MATH1318 Final Project 2021: Time Series Prediction Analysis of Annual Gold Price

Anushka Sharma (s3817004) and Adity Bhargav (s3814507)

Table of Contents

- Introduction
- Data Source and Description
- Objective & Methodology
- Data Prepossessing
- Model Specification
- Parameter Estimation and Model Diagnostics
- Residual Analysis
- Forecasting of gold prices (USD) for next 10 years
- Discussion and Conclusion
- References
- Appendix

Introduction

Gold has been a symbol of prestige, power, and prosperity for a very long period of time. It provides unwavering stability as an asset that has historically kept its value in the face of political and economic turmoil. The prediction of gold prices is, therefore, one of the most popular and interesting subjects that analysts research and work on. The prediction happens for a different number of time periods like monthly, weekly,daily etc. This project will utilize an annual data of gold prices to perform time-series analysis and predict the gold prices for the next 10 years.

Data Source and Description

The dataset `annual_csv` has been sourced from: Gold Prices. (2021). Retrieved 7 June 2021, from https://datahub.io/core/gold-prices#data-cli (https://datahub.io/core/gold-prices#data-cli). The dataset includes two variables- `Date` and `Price` . The `Date` represents yearly time period from 1950-2019 and the `Price` represents gold prices for the respective year.

The time series analysis tools will be used on the dataset to find the best model which fits the dataset which will be further used to predict the gold prices for the next 10 years (2020-2029).

Objective & Methodology

The main aim is to perform a time series analysis on the annual_csv dataset to analyze the change in gold prices from 1950-2019. Initially the dataset will be prepossessed to make it clean and error-free. The gold price data will then be converted to time series to make it ready for analysis. The nature of the series will then be analysed by plotting a time-series, scatterplot and calculating correlation. Model specification will begin nwxt by first checking whether the series is stationary or not using ADF. If the series is not stationary, first differencing, second differencing and so on will be performed until the data becomes stationary. ACF,PACF,EACF and BIC will be computed for the stationary series which will yield possible ARIMA models for the gold price series.

Parameter estimation, model diagnostics and residual analysis will be performed to compute the best ARIMA model for the gold price series. Finally, the best ARIMA model will then be used to forecast and plot the prices of gold for the next 10 years (2019-2029).

Data Prepossessing

Before executing both the tasks, we need to perform some necessary data prepossessing steps to make the dataset ready for model-fitting.

1. Importing necessary libraries and dataset into Rstudio:

```
# Importing libraries

library(TSA)
library(fUnitRoots)
library(lmtest)
library(FitAR)
library(tseries)
library(readr)
library(dplyr)
library(magrittr)
library(tidyr)
library(stats)
library(forecast)
```

```
# Importing dataset `gold`

gold <- read.csv("C:/Users/bharg/Desktop/Semester 1 2021/Time Series Analysis MATH1318/annual
_csv.csv")

# Checking dataset has been loaded successfully

head(gold)
```

	Date <chr>	Price <dbl>
1	1950-12	34.72
2	1951-12	34.66
3	1952-12	34.79
4	1953-12	34.85
5	1954-12	35.04
6	1955-12	34.97

6 rows

2. Checking for missing and special values:

Checking the dataset for missing values-

```
# Checking for missing values

colSums(is.na(gold))
```

```
##  Date Price
##     0     0
```

As per the above output there are no missing values in the dataset gold, which makes it a consistent dataset. Now checking for special values-

```
#Checking for special values

is.specialorNA <- function(x){
  if (is.numeric(x)) (is.infinite(x) | is.nan(x) | is.na(x))
}
sapply(gold, function(x) sum( is.specialorNA(x) ))
```

```
##  Date Price
##     0     0
```

As per the above outputs, there are no missing and special values found which means the dataset gold is clean.

3. Summary statistics for gold prices from 1950-2019:

Generating a summary of the gold prices from 1950-2019-

```
#Calculating summary statistics for the response variable price

par(mfrow=c(1,1))
summaryPrice<- gold %>% summarise(Min= min(Price, na.rm = TRUE),
                                  Median= median(Price, na.rm = TRUE),
                                  Max= max(Price, na.rm = TRUE),
                                  Mean= mean(Price, na.rm = TRUE),
                                  N= n())
summaryPrice
```

Min	Median	Max	Mean	N
<dbl>	<dbl>	<dbl>	<dbl>	<int>
34.66	320.8035	1687.342	412.7765	70

1 row

From the above summary generated for the gold prices, we get the following information-

- The minimum price for gold was $34.66 and the maximum price reached $1687.342 in the time span of 1950-2019.
- The average gold price from the year 1950-2019 was $412.7765.

Converting dataset to time-series:

The dataset will be converted to time-series before we begin the time-series analysis for the dataset.

```
#Deleting the first column
gold<- gold[,-1]

#Converting dataset `gold` to time-series
gold <- ts(as.vector(gold), start= 1950, frequency = 1)

#Checking conversion has been done successfully
class(gold)
```

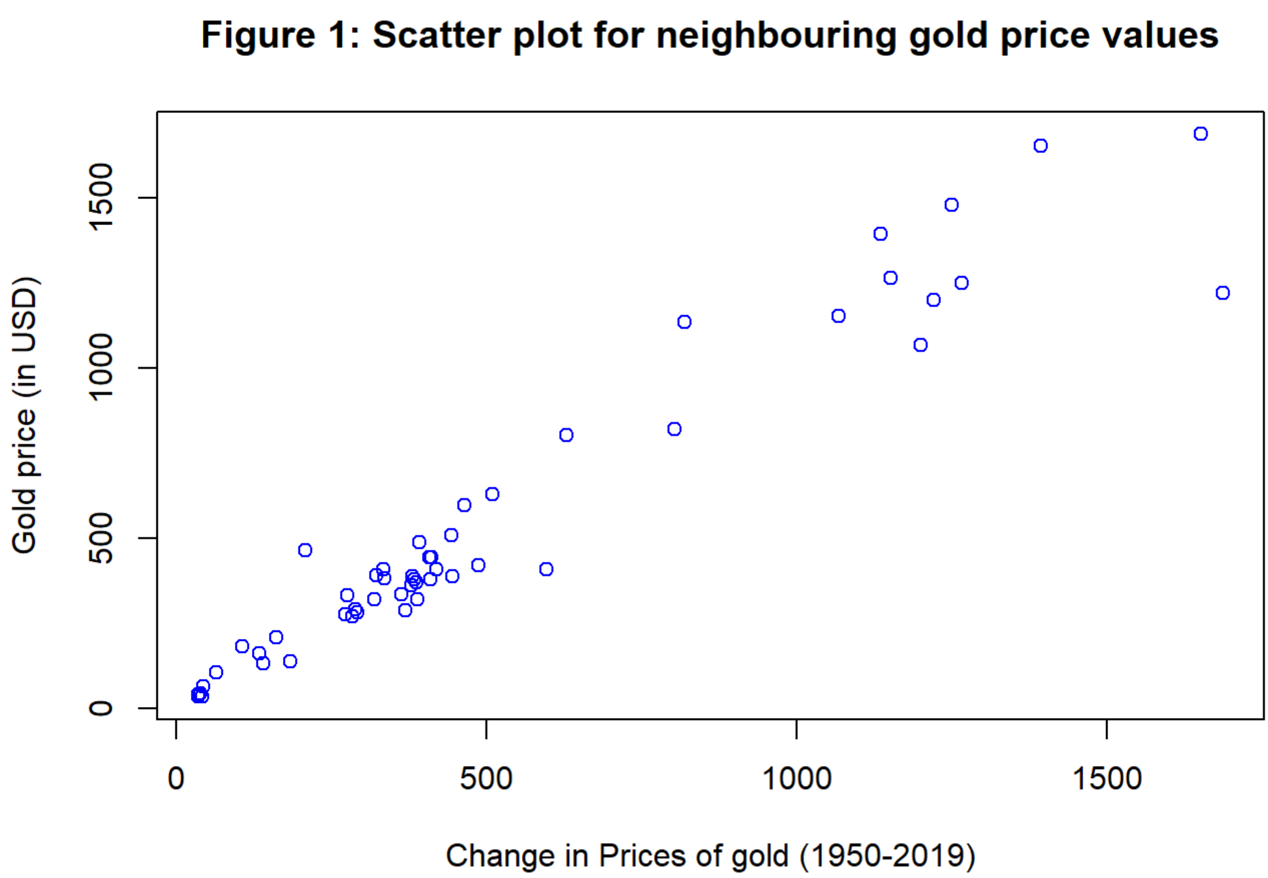
```
## [1] "ts"
```

From the above output, it is seen that the gold price dataset has been successfully converted to a time-series data.

In order to understand the nature of gold price data the impact of previous years price on the next years will be checked by finding correlation between the two.

A scatterplot of the gold price and its first lag -

```
plot(y=gold,x=zlag(gold),ylab='Gold price (in USD)', xlab='Change in Prices of gold (1950-201
9)', col='blue', main = "Figure 1: Scatter plot for neighbouring gold price values")
```



From the above scatterplot it is observed that a very high positive correlation exist between the succeeding gold prices in consecutive years. The upward trend indicates that low prices of gold are followed by low prices and high prices are followed by high gold prices. Let’s find if there exist a moderate correlation mathematically by calculating the correlation of neighbouring points.

Finding Correlation between neighbouring points

```
# Finding Correlation

y = gold
x = zlag(gold)
i = 2:length(x)
correlation = cor(y[i], x[i])

# Rounding-off correlation to get an appropriate value

round(correlation,2)
```

```
## [1] 0.97
```

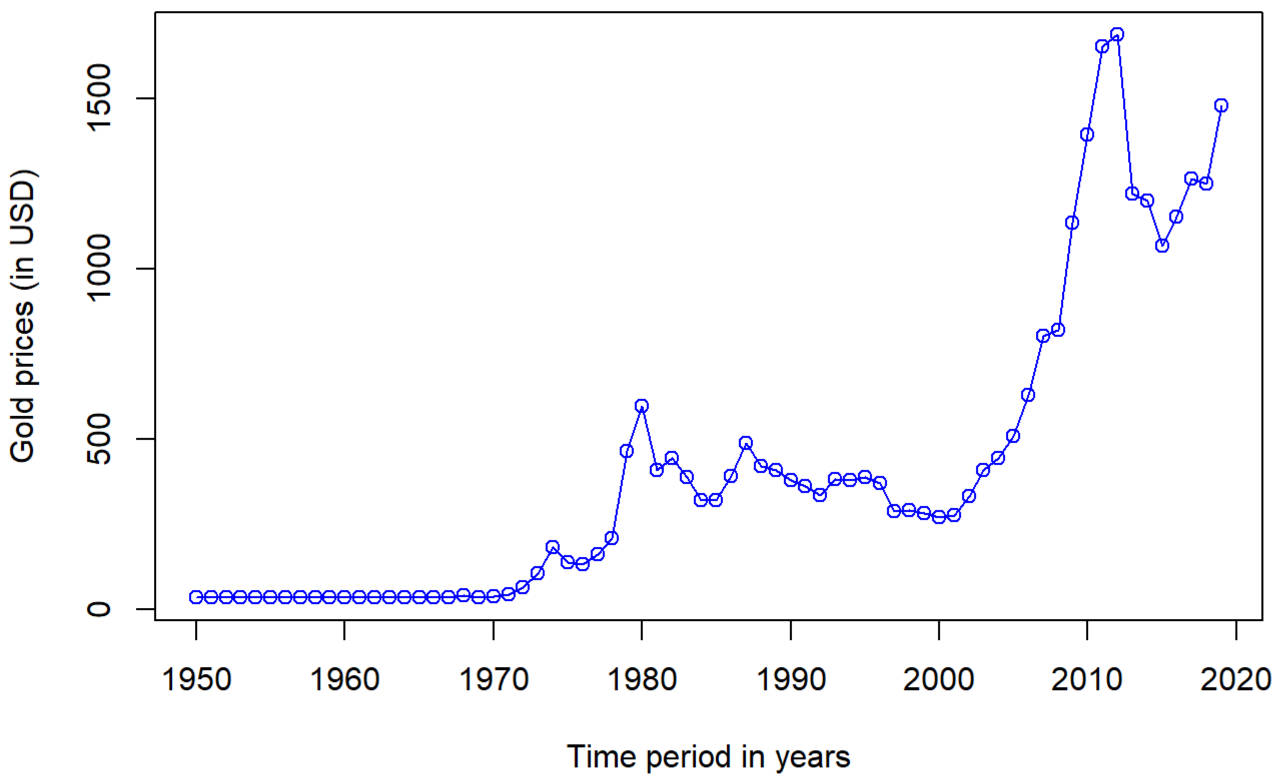
From the correlation calculated (0.97), it can be said that there is a very high correlation between the succeeding prices of gold in consecutive years.`

