STA 137 Final Project

Zhuorui He, Shiyu Wu, Sulei Wang, Zhan Shi

2024-03-12

Contents

Introduction

This study examines the annual exports data of the Central African Republic, represented as a percentage of its total GDP from 1960 to 2017. Tackling real-world datasets often involves navigating their inherent complexity, as they seldom present themselves in a straightforward manner. The methodology employed here includes applying decomposition and transformations to facilitate more effective analysis, identifying the optimal ARIMA parameters, and performing residual diagnostics to ensure model reliability. The ultimate goal is to develop an ARIMA model capable of forecasting future exports with a reasonable degree of precision.

Background

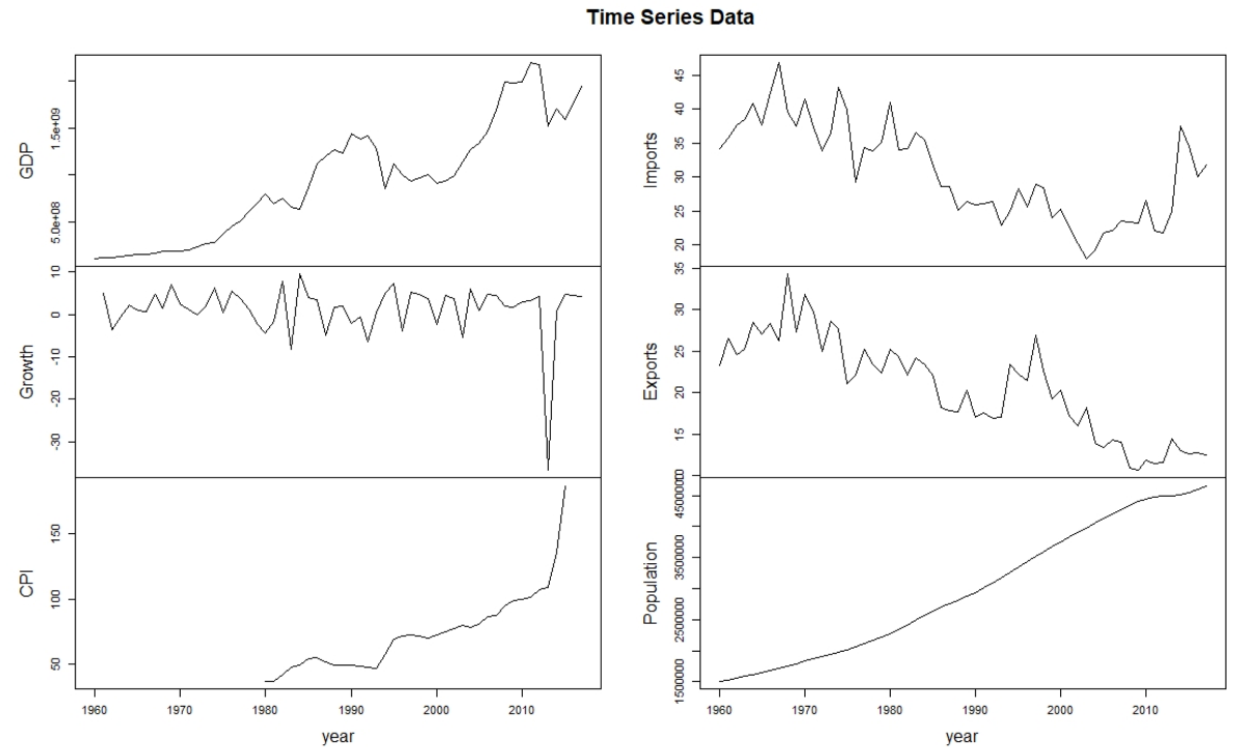
The Central African Republic (CAR), a landlocked nation at Africa's heart, ranked 183rd globally in exports in 2022 per the Observatory of Economic Complexity (OEC). Its economy, rich in natural resources, focuses on agriculture, services, and exporting minerals, oil, timber, and agricultural products, with gold and diamonds among its top exports. Despite agriculture being pivotal, its contribution to exports is lesser. From 1960 to 2017, export volumes rose, but their GDP percentage has been declining since 1968. Major importers include the UAE, Italy, Pakistan, China, and France.

However, CAR faces challenges like political instability, violence, and inadequate resource management, hindering sustainable growth and global market competitiveness. Reliance on international aid is critical due to slow growth and a rising population, with poverty affecting 65.7% of the populace by 2021.

Exploratory Data Analysis

Data Description

This dataset, sourced from the World Bank, encompasses detailed yearly data on the Gross Domestic Product (GDP), imports, and exports as a percentage of total GDP, population figures, and the GDP growth of the Central African Republic from 1960 to 2017. The focal point of our report is the Exports data. In this dataset, exports are measured as a percentage of GDP, indicating the total value of the country's exports of goods and services in relation to the size of its Gross Domestic Product. Figure 1 provides a compelling overview of the country's economic trends over the past decades.



Missing Value Plots

Upon examining the dataset for missing values (Figure 2), it was found that the Exports data is free from any gaps, affirming the dataset's completeness and its suitability for conducting time series model analyses focused on Exports.

```r
gg_miss_var(finalPro_data)+
  labs(caption = "Figure 2: Missing Plot")+
  theme(plot.caption = element_text(size = 16,hjust=0.5))
```

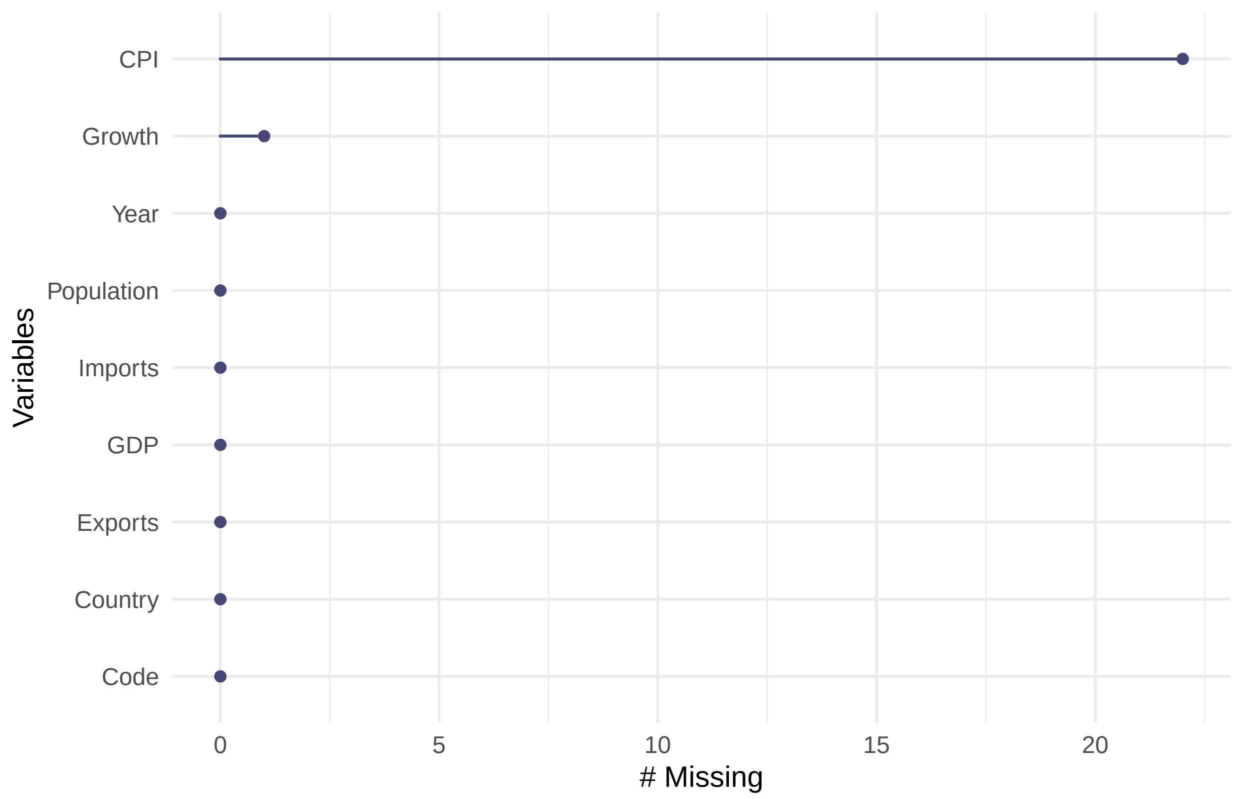
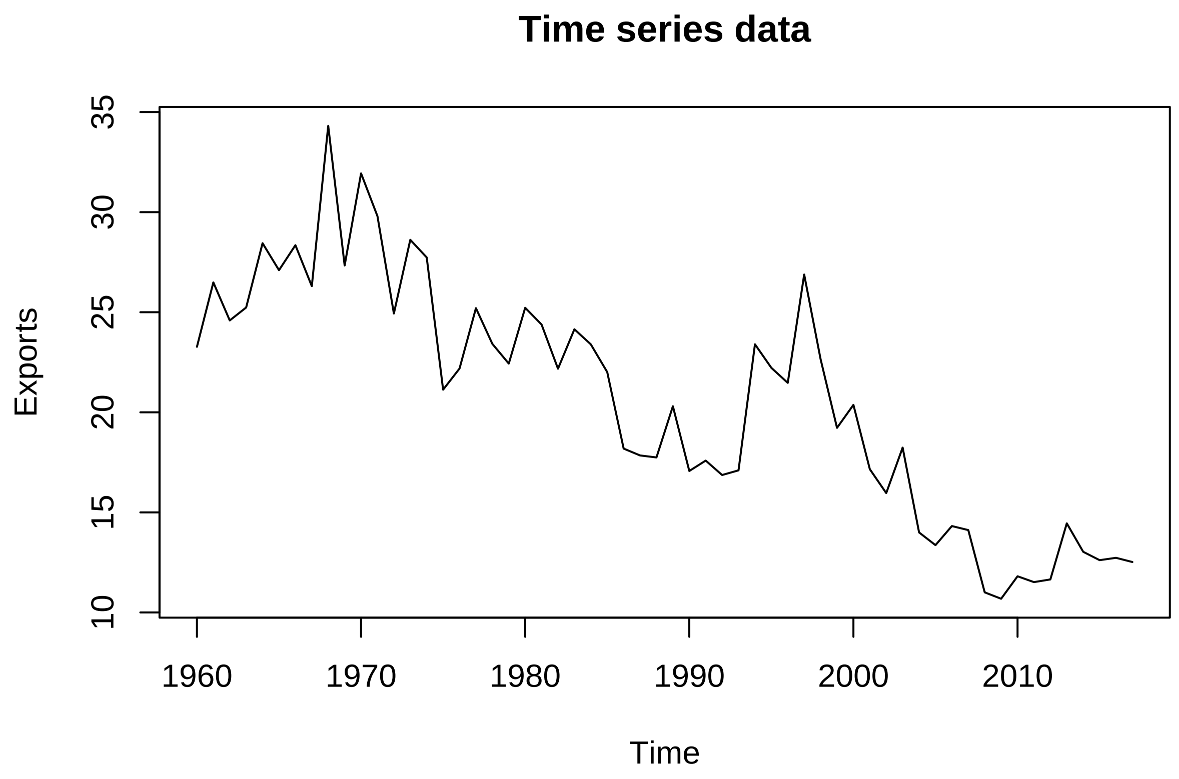


