ASX Data Analysis Report

Wong Yi Wei S3966890

2023-08-25

1.0 Introduction

The aim of this report is to give a precise analysis of the series data through the investigation the existence of nonstationary, impact of the components of a time series data, and to seek the most accurate and suitable distributed lag model using the dataset of monthly averages of ASX All Ordinaries (Ords) Price index, Gold Price (AUD), Crude Oil (Brent, USD/bbl), and Copper (USD/tonne).

To perform these investigations, we will first prep the data for analysis and do further testing through various analysis methods.

2.0 Data Description and Preprocessing

The dataset which we will be using for this analysis is the csv file `ASX_data.csv`, which includes the monthly averages of ASX All Ordinaries (Ords) Price index, Gold Price (AUD), Crude Oil (Brent, USD/bbl), and Copper (USD/tonne) starting from January 2004.

```
asx<-read_csv("ASX_data.csv")
```

```
## Rows: 161 Columns: 4
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## dbl (2): ASX price, Crude Oil (Brent)_USD/bbl
## num (2): Gold price, Copper_USD/tonne
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
class(asx)
```

```
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

```
str(asx)
```

```
## spc_tbl_ [161 × 4] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ASX price               : num [1:161] 2935 2778 2849 2971 2980 ...
##  $ Gold price              : num [1:161] 612 603 566 539 549 ...
##  $ Crude Oil (Brent)_USD/bbl: num [1:161] 31.3 32.6 30.3 25 25.8 ...
##  $ Copper_USD/tonne        : num [1:161] 1650 1682 1656 1588 1651 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   `ASX price` = col_double(),
##   ..   `Gold price` = col_number(),
##   ..   `Crude Oil (Brent)_USD/bbl` = col_double(),
##   ..   `Copper_USD/tonne` = col_number()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
head(asx)
```

```
## # A tibble: 6 × 4
##   `ASX price` `Gold price` `Crude Oil (Brent)_USD/bbl` `Copper_USD/tonne`
##         <dbl>        <dbl>                       <dbl>              <dbl>
## 1       2935.         612.                        31.3               1650
## 2       2778.         603.                        32.6               1682
## 3       2849.         566.                        30.3               1656
## 4       2971.         539.                        25.0               1588
## 5       2980.         549.                        25.8               1651
## 6       3000.         536.                        27.6               1685
```

From the code chunk above, we first import the dataset into R using the `read_csv` function and assign it with the name asx. Then, we check the data class by using the function `class`, and we can see that it is imported as a data frame, and `str` function checks the structure of the data and the variable types. We can see here that the variables are all in numerical variable type which is correct. Then, we use the `head` function to view the first 6 observations of the data.

```
ords.ts<-ts(asx$`ASX price`,start=2004,frequency = 12)
gold.ts<-ts(asx$`Gold price`,start=2004,frequency = 12)
crude.ts<-ts(asx$`Crude Oil (Brent)_USD/bbl`,start=c(2004),frequency = 12)
copper.ts<-ts(asx$`Copper_USD/tonne`,start=2004,frequency = 12)
data.ts<-ts(asx,start=2004,frequency = 12)
```

To process the data for further analysis, we must first convert the objects in the data frame `asx` to time series objects. By using the `ts` function, we convert each variables in the dataset into time series objects and name the new objects `ords.ts` , `gold.ts` , `crude.ts` , `copper.ts` , and convert the dataframe as a time series object with all variables as `data.ts`. Since we know that the dataset starts from January 2004 and it contains monthly values, we can write the arguments `start` to 2004, and `frequency` to 12 for all conversions.

3.0 Data Visualisation

Here we will plot each time series objects as a graph to gain a further understanding of the data through visual inspection. We will use the `plot` function to plot the graphs.

```
plot(ords.ts,ylab="ASX Ordinaries Price",xlab="Year",main="Time Series Plot of Monthly ASX
        All Ordinaries Price Index",type="o")
```

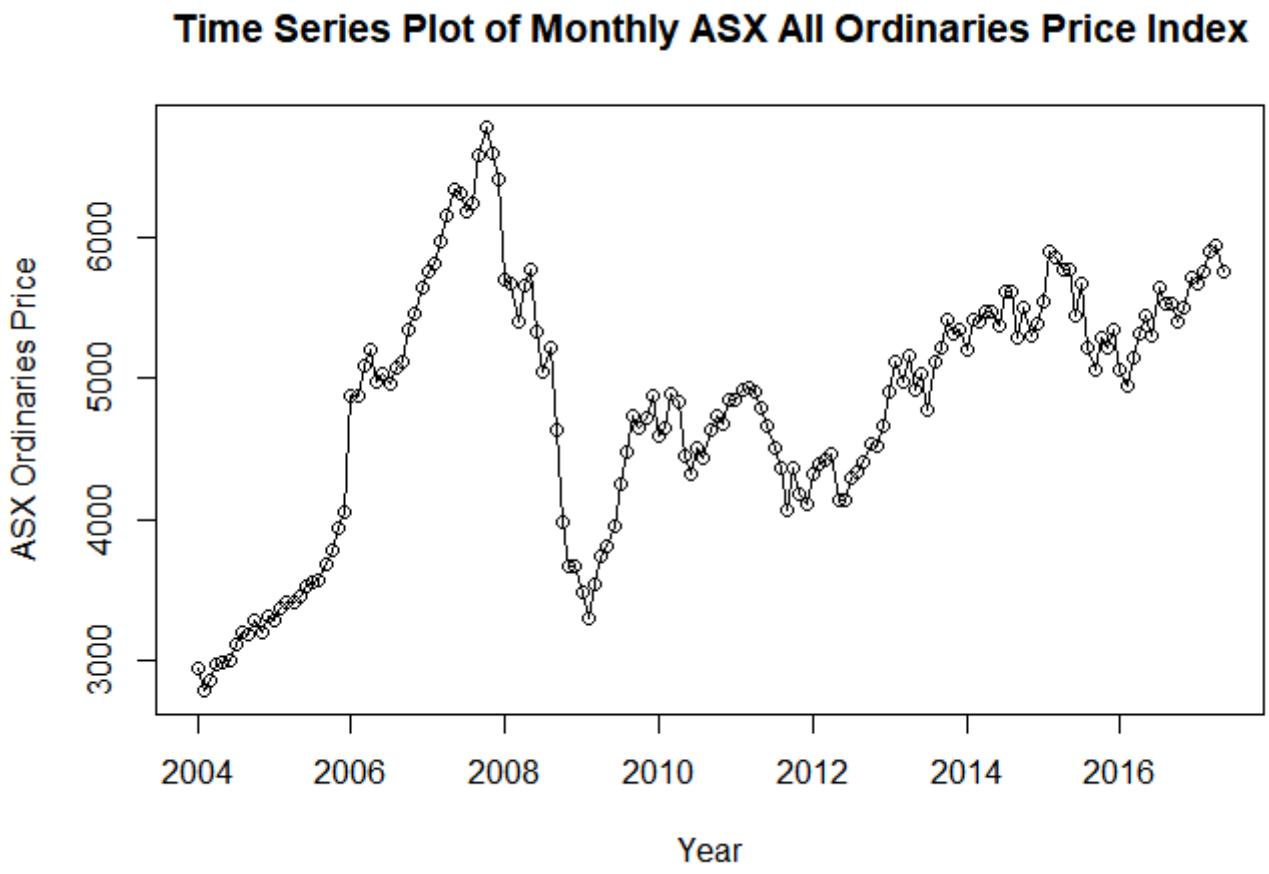

Figure 1

In `Figure 1`, the time series plot of monthly ASX all ordinaries price index contains a definite upward trend, that changes somewhat over time. There is no sign of seasonality, changing variance is not obvious, and there is sign of intervention around 2008. Succeeding points and fluctuations implies the existence of an autoregressive behaviour.

```
plot(gold.ts,ylab="Gold Price (AUD)",xlab="Year",main="Time Series Plot of Monthly Gold
        Price",type="o")
```

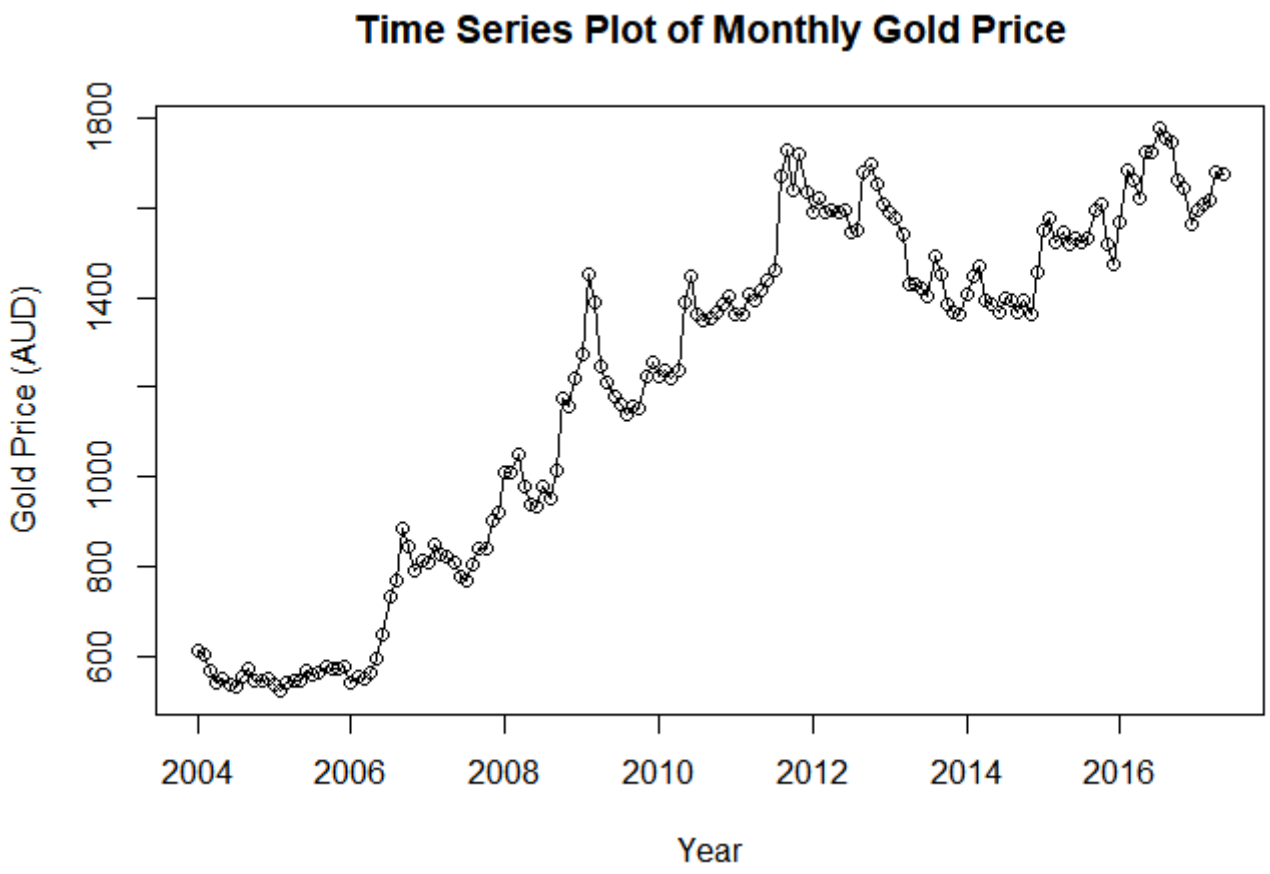

Figure 2

In `Figure 2`, the time series plot of monthly gold price contains a definite upwards trend over time. There is no sign of a seasonality, changing variance or an intervention. Succeeding points and fluctuations implies the existence of an autoregressive behaviour.

```
plot(crude.ts,ylab="Crude Oil Price (Brent, USD/bbl)",xlab="Year",main="Time Series Plot of
        Monthly Crude Oil Price",type="o")
```

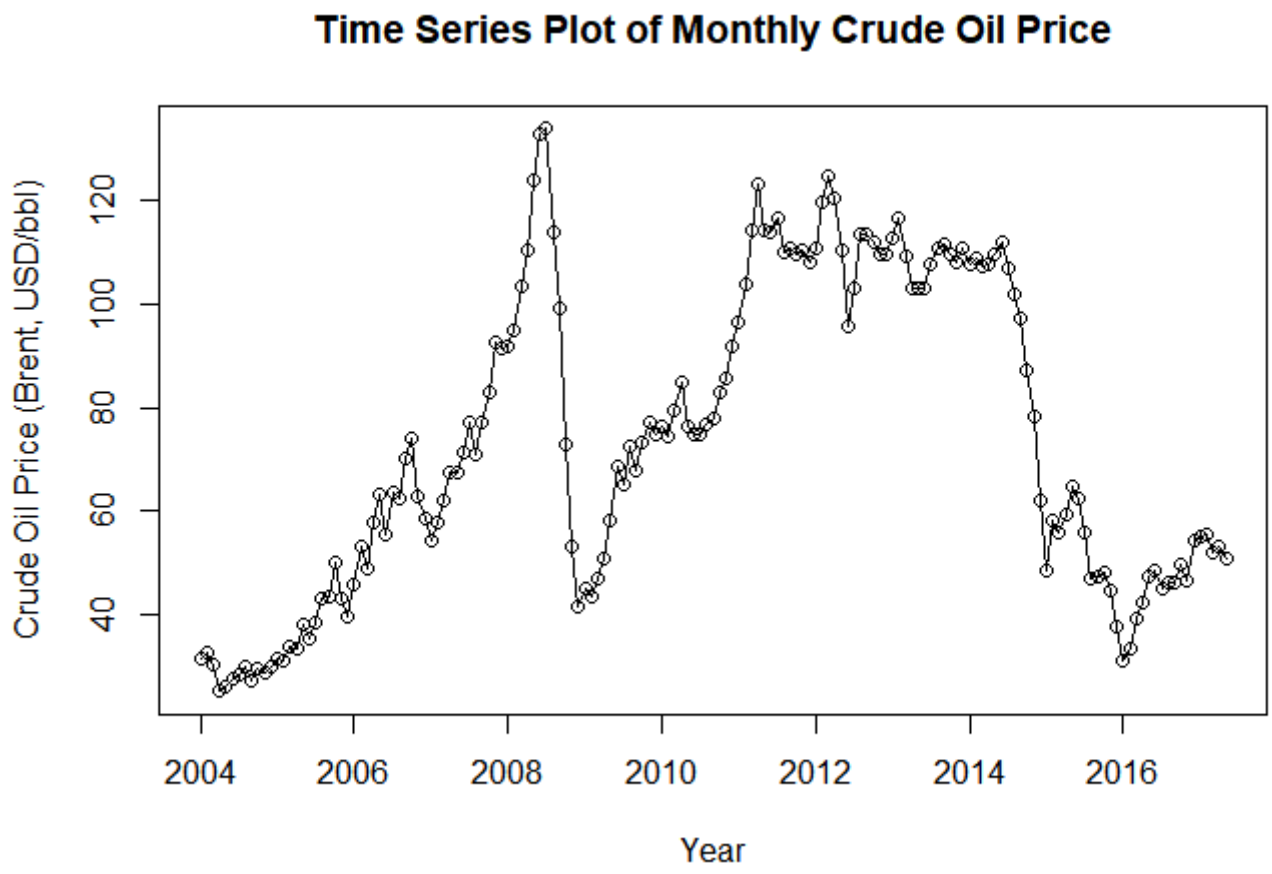

Figure 3

In `Figure 3`, the time series plot of monthly crude oil price. There is no signs of trend that one would forecast, changing variance and there is sign of intervention around 2008. Succeeding points and fluctuations implies the existence of an autoregressive behaviour.

```
plot(copper.ts,ylab="Copper Price (USD/Tonne)",xlab="Year",main="Time Series Plot of
         Monthly Copper Price",type="o")
```

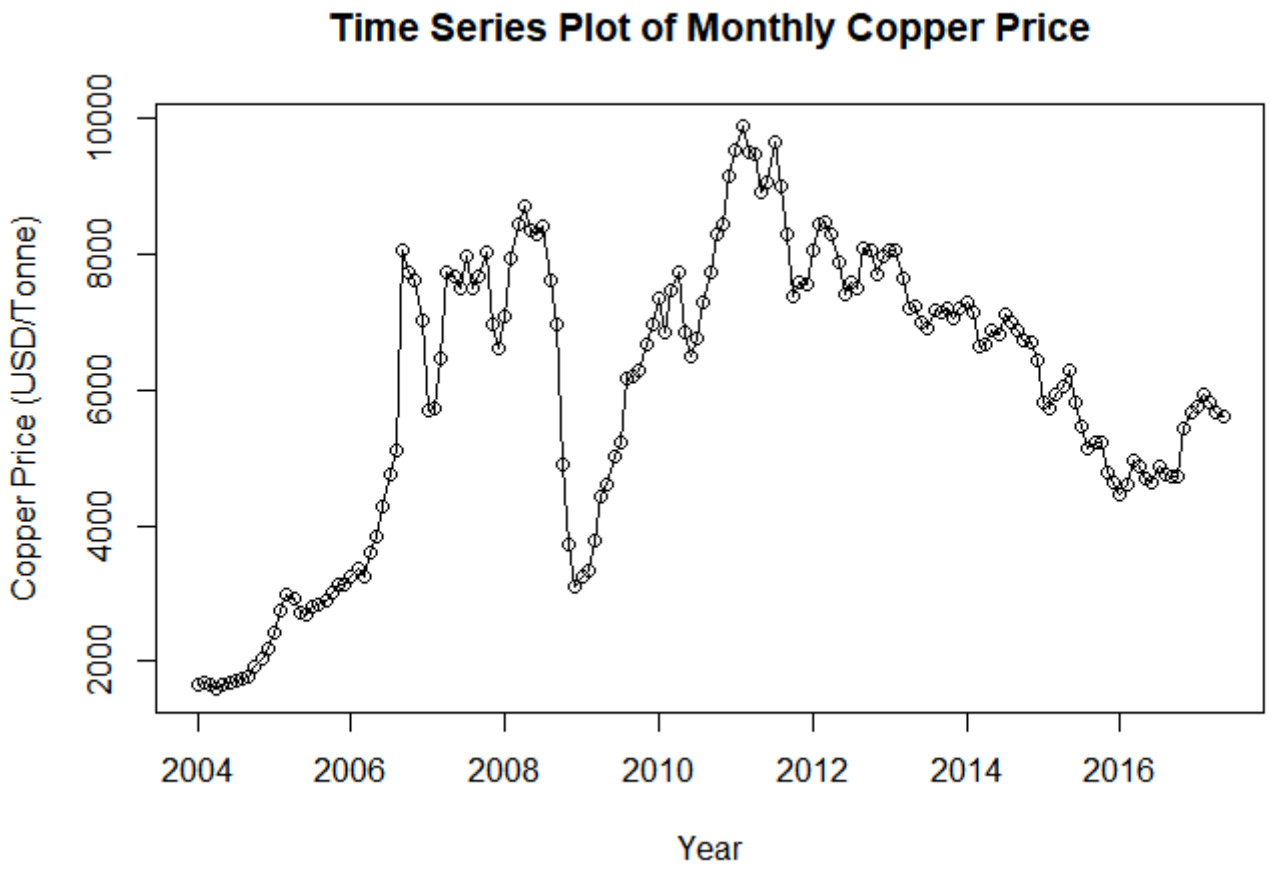

Figure 4

In `Figure 4`, the time series plot of monthly copper price appears to have a trend. There are no signs of seasonality, changing variance and there is sign of intervention around 2008. Succeeding points and fluctuations implies the existence of an autoregressive behaviour.