Time-series plot of Gold price series:

A time-series plot of the gold price data set was plotted using plot function.

```
plot(gold, ylab='Gold prices (in USD)', xlab='Time period in years', type='o', col='blue', ma
in = 'Figure 2: Time-series plot of change in prices of gold from 1950-2019')
```

Figure 2: Time-series plot of change in prices of gold from 1950-2019



Following observations were seen from the time series plot of the gold price dataset-

1.) Trend : An obvious upward (positive) trend can be observed in the Gold price series.

2.) Seasonality : No seasonality can be observed. *This rules out the possibility of fitting a cyclic, cosine or any other seasonal model.*

3.) Changing variance : A slight change in variance can be observed.

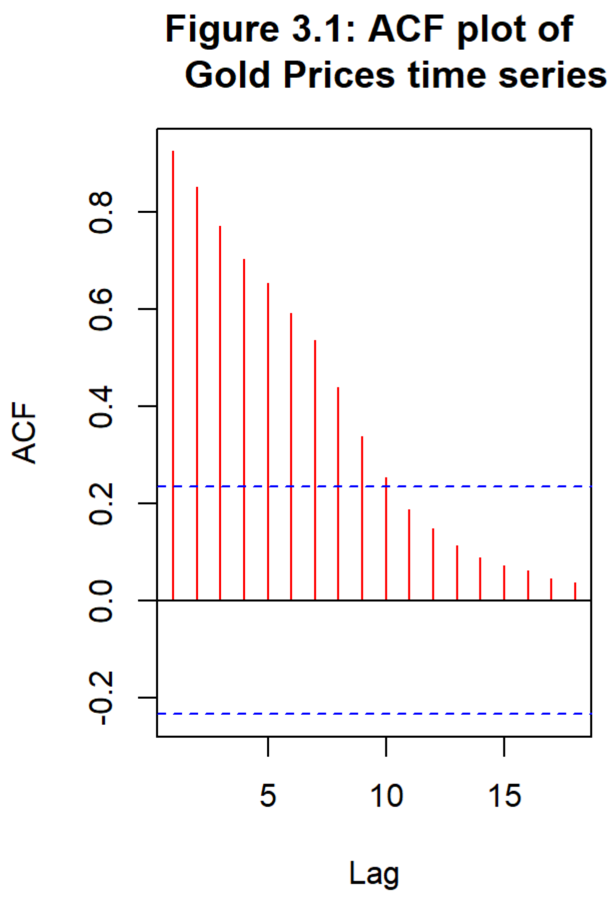
4.) Intervention : Visible intervention point around the year 2000 can be observed in the time series plot.

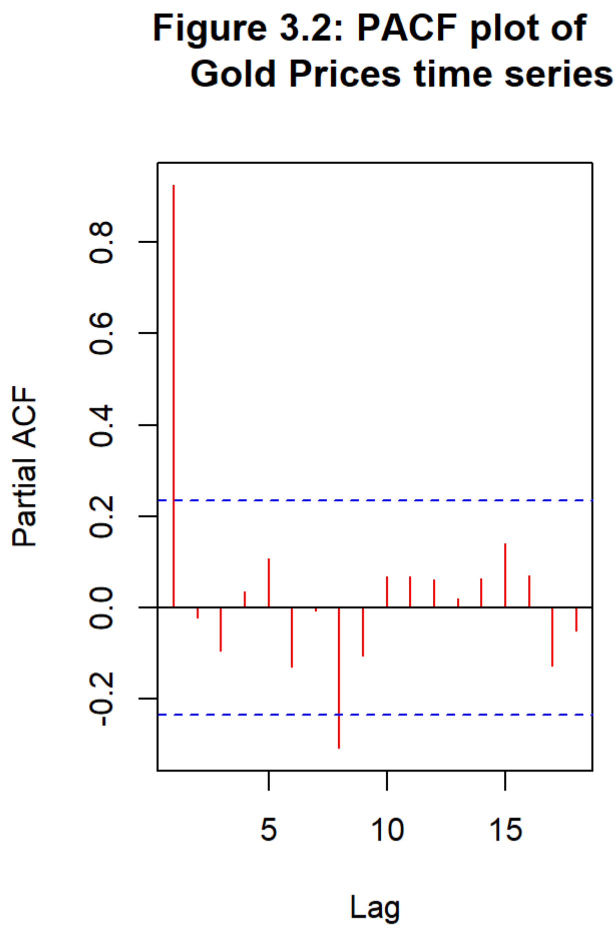
5.) Behavior : Several successive points can be observed in the series along with fluctuations along the mean level. Therefore, an AR as well as MA behaviour can be observed.

Sample Autocorrelation Fuctions

ACF and PACF:

```
# Generating ACF and PACF plot for the data set gold


par(mfrow = c(1,2))
acf(gold, main = "Figure 3.1: ACF plot of
    Gold Prices time series", col='red')
pacf(gold, main = "Figure 3.2: PACF plot of
     Gold Prices time series" , col='red')
```






```
par(mfrow = c(1,1))
```

The presence of trend and nonstationarity may be seen in the ACF plot, which shows a slowly declining pattern in the lags and a very high initial correlation in PACF. As a result, we run the Dickey-Fuller test to see if the gold price series is non-stationary.

Dicker-Fuller Unit-Root test (ADF) for gold price series:

The statistical hypotheses for Dickey-Fuller test is as follows:

```
H0 : The process is difference non-stationary
Ha : The process is stationary
```

```
#Performing ADF test on gold price dataset

adf.test(gold, alternative = "stationary")
```

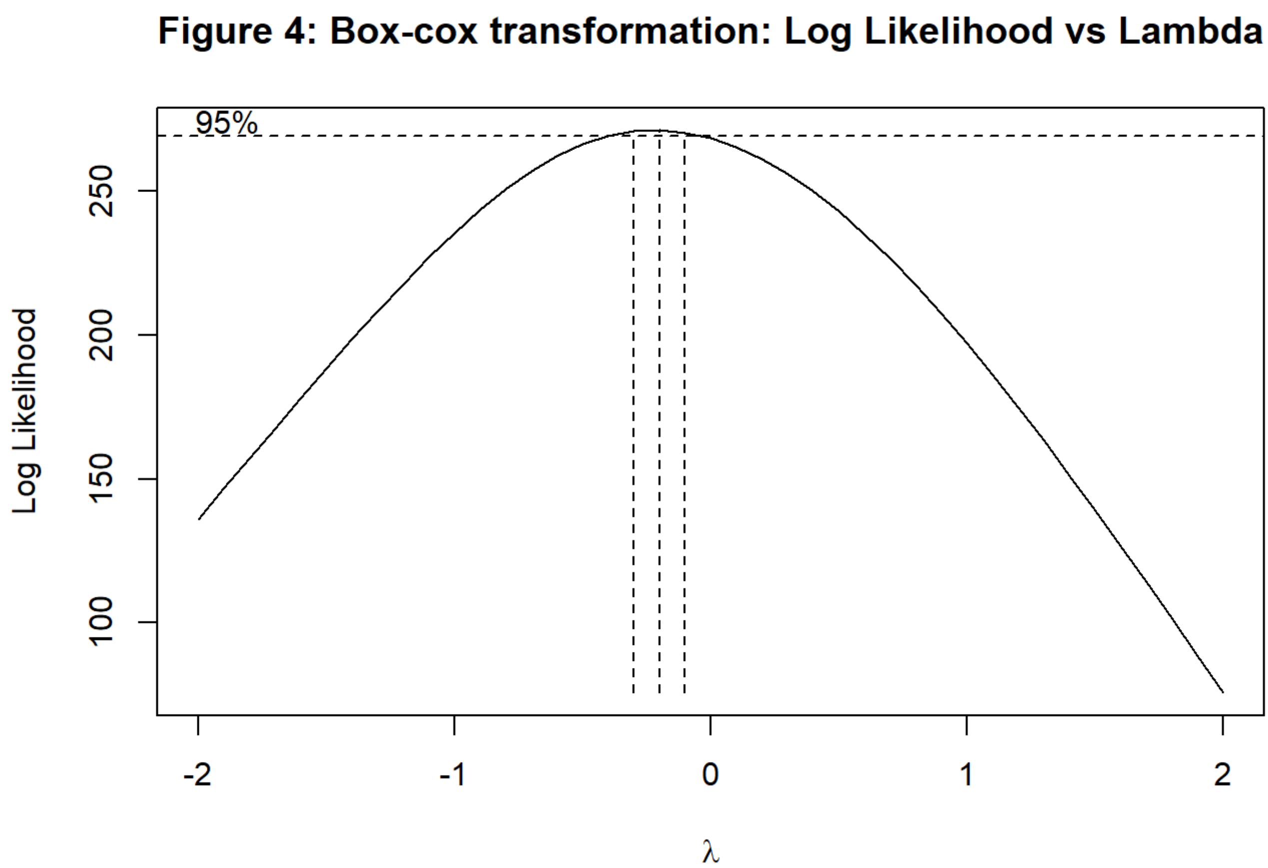
```
##
##  Augmented Dickey-Fuller Test
##
## data:  gold
## Dickey-Fuller = -1.3704, Lag order = 4, p-value = 0.8313
## alternative hypothesis: stationary
```

For a 5% significance level, the p-value from the adf.test() is greater than 0.05, which indicates that we fail to reject the null hypothesis and assume that the series is non-stationary. Therefore, the observed trend in the series along with the results of Dickey-FUller test can confirm non-stationarity in the gold price series.

Box-Cox Transformation:

As our dataset gold showed signs of changing variance, we'll use the box-cox transformation to reduce variance.

```
#Applying Box-cox tranformation

BC = BoxCox.ar(gold)
title(main="Figure 4: Box-cox transformation: Log Likelihood vs Lambda")
```



From the above plot generated, the optimum value of lambda should be something between 0 and -0.3. To get a better idea of the interval, we find out the confidence interval values for lambda.

