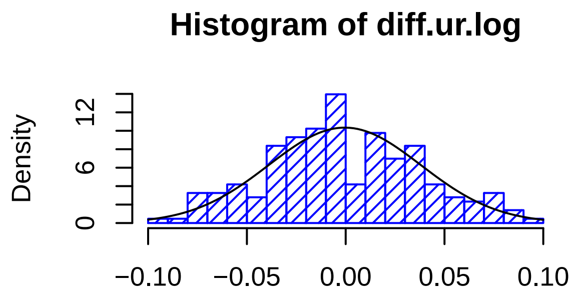
From the graph of $\ln(U_t)$ differenced at lag 12 and lag 1, we see that the trend component has been removed, there is still no apparent seasonality, and that the double-differenced data's variance of 0.001487522 is lower than 0.03044997. Thus, the log transformed data after double differencing seems to be stationary. To confirm the data's stationarity, we plot the ACFs of the $\ln(U_t)$, $\ln(U_t)$ differenced at lag=12, and $\ln(U_t)$ differenced at lag 12 and lag 1:





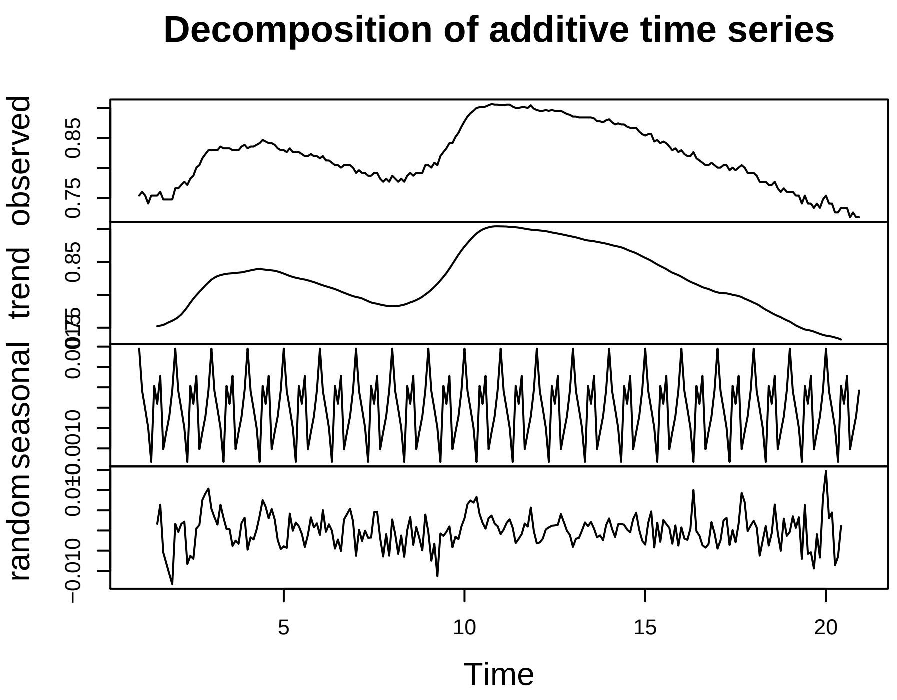
In the ACF plots of $\ln(U_t)$ and $\ln(U_t)$ differenced at lag=12, we see that the data is non-stationary due to them both having very large and slowly decaying ACFs. However, the ACF plot of $\ln(U_t)$ differenced at lag=12 and lag=1 has an ACF decay that greatly resembles a stationary process. Thus, we conclude that $\Delta_1\Delta_{12}ln(U_t)$ is suitable to use for model selection. We then plot the histogram of $\Delta_1\Delta_{12}ln(U_t)$ to confirm that it resembles a Gaussian distribution:

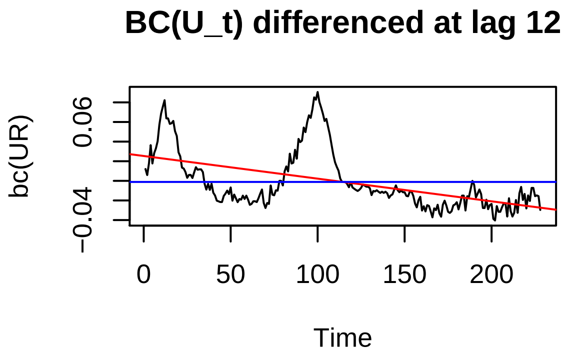


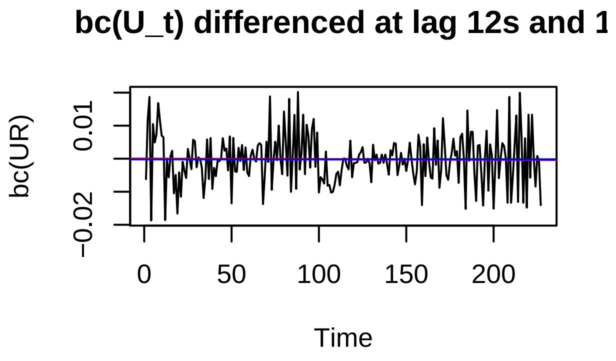


We confirm that the histogram of $\Delta_1\Delta_{12}ln(U_t)$ looks symmetric and approximately Gaussian, confirming the suitability of $\Delta_1\Delta_{12}ln(U_t)$.

Next, we proceed to repeat these steps of analysis for the Box-Cox transformed data to confirm the Box-Cox transformation's suitability and to ultimately determine which transformation would be better to perform on the unemployment rate data:







```r
#calculate variance before differencing
var(ur.bc)
```

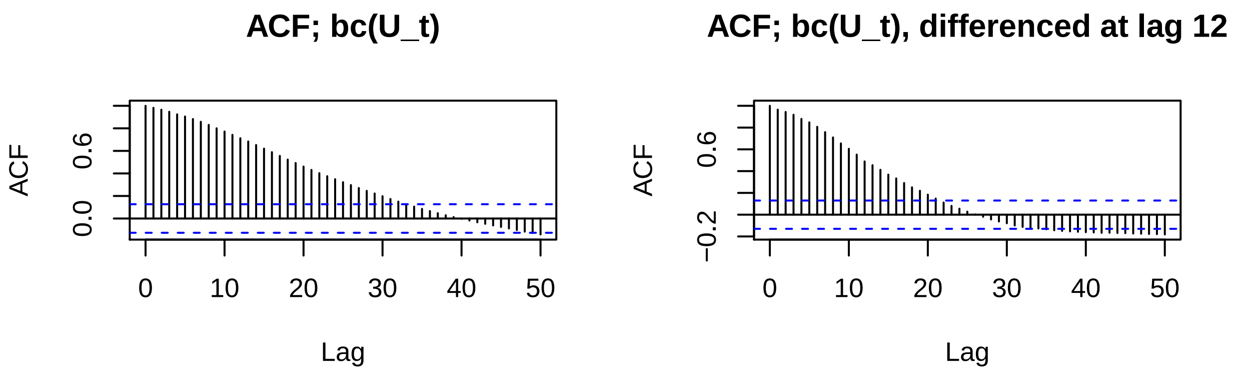
```
## [1] 0.002581586
```

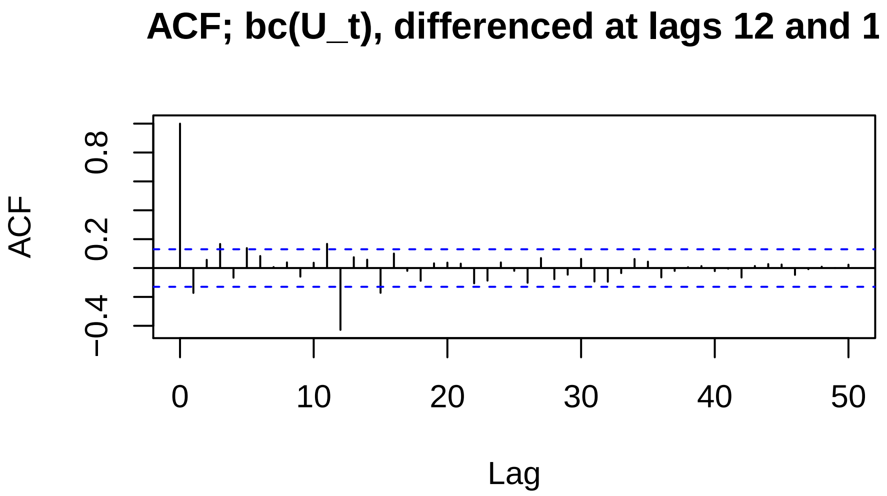
```r
#calculate variance after differencing at lag=12
var(ur.bc_12) #this is lower!
```

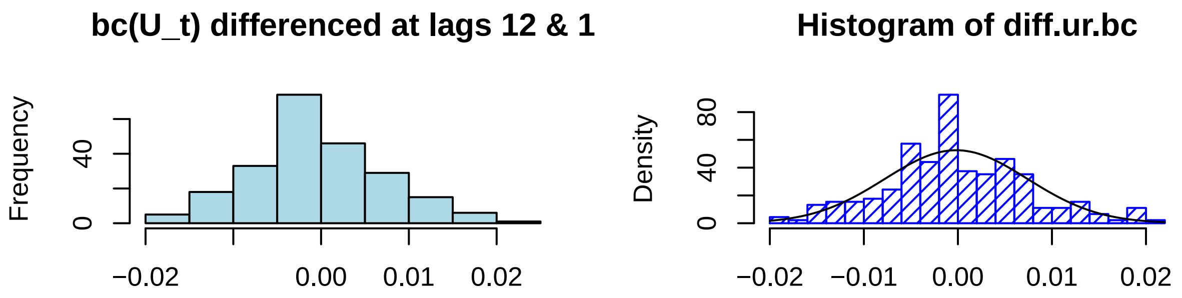
```
## [1] 0.0008800809
```

```r
#calculate variance after differencing at lag=12 and lag=1
var(diff.ur.bc) #this is lower!
```

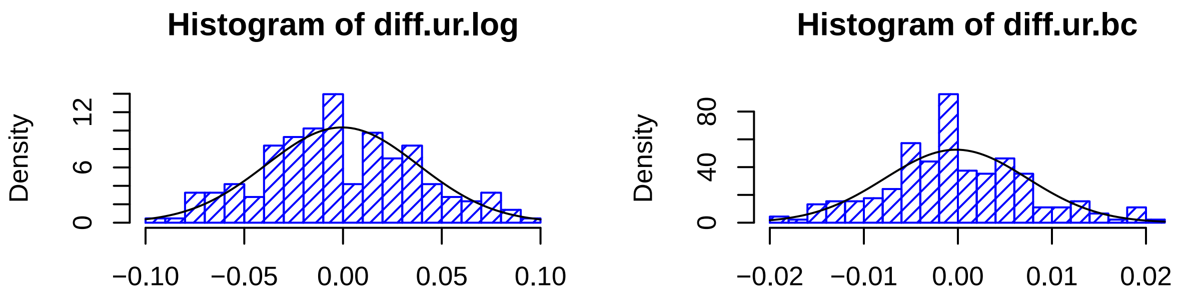
```
## [1] 5.763219e-05
```







Looking at all of the outputs for the Box-Cox transformation suitability tests, we see that all of the results are extremely similar to the results of the log transformation suitability tests and that both transformations pass each test. However, when comparing the histograms of the two, the histogram of $\Delta_1\Delta_{12}ln(U_t)$ seems more approximately Gaussian than $\Delta_1\Delta_{12}bc(U_t)$:



This speculation can be confirmed by using the Shapiro-Wilk test of normality, since the test p-value of 0.2811 for $\Delta_1\Delta_{12}ln(U_t)$ is greater than the test p-value of 0.058 for $\Delta_1\Delta_{12}bc(U_t)$, indicating that $\Delta_1\Delta_{12}ln(U_t)$ is more approximately Gaussian.