Figure 2: Missing Plot

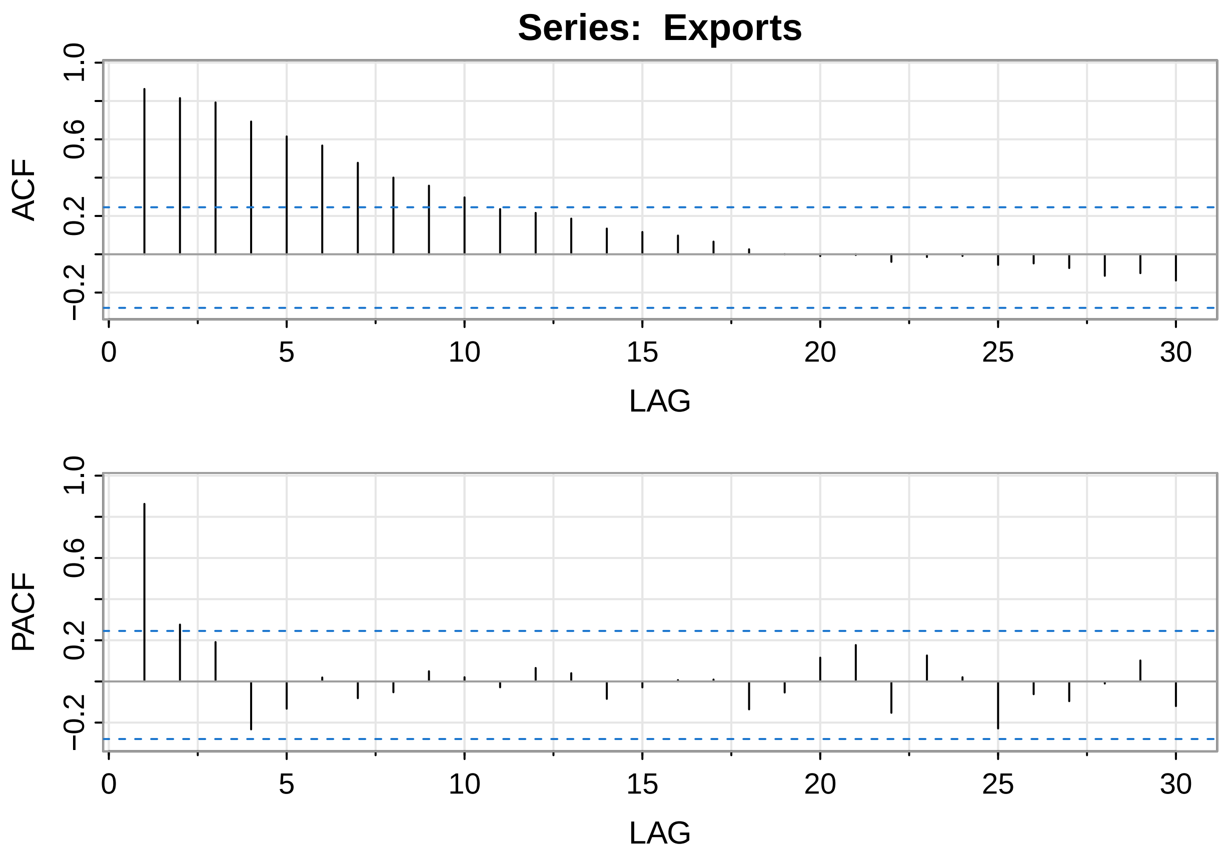
Visualization

The preliminary analysis commenced with an examination of the time series data for Exports. A subtle downward trend observed in exports, as illustrated in Figure 3, suggests the presence of a non-constant mean, indicating that the time series is non-stationary. Further investigation was conducted on the dataset's Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF). These analyses are crucial for understanding the direct influence of past data points on future values and for determining the appropriate order of the time series model (refer to Figure 4).

```r
# Plot the Export as time series data
ts.plot(Exports,main = "Time series data")
```



```
acf2(Exports,max.lag=30)
```



```
##       [,1] [,2] [,3]  [,4]  [,5] [,6]  [,7]  [,8] [,9] [,10] [,11] [,12] [,13]
## ACF  0.86 0.82 0.79  0.69  0.62 0.57  0.48  0.40 0.36  0.30  0.24  0.22  0.19
## PACF 0.86 0.28 0.19 -0.23 -0.13 0.02 -0.08 -0.05 0.05  0.02 -0.03  0.07  0.04
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25]
## ACF   0.13  0.12  0.10  0.07  0.03  0.00 -0.01  0.00 -0.04 -0.01 -0.01 -0.06
## PACF -0.08 -0.03  0.01  0.01 -0.14 -0.05  0.12  0.18 -0.15  0.13  0.02 -0.23
##      [,26] [,27] [,28] [,29] [,30]
## ACF  -0.05 -0.07 -0.11  -0.1 -0.14
## PACF -0.06 -0.10 -0.01   0.1 -0.12
```

Inferential Analysis

ADF and KPSS Tests for Stationary

To evaluate the stationarity of the dataset, we applied two distinct methods: the Augmented Dickey-Fuller (ADF) test and the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test, both set at a significance level of 0.05. The ADF test deems a time series as stationary if the p-values are less than 0.05, whereas the KPSS test considers a time series stationary if the p-values exceed 0.05. Results from both tests led to the rejection of the hypothesis that the time series is stationary, confirming that the dataset is indeed non-stationary. Consequently, further measures are required to achieve stationarity for effective analysis and prediction purposes.

```
sig_value <- 0.05 #significant value of test
# ADF: if p < sig, stationary
adf.test(Exports)$p.value
```

```
## [1] 0.1005534
```

```
#KPSS: if p > sig, stationary
kpss.test(Exports)$p.value
```

```
## Warning in kpss.test(Exports): p-value smaller than printed p-value
```

```
## [1] 0.01
```

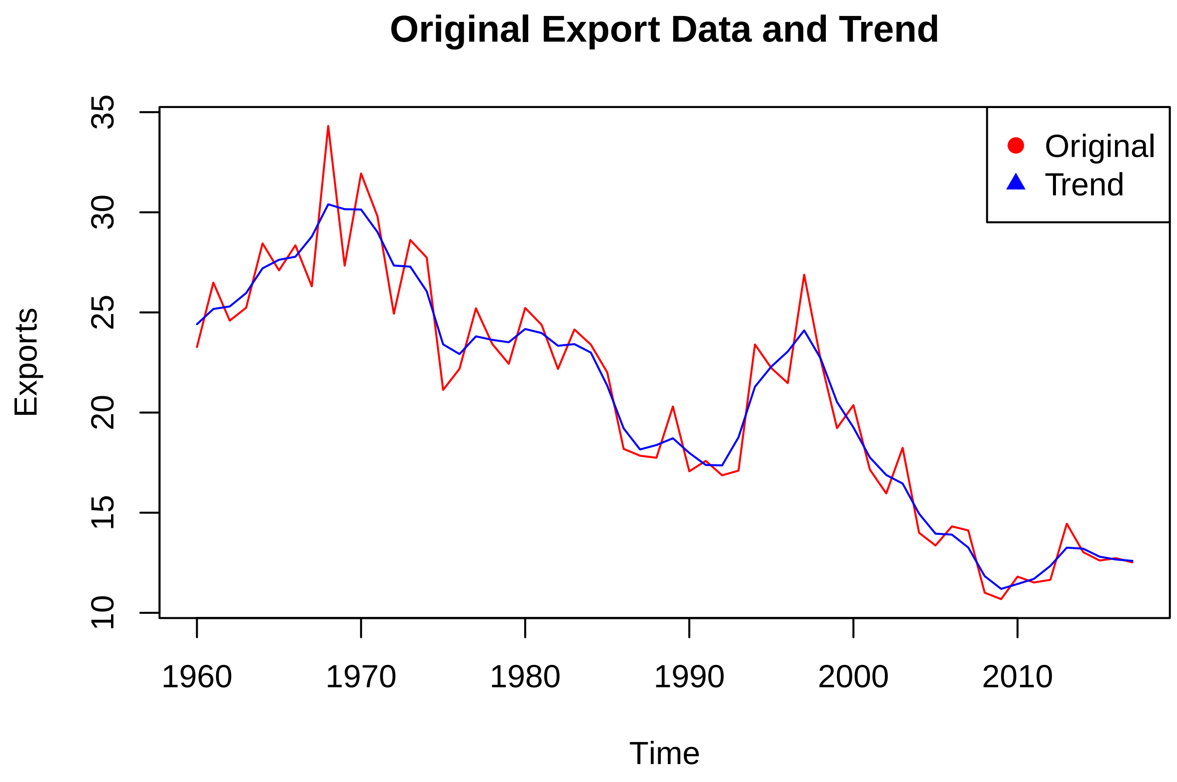
```
# Both tests indicate non-stationary.
```