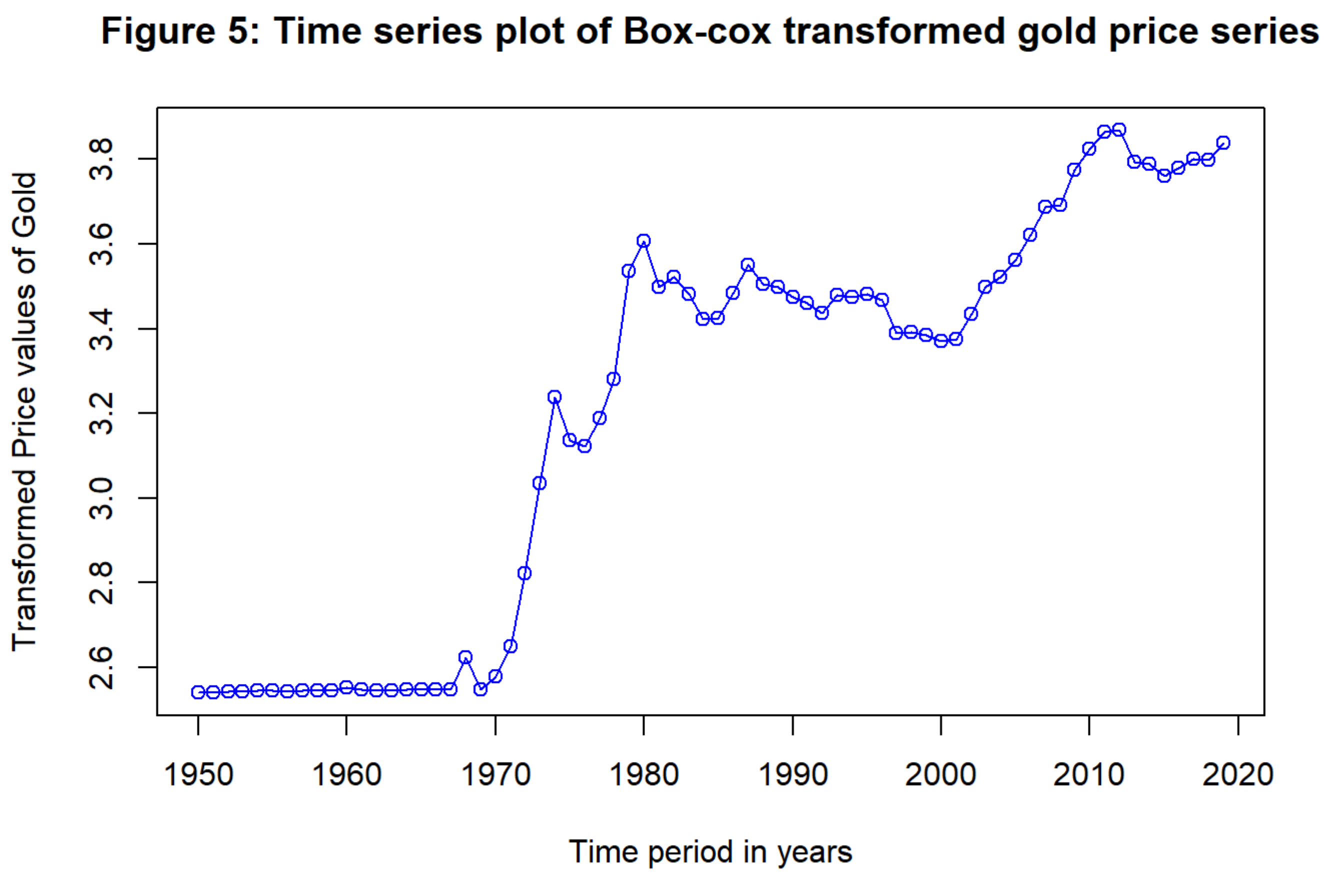
```
# Finding Confidence interval for lambda values

 BC$ci
```

```
## [1] -0.3 -0.1
```

We will use the mid-point of the Confidence interval as our lambda value. Calculating the mid-point and generating the box-cox transformed data.

```
#Generating Box-cox transformed data

lambda = BC$lambda[which(max(BC$loglike) == BC$loglike)]
BC.gold = (gold^lambda-1)/lambda
```

```
plot(BC.gold,type='o',main='Figure 5: Time series plot of Box-cox transformed gold price seri
es', ylab = "Transformed Price values of Gold",col='blue',xlab="Time period in years")
```

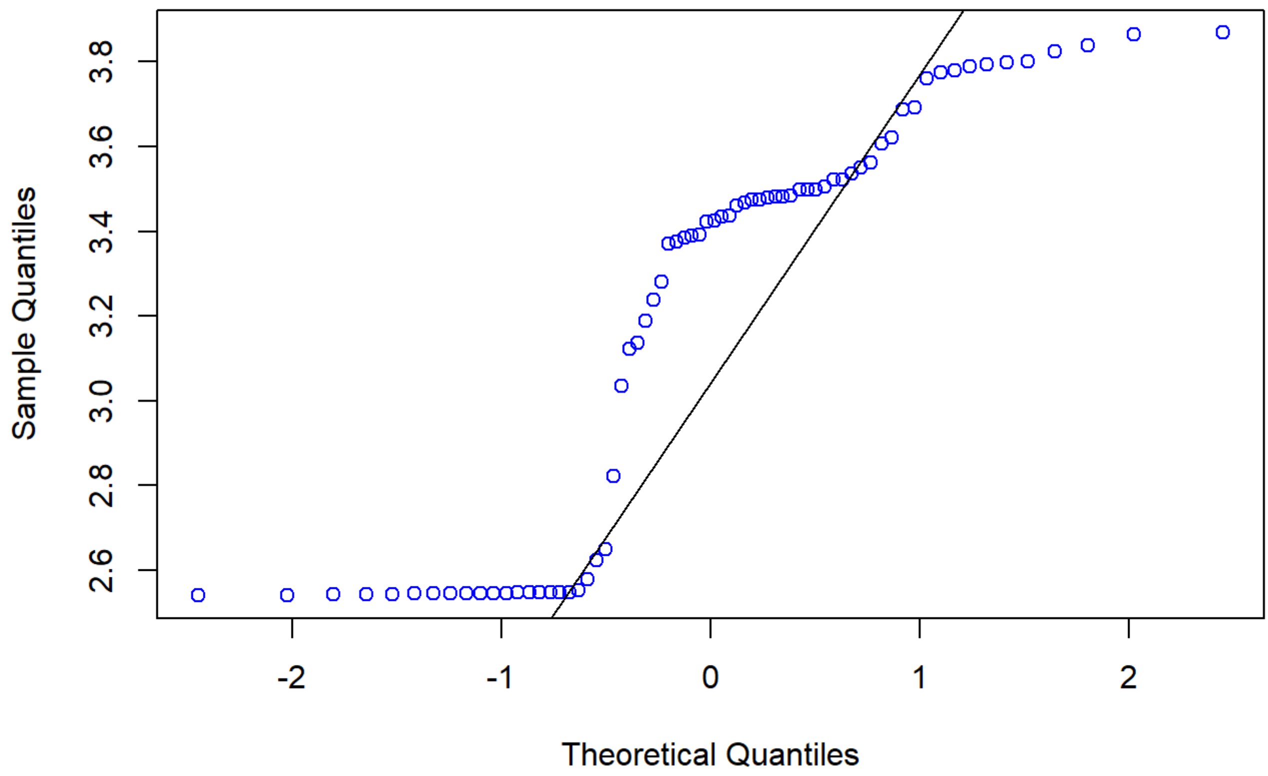


There is a slightly reduced variance that can be seen from the above plot. Thus, Box-Cox transformation was a little helpful in reducing the variance in the series. Although, Box-Cox transformation can also be helpful in improving the normality of observations.

Checking for normality by plotting Q-Q plot for the gold price series-

```
# Plotting Q-Q plot for box-cox transformed data

qqnorm(BC.gold, col='blue',main = 'Figure 6: Q-Q plot of Box-Cox transformed gold price serie
s')
qqline(BC.gold)
```

Figure 6: Q-Q plot of Box-Cox transformed gold price series



In the QQ plot, the distribution is far from the normality, as the points do not seem to follow the quantile line at all. For verification, performing the Shapiro-Wilk test-

```
#Performing normality test

shapiro.test(BC.gold)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  BC.gold
## W = 0.83173, p-value = 1.851e-07
```

The p-value of Shapiro test is way less than 0.05 level of significance, which matches the results obtained from the Q-Q plot. Therefore, we have enough evidence to reject the normality hypothesis. In conclusion, the Box-Cox transformation did not significantly help to improve normality of the observations. But, due to the effect of transformation on variance, the transformed data will be used for further analysis during the course of this project.

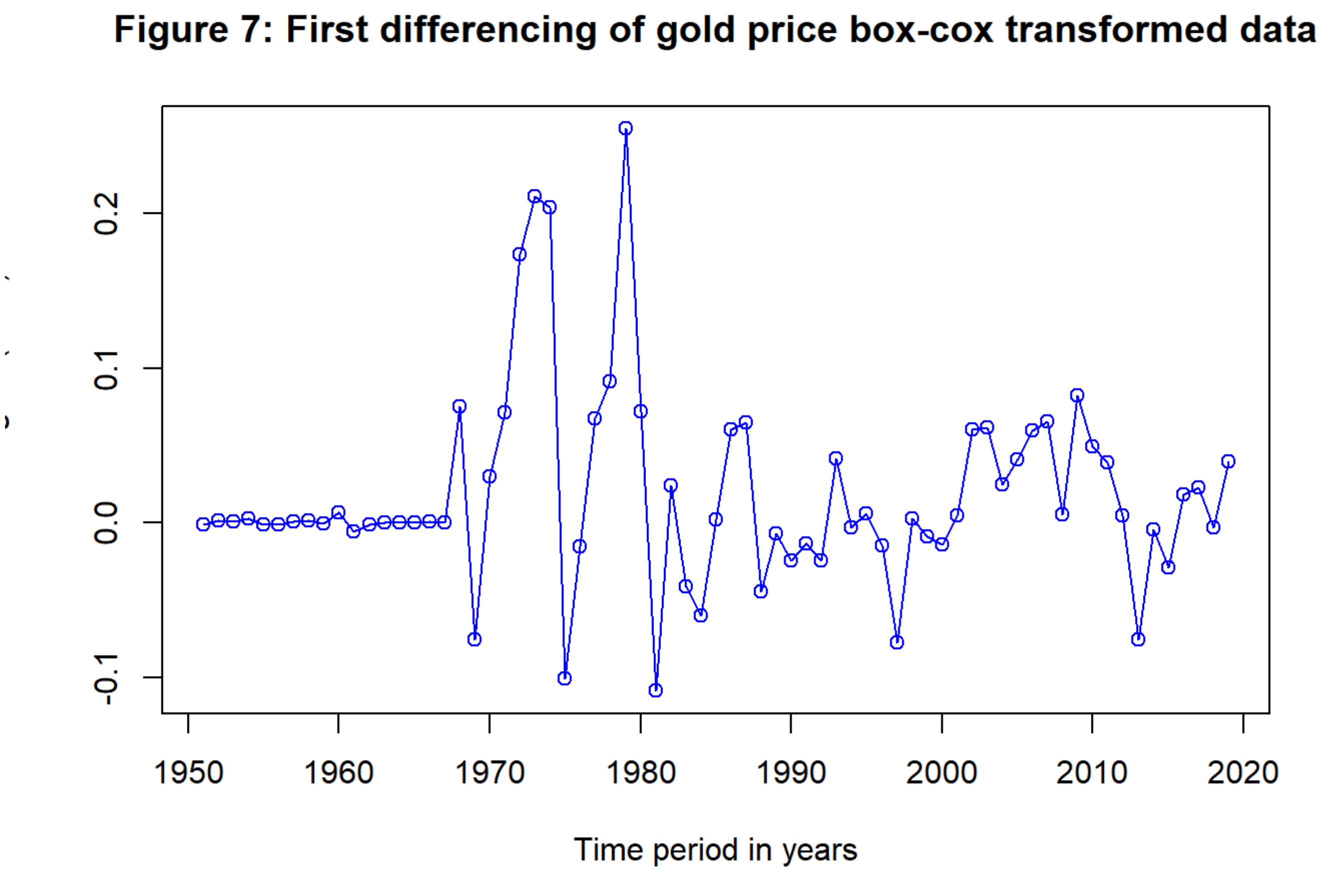
Stationarity through First differencing:

As the gold price series is non-stationary, first differencing will be performed to make it stationary.

```
# Performing first differencing

gold_fd = diff(BC.gold)
```

Plotting First differencing:

```
# Plotting first differencing of gold series data

plot(gold_fd,type='o', ylab='Price of gold (USD)
     ', xlab='Time period in years',col='blue', main='Figure 7: First differencing of gold pr
ice box-cox transformed data')
```



The above plot shows a slight trend in the series even after first differencing. To verify, ADF test was performed.