```
data.scaled<-scale(data.ts)

plot(data.scaled,plot.type="s",col=c("red","green","blue","black"),main="Time Series Plot
         of Scaled ASX Data",xlab="Year",ylab="Scaled Data")
legend("topleft",lty = 1,
         text.width=1.7,col=c("red","green","blue","black"),c("Ords","Gold","Crude
         Oil","Copper"))
```

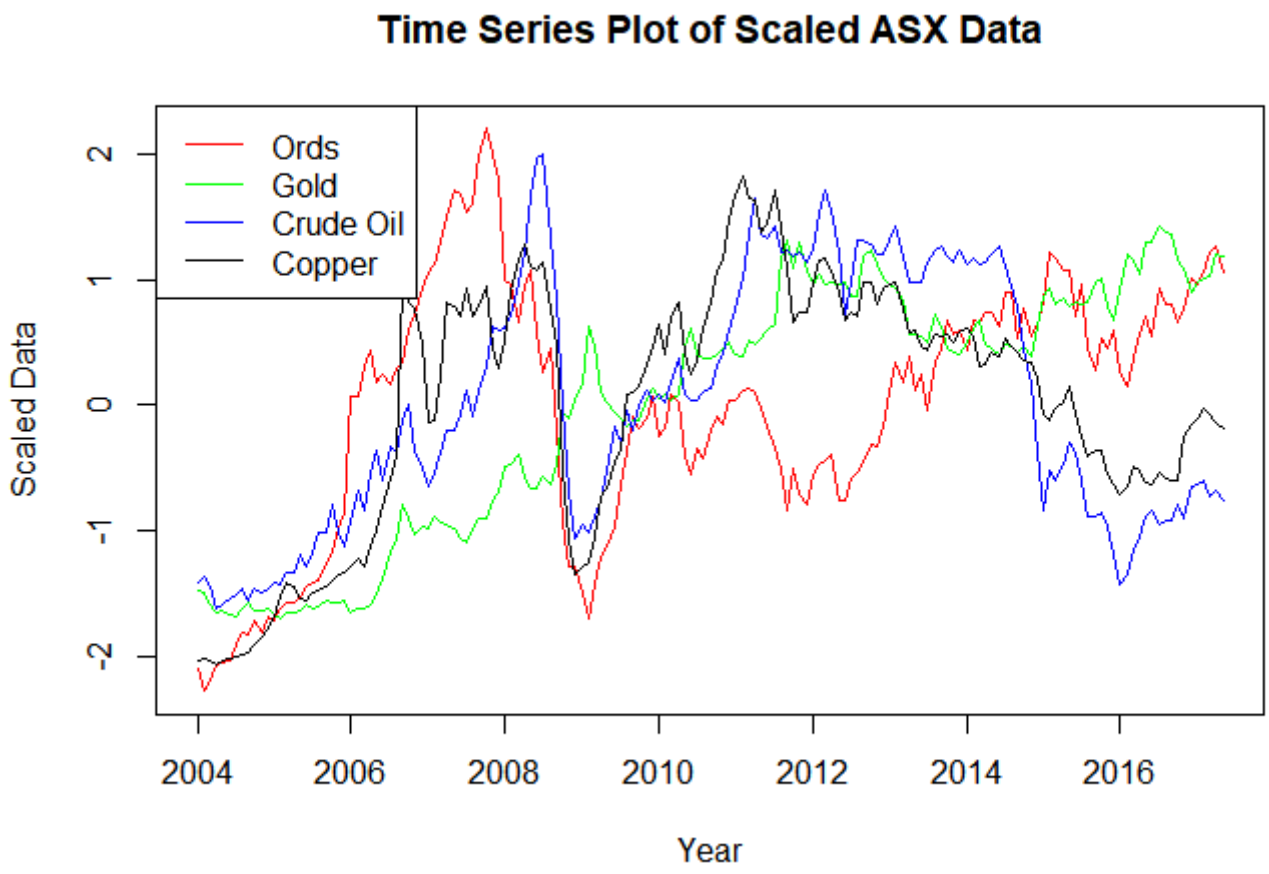

Figure 5

`Figure 5` contains a time series plot of all Ords, index, crude oil, and copper price. To compare these data and fit them in a graph, scaling and centering is done to the data by using the `scale` function.

4.0 Existence of Non-Stationary Test

To test the existence of non-stationary, we will conduct the test first by plotting the ACF and PACF plots using the functions `ACF` and `PACF`. Then, we shall conduct the Augmented Dickey-Fuller test (ADF) and Philip-Perron (PP) test to test the hypothesis. The Augmented Dickey-Fuller test will use the function `adf.test`, and the Philip-Perron test uses the function `pp.test`.

4.1 ASX All Ordinaries (Ords) Price Index

```
par(mfrow=c(1,2))
acf(ords.ts,main="Sample ACF for Ords Series",cex.main=0.65)
pacf(ords.ts,main="Sample PACF for Ords Series",cex.main=0.05)
```

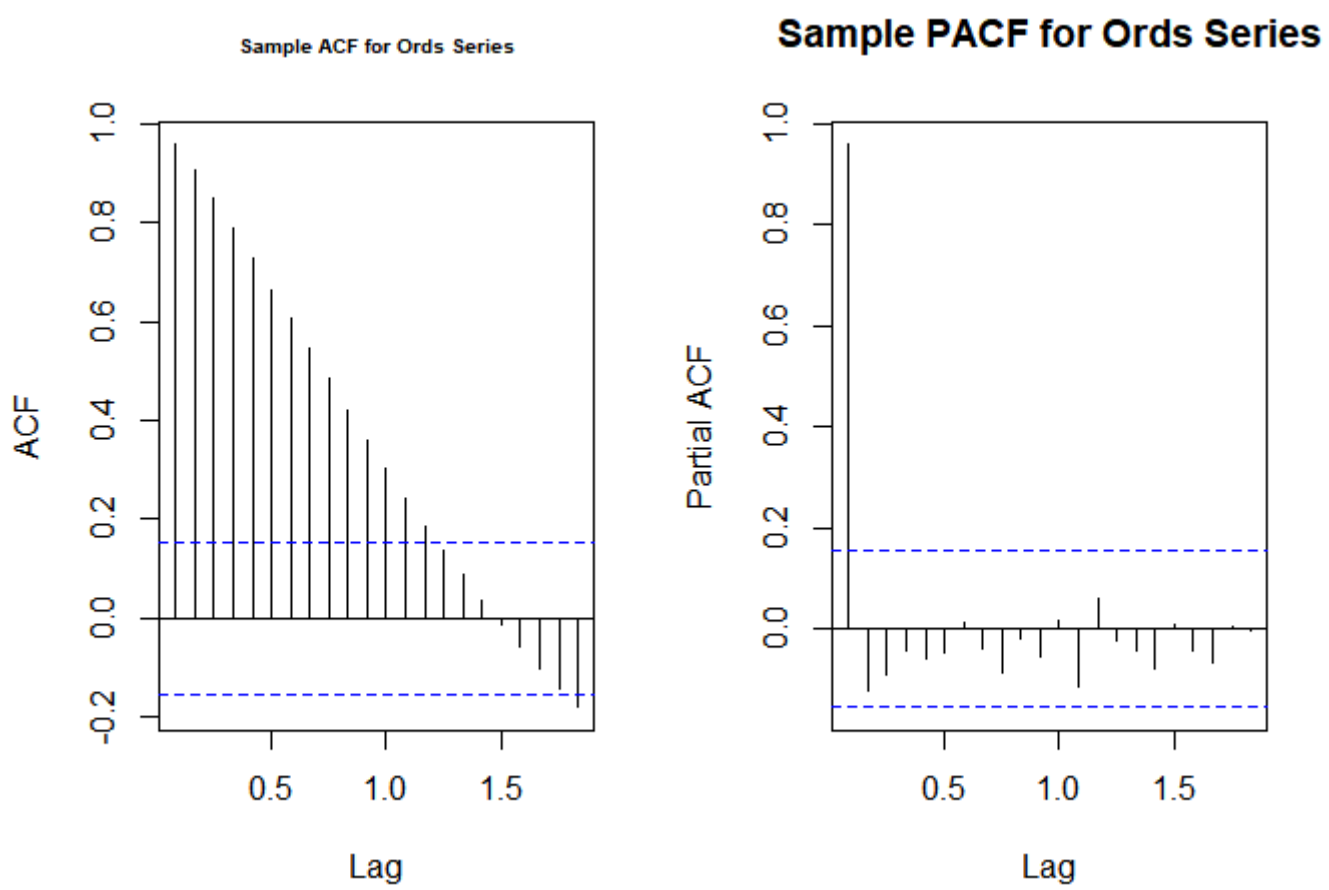

Figure 6

`Figure 6` above shows the ACF and PACF plots for the time series of monthly ords price. Here we can observe that there is an decaying pattern and most lags are above the 95% confidence interval in the ACF plot. In the PACF plot, we can observe that the first lag is significant, giving evidence that the time series is non-stationary. However, further testing needs to be conducted to be sure.

```
#Augmented Dickey-Fuller Test
ar(ords.ts)
```

```
##
## Call:
## ar(x = ords.ts)
##
## Coefficients:
##       1        2
##  1.0726  -0.1208
##
## Order selected 2  sigma^2 estimated as  67176
```

```
adf.test(ords.ts,k=2)
```

```
##
##  Augmented Dickey-Fuller Test
```

```
##
## data:  ords.ts
## Dickey-Fuller = -2.442, Lag order = 2, p-value = 0.392
## alternative hypothesis: stationary
```

```
k=trunc(12*((length(ords.ts)/100)^(1/4)))
print(k)
```

```
## [1] 13
```

```
adf.test(ords.ts,k=k)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ords.ts
## Dickey-Fuller = -2.8677, Lag order = 13, p-value = 0.2144
## alternative hypothesis: stationary
```

```
adf.test(ords.ts)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ords.ts
## Dickey-Fuller = -2.6995, Lag order = 5, p-value = 0.2846
## alternative hypothesis: stationary
```

In the above output, we used 3 approaches of the ADF test for the monthly ords price, first approach uses 2 lag length, second approach uses 13 lag length, and the third approach uses 5 lag length. In all approaches, we can conclude that the series are non stationary at 5% level of significance as p value is greater than 0.05.

```
#Phillips-Perron Test
pp.test(ords.ts,lshort=TRUE)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  ords.ts
## Dickey-Fuller Z(alpha) = -8.7417, Truncation lag parameter = 4, p-value
```

```
## = 0.6135
## alternative hypothesis: stationary
```

```
pp.test(ords.ts,lshort=FALSE)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  ords.ts
## Dickey-Fuller Z(alpha) = -10.292, Truncation lag parameter = 13,
## p-value = 0.5246
## alternative hypothesis: stationary
```

In the above output of the PP test, we used 2 approaches where the difference was the argument `lshort`, with 1 output being `true` and another being `false`. According to the PP test, we can conclude that the monthly ords price series is non-stationary at 5% level of significance as p value is greater than 0.05.

4.2 Gold Price

```
par(mfrow=c(1,2))
acf(gold.ts,main="Sample ACF for Gold Price Series",cex.main=0.65)
pacf(gold.ts,main="Sample PACF for Gold Price Series",cex.main=0.05)
```

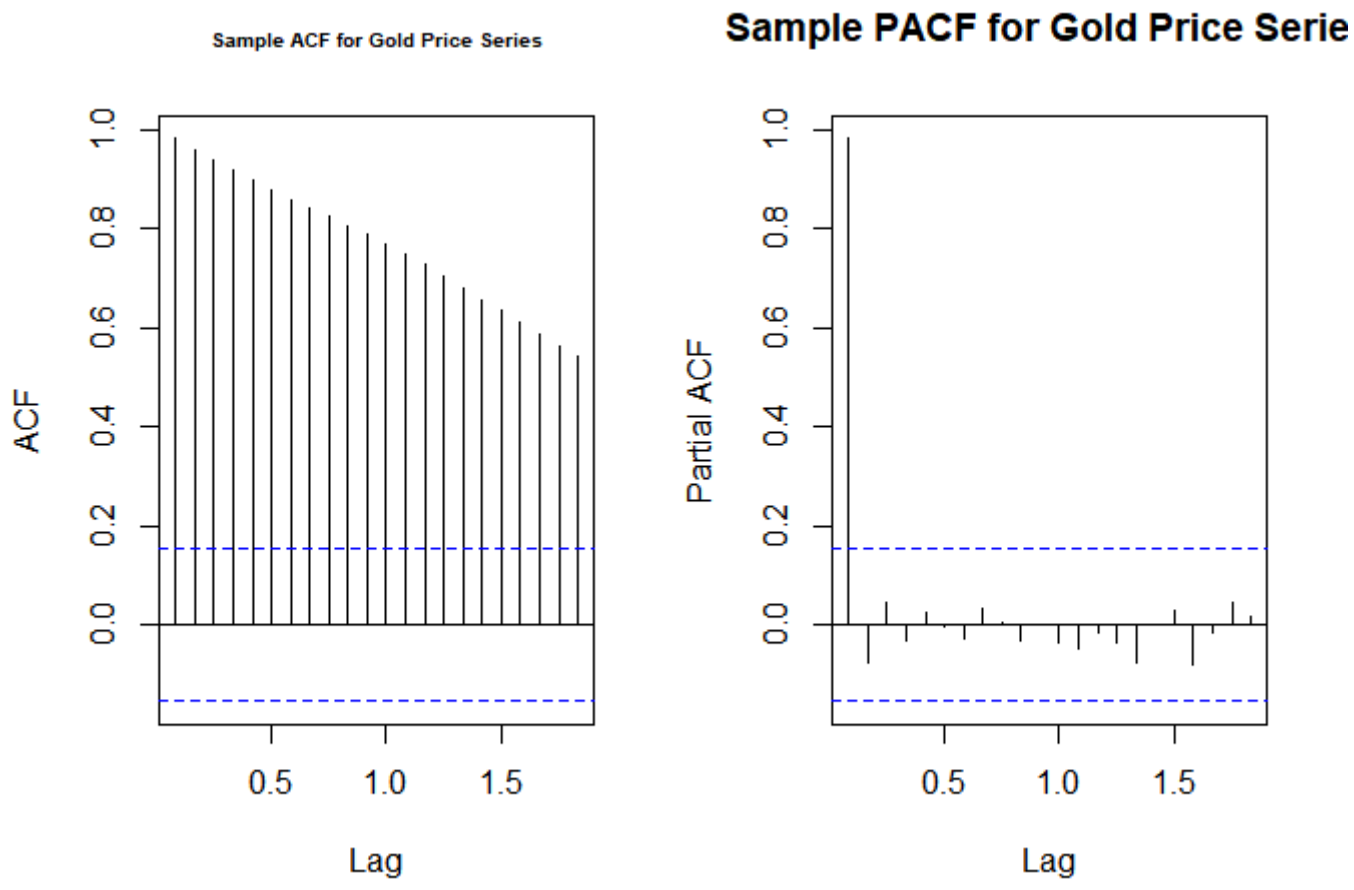

Figure 7

`Figure 7` above shows the ACF and PACF plots for the time series of monthly gold price. Here we can observe that there is an decaying pattern and all lags are above the 95% confidence interval in the ACF plot. In the PACF plot, we can observe that the first lag is significant, giving evidence that the time series is non-stationary. However, further testing needs to be conducted to be sure.

```
#Augmented Dickey-Fuller Test
ar(gold.ts)
```

```
##
## Call:
## ar(x = gold.ts)
##
## Coefficients:
##      1
## 0.9803
##
## Order selected 1  sigma^2 estimated as  6329
```

```
adf.test(gold.ts,k=1)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  gold.ts
## Dickey-Fuller = -2.2824, Lag order = 1, p-value = 0.4585
## alternative hypothesis: stationary
```

```
k=trunc(12*((length(gold.ts)/100)^(1/4)))
print(k)
```

```
## [1] 13
```

```
adf.test(gold.ts,k=k)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  gold.ts
```

```
## Dickey-Fuller = -1.4051, Lag order = 13, p-value = 0.8245
## alternative hypothesis: stationary
```

```
adf.test(gold.ts)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  gold.ts
## Dickey-Fuller = -1.8369, Lag order = 5, p-value = 0.6444
## alternative hypothesis: stationary
```

In the above output, we used 3 approaches of the ADF test for the monthly gold price, first approach uses 1 lag length, second approach uses 13 lag length, and the third approach uses 5 lag length. In all approaches, we can conclude that the series are non stationary at 5% level of significance as p value is greater than 0.05.