Decomposition

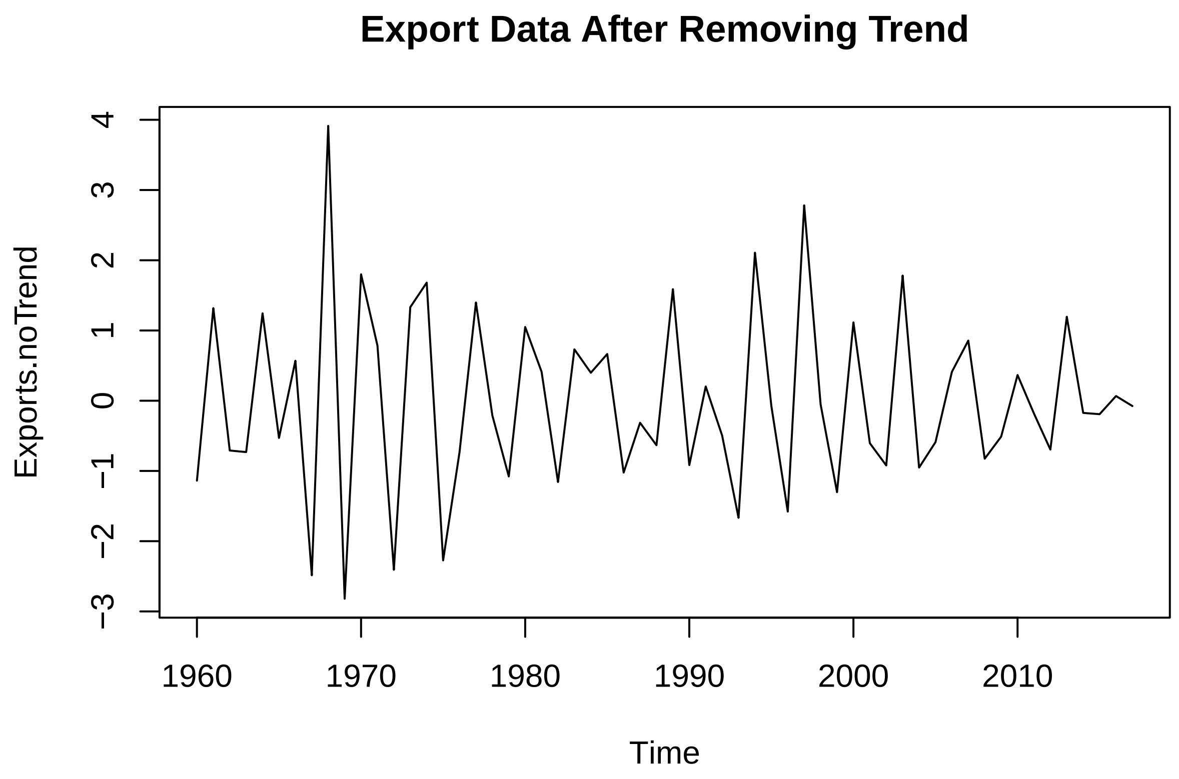
To address the non-stationarity of the dataset, four different methods were employed to detrend the data, aiming to isolate a suitable residual component. The initial approach involved Kernel smoothing, a technique utilized to eliminate the overarching decreasing trend observed over time. This process was intended to distill the data down to its cyclical trend component, as depicted in Figures 5 and 6. The goal was to refine the data in such a way that the remaining variability could be attributed primarily to cyclical fluctuations, thereby facilitating a more focused analysis of the time series.

```
#Try kernal smoothing for trend
kernel.type <- "gaussian"
bandwidth <- dpill(time(Exports), Exports)
smoothed_values <- ksmooth(x=time(Exports),y=Exports,kernel='normal',bandwidth = bandwidth,n.points = le

plot(Exports,col='red',main="Original Export Data and Trend")
lines(smoothed_values$x, smoothed_values$y,col='blue')
legend("topright", legend=c("Original", "Trend"), col=c("red", "blue"), pch=c(19, 17))
```

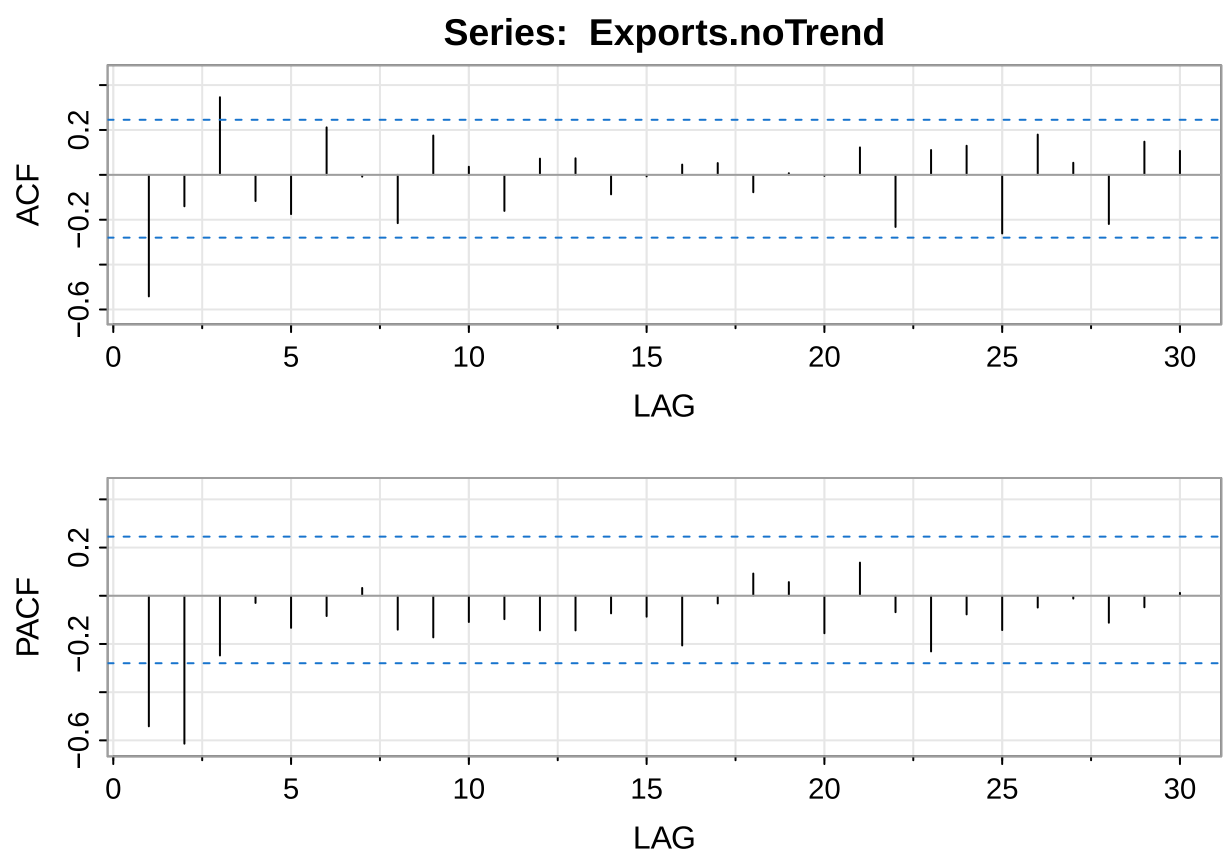


```
# Remove the Trend-cycle effect
Exports.noTrend <- Exports - smoothed_values$y
plot(Exports.noTrend,main='Export Data After Removing Trend')
```



To delve deeper into the model exploration, calculations of the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) were conducted. The ACF revealed a very strong correlation at a lag of 1, while the PACF displayed two pronounced correlations at lags 1 and 2. These findings suggest the potential suitability of AR(2) and MA(1) models for the de-trended data.

```
acf2(Exports.noTrend,max.lag= 30)
```



```
##       [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10] [,11] [,12]
## ACF  -0.54 -0.14  0.35 -0.12 -0.17  0.21 -0.01 -0.22  0.18  0.04 -0.16  0.07
## PACF -0.54 -0.61 -0.25 -0.03 -0.13 -0.08  0.03 -0.14 -0.17 -0.11 -0.10 -0.14
##      [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
## ACF   0.07 -0.09 -0.01  0.05  0.05 -0.08  0.01  0.00  0.12 -0.23  0.11  0.13
## PACF -0.14 -0.07 -0.09 -0.21 -0.03  0.09  0.06 -0.16  0.14 -0.07 -0.23 -0.08
##      [,25] [,26] [,27] [,28] [,29] [,30]
## ACF  -0.26  0.18  0.05 -0.22  0.15  0.11
## PACF -0.14 -0.05 -0.01 -0.11 -0.05  0.01
```

```
#adf and kpss tests
adf.test(Exports.noTrend)
```

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

##
##  Augmented Dickey-Fuller Test
##
## data:  Exports.noTrend
## Dickey-Fuller = -5.4555, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
```