Dicker-Fuller Unit-Root test (ADF) for first differencing:

```
# Performing ADF on first differencing gold price data

adf.test(gold_fd)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  gold_fd
## Dickey-Fuller = -2.8898, Lag order = 4, p-value = 0.2136
## alternative hypothesis: stationary
```

The p-value is greater than 5% significance level. Hence, we cannot reject the null hypothesis. So, we can say that first differencing did not help in transforming the data to stationary.

Stationarity through Second differencing:

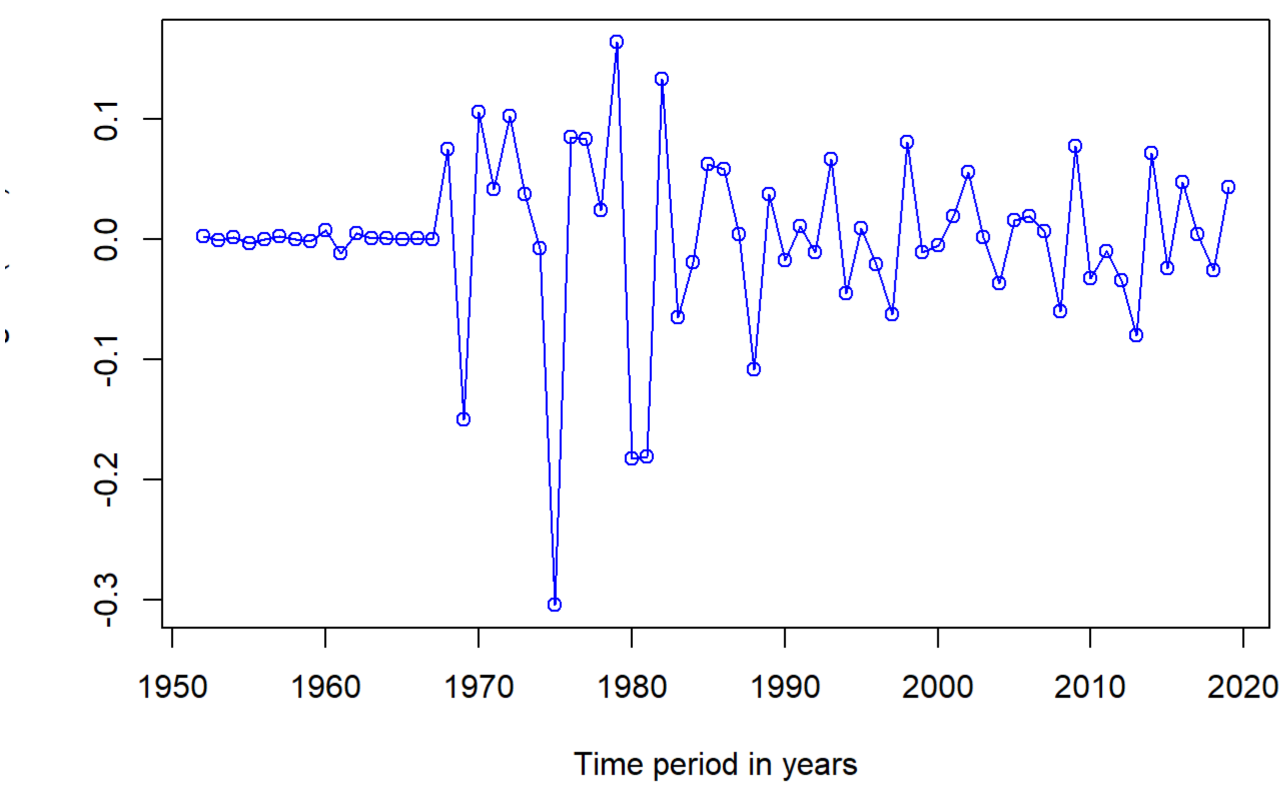
As first differencing failed to achieve stationary, second differencing is performed on the box-cox transformed gold price data.

```
#Performing Second differencing on gold price data

gold_sd <- diff(BC.gold, differences = 2)
```

Plotting Second differencing:

```
# Plotting second differencing of gold series data

plot(gold_sd,type='o', ylab='Price of gold (USD)
     ', xlab='Time period in years',col='blue', main='Figure 8: Second differencing of gold p
rice box-cox transformed data')
```

Figure 8: Second differencing of gold price box-cox transformed data



The above plot shows that the second differencing helped to transform data to stationary. ADF test will be performed to confirm this mathematically.

Dicker-Fuller Unit-Root test (ADF) for second differencing:

```
# Performing ADF on second differencing gold price data

adf.test(gold_sd)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  gold_sd
## Dickey-Fuller = -7.1926, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

The p-value is less than 5% significance level. Hence, null hypothesis is rejected. So, we can say that the gold price data has been successfully transformed to stationary from second differencing. Hence, we can move forward to model specification for the second differenced gold price series.

Model Specification

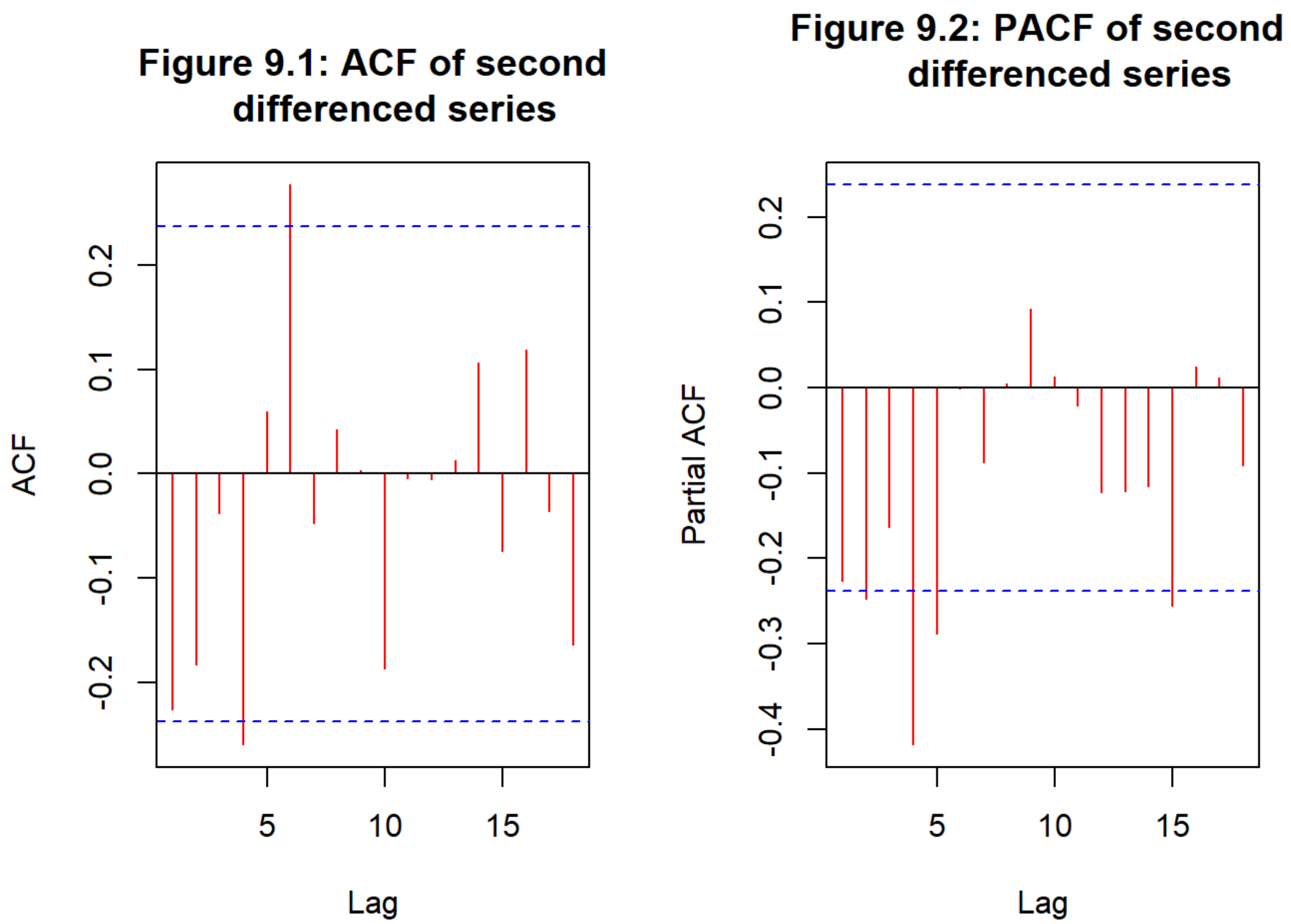
In this section, we will find the most suitable ARIMA(p,d,q) model using the model specification tools known. AR and MA signs were observed when we plotted the time series plot for the gold price data set.

Computing Suitable ARIMA(p,d,q) models

Since, the series is now stationary, suitable ARIMA models will be computed from model specification tools.

ACF and PACF plot:

```
par(mfrow = c(1,2))
acf(gold_sd,  main = "Figure 9.1: ACF of second
    differenced series", col="red")
pacf(gold_sd, main = "Figure 9.2: PACF of second
     differenced series",col="red")
```




```
par(mfrow = c(1,1))
```

From the ACF plot, two significant and one slightly insignificant lags can be observed. The order for MA terms is thus given by values of q which are found to be 2,3. The PACF gives order for AR and it suggests p to be equal to 3,4. The order of differencing is 2. So, the final ARIMA models yielded from ACF and PACF plots are **ARIMA(3,2,2), ARIMA(3,2,3),ARIMA(4,2,2) and ARIMA(4,2,3).**

Extended Autocorrelation Function (EACF):

