Gold Ounce Price Prediction Trials

Data Preparation

In this self-developed study, my aim was to forecast gold ounce prices in terms of USD by using Gold ounce prices daily history data from 29.12.1978 to 28.02.2020 and it obtained from (Gold.org) https://www.gold.org/download/file/8369/Prices.xlsx (https://www.gold.org/download/file/8369/Prices.xlsx) (the webpage may require membership to download the dataset)

(This content is for informal purposes only, NO INVESTMENT ADVICE!)

Note : I learned theory of these content below from several MOOCs but especially from Practical Time Series Analysis by The State University of New York provided from COURSERA https://www.coursera.org/learn/practical-time-series-analysis (https://www.coursera.org/learn/practical-time-series-analysis)

```
library(tidyverse)
library(readxl)
library(astsa)
library(forecast)
library(zoo)
```

```
df <- read_excel("prices.xlsx", skip = 8, sheet = "Daily")
```

```
## New names:
## * `US dollar` -> `US dollar...2`
## * Euro -> Euro...3
## * `Canadian dollar` -> `Canadian dollar...6`
## * `Chinese renmimbi` -> `Chinese renmimbi...9`
## * `US dollar` -> `US dollar...10`
## * ... and 4 more problems
```

```
df <- df[1:2]
colnames(df) <- c("dates", "Ounce_Price")
str(df)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    10741 obs. of  2 variables:
##  $ dates      : POSIXct, format: "1978-12-29" "1979-01-01" ...
##  $ Ounce_Price: num  226 226 227 219 223 ...
```

```
head(df)
```

```
## # A tibble: 6 x 2
##   dates               Ounce_Price
##   <dttm>                    <dbl>
## 1 1978-12-29 00:00:00        226
## 2 1979-01-01 00:00:00        226
## 3 1979-01-02 00:00:00        227.
## 4 1979-01-03 00:00:00        219.
## 5 1979-01-04 00:00:00        223.
## 6 1979-01-05 00:00:00        226.
```

```
tail(df)
```

```
## # A tibble: 6 x 2
##   dates               Ounce_Price
##   <dttm>                    <dbl>
## 1 2020-02-21 00:00:00       1643.
## 2 2020-02-24 00:00:00       1672.
## 3 2020-02-25 00:00:00       1650.
## 4 2020-02-26 00:00:00       1635.
## 5 2020-02-27 00:00:00       1652
## 6 2020-02-28 00:00:00       1610.
```

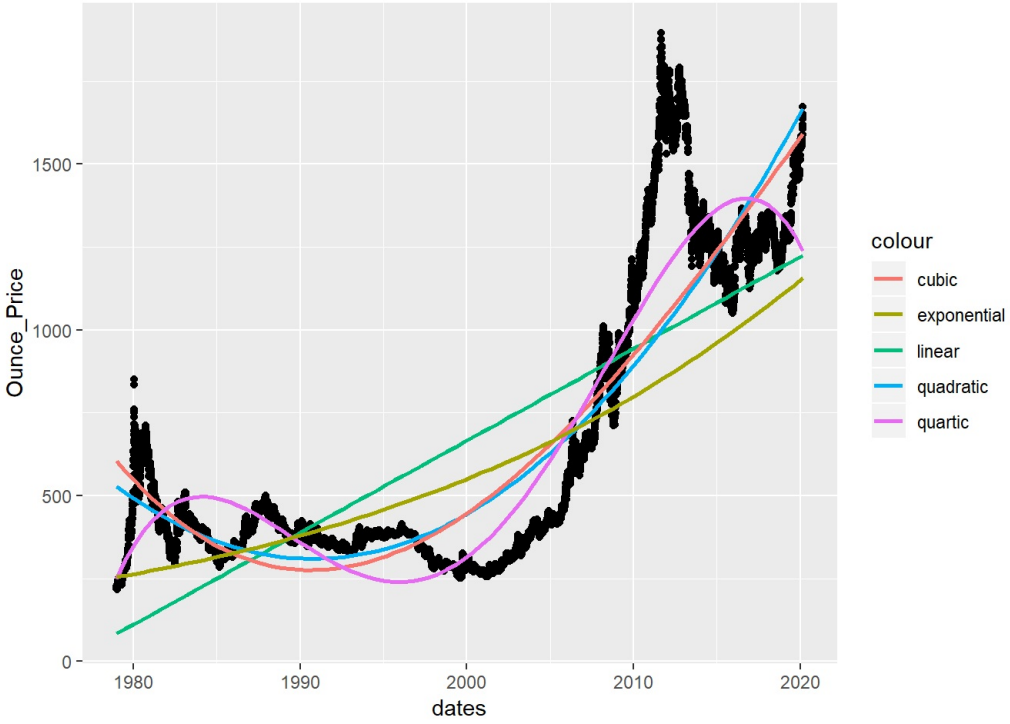
It seems that the dataset does not include financial holidays, like weekends etc. So I first planned to complete missing dates with the Ounce Price values of previous days of filled ones. So I used `complete()` function to add missing dates and when the missing dates filled, Ounce Prices corresponds to filled dates filled with NA values. So I used `na.locf()` fuction from `zoo` package to fill **NA** values with previous values.

```
df1 <- df %>%
  complete(dates = seq(dates[1], as.POSIXct(strptime("2020-02-29", "%Y-%m-%d")), by = "1 day"),
           fill = list(Ounce_Price = NA))

df1 <- na.locf(df1)

df1$dates <- as.Date(df1$dates)
```

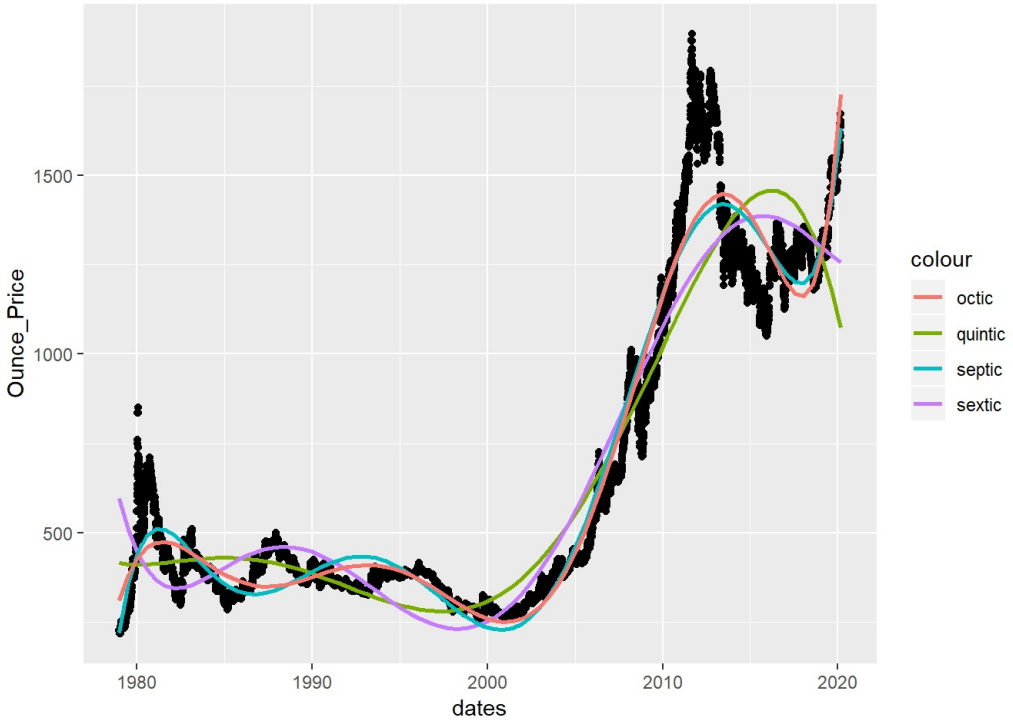
Linear Regression

Under this topic, I first tried Linear Regression. I set my model by taking logarithm of Ounce_Price as my target variable and using dates as predictors.

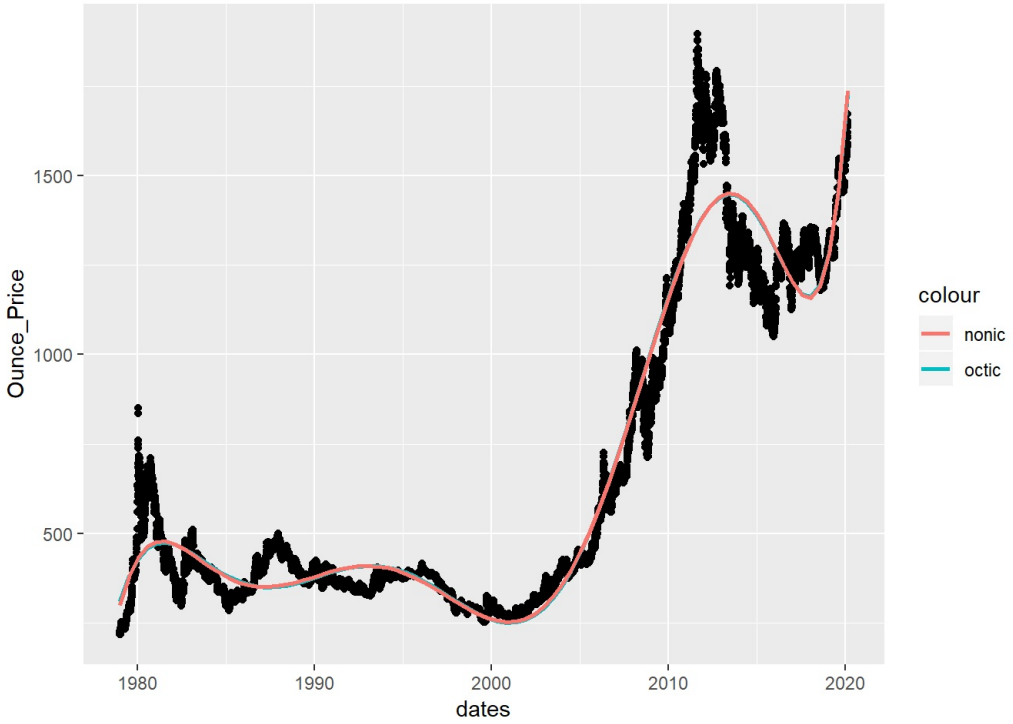
```
exp.model <- lm(log(Ounce_Price)~dates,data = df1)
exp.model.df <- data.frame(x=df1$dates,
                           y=exp(fitted(exp.model)))

ggplot(df1, aes(x = dates, y = Ounce_Price)) + geom_point() +
  stat_smooth(method = 'lm', aes(colour = 'linear'), se = FALSE) +
  stat_smooth(method = 'lm', formula = y ~ poly(x,2), aes(colour = 'quadratic'), se= FALSE) +
  stat_smooth(method = 'lm', formula = y ~ poly(x,3), aes(colour = 'cubic'), se = FALSE)+
  stat_smooth(method = 'lm', formula = y ~ poly(x,4), aes(colour = 'quartic'), se = FALSE)+
  stat_smooth(data=exp.model.df, method = 'loess',aes(x,y,colour = 'exponential'), se = FALSE)
```




```
ggplot(df1, aes(x = dates, y = Ounce_Price)) + geom_point() +
  stat_smooth(method = 'lm', formula = y ~ poly(x,5), aes(colour = 'quintic'), se = FALSE)+
  stat_smooth(method = 'lm', formula = y ~ poly(x,6), aes(colour = 'sextic'), se = FALSE)+
  stat_smooth(method = 'lm', formula = y ~ poly(x,7), aes(colour = 'septic'), se = FALSE)+
  stat_smooth(method = 'lm', formula = y ~ poly(x,8), aes(colour = 'octic'), se = FALSE)
```




```
ggplot(df1, aes(x = dates, y = Ounce_Price)) + geom_point() +
  stat_smooth(method = 'lm', formula = y ~ poly(x,8), aes(colour = 'octic'), se = FALSE)+
  stat_smooth(method = 'lm', formula = y ~ poly(x,9), aes(colour = 'nonic'), se = FALSE)
```