```
kpss.test(Exports.noTrend)
```

```
## Warning in kpss.test(Exports.noTrend): p-value greater than printed p-value
```

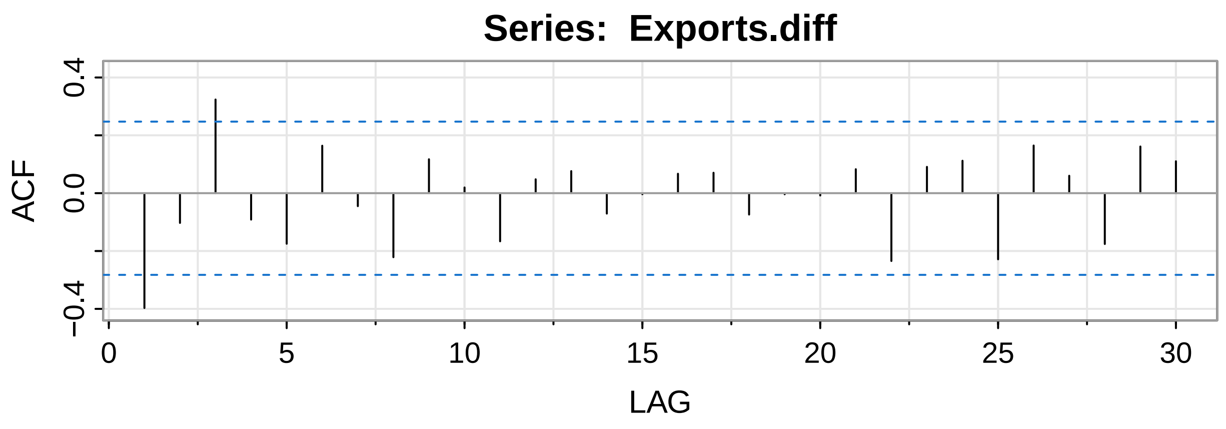
```
##
##  KPSS Test for Level Stationarity
##
## data:  Exports.noTrend
## KPSS Level = 0.032693, Truncation lag parameter = 3, p-value = 0.1
```

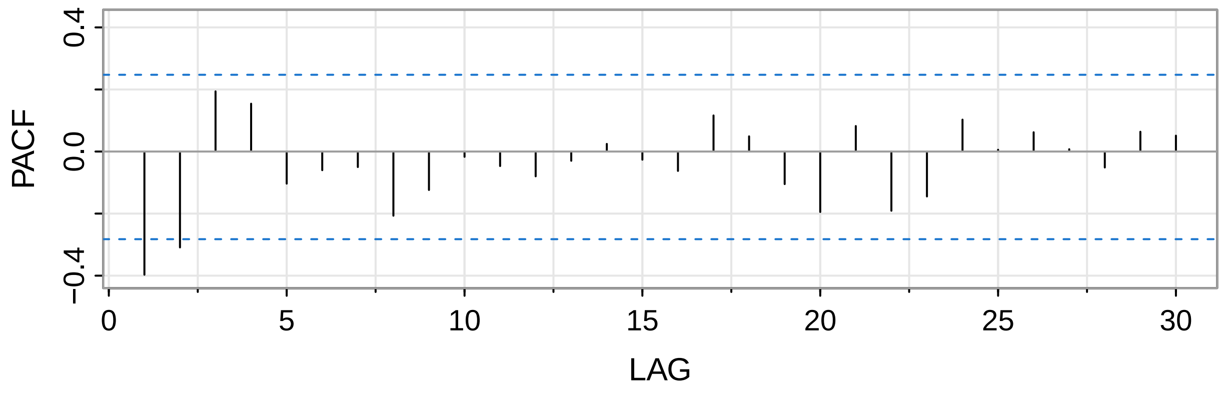
```
# Stationary now, indicating a AR(2) MA(1) possibly. Already stationary, no need for differential.
```

First-order Difference

Another transformation attempt was made using the first-order difference, which similarly indicated the appropriateness of an AR(2), MA(1) model for the dataset. Despite this, the first-order difference transformation failed to pass the ADF test, indicating that it did not achieve stationarity. Given this outcome, the decision was made to abandon the first-order difference approach.

```
# Take the 1st-order difference and test if the data is stationary
Exports.diff<-diff(Exports)
acf2(Exports.diff,max.lag= 30)
```





```
##      [,1]  [,2] [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10] [,11] [,12]
## ACF  -0.4 -0.10 0.32 -0.09 -0.17  0.16 -0.04 -0.22  0.12  0.02 -0.17  0.05
## PACF -0.4 -0.31 0.19  0.15 -0.10 -0.06 -0.05 -0.21 -0.12 -0.02 -0.05 -0.08
##      [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
## ACF   0.08 -0.07  0.00  0.07  0.07 -0.07   0.0 -0.01  0.08 -0.23  0.09  0.11
## PACF -0.03  0.02 -0.03 -0.06  0.12  0.05  -0.1 -0.19  0.08 -0.19 -0.14  0.10
##      [,25] [,26] [,27] [,28] [,29] [,30]
## ACF  -0.23  0.16  0.06 -0.18  0.16  0.11
## PACF  0.01  0.06  0.01 -0.05  0.06  0.05
```

```r
adf.test(Exports.diff)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  Exports.diff
## Dickey-Fuller = -3.2518, Lag order = 3, p-value = 0.08812
## alternative hypothesis: stationary
```

```r
kpss.test(Exports.diff)
```

```
## Warning in kpss.test(Exports.diff): p-value greater than printed p-value
```

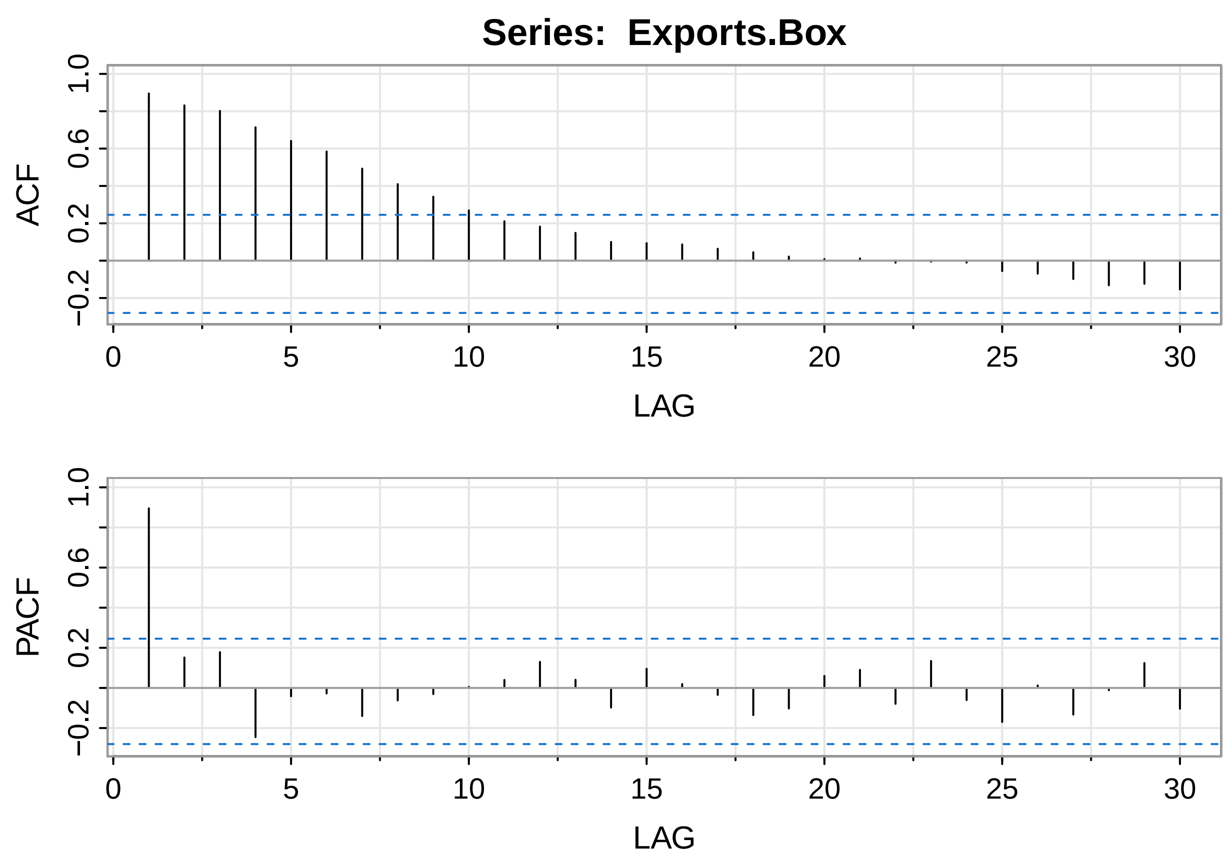
```
##
##  KPSS Test for Level Stationarity
##
## data:  Exports.diff
## KPSS Level = 0.092232, Truncation lag parameter = 3, p-value = 0.1
```

```r
#suggesting AR(2), MA(1), but does not pass ADF test.
```

Box-Cox Transformation

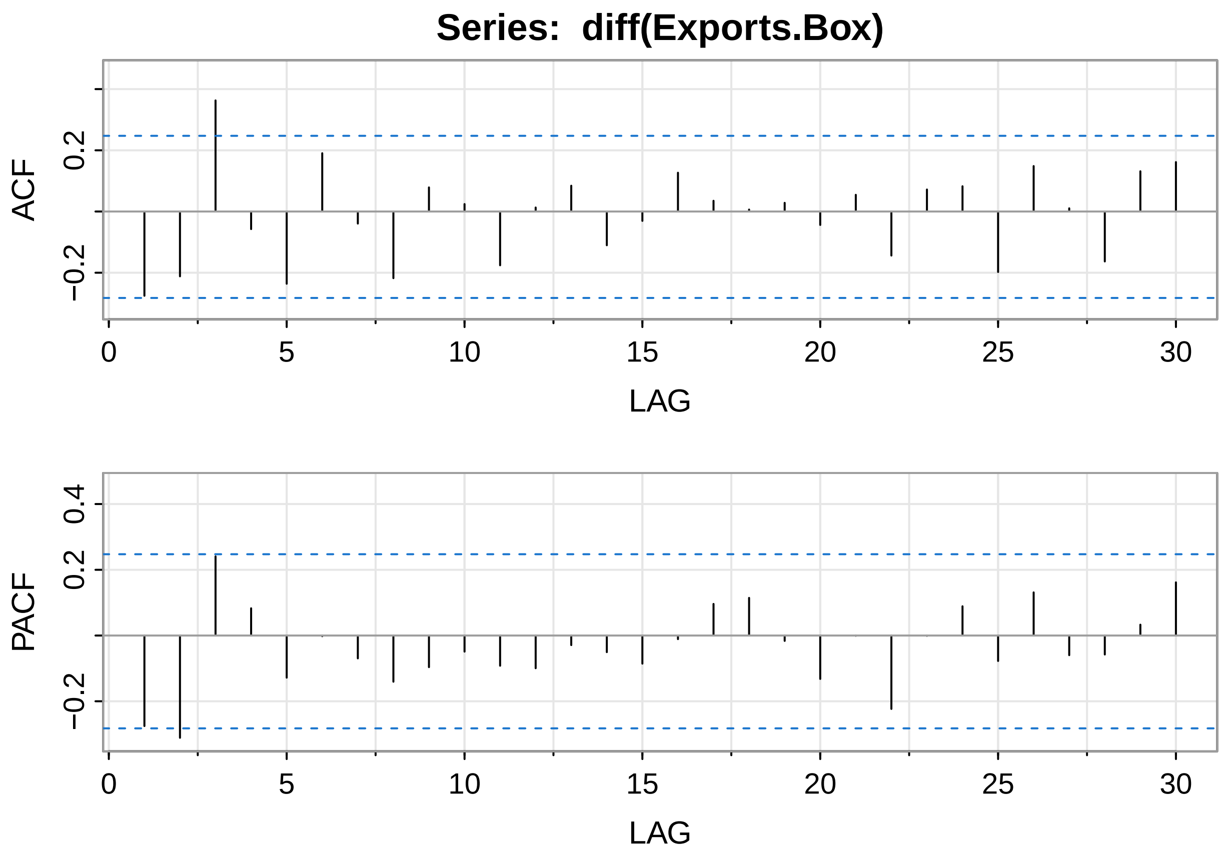
The application of both Box-Cox Transformation and Log Transformation was explored to address the non-stationary nature of the time series. However, subsequent analysis revealed that both transformations resulted in time series that remained non-stationary, as evidenced in Figure 8. Following the application of differencing to these transformed datasets, the ACF and PACF analyses of the differences suggested characteristics akin to a random series, resembling white noise. Consequently, due to these outcomes, both the Box-Cox and Log Transformations were deemed unsuitable for further consideration and were thus discarded from the analysis.