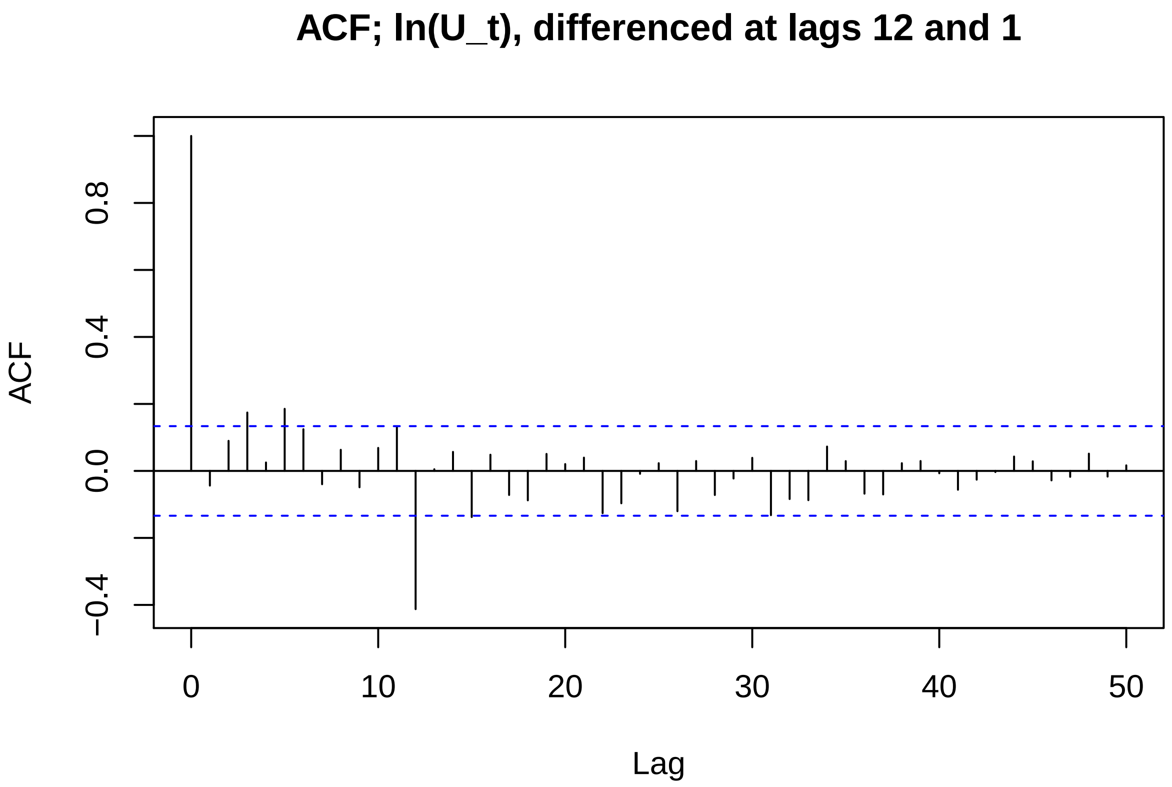
```
##
##  Shapiro-Wilk normality test
##
## data:  diff.ur.bc
## W = 0.98817, p-value = 0.058
```

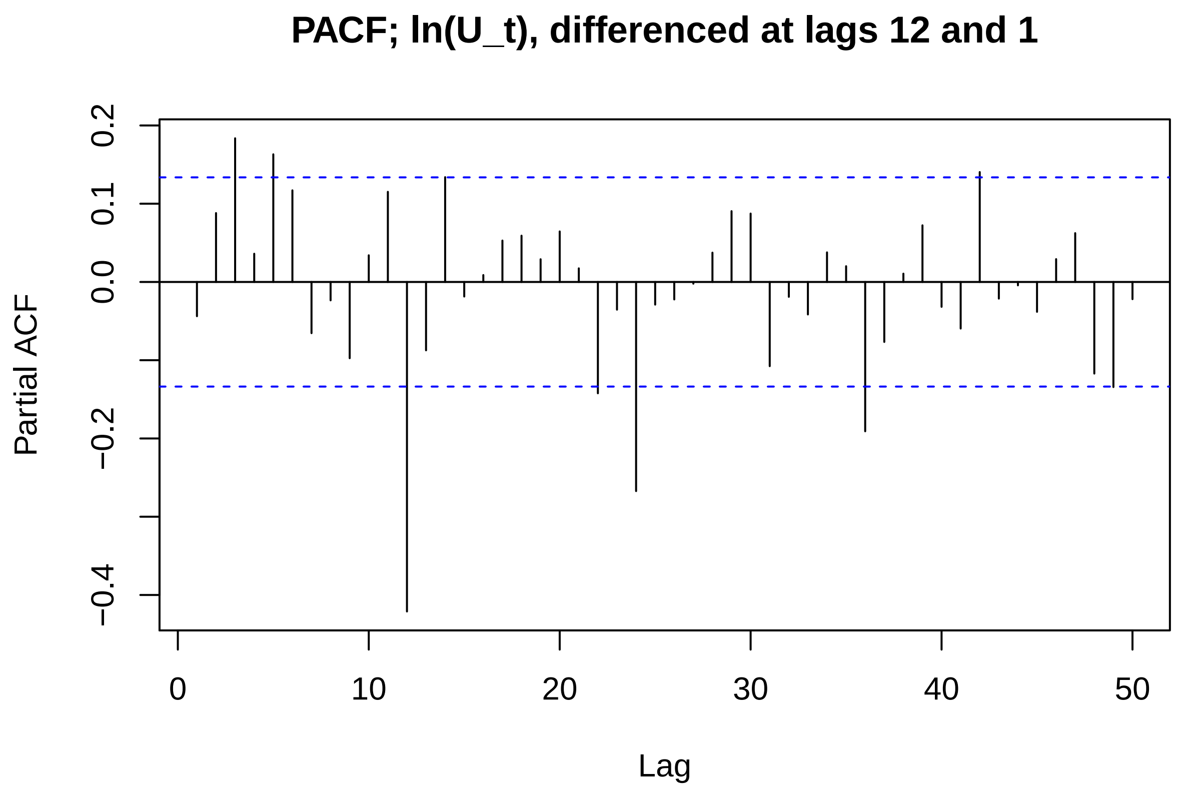
```
##
##  Shapiro-Wilk normality test
##
## data:  diff.ur.log
## W = 0.99191, p-value = 0.2811
```

As a result, we choose to proceed using the $\Delta_1\Delta_{12}ln(U_t)$ data (the log transformed training set data differenced at lag 12 and lag 1) for our model selection process.

Plot and analyze ACF and PACF

For preliminary model identification, we begin our selection process by analyzing the ACF and PACF of $\Delta_1\Delta_{12}ln(U_t)$ to determine suitable p, q, P, and Q values:





The ACF plot shows us that the ACFs are outside the confidence intervals at lag 3, lag 5, and lag 12. Thus, the suitable values for q are 3 and 5. 12 is not considered a suitable value for q (or p) because the seasonal length of the data (the s value) is 12. Since the ACF plot shows no ACF spikes at lags that are multiples of 12, the suitable values for Q are either 0 or 1. The PACF plot shows us that the PACFs are outside the confidence intervals at lag 3, lag 5, and lag 12, meaning that the suitable values for p are, once again, 3 and 5. The PACF plot always shows us that there are PACF spikes at lag 24 and lag 36, which are multiples of 12. As such, suitable values for P include 1,2, and 3. The D value and d value are both 1, since we performed a seasonal differencing at lag 12 and a non-seasonal differencing at lag 1. Thus the list of candidate models that are worth testing can be given by:

SARIMA for ln(U_t): s=12, D=1, d=1

- p=3 or 5
- q=3 or 5
- P=1, 2, or 3
- Q=0 or 1

We will test the candidate models by using the following formula to determine which models produce the lowest AIC(C) values:

```r
#AICc() function
AICc <- function(object)
{
  aic <- AIC(object)
  if (!is.numeric(aic)) stop("Cannot calculate AIC!")
  k <- length(coef(object))
  n <- length(residuals(object))
```

```
    aic + ((2 * k * (k + 1))/(n - k - 1))
}
```

Trying out models

We begin by trying out all SAR models (p=3 or 5; P=1, 2, or 3) and find that the SAR model that produces the lowest AIC(C) value is the model where p=5 and P=3:

```
#SAR model with lowest AIC(C)
arima(ur.log, order=c(5,1,0), seasonal=list(order=c(3,1,0), period=12), method="ML")
```

```
##
## Call:
## arima(x = ur.log, order = c(5, 1, 0), seasonal = list(order = c(3, 1, 0), period = 12),
##     method = "ML")
##
## Coefficients:
##           ar1     ar2     ar3     ar4     ar5     sar1     sar2     sar3
##       -0.0266  0.2193  0.1413  0.0813  0.2560  -0.8488  -0.5975  -0.3608
## s.e.   0.0675  0.0691  0.0704  0.0696  0.0701   0.0752   0.0851   0.0735
##
## sigma^2 estimated as 0.0008167:  log likelihood = 453.15,  aic = -888.3
```

```
AICc(arima(ur.log, order=c(5,1,0), seasonal=list(order=c(3,1,0), period=12), method="ML"))
```

```
## [1] -887.6405
```

Based on the standard errors of the coefficients for the model, the confidence intervals for the AR(1) and AR(4) coefficients contain 0, so we try setting those coefficients to 0 and see whether or not the AIC(C) value decreases:

```
#test SAR model with lowest AIC(C) (fixed 0's)
AICc(arima(ur.log, order=c(5,1,0), seasonal=list(order=c(3,1,0), period=12),
           fixed=c(0,NA,NA,NA,NA,NA,NA,NA), method="ML"))
```

```
## [1] -889.4854
```

```
#AIC(C) = -889.4854 < -887.6405, so set AR(1) coef to 0
AICc(arima(ur.log, order=c(5,1,0), seasonal=list(order=c(3,1,0), period=12),
           fixed=c(0,NA,NA,0,NA,NA,NA,NA), method="ML"))
```

```
## [1] -890.2497
```

```
#AIC(C) = -890.2496 < -889.4854, so set AR(4) coef to 0 (best SAR model)
```

```
#best SAR model
arima(ur.log, order=c(5,1,0), seasonal=list(order=c(3,1,0), period=12),
      fixed=c(0,NA,NA,0,NA,NA,NA,NA), method="ML")
```

```
##
## Call:
## arima(x = ur.log, order = c(5, 1, 0), seasonal = list(order = c(3, 1, 0), period = 12),
##     fixed = c(0, NA, NA, 0, NA, NA, NA, NA), method = "ML")
##
## Coefficients:
##       ar1     ar2     ar3  ar4     ar5     sar1     sar2     sar3
##         0  0.2418  0.1396    0  0.2560  -0.8445  -0.5856  -0.3643
## s.e.    0  0.0667  0.0681    0  0.0705   0.0748   0.0845   0.0734
##
## sigma^2 estimated as 0.0008219:  log likelihood = 452.45,  aic = -890.91
```

```
AICc(arima(ur.log, order=c(5,1,0), seasonal=list(order=c(3,1,0), period=12),
           fixed=c(0,NA,NA,0,NA,NA,NA,NA), method="ML"))
```

```
## [1] -890.2497
```

After determining that setting the AR(1) and AR(4) coefficients to 0 does decrease the AIC(C) value of the best SAR model, we proceed to try out all candidate SMA models (q=3 or 5; Q=0 or 1). We find that the SMA model that produces the lowest AIC(C) value is the model where q=5 and q=3:

```
#SMA model with lowest AIC(C)
arima(ur.log, order=c(0,1,5), seasonal=list(order=c(0,1,1), period=12), method="ML")
```

```
##
## Call:
## arima(x = ur.log, order = c(0, 1, 5), seasonal = list(order = c(0, 1, 1), period = 12),
##     method = "ML")
##
## Coefficients:
##           ma1     ma2     ma3     ma4     ma5     sma1
##       -0.0644  0.1919  0.1077  0.0606  0.2069  -1.0000
## s.e.   0.0716  0.0711  0.0641  0.0742  0.0575   0.0585
##
## sigma^2 estimated as 0.000671:  log likelihood = 462.7,  aic = -911.39
```

```
AICc(arima(ur.log, order=c(0,1,5), seasonal=list(order=c(0,1,1), period=12), method="ML"))
```

```
## [1] -911.0144
```

Based on the standard errors of the coefficients for the model, the confidence intervals for the MA(1), MA(3), and MA(4) coefficients contain 0, so we try setting those coefficients to 0 and see whether or not the AIC(C) value decreases:

```
#test SMA model with lowest AIC(C) (fixed 0's)
AICc(arima(ur.log, order=c(0,1,5), seasonal=list(order=c(0,1,1), period=12),
           fixed=c(0,NA,NA,NA,NA,NA), method="ML"))
```

```
## [1] -912.2036
```

```r
#AIC(C) = -912.2036 < -911.0144, so set MA(1) coef to 0
AICc(arima(ur.log, order=c(0,1,5), seasonal=list(order=c(0,1,1), period=12),
           fixed=c(0,NA,0,NA,NA,NA), method="ML"))
```

```
## [1] -910.9523
```

```r
#AIC(C) = -910.9523 > -912.2036, so don't set MA(2) coef to 0
AICc(arima(ur.log, order=c(0,1,5), seasonal=list(order=c(0,1,1), period=12),
           fixed=c(0,NA,NA,0,NA,NA), method="ML"))
```

```
## [1] -913.1596
```

```r
#AIC(C) = -913.1596 < -912.2036, so set MA(4) coef to 0 (best SMA model)
```

```r
#best SMA model
arima(ur.log, order=c(0,1,5), seasonal=list(order=c(0,1,1), period=12),
      fixed=c(0,NA,NA,0,NA,NA), method="ML")
```

```
##
## Call:
## arima(x = ur.log, order = c(0, 1, 5), seasonal = list(order = c(0, 1, 1), period = 12),
##     fixed = c(0, NA, NA, 0, NA, NA), method = "ML")
##
## Coefficients:
##       ma1     ma2     ma3  ma4     ma5     sma1
##         0  0.2036  0.1309    0  0.1948  -1.0000
## s.e.    0  0.0668  0.0595    0  0.0569   0.0589
##
## sigma^2 estimated as 0.0006767:  log likelihood = 461.77,  aic = -913.54
```

```r
AICc(arima(ur.log, order=c(0,1,5), seasonal=list(order=c(0,1,1), period=12),
           fixed=c(0,NA,NA,0,NA,NA), method="ML"))
```

```
## [1] -913.1596
```