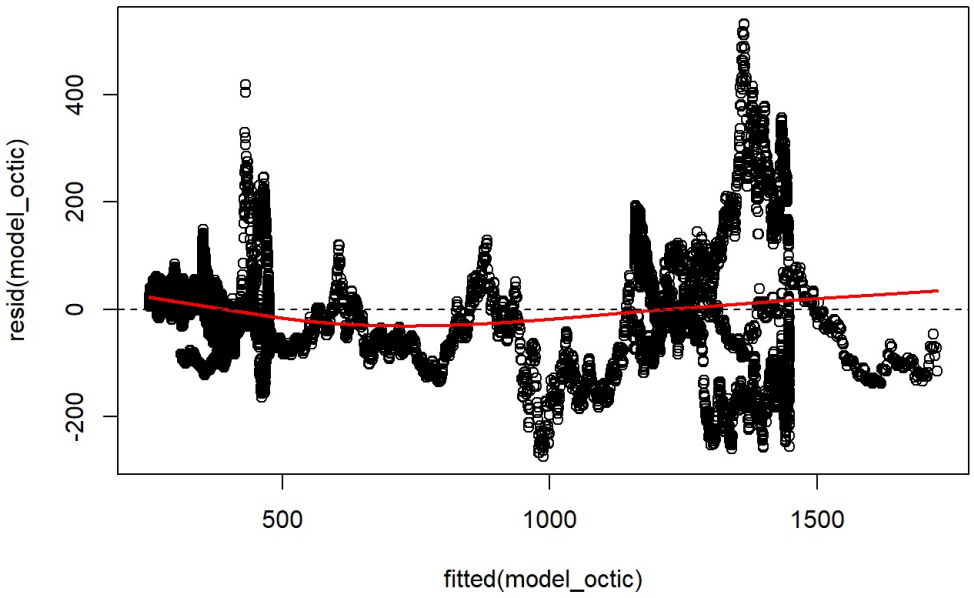


From generated graphs, best fitted line seems to be generated from taking into account octic polynomial degree of dates. So I fitted model by taking octic polynomial degree of dates as suggested.

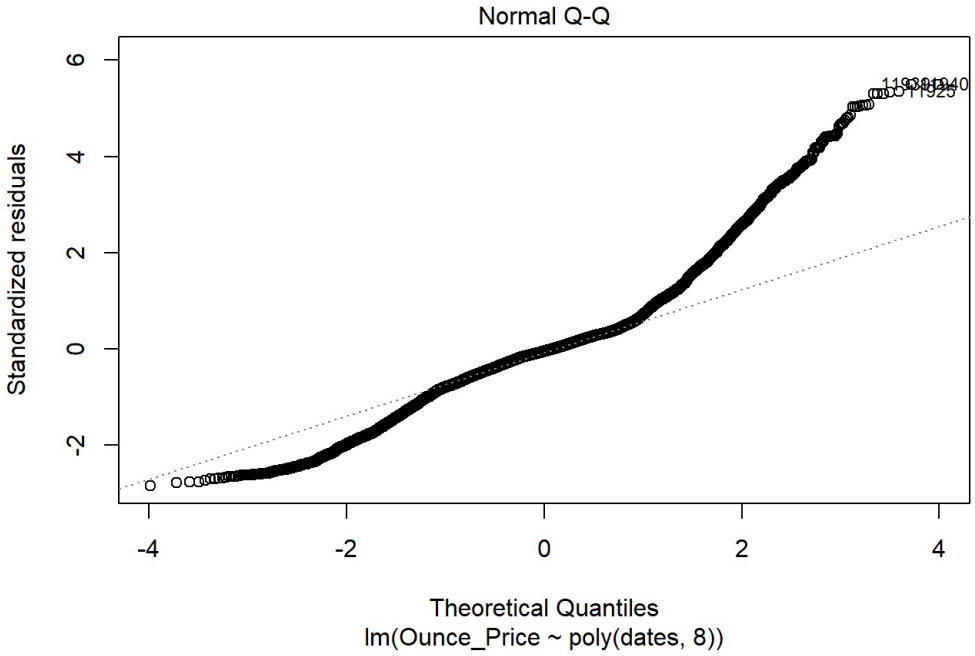
```
model_octic <- lm(data = df1, Ounce_Price~poly(dates,8))
summary(model_octic)
```

```
##
## Call:
## lm(formula = Ounce_Price ~ poly(dates, 8), data = df1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -274.55  -50.30   -4.55   35.37  531.36
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       6.544e+02  7.877e-01  830.81   <2e-16 ***
## poly(dates, 8)1   4.041e+04  9.659e+01  418.36   <2e-16 ***
## poly(dates, 8)2   2.434e+04  9.659e+01  252.02   <2e-16 ***
## poly(dates, 8)3 -3.552e+03  9.659e+01  -36.78   <2e-16 ***
## poly(dates, 8)4 -1.447e+04  9.659e+01 -149.84   <2e-16 ***
## poly(dates, 8)5 -6.099e+03  9.659e+01  -63.15   <2e-16 ***
## poly(dates, 8)6  6.196e+03  9.659e+01   64.15   <2e-16 ***
## poly(dates, 8)7  1.199e+04  9.659e+01  124.18   <2e-16 ***
## poly(dates, 8)8  2.722e+03  9.659e+01   28.18   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 96.59 on 15028 degrees of freedom
## Multiple R-squared:  0.9502, Adjusted R-squared:  0.9502
## F-statistic: 3.583e+04 on 8 and 15028 DF,  p-value: < 2.2e-16
```

```
lo <- loess(resid(model_octic) ~ fitted(model_octic), degree = 1, span=0.8)
plot(fitted(model_octic),resid(model_octic))
lines(fitted(model_octic),predict(lo), col='red', lwd=2)
abline(a=0, b=0, lty=2)
```




```
plot(model_octic, which =2)
```



Although model coefficients are statistically significant, yet the model assumptions still does not met in terms of Normality and homoscedastisity of variances among residuals.

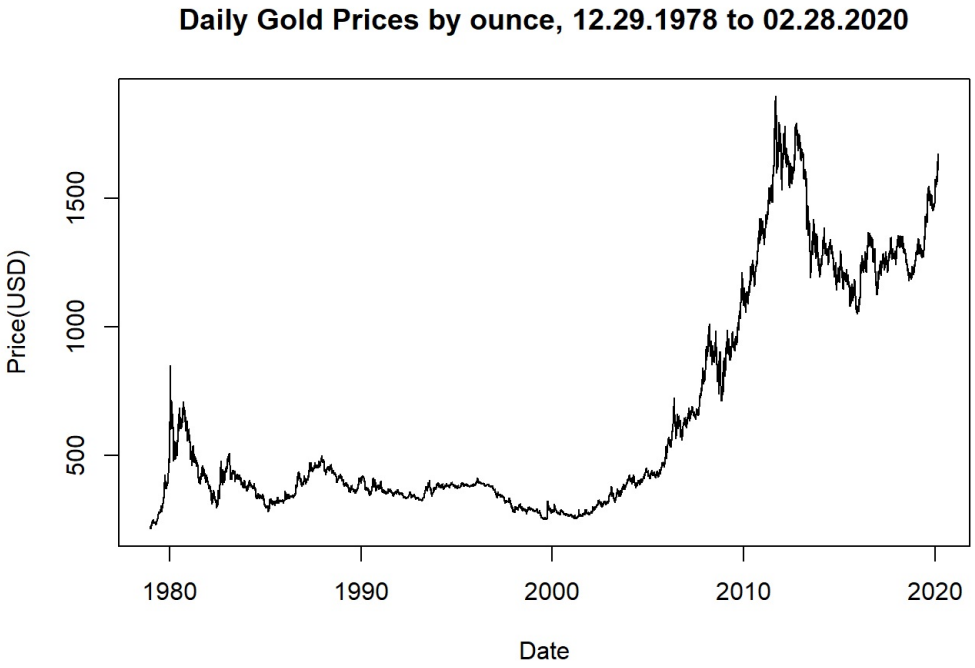
So I skip this model and proceed with some other methods.

ARIMA & SARIMA

In this section, I tried to fit an (S)ARIMA model to forecast.

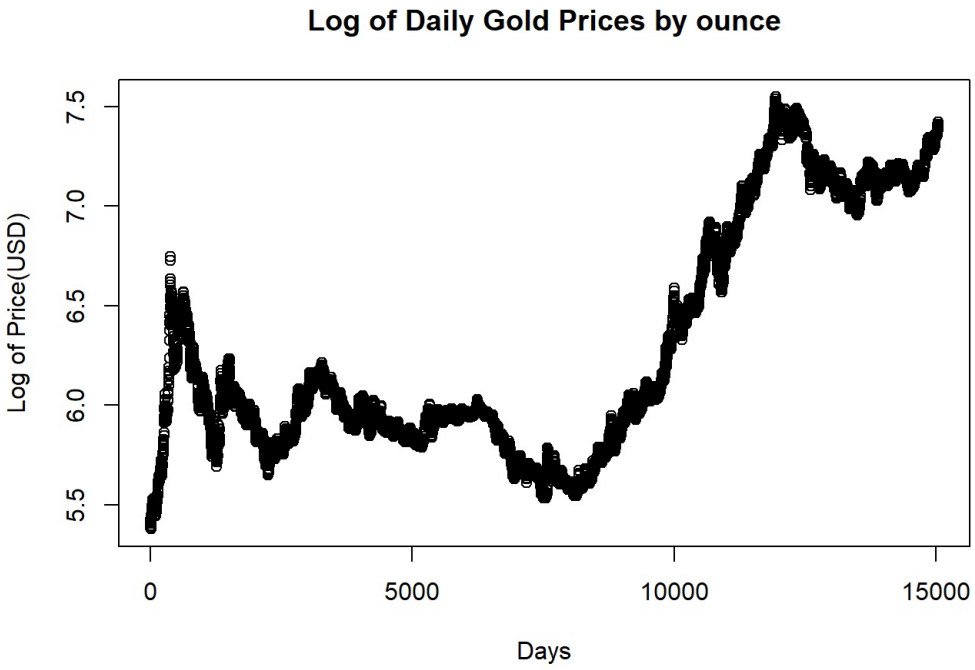
But before fitting models I wanted to see and to try to understand if there is a trend or seasonality in data;

```
plot(df1, main="Daily Gold Prices by ounce, 12.29.1978 to 02.28.2020",
     ylab = "Price(USD)", xlab = "Date", type="l")
```