```
# Generating EACF

eacf(gold_sd,ma.max = 6, ar.max = 6)
```

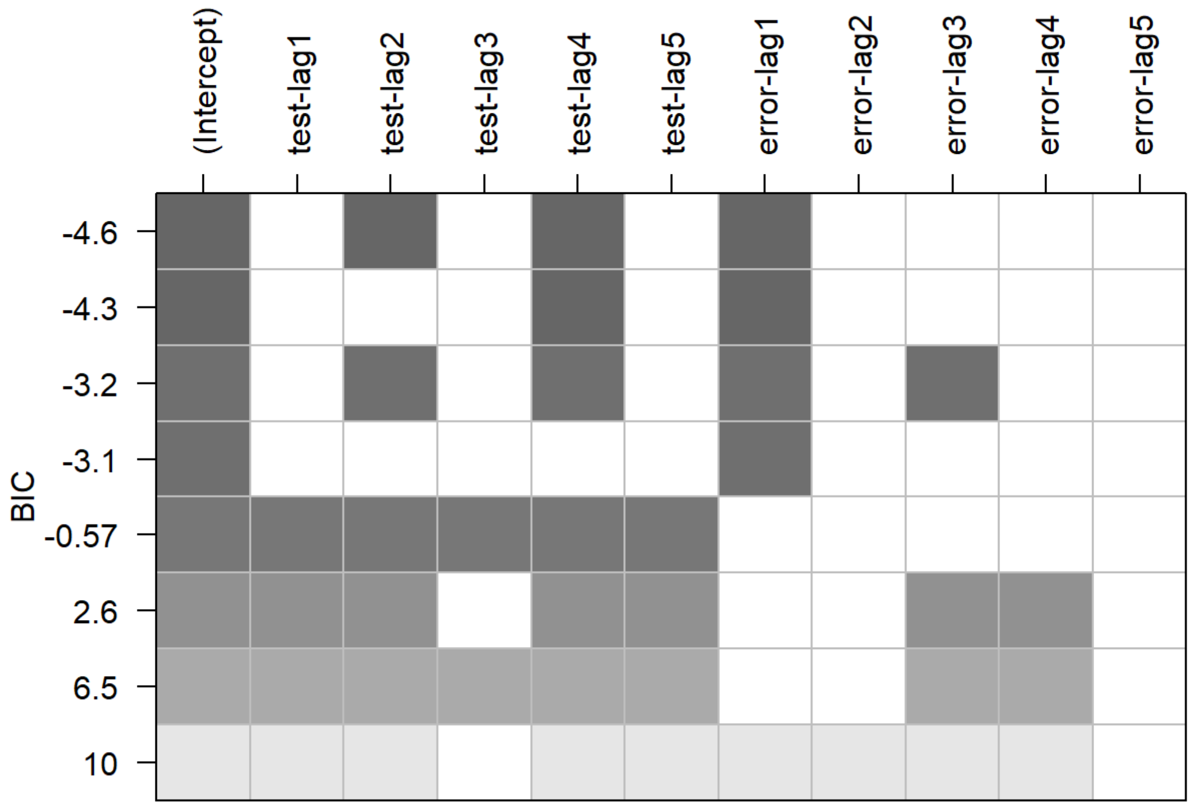
```
## AR/MA
##   0 1 2 3 4 5 6
## 0 o o o x o x o
## 1 x o o x o x o
## 2 x o o x o x o
## 3 x x o x o x o
## 4 x x x x o o o
## 5 o x o x o o o
## 6 o x o o o o o
```

The EACF generated above provide a clear vertex. The top left vertex suggests the possible values of p to be 0,1 and q to be 0,1,2. Hence, the ARIMA models yielded from EACF plot are:
ARIMA(0,2,1),ARIMA(0,2,2),ARIMA(1,2,1) and ARIMA(1,2,2) (with lower orders for AR and MA is included into the set of possible models for the gold price series data.

Bayesian Information Criterion (BIC):

```
# Generating and plotting BIC

bic.gold = armasubsets(y=gold_sd,nar=5,nma=5,y.name='test',ar.method='ols')

plot(bic.gold)
title(main = 'Figure 9: BIC table for second differenced gold price data', line= 6)
```

Figure 9: BIC table for second differenced gold price data



The above table is observed and the AR term (p) is found to be 2,4 while the MA term(q) is found to be only 1 (q=1). Thus, from the BIC table, two ARIMA models are yielded which are: **ARIMA(2,2,1) and ARIMA(4,2,1)**.

Proposed Models:

After performing all the required processes, we found 11 ARIMA models which are:

1. ARIMA(3,2,2), ARIMA(3,2,3),ARIMA(4,2,2) and ARIMA(4,2,3) from the ACF and PACF plots.
2. ARIMA(0,2,1) and ARIMA(0,2,2),ARIMA(1,2,1) and ARIMA(1,2,2) from EACF plot.
3. ARIMA(2,2,1) and ARIMA(4,2,1) from the BIC table.

Parameter Estimation and Model Diagnostics

We will perform model diagnosis for each proposed ARIMA model in order to find the best model for performing forecasting. Parameter estimation will be performed using Maximum likelihood (ML), CSS and both CSS-ML method in order to find the best ARIMA model with significant terms.

ARIMA(2,2,1)

Parameter estimation using CSS, ML and CSS-ML method:

```
#Performing parameter estimation using CSS, ML and CSS-ML method

css_221 = arima(BC.gold, order = c(2,2,1), method = 'CSS')
coeftest(css_221)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error  z value  Pr(>|z|)
## ar1  0.430408   0.122677   3.5085 0.0004507 ***
## ar2 -0.110039   0.123209  -0.8931 0.3718003
## ma1 -0.982211   0.026811 -36.6346 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
ml_221 = arima(BC.gold, order = c(2,2,1), method = 'ML')
coeftest(ml_221)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error  z value  Pr(>|z|)
## ar1  0.426113   0.120405   3.5390 0.0004016 ***
## ar2 -0.111864   0.119653  -0.9349 0.3498410
## ma1 -0.999998   0.047863 -20.8928 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
cssml_221 = arima(BC.gold, order = c(2,2,1), method = 'CSS-ML')
coeftest(cssml_221)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error  z value  Pr(>|z|)
## ar1  0.426175   0.120405   3.5395 0.0004009 ***
## ar2 -0.111842   0.119654  -0.9347 0.3499370
## ma1 -0.999999   0.047858 -20.8949 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the results generated using CSS, ML and CSS-ML method, it is observed that p-value for ar2 term is greater than the 5% level of significance. Therefore, the model is not significant.

ARIMA(3,2,2)

Parameter estimation using CSS, ML and CSS-ML method:

```
#Performing parameter estimation using CSS, ML and CSS-ML method

css_322 = arima(BC.gold, order = c(3,2,2), method = 'CSS')
coeftest(css_322)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value Pr(>|z|)
## ar1 -0.198508   0.693085 -0.2864   0.7746
## ar2  0.150855   0.324875  0.4643   0.6424
## ar3 -0.032717   0.151592 -0.2158   0.8291
## ma1 -0.348767   0.683843 -0.5100   0.6100
## ma2 -0.630984   0.671142 -0.9402   0.3471
```

```
ml_322 = arima(BC.gold, order = c(3,2,2), method = 'ML')
coeftest(ml_322)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value Pr(>|z|)
## ar1 -0.242431   0.668826 -0.3625   0.7170
## ar2  0.161909   0.313979  0.5157   0.6061
## ar3 -0.043242   0.153864 -0.2810   0.7787
## ma1 -0.325760   0.657212 -0.4957   0.6201
## ma2 -0.674203   0.656245 -1.0274   0.3042
```

```
cssml_322 = arima(BC.gold, order = c(3,2,2), method = 'CSS-ML')
coeftest(cssml_322)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value Pr(>|z|)
## ar1 -0.250362   0.669745 -0.3738   0.7085
## ar2  0.165372   0.315530  0.5241   0.6002
## ar3 -0.044424   0.156092 -0.2846   0.7760
## ma1 -0.317950   0.657428 -0.4836   0.6287
## ma2 -0.682049   0.656524 -1.0389   0.2989
```

From the results generated using CSS, ML and CSS-ML method, it is observed that p-value for all the AR and MA terms is greater than the 5% level of significance. Therefore, all the parameters estimations are insignificant.

ARIMA(4,2,1)

Parameter estimation using CSS, ML and CSS-ML method:

```
#Performing parameter estimation using CSS, ML and CSS-ML method

css_421 = arima(BC.gold, order = c(4,2,1), method = 'CSS')
coeftest(css_421)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value Pr(>|z|)
## ar1 -0.016944   0.178356 -0.0950 0.924314
## ar2 -0.300117   0.122387 -2.4522 0.014199 *
## ar3 -0.194548   0.121284 -1.6041 0.108698
## ar4 -0.382758   0.127155 -3.0102 0.002611 **
## ma1 -0.503434   0.174134 -2.8911 0.003839 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
ml_421 = arima(BC.gold, order = c(4,2,1), method = 'ML')
coeftest(ml_421)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value Pr(>|z|)
## ar1 -0.015584   0.180283 -0.0864 0.931117
## ar2 -0.291377   0.121631 -2.3956 0.016594 *
## ar3 -0.186420   0.118622 -1.5715 0.116056
## ar4 -0.361357   0.122848 -2.9415 0.003266 **
## ma1 -0.496819   0.178530 -2.7828 0.005389 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
cssml_421 = arima(BC.gold, order = c(4,2,1), method = 'CSS-ML')
coeftest(cssml_421)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ar1 -0.01549    0.18027 -0.0859 0.931526
## ar2 -0.29130    0.12164 -2.3948 0.016629 *
## ar3 -0.18642    0.11862 -1.5715 0.116071
## ar4 -0.36129    0.12286 -2.9407 0.003275 **
## ma1 -0.49695    0.17851 -2.7839 0.005371 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the results generated using CSS, ML and CSS-ML method, it is observed that p-value for AR(1) and AR(3) terms is greater than the 5% level of significance. Therefore, not all the parameters are significant.

ARIMA(3,2,3)

Parameter estimation using CSS, ML and CSS-ML method:

```
#Performing parameter estimation using CSS, ML and CSS-ML method

css_323 = arima(BC.gold, order = c(3,2,3), method = 'CSS')
coeftest(css_323)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ar1  0.20538    1.34305  0.1529   0.8785
## ar2  0.19967    0.58425  0.3417   0.7325
## ar3 -0.10052    0.22263 -0.4515   0.6516
## ma1 -0.75174    1.35037 -0.5567   0.5777
## ma2 -0.46286    1.20936 -0.3827   0.7019
## ma3  0.22766    0.45686  0.4983   0.6183
```

```
ml_323 = arima(BC.gold, order = c(3,2,3), method = 'ML')
coeftest(ml_323)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ar1  0.03772    0.86563  0.0436   0.9652
## ar2  0.25300    0.47684  0.5306   0.5957
## ar3 -0.12315    0.21765 -0.5658   0.5715
## ma1 -0.60569    0.85860 -0.7054   0.4805
## ma2 -0.60975    0.80599 -0.7565   0.4493
## ma3  0.21544    0.41829  0.5150   0.6065
```