```r
# Box-Cox transformation
lambda<-BoxCox.lambda(Exports)
Exports.Box<-BoxCox(Exports,lambda)
acf2(Exports.Box,max.lag = 30)
```



```
##      [,1] [,2] [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10] [,11] [,12] [,13]
## ACF   0.9 0.83 0.80  0.71  0.64  0.58  0.49  0.41  0.34  0.27  0.21  0.18  0.15
## PACF  0.9 0.15 0.18 -0.25 -0.04 -0.03 -0.14 -0.06 -0.03  0.01  0.04  0.13  0.04
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25]
## ACF    0.1  0.09  0.09  0.06  0.05  0.02  0.01  0.01 -0.01 -0.01 -0.01 -0.06
## PACF  -0.1  0.10  0.02 -0.03 -0.14 -0.10  0.06  0.09 -0.08  0.13 -0.06 -0.17
##      [,26] [,27] [,28] [,29] [,30]
## ACF  -0.07 -0.10 -0.13 -0.12 -0.15
## PACF  0.01 -0.13 -0.01  0.12 -0.10
```

```
#ts.plot(Exports.Box,main = "Time series data after Box-Cox")
acf2(diff(Exports.Box),max.lag = 30)
```

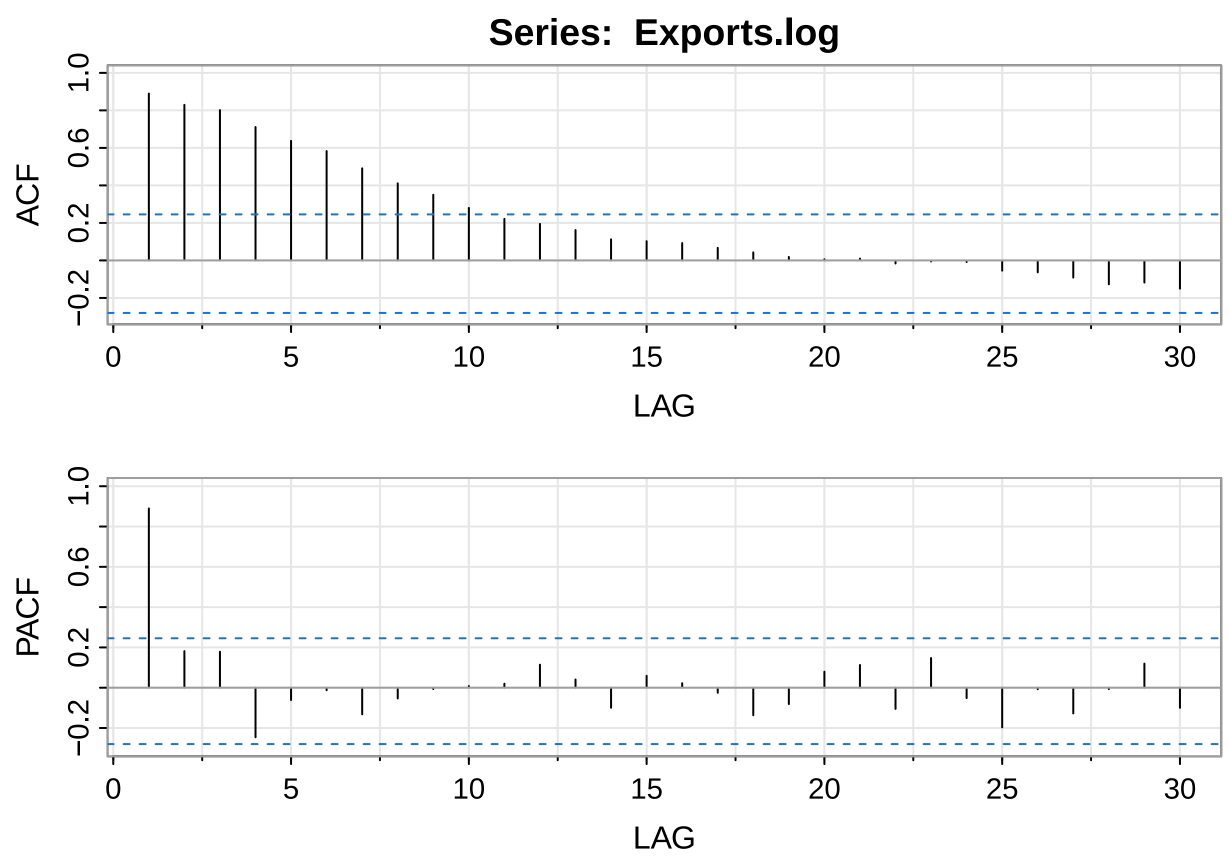


```
##       [,1]  [,2] [,3]  [,4]  [,5] [,6]  [,7]  [,8]  [,9] [,10] [,11] [,12]
## ACF  -0.28 -0.21 0.36 -0.06 -0.24 0.19 -0.04 -0.22  0.08  0.02 -0.18  0.01
## PACF -0.28 -0.31 0.24  0.08 -0.13 0.00 -0.07 -0.14 -0.10 -0.05 -0.09 -0.10
##      [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
## ACF   0.08 -0.11 -0.03  0.13  0.04  0.01  0.03 -0.04  0.05 -0.14  0.07  0.08
## PACF -0.03 -0.05 -0.09 -0.01  0.10  0.11 -0.02 -0.13  0.00 -0.22  0.00  0.09
##      [,25] [,26] [,27] [,28] [,29] [,30]
## ACF  -0.20  0.15  0.01 -0.16  0.13  0.16
## PACF -0.08  0.13 -0.06 -0.06  0.03  0.16
```

```
# suggesting white noise, so discard this transformation
```

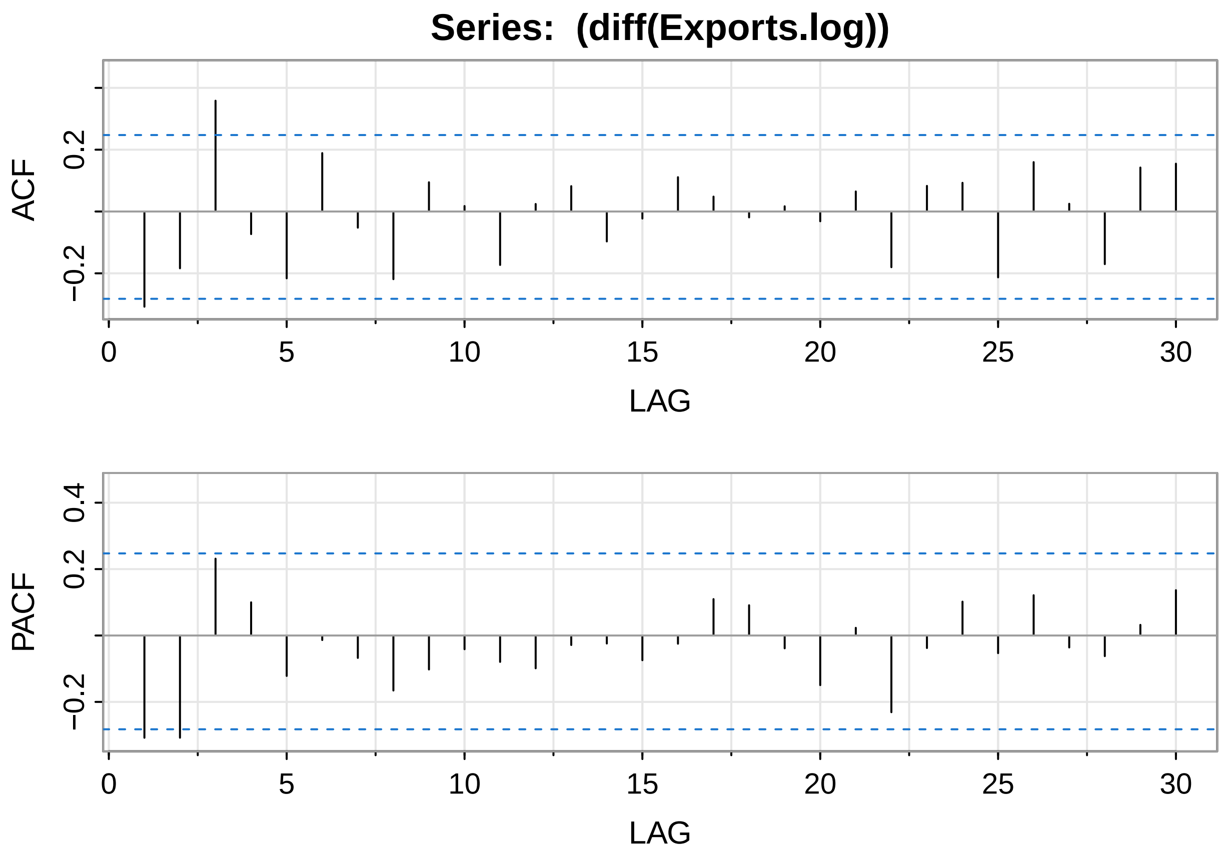
Log Transformation

```
# Take the 1st-order difference and test if the data is stationary
Exports.log<-log(Exports)
acf2(Exports.log,max.lag= 30)
```



```
##      [,1] [,2] [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10] [,11] [,12] [,13]
## ACF  0.89 0.83 0.80  0.71  0.64  0.58  0.49  0.41  0.35  0.28  0.22  0.19  0.16
## PACF 0.89 0.18 0.18 -0.25 -0.06 -0.01 -0.13 -0.05 -0.01  0.01  0.02  0.11  0.04
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25]
## ACF   0.11  0.10  0.09  0.07  0.04  0.02  0.01  0.01 -0.02 -0.01 -0.01 -0.05
## PACF -0.10  0.06  0.02 -0.03 -0.14 -0.08  0.08  0.11 -0.11  0.15 -0.05 -0.20
##      [,26] [,27] [,28] [,29] [,30]
## ACF  -0.06 -0.09 -0.13 -0.12 -0.15
## PACF -0.01 -0.13 -0.01  0.12 -0.10
```