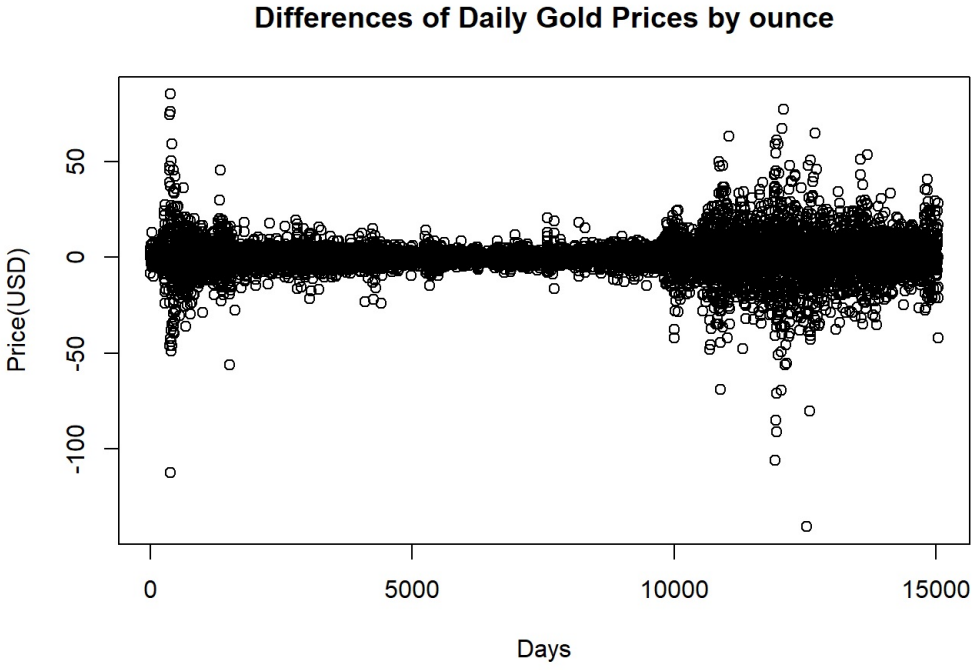
It seems there is instability in variance and a clear upward trend on data so first I will take logarithm to stabilize variance.

```
plot(log(df1$Ounce_Price), main = "Log of Daily Gold Prices by ounce",
     ylab = "Log of Price(USD)", xlab = "Days")
```



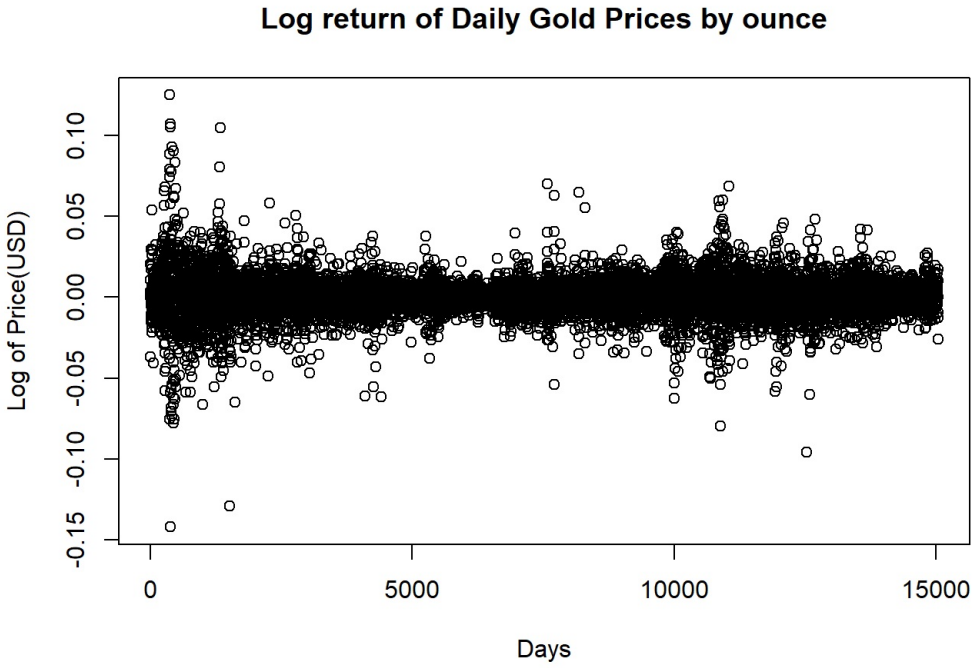
This time I tried to remove seasonality by taking difference of data.

```
plot(diff(df1$Ounce_Price), main = "Differences of Daily Gold Prices by ounce",
     ylab = "Price(USD)", xlab = "Days")
```



After trials, I took log-return of the dataset to take care of both seasonality and trend.

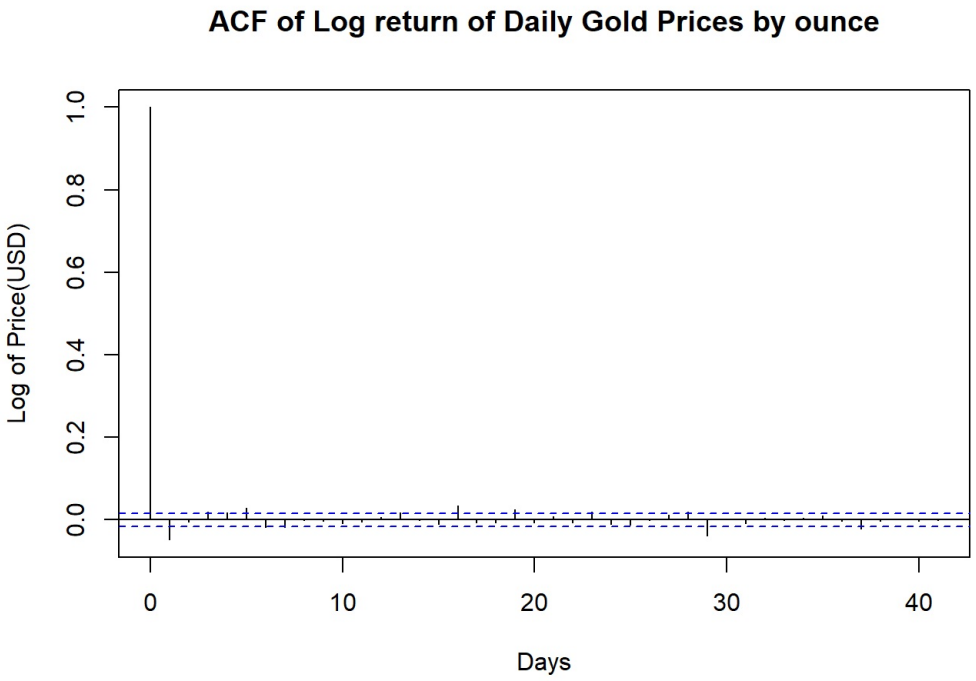
```
plot(diff(log(df1$Ounce_Price)), main = "Log return of Daily Gold Prices by ounce", ylab = "Log of Price(USD)", xl
ab = "Days")
```



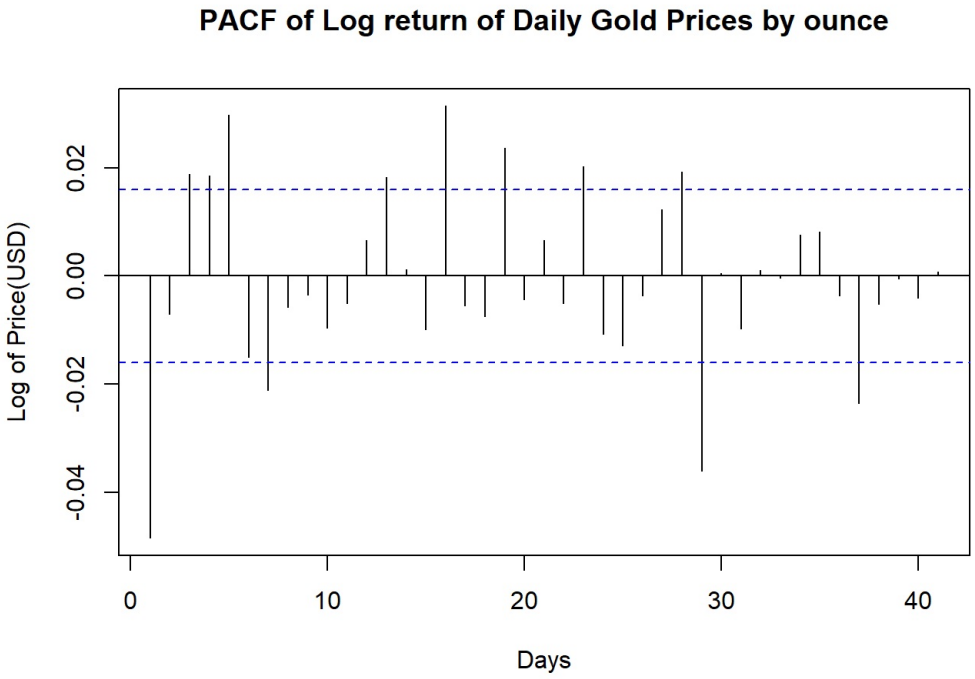
I also added several other transformation trials but this last graph gives the best result so far.

So, I decided to use log-return as my final transformation method. After this decision, I drawed Autocorrelation Function(ACF) and Partial Autocorrelation Function (PACF) to see if there are spikes jump over alpha level, which will help me to decide my AR and MA levels in my ARIMA model and also SAR and SMA levels as well for my Seasonal ARIMA models.

```
acf(diff(log(df1$Ounce_Price)), main = "ACF of Log return of Daily Gold Prices by ounce",
    ylab = "Log of Price(USD)", xlab = "Days")
```




```
pacf(diff(log(df1$Ounce_Price)), main = "PACF of Log return of Daily Gold Prices by ounce",
    ylab = "Log of Price(USD)", xlab = "Days")
```



There are clear cyclic trends in both graphs. As I mentioned above ACF suggests order of MA process(q), Seasonal MA process(Q) and PACF suggests order of AR process (p) and Seasonal AR process (P) for ARIMA and SARIMA.

As more clear explanation, for example when taking into account ACF plot, if there is a clear spike(s) in beginning lags it refers to the order of MA(q) and if there is a clear spike(s) after several lags from beginning it gives an idea about the order of Seasonal MA(Q) process.

This appoach also works same with PACF plot to determine order of AR(p) and SAR(P) process.