We confirm that setting the MA(1) and MA(4) coefficients to 0 does decrease the AIC(C) value of the best SMA model. Next, we try out all the candidate SARIMA models to determine which SARIMA models produce the lowest AIC(C). The two SARIMA models that produce the lowest AIC(C) turn out to be $SARIMA(3,1,5) \times (3,1,1)_{12}$ and $SARIMA(5,1,3) \times (3,1,1)_{12}$:

```r
#SARIMA with lowest AIC(C)
arima(ur.log, order=c(3,1,5), seasonal=list(order=c(3,1,1), period=12), method="ML")
```

```
##
## Call:
## arima(x = ur.log, order = c(3, 1, 5), seasonal = list(order = c(3, 1, 1), period = 12),
##     method = "ML")
##
## Coefficients:
##          ar1      ar2     ar3      ma1     ma2      ma3     ma4     ma5
##       1.4645  -1.4109  0.9120  -1.6027  1.8477  -1.2806  0.3320  0.0335
## s.e.  0.0340   0.0520  0.0444   0.0759  0.1538   0.1808  0.1345  0.0793
##          sar1     sar2     sar3     sma1
##       -0.3532  -0.3260  -0.2514  -0.9976
## s.e.   0.0758   0.0771   0.0775   0.0952
##
## sigma^2 estimated as 0.0004828:  log likelihood = 488.25,  aic = -950.5
```

```r
AICc(arima(ur.log, order=c(3,1,5), seasonal=list(order=c(3,1,1), period=12), method="ML"))
```

```
## [1] -949.053
```

```r
#SARIMA with second lowest AIC(C)
arima(ur.log, order=c(5,1,3), seasonal=list(order=c(3,1,1), period=12), method="ML")
```

```
##
## Call:
## arima(x = ur.log, order = c(5, 1, 3), seasonal = list(order = c(3, 1, 1), period = 12),
##     method = "ML")
##
## Coefficients:
##          ar1      ar2     ar3     ar4     ar5      ma1     ma2      ma3
##       0.9651  -0.5752  0.2000  0.1522  0.2049  -1.0702  0.9145  -0.3765
## s.e.  0.2114   0.3504  0.2483  0.1252  0.1018   0.2071  0.3652   0.2634
##          sar1     sar2     sar3     sma1
##       -0.3668  -0.2978  -0.2565  -0.9999
## s.e.   0.0853   0.0779   0.0796   0.1269
##
## sigma^2 estimated as 0.0004975:  log likelihood = 486.35,  aic = -946.69
```

```r
AICc(arima(ur.log, order=c(5,1,3), seasonal=list(order=c(3,1,1), period=12), method="ML"))
```

```
## [1] -945.2402
```

We see that, once again, the confidence intervals for some of the coefficients for each SARIMA models contain 0, so we try setting those coefficients (the MA(5) coefficient for the $SARIMA(3,1,5) \times (3,1,1)_{12}$ model and the AR(2), AR(3), AR(4), and MA(3) coefficients for the $SARIMA(5,1,3) \times (3,1,1)_{12}$ model) to 0 and see whether or not the AIC(C) values decrease for either model:

```r
#test SARIMA(3,1,5)x(3,1,1)_{12} (fixed 0's)
AICc(arima(ur.log, order=c(3,1,5), seasonal=list(order=c(3,1,1), period=12),
           fixed=c(NA,NA,NA,NA,NA,NA,NA,0,NA,NA,NA,NA), method="ML"))
```

```
## [1] -946.9846
```

```r
#-946.9846 > -949.053, so don't set MA(5) coef to 0
```

```r
#test SARIMA(5,1,3)x(3,1,1)_{12} (fixed 0's)
AICc(arima(ur.log, order=c(5,1,3), seasonal=list(order=c(3,1,1), period=12),
           fixed=c(NA,0,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA), method="ML"))
```

```
## [1] -945.8834
```

```r
#-945.8834 < -945.2402 but sma1 coef > |1|
AICc(arima(ur.log, order=c(5,1,3), seasonal=list(order=c(3,1,1), period=12),
           fixed=c(NA,0,0,NA,NA,NA,NA,NA,NA,NA,NA,NA), method="ML"))
```

```
## [1] -947.7885
```

```r
#-947.7885 < -945.8834 and sma1 < |1|
AICc(arima(ur.log, order=c(5,1,3), seasonal=list(order=c(3,1,1), period=12),
           fixed=c(NA,0,0,0,NA,NA,NA,NA,NA,NA,NA,NA), method="ML"))
```

```
## [1] -949.5741
```

```r
#-949.5741 < -947.7885 but sma = |1|
AICc(arima(ur.log, order=c(5,1,3), seasonal=list(order=c(3,1,1), period=12),
           fixed=c(NA,0,0,0,NA,NA,NA,0,NA,NA,NA,NA), method="ML"))
```

```
## [1] -951.3114
```

```r
#-951.3114 < -949.5741 but sma > |1|
#we choose the best model with sma1 coef < |1|
arima(ur.log, order=c(5,1,3), seasonal=list(order=c(3,1,1), period=12),
      fixed=c(NA,0,0,NA,NA,NA,NA,NA,NA,NA,NA,NA), method="ML")
```

```
##
## Call:
## arima(x = ur.log, order = c(5, 1, 3), seasonal = list(order = c(3, 1, 1), period = 12),
##     fixed = c(NA, 0, 0, NA, NA, NA, NA, NA, NA, NA, NA, NA), method = "ML")
##
## Coefficients:
##          ar1  ar2  ar3     ar4     ar5      ma1     ma2      ma3     sar1
##       0.6586    0    0  0.0401  0.2460  -0.7675  0.2978  -0.0514  -0.3668
## s.e.  0.1104    0    0  0.0855  0.0837   0.1245  0.0873   0.0780   0.0739
##          sar2     sar3     sma1
##       -0.2927  -0.2458  -0.9999
## s.e.   0.0789   0.0771   0.1113
##
## sigma^2 estimated as 0.0005019:  log likelihood = 485.62,  aic = -949.24
```

In the end, setting the MA(5) coefficient for the $SARIMA(3,1,5) \times (3,1,1)_{12}$ model to 0 did not decrease the AIC(C) but setting the AR(2) and AR(3) coefficients to 0 for the $SARIMA(5,1,3) \times (3,1,1)_{12}$ model did decrease the AIC(C) and actually caused the $SARIMA(5,1,3) \times (3,1,1)_{12}$ model to become the SARIMA model that produces the lowest AIC(C).

Thus, the two best SARIMA models are the following:

```r
#second-best SARIMA model (A)
arima(ur.log, order=c(3,1,5), seasonal=list(order=c(3,1,1), period=12), method="ML")
```

```
##
## Call:
## arima(x = ur.log, order = c(3, 1, 5), seasonal = list(order = c(3, 1, 1), period = 12),
##     method = "ML")
##
## Coefficients:
##          ar1      ar2     ar3      ma1     ma2      ma3     ma4     ma5
##       1.4645  -1.4109  0.9120  -1.6027  1.8477  -1.2806  0.3320  0.0335
## s.e.  0.0340   0.0520  0.0444   0.0759  0.1538   0.1808  0.1345  0.0793
##          sar1     sar2     sar3     sma1
##       -0.3532  -0.3260  -0.2514  -0.9976
## s.e.   0.0758   0.0771   0.0775   0.0952
##
## sigma^2 estimated as 0.0004828:  log likelihood = 488.25,  aic = -950.5
```

```
AICc(arima(ur.log, order=c(3,1,5), seasonal=list(order=c(3,1,1), period=12), method="ML"))
```

```
## [1] -949.053
```

```
#best SARIMA model (B)
arima(ur.log, order=c(5,1,3), seasonal=list(order=c(3,1,1), period=12),
      fixed=c(NA,0,0,NA,NA,NA,NA,NA,NA,NA,NA,NA), method="ML")
```

```
##
## Call:
## arima(x = ur.log, order = c(5, 1, 3), seasonal = list(order = c(3, 1, 1), period = 12),
##     fixed = c(NA, 0, 0, NA, NA, NA, NA, NA, NA, NA, NA, NA), method = "ML")
##
## Coefficients:
##          ar1  ar2  ar3     ar4     ar5      ma1     ma2      ma3     sar1
##       0.6586    0    0  0.0401  0.2460  -0.7675  0.2978  -0.0514  -0.3668
## s.e.  0.1104    0    0  0.0855  0.0837   0.1245  0.0873   0.0780   0.0739
##          sar2     sar3     sma1
##       -0.2927  -0.2458  -0.9999
## s.e.   0.0789   0.0771   0.1113
##
## sigma^2 estimated as 0.0005019:  log likelihood = 485.62,  aic = -949.24
```

```
AICc(arima(ur.log, order=c(5,1,3), seasonal=list(order=c(3,1,1), period=12),
           fixed=c(NA,0,0,NA,NA,NA,NA,NA,NA,NA,NA,NA), method="ML"))
```

```
## [1] -947.7885
```

Out of the best SAR, SMA, and SARIMA models, the two models that produce the lowest AIC(C) values are the two SARIMA models. Thus, we proceed to fit both $SARIMA(3,1,5)\times(3,1,1)_{12}$ and $SARIMA(5,1,3)\times(3,1,1)_{12}$ and perform diagnostic checking on both of them to determine whether or not they are suitable for forecasting.

Fit models and perform diagnostic checking

Our final two models that will be tested and compared are:

- Model (A): $\Delta_1\Delta_{12}(1 - 1.4645_{(0.0340)}B + 1.4109_{(0.0520)}B^2 - 0.9120_{(0.0444)}B^3)(1 + 0.3532_{(0.0758)}B^{12} + 0.3260_{(0.0771)]}B^{24} + 0.2514_{(0.0775)}B^{36})ln(U_t) = (1 - 1.6027_{(0.0759)}B + 1.8477_{(0.1538)}B^2 - 1.2806_{(0.1808)}B^3 + 0.3320_{(0.1345)}B^4 + 0.0335_{(0.0793)}B^5)(1 - 0.9976_{(0.0952)}B^{12})Z_t$

$\sigma_Z^2 = 0.0004828$

- Model (B): $\Delta_1\Delta_{12}(1 - 0.6586_{(0.1104)}B - 0.0401_{(0.0855)}B^4 - 0.2460_{(0.0837)}B^5)(1 + 0.3668_{(0.0739)}B^{12} + 0.2927_{(0.0789)}B^{24} + 0.2458_{(0.0771)}B^{36})ln(U_t) = (1 - 0.7675_{(0.1245)}B + 0.2978_{(0.0873)}B^2 - 0.0514_{(0.0780)}B^3)(1 - 0.9999_{(0.1074)}B^{12})Z_t$

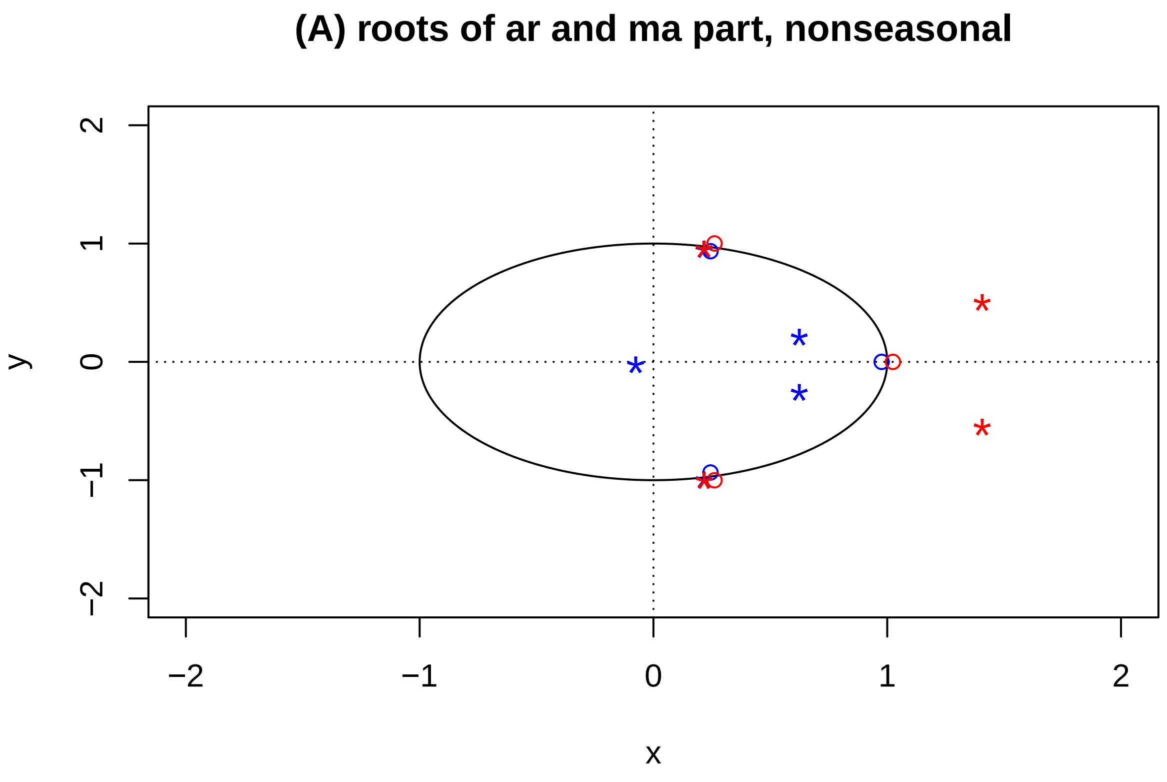
$\sigma_Z^2 = 0.0005019$

We proceed to fit both models to proceed with diagnostic checking:

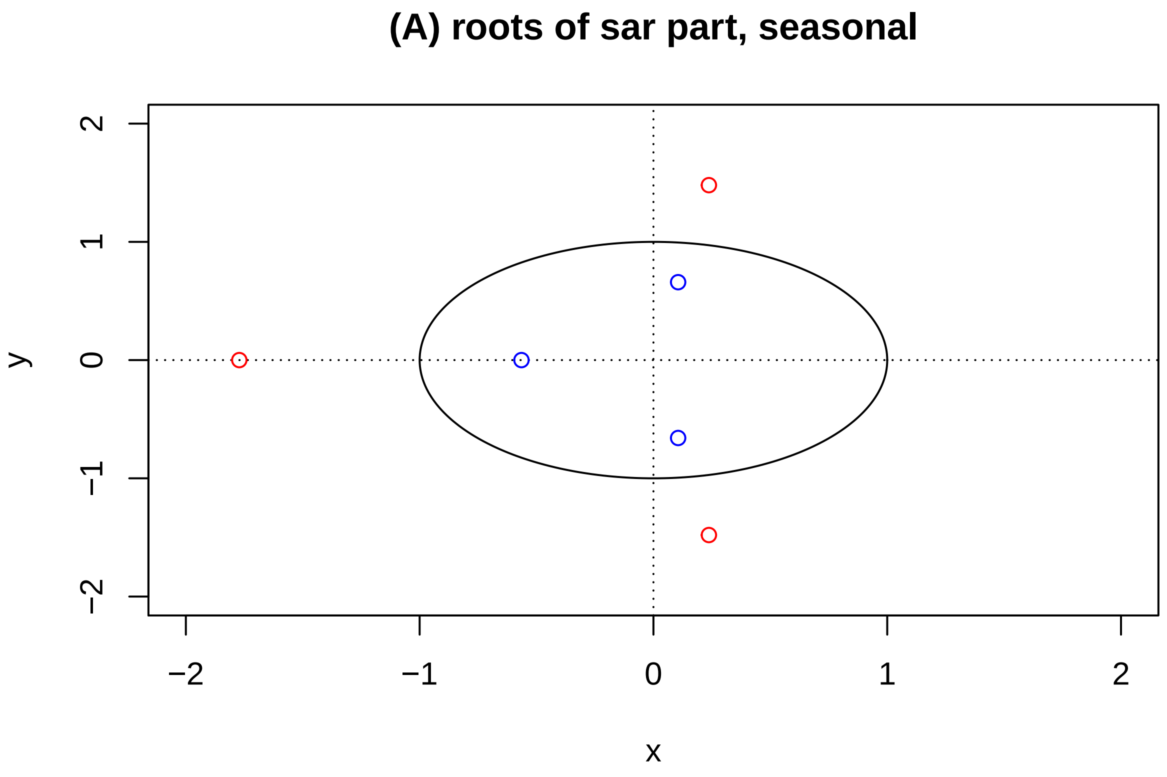
```
#fit Model A
fit.A = arima(ur.log, order=c(3,1,5), seasonal=list(order=c(3,1,1), period=12), method="ML")
#fit Model B
fit.B = arima(ur.log, order=c(5,1,3), seasonal=list(order=c(3,1,1), period=12),
              fixed=c(NA,0,0,NA,NA,NA,NA,NA,NA,NA,NA,NA), method="ML")
```

The first diagnostic that we will check is whether or not each model is both invertible and stationary by using the plot.roots() function, starting with Model A (check Appendix for plot.roots() function code):

```
#check invertibility/stationarity of Model (A)
plot.roots(polyroot(c(1,-1.4645,1.4109,-0.9120)),
           polyroot(c(1,-1.6027,1.8477,-1.2806,0.3320,0.0335)),
           main="(A) roots of ar and ma part, nonseasonal")
```

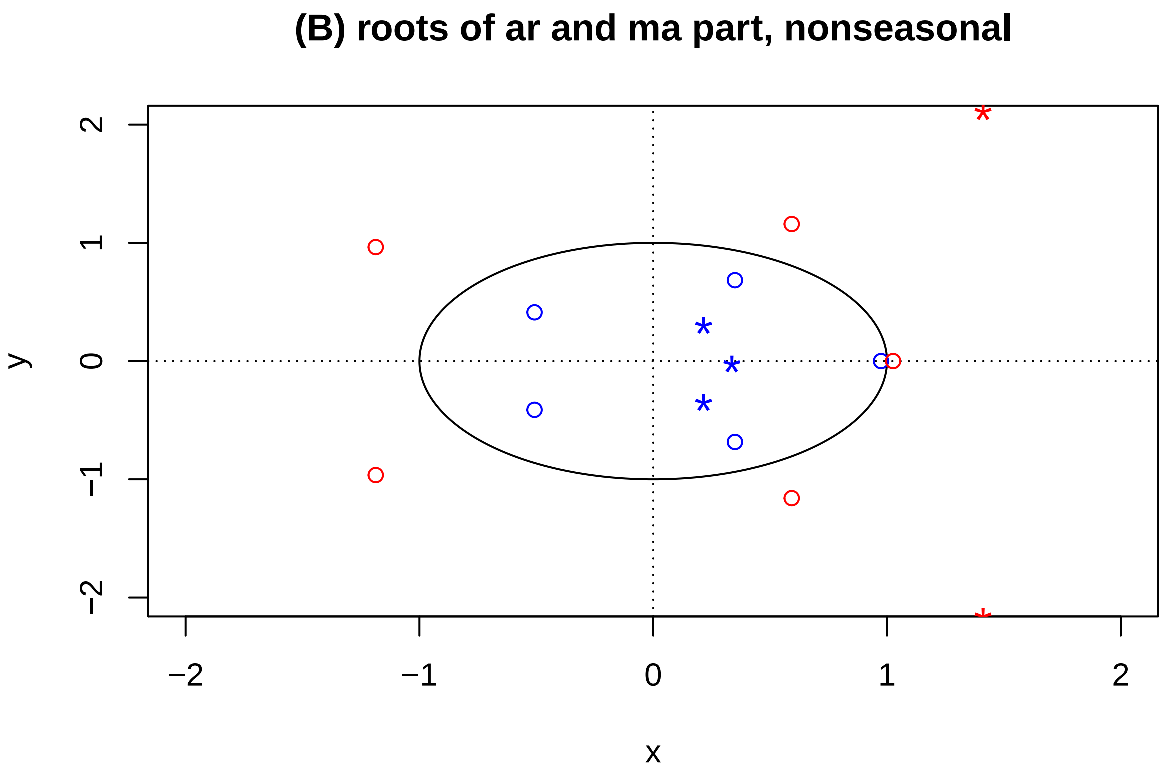


```r
plot.roots(polyroot(c(1,0.3532,0.3260,0.2514)), main="(A) roots of sar part, seasonal")
```

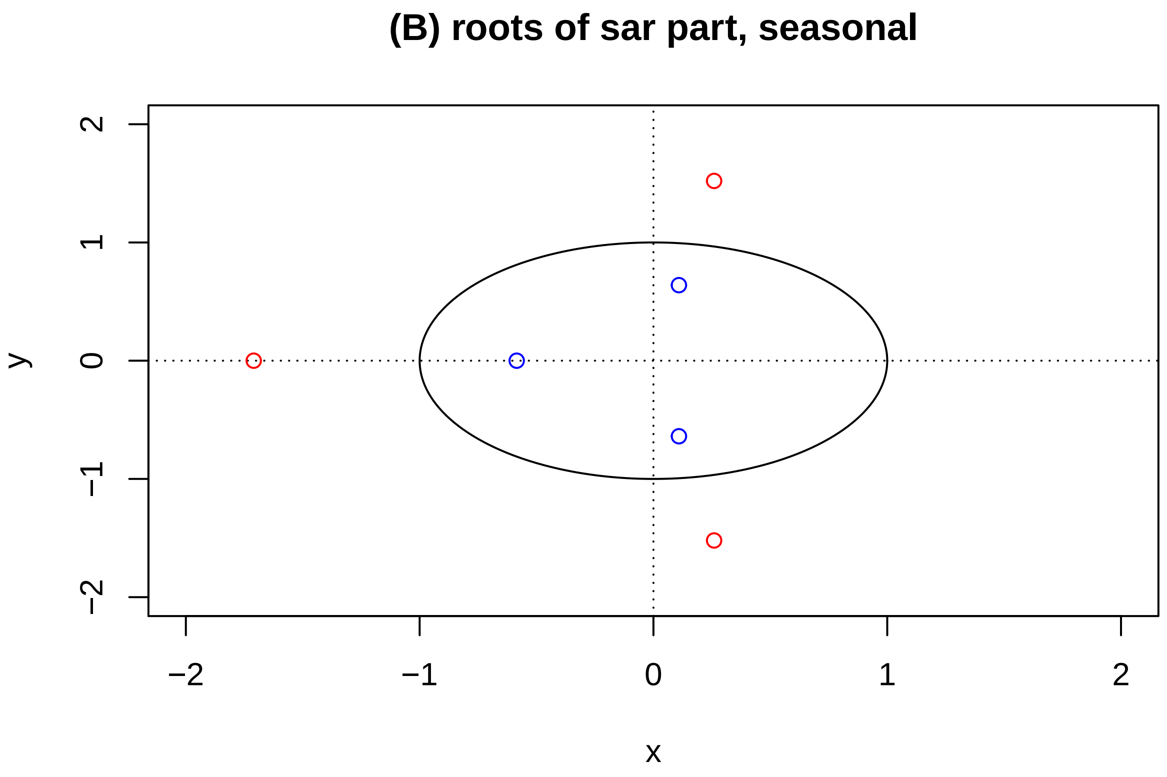


From the plot.roots() output, we see that 2 of the roots from the nonseasonal MA part are on the unit circle, and, as a result, Model A is not invertible. As such, we cannot proceed to forecasting using the $SARIMA(3,1,5) \times (3,1,1)_{12}$ model. We then move on to testing the invertibility and stationarity of Model B:

```r
#check invertibility/stationarity of Model (B)
plot.roots(polyroot(c(1,-0.6586,0,0,-0.0401,-0.2460)),
           polyroot(c(1,-0.7675,0.2978,-0.0514)),
           main="(B) roots of ar and ma part, nonseasonal")
```



```
plot.roots(polyroot(c(1,0.3668,0.2927,0.2458)), main="(B) roots of sar part, seasonal")
```



In the plot.roots() output, all the roots in both the nonseasonal MA and AR parts and the seasonal SAR part are outside of the unit circle. Since the seasonal SMA part only has one coefficient that is less than $|1|$, we conclude that Model B is both stationary and invertible. Thus, our "best" model is $SARIMA(5,1,3)\times$

$(3,1,1)_{12}$ (Model B) due to the fact that Model B produces the lowest AIC(C) value and the fact that $SARIMA(3,1,5) \times (3,1,1)_{12}$ (Model A) didn't pass the unit circle invertibility check. This model (Model B) obtained by using AIC(C) is in fact the same as one of the models that was preliminarily suggested by the ACF and PACF of $\Delta_1\Delta_{12}ln(U_t)$.