```
acf2((diff(Exports.log)),max.lag= 30)
```



```
##       [,1]  [,2] [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10] [,11] [,12]
## ACF  -0.31 -0.18 0.36 -0.07 -0.22  0.19 -0.05 -0.22  0.09  0.02 -0.17  0.02
## PACF -0.31 -0.31 0.23  0.10 -0.12 -0.01 -0.07 -0.17 -0.10 -0.04 -0.08 -0.10
##      [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
## ACF   0.08 -0.10 -0.02  0.11  0.05 -0.02  0.02 -0.03  0.06 -0.18  0.08  0.09
## PACF -0.03 -0.02 -0.07 -0.02  0.11  0.09 -0.04 -0.15  0.02 -0.23 -0.04  0.10
##      [,25] [,26] [,27] [,28] [,29] [,30]
## ACF  -0.21  0.16  0.03 -0.17  0.14  0.15
## PACF -0.05  0.12 -0.04 -0.06  0.03  0.14
```

```
#suggesting white noise, so discard this transformation
```

After thorough comparison and careful consideration, the decision was made to select the detrending method as the preferred approach for preparing our data for model fitting in the subsequent phase of analysis.

ARIMA Model

(Post the formula here by using LaTeX)

For the ARIMA(q,d,p) model

$$X'_t = \sum_{i=1}^{p} \alpha_p X'_{t-p} + \sum_{i=1}^{q} \theta_q \omega_{t-q}$$

where $X'_t = (1 - B)^d X_t$ The model exhibits an order in the form of (p, d, q) where: P =The order of the Auto Regressive Model d = the order of differencing Q = The order of the Moving Average Following the detrending of the Exports dataset, the next step involved establishing the ARIMA model. To accurately determine the optimal parameters for the ARIMA(p,d,q) model, an automatic selection process was employed. This method suggested ARIMA(2,0,1) as the most fitting model for our data.

```r
# Auto select the best p,d,q combination for ARIMA(p,d,q)
Exports.noTrend.auto <- auto.arima(Exports.noTrend)
summary(Exports.noTrend.auto)
```

```
## Series: Exports.noTrend
## ARIMA(2,0,1) with zero mean
##
## Coefficients:
##           ar1      ar2      ma1
##       -0.5731  -0.4330  -0.5029
## s.e.   0.2122   0.1767   0.2619
##
## sigma^2 = 0.7146:  log likelihood = -71.79
## AIC=151.58   AICc=152.33   BIC=159.82
##
## Training set error measures:
##                      ME      RMSE       MAE      MPE     MAPE      MASE
## Training set 0.00429613 0.8231794 0.6252085 -15.1433 142.2494 0.3526863
##                     ACF1
## Training set 0.005845486
```

Model Comparison

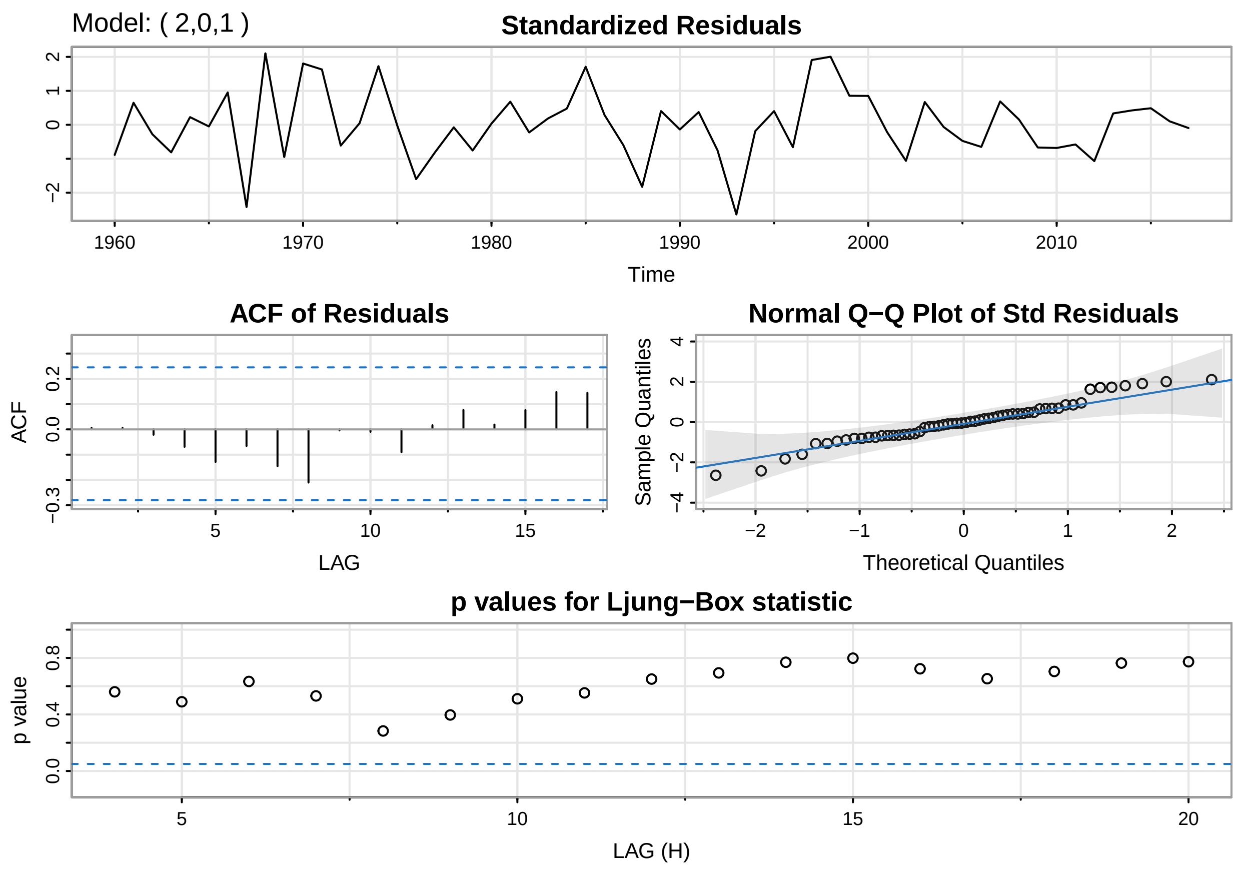
To discern the most effective model, Mean Squared Error (MSE) and Akaike Information Criterion (AIC) were employed as the primary metrics. MSE helps estimate the average of squared errors during model training, while AIC assesses both the prediction error and the extent of information loss during model fitting.

The SARIMA model was utilized for evaluation, showing that the Autocorrelation Function (ACF) of residuals mostly fell within the expected confidence range, with the Q-Q plot revealing a good alignment of most points along the fit line, albeit with some evidence of heavy tails. Comparisons were made between ARIMA(2,1,1) and ARIMA(3,0,1), against the backdrop of ARIMA(2,0,1). While ARIMA(2,1,1) exhibited a lower MSE of 0.0128 compared to ARIMA(3,0,1)'s MSE of 0.0735 and ARIMA(2,0,1)'s MSE of 0.00430, it was disqualified due to its p-values falling below the significance line for lags less than 5, indicating autocorrelation and dependency within white noise, rendering it suboptimal.

Despite ARIMA(3,0,1) presenting a slightly better AIC of 2.544 against ARIMA(2,0,1)'s AIC of 2.613, the minimal difference and the preference for simplicity guided the choice towards ARIMA(2,0,1) for prediction purposes, with the specific equation provided in the following segment.

```r
# Use SARIMA to determent the performance, acf of residual all need to be in range, p-value needs to be
# ARIMA(2,0,1)
fit1<-sarima(Exports.noTrend, 2,0,1, no.constant=TRUE)
```

```
## initial  value 0.262957
## iter   2 value -0.129810
## iter   3 value -0.161969
## iter   4 value -0.180723
## iter   5 value -0.181623
## iter   6 value -0.182230
## iter   7 value -0.183361
## iter   8 value -0.183842
## iter   9 value -0.183860
## iter  10 value -0.183875
## iter  11 value -0.183879
## iter  12 value -0.183886
## iter  13 value -0.183886
## iter  14 value -0.183887
## iter  14 value -0.183887
## iter  14 value -0.183887
## final  value -0.183887
## converged
## initial  value -0.181011
## iter   2 value -0.181049
## iter   3 value -0.181133
## iter   4 value -0.181178
## iter   5 value -0.181212
## iter   6 value -0.181214
## iter   7 value -0.181214
## iter   7 value -0.181214
## iter   7 value -0.181214
## final  value -0.181214
## converged
## <><><><><><><><><><><><><><>
##
## Coefficients:
##     Estimate     SE t.value p.value
## ar1  -0.5731 0.2122 -2.7011  0.0092
## ar2  -0.4330 0.1767 -2.4497  0.0175
## ma1  -0.5029 0.2619 -1.9199  0.0601
##
## sigma^2 estimated as 0.6776243 on 55 degrees of freedom
##
## AIC = 2.61338  AICc = 2.621042  BIC = 2.755479
##
```