After all, I tried to find best fitted model in the light of these informations. I created a for loop and since I took non-seasonal difference but not seasonal difference of observations, I manually wrote down d=1 and D=0 when creating the loop and tried best options for both AR(p), MA(q) and Seasonal-AR(P),Seasonal-MA(Q). And I also added if condition to keep sum of processes (p+q+d+P+Q+D) equal to or under 6, I put that condition to find simplier fitted model rather than complex ones in other words aim to obey parcimony principle rule.

```
d=1
D=0

for(p in 0:5){
  for(q in 0:4){
    for(P in 0:6){
      for(Q in 0:5){
        if( p+q+P+Q+d+D <= 6){ # parcimony principle
          model <- arima(x= log(df1$Ounce_Price), order=c(p,d,q),
                         seasonal = list(order=c(P,D,Q), frequency = 1))
          sse <- sum(resid(model)^2)
          order <- paste(p,d,q,P,D,Q)
          pval<-Box.test(model$residuals, lag=log(length(model$residuals)))
          cat(p,d,q,P,D,Q,365, 'AIC=', model$aic, ' SSE=',sse,' p-VALUE=',
              pval$p.value,'\n')
        }
      }
    }
  }
}
```

```
## 0 1 0 0 0 0 365 AIC= -95803.35  SSE= 1.50473  p-VALUE= 1.01415e-10
## 0 1 0 0 0 1 365 AIC= -95836.84  SSE= 1.501183  p-VALUE= 0.0001377116
## 0 1 0 0 0 2 365 AIC= -95834.92  SSE= 1.501175  p-VALUE= 0.0001460444
## 0 1 0 0 0 3 365 AIC= -95840.22  SSE= 1.500446  p-VALUE= 0.001781602
## 0 1 0 0 0 4 365 AIC= -95844.41  SSE= 1.499829  p-VALUE= 0.0153521
## 0 1 0 0 0 5 365 AIC= -95853.12  SSE= 1.49876  p-VALUE= 0.3057976
## 0 1 0 1 0 0 365 AIC= -95836.5  SSE= 1.501216  p-VALUE= 0.0001190374
## 0 1 0 1 0 1 365 AIC= -95834.71  SSE= 1.501196  p-VALUE= 0.0001300751
## 0 1 0 1 0 2 365 AIC= -95832.96  SSE= 1.501171  p-VALUE= 0.0001488464
## 0 1 0 1 0 3 365 AIC= -95844.54  SSE= 1.499816  p-VALUE= 0.01542154
## 0 1 0 1 0 4 365 AIC= -95845.47  SSE= 1.499523  p-VALUE= 0.0388503
## 0 1 0 2 0 0 365 AIC= -95835.21  SSE= 1.501146  p-VALUE= 0.000163732
## 0 1 0 2 0 1 365 AIC= -95833.11  SSE= 1.501156  p-VALUE= 0.0001582938
## 0 1 0 2 0 2 365 AIC= -95834.33  SSE= 1.500835  p-VALUE= 0.0004937969
## 0 1 0 2 0 3 365 AIC= -95840.28  SSE= 1.500041  p-VALUE= 0.006696673
## 0 1 0 3 0 0 365 AIC= -95838.64  SSE= 1.500604  p-VALUE= 0.001008024
## 0 1 0 3 0 1 365 AIC= -95842.97  SSE= 1.499972  p-VALUE= 0.00898936
```

```
## 0 1 0 3 0 2 365 AIC= -95838.68  SSE= 1.500201  p-VALUE= 0.003833601
## 0 1 0 4 0 0 365 AIC= -95841.92  SSE= 1.500077  p-VALUE= 0.006556447
## 0 1 0 4 0 1 365 AIC= -95843.82  SSE= 1.499688  p-VALUE= 0.0232791
## 0 1 0 5 0 0 365 AIC= -95853.45  SSE= 1.498727  p-VALUE= 0.3437408
## 0 1 1 0 0 0 365 AIC= -95836.84  SSE= 1.501183  p-VALUE= 0.0001377116
## 0 1 1 0 0 1 365 AIC= -95834.79  SSE= 1.501188  p-VALUE= 0.0001343345
## 0 1 1 0 0 2 365 AIC= -95832.97  SSE= 1.50117  p-VALUE= 0.0001494055
## 0 1 1 0 0 3 365 AIC= -95839.9  SSE= 1.500279  p-VALUE= 0.003256196
## 0 1 1 0 0 4 365 AIC= -95853.12  SSE= 1.49876  p-VALUE= 0.3051371
## 0 1 1 1 0 0 365 AIC= -95834.71  SSE= 1.501196  p-VALUE= 0.0001300751
## 0 1 1 1 0 1 365 AIC= -95832.73  SSE= 1.501194  p-VALUE= 0.0001311456
## 0 1 1 1 0 2 365 AIC= -95830.99  SSE= 1.501168  p-VALUE= 0.0001504628
## 0 1 1 1 0 3 365 AIC= -95843.4  SSE= 1.499729  p-VALUE= 0.02045031
## 0 1 1 2 0 0 365 AIC= -95833.11  SSE= 1.501156  p-VALUE= 0.0001582938
## 0 1 1 2 0 1 365 AIC= -95831.09  SSE= 1.501158  p-VALUE= 0.0001568134
## 0 1 1 2 0 2 365 AIC= -95843.85  SSE= 1.499685  p-VALUE= 0.0252771
## 0 1 1 3 0 0 365 AIC= -95842.97  SSE= 1.499972  p-VALUE= 0.00898936
## 0 1 1 3 0 1 365 AIC= -95842.95  SSE= 1.499775  p-VALUE= 0.01746349
## 0 1 1 4 0 0 365 AIC= -95843.82  SSE= 1.499688  p-VALUE= 0.0232791
## 0 1 2 0 0 0 365 AIC= -95834.92  SSE= 1.501175  p-VALUE= 0.0001460444
## 0 1 2 0 0 1 365 AIC= -95832.97  SSE= 1.50117  p-VALUE= 0.0001494055
## 0 1 2 0 0 2 365 AIC= -95830.98  SSE= 1.501169  p-VALUE= 0.0001492919
## 0 1 2 0 0 3 365 AIC= -95853.11  SSE= 1.498761  p-VALUE= 0.3053882
## 0 1 2 1 0 0 365 AIC= -95832.96  SSE= 1.501171  p-VALUE= 0.0001488464
## 0 1 2 1 0 1 365 AIC= -95830.99  SSE= 1.501168  p-VALUE= 0.0001504628
## 0 1 2 1 0 2 365 AIC= -95828.99  SSE= 1.501168  p-VALUE= 0.0001505077
## 0 1 2 2 0 0 365 AIC= -95834.33  SSE= 1.500835  p-VALUE= 0.0004937969
## 0 1 2 2 0 1 365 AIC= -95843.85  SSE= 1.499685  p-VALUE= 0.02527748
## 0 1 2 3 0 0 365 AIC= -95838.68  SSE= 1.500201  p-VALUE= 0.003833601
## 0 1 3 0 0 0 365 AIC= -95840.22  SSE= 1.500446  p-VALUE= 0.001781602
## 0 1 3 0 0 1 365 AIC= -95839.9  SSE= 1.500279  p-VALUE= 0.003256196
## 0 1 3 0 0 2 365 AIC= -95853.11  SSE= 1.498761  p-VALUE= 0.3053882
## 0 1 3 1 0 0 365 AIC= -95844.54  SSE= 1.499816  p-VALUE= 0.01542154
## 0 1 3 1 0 1 365 AIC= -95843.4  SSE= 1.499729  p-VALUE= 0.02045031
## 0 1 3 2 0 0 365 AIC= -95840.28  SSE= 1.500041  p-VALUE= 0.006696673
## 0 1 4 0 0 0 365 AIC= -95844.41  SSE= 1.499829  p-VALUE= 0.0153521
## 0 1 4 0 0 1 365 AIC= -95853.12  SSE= 1.49876  p-VALUE= 0.3051381
## 0 1 4 1 0 0 365 AIC= -95845.47  SSE= 1.499523  p-VALUE= 0.0388503
## 1 1 0 0 0 0 365 AIC= -95836.5  SSE= 1.501216  p-VALUE= 0.0001190374
## 1 1 0 0 0 1 365 AIC= -95834.71  SSE= 1.501196  p-VALUE= 0.0001300751
## 1 1 0 0 0 2 365 AIC= -95832.96  SSE= 1.501171  p-VALUE= 0.0001488464
## 1 1 0 0 0 3 365 AIC= -95844.54  SSE= 1.499816  p-VALUE= 0.01542154
## 1 1 0 0 0 4 365 AIC= -95845.47  SSE= 1.499523  p-VALUE= 0.0388503
## 1 1 0 1 0 0 365 AIC= -95834.62  SSE= 1.501205  p-VALUE= 0.0001248563
## 1 1 0 1 0 1 365 AIC= -95832.69  SSE= 1.501197  p-VALUE= 0.0001290157
## 1 1 0 1 0 2 365 AIC= -95830.99  SSE= 1.501168  p-VALUE= 0.0001502445
## 1 1 0 1 0 3 365 AIC= -95844.25  SSE= 1.499645  p-VALUE= 0.0269792
## 1 1 0 2 0 0 365 AIC= -95833.12  SSE= 1.501155  p-VALUE= 0.0001587021
## 1 1 0 2 0 1 365 AIC= -95831.09  SSE= 1.501158  p-VALUE= 0.000156941
## 1 1 0 2 0 2 365 AIC= -95845.03  SSE= 1.499567  p-VALUE= 0.03482763
## 1 1 0 3 0 0 365 AIC= -95841.91  SSE= 1.500078  p-VALUE= 0.006551328
## 1 1 0 3 0 1 365 AIC= -95843.78  SSE= 1.499692  p-VALUE= 0.02308623
## 1 1 0 4 0 0 365 AIC= -95842.72  SSE= 1.499798  p-VALUE= 0.01665575
## 1 1 1 0 0 0 365 AIC= -95834.71  SSE= 1.501196  p-VALUE= 0.0001300751
## 1 1 1 0 0 1 365 AIC= -95832.73  SSE= 1.501194  p-VALUE= 0.0001311456
## 1 1 1 0 0 2 365 AIC= -95830.99  SSE= 1.501168  p-VALUE= 0.0001504628
## 1 1 1 0 0 3 365 AIC= -95843.4  SSE= 1.499729  p-VALUE= 0.02045031
## 1 1 1 1 0 0 365 AIC= -95832.69  SSE= 1.501197  p-VALUE= 0.0001290157
## 1 1 1 1 0 1 365 AIC= -95830.71  SSE= 1.501196  p-VALUE= 0.000130047
## 1 1 1 1 0 2 365 AIC= -95829  SSE= 1.501167  p-VALUE= 0.0001510066
## 1 1 1 2 0 0 365 AIC= -95831.09  SSE= 1.501158  p-VALUE= 0.000156941
## 1 1 1 2 0 1 365 AIC= -95829.08  SSE= 1.501159  p-VALUE= 0.0001560564
## 1 1 1 3 0 0 365 AIC= -95843.78  SSE= 1.499692  p-VALUE= 0.02308623
## 1 1 2 0 0 0 365 AIC= -95832.96  SSE= 1.501171  p-VALUE= 0.0001488464
## 1 1 2 0 0 1 365 AIC= -95830.99  SSE= 1.501168  p-VALUE= 0.0001504628
## 1 1 2 0 0 2 365 AIC= -95828.99  SSE= 1.501168  p-VALUE= 0.0001505077
## 1 1 2 1 0 0 365 AIC= -95830.99  SSE= 1.501168  p-VALUE= 0.0001502445
## 1 1 2 1 0 1 365 AIC= -95829  SSE= 1.501167  p-VALUE= 0.0001510066
## 1 1 2 2 0 0 365 AIC= -95845.03  SSE= 1.499567  p-VALUE= 0.03479756
## 1 1 3 0 0 0 365 AIC= -95844.54  SSE= 1.499816  p-VALUE= 0.01542154
## 1 1 3 0 0 1 365 AIC= -95843.4  SSE= 1.499729  p-VALUE= 0.02045031
## 1 1 3 1 0 0 365 AIC= -95844.25  SSE= 1.499645  p-VALUE= 0.0269792
## 1 1 4 0 0 0 365 AIC= -95845.47  SSE= 1.499523  p-VALUE= 0.0388503
## 2 1 0 0 0 0 365 AIC= -95835.21  SSE= 1.501146  p-VALUE= 0.000163732
## 2 1 0 0 0 1 365 AIC= -95833.11  SSE= 1.501156  p-VALUE= 0.0001582938
## 2 1 0 0 0 2 365 AIC= -95834.33  SSE= 1.500835  p-VALUE= 0.0004937969
## 2 1 0 0 0 3 365 AIC= -95840.28  SSE= 1.500041  p-VALUE= 0.006696673
## 2 1 0 1 0 0 365 AIC= -95833.12  SSE= 1.501155  p-VALUE= 0.0001587021
## 2 1 0 1 0 1 365 AIC= -95831.09  SSE= 1.501158  p-VALUE= 0.000156941
```