```
#Phillips-Perron Test
pp.test(gold.ts,lshort=TRUE)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  gold.ts
## Dickey-Fuller Z(alpha) = -9.2725, Truncation lag parameter = 4, p-value
## = 0.5831
## alternative hypothesis: stationary
```

```
pp.test(gold.ts,lshort=FALSE)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  gold.ts
## Dickey-Fuller Z(alpha) = -7.3731, Truncation lag parameter = 13,
## p-value = 0.692
## alternative hypothesis: stationary
```

In the above output of the PP test, we used 2 approaches where the difference was the argument `lshort`, with 1 output being `true` and another being `false`. According to the PP test, we can conclude that the monthly gold price series is non-stationary at 5% level of significance as p value is greater than 0.05.

4.3 Crude Oil Price

```
par(mfrow=c(1,2))
acf(crude.ts,main="Sample ACF for Crude Oil Series",cex.main=0.65)
pacf(crude.ts,main="Sample PACF for Crude Oil Series",cex.main=0.05)
```

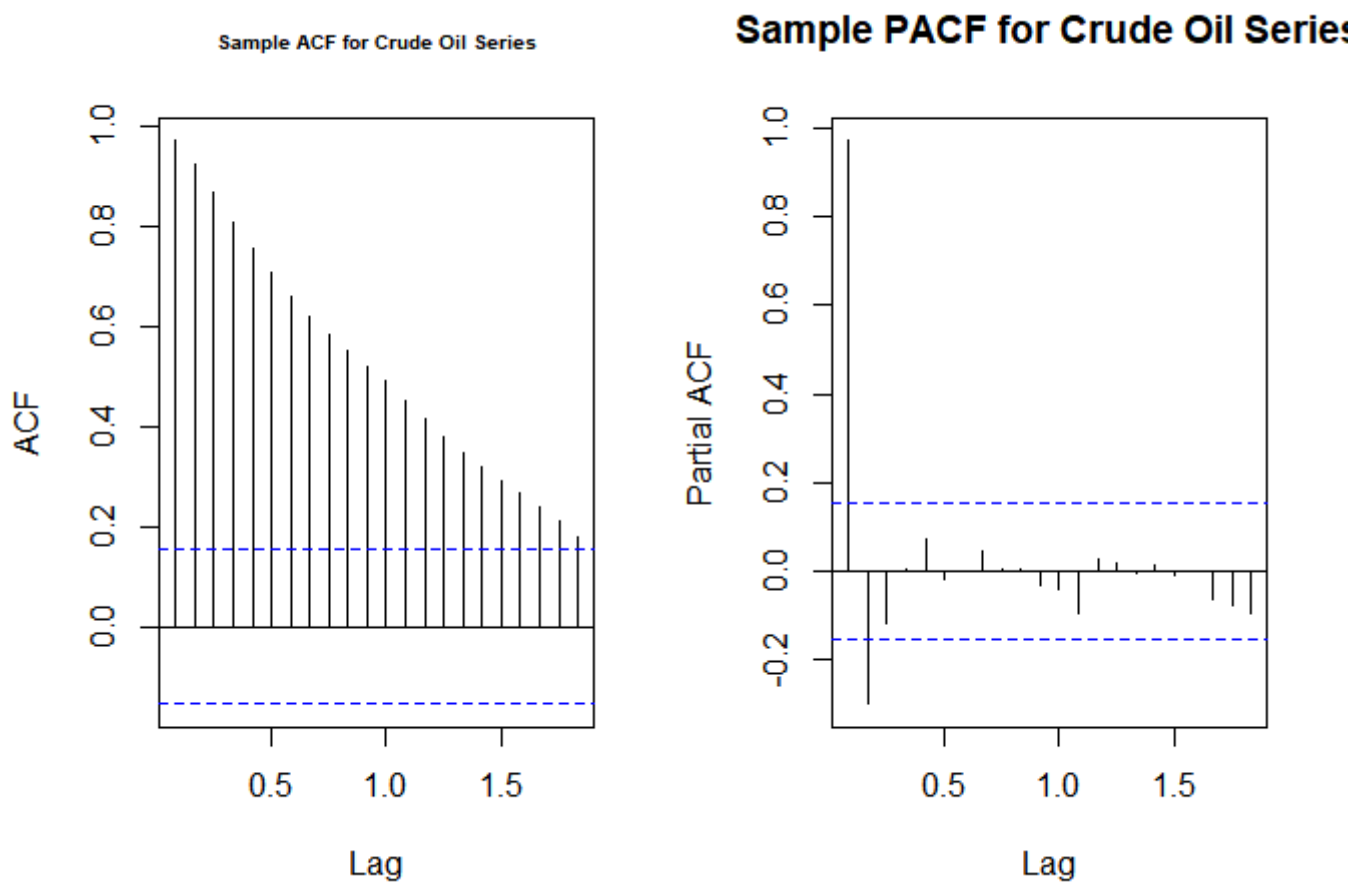

Figure 8

`Figure 8` above shows the ACF and PACF plots for the time series of monthly crude oil price. Here we can observe that there is an decaying pattern and all lags are above the 95% confidence interval in the ACF plot. In the PACF plot, we can observe that the first lag is significant, giving evidence that the time series is non-stationary. However, further testing needs to be conducted to be sure.

```
#Augmented Dickey-Fuller Test
ar(crude.ts)
```

```
##
## Call:
## ar(x = crude.ts)
##
## Coefficients:
##       1        2        3
##  1.2243  -0.1493  -0.1186
```

```
##
## Order selected 3  sigma^2 estimated as  48.96
```

```
adf.test(crude.ts,k=1)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  crude.ts
## Dickey-Fuller = -1.9961, Lag order = 1, p-value = 0.578
## alternative hypothesis: stationary
```

```
k=trunc(12*((length(crude.ts)/100)^(1/4)))
print(k)
```

```
## [1] 13
```

```
adf.test(crude.ts,k=k)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  crude.ts
## Dickey-Fuller = -1.4909, Lag order = 13, p-value = 0.7887
## alternative hypothesis: stationary
```

```
adf.test(crude.ts)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  crude.ts
## Dickey-Fuller = -1.8523, Lag order = 5, p-value = 0.6379
## alternative hypothesis: stationary
```

In the above output, we used 3 approaches of the ADF test for the monthly crude oil price, first approach uses 3 lag length, second approach uses 13 lag length, and the third approach uses 5 lag length. In all approaches, we can conclude that the series are non stationary at 5% level of significance as p value is greater than 0.05.

```
#Phillips-Perron Test
pp.test(crude.ts,lshort=TRUE)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  crude.ts
## Dickey-Fuller Z(alpha) = -6.5927, Truncation lag parameter = 4, p-value
## = 0.7367
## alternative hypothesis: stationary
```

```
pp.test(crude.ts,lshort=FALSE)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  crude.ts
## Dickey-Fuller Z(alpha) = -5.0232, Truncation lag parameter = 13,
## p-value = 0.8267
## alternative hypothesis: stationary
```

In the above output of the PP test, we used 2 approaches where the difference was the argument `lshort`, with 1 output being `true` and another being `false`. According to the PP test, we can conclude that the monthly crude oil price series is non-stationary at 5% level of significance as p value is greater than 0.05.

4.4 Copper Price

```
par(mfrow=c(1,2))
acf(copper.ts,main="Sample ACF for Copper Series",cex.main=0.65)
pacf(copper.ts,main="Sample PACF for Copper Series",cex.main=0.05)
```

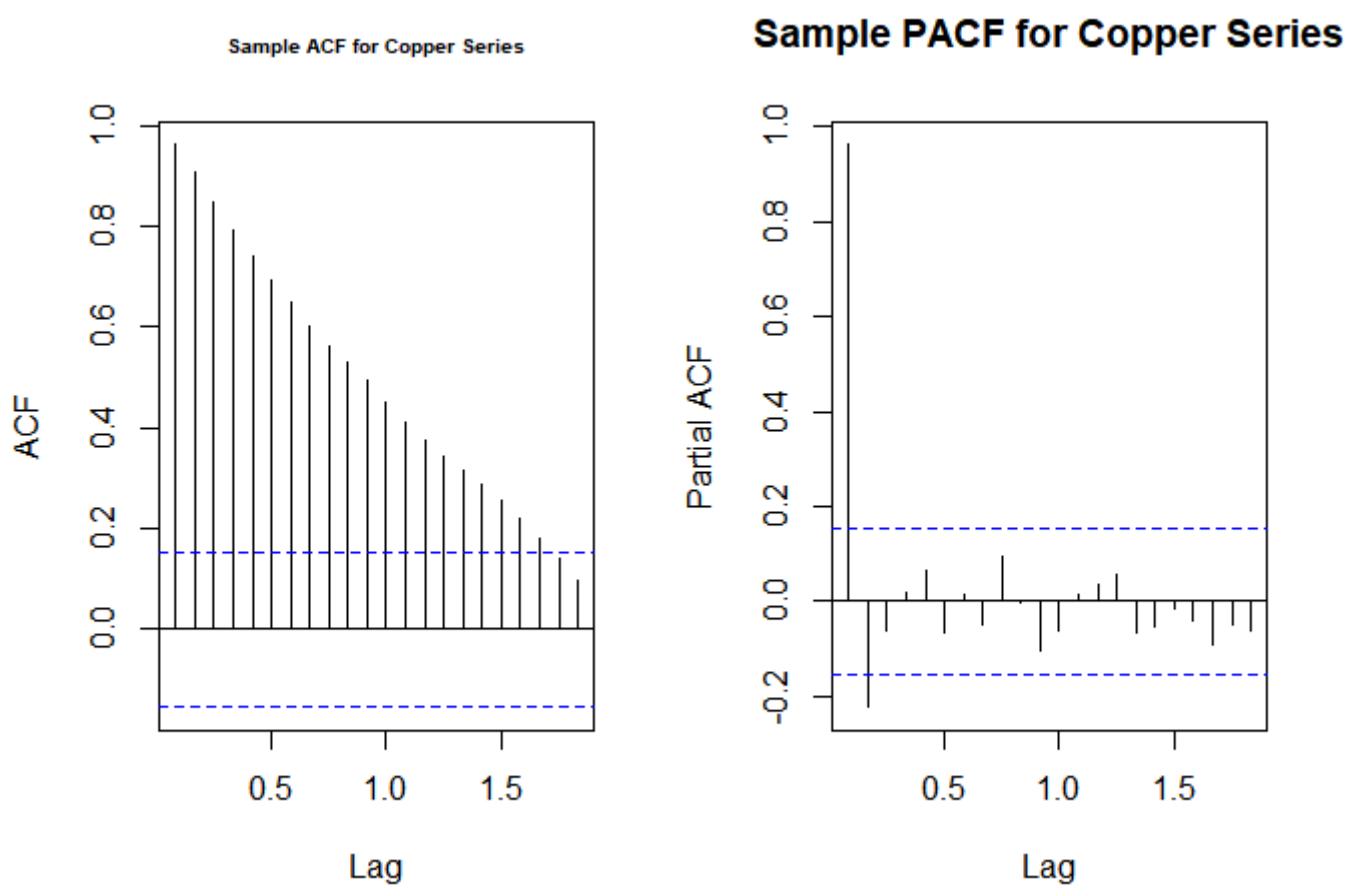

Figure 9

`Figure 9` above shows the ACF and PACF plots for the time series of monthly copper price. Here we can observe that there is an decaying pattern and most lags are above the 95% confidence interval in the ACF plot. In the PACF plot, we can observe that the first lag is significant, giving evidence that the time series is non-stationary. However, further testing needs to be conducted to be sure.

```
#Augmented Dickey-Fuller Test
ar(copper.ts)
```

```
##
## Call:
## ar(x = copper.ts)
##
## Coefficients:
##       1        2
##  1.1756  -0.2228
##
## Order selected 2  sigma^2 estimated as  331275
```

```
adf.test(copper.ts,k=1)
```

```
##
##  Augmented Dickey-Fuller Test
```

```
##
## data:  copper.ts
## Dickey-Fuller = -2.3148, Lag order = 1, p-value = 0.445
## alternative hypothesis: stationary
```

```
k=trunc(12*((length(copper.ts)/100)^(1/4)))
print(k)
```

```
## [1] 13
```

```
adf.test(copper.ts,k=k)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  copper.ts
## Dickey-Fuller = -1.9739, Lag order = 13, p-value = 0.5872
## alternative hypothesis: stationary
```

```
adf.test(copper.ts)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  copper.ts
## Dickey-Fuller = -2.2502, Lag order = 5, p-value = 0.472
## alternative hypothesis: stationary
```

In the above output, we used 3 approaches of the ADF test for the monthly copper price, first approach uses 2 lag length, second approach uses 13 lag length, and the third approach uses 5 lag length. In all approaches, we can conclude that the series are non stationary at 5% level of significance as p value is greater than 0.05.

```
#Phillips-Perron Test
pp.test(copper.ts,lshort=TRUE)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  copper.ts
## Dickey-Fuller Z(alpha) = -7.8883, Truncation lag parameter = 4, p-value
```

```
## = 0.6624
## alternative hypothesis: stationary
```

```
pp.test(copper.ts,lshort=FALSE)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  copper.ts
## Dickey-Fuller Z(alpha) = -6.5121, Truncation lag parameter = 13,
## p-value = 0.7413
## alternative hypothesis: stationary
```

In the above output of the PP test, we used 2 approaches where the difference was the argument `lshort`, with 1 output being `true` and another being `false`. According to the PP test, we can conclude that the monthly copper price series is non-stationary at 5% level of significance as p value is greater than 0.05.

For all the time series Ords, gold, crude oil, and copper, we can confirm that there is an existence of non-stationery as all the test concludes that the prices are non-stationary at 5% level of significance.

5.0 Decomposition

To look into the impact of the components in the time series data, we will use various decomposition methods to have a further insight of the independent seasonal and trend characteristics of the time series data. In our case, we will use the X-12-ARIMA decomposition, as well as the STL decomposition methods to decompose the data. We will not be using the classical decomposition method as there are other methods more efficient than this method.

In the first part, we will use the X-12-ARIMA decomposition as this method is an upgrade from the classical decomposition method as it accounts for day variation, holiday effects and other predictors. We will use the functions `x12` and `plotSeasFac` to decompose and visualize the time series data.

In the second part, we will use the STL decomposition as this method can handle any type of seasonality and there are more user controllable selections. We will use the function `stl` to decompose and visualize the time series data.

5.1 ASX All Ordinaries (Ords) Price Index

```
#X-12-ARIMA Decomposition
ords.x12<-x12(ords.ts)
plot(ords.x12,sa=TRUE,trend=TRUE,forecast=TRUE)
```

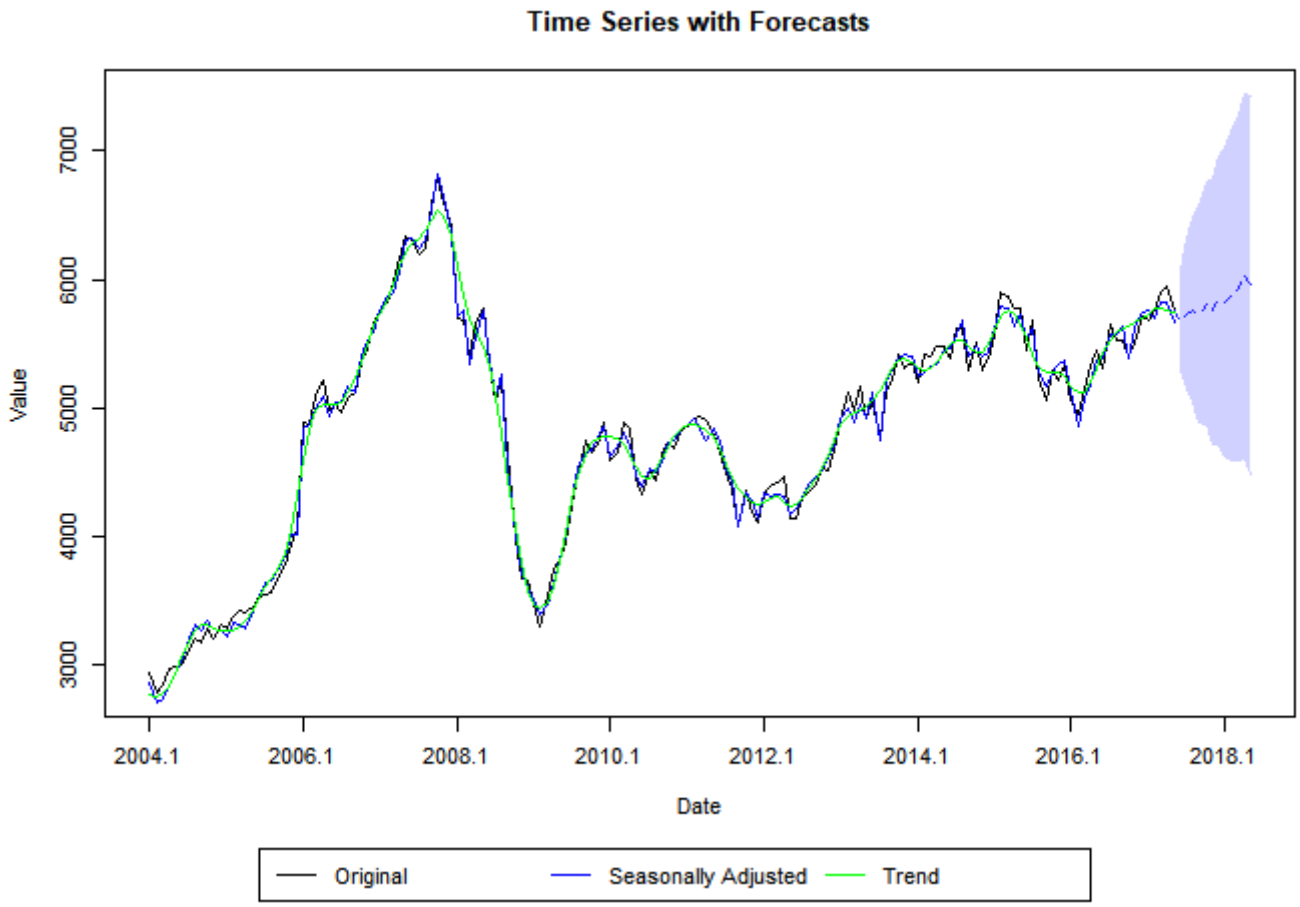

Figure 10

`Figure 10` shows the X-12-ARIMA output of the time series data for Ords price. The arguments `sa=TRUE` , `trend=TRUE` ,and `forecast=TRUE` allows us to seasonally adjust the data, show the trend line, and applies a 95% forecast at the end. Here we can observe that the seasonally adjusted series closely follows the original series of the Ords price.