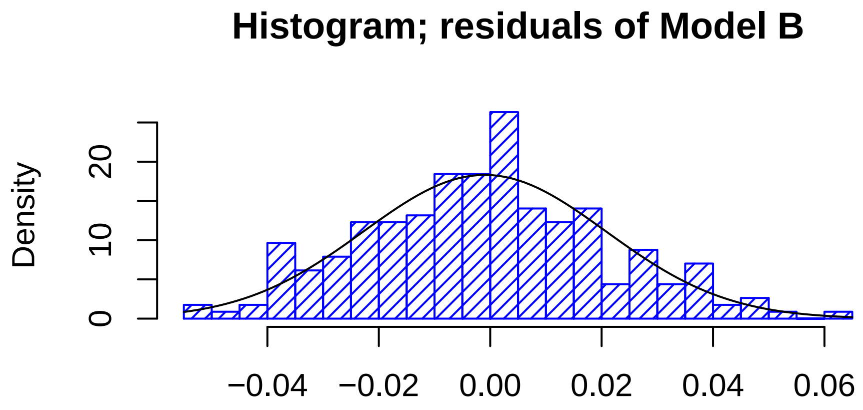
Best Model:

$$\Delta_1\Delta_{12}(1-0.6586_{(0.1104)}B-0.0401_{(0.0855)}B^4-0.2460_{(0.0837)}B^5)(1+0.3668_{(0.0739)}B^{12}+0.2927_{(0.0789)}B^{24}+0.2458_{(0.0771)}B^{36})ln(U_t)=(1-0.7675_{(0.1245)}B+0.2978_{(0.0873)}B^2-0.0514_{(0.0780)}B^3)(1-0.9999_{(0.1074)}B^{12})Z_t$$

$\sigma_Z^2 = 0.0005019$

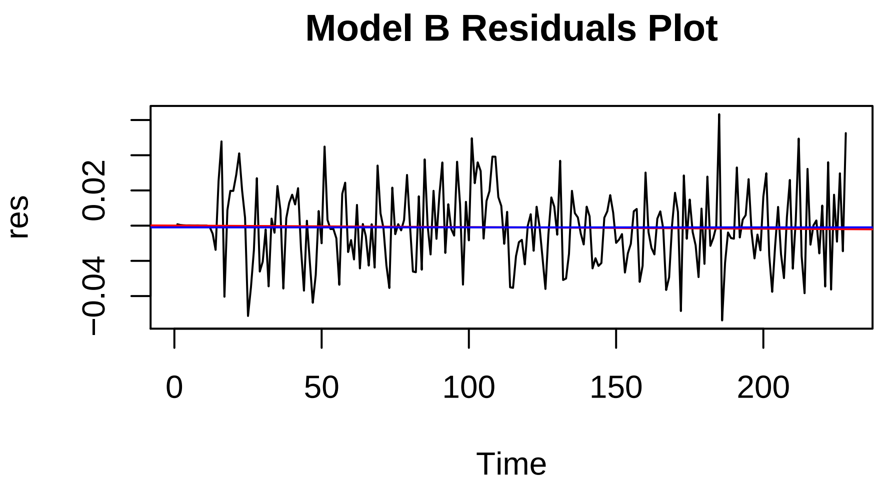
Next, we perform analysis of residuals on our best model to determine if the model can be used for forecasting:

```r
#analysis of residuals for Model B
#fit residuals
res = residuals(fit.B)
#plot histogram of residuals
hist(as.numeric(res), density=20, breaks=20, col="blue", xlab="", prob=TRUE,
     main="Histogram; residuals of Model B")
m = mean(res)
std = sqrt(var(res))
curve(dnorm(x,m,std), add=TRUE)
```



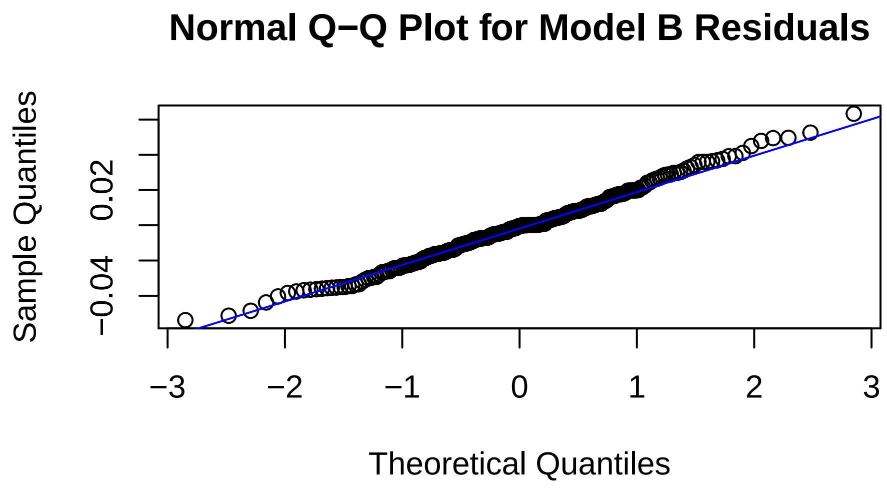
The histogram of the residuals looks approximately Gaussian, so we continue with our diagnostic checks.

```r
#plot residuals
plot.ts(res, main="Model B Residuals Plot")
fitt = lm(res ~ as.numeric(1:length(res))); abline(fitt, col="red")
abline(h=mean(res), col="blue")
```



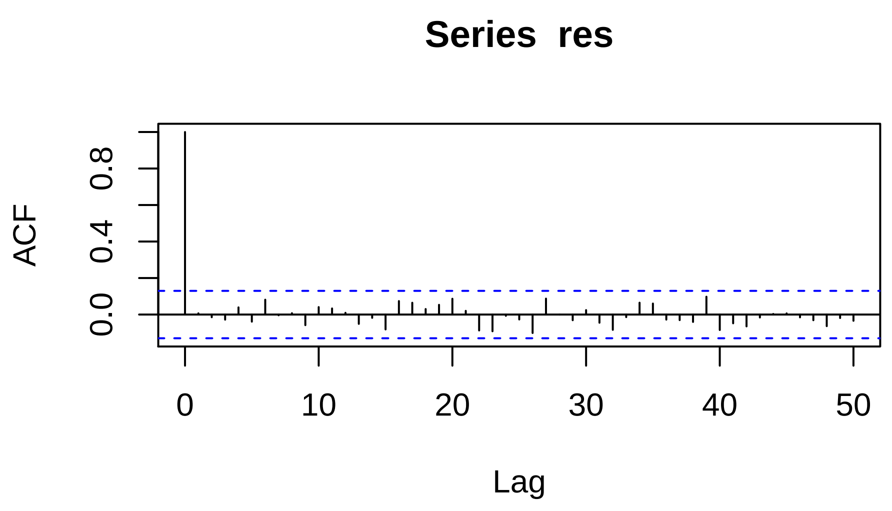
The plot of the residuals resembles White Noise and has little to no trend, no seasonality, and no visible change in variance. The sample mean of the residuals (-0.0009784136) is also almost 0. Since the residuals pass this check, we continue.

```
#Q-Q plot of residuals
qqnorm(res,main="Normal Q-Q Plot for Model B Residuals")
qqline(res,col="blue")
```

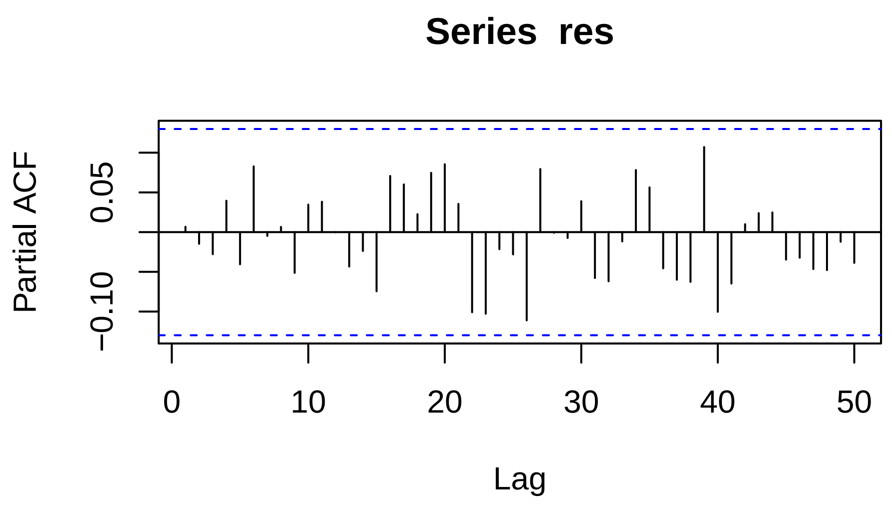


The Normal Q-Q plot for the residuals also seems fine due to at least 95 percent of the values falling within 2 standard deviations of the mean and those values fitting the line fairly well. Next, we plot and analyze the ACF and PACF of the residuals:

```
#plot acf of residuals
acf(res, lag.max=50)
```



```r
#plot pacf of residuals
pacf(res, lag.max=50)
```



All ACF and PACF of the residuals are within the confidence intervals and can be counted as zeros, indicating that the residuals follow a White Noise distribution. We then continue by conducting the Shapiro-Wilk test of normality and several Portmanteau tests on the residuals:

```r
#conduct Shapiro-Wilk test
shapiro.test(res)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res
## W = 0.99381, p-value = 0.4676
```

```r
#conduct Box-Pierce test                              #h-(p+q) = 15-10 = 5
Box.test(res, lag=15, type=c("Box-Pierce"), fitdf=10) #lag approximately sqrt(n=240)
```

```
##
##  Box-Pierce test
##
## data:  res
## X-squared = 6.075, df = 5, p-value = 0.299
```

```
#conduct Ljung-Box test
Box.test(res, lag=15, type=c("Ljung-Box"), fitdf=10) #lag approximately sqrt(n=240)
```

```
##
##  Box-Ljung test
##
## data:  res
## X-squared = 6.3996, df = 5, p-value = 0.2693
```

```
#conduct McLeod-Li test
Box.test(res^2, lag=15, type=c("Ljung-Box"), fitdf=0)
```

```
##
##  Box-Ljung test
##
## data:  res^2
## X-squared = 17.759, df = 15, p-value = 0.2755
```

Since the residuals have p-values greater than $\alpha = 0.05$ for all of these tests, the residuals pass this diagnostic check. We then proceed to our final diagnostic check of finding the AR order that the residuals are fitted to:

```
#find AR order for the residuals
ar(res, aic=TRUE, order.max=NULL, method=c("yule-walker"))
```

```
##
## Call:
## ar(x = res, aic = TRUE, order.max = NULL, method = c("yule-walker"))
##
##
## Order selected 0  sigma^2 estimated as  0.0004745
```

The residuals were fitted to AR(0), which is equivalent to White Noise. Thus, we finish diagnostic checking and can conclude that our best model is ready to be used for forecasting.

Final Model for the logarithm transformation of original data:

$\ln(U_t)$ follows a $SARIMA(5,1,3) \times (3,1,1)_{12}$ model

$\Delta_1\Delta_{12}(1-0.6586_{(0.1104)}B-0.0401_{(0.0855)}B^4-0.2460_{(0.0837)}B^5)(1+0.3668_{(0.0739)}B^{12}+0.2927_{(0.0789)}B^{24}+0.2458_{(0.0771)}B^{36})ln(U_t) = (1-0.7675_{(0.1245)}B+0.2978_{(0.0873)}B^2-0.0514_{(0.0780)}B^3)(1-0.9999_{(0.1074)}B^{12})Z_t$

$\sigma_Z^2 = 0.0005019$

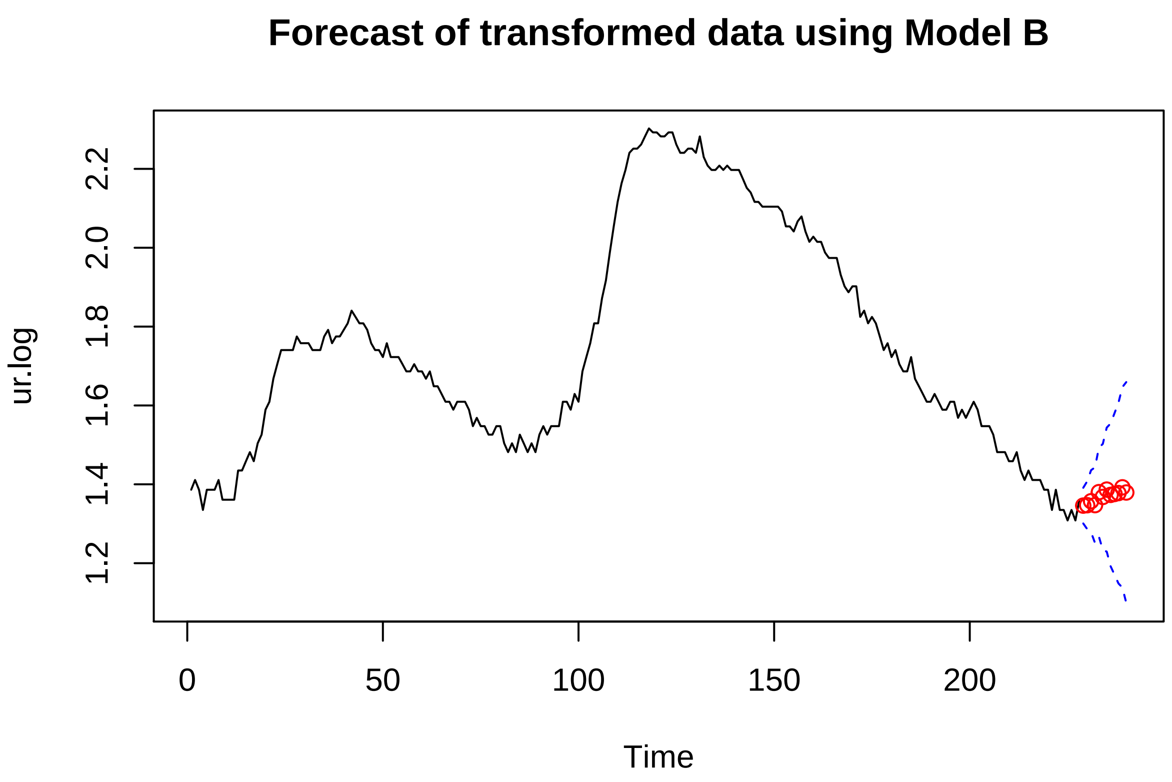
Use chosen model for forecasting

After confirming that our final model passes all diagnostic checks, we proceed with forecasting the last 12 years of the unemployment rate dataset:

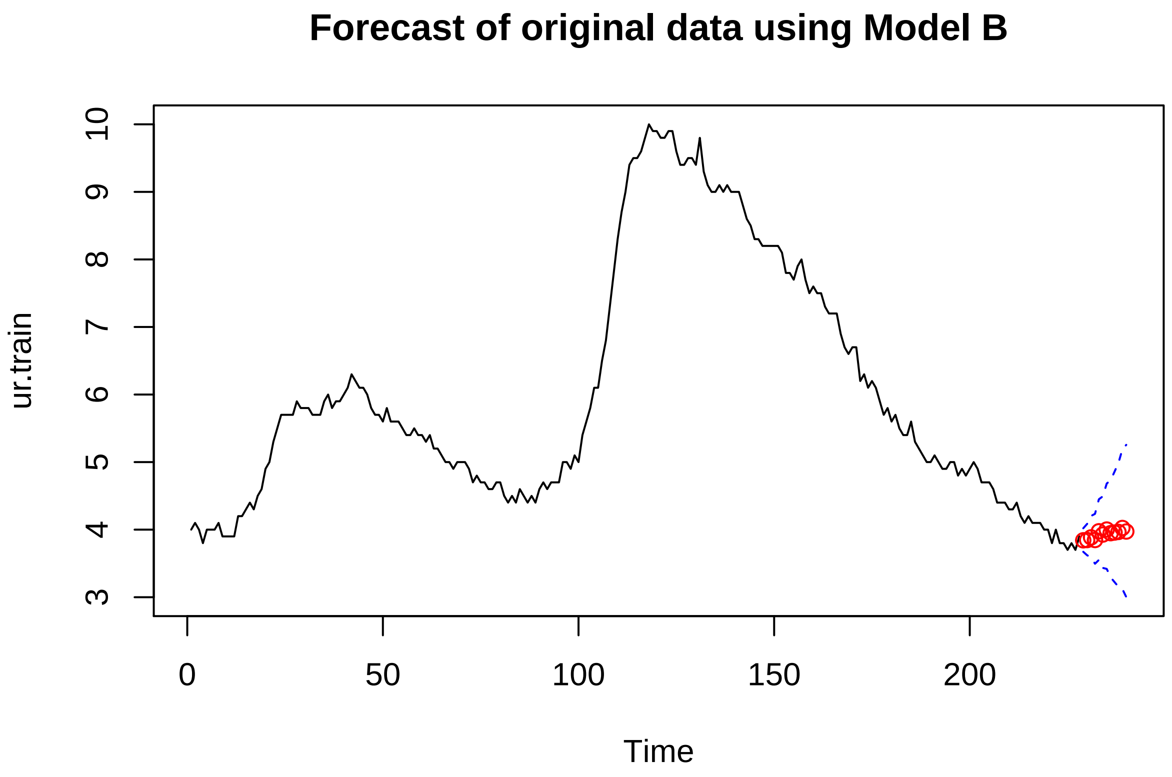
```r
#print forecasts with prediction bounds in a table
fit.B=arima(ur.log, order=c(5,1,3), seasonal=list(order=c(3,1,1), period=12),
            fixed=c(NA,0,0,NA,NA,NA,NA,NA,NA,NA,NA,NA), method="ML")
forecast(fit.B)
```

```
##     Point Forecast    Lo 80    Hi 80     Lo 95    Hi 95
## 229       1.346034 1.316476 1.375592 1.3008286 1.391239
## 230       1.347043 1.307496 1.386589 1.2865613 1.407524
## 231       1.357146 1.305691 1.408602 1.2784523 1.435840
## 232       1.347219 1.284556 1.409881 1.2513849 1.443052
## 233       1.380197 1.306494 1.453901 1.2674773 1.492917
## 234       1.368046 1.280140 1.455952 1.2336047 1.502487
## 235       1.386603 1.283770 1.489436 1.2293335 1.543872
## 236       1.373193 1.254798 1.491587 1.1921237 1.554261
## 237       1.375557 1.241457 1.509657 1.1704692 1.580645
## 238       1.377854 1.227992 1.527716 1.1486599 1.607048
## 239       1.392386 1.226155 1.558617 1.1381574 1.646614
## 240       1.379253 1.196151 1.562356 1.0992223 1.659284
## 241       1.381747 1.184888 1.578606 1.0806770 1.682817
## 242       1.382659 1.171085 1.594233 1.0590849 1.706234
## 243       1.391731 1.165742 1.617720 1.0461105 1.737351
## 244       1.391493 1.150895 1.632091 1.0235300 1.759455
## 245       1.393413 1.138031 1.648795 1.0028402 1.783986
## 246       1.394337 1.124628 1.664046 0.9818524 1.806822
## 247       1.398152 1.114193 1.682112 0.9638746 1.832430
## 248       1.397063 1.098947 1.695179 0.9411334 1.852992
## 249       1.401473 1.089206 1.713741 0.9239021 1.879045
## 250       1.398964 1.072521 1.725407 0.8997124 1.898216
## 251       1.397429 1.056892 1.737967 0.8766218 1.918236
## 252       1.392386 1.037829 1.746943 0.8501383 1.934634
```

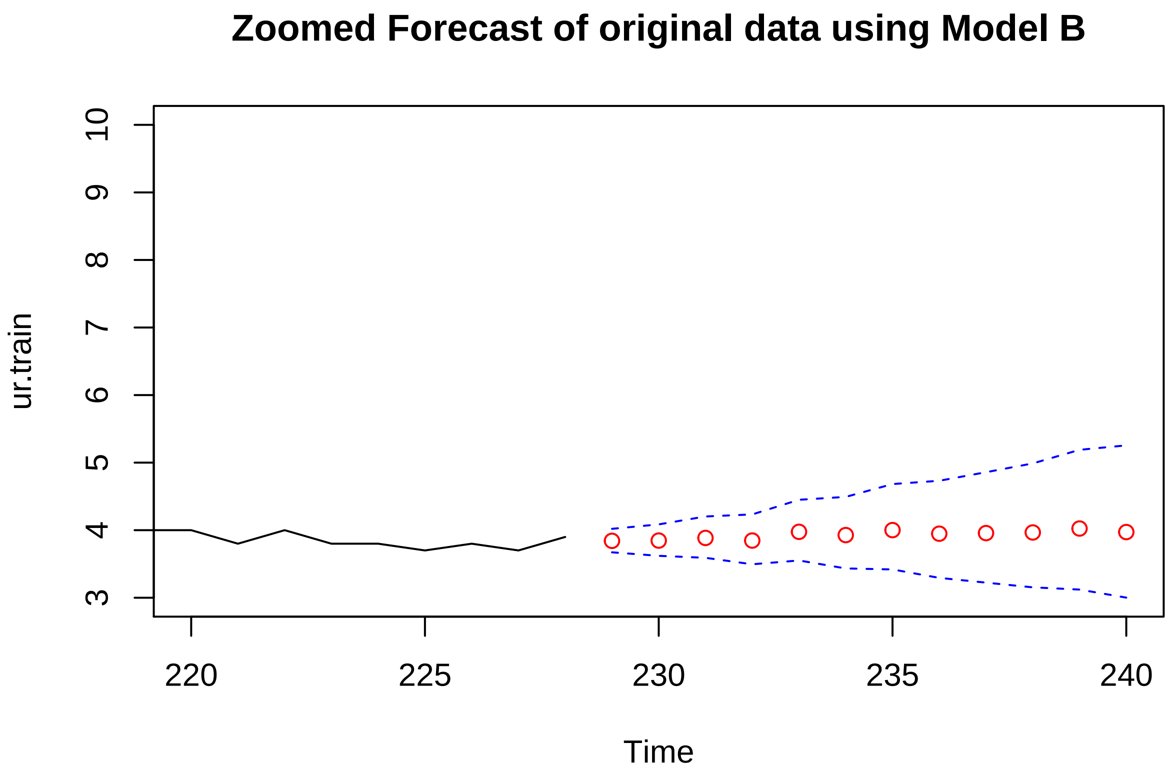
We then plot these forecasted values on the log transformed data:



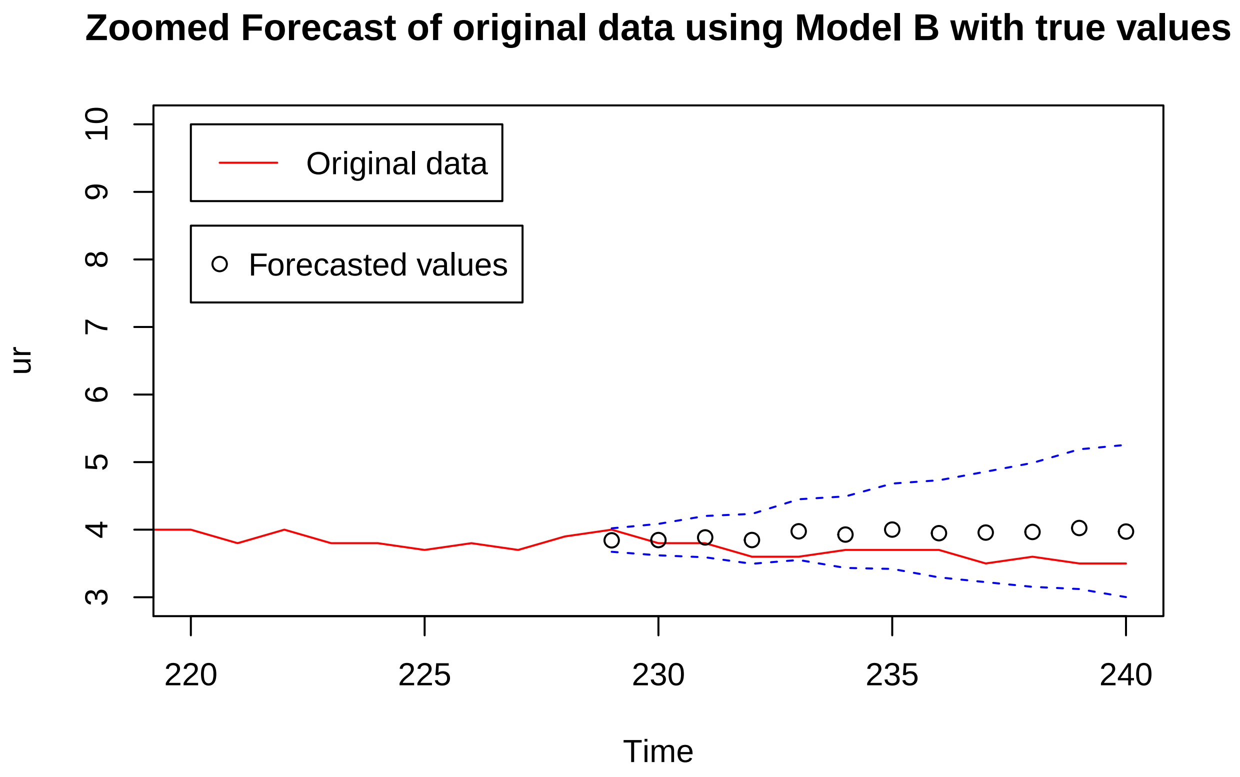
We then perform the inverse operation (exp()) on the log transformed values to plot the forecasted values on the original data:



We then zoom in on the forecasted values in the plot with original data, starting with entry 220:



Lastly, we plot both the true and forecasted values for the last 12 values in the original dataset on the zoomed plot with original data:



From the final plot that shows a close-up of the true and forecasted values for the test dataset, we see that the true test values are all within the 95 percent prediction interval and that the forecasted values are fairly close to the true test values. Thus, we can consider our forecasting a success and conclude that our final model $SARIMA(5,1,3) \times (3,1,1)_{12}$ for the logarithm transformation of the original data can be used to adequately forecast future values of U.S. unemployment rate data.

Conclusion

In the end, through the use of data-splitting, transformations and differencing, ACF and PACF analyses, AIC(C) value comparisons, and diagnostic checks, we were able to successfully achieve our goal of finding a SARIMA model that is able to successfully forecast unemployment rates in the U.S. based on preexisting data gathered by the U.S. Bureau of Labor Statistics. Our final model, given by the equation

$\Delta_1\Delta_{12}(1-0.6586_{(0.1104)}B-0.0401_{(0.0855)}B^4-0.2460_{(0.0837)}B^5)(1+0.3668_{(0.0739)}B^{12}+0.2927_{(0.0789)}B^{24}+0.2458_{(0.0771)}B^{36})ln(U_t) = (1-0.7675_{(0.1245)}B+0.2978_{(0.0873)}B^2-0.0514_{(0.0780)}B^3)(1-0.9999_{(0.1074)}B^{12})Z_t$ with $\sigma_Z^2 = 0.0005019$,

was able to forecast the true values of the test set within a 95 percent prediction interval. As such, we consider this time series forecasting project a success and conclude that a SARIMA model can in fact be used to adequately forecast future U.S. unemployment rates based on preexisting data.