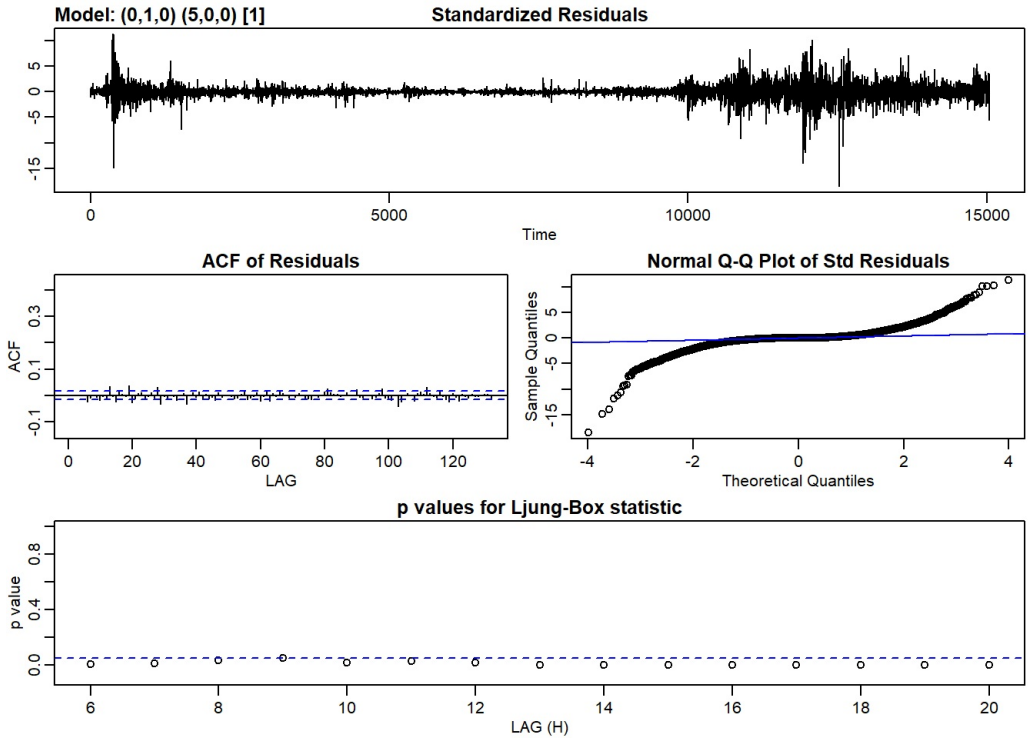
```
## 2 1 0 1 0 2 365 AIC= -95845.03  SSE= 1.499567  p-VALUE= 0.03478656
## 2 1 0 2 0 0 365 AIC= -95831.12  SSE= 1.501155  p-VALUE= 0.0001582195
## 2 1 0 2 0 1 365 AIC= -95829.09  SSE= 1.501158  p-VALUE= 0.0001567065
## 2 1 0 3 0 0 365 AIC= -95853.45  SSE= 1.498727  p-VALUE= 0.3436824
## 2 1 1 0 0 0 365 AIC= -95833.11  SSE= 1.501156  p-VALUE= 0.0001582938
## 2 1 1 0 0 1 365 AIC= -95831.09  SSE= 1.501158  p-VALUE= 0.0001568134
## 2 1 1 0 0 2 365 AIC= -95843.85  SSE= 1.499685  p-VALUE= 0.02530235
## 2 1 1 1 0 0 365 AIC= -95831.09  SSE= 1.501158  p-VALUE= 0.000156941
## 2 1 1 1 0 1 365 AIC= -95829.08  SSE= 1.501159  p-VALUE= 0.0001560564
## 2 1 1 2 0 0 365 AIC= -95829.09  SSE= 1.501158  p-VALUE= 0.0001567065
## 2 1 2 0 0 0 365 AIC= -95834.33  SSE= 1.500835  p-VALUE= 0.0004937969
## 2 1 2 0 0 1 365 AIC= -95843.85  SSE= 1.499685  p-VALUE= 0.02529395
## 2 1 2 1 0 0 365 AIC= -95845.03  SSE= 1.499568  p-VALUE= 0.03473531
## 2 1 3 0 0 0 365 AIC= -95840.28  SSE= 1.500041  p-VALUE= 0.006696673
## 3 1 0 0 0 0 365 AIC= -95838.64  SSE= 1.500604  p-VALUE= 0.001008024
## 3 1 0 0 0 1 365 AIC= -95842.97  SSE= 1.499972  p-VALUE= 0.00898936
## 3 1 0 0 0 2 365 AIC= -95838.68  SSE= 1.500201  p-VALUE= 0.003833601
## 3 1 0 1 0 0 365 AIC= -95841.91  SSE= 1.500078  p-VALUE= 0.006551328
## 3 1 0 1 0 1 365 AIC= -95843.78  SSE= 1.499692  p-VALUE= 0.02308623
## 3 1 0 2 0 0 365 AIC= -95853.45  SSE= 1.498727  p-VALUE= 0.3436823
## 3 1 1 0 0 0 365 AIC= -95842.97  SSE= 1.499972  p-VALUE= 0.00898936
## 3 1 1 0 0 1 365 AIC= -95842.95  SSE= 1.499775  p-VALUE= 0.01746349
## 3 1 1 1 0 0 365 AIC= -95843.78  SSE= 1.499692  p-VALUE= 0.02308623
## 3 1 2 0 0 0 365 AIC= -95838.68  SSE= 1.500201  p-VALUE= 0.003833601
## 4 1 0 0 0 0 365 AIC= -95841.92  SSE= 1.500077  p-VALUE= 0.006556447
## 4 1 0 0 0 1 365 AIC= -95843.82  SSE= 1.499688  p-VALUE= 0.0232791
## 4 1 0 1 0 0 365 AIC= -95842.72  SSE= 1.499798  p-VALUE= 0.01665575
## 4 1 1 0 0 0 365 AIC= -95843.82  SSE= 1.499688  p-VALUE= 0.0232791
## 5 1 0 0 0 0 365 AIC= -95853.45  SSE= 1.498727  p-VALUE= 0.3437408
```

From printed results it can be seen that (0 1 0 5 0 0 1) model gives lowest AIC score among others with AIC= -95853.45 and SSE= 1.498727 and also p-VALUE= 0.3437408. This high P-value refers to the Ljung-Box statistic and allow me to reject null hypothesis in which proves that there is no any significant autocorrelation left in residuals. Same AIC level also occured in (2,1,0,3,0,0) level, but I will use first was because it is simplier.

This time I will use ````sarima()`````function to check if the assumptions met with this AR and MA values.

```
sarima(df1$Ounce_Price, 0,1,0,5,0,0,1)
```

```
## initial  value 2.023502
## iter   2 value 2.022654
## iter   3 value 2.022652
## iter   3 value 2.022652
## iter   3 value 2.022652
## final  value 2.022652
## converged
## initial  value 2.022526
## iter   1 value 2.022526
## final  value 2.022526
## converged
```

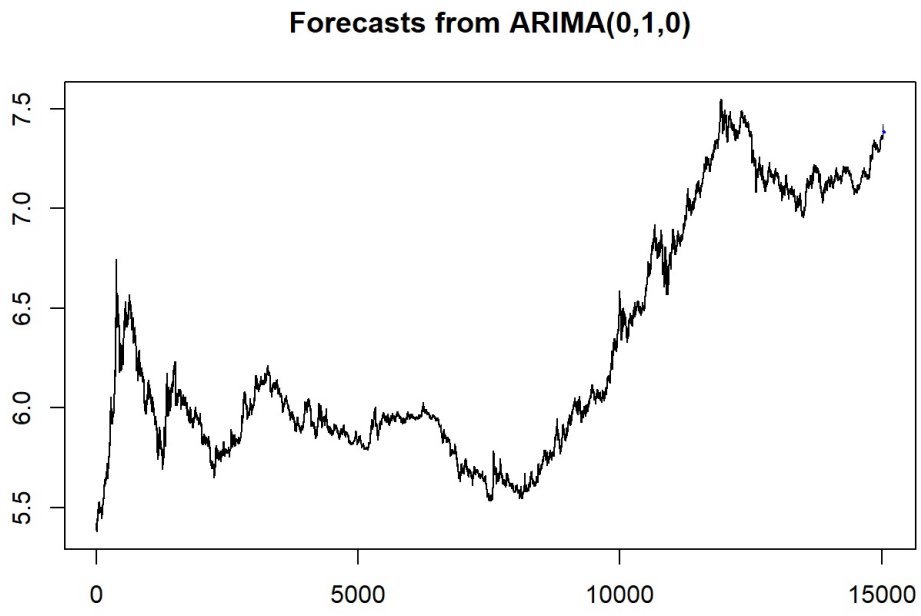


```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), xreg = constant, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##          sar1     sar2    sar3    sar4    sar5  constant
##       -0.0261  -0.0128  0.0283  0.0053  0.0058    0.0920
## s.e.   0.0082   0.0082  0.0082  0.0082  0.0082    0.0617
##
## sigma^2 estimated as 57.11:  log likelihood = -51745.87,  aic = 103505.7
##
## $degrees_of_freedom
## [1] 15030
##
## $ttable
##          Estimate     SE t.value p.value
## sar1      -0.0261 0.0082 -3.1990  0.0014
## sar2      -0.0128 0.0082 -1.5617  0.1184
## sar3       0.0283 0.0082  3.4625  0.0005
## sar4       0.0053 0.0082  0.6525  0.5141
## sar5       0.0058 0.0082  0.7126  0.4761
## constant   0.0920 0.0617  1.4927  0.1355
##
## $AIC
## [1] 6.883861
##
## $AICc
## [1] 6.883861
##
## $BIC
## [1] 6.887407
```

From produced graphs above, although I expected a white noise in Residual distribution, it did not seem that way. Also from QQ plot I cannot say that residuals distibuted normally. Lastly from ACF of residuals I can still observe some spikes above alpha and p values mostly positioned below alpha, so I rejected null hypothesis that there is no any significant autocorrelation left in the residuals.