```
plotSeasFac(ords.x12)
```

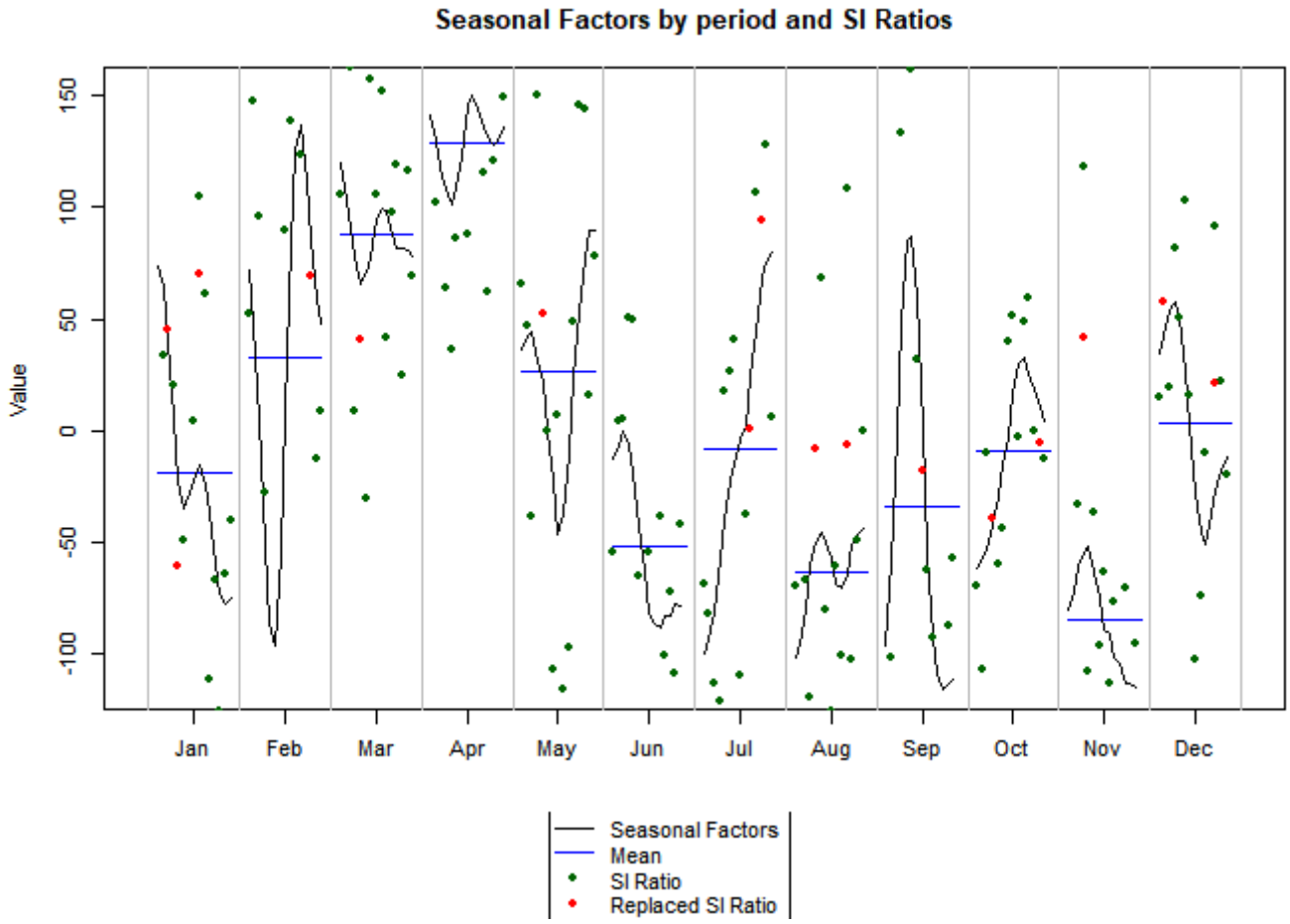

Figure 11

`Figure 11` shows the seasonal factor plots and SI ratios by period for Ords price. Here we can observe that for most of the months the seasonal factors divert from the mean level. Furthermore, we can observe from the SI ratios that there are some outliers or influential observations in the plot.

```
#STL Decomposition
ords.stl<-stl(ords.ts,t.window = 15,s.window="periodic",robust=TRUE)
plot(ords.stl)
```

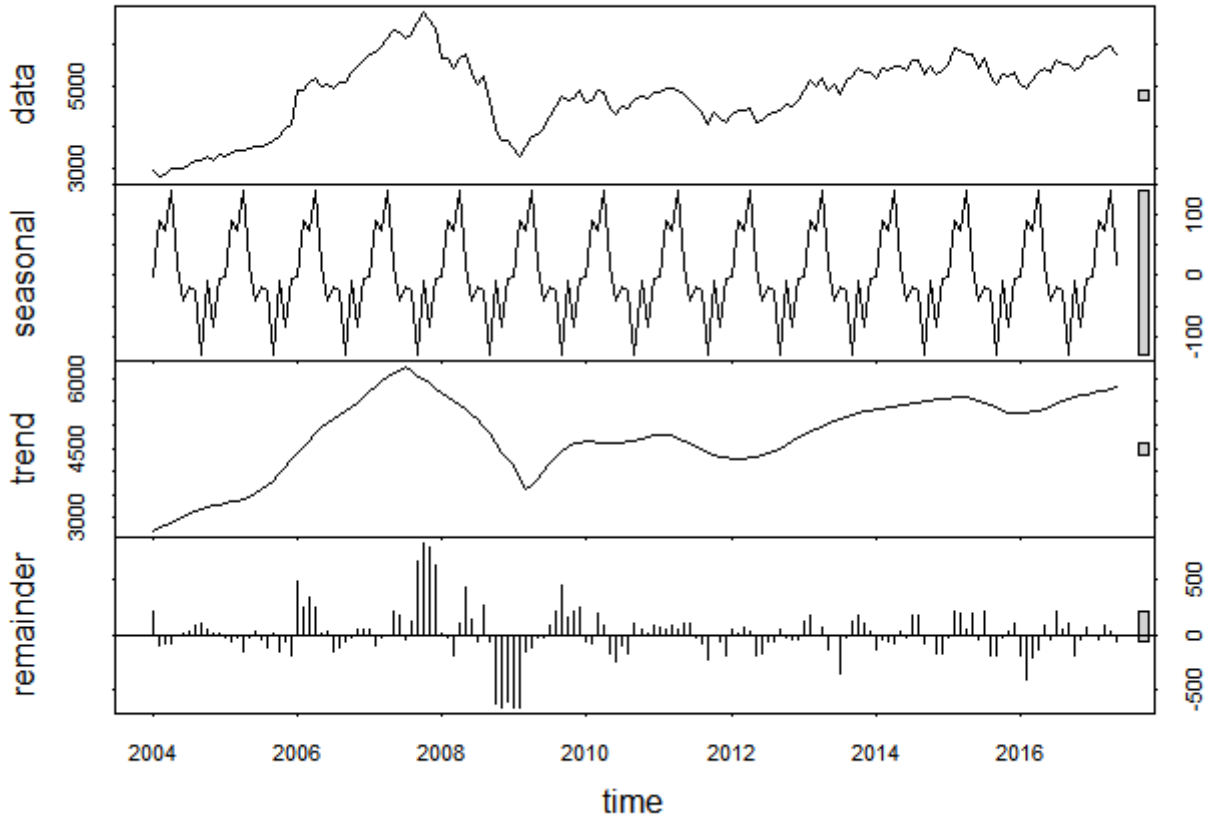

Figure 12

`Figure 12` shows the STL graph for Ords price, where we can observe that the there is no seasonal effect, and the trend follows the original series closely, and the remainder in the series is not smooth. Furthermore, here we can see the intervention point around 2008.

5.2 Gold Price

```
#X-12-ARIMA Decomposition
gold.x12<-x12(gold.ts)
plot(gold.x12,sa=TRUE,trend=TRUE,forecast=TRUE)
```

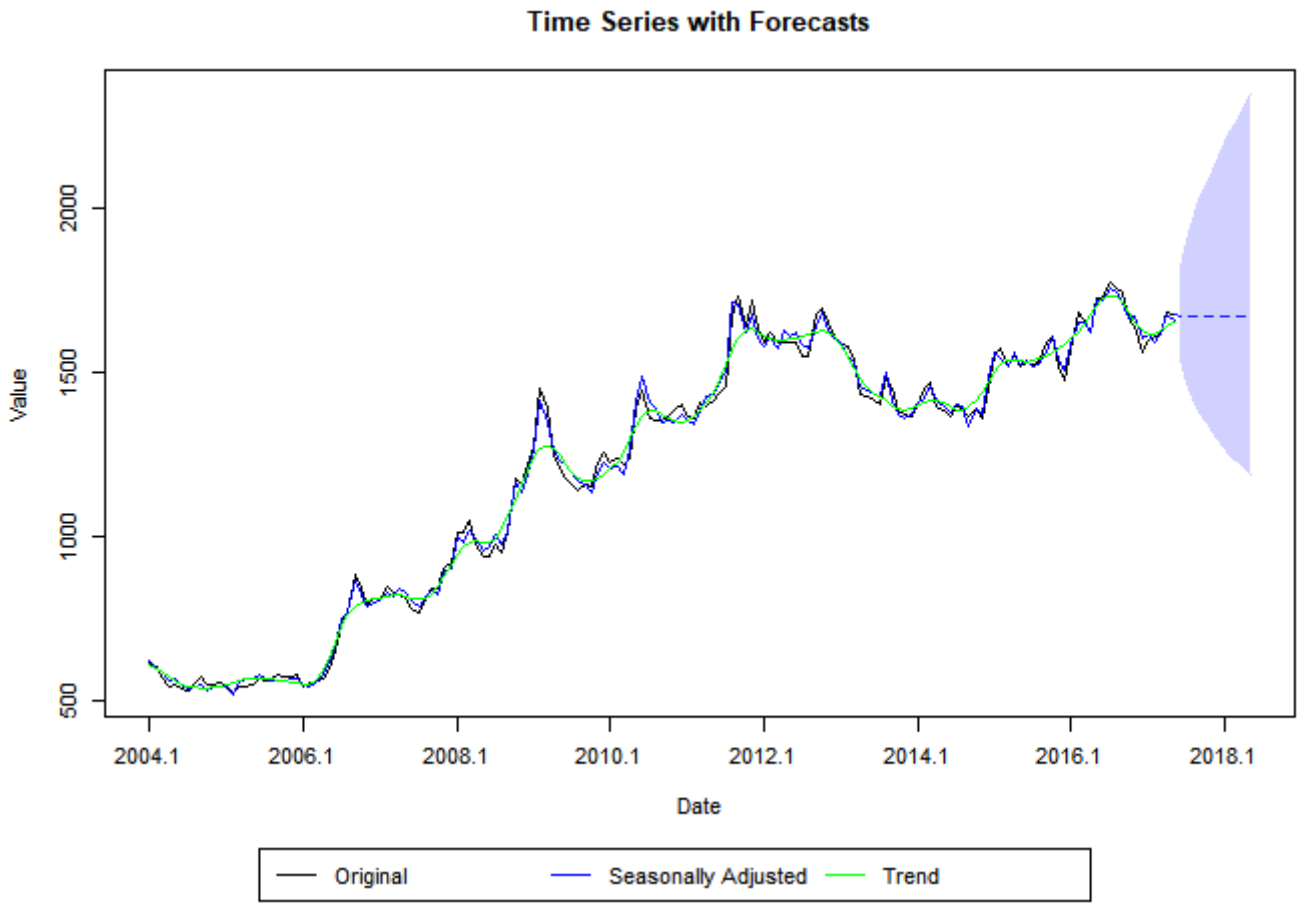

Figure 13

`Figure 13` shows the X-12-ARIMA output of the time series data for gold price. The arguments `sa=TRUE` , `trend=TRUE` ,and `forecast=TRUE` allows us to seasonally adjust the data, show the trend line, and applies a 95% forecast at the end. Here we can observe that the seasonally adjusted series closely follows the original series of the gold price.

```
plotSeasFac(gold.x12)
```

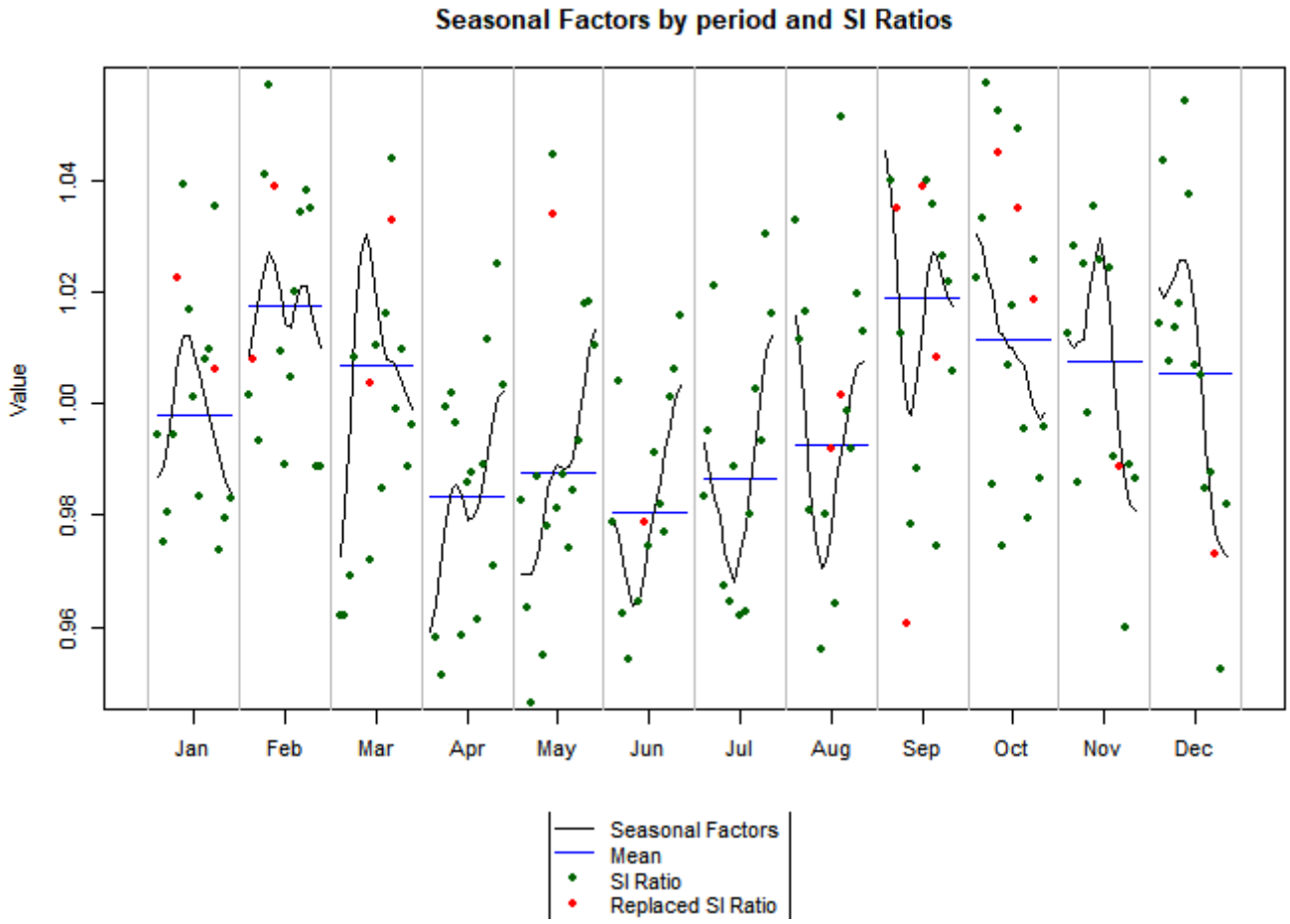

Figure 14

`Figure 14` shows the seasonal factor plots and SI ratios by period for gold price. Here we can observe that for most of the months the seasonal factors divert from the mean level. Furthermore, we can observe from the SI ratios that there are some outliers or influential observations in the plot.