```r
summary(fit1)
```

```
##                      Length Class  Mode
## fit                  14     Arima  list
## degrees_of_freedom    1     -none- numeric
## ttable               12     -none- numeric
## ICs                   3     -none- numeric
```

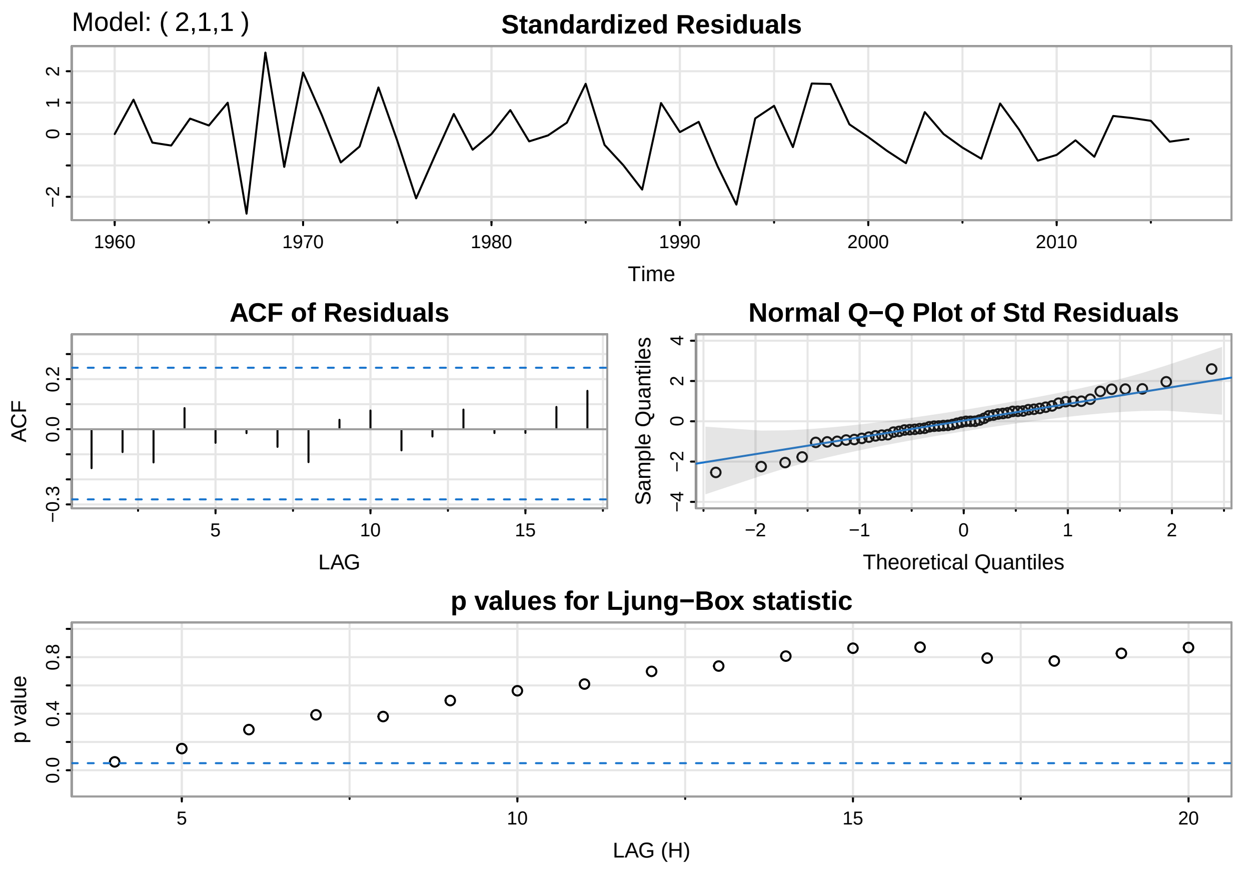
```r
# MSE is a standard to evaluate the model. The smaller is the better.
MSE1<-sum(fit1$fit$residuals)/58
print(paste(c("Training MSE:",MSE1)))
```

```
## [1] "Training MSE:"       "0.00429612956658058"
```

```r
# ARIMA(2,1,1)
fit2<-sarima(Exports.noTrend, 2,1,1, no.constant=TRUE)
```

```
## initial  value 0.831410
## iter   2 value 0.261456
## iter   3 value -0.021518
## iter   4 value -0.046030
## iter   5 value -0.090839
## iter   6 value -0.093317
## iter   7 value -0.102508
## iter   8 value -0.113413
## iter   9 value -0.122468
## iter  10 value -0.123218
## iter  11 value -0.133289
## iter  12 value -0.133723
## iter  13 value -0.135805
## iter  14 value -0.141059
## iter  15 value -0.144837
## iter  16 value -0.152137
## iter  17 value -0.162803
## iter  18 value -0.165017
## iter  19 value -0.165608
## iter  20 value -0.176908
## iter  21 value -0.177134
## iter  21 value -0.177134
## iter  22 value -0.177252
## iter  23 value -0.177407
## iter  23 value -0.177407
## iter  24 value -0.177410
## iter  24 value -0.177410
## iter  25 value -0.177411
## iter  25 value -0.177411
## iter  26 value -0.177412
## iter  26 value -0.177412
## iter  26 value -0.177412
## final  value -0.177412
## converged
## initial  value -0.011219
## iter   2 value -0.054025
## iter   3 value -0.070015
## iter   4 value -0.073149
## iter   5 value -0.083671
## iter   6 value -0.083946
## iter   7 value -0.083955
## iter   8 value -0.083960
## iter   9 value -0.083960
## iter  10 value -0.083960
## iter  11 value -0.083960
## iter  12 value -0.083960
## iter  12 value -0.083960
## iter  12 value -0.083960
## final  value -0.083960
## converged
## <><><><><><><><><><><><><><><><>
##
## Coefficients:
##     Estimate     SE  t.value p.value
## ar1  -0.8604 0.1036  -8.3076       0
## ar2  -0.5963 0.1021  -5.8392       0
## ma1  -1.0000 0.0470 -21.2753       0
##
## sigma^2 estimated as 0.7470543 on 54 degrees of freedom
##
## AIC = 2.810309  AICc = 2.818253  BIC = 2.953681
```

```
##
```



```r
summary(fit2)
```

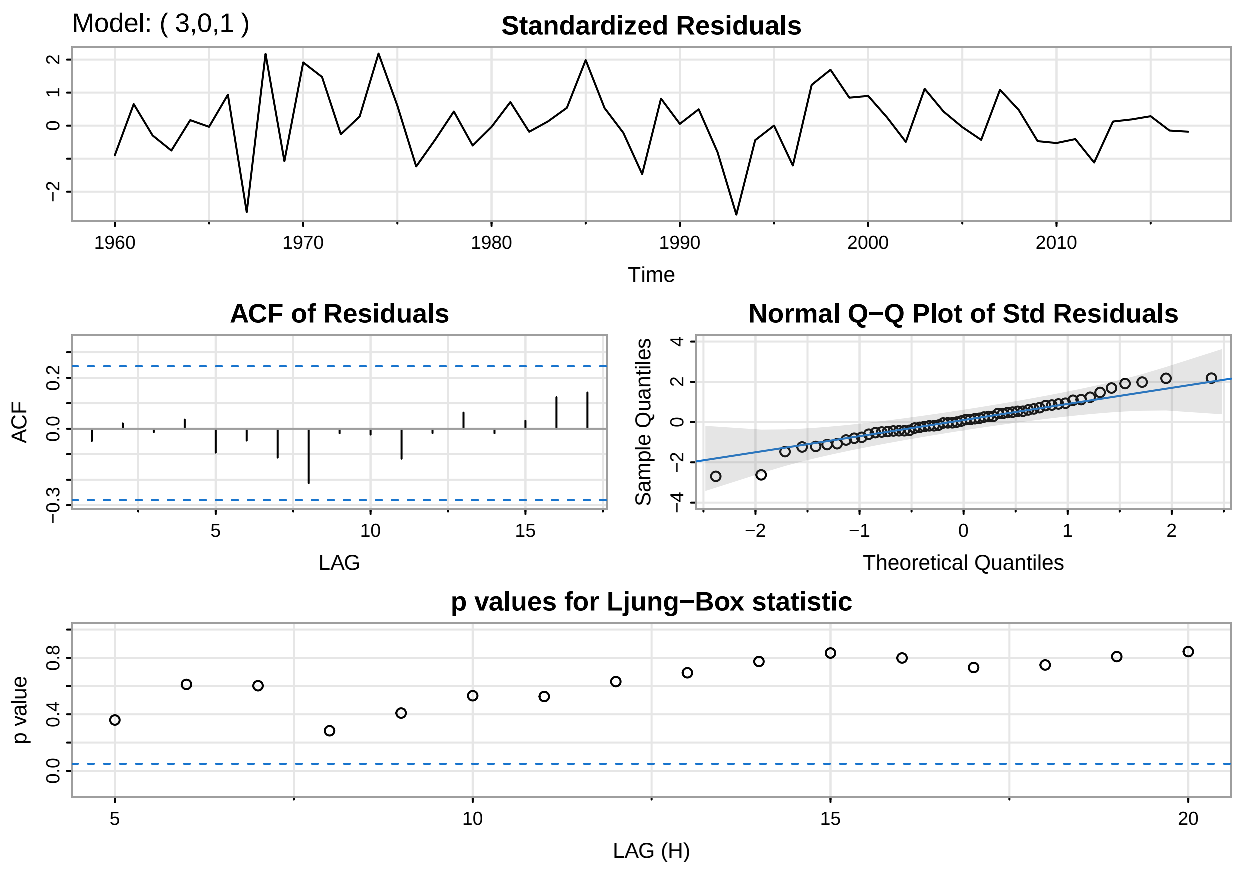
```
##                     Length Class  Mode   
## fit                 14     Arima  list   
## degrees_of_freedom   1     -none- numeric
## ttable              12     -none- numeric
## ICs                  3     -none- numeric
```

```r
# MSE
MSE2<-sum(fit2$fit$residuals)/58
print(paste(c("Training MSE:",MSE2)))
```

```
## [1] "Training MSE:"      "0.0128333615678964"
```

```r
#ARIMA(3,0,1)
fit3<-sarima(Exports.noTrend, 3,0,1, no.constant=TRUE)
```

```
## initial  value 0.269306 
## iter   2 value -0.117032
## iter   3 value -0.188298
## iter   4 value -0.195805
## iter   5 value -0.204950
## iter   6 value -0.212512
## iter   7 value -0.215821
## iter   8 value -0.219087
## iter   9 value -0.219135
## iter  10 value -0.219187
## iter  11 value -0.219189
## iter  12 value -0.219189
## iter  12 value -0.219189
## iter  12 value -0.219189
## final  value -0.219189
## converged
## initial  value -0.224429
## iter   2 value -0.231448
## iter   3 value -0.231736
## iter   4 value -0.233362
## iter   5 value -0.233379
## iter   6 value -0.233390
## iter   7 value -0.233390
## iter   8 value -0.233390
## iter   8 value -0.233390
## iter   8 value -0.233390
## final  value -0.233390
## converged
## <><><><><><><><><><><><><><>
##
## Coefficients:
##     Estimate     SE  t.value p.value
## ar1  -0.1936 0.1272  -1.5225  0.1337
## ar2  -0.0848 0.1297  -0.6541  0.5158
## ar3   0.2884 0.1259   2.2899  0.0260
## ma1  -1.0000 0.0547 -18.2701  0.0000
##
## sigma^2 estimated as 0.5809476 on 54 degrees of freedom
##
## AIC = 2.543511  AICc = 2.556523  BIC = 2.721135
##
```