Long story short, this is not a good fitted model to this data. But I also wanted to see how this model forecasts. So I fitted a model generated with the (2 1 1 0 0 2) values and forecast 50 day by using this model. However the results was not enlighting as expected. It predicts same price for all 50 of predicted days. The forecast graph and results can be seen below.

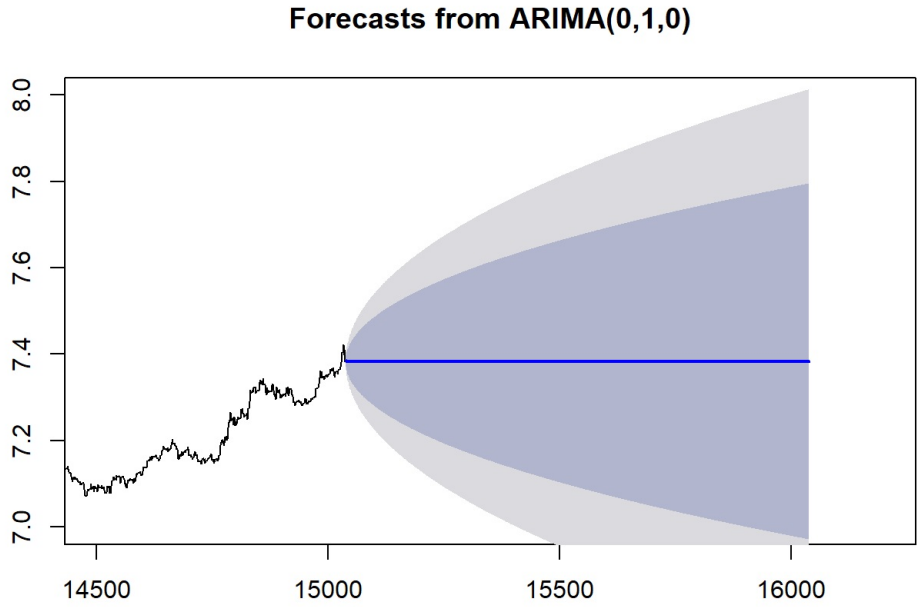
```
modelx <- arima(x=log(df1$Ounce_Price), order = c(0,1,0),
                seasonal = list(order=c(0,0,5), period=1))
plot(forecast(modelx))
```




```
forecast(modelx, 50)
```

```
##       Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 15038       7.385075 7.372281 7.397870 7.365508 7.404643
## 15039       7.384802 7.367141 7.402463 7.357792 7.411812
## 15040       7.384138 7.362704 7.405572 7.351358 7.416918
## 15041       7.383904 7.359126 7.408681 7.346010 7.421798
## 15042       7.383215 7.355383 7.411048 7.340649 7.425781
## 15043       7.383215 7.352486 7.413944 7.336219 7.430211
## 15044       7.383215 7.349840 7.416591 7.332172 7.434258
## 15045       7.383215 7.347389 7.419042 7.328424 7.438007
## 15046       7.383215 7.345095 7.421336 7.324915 7.441515
## 15047       7.383215 7.342932 7.423499 7.321607 7.444824
## 15048       7.383215 7.340878 7.425552 7.318467 7.447964
## 15049       7.383215 7.338920 7.427510 7.315472 7.450959
## 15050       7.383215 7.337045 7.429386 7.312604 7.453826
## 15051       7.383215 7.335243 7.431187 7.309849 7.456582
## 15052       7.383215 7.333507 7.432924 7.307193 7.459238
## 15053       7.383215 7.331829 7.434602 7.304627 7.461804
## 15054       7.383215 7.330204 7.436227 7.302142 7.464289
## 15055       7.383215 7.328628 7.437803 7.299731 7.466700
## 15056       7.383215 7.327095 7.439335 7.297387 7.469044
## 15057       7.383215 7.325604 7.440827 7.295106 7.471325
## 15058       7.383215 7.324150 7.442281 7.292883 7.473548
## 15059       7.383215 7.322731 7.443700 7.290713 7.475718
## 15060       7.383215 7.321345 7.445086 7.288592 7.477838
## 15061       7.383215 7.319989 7.446442 7.286519 7.479912
## 15062       7.383215 7.318661 7.447769 7.284488 7.481942
## 15063       7.383215 7.317361 7.449070 7.282499 7.483932
## 15064       7.383215 7.316085 7.450346 7.280548 7.485882
## 15065       7.383215 7.314833 7.451598 7.278634 7.487797
## 15066       7.383215 7.313604 7.452827 7.276754 7.489677
## 15067       7.383215 7.312396 7.454035 7.274907 7.491524
## 15068       7.383215 7.311208 7.455222 7.273090 7.493341
## 15069       7.383215 7.310040 7.456391 7.271303 7.495127
## 15070       7.383215 7.308890 7.457541 7.269545 7.496886
## 15071       7.383215 7.307757 7.458673 7.267812 7.498618
## 15072       7.383215 7.306642 7.459789 7.266106 7.500325
## 15073       7.383215 7.305542 7.460889 7.264424 7.502007
## 15074       7.383215 7.304458 7.461973 7.262766 7.503665
## 15075       7.383215 7.303388 7.463043 7.261130 7.505301
## 15076       7.383215 7.302333 7.464098 7.259516 7.506915
## 15077       7.383215 7.301291 7.465140 7.257922 7.508509
## 15078       7.383215 7.300262 7.466169 7.256349 7.510082
## 15079       7.383215 7.299246 7.467185 7.254795 7.511636
## 15080       7.383215 7.298242 7.468189 7.253259 7.513172
## 15081       7.383215 7.297249 7.469182 7.251741 7.514689
## 15082       7.383215 7.296268 7.470163 7.250241 7.516190
## 15083       7.383215 7.295298 7.471133 7.248758 7.517673
## 15084       7.383215 7.294339 7.472092 7.247290 7.519141
## 15085       7.383215 7.293389 7.473041 7.245838 7.520592
## 15086       7.383215 7.292450 7.473981 7.244402 7.522029
## 15087       7.383215 7.291520 7.474910 7.242980 7.523451
```

```
plot(forecast(modelx ,1000),xlim=c(14500,16200),ylim=c(7,8))
```

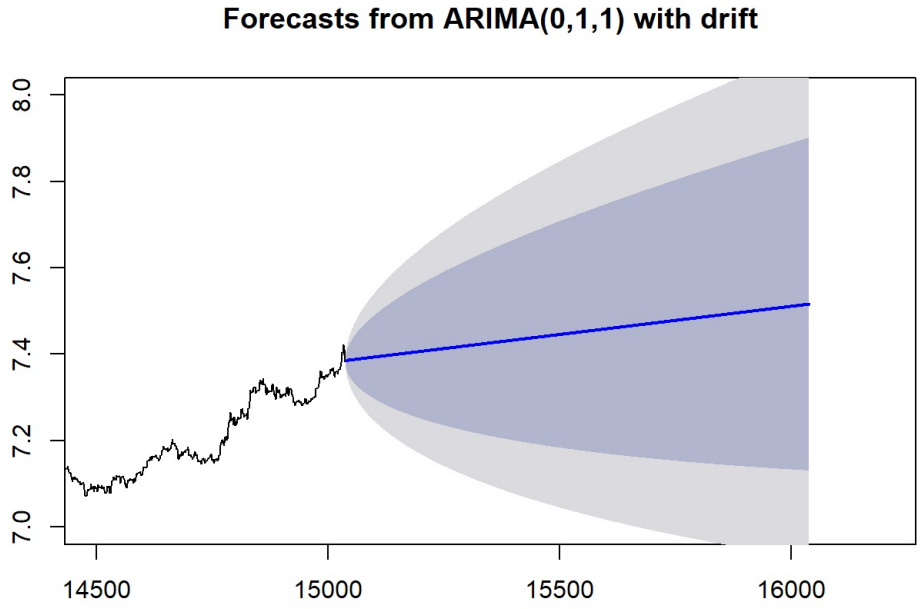


And then I tried `auto.arima()` function to forecast with the codes below;

```
summary(forecast(auto.arima(log(df1$Ounce_Price), seasonal = TRUE)))
```

```
##
## Forecast method: ARIMA(0,1,1) with drift
##
## Model Information:
## Series: log(df1$Ounce_Price)
## ARIMA(0,1,1) with drift
##
## Coefficients:
##           ma1  drift
##       -0.0489  1e-04
## s.e.   0.0082  1e-04
##
## sigma^2 estimated as 9.983e-05:  log likelihood=47921.84
## AIC=-95837.68   AICc=-95837.68   BIC=-95814.82
##
## Error measures:
##                       ME        RMSE         MAE           MPE       MAPE
## Training set 4.458047e-07 0.009990684 0.005445101 -0.0001622851 0.08671057
##                   MASE         ACF1
## Training set 1.011046 0.0001251136
##
## Forecasts:
##       Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 15038       7.385275 7.372470 7.398079 7.365691 7.404858
## 15039       7.385405 7.367734 7.403076 7.358379 7.412431
## 15040       7.385536 7.364075 7.406997 7.352714 7.418358
## 15041       7.385666 7.360991 7.410342 7.347928 7.423405
## 15042       7.385797 7.358280 7.413314 7.343713 7.427881
## 15043       7.385928 7.355836 7.416019 7.339906 7.431949
## 15044       7.386058 7.353596 7.418521 7.336411 7.435705
## 15045       7.386189 7.351517 7.420860 7.333163 7.439215
## 15046       7.386319 7.349571 7.423068 7.330118 7.442521
## 15047       7.386450 7.347736 7.425164 7.327242 7.445657
```

```
plot(forecast(auto.arima(log(df1$Ounce_Price), seasonal = TRUE),1000)
     ,xlim=c(14500,16200),ylim=c(7,8))
```



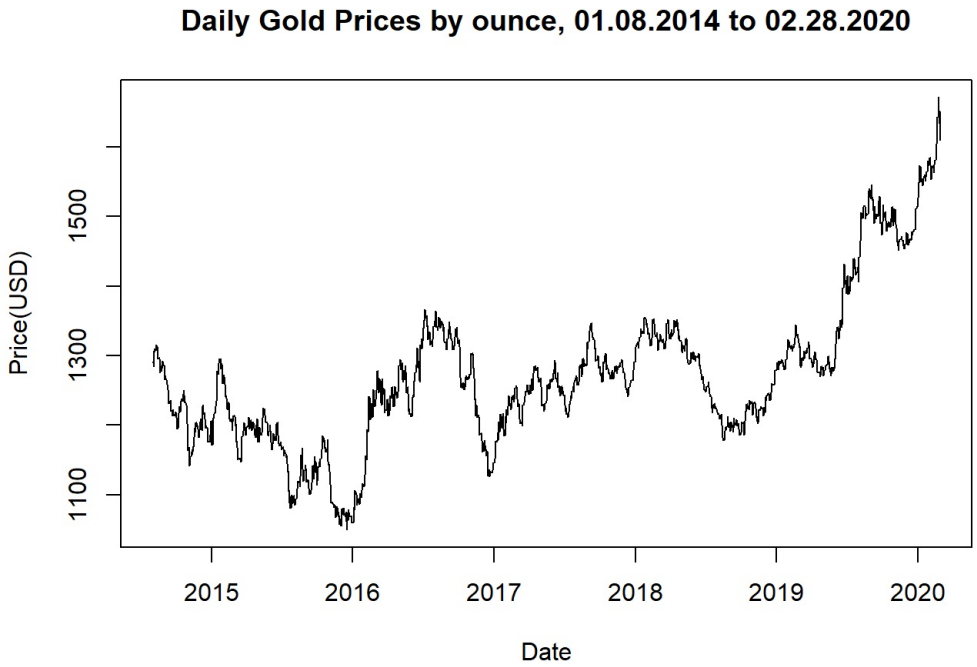
Auto arima function provides ARIMA(0,1,1) model with seasonal drift. But it provides AIC=-96073.66 , which is higher than my first model, which means worse produced results expected with this model. However its forecast results tend to increase constantly this time.