```
#STL Decomposition
gold.stl<-stl(gold.ts,t.window = 15,s.window="periodic",robust=TRUE)
plot(gold.stl)
```

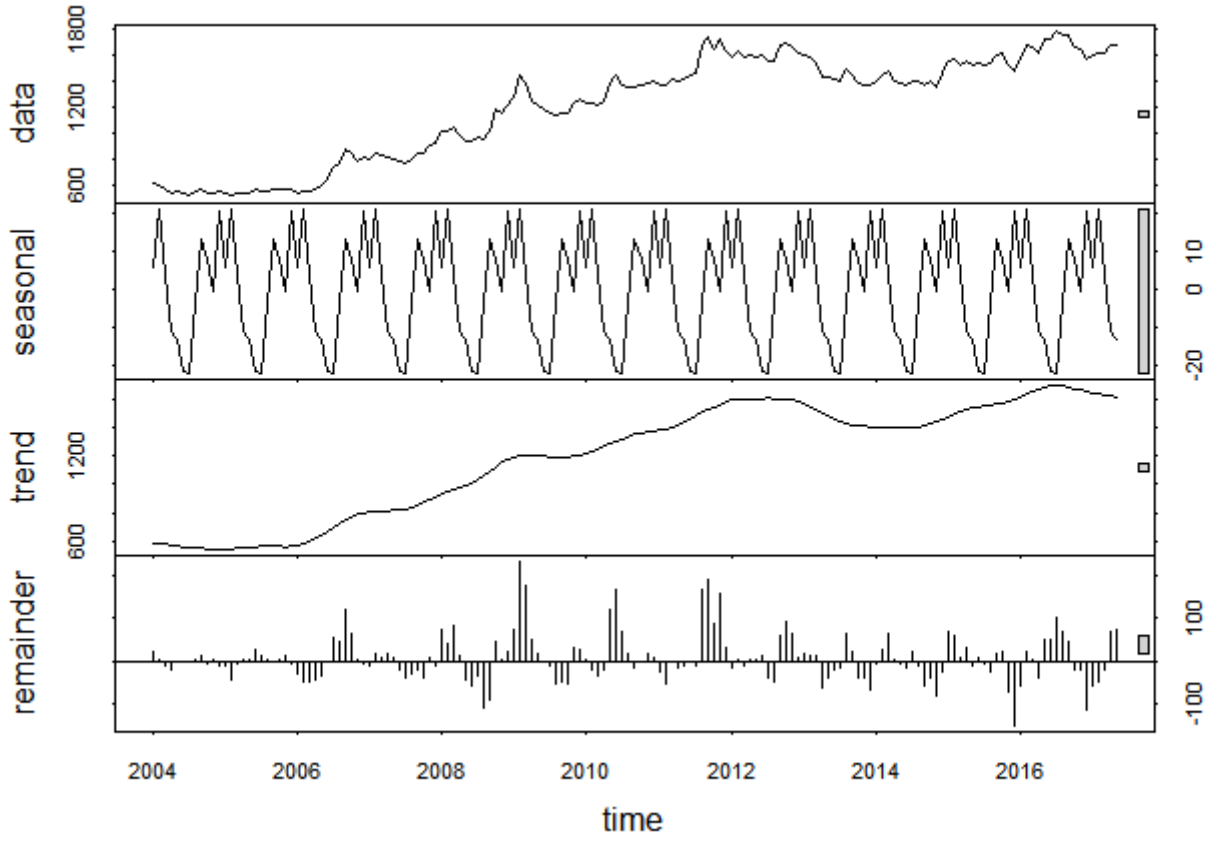

Figure 15

`Figure 15` shows the STL graph for gold price, where we can observe that the there is no seasonal effect, and the trend follows the original series closely, which is upwards, and the remainder in the series is not smooth.

5.3 Crude Oil Price

```
#X-12-ARIMA Decomposition
crude.x12<-x12(crude.ts)
plot(crude.x12,sa=TRUE,trend=TRUE,forecast=TRUE)
```

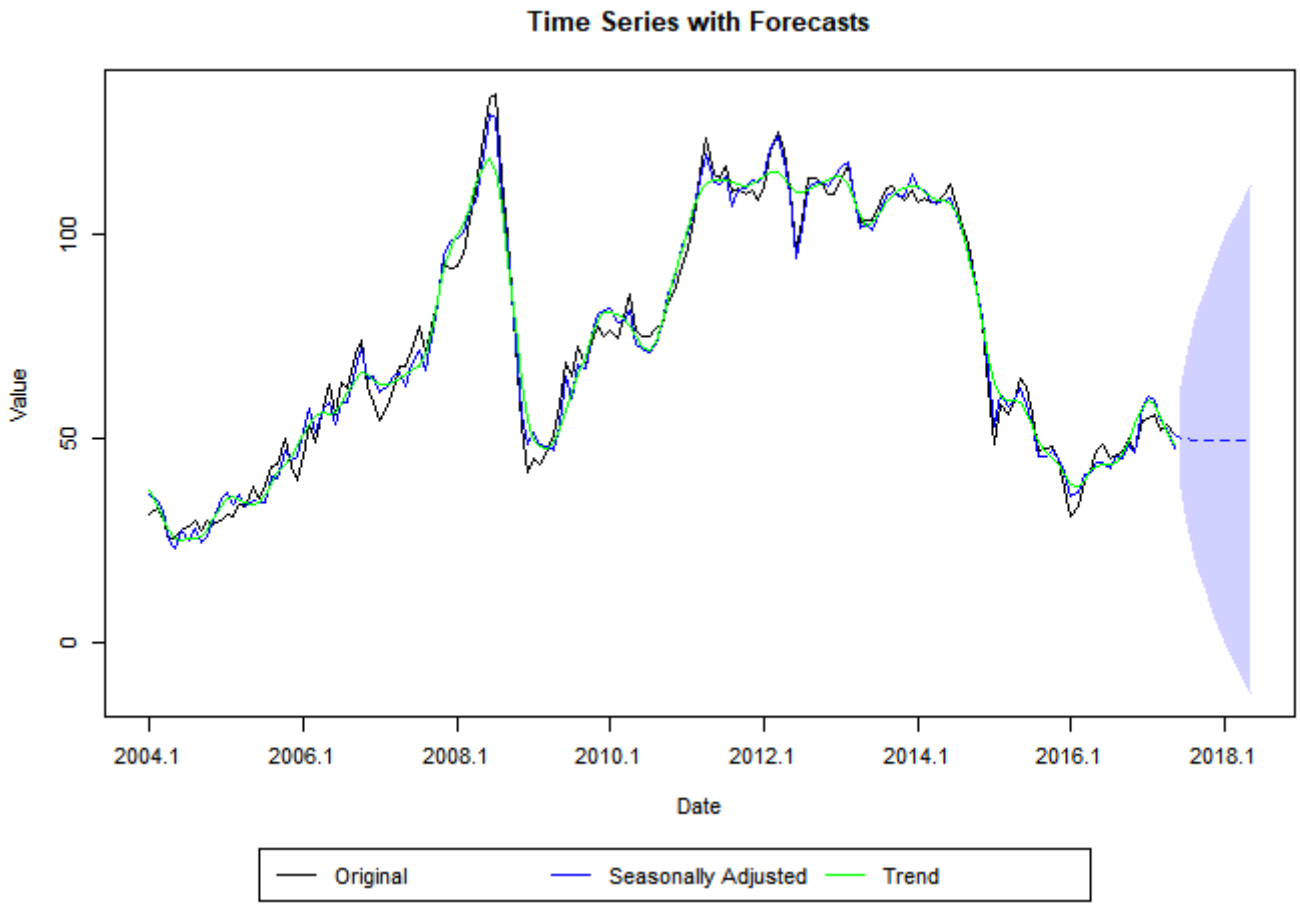

Figure 16

`Figure 16` shows the X-12-ARIMA output of the time series data for crude oil price. The arguments `sa=TRUE` , `trend=TRUE` ,and `forecast=TRUE` allows us to seasonally adjust the data, show the trend line, and applies a 95% forecast at the end. Here we can observe that the seasonally adjusted series closely follows the original series of the crude oil price.

```
plotSeasFac(crude.x12)
```

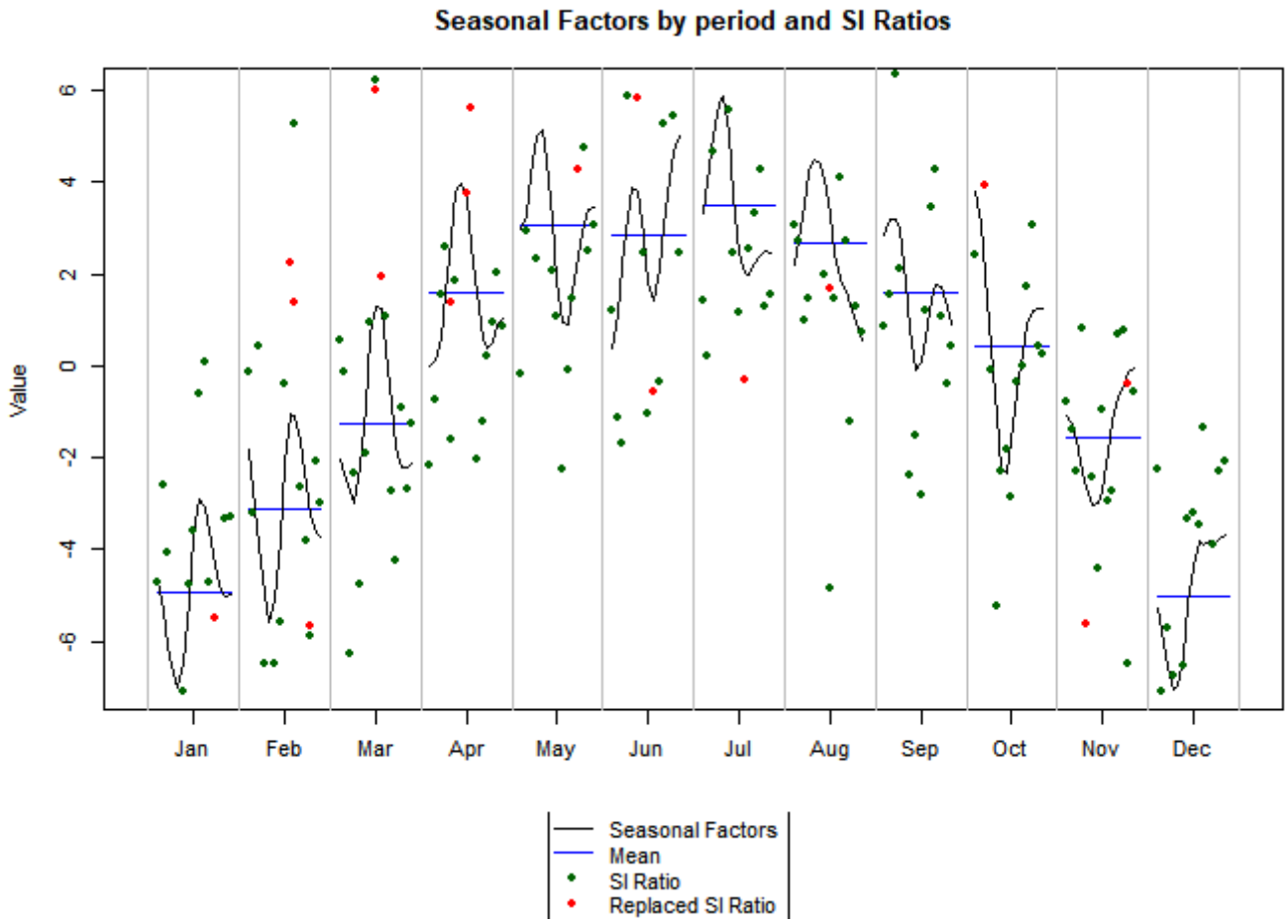

Figure 17

`Figure 17` shows the seasonal factor plots and SI ratios by period for crude oil price. Here we can observe that for most of the months the seasonal factors divert from the mean level. Furthermore, we can observe from the SI ratios that there are some outliers or influential observations in the plot.

```
#STL Decomposition
crude.stl<-stl(crude.ts,t.window = 15,s.window="periodic",robust=TRUE)
plot(crude.stl)
```

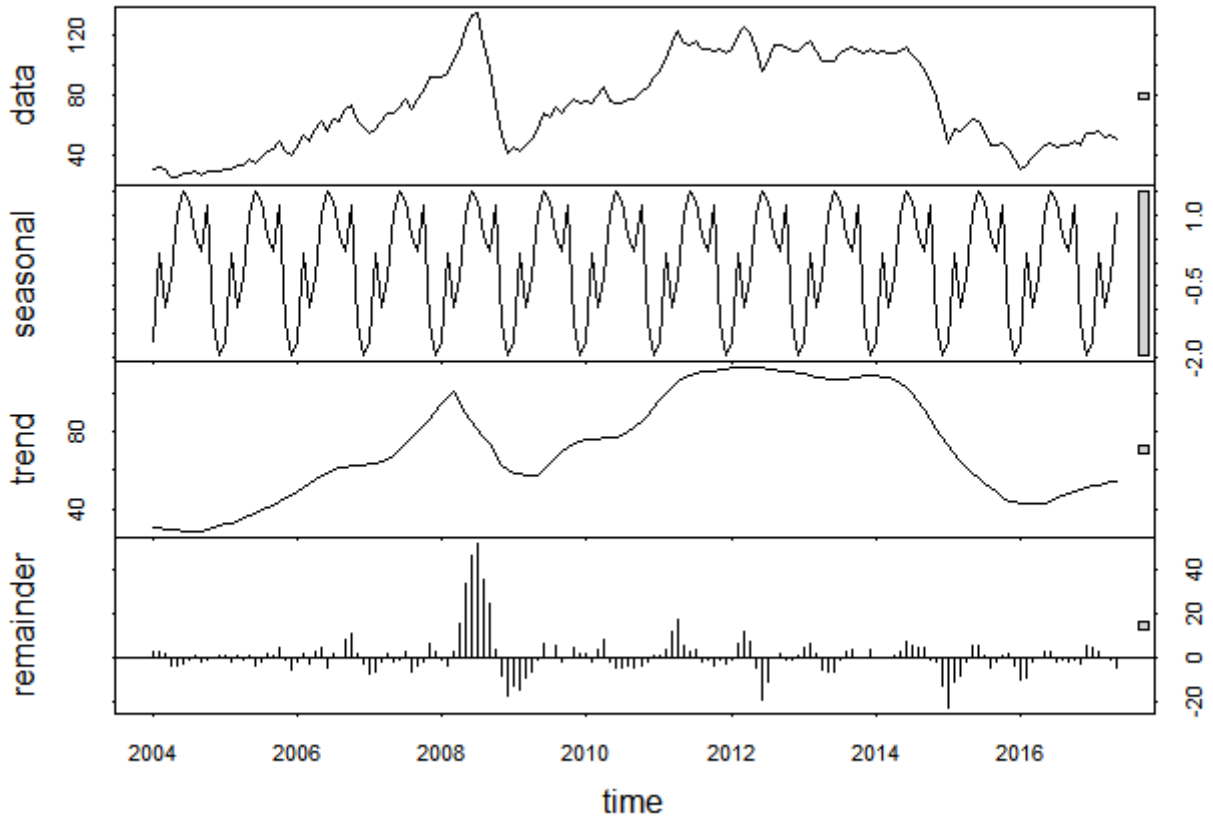

Figure 18

`Figure 18` shows the STL graph for crude oil price, where we can observe that the there is no seasonal effect, and the trend follows the original series closely, which is upwards, and the remainder in the series is not smooth. Furthermore, here we can see the intervention point around 2008.

5.4 Copper Price

```
#X-12-ARIMA Decomposition
copper.x12<-x12(copper.ts)
plot(copper.x12,sa=TRUE,trend=TRUE,forecast=TRUE)
```

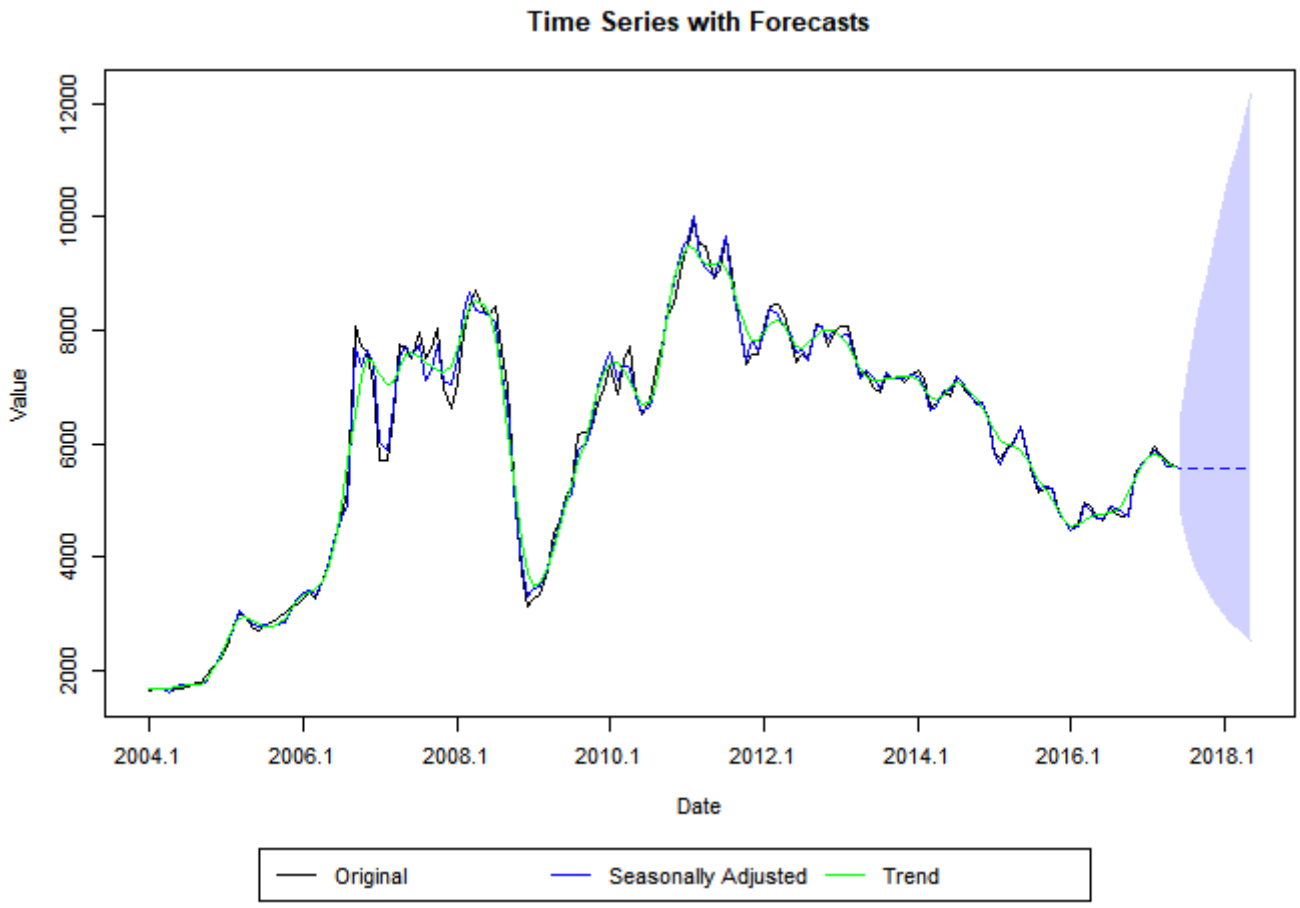

Figure 19

`Figure 19` shows the X-12-ARIMA output of the time series data for copper price. The arguments `sa=TRUE` , `trend=TRUE` ,and `forecast=TRUE` allows us to seasonally adjust the data, show the trend line, and applies a 95% forecast at the end. Here we can observe that the seasonally adjusted series closely follows the original series of the copper oil price.