```
cssml_323 = arima(BC.gold, order = c(3,2,3), method = 'CSS-ML')
coeftest(cssml_323)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value Pr(>|z|)
## ar1  0.056005   0.859313  0.0652   0.9480
## ar2  0.246862   0.474723  0.5200   0.6031
## ar3 -0.121339   0.213375 -0.5687   0.5696
## ma1 -0.623776   0.853133 -0.7312   0.4647
## ma2 -0.593676   0.805934 -0.7366   0.4613
## ma3  0.217454   0.414456  0.5247   0.5998
```

From the results generated using CSS, ML and CSS-ML method, it is observed that p-value for all the AR and MA terms is greater than the 5% level of significance. Therefore, all the parameters estimations are insignificant.

ARIMA(1,2,2)

Parameter estimation using CSS, ML and CSS-ML method:

```
#Performing parameter estimation using CSS, ML and CSS-ML method

css_122 = arima(BC.gold, order = c(1,2,2), method = 'CSS')
coeftest(css_122)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ar1  0.18714    0.34006  0.5503  0.58211
## ma1 -0.73839    0.34666 -2.1300  0.03317 *
## ma2 -0.24704    0.33419 -0.7392  0.45978
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
ml_122 = arima(BC.gold, order = c(1,2,2), method = 'ML')
coeftest(ml_122)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ar1  0.14641    0.33502  0.4370  0.66210
## ma1 -0.71934    0.33352 -2.1568  0.03102 *
## ma2 -0.28059    0.33043 -0.8492  0.39578
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
cssml_122 = arima(BC.gold, order = c(1,2,2), method = 'CSS-ML')
coeftest(cssml_122)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ar1  0.14789    0.33458  0.4420  0.65847
## ma1 -0.72057    0.33335 -2.1616  0.03065 *
## ma2 -0.27931    0.33022 -0.8458  0.39765
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the results generated using CSS, ML and CSS-ML method, it is observed that p-value for all AR(1) and MA(2) terms is greater than the 5% level of significance. Therefore, not all the parameters estimation are significant.

ARIMA(1,2,1)

Parameter estimation using CSS, ML and CSS-ML method:

```
#Performing parameter estimation using CSS, ML and CSS-ML method

css_121 = arima(BC.gold, order = c(1,2,1), method = 'CSS')
coeftest(css_121)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value  Pr(>|z|)
## ar1  0.400077   0.116235   3.442 0.0005775 ***
## ma1 -0.993002   0.021744 -45.668 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
ml_121 = arima(BC.gold, order = c(1,2,1), method = 'ML')
coeftest(ml_121)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error  z value  Pr(>|z|)
## ar1  0.386166   0.113459   3.4036 0.0006651 ***
## ma1 -0.999999   0.044361 -22.5423 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
cssml_121 = arima(BC.gold, order = c(1,2,1), method = 'CSS-ML')
coeftest(cssml_121)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error  z value  Pr(>|z|)
## ar1  0.386165   0.113459   3.4036 0.0006651 ***
## ma1 -1.000000   0.044361 -22.5423 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the results generated using CSS, ML and CSS-ML method, it is observed that p-value for all AR and MA terms is less than the 5% level of significance. Therefore, all the parameter estimations are significant.

ARIMA(0,2,2)

Parameter estimation using CSS, ML and CSS-ML method:

```
#Performing parameter estimation using CSS, ML and CSS-ML method

css_022 = arima(BC.gold, order = c(0,2,2), method = 'CSS')
coeftest(css_022)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value  Pr(>|z|)
## ma1 -0.56293    0.10991 -5.1216  3.03e-07 ***
## ma2 -0.40733    0.11026 -3.6942 0.0002206 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
ml_022 = arima(BC.gold, order = c(0,2,2), method = 'ML')
coeftest(ml_022)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value  Pr(>|z|)
## ma1 -0.59005    0.11951 -4.9372 7.925e-07 ***
## ma2 -0.40994    0.11073 -3.7022 0.0002137 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
cssml_022 = arima(BC.gold, order = c(0,2,2), method = 'CSS-ML')
coeftest(cssml_022)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value  Pr(>|z|)
## ma1 -0.59005    0.11951 -4.9372 7.926e-07 ***
## ma2 -0.40993    0.11073 -3.7021 0.0002138 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the results generated using CSS, ML and CSS-ML method, it is observed that p-value for all MA terms is less than the 5% level of significance. Therefore, all the parameter estimation are significant.

ARIMA(4,2,2)

Parameter estimation using CSS, ML and CSS-ML method:

```
#Performing parameter estimation using CSS, ML and CSS-ML method

css_422 = arima(BC.gold, order = c(4,2,2), method = 'CSS')
coeftest(css_422)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ar1  0.21501    0.47207  0.4555  0.64877
## ar2 -0.41196    0.19069 -2.1603  0.03075 *
## ar3 -0.17698    0.12720 -1.3914  0.16411
## ar4 -0.37142    0.15714 -2.3635  0.01810 *
## ma1 -0.75811    0.51317 -1.4773  0.13960
## ma2  0.26376    0.42021  0.6277  0.53021
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
ml_422 = arima(BC.gold, order = c(4,2,2), method = 'ML')
coeftest(ml_422)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ar1  0.20211    0.44138  0.4579  0.64703
## ar2 -0.39815    0.18515 -2.1504  0.03152 *
## ar3 -0.17123    0.12259 -1.3968  0.16246
## ar4 -0.35340    0.14588 -2.4226  0.01541 *
## ma1 -0.73504    0.47863 -1.5357  0.12461
## ma2  0.24737    0.38939  0.6353  0.52525
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
cssml_422 = arima(BC.gold, order = c(4,2,2), method = 'CSS-ML')
coeftest(cssml_422)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ar1  0.20392    0.44436  0.4589  0.64630
## ar2 -0.39876    0.18555 -2.1490  0.03163 *
## ar3 -0.17112    0.12274 -1.3941  0.16329
## ar4 -0.35307    0.14646 -2.4107  0.01592 *
## ma1 -0.73703    0.48193 -1.5293  0.12618
## ma2  0.24908    0.39179  0.6357  0.52495
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the results generated using CSS, ML and CSS-ML method, it is observed that p-value for AR(1), AR(3) and all MA terms (MA(1) and MA(2)) are greater than the 5% level of significance. Therefore, not all the parameters estimation generated are significant.

ARIMA(4,2,3)

Parameter estimation using CSS, ML and CSS-ML method:

```
#Performing parameter estimation using CSS, ML and CSS-ML method

css_423 = arima(BC.gold, order = c(4,2,3), method = 'CSS')
coeftest(css_423)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ar1  0.46332    0.37579  1.2329 0.217611
## ar2 -0.17111    0.37578 -0.4554 0.648848
## ar3 -0.30646    0.24129 -1.2701 0.204049
## ar4 -0.25915    0.14814 -1.7494 0.080220 .
## ma1 -1.06394    0.37340 -2.8494 0.004381 **
## ma2  0.17893    0.55670  0.3214 0.747892
## ma3  0.41178    0.33017  1.2472 0.212335
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
ml_423 = arima(BC.gold, order = c(4,2,3), method = 'ML')
coeftest(ml_423)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value Pr(>|z|)
## ar1  0.482083   0.376336  1.2810  0.20020
## ar2 -0.098481   0.400968 -0.2456  0.80599
## ar3 -0.321465   0.255030 -1.2605  0.20749
## ar4 -0.223646   0.127803 -1.7499  0.08013 .
## ma1 -1.117963   0.588161 -1.9008  0.05733 .
## ma2  0.152550   0.674134  0.2263  0.82098
## ma3  0.507924   0.459197  1.1061  0.26868
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
cssml_423 = arima(BC.gold, order = c(4,2,3), method = 'CSS-ML')
coeftest(cssml_423)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ar1  0.48764    0.37930  1.2856  0.19857
## ar2 -0.10432    0.40218 -0.2594  0.79533
## ar3 -0.31806    0.25504 -1.2471  0.21236
## ar4 -0.22314    0.12829 -1.7393  0.08198 .
## ma1 -1.12433    0.63663 -1.7661  0.07739 .
## ma2  0.16222    0.70078  0.2315  0.81694
## ma3  0.50300    0.47864  1.0509  0.29331
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the results generated using CSS, ML and CSS-ML method, it is observed that p-value for all AR and MA terms are greater than the 5% level of significance. Therefore, no parameters estimation generated are significant.

ARIMA(0,2,1)

Parameter estimation using CSS, ML and CSS-ML method:

```
#Performing parameter estimation using CSS, ML and CSS-ML method

css_021 = arima(BC.gold, order = c(0,2,1), method = 'CSS')
coeftest(css_021)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value  Pr(>|z|)
## ma1 -0.83093    0.12904 -6.4394 1.199e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
ml_021 = arima(BC.gold, order = c(0,2,1), method = 'ML')
coeftest(ml_021)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value Pr(>|z|)
## ma1 -0.79518    0.12237 -6.4982 8.13e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
cssml_021 = arima(BC.gold, order = c(0,2,1), method = 'CSS-ML')
coeftest(cssml_021)
```

```
##
## z test of coefficients:
##
##     Estimate Std. Error z value  Pr(>|z|)
## ma1 -0.79518    0.12237 -6.4982 8.128e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the results generated using CSS, ML and CSS-ML method, it is observed that there is only MA term in the model MA(1) whose p-value is less than the 5% level of significance. Therefore, the single parameters estimation generated is significant.

Insight from Parameter Estimation performed:

The results generated after performing parameter estimation for all the proposed ARIMA models suggested that only **ARIMA(0,2,2), ARIMA(1,2,1) and ARIMA(0,2,1)** have significant AR and MA terms. Thus, these three models can be considered suitable estimated models out of all the proposed ARIMA models.

Finding Best Model based on AIC and BIC score:

A sort.score user-defined function will be used to calculate the best model out of the 10 ARIMA models proposed. The function will take two parameters- one is the model and other is the score whose value can be only AIC and BIC. The function will collectively calculate AIC and BIC score for all the models and sort them in ascending order of their AIC/BIC score.