ARIMA & SARIMA with less data

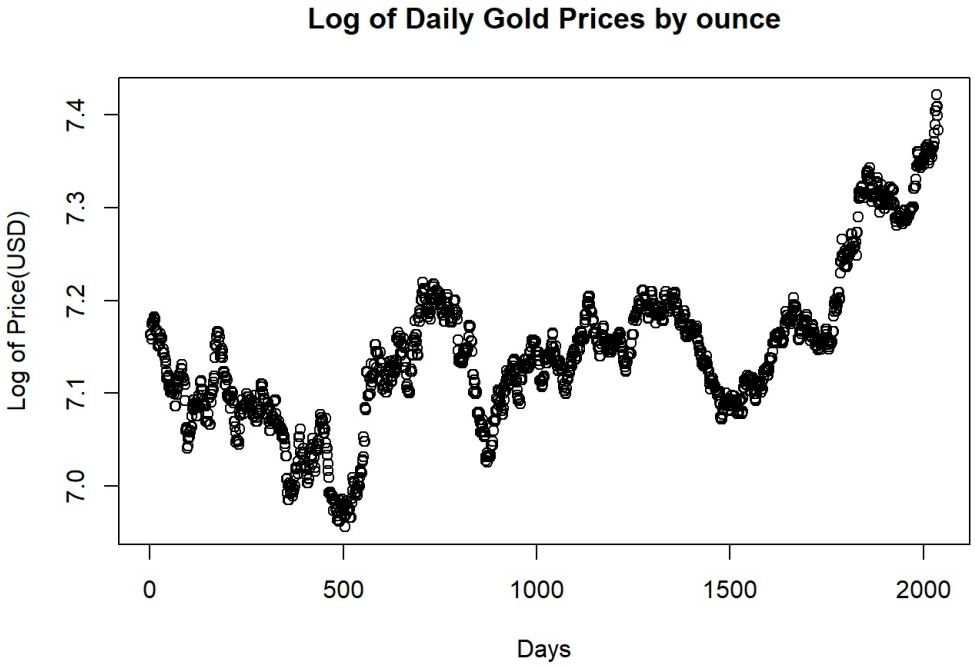
I thought this mess in my forecasts and model might be occured because of the length of dataset. So I decided to cut the dataset and took into account some part of it when fitting model and making predictions.

I cut data beginning from 2014 August till 2020 Feb just randomly. And did some visualizations again to understand data and decide the seasonality and trend.

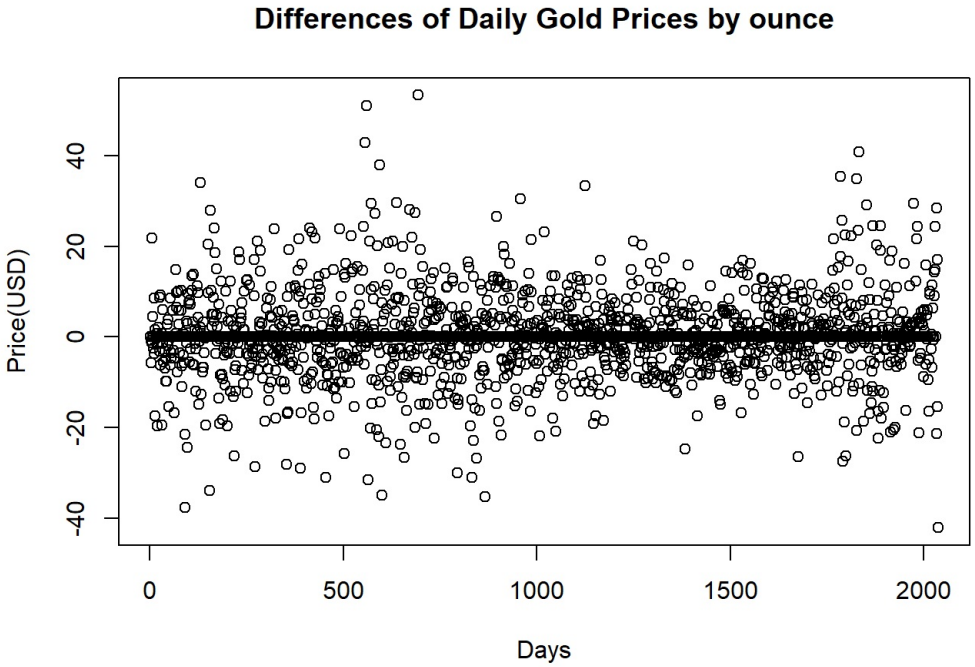
```
df2 <- df1[13000:nrow(df1),]


plot(df2, main="Daily Gold Prices by ounce, 01.08.2014 to 02.28.2020",
     ylab = "Price(USD)", xlab = "Date", type="l")
```



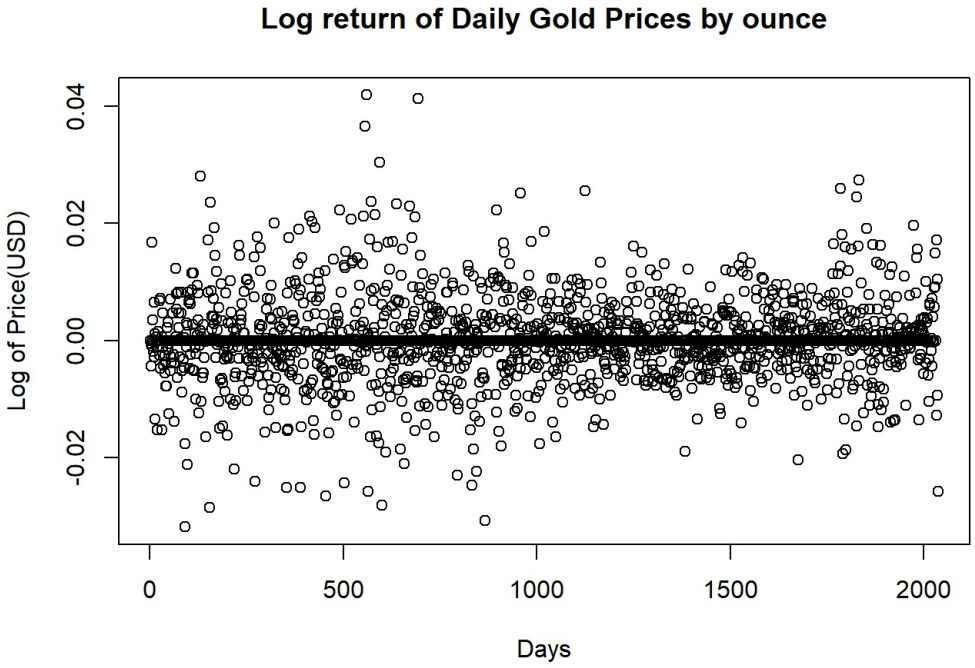

```
# it seems there is instability in variance and a clear upward trend on data
# so first I will take logarithm to stabilize variance
plot(log(df2$Ounce_Price), main = "Log of Daily Gold Prices by ounce",
     ylab = "Log of Price(USD)", xlab = "Days")
```




```
# now I will try to remove trend by taking difference of data
plot(diff(df2$Ounce_Price), main = "Differences of Daily Gold Prices by ounce",
     ylab = "Price(USD)", xlab = "Days")
```

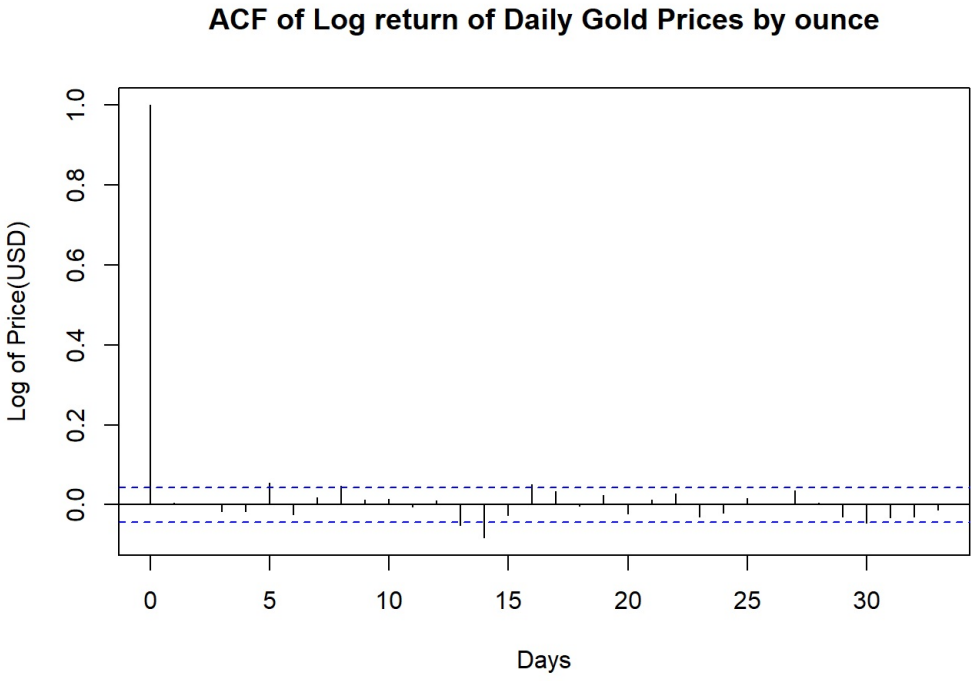



```
# it helped to remove trend as expected.
# Now I will take log-return of data
plot(diff(log(df2$Ounce_Price)), main = "Log return of Daily Gold Prices by ounce",
     ylab = "Log of Price(USD)", xlab = "Days")
```