```
plotSeasFac(copper.x12)
```

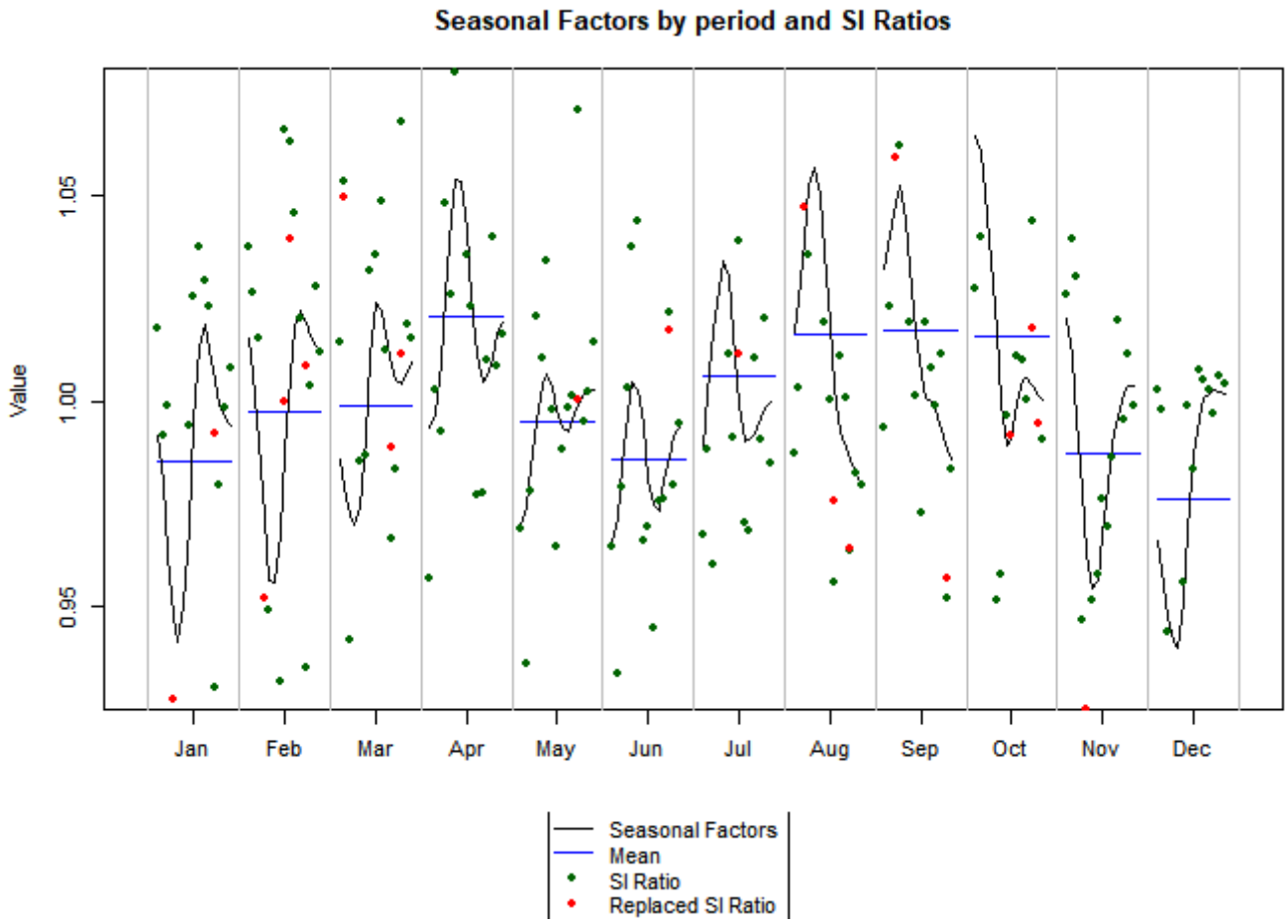

Figure 20

`Figure 20` shows the seasonal factor plots and SI ratios by period for copper price. Here we can observe that for most of the months the seasonal factors divert from the mean level. Furthermore, we can observe from the SI ratios that there are some outliers or influential observations in the plot.

```
#STL Decomposition
copper.stl<-stl(copper.ts,t.window = 15,s.window="periodic",robust=TRUE)
plot(copper.stl)
```

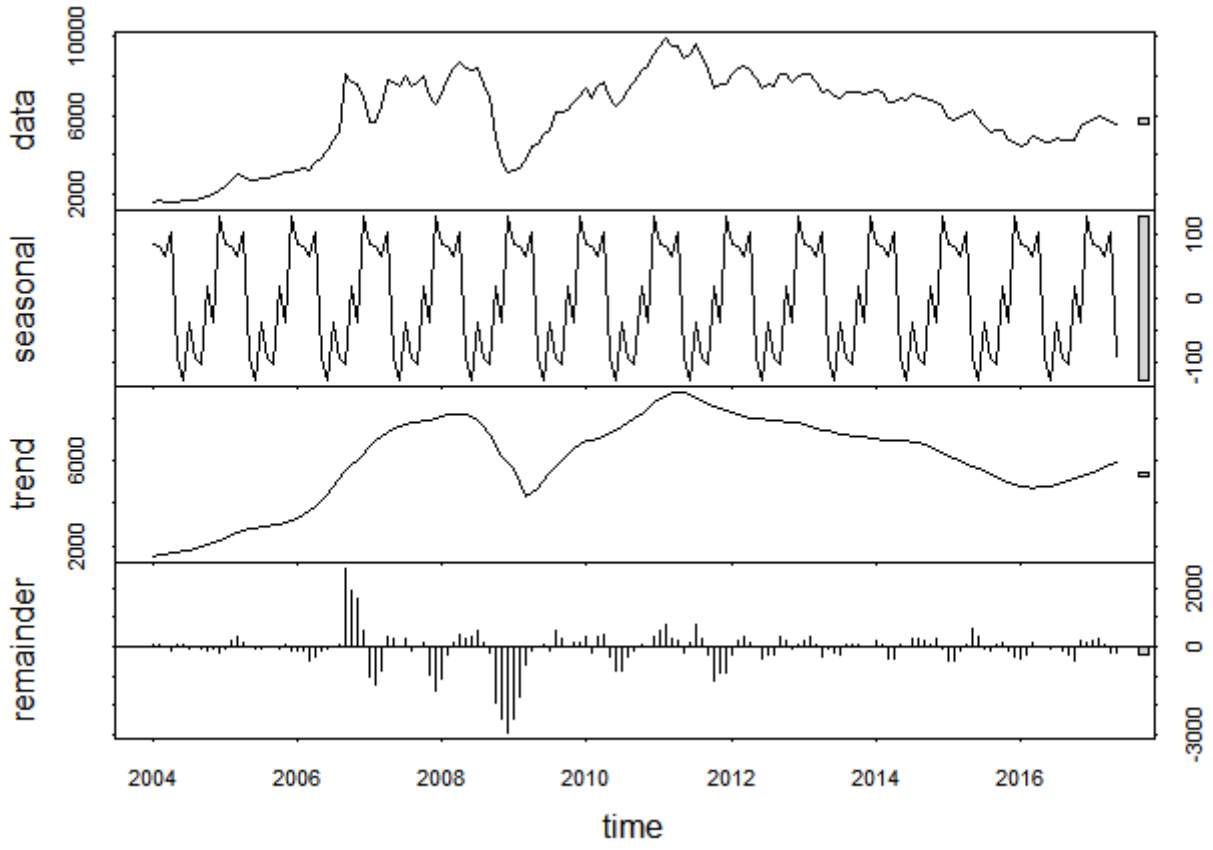

Figure 21

`Figure 21` shows the STL graph for copper price, where we can observe that the there is no seasonal effect, and the trend follows the original series closely, which is upwards, and the remainder in the series is not smooth. Furthermore, here we can see the intervention point around 2008.

6.0 Distributed Lag Modelling

In this section, we will look to investigate the most accurate and suitable distributed lag model (DLM) for the ASX price index. There will be four different models that we will be testing which are the Finite DLM, Polynomial DLMl, Koyck DLM, and Autoregressive DLM. Before we proceed to testing, we will first look at the correlation matrix for Ords, gold, crude oil, and copper price by using the `cor` function.

```
cor(data.ts)
```

```
##                           ASX price Gold price Crude Oil (Brent)_USD/bbl
## ASX price                 1.0000000  0.3431908                 0.3290338
## Gold price                0.3431908  1.0000000                 0.4366382
## Crude Oil (Brent)_USD/bbl 0.3290338  0.4366382                 1.0000000
## Copper_USD/tonne          0.5617864  0.5364213                 0.8664296
##                           Copper_USD/tonne
## ASX price                        0.5617864
## Gold price                       0.5364213
```

```
## Crude Oil (Brent)_USD/bbl        0.8664296
## Copper_USD/tonne                 1.0000000
```

From the correlation matrix output above, we can see that crude oil has the lowest correlation with ASX price, hence it is not recommended to use for testing. Gold price has a moderate correlation with ASX price, and copper price has the highest correlation with ASX price. Thus, ASX Price should be used as dependent variable y and the rest of the variables as independent variables x.

6.1 Finite Distributed Lag Model

For Finite DLM, we will be using the predictors gold price and copper price as the independent variable, and the ASX price will be the dependent variable. To perform this test, we will first create a loop function and use the `dlm` function to identify most appropriate number of lags for the DLM.

```
asx.dlm<-asx
colnames(asx.dlm)<-c("y","x1","x2","x3")
for ( i in 1:12){
  model1.1= dlm(formula = y ~ x1 + x3, data = data.frame(asx.dlm), q = i )
  cat("q =", i, "AIC =", AIC(model1.1$model), "BIC =", BIC(model1.1$model),"\n")
}
```

```
## q = 1 AIC = 2577.989 BIC = 2596.44
## q = 2 AIC = 2564.871 BIC = 2589.423
## q = 3 AIC = 2551.583 BIC = 2582.209
## q = 4 AIC = 2538.084 BIC = 2574.759
## q = 5 AIC = 2523.864 BIC = 2566.562
## q = 6 AIC = 2509.866 BIC = 2558.561
## q = 7 AIC = 2495.447 BIC = 2550.112
## q = 8 AIC = 2481.403 BIC = 2542.012
## q = 9 AIC = 2467.268 BIC = 2533.793
## q = 10 AIC = 2454.097 BIC = 2526.512
## q = 11 AIC = 2440.381 BIC = 2518.658
## q = 12 AIC = 2426.737 BIC = 2510.848
```

```
model1.2<-dlm(formula = y ~ x1 + x3, data = data.frame(asx.dlm),q=12)
summary(model1.2)
```

```
##
## Call:
## lm(formula = as.formula(model.formula), data = design)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1207.47  -600.48    80.48   521.41  1620.51
##
```

```
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.013e+03  2.698e+02  14.872   <2e-16 ***
## x1.t        -8.089e-01  1.290e+00  -0.627    0.532
## x1.1         2.345e-01  1.883e+00   0.125    0.901
## x1.2         1.681e-01  1.916e+00   0.088    0.930
## x1.3         1.664e-01  1.922e+00   0.087    0.931
## x1.4        -2.845e-01  1.910e+00  -0.149    0.882
## x1.5         3.780e-01  1.916e+00   0.197    0.844
## x1.6        -1.767e-01  1.946e+00  -0.091    0.928
## x1.7         5.962e-01  1.954e+00   0.305    0.761
## x1.8        -4.992e-01  1.932e+00  -0.258    0.797
## x1.9         1.807e-01  1.958e+00   0.092    0.927
## x1.10       -4.389e-03  1.971e+00  -0.002    0.998
## x1.11        5.442e-02  1.964e+00   0.028    0.978
## x1.12        2.967e-01  1.292e+00   0.230    0.819
## x3.t         1.306e-01  1.467e-01   0.890    0.375
## x3.1         2.777e-02  2.390e-01   0.116    0.908
## x3.2         4.097e-02  2.395e-01   0.171    0.864
## x3.3         1.752e-02  2.354e-01   0.074    0.941
## x3.4         4.280e-03  2.344e-01   0.018    0.985
## x3.5        -2.112e-02  2.379e-01  -0.089    0.929
## x3.6         3.132e-02  2.392e-01   0.131    0.896
## x3.7         2.104e-04  2.404e-01   0.001    0.999
## x3.8        -1.440e-02  2.428e-01  -0.059    0.953
## x3.9        -5.916e-02  2.445e-01  -0.242    0.809
## x3.10       -3.589e-02  2.426e-01  -0.148    0.883
## x3.11        1.175e-02  2.385e-01   0.049    0.961
## x3.12       -4.120e-02  1.510e-01  -0.273    0.785
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 762.4 on 122 degrees of freedom
## Multiple R-squared:  0.1935, Adjusted R-squared:  0.02168
## F-statistic: 1.126 on 26 and 122 DF,  p-value: 0.3234
##
## AIC and BIC values for the model:
##        AIC      BIC
## 1 2426.737 2510.848
```

In the output above, we can see that since a `AIC` and `BIC` values are lowest at 12 lags, we will use `q=12` as an argument to perform the DLM testing. By using the `summary` function, we can observe that in `model1.2` the lag weights of the predictors are not significant at the 5% statistical level as p-value > 0.05. Furthermore, there is a low adjusted R-square value of 0.02168.

```
checkresiduals(model1.2$model$residuals)
```

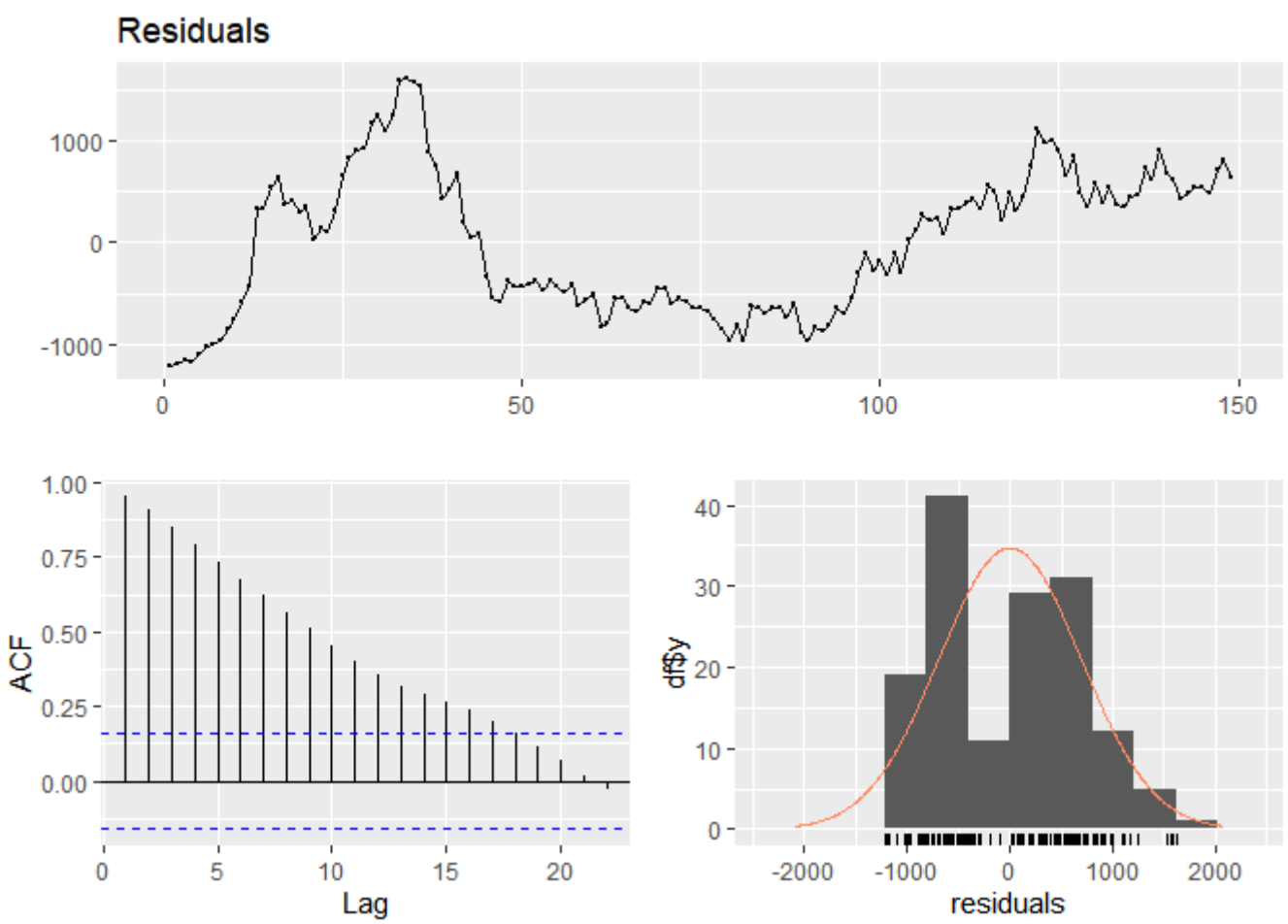

Figure 22