```
#Defining function sort.score

sort.score <- function(x, score = c("bic", "aic")){
  if (score == "aic"){
    x[with(x, order(AIC)),]
  } else if (score == "bic") {
    x[with(x, order(BIC)),]
  } else {
    warning('score = "x" only accepts valid arguments ("aic","bic")')
  }
}
```

AIC:

Calculating AIC score collectively for all models and sorting them in ascending order based on their AIC score-

```
#Sorting based on AIC

sort.score(AIC(cssml_322,cssml_323, cssml_422, cssml_423, cssml_021, cssml_121, cssml_022, cs
sml_221, cssml_421, cssml_122), score = "aic")
```

	df <dbl>	AIC <dbl>
cssml_022	3	-182.4274
cssml_121	3	-181.8243
cssml_421	6	-180.7765
cssml_221	4	-180.6885
cssml_122	4	-180.6105
cssml_422	7	-179.3425
cssml_423	8	-178.9839
cssml_322	6	-176.8207
cssml_323	7	-174.9940
cssml_021	2	-174.0301

1-10 of 10 rows

From the above ouput, it is seen that ARIMA(0,2,2) is the best model as it has the lowest AIC value among the other ARIMA models. However, ARIMA(1,2,1) has a comparatively smaller score as well.

BIC:

Calculating BIC score collectively for all models and sorting them in ascending order based on their BIC score-

```
#Sorting based on AIC

sort.score(BIC(cssml_322,cssml_323, cssml_422, cssml_423, cssml_021, cssml_121, cssml_022, cs
sml_221, cssml_421, cssml_122), score = "bic")
```

	df	BIC
	<dbl>	<dbl>
cssml_022	3	-175.7688
cssml_121	3	-175.1658
cssml_221	4	-171.8105
cssml_122	4	-171.7325
cssml_021	2	-169.5910
cssml_421	6	-167.4594
cssml_422	7	-163.8060
cssml_322	6	-163.5036
cssml_423	8	-161.2278
cssml_323	7	-159.4575

1-10 of 10 rows

From the above output, it is seen that ARIMA(0,2,2) has the lowest BIC value, followed by ARIMA(1,2,1), with a very slight difference in the scores, among the other ARIMA models. Therefore, residual analysis for both the models will be performed.

Residual Analysis

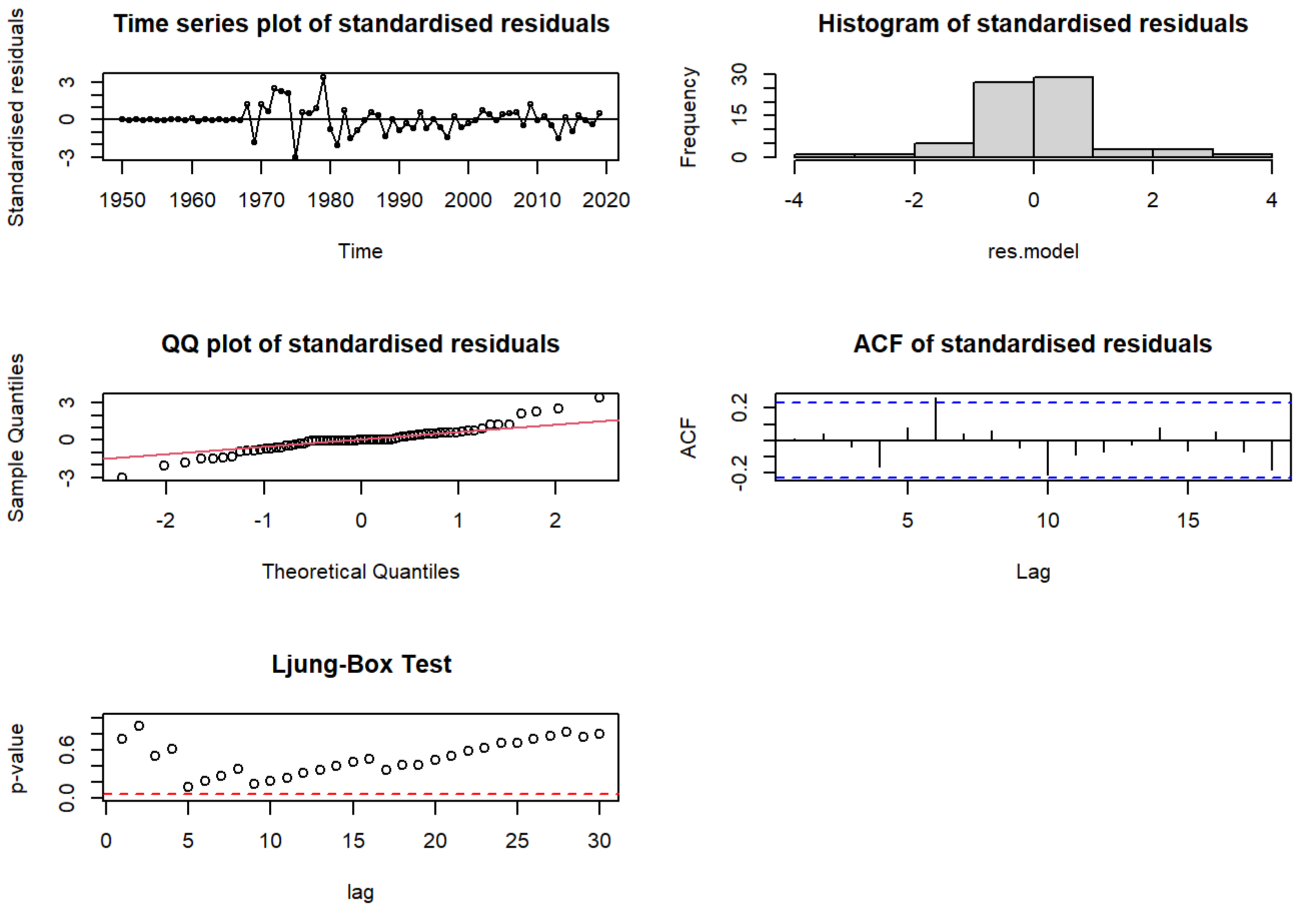
A user-defined function residual.analysis will be created to perform residual analysis for ARIMA models. The function will generate time-series plot, Histogram, Q-Q plot, ACF,Shapiro-Wilk and Ljung box test for the ARIMA model passed in the function.

```
residual.analysis <- function(model, std = TRUE,start = 2, class = c("ARIMA")[1]){
  if (class == "ARIMA"){
    if (std == TRUE){
      res.model = rstandard(model)
    }else{
      res.model = residuals(model)
    }
  }
  else {
    stop("The argument 'class' must be = 'ARIMA' ")
  }
  par(mfrow=c(3,2))
  plot(res.model,type='o',ylab='Standardised residuals', main="Time series plot of standardis
ed residuals")
  abline(h=0)
  hist(res.model,main="Histogram of standardised residuals")
  qqnorm(res.model,main="QQ plot of standardised residuals")
  qqline(res.model, col = 2)
  acf(res.model,main="ACF of standardised residuals")
  print(shapiro.test(res.model))
  k=0
  LBQPlot(res.model, lag.max = 30, StartLag = k + 1, k = 0, SquaredQ = FALSE)
  par(mfrow=c(1,1))
}
```

Performing residual analysis for ARIMA(0,2,2):

```
#Residual analysis for cssml_022

residual.analysis(model = cssml_022)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.9246, p-value = 0.0004221
```




```
par(mar=c(1,1,1,1))
```

Intepretation of Residual Analysis for ARIMA(0,2,2):

1. Shapiro-Wilk test: The Shapiro-Wilk test resulted in a p-value less than 0.05, which indicates that there is enough evidence to state that residuals are not normally distributed. Therefore, the assumption of normal distribution of residuals is violated.

2. Time Series plot: Standardization makes it much easier to spot residuals of unusual magnitude. The standardised residuals from the ARIMA(0,2,2) model fitted to the gold price series shows no trend as the residuals scatter around the horizontal mean level.

3. Histogram: The histogram is not symmetrical(normal) with some non-symmetric bars on both the edges and in the middle. The histogram shows that the residuals do not seem to follow a normal distribution and support the results obtained from Shapiro-Wilk test.

4. Q-Q plot: From the Q-Q plot, the residuals do not seem to follow the quantile line especially during the initial points and in the end. The residuals are not following a normal distribution as per the Q-Q plot and supports the results obtained from Shapiro-Wilk test.

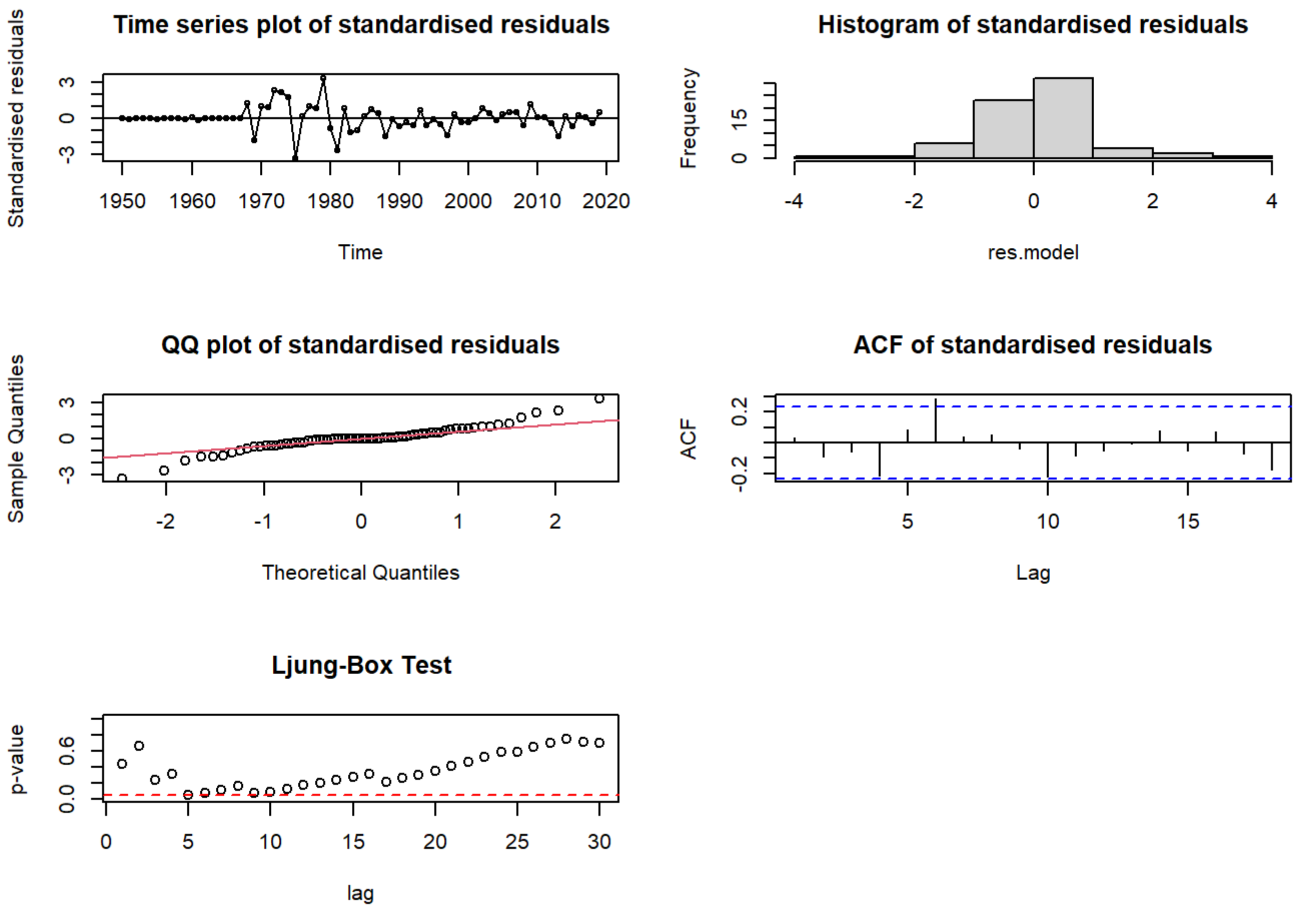
5. ACF: The sample autocorrelation function of the residuals is used to assess the independence of the noise components in the model. The lag 6 autocorrelation here is greater than 0.1. The lag 10 autocorrelation is greater than -0.2. We conclude that the graph does not show statistically significant evidence of non-zero autocorrelation in the residuals.

6. Ljung-Box Test: The Ljung-Box test is a comprehensive technique for examining residual correlations in general. From the graph it is observed that the estimated model seem to be capturing the dependence structure of the gold price time series well. Hence, the error terms are not uncorrelated.

Performing residual analysis for ARIMA(1,2,1):