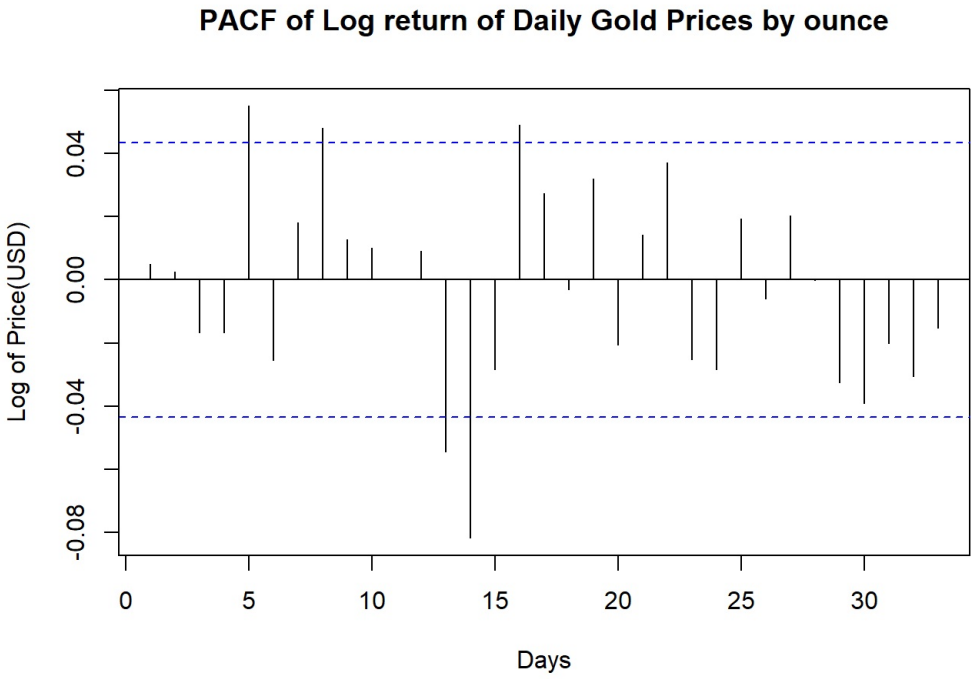


There is still an upward trend observed and also seasonality. So again I decided to proceed with implementing log return.

```
acf(diff(log(df2$Ounce_Price)), main = "ACF of Log return of Daily Gold Prices by ounce",
    ylab = "Log of Price(USD)", xlab = "Days")
```




```
#ø it seems there is a cyclic seasonality in the dataset fro ACF plot.

pacf(diff(log(df2$Ounce_Price)), main = "PACF of Log return of Daily Gold Prices by ounce",
     ylab = "Log of Price(USD)", xlab = "Days")
```



The ACF plot suggests MA(0) because there is not any spikes at the beginning and SMA(>=2) due to the spikes after several lag. The PACF refers that AR(0) again because there is not any spikes at the beginning but SAR(>=2) due to the spikes after several lag.

So as I decided to take non seasonal difference but not seasonal, d becomes 1 and D=0. Again I looped several options for p <= 1 q<= 1 and P <= 4 Q<=5 with if condition to keep their sum below 7. But this time I also determined frequency to the week with value of 7 instead of day as 1.

```
d=1
D=0

for(p in 0:1){
  for(q in 0:1){
    for(P in 0:4){
      for(Q in 0:5){
        if( p+q+P+Q+d+D <= 6){ # parcimony principle
          model <- arima(x= log(df2$Ounce_Price), order=c(p,d,q),
                         seasonal = list(order=c(P,D,Q), frequency = 7))
          sse <- sum(resid(model)^2)
          order <- paste(p,d,q,P,D,Q)
          pval<-Box.test(model$residuals, lag=log(length(model$residuals)))
          cat(p,d,q,P,D,Q,7, 'AIC=', model$aic, ' SSE=',sse,' p-VALUE=',
              pval$p.value,'\n')
        }
      }
    }
  }
}
```

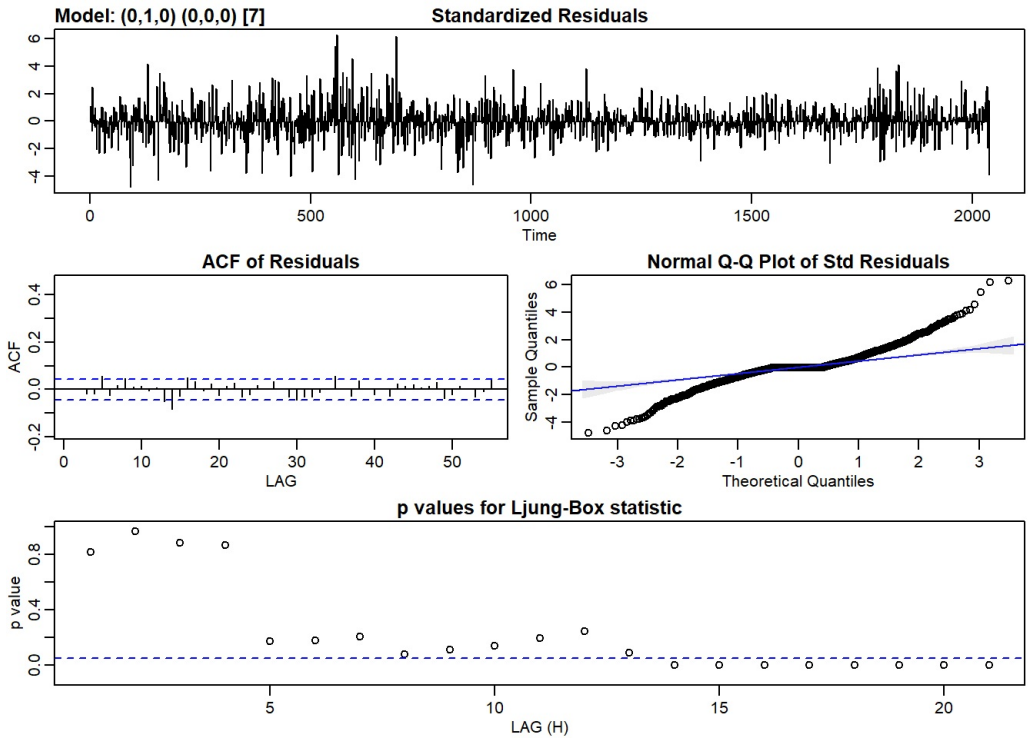
```
## 0 1 0 0 0 0 7 AIC= -14617.12  SSE= 0.09118448  p-VALUE= 0.2608558
## 0 1 0 0 0 1 7 AIC= -14615.18  SSE= 0.09118186  p-VALUE= 0.2584731
## 0 1 0 0 0 2 7 AIC= -14613.2  SSE= 0.09118105  p-VALUE= 0.2590366
## 0 1 0 0 0 3 7 AIC= -14611.8  SSE= 0.09115392  p-VALUE= 0.2925342
## 0 1 0 0 0 4 7 AIC= -14610.3  SSE= 0.09113158  p-VALUE= 0.3348895
## 0 1 0 0 0 5 7 AIC= -14615.1  SSE= 0.09082744  p-VALUE= 0.9784605
## 0 1 0 1 0 0 7 AIC= -14615.18  SSE= 0.09118185  p-VALUE= 0.258453
## 0 1 0 1 0 1 7 AIC= -14613.18  SSE= 0.09118186  p-VALUE= 0.2584664
## 0 1 0 1 0 2 7 AIC= -14611.2  SSE= 0.09118107  p-VALUE= 0.2590509
## 0 1 0 1 0 3 7 AIC= -14609.81  SSE= 0.09115376  p-VALUE= 0.2926626
## 0 1 0 1 0 4 7 AIC= -14611.88  SSE= 0.09097124  p-VALUE= 0.8552784
## 0 1 0 2 0 0 7 AIC= -14613.19  SSE= 0.09118117  p-VALUE= 0.25904
## 0 1 0 2 0 1 7 AIC= -14611.2  SSE= 0.09118115  p-VALUE= 0.2590383
## 0 1 0 2 0 2 7 AIC= -14609.2  SSE= 0.0911811  p-VALUE= 0.2590801
## 0 1 0 2 0 3 7 AIC= -14615.66  SSE= 0.09080153  p-VALUE= 0.5245891
## 0 1 0 3 0 0 7 AIC= -14611.75  SSE= 0.0911562  p-VALUE= 0.2876077
## 0 1 0 3 0 1 7 AIC= -14609.75  SSE= 0.0911563  p-VALUE= 0.2877375
## 0 1 0 3 0 2 7 AIC= -14615.68  SSE= 0.09080062  p-VALUE= 0.5201874
## 0 1 0 4 0 0 7 AIC= -14610.3  SSE= 0.09113154  p-VALUE= 0.3329942
## 0 1 0 4 0 1 7 AIC= -14610.03  SSE= 0.09105315  p-VALUE= 0.5986666
## 0 1 1 0 0 0 7 AIC= -14615.18  SSE= 0.09118186  p-VALUE= 0.2584731
## 0 1 1 0 0 1 7 AIC= -14613.18  SSE= 0.09118186  p-VALUE= 0.2584658
## 0 1 1 0 0 2 7 AIC= -14611.2  SSE= 0.09118107  p-VALUE= 0.2590495
## 0 1 1 0 0 3 7 AIC= -14609.81  SSE= 0.09115376  p-VALUE= 0.2926666
## 0 1 1 0 0 4 7 AIC= -14609.62  SSE= 0.09107263  p-VALUE= 0.4582706
## 0 1 1 1 0 0 7 AIC= -14613.18  SSE= 0.09118186  p-VALUE= 0.2584664
## 0 1 1 1 0 1 7 AIC= -14611.18  SSE= 0.09118186  p-VALUE= 0.2584693
## 0 1 1 1 0 2 7 AIC= -14609.2  SSE= 0.09118108  p-VALUE= 0.2590807
## 0 1 1 1 0 3 7 AIC= -14607.81  SSE= 0.09115369  p-VALUE= 0.2923721
## 0 1 1 2 0 0 7 AIC= -14611.2  SSE= 0.09118115  p-VALUE= 0.2590383
## 0 1 1 2 0 1 7 AIC= -14609.2  SSE= 0.09118114  p-VALUE= 0.2590515
## 0 1 1 2 0 2 7 AIC= -14607.2  SSE= 0.0911811  p-VALUE= 0.2591639
## 0 1 1 3 0 0 7 AIC= -14609.75  SSE= 0.0911563  p-VALUE= 0.2877375
## 0 1 1 3 0 1 7 AIC= -14607.75  SSE= 0.09115633  p-VALUE= 0.287212
## 0 1 1 4 0 0 7 AIC= -14610.03  SSE= 0.09105315  p-VALUE= 0.5986665
## 1 1 0 0 0 0 7 AIC= -14615.18  SSE= 0.09118185  p-VALUE= 