```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 813.94, df = 10, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 10
```

```
bgtest(model1.2$model)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model1.2$model
## LM test = 135.97, df = 1, p-value < 2.2e-16
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 22`. Furthermore, the Breush-Godfrey test is also conducted here using the `bgtest` function. From the test we can observe that residuals are not randomly distributed, and the ACF plot we can conclude that serial correlation left in residuals are highly significant.

```
VIF.model1.2<-vif(model1.2$model)
VIF.model1.2
```

```
##      x1.t      x1.1      x1.2      x1.3      x1.4      x1.5      x1.6      x1.7
##  58.34747 126.11918 132.46580 135.37636 135.54139 138.29564 144.97845 147.77257
##      x1.8      x1.9     x1.10     x1.11     x1.12      x3.t      x3.1      x3.2
## 145.92043 150.61502 152.89970 151.69899  65.62082  18.38101  50.42609  52.36998
##      x3.3      x3.4      x3.5      x3.6      x3.7      x3.8      x3.9     x3.10
##  52.36940  53.81478  57.34063  59.91671  62.44331  65.51811  68.35053  69.10579
##     x3.11     x3.12
##  68.56418  28.14387
```

```
VIF.model1.2>10
```

```
##  x1.t  x1.1  x1.2  x1.3  x1.4  x1.5  x1.6  x1.7  x1.8  x1.9 x1.10 x1.11 x1.12
##  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  x3.t  x3.1  x3.2  x3.3  x3.4  x3.5  x3.6  x3.7  x3.8  x3.9 x3.10 x3.11 x3.12
##  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

In the output shown above, which is the test for multicollinearity by using the variance inflation factors (VIF) using the `vif` function, we can observe that the estimates of the finite DLM coefficients are suffering from the multicollinearity. The reason for this is because all the data in the VIF are more than 10, hence the effect of multicollinearity is high.

Therefore, it will be not recommended to use the Finite DLM as a model for further analysis.

6.2 Polynomial Distribute Lag Model

Polynomial DLM imposes a polynomial shape on the lag distribution to reduce the effect of multicollinearity. For the Polynomial DLM, we will be using the predictor copper price as our independent variable, and ASX price as our dependent variable. Copper price is chosen in the case as the correlation is the highest with ASX price as seen previously.

```
model2.1<-polyDlm(x=as.vector(asx.dlm$x3),y=as.vector(asx.dlm$y),q=12,k=1,show.beta=TRUE)
```

```
## Estimates and t-tests for beta coefficients:
##          Estimate Std. Error t value  P(>|t|)
## beta.0   0.056200    0.01180   4.750 5.10e-06
## beta.1   0.048300    0.01000   4.810 3.91e-06
## beta.2   0.040400    0.00826   4.890 2.81e-06
## beta.3   0.032500    0.00652   4.980 1.90e-06
## beta.4   0.024600    0.00487   5.050 1.40e-06
## beta.5   0.016600    0.00341   4.880 2.85e-06
```

```
## beta.6   0.008710    0.00252   3.450 7.34e-04
## beta.7   0.000794    0.00282   0.282 7.79e-01
## beta.8  -0.007130    0.00405  -1.760 8.06e-02
## beta.9  -0.015000    0.00563  -2.670 8.39e-03
## beta.10 -0.023000    0.00733  -3.130 2.12e-03
## beta.11 -0.030900    0.00909  -3.400 8.91e-04
## beta.12 -0.038800    0.01090  -3.570 4.99e-04
```

```
summary(model2.1)
```

```
##
## Call:
## "Y ~ (Intercept) + X.t"
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1246.73  -594.20    91.26   546.36  1614.17
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.219e+03  2.127e+02  19.838  < 2e-16 ***
## z.t0         5.624e-02  1.184e-02   4.748 4.85e-06 ***
## z.t1        -7.921e-03  1.848e-03  -4.285 3.29e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 716.1 on 146 degrees of freedom
## Multiple R-squared:  0.1486, Adjusted R-squared:  0.137
## F-statistic: 12.74 on 2 and 146 DF,  p-value: 7.915e-06
```

In the `model2.1` output above, we have used the `polyDLM` function to perform the DLM. Similarly in the we used a `q=12` similar to Finite DLM and `k=1`. Besides that, an argument of `show.beta=TRUE` is used to generate estimates of lag weights and their standard deviations using standard regression.

In the summary for `model2.1`, the lag weights of the predictors are significant at the 5% statistical level as p-value < 0.05. Furthermore, the adjusted R-squared shows a value of 0.1486, which is considerably low.

```
checkresiduals(model2.1$model$residuals)
```

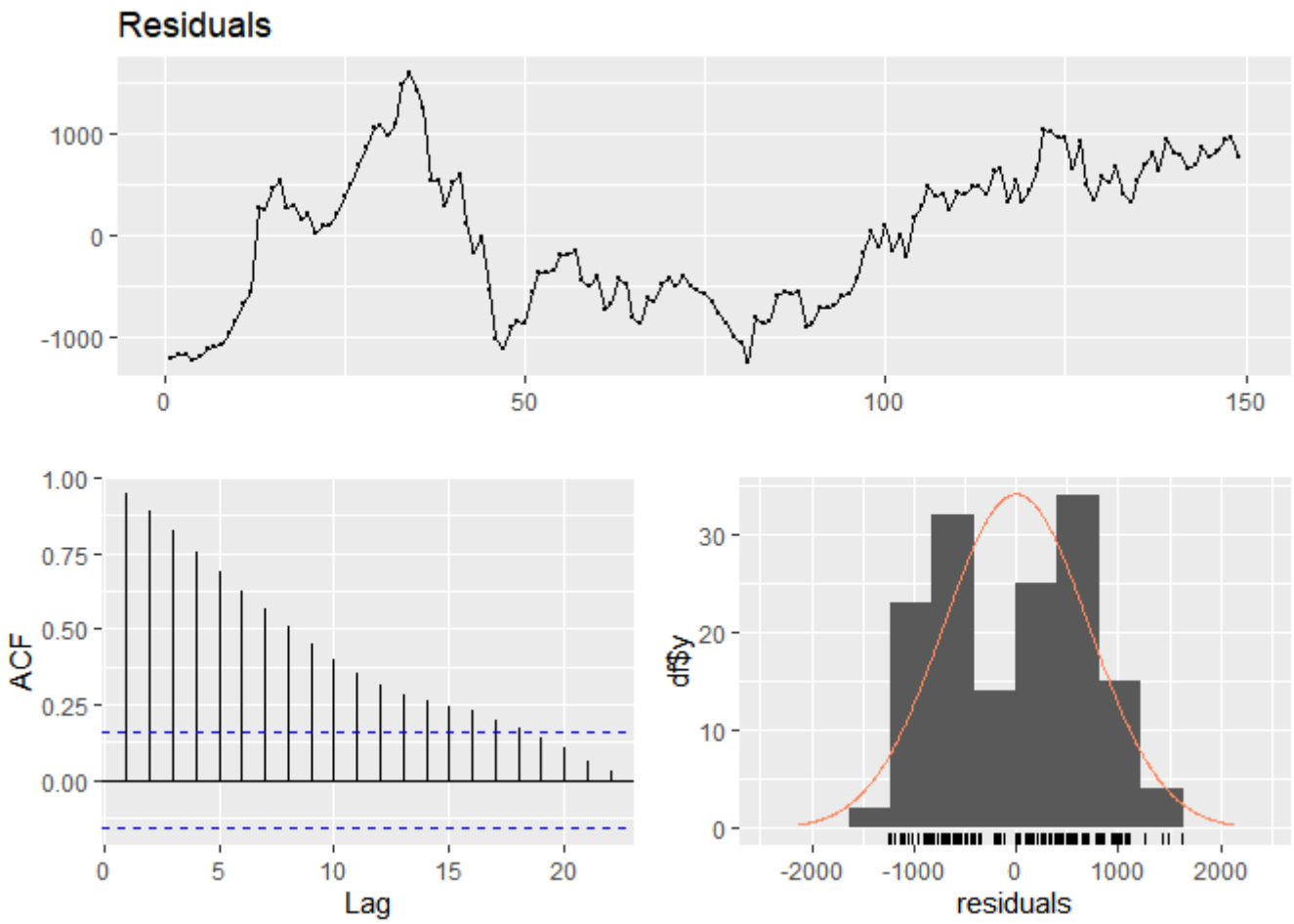

Figure 23

```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 738.3, df = 10, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 10
```

```
bgtest(model2.1$model)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model2.1$model
## LM test = 134.56, df = 1, p-value < 2.2e-16
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 23`. Furthermore, the Breush-Godfrey test is also conducted here using the `bgtest` function. From the test we can observe that residuals are not randomly distributed, and the ACF plot we can conclude that serial correlation left in residuals are highly significant.

```
VIF.model2.1<-vif(model2.1$model)
VIF.model2.1
```

```
##     z.t0     z.t1
## 22.91477 22.91477
```

```
VIF.model2.1>10
```

```
## z.t0 z.t1
## TRUE TRUE
```

In the output shown above, which is the test for multicollinearity by using the variance inflation factors (VIF) using the `vif` function, we can observe that the estimates of the finite DLM coefficients are suffering from the multicollinearity. The reason for this is because all the data in the VIF are more than 10, hence the effect of multicollinearity is high.

Therefore, it will be not recommended to use the Polynomial DLM as a model for further analysis.

6.3 Koyck Distributed Lag Model

In the Koyck DLM similar to Polynomial DLM, we have used the independent predictor copper price as it is the highest correlated with ASX price. The Koyck DLM transformation is useful when dealing with infinite DLM.

```
model3.1<-koyckDlm(x=as.vector(asx.dlm$x3),y=as.vector(asx.dlm$y))
summary(model3.1,diagnostics=TRUE)
```

```
##
## Call:
## "Y ~ (Intercept) + Y.1 + X.t"
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -689.64 -108.62   12.78  140.20  771.79
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 189.368812  87.644648   2.161   0.0322 *
## Y.1           0.971621   0.021895  44.376   <2e-16 ***
## X.t          -0.005864   0.009517  -0.616   0.5387
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 201.9 on 157 degrees of freedom
```

```
## Multiple R-Squared: 0.9485,  Adjusted R-squared: 0.9479
## Wald test:  1448 on 2 and 157 DF,  p-value: < 2.2e-16
##
## Diagnostic tests:
##                  df1 df2  statistic      p-value
## Weak instruments   1 157 1966.86799 1.043205e-90
## Wu-Hausman         1 156   10.97528 1.147725e-03
##
##                             alpha         beta       phi
## Geometric coefficients:  6672.885 -0.005863623 0.9716211
```

In the `model3.1` output above, we have used the function `koyckDLM` to perform the Koyck DLM Transformation. Since the Koyck DLM approach does not produce a standard error, we have added an argument `diagnostics=TRUE` as it will show the F test, Wu-Hausman test, and the Sargan Test.

From the summary output, we can see that in the Weak Instruments line we can conclude that the model at the first stage of the least-squares fitting is significant at 5% level of significance.

Besides that, we can also observe that in the Wu-Hausman test, there is a significant correlation between the explanatory variable and the error term at 5% level as p-value < 0.05. Furthermore, the adjusted R-squared shows a value of 0.9479, which is considerably high.

```
checkresiduals(model3.1$model$residuals)
```

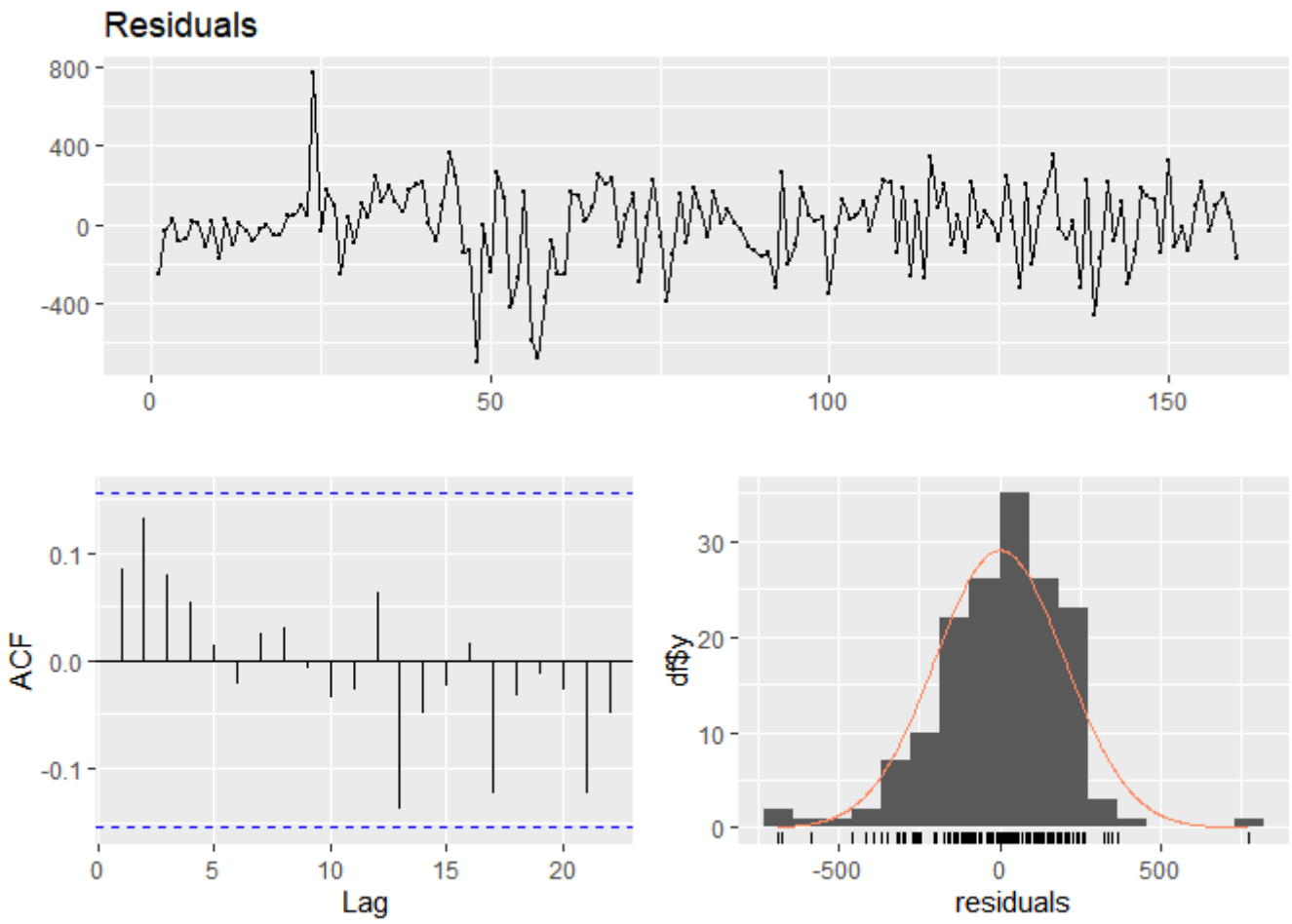

Figure 24

```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 6.2327, df = 10, p-value = 0.7953
##
## Model df: 0.   Total lags used: 10
```

```
bgtest(model3.1$model)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model3.1$model
## LM test = 1.5021, df = 1, p-value = 0.2203
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 24`. Furthermore, the Breush-Godfrey test is also conducted here using the `bgtest` function. From the test we can observe that residuals are randomly distributed, and the ACF plot we can conclude that there are no serial correlation in residuals.

```
VIF.model3.1<-vif(model3.1$model)
VIF.model3.1
```

```
##      Y.1      X.t
## 1.493966 1.493966
```

```
VIF.model3.1>10
```

```
##   Y.1   X.t
## FALSE FALSE
```

In the output shown above, which is the test for multicollinearity by using the variance inflation factors (VIF) using the `vif` function, we can observe that the estimates of the finite DLM coefficients are not suffering from the multicollinearity. The reason for this is because all the data in the VIF are not more than 10, suggesting that there is no multicollinearity.

Based on the diagnosis, there is a possibility we can investigate using this model further.

6.4 Autoregressive Distributed Lag Model

Autoregressive DLM is normally used when a suitable solution with other DLMs cannot be found for infinite DLM. Similarly to Finite DLM, we will be using the predictors gold price and copper price as the independent variable, and the ASX price will be the dependent variable. To perform this test, we will first create a loop function for p and q and use the `ardlDLM` function to identify most appropriate number of lags for the DLM.