```
#Residual analysis for cssml_121

residual.analysis(model = cssml_121)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.91967, p-value = 0.0002547
```




```
par(mar=c(1,1,1,1))
```

Intepretation of Residual Analysis for ARIMA(1,2,1):

1. Shapiro-Wilk test: The p-value was less than 0.05, indicating that there is sufficient evidence to conclude that residuals are not normally distributed. As a result, the assumption of normal residual distribution is violated for this model as well.

2. Time Series plot: As the residuals spread around the horizontal mean level, the standardised residuals from the ARIMA(1,2,1) model fitted to the gold price series indicate no trend.

3. Histogram: The histogram demonstrates asymmetric behavior indicating that the residuals do not appear to follow a normal distribution, confirming the Shapiro-Wilk test results.

4. Q-Q plot: The residuals do not appear to follow the quantile line in the Q-Q plot, especially at the beginning and end. According to the Q-Q plot, the residuals do not follow a normal distribution, which verifies the Shapiro-Wilk test results.

5. ACF: The ACF plot shows three significant lags beyond the confidence limits for this model. Moreover, the model ARIMA(0,2,2) had first insignificance at a lag further away from the origin.

6. Ljung-Box Test: From the graph it is observed that the estimated model seem have some moderate autocorrelations.

To sum up, residual analysis for the models ARIMA(0,2,2) and ARIMA(1,2,1) violates the assumption of normality. However, ARIMA(0,2,2) have uncorrelated residuals as compared to ARIMA(1,2,1). Moreover, ARIMA(0,2,2), has lower AIC and BIC scores, therefore, it is considered as the most suitable model representing the gold price time series.

Overfitting and Parameter Redundancy:

Another approach for detecting abnormalities in terms of goodness of fit is overfitting. We acquire a somewhat more broad model after describing and fitting what we think to be an appropriate model; that is, a model “near by” that incorporates the original model as a special instance.

Overfitting model for ARIMA(0,2,2) are : ARIMA(1,2,2) and ARIMA(0,2,3). Performing parameter estimation and model diagnostics on ARIMA(0,2,3) as parameters has been estimated previously for ARIMA(1,2,2).

Parameter estimation for ARIMA(0,2,3):

Using CSS, ML and CSS-ML method:

```
#Performing parameter estimation using CSS, ML and CSS-ML method

css_023 = arima(BC.gold, order = c(0,2,3), method = 'CSS')
coeftest(css_023)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value Pr(>|z|)
## ma1 -0.551163   0.118929 -4.6344 3.58e-06 ***
## ma2 -0.372656   0.152579 -2.4424  0.01459 *
## ma3 -0.048068   0.127426 -0.3772  0.70601
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
ml_023 = arima(BC.gold, order = c(0,2,3), method = 'ML')
coeftest(ml_023)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value  Pr(>|z|)
## ma1 -0.577416   0.127728 -4.5207 6.164e-06 ***
## ma2 -0.375243   0.152883 -2.4544   0.01411 *
## ma3 -0.047326   0.126972 -0.3727   0.70935
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
cssml_023 = arima(BC.gold, order = c(0,2,3), method = 'CSS-ML')
coeftest(cssml_023)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error z value  Pr(>|z|)
## ma1 -0.577440   0.127731 -4.5207 6.162e-06 ***
## ma2 -0.375212   0.152897 -2.4540   0.01413 *
## ma3 -0.047347   0.126977 -0.3729   0.70924
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the above outputs, it is observed that the coefficients generated from all the 3 methods obtained one insignificant parameter for the model as MA(3) term have p-value greater than 5% significance level. Hence, not all the parameters estimated are significant. Therefore, the overfitted model ARIMA(0,2,3) will not be considered suitable in comparison to other models. Moreover, the other neighbouring model ARIMA(1,2,2), resulted in insignificant AR(1) and MA(2) terms. Therefore, it is also not considered influential.

Final Result:

The residual analysis was performed on all the other proposed ARIMA models as well using the user-defined residual.analysis() function and it generated similar results which matched the ARIMA(0,2,2) and hence the residual analysis for them are not included in the report.

So, this resulted in moving to the other criteria to find the best model which is the AIC and BIC. Based on AIC and BIC, the best model was found to be ARIMA(0,2,2). Moreover, the overfiiting models for ARIMA(0,2,2) showed insignificant result.

So ,for the best model, the residual analysis generated shows that the best model satisfies all the ideal model assumptions except the assumption of normality which seems to be violated.

Best Model:

From the parameter estimation,model diagnostics and residual analysis performed on all the possible ARIMA models, it is found that ARIMA(0,2,2) is the best model.

Forecasting of gold prices (USD) for next 10 years

The ARIMA(0,2,2) model is the best model and it will be utilized to predict the gold prices for the next 10 years i.e. from 2020-2029.

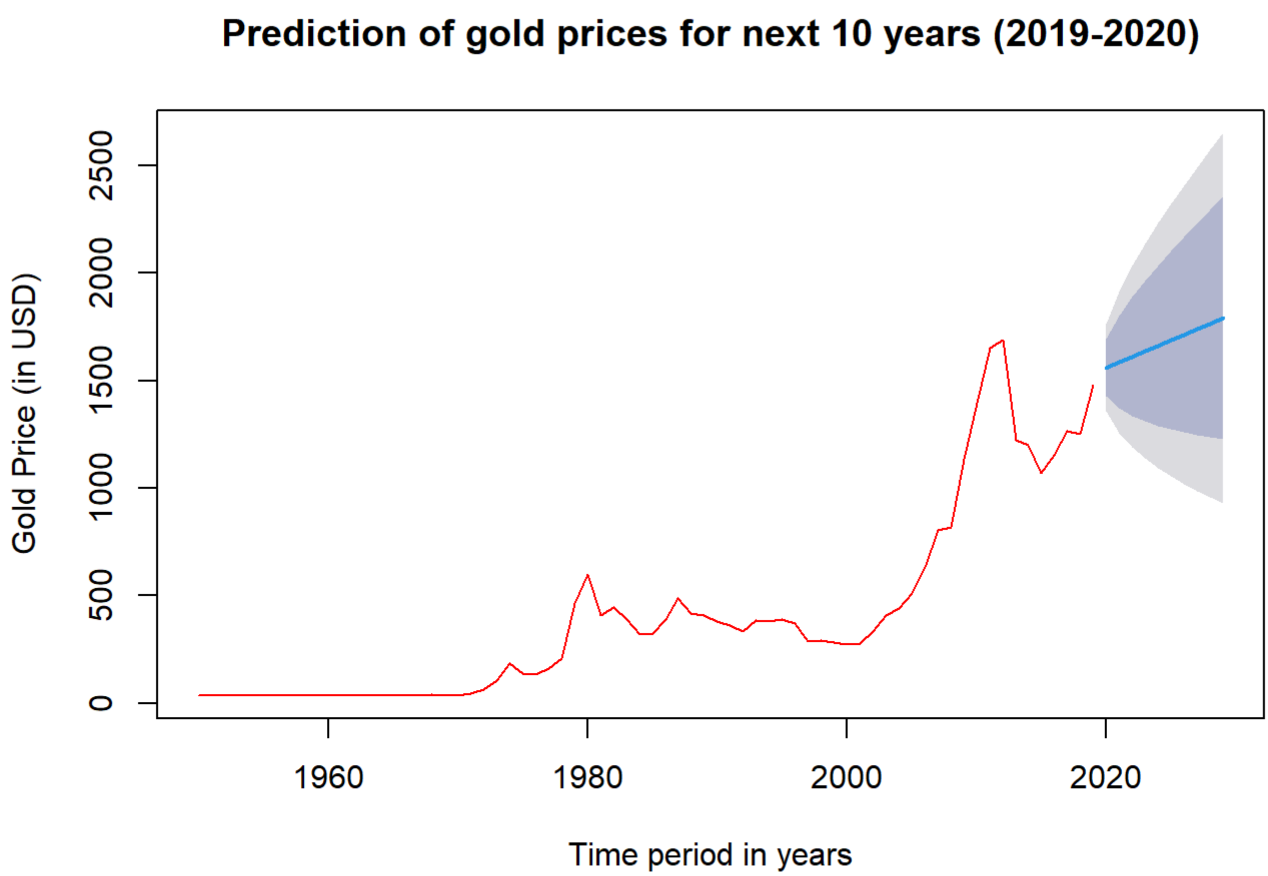
```
#Predicting gold prices for next 10 years using ARIMA(0,2,2)

fit = Arima(gold,c(0,2,2))
forecast(fit,h=10)
```

```
##      Point Forecast    Lo 80    Hi 80     Lo 95    Hi 95
## 2020       1560.488 1427.674 1693.302 1357.3659 1763.610
## 2021       1585.972 1371.135 1800.809 1257.4071 1914.537
## 2022       1611.456 1336.109 1886.804 1190.3490 2032.564
## 2023       1636.941 1310.384 1963.497 1137.5158 2136.366
## 2024       1662.425 1290.082 2034.768 1092.9748 2231.875
## 2025       1687.909 1273.381 2102.438 1053.9427 2321.876
## 2026       1713.393 1259.260 2167.527 1018.8558 2407.931
## 2027       1738.878 1247.078 2230.677  986.7356 2491.020
## 2028       1764.362 1236.406 2292.317  956.9238 2571.800
## 2029       1789.846 1226.939 2352.753  928.9537 2650.739
```

The above output shows predicted gold prices for the next 10 years from 2020-2029 (Point Forecast). It can be seen that the gold prices seems to be increasing even though at a small rate for the next 10 years. The predicted gold price for the year 2020 is $1560.488 (USD) and for 2029 it gradually increased to $1789.846 (USD). So, the forecast suggests that the gold prices will increase to more than $200(USD) from 2020 to 2029.

Plotting Forecast:

```
plot(forecast(fit,h=10),
     main = "Prediction of gold prices for next 10 years (2019-2020)",
     xlab = "Time period in years",
     ylab = "Gold Price (in USD)", col="red")
```



The above time-series plot suggests that the prices of gold seems to be increasing with time, hence, the gold prices will continue to follow an upward trend for the next 10 years (2019-2029).

Discussion and Conclusion

The time-series analysis was successfully performed on the annual gold price dataset. Due to non-seasonal behavior of the series, only ARIMA models were computed for the gold price series. Using the model specification tools, suitable ARIMA models were proposed. Parameter estimation, model diagnostics, and residual analysis was performed on each of the proposed ARIMA model. As the latter performed for all the models yielded similar results, AIC and BIC criteria was used to compute the best ARIMA model. From the AIC and BIC criterion, the best model was found to be ARIMA(0,2,2). The best model was finally used for predicting the prices of gold for the coming 10 years. The gold prices predicted for the next 10 years showed that the prices are going to increase gradually each year and in the time span of the next 10 years they will increase to more than $200 (USD).

References

- Annual Gold Prices. (2021). Retrieved 7 June 2021, from https://datahub.io/core/gold-prices#data-cli (https://datahub.io/core/gold-prices#data-cli).

Appendix

1. The packages used were TSA, fUnitRoots, lmtest, FitAR, tseries, readr, dplyr, magrittr,tidyr,stats,forecast.

2. Abbreviations used are given below:

- **AR** - Auto Regressive model
- **MA** - Moving average model
- **ARIMA** -Auto Regressive integrated moving average model
- **AIC**- Akaike information criterion
- **BIC**- Bayesian information criterion