References

"Top Picks (Most Requested Statistics)." U.S. Bureau of Labor Statistics, U.S. Bureau of Labor Statistics, data.bls.gov/cgi-bin/surveymost?bls.

Appendix

```r
library(tidyr)
library(tidyverse)
library(dplyr)
library(tseries)
library(MASS)
library(astsa)
library(ggplot2)
library(ggfortify)
library(forecast)
#read in dataset
usur.csv = read.csv("usur.csv")
#set unemployment rate from dataset as vector
ur = as.vector(usur.csv$Value)
#remove any NA values
ur = (na.omit(ur))
#plot raw data
ur.ts = ts(ur, start = 2000, end = 2019, frequency=12)
#ts.plot(ur.ts, main="Raw Data")
#create t variable
t = as.numeric(1:length(ur))
#fit unemployment rate data
fit = lm(ur ~ t)
#plot unemployment rate data with months on x-axis
plot.ts(ur, xlab="Time (Months)", ylab="Unemployment Rate",
        main="U.S. Unemployment Rates (2000-2019)")
#add trend line to plot
abline(fit, col="red")
#add constant mean to plot
#mean(ur) #5.88
abline(h=mean(ur), col="blue")
#create training dataset
ur.train = ur[c(1:228)]
#create test dataset
ur.test = ur[c(229:240)]
#plot training set
plot.ts(ur.train, xlab="Time (Months)", ylab="Unemployment Rate", main="Training Dataset")
#add training set trendline
fit = lm(ur.train ~ as.numeric(1:length(ur.train))); abline(fit, col="red")
#add training set mean line
abline(h=mean(ur.train), col="blue")
op = par(mfrow=c(2,2))
#create training set histogram
hist(ur.train, col="light blue", xlab="", main="Histogram; Unemployment Rate Data")
#plot training set acf
acf(ur.train,lag.max=50, main="ACF of Unemployment Rate Data")
par(op)
#box-cox transformation
bcTransform = boxcox(ur.train ~ as.numeric(1:length(ur.train)),plotit = TRUE)
lambda = bcTransform$x[which(bcTransform$y == max(bcTransform$y))]
ur.bc = (1/lambda)*(ur^lambda-1)
#log transformation
ur.log = log(ur.train)
#square root transformation
ur.sqrt = sqrt(ur.train)
#plot original & transformations
ts.plot(ur.train, main = "Original TS")
ts.plot(ur.log, main = "ln Transformation")
ts.plot(ur.bc, main = "Box-Cox Transformation")
ts.plot(ur.sqrt, main = "Square Root Transformation")
#plot original & transformation histograms
op = par(mfrow=c(2,2))
hist(ur.train, col="light blue", xlab="", main="Histogram; U_t")
hist(ur.bc, col="light blue", xlab="", main="Histogram; bc(U_t)")
hist(ur.log, col="light blue", xlab="", main="Histogram; ln(U_t)")
hist(ur.sqrt, col="light blue", xlab="", main="Histogram; sqrt(U_t)")
par(op)
#produce decomposition of ln(U_t)
y = ts(as.ts(ur.log), frequency = 12)
decomposition = decompose(y)
plot(decomposition)
#check suitability of ln(U_t) transformation:
#perform differencing at lag=12 to remove seasonal component
ur.log_12 = diff(ur.log,lag=12)
#plot ln-transformed training set differenced at lag=12
plot.ts(ur.log_12, xlab="Time", ylab="ln(UR)", main="ln(U_t) differenced at lag 12")
fit = lm(ur.log_12 ~ as.numeric(1:length(ur.log_12))); abline(fit, col="red")
#mean(ur.log_12)
```

```
abline(h=mean(ur.log_12), col="blue")
#calculate variance before differencing
var(ur.log)
#calculate variance after differencing at lag=12
var(ur.log_12) #variance is lower!
#perform differencing at lag=1 to remove trend component
diff.ur.log = diff(ur.log_12,lag=1)
#plot ln transformed training set differenced at lag=12 and lag=1
plot.ts(diff.ur.log, xlab="Time", ylab="ln(UR)", main="ln(U_t) differenced at lags 12 and 1")
fit = lm(diff.ur.log ~ as.numeric(1:length(diff.ur.log))); abline(fit, col="red")
#mean(diff.ur.log)
abline(h=mean(diff.ur.log), col="blue")
#diff.ur.log is ln transformed truncated data, differenced at lags 12 and then 1
#calculate variance after differencing at lag=12 and lag=1
var(diff.ur.log) #variance is lower than when only differenced at lag=12
op = par(mfrow=c(2,2))
#plot acfs
acf(ur.log,lag.max=50,main="ACF; ln(U_t)")
acf(ur.log_12,lag.max=50,main="ACF; ln(U_t), differenced at lag 12")
par(op)
acf(diff.ur.log,lag.max=50,main="ACF; ln(U_t), differenced at lags 12 and 1")
#plot differenced U_t histogram
op = par(mfrow=c(2,2))
hist(diff.ur.log, col="light blue", xlab="", main="ln(U_t) differenced at lags 12 & 1")
#plot histogram of transformed and differenced data with normal curve
hist(diff.ur.log, density=20,breaks=20, col="blue", xlab="", prob=TRUE)
m = mean(diff.ur.log)
std = sqrt(var(diff.ur.log))
curve(dnorm(x,m,std), add=TRUE)
par(op)
#produce decomposition of bc(U_t)
y = ts(as.ts(ur.bc), frequency = 12)
decomposition = decompose(y)
plot(decomposition)
#check suitability of bc(U_t) transformation:
#calculate variance before differencing
#var(ur.bc)
#perform differencing at lag=12 to remove seasonal component
ur.bc_12 = diff(ur.bc,lag=12)
#calculate variance after differencing at lag=12
#var(ur.bc_12)
op = par(mfrow=c(2,2))
#plot box-cox transformed training set differenced at lag=12
plot.ts(ur.bc_12, xlab="Time", ylab="bc(UR)", main="BC(U_t) differenced at lag 12")
fit = lm(ur.bc_12 ~ as.numeric(1:length(ur.bc_12))); abline(fit, col="red")
#mean(ur.bc_12)
abline(h=mean(ur.bc_12), col="blue")
#perform differencing at lag=1 to remove trend component
diff.ur.bc = diff(ur.bc_12,lag=1)
#calculate variance after differencing at lag=12 and lag=1
#var(diff.ur.bc)
#plot box-cox transformed training set differenced at lag=12 and lag=1
plot.ts(diff.ur.bc, xlab="Time", ylab="bc(UR)", main="bc(U_t) differenced at lag 12s and 1")
```

```r
fit = lm(diff.ur.bc ~ as.numeric(1:length(diff.ur.bc))); abline(fit, col="red")
#mean(diff.ur.bc)
abline(h=mean(diff.ur.bc), col="blue")
par(op)
#diff.ur.bc is box-cox transformed truncated data, differenced at lags 12 and then 1
#calculate variance before differencing
var(ur.bc)
#calculate variance after differencing at lag=12
var(ur.bc_12) #this is lower!
#calculate variance after differencing at lag=12 and lag=1
var(diff.ur.bc) #this is lower!
op = par(mfrow=c(2,2))
#plot acfs
acf(ur.bc,lag.max=50,main="ACF; bc(U_t)")
acf(ur.bc_12,lag.max=50,main="ACF; bc(U_t), differenced at lag 12")
acf(diff.ur.bc,lag.max=50,main="ACF; bc(U_t), differenced at lags 12 and 1")
#plot differenced U_t histogram
op = par(mfrow=c(2,2))
hist(diff.ur.bc, col="light blue", xlab="", main="bc(U_t) differenced at lags 12 & 1")
#plot histogram of transformed and differenced data with normal curve
hist(diff.ur.bc, density=20,breaks=20, col="blue", xlab="", prob=TRUE)
m = mean(diff.ur.bc)
std = sqrt(var(diff.ur.bc))
curve(dnorm(x,m,std), add=TRUE)
par(op)
op = par(mfrow=c(2,2))
#plot histogram of log transformed and differenced data with normal curve
hist(diff.ur.log, density=20,breaks=20, col="blue", xlab="", prob=TRUE)
m = mean(diff.ur.log)
std = sqrt(var(diff.ur.log))
curve(dnorm(x,m,std), add=TRUE)
#plot histogram of box-cox transformed and differenced data with normal curve
hist(diff.ur.bc, density=20,breaks=20, col="blue", xlab="", prob=TRUE)
m = mean(diff.ur.bc)
std = sqrt(var(diff.ur.bc))
curve(dnorm(x,m,std), add=TRUE)
par(op)
#perform Shapiro-Wilk test of normality
shapiro.test(diff.ur.bc)
shapiro.test(diff.ur.log)
#plot acf and pacf for differenced ln(U_t)
acf(diff.ur.log, lag.max=50, main="ACF; ln(U_t), differenced at lags 12 and 1")
pacf(diff.ur.log, lag.max=50, main="PACF; ln(U_t), differenced at lags 12 and 1")
#AICc() function
AICc <- function(object)
{
  aic <- AIC(object)
  if (!is.numeric(aic)) stop("Cannot calculate AIC!")
  k <- length(coef(object))
  n <- length(residuals(object))
    aic + ((2 * k * (k + 1))/(n - k - 1))
}
#calculate AICc for all SAR possibilities
```

```r
AICc(arima(ur.log, order=c(3,1,0), seasonal=list(order=c(1,1,0), period=12), method="ML"))
AICc(arima(ur.log, order=c(5,1,0), seasonal=list(order=c(1,1,0), period=12), method="ML"))
AICc(arima(ur.log, order=c(3,1,0), seasonal=list(order=c(2,1,0), period=12), method="ML"))
AICc(arima(ur.log, order=c(5,1,0), seasonal=list(order=c(2,1,0), period=12), method="ML"))
AICc(arima(ur.log, order=c(3,1,0), seasonal=list(order=c(3,1,0), period=12), method="ML"))
AICc(arima(ur.log, order=c(5,1,0), seasonal=list(order=c(3,1,0), period=12), method="ML"))
#-887.6405
#calculate AICc for all SMA possibilities
AICc(arima(ur.log, order=c(0,1,3), seasonal=list(order=c(0,1,0), period=12), method="ML"))
AICc(arima(ur.log, order=c(0,1,5), seasonal=list(order=c(0,1,0), period=12), method="ML"))
AICc(arima(ur.log, order=c(0,1,3), seasonal=list(order=c(0,1,1), period=12), method="ML"))
AICc(arima(ur.log, order=c(0,1,5), seasonal=list(order=c(0,1,1), period=12), method="ML"))
#-911.0144
#calculate AICc for all SARIMA possibilities
AICc(arima(ur.log, order=c(3,1,3), seasonal=list(order=c(1,1,0), period=12), method="ML"))
AICc(arima(ur.log, order=c(3,1,5), seasonal=list(order=c(1,1,0), period=12), method="ML"))
AICc(arima(ur.log, order=c(5,1,3), seasonal=list(order=c(1,1,0), period=12), method="ML"))
AICc(arima(ur.log, order=c(5,1,5), seasonal=list(order=c(1,1,0), period=12), method="ML"))
AICc(arima(ur.log, order=c(3,1,3), seasonal=list(order=c(2,1,0), period=12), method="ML"))
AICc(arima(ur.log, order=c(3,1,5), seasonal=list(order=c(2,1,0), period=12), method="ML"))
AICc(arima(ur.log, order=c(5,1,3), seasonal=list(order=c(2,1,0), period=12), method="ML"))
AICc(arima(ur.log, order=c(5,1,5), seasonal=list(order=c(2,1,0), period=12), method="ML"))
AICc(arima(ur.log, order=c(3,1,3), seasonal=list(order=c(3,1,0), period=12), method="ML"))
#-896.7059
AICc(arima(ur.log, order=c(3,1,5), seasonal=list(order=c(3,1,0), period=12), method="ML"))
#-899.3479
AICc(arima(ur.log, order=c(5,1,3), seasonal=list(order=c(3,1,0), period=12), method="ML"))
#-896.0036
#AICc(arima(ur.log, order=c(5,1,5), seasonal=list(order=c(3,1,0), period=12), method="ML"))
AICc(arima(ur.log, order=c(3,1,3), seasonal=list(order=c(1,1,1), period=12), method="ML"))
AICc(arima(ur.log, order=c(3,1,5), seasonal=list(order=c(1,1,1), period=12), method="ML"))
AICc(arima(ur.log, order=c(5,1,3), seasonal=list(order=c(1,1,1), period=12), method="ML"))
AICc(arima(ur.log, order=c(5,1,5), seasonal=list(order=c(1,1,1), period=12), method="ML"))
AICc(arima(ur.log, order=c(3,1,3), seasonal=list(order=c(2,1,1), period=12), method="ML"))