```
for (i in 1:6){
  for (j in 1:6){
    model4.1<-ardlDlm(formula = y ~ x1 + x3, data=asx.dlm, p=i, q=j)
    cat("p =",i,"q =",j,"AIC =",AIC(model4.1$model),"BIC =", BIC(model4.1$model),
        "\n")
  }
}
```

```
## p = 1 q = 1 AIC = 2135.427 BIC = 2156.954
## p = 1 q = 2 AIC = 2123.2 BIC = 2147.752
## p = 1 q = 3 AIC = 2108.165 BIC = 2135.728
## p = 1 q = 4 AIC = 2097.165 BIC = 2127.728
## p = 1 q = 5 AIC = 2086.767 BIC = 2120.315
## p = 1 q = 6 AIC = 2075.998 BIC = 2112.519
## p = 2 q = 1 AIC = 2119.917 BIC = 2147.537
## p = 2 q = 2 AIC = 2121.523 BIC = 2152.212
## p = 2 q = 3 AIC = 2107.82 BIC = 2141.508
## p = 2 q = 4 AIC = 2096.569 BIC = 2133.244
## p = 2 q = 5 AIC = 2086.252 BIC = 2125.9
## p = 2 q = 6 AIC = 2075.537 BIC = 2118.145
## p = 3 q = 1 AIC = 2109.717 BIC = 2143.406
## p = 3 q = 2 AIC = 2110.906 BIC = 2147.657
## p = 3 q = 3 AIC = 2109.982 BIC = 2149.795
## p = 3 q = 4 AIC = 2098.603 BIC = 2141.39
## p = 3 q = 5 AIC = 2088.342 BIC = 2134.09
## p = 3 q = 6 AIC = 2077.726 BIC = 2126.42
## p = 4 q = 1 AIC = 2100.046 BIC = 2139.777
## p = 4 q = 2 AIC = 2101.141 BIC = 2143.929
## p = 4 q = 3 AIC = 2100.586 BIC = 2146.43
## p = 4 q = 4 AIC = 2102.043 BIC = 2150.943
## p = 4 q = 5 AIC = 2091.733 BIC = 2143.58
## p = 4 q = 6 AIC = 2081.005 BIC = 2135.786
## p = 5 q = 1 AIC = 2089.626 BIC = 2135.374
## p = 5 q = 2 AIC = 2090.472 BIC = 2139.27
## p = 5 q = 3 AIC = 2089.878 BIC = 2141.726
## p = 5 q = 4 AIC = 2091.428 BIC = 2146.326
## p = 5 q = 5 AIC = 2092.86 BIC = 2150.807
## p = 5 q = 6 AIC = 2082.041 BIC = 2142.909
## p = 6 q = 1 AIC = 2079.751 BIC = 2131.489
## p = 6 q = 2 AIC = 2080.725 BIC = 2135.506
## p = 6 q = 3 AIC = 2079.933 BIC = 2137.758
## p = 6 q = 4 AIC = 2081.577 BIC = 2142.445
```

```
## p = 6 q = 5 AIC = 2082.692 BIC = 2146.604
## p = 6 q = 6 AIC = 2084.692 BIC = 2151.647
```

From the loop function output above, we can see that p = 2, q = 6 contains the lowest AIC value, and p = 1, q= 6 contains the lowest BIC value. Hence, we will be using these 2 values for the Autoregressive DLM.

```
model4.2<-ardlDlm(formula = y ~ x1 + x3, data=asx.dlm,p=2,q=6)
summary(model4.2)
```

```
##
## Time series regression with "ts" data:
## Start = 7, End = 161
##
## Call:
## dynlm(formula = as.formula(model.text), data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -585.87 -112.09   -0.89   95.88  701.09
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 208.80506   93.93781   2.223 0.027810 *
## x1.t         -1.15884    0.29887  -3.877 0.000161 ***
## x1.1          1.28827    0.44156   2.918 0.004103 **
## x1.2         -0.10847    0.31160  -0.348 0.728271
## x3.t          0.07485    0.03378   2.216 0.028299 *
## x3.1         -0.01204    0.05365  -0.225 0.822687
## x3.2         -0.06866    0.03480  -1.973 0.050443 .
## y.1           0.90725    0.08374  10.834  < 2e-16 ***
## y.2           0.18891    0.11208   1.686 0.094075 .
## y.3          -0.06174    0.11378  -0.543 0.588248
## y.4          -0.07614    0.10978  -0.694 0.489075
## y.5           0.01954    0.11032   0.177 0.859677
## y.6          -0.01470    0.07736  -0.190 0.849605
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 186.8 on 142 degrees of freedom
## Multiple R-squared:  0.953,  Adjusted R-squared:  0.9491
## F-statistic: 240.2 on 12 and 142 DF,  p-value: < 2.2e-16
```

From the above output for `model4.2`, we can see that the lag weights are significant at the 5% statistical level as p-value < 0.05. Furthermore, the adjusted R-squared shows a value of 0.9491, which is considerably high.

```
checkresiduals(model4.2$model$residuals)
```

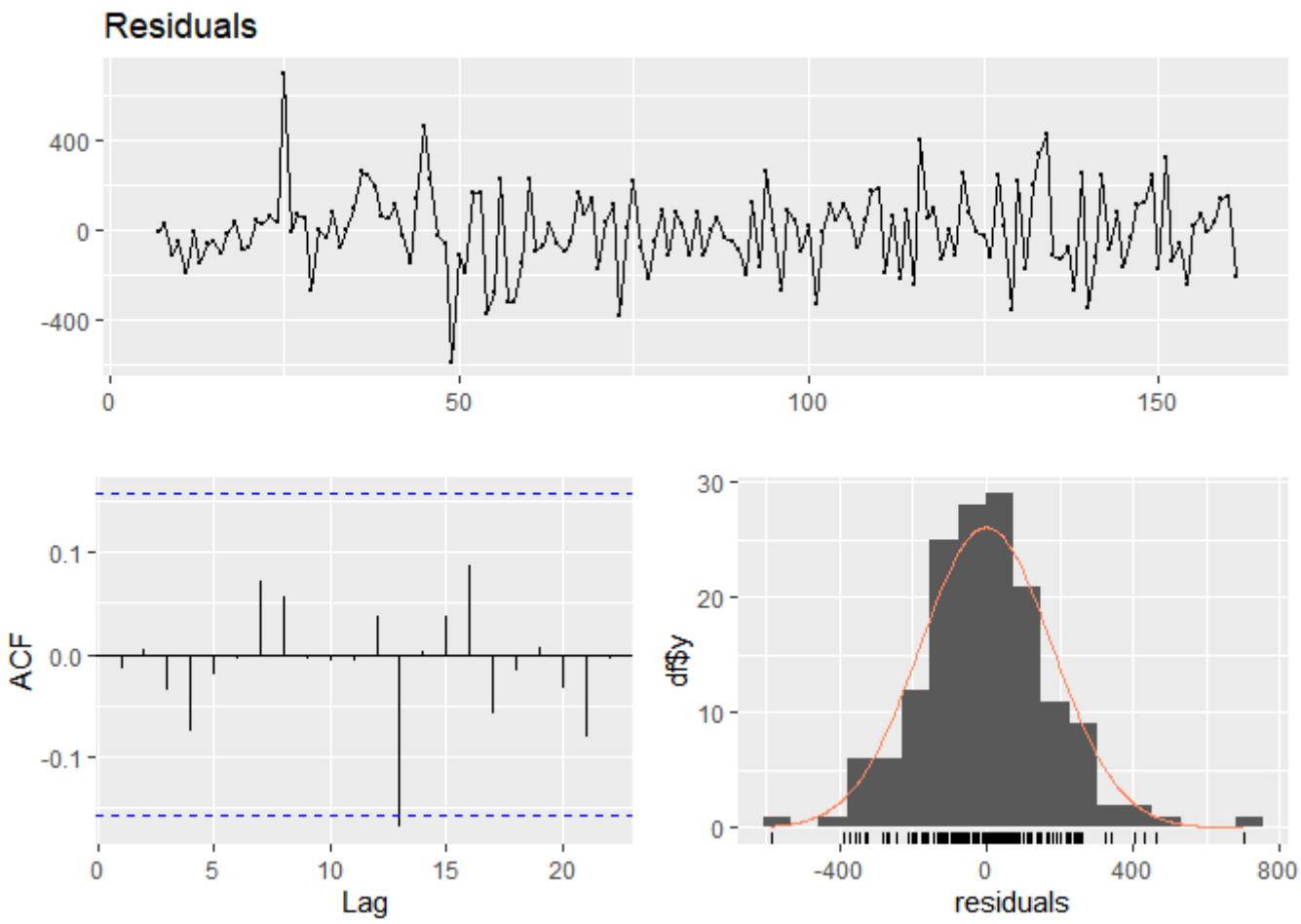

Figure 25

```
##
##  Ljung-Box test
##
## data:  Residuals
## Q* = 2.6072, df = 10, p-value = 0.9892
##
## Model df: 0.   Total lags used: 10
```

```
bgtest(model4.2$model)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model4.2$model
## LM test = 1.1378, df = 1, p-value = 0.2861
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 25`. Furthermore, the Breush-Godfrey test is also conducted here

using the `bgtest` function. From the test we can observe that residuals are randomly distributed, and the ACF plot we can conclude that there are no serial correlation in residuals.

```
VIF.model4.2<-vif(model4.2$model)
VIF.model4.2
```

```
##        x1  L(x1, 1)  L(x1, 2)        x3  L(x3, 1)  L(x3, 2)   L(y, 1)   L(y, 2)
##  59.52282 131.42608  66.07815  19.94677  51.92993  22.53465  21.75838  39.82060
##   L(y, 3)   L(y, 4)   L(y, 5)   L(y, 6)
##  41.91757  40.06300  41.62030  20.90242
```

```
VIF.model4.2>10
```

```
##       x1 L(x1, 1) L(x1, 2)       x3 L(x3, 1) L(x3, 2)  L(y, 1)  L(y, 2)
##     TRUE     TRUE     TRUE     TRUE     TRUE     TRUE     TRUE     TRUE
##  L(y, 3)  L(y, 4)  L(y, 5)  L(y, 6)
##     TRUE     TRUE     TRUE     TRUE
```

In the output shown above, which is the test for multicollinearity by using the variance inflation factors (VIF) using the `vif` function, we can observe that the estimates of the finite DLM coefficients are suffering from the multicollinearity. The reason for this is because all the data in the VIF are more than 10, hence the effect of multicollinearity is high.

```
model4.3<-ardlDlm(formula = y ~ x1 + x3, data=asx.dlm,p=1,q=6)
summary(model4.3)
```

```
##
## Time series regression with "ts" data:
## Start = 7, End = 161
##
## Call:
## dynlm(formula = as.formula(model.text), data = data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -603.5 -106.8   -0.5   97.2  693.9
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 218.43528   94.51184   2.311 0.022241 *
## x1.t         -1.17994    0.30019  -3.931 0.000131 ***
## x1.1          1.18799    0.29609   4.012 9.63e-05 ***
## x3.t          0.09368    0.03234   2.896 0.004364 **
## x3.1         -0.09563    0.03246  -2.946 0.003752 **
```

```
## y.1           0.92941    0.07898  11.768  < 2e-16 ***
## y.2           0.18713    0.11240   1.665 0.098118 .
## y.3          -0.09666    0.11017  -0.877 0.381736
## y.4          -0.06681    0.11024  -0.606 0.545433
## y.5           0.03024    0.11055   0.274 0.784835
## y.6          -0.02358    0.07724  -0.305 0.760630
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 188.1 on 144 degrees of freedom
## Multiple R-squared:  0.9517, Adjusted R-squared:  0.9483
## F-statistic: 283.6 on 10 and 144 DF,  p-value: < 2.2e-16
```

From the above output for `model4.3`, we can see that the lag weights are significant at the 5% statistical level as p-value < 0.05. Furthermore, the adjusted R-squared shows a value of 0.9483, which is considerably high.

```
checkresiduals(model4.3$model$residuals)
```

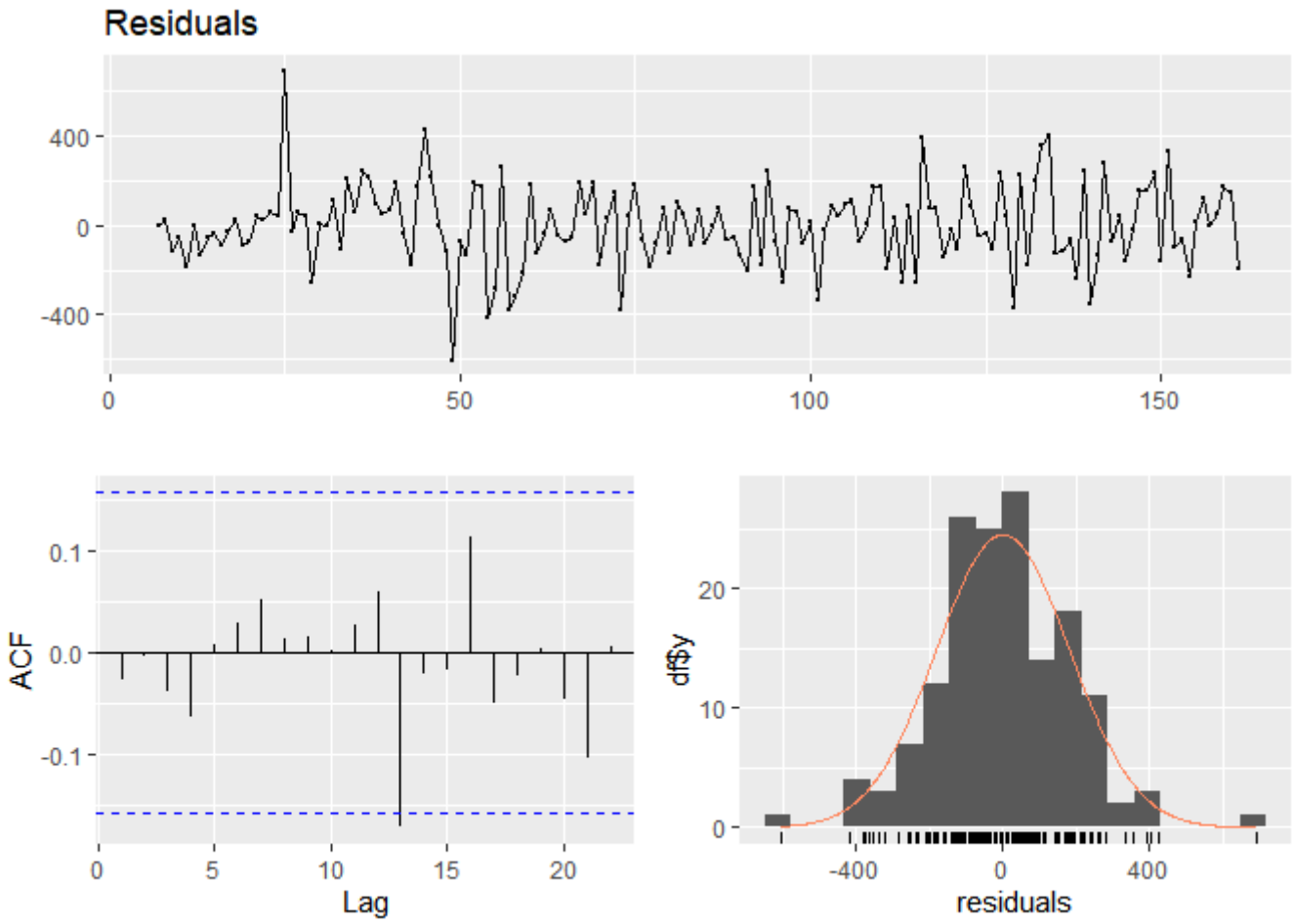

Figure 26

```
##
##  Ljung-Box test
##
## data:  Residuals
```

```
## Q* = 1.6642, df = 10, p-value = 0.9983
##
## Model df: 0.   Total lags used: 10
```

```
bgtest(model4.3$model)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  model4.3$model
## LM test = 0.75679, df = 1, p-value = 0.3843
```

To perform a diagnostic check, we will use the function `checkresiduals` to check the data for residuals as shown in `Figure 26`. Furthermore, the Breush-Godfrey test is also conducted here using the `bgtest` function. From the test we can observe that residuals are randomly distributed, and the ACF plot we can conclude that there are no serial correlation in residuals.

```
VIF.model4.3<-vif(model4.3$model)
VIF.model4.3
```

```
##       x1 L(x1, 1)       x3 L(x3, 1)  L(y, 1)  L(y, 2)  L(y, 3)  L(y, 4)
## 59.17056 58.22605 18.01778 18.72549 19.07042 39.46624 38.72065 39.80853
##  L(y, 5)  L(y, 6)
## 41.18561 20.53259
```

```
VIF.model4.3>10
```

```
##       x1 L(x1, 1)       x3 L(x3, 1)  L(y, 1)  L(y, 2)  L(y, 3)  L(y, 4)
##     TRUE     TRUE     TRUE     TRUE     TRUE     TRUE     TRUE     TRUE
##  L(y, 5)  L(y, 6)
##     TRUE     TRUE
```

In the output shown above, which is the test for multicollinearity by using the variance inflation factors (VIF) using the `vif` function, we can observe that the estimates of the finite DLM coefficients are suffering from the multicollinearity. The reason for this is because all the data in the VIF are more than 10, hence the effect of multicollinearity is high.

Based on the diagnosis, there is a possibility we can investigate using this model further, however due to high multicollinearity, it has to be investigated further.

7.0 Conclusion

In conclusion, in this report we have addressed the questions to check the existence of nonstationarity, impact of the components of the time series data, and choosing the most accurate and suitable distributed lag model. Firstly for the existence of nonstationarity, we can conclude that there is an existence of nonstationarity for all predictors in the dataset at 5% level of significance. In terms of the impact of components of the time series data, decomposition of the series using X12 and STL methods to test trend and seasonality were implemented. Lastly, for choosing the most accurate and suitable distributed lag model, we can recommend the use of the Koyck DLM as it has shown evidence of random distribution and non multicollinearity.