```r
summary(fit3)
```

```
##                     Length Class  Mode
## fit                 14     Arima  list
## degrees_of_freedom   1     -none- numeric
## ttable              16     -none- numeric
## ICs                  3     -none- numeric
```

```r
# MSE
MSE3<-sum(fit3$fit$residuals)/58
print(paste(c("Training MSE:",MSE3)))
```

```
## [1] "Training MSE:"      "0.0735348695589228"
```

Forecast

After comparing models and conducting diagnostics, ARIMA(2,0,1) with detrended data was identified as the optimal choice. The model formula is

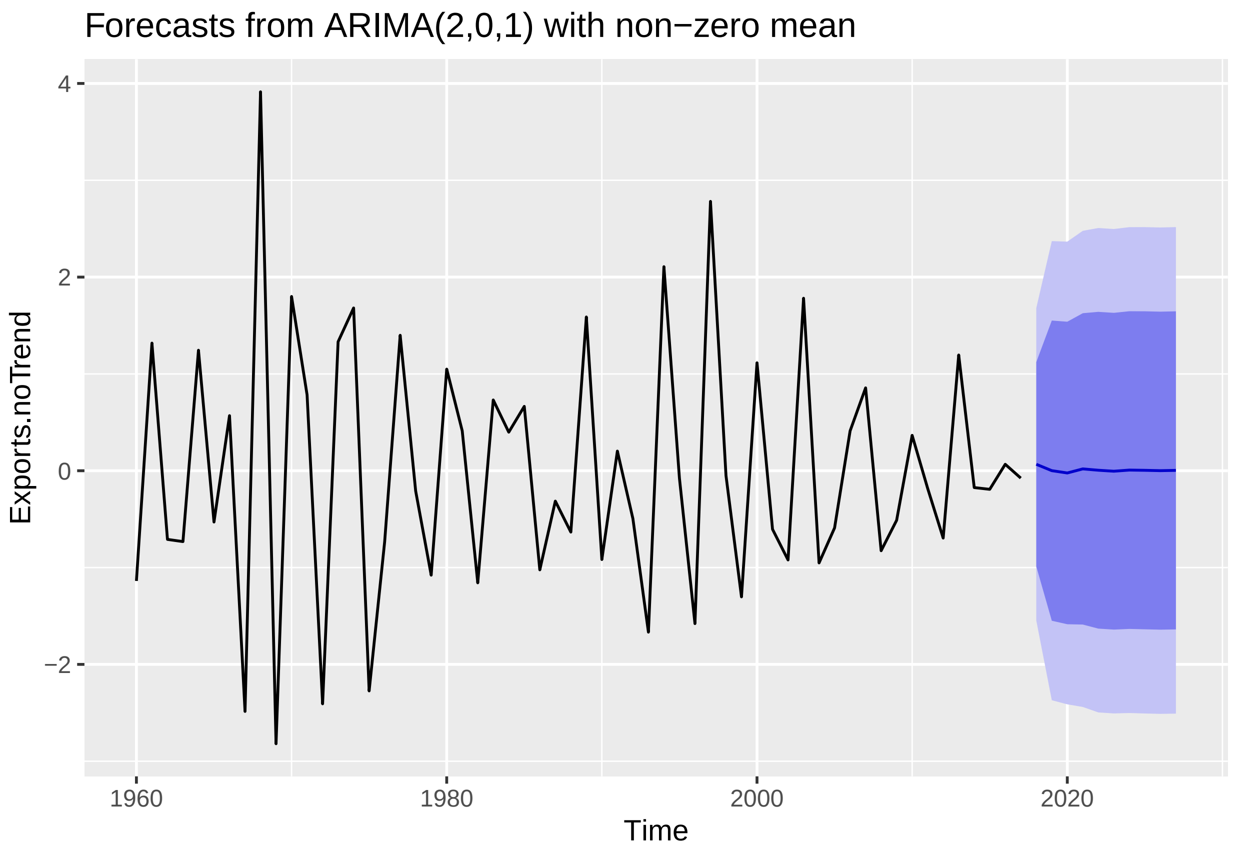
$$(1 - \phi_1 B - \phi_2 B^2)x_t = (1 + B)w_t$$

Utilizing this model, we forecasted the white noise in exports for the next six years with a significance level $\alpha = 0.05$.

```r
forecast(arima(Exports.noTrend,c(2,0,1)))
```

```
##      Point Forecast      Lo 80    Hi 80     Lo 95    Hi 95
## 2018   0.0668894555 -0.9879141 1.121693 -1.546293 1.680072
## 2019   0.0010087113 -1.5490973 1.551115 -2.369674 2.371691
## 2020  -0.0230010742 -1.5851947 1.539193 -2.412170 2.366168
## 2021   0.0191908473 -1.5885004 1.626882 -2.439561 2.477942
## 2022   0.0054325948 -1.6301906 1.641056 -2.496037 2.506902
## 2023  -0.0049255498 -1.6405687 1.630718 -2.506426 2.496575
## 2024   0.0069419949 -1.6335014 1.647385 -2.501900 2.515784
## 2025   0.0046293931 -1.6375372 1.646796 -2.506848 2.516107
## 2026   0.0008252783 -1.6413792 1.643030 -2.510710 2.512360
## 2027   0.0039999627 -1.6386646 1.646665 -2.508239 2.516239
```

```r
autoplot(forecast(arima(Exports.noTrend,c(2,0,1))))
```



The forecast depicted focuses solely on the random white noise component of exports over the next six years, not the actual future exports. To accurately forecast total exports, a linear fit and prediction of the trend using a separate linear model would be necessary. However, this paper concentrates primarily on analyzing the random noise within the time series, not on predicting the trend.

Conclusion

In this project, detrending the data was chosen as the necessary step to adhere to the stationary assumption and reduce errors, leading to the establishment of an ARIMA(2,0,1) model. However, the forecast process encounters a limitation due to its inability to predict future trends, offering only a 95% confidence interval for predictions that exclude any trend component. This constitutes a significant constraint of the study.

Looking ahead, to address this limitation and enhance the project, an initial step could involve identifying suitable functions or distributions to estimate the trend, which would allow for more comprehensive confidence intervals that incorporate the trend. Additionally, the introduction of more data and the exploration of alternative modeling approaches, such as the Long Short-Term Memory (LSTM) model, could offer improved outcomes that include trend analysis, thus broadening the scope and applicability of the findings.

Reference

(If any)

Code Appendix

```
# include=false will not performance our codes and results in the html file; echo=false will only show
# import libraries and data here

library(ggplot2)
library(dplyr)
library(naniar)
library(astsa)
library(forecast)
library(TTR)
library(KernSmooth)
library(tseries)
load("finalproject.Rdata")
Exports<-ts(finalPro_data$Exports,start = 1960)
gg_miss_var(finalPro_data)+
  labs(caption = "Figure 2: Missing Plot")+
  theme(plot.caption = element_text(size = 16,hjust=0.5))
# Plot the Export as time series data
ts.plot(Exports,main = "Time series data")
acf2(Exports,max.lag=30)
sig_value <- 0.05 #significant value of test
# ADF: if p < sig, stationary
adf.test(Exports)$p.value
#KPSS: if p > sig, stationary
kpss.test(Exports)$p.value

# Both tests indicate non-stationary.
#Try kernal smoothing for trend
kernel.type <- "gaussian"
bandwidth <- dpill(time(Exports), Exports)
smoothed_values <- ksmooth(x=time(Exports),y=Exports,kernel='normal',bandwidth = bandwidth,n.points = l

plot(Exports,col='red',main="Original Export Data and Trend")
lines(smoothed_values$x, smoothed_values$y,col='blue')
legend("topright", legend=c("Original", "Trend"), col=c("red", "blue"), pch=c(19, 17))

# Remove the Trend-cycle effect
Exports.noTrend <- Exports - smoothed_values$y
plot(Exports.noTrend,main='Export Data After Removing Trend')
acf2(Exports.noTrend,max.lag= 30)

#adf and kpss tests
adf.test(Exports.noTrend)
kpss.test(Exports.noTrend)

# Stationary now, indicating a AR(2) MA(1) possibly. Already stationary, no need for differential.
# Take the 1st-order difference and test if the data is stationary
Exports.diff<-diff(Exports)
acf2(Exports.diff,max.lag= 30)

adf.test(Exports.diff)
kpss.test(Exports.diff)

#suggesting AR(2), MA(1), but does not pass ADF test.
# Box-Cox transformation
lambda<-BoxCox.lambda(Exports)
Exports.Box<-BoxCox(Exports,lambda)
acf2(Exports.Box,max.lag = 30)
#ts.plot(Exports.Box,main = "Time series data after Box-Cox")
acf2(diff(Exports.Box),max.lag = 30)

# suggesting white noise, so discard this transformation

# Take the 1st-order difference and test if the data is stationary
Exports.log<-log(Exports)
acf2(Exports.log,max.lag= 30)
acf2((diff(Exports.log)),max.lag= 30)

#suggesting white noise, so discard this transformation
# Auto select the best p,d,q combination for ARIMA(p,d,q)
Exports.noTrend.auto <- auto.arima(Exports.noTrend)
summary(Exports.noTrend.auto)
# Use SARIMA to determent the performance, acf of residual all need to be in range, p-value needs to be
# ARIMA(2,0,1)
fit1<-sarima(Exports.noTrend, 2,0,1, no.constant=TRUE)
summary(fit1)

# MSE is a standard to evaluate the model. The smaller is the better.
MSE1<-sum(fit1$fit$residuals)/58
print(paste(c("Training MSE:",MSE1)))
# ARIMA(2,1,1)
fit2<-sarima(Exports.noTrend, 2,1,1, no.constant=TRUE)
summary(fit2)

# MSE
MSE2<-sum(fit2$fit$residuals)/58
print(paste(c("Training MSE:",MSE2)))
#ARIMA(3,0,1)
fit3<-sarima(Exports.noTrend, 3,0,1, no.constant=TRUE)
summary(fit3)

# MSE
MSE3<-sum(fit3$fit$residuals)/58
print(paste(c("Training MSE:",MSE3)))

forecast(arima(Exports.noTrend,c(2,0,1)))
autoplot(forecast(arima(Exports.noTrend,c(2,0,1))))
```