AICc(arima(ur.log, order=c(3,1,5), seasonal=list(order=c(2,1,1), period=12), method="ML"))
#-937.1948
AICc(arima(ur.log, order=c(5,1,3), seasonal=list(order=c(2,1,1), period=12), method="ML"))
#-937.5835
#AICc(arima(ur.log, order=c(5,1,5), seasonal=list(order=c(2,1,1), period=12), method="ML"))
AICc(arima(ur.log, order=c(3,1,3), seasonal=list(order=c(3,1,1), period=12), method="ML"))
AICc(arima(ur.log, order=c(3,1,5), seasonal=list(order=c(3,1,1), period=12), method="ML"))
#-949.053
AICc(arima(ur.log, order=c(5,1,3), seasonal=list(order=c(3,1,1), period=12), method="ML"))
#-945.2402
#AICc(arima(ur.log, order=c(5,1,5), seasonal=list(order=c(3,1,1), period=12), method="ML"))
#SAR model with lowest AICc
arima(ur.log, order=c(5,1,0), seasonal=list(order=c(3,1,0), period=12), method="ML")
AICc(arima(ur.log, order=c(5,1,0), seasonal=list(order=c(3,1,0), period=12), method="ML"))
#test SAR model with lowest AICc (fixed 0's)
AICc(arima(ur.log, order=c(5,1,0), seasonal=list(order=c(3,1,0), period=12),
           fixed=c(0,NA,NA,NA,NA,NA,NA,NA), method="ML"))
#AICc = -889.4854 < -887.6405, so set AR(1) coef to 0
AICc(arima(ur.log, order=c(5,1,0), seasonal=list(order=c(3,1,0), period=12),
           fixed=c(0,NA,NA,0,NA,NA,NA,NA), method="ML"))
#AICc = -890.2496 < -889.4854, so set AR(4) coef to 0 (best SAR model)
#best SAR model
arima(ur.log, order=c(5,1,0), seasonal=list(order=c(3,1,0), period=12),
      fixed=c(0,NA,NA,0,NA,NA,NA,NA), method="ML")
AICc(arima(ur.log, order=c(5,1,0), seasonal=list(order=c(3,1,0), period=12),
           fixed=c(0,NA,NA,0,NA,NA,NA,NA), method="ML"))
#SMA model with lowest AICc
arima(ur.log, order=c(0,1,5), seasonal=list(order=c(0,1,1), period=12), method="ML")
AICc(arima(ur.log, order=c(0,1,5), seasonal=list(order=c(0,1,1), period=12), method="ML"))
#test SMA model with lowest AICc (fixed 0's)
AICc(arima(ur.log, order=c(0,1,5), seasonal=list(order=c(0,1,1), period=12),
           fixed=c(0,NA,NA,NA,NA,NA), method="ML"))
#AICc = -912.2036 < -911.0144, so set MA(1) coef to 0
AICc(arima(ur.log, order=c(0,1,5), seasonal=list(order=c(0,1,1), period=12),
           fixed=c(0,NA,0,NA,NA,NA), method="ML"))
#AICc = -910.9523 > -912.2036, so don't set MA(2) coef to 0
AICc(arima(ur.log, order=c(0,1,5), seasonal=list(order=c(0,1,1), period=12),
           fixed=c(0,NA,NA,0,NA,NA), method="ML"))
#AICc = -913.1596 < -912.2036, so set MA(4) coef to 0 (best SMA model)
#best SMA model
arima(ur.log, order=c(0,1,5), seasonal=list(order=c(0,1,1), period=12),
      fixed=c(0,NA,NA,0,NA,NA), method="ML")
AICc(arima(ur.log, order=c(0,1,5), seasonal=list(order=c(0,1,1), period=12),
           fixed=c(0,NA,NA,0,NA,NA), method="ML"))
#SARIMA with lowest AICc
arima(ur.log, order=c(3,1,5), seasonal=list(order=c(3,1,1), period=12), method="ML")
AICc(arima(ur.log, order=c(3,1,5), seasonal=list(order=c(3,1,1), period=12), method="ML"))
#AICc = -949.053
#SARIMA with second lowest AICc
arima(ur.log, order=c(5,1,3), seasonal=list(order=c(3,1,1), period=12), method="ML")
AICc(arima(ur.log, order=c(5,1,3), seasonal=list(order=c(3,1,1), period=12), method="ML"))
#AICc = -945.2402
#test SARIMA(3,1,5)x(3,1,1)_{12} (fixed 0's)
AICc(arima(ur.log, order=c(3,1,5), seasonal=list(order=c(3,1,1), period=12),
           fixed=c(NA,NA,NA,NA,NA,NA,NA,0,NA,NA,NA,NA), method="ML"))
#-946.9846 > -949.053, so don't set MA(5) coef to 0
#test SARIMA(5,1,3)x(3,1,1)_{12} (fixed 0's)
AICc(arima(ur.log, order=c(5,1,3), seasonal=list(order=c(3,1,1), period=12),
           fixed=c(NA,0,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA), method="ML"))
#-945.8834 < -945.2402 but sma1 coef > |1|
AICc(arima(ur.log, order=c(5,1,3), seasonal=list(order=c(3,1,1), period=12),
           fixed=c(NA,0,0,NA,NA,NA,NA,NA,NA,NA,NA,NA), method="ML"))
#-947.7885 < -945.8834 and sma1 < |1|
AICc(arima(ur.log, order=c(5,1,3), seasonal=list(order=c(3,1,1), period=12),
           fixed=c(NA,0,0,0,NA,NA,NA,NA,NA,NA,NA,NA), method="ML"))
#-949.5741 < -947.7885 but sma = |1|
AICc(arima(ur.log, order=c(5,1,3), seasonal=list(order=c(3,1,1), period=12),
           fixed=c(NA,0,0,0,NA,NA,NA,0,NA,NA,NA,NA), method="ML"))
#-951.3114 < -949.5741 but sma > |1|
#we choose the best model with sma1 coef < |1|
arima(ur.log, order=c(5,1,3), seasonal=list(order=c(3,1,1), period=12),
      fixed=c(NA,0,0,NA,NA,NA,NA,NA,NA,NA,NA,NA), method="ML")
```

```r
#second-best SARIMA model (A)
arima(ur.log, order=c(3,1,5), seasonal=list(order=c(3,1,1), period=12), method="ML")
AICc(arima(ur.log, order=c(3,1,5), seasonal=list(order=c(3,1,1), period=12), method="ML"))
#best SARIMA model (B)
arima(ur.log, order=c(5,1,3), seasonal=list(order=c(3,1,1), period=12),
      fixed=c(NA,0,0,NA,NA,NA,NA,NA,NA,NA,NA,NA), method="ML")
AICc(arima(ur.log, order=c(5,1,3), seasonal=list(order=c(3,1,1), period=12),
           fixed=c(NA,0,0,NA,NA,NA,NA,NA,NA,NA,NA,NA), method="ML"))
#fit Model A
fit.A = arima(ur.log, order=c(3,1,5), seasonal=list(order=c(3,1,1), period=12), method="ML")
#fit Model B
fit.B = arima(ur.log, order=c(5,1,3), seasonal=list(order=c(3,1,1), period=12),
              fixed=c(NA,0,0,NA,NA,NA,NA,NA,NA,NA,NA,NA), method="ML")
#import plot.roots function
source("plot.roots.R")
#plot.roots() function
plot.roots <- function(ar.roots=NULL, ma.roots=NULL, size=2, angles=FALSE, special=NULL, sqecial=NULL,m
{xylims <- c(-size,size)
      omegas <- seq(0,2*pi,pi/500)
      temp <- exp(complex(real=rep(0,length(omegas)),imag=omegas))
      plot(Re(temp),Im(temp),typ="l",xlab="x",ylab="y",xlim=xylims,ylim=xylims,main=main)
      abline(v=0,lty="dotted")
      abline(h=0,lty="dotted")
      if(!is.null(ar.roots))
        {
          points(Re(1/ar.roots),Im(1/ar.roots),col=first.col,pch=my.pch)
          points(Re(ar.roots),Im(ar.roots),col=second.col,pch=my.pch)
        }
      if(!is.null(ma.roots))
        {
          points(Re(1/ma.roots),Im(1/ma.roots),pch="*",cex=1.5,col=first.col)
          points(Re(ma.roots),Im(ma.roots),pch="*",cex=1.5,col=second.col)
        }
      if(angles)
        {
          if(!is.null(ar.roots))
            {
              abline(a=0,b=Im(ar.roots[1])/Re(ar.roots[1]),lty="dotted")
              abline(a=0,b=Im(ar.roots[2])/Re(ar.roots[2]),lty="dotted")
            }
          if(!is.null(ma.roots))
            {
              sapply(1:length(ma.roots), function(j) abline(a=0,b=Im(ma.roots[j])/Re(ma.roots[j]),lty="
            }
        }
      if(!is.null(special))
        {
          lines(Re(special),Im(special),lwd=2)
        }
      if(!is.null(sqecial))
        {
          lines(Re(sqecial),Im(sqecial),lwd=2)
        }
```

```r
}
#check invertibility/stationarity of Model (A)
plot.roots(polyroot(c(1,-1.4645,1.4109,-0.9120)),
           polyroot(c(1,-1.6027,1.8477,-1.2806,0.3320,0.0335)),
           main="(A) roots of ar and ma part, nonseasonal")
plot.roots(polyroot(c(1,0.3532,0.3260,0.2514)), main="(A) roots of sar part, seasonal")
#check invertibility/stationarity of Model (B)
plot.roots(polyroot(c(1,-0.6586,0,0,-0.0401,-0.2460)),
           polyroot(c(1,-0.7675,0.2978,-0.0514)),
           main="(B) roots of ar and ma part, nonseasonal")
plot.roots(polyroot(c(1,0.3668,0.2927,0.2458)), main="(B) roots of sar part, seasonal")
#analysis of residuals for Model B
#fit residuals
res = residuals(fit.B)
#plot histogram of residuals
hist(as.numeric(res), density=20, breaks=20, col="blue", xlab="", prob=TRUE, main="Histogram; residuals
m = mean(res)
std = sqrt(var(res))
curve(dnorm(x,m,std), add=TRUE)
#plot residuals
plot.ts(res, main="Model B Residuals Plot")
fitt = lm(res ~ as.numeric(1:length(res))); abline(fitt, col="red")
abline(h=mean(res), col="blue")
#Q-Q plot of residuals
qqnorm(res,main="Normal Q-Q Plot for Model B Residuals")
qqline(res,col="blue")
#plot acf of residuals
acf(res, lag.max=50)
#plot pacf of residuals
pacf(res, lag.max=50)
#conduct Shapiro-Wilk test
shapiro.test(res)
#conduct Box-Pierce test                              #h-(p+q) = 15-10 = 5
Box.test(res, lag=15, type=c("Box-Pierce"), fitdf=10) #lag approximately sqrt(n=240)
#conduct Ljung-Box test
Box.test(res, lag=15, type=c("Ljung-Box"), fitdf=10) #lag approximately sqrt(n=240)
#conduct McLeod-Li test
Box.test(res^2, lag=15, type=c("Ljung-Box"), fitdf=0)
#find AR order for the residuals
ar(res, aic=TRUE, order.max=NULL, method=c("yule-walker"))
#print forecasts with prediction bounds in a table
fit.B=arima(ur.log, order=c(5,1,3), seasonal=list(order=c(3,1,1), period=12), fixed=c(NA,0,0,NA,NA,NA,N
#plot with 12 forecasts on ln transformed data:
pred.tr=predict(fit.B, n.ahead=12)
U.tr=pred.tr$pred+1.96*pred.tr$se  #upper bound of prediction interval
L.tr=pred.tr$pred-1.96*pred.tr$se  #lower bound
ts.plot(ur.log, xlim=c(1,length(ur.log)+12), ylim = c(1.1,2.3),
        main="Forecast of transformed data using Model B")
lines(U.tr, col="blue", lty="dashed")
lines(L.tr, col="blue", lty="dashed")
points((length(ur.log)+1):(length(ur.log)+12), pred.tr$pred,col="red")
#plot with forecasts on original data
pred.orig=exp(pred.tr$pred)
```

```r
U=exp(U.tr)
L=exp(L.tr)
ts.plot(ur.train, xlim=c(1,length(ur.train)+12), ylim=c(3,10),
        main="Forecast of original data using Model B")
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points((length(ur.train)+1):(length(ur.train)+12), pred.orig, col="red")
#zoom plot starting at entry 220
ts.plot(ur.train, xlim = c(220,length(ur.train)+12), ylim=c(3,10),
        main="Zoomed Forecast of original data using Model B")
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points((length(ur.train)+1):(length(ur.train)+12), pred.orig, col="red")
#plot zoomed forecasts and true values
ts.plot(ur, xlim = c(220,length(ur.train)+12), ylim=c(3,10), col="red",
        main="Zoomed Forecast of original data using Model B with true values")
legend(220,10, legend="Original data", col="red", lty=1, cex=1)
legend(220,9, legend="Forecasted values", col="black", pch=1, cex=1)
lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")
points((length(ur.train)+1):(length(ur.train)+12), pred.orig, col="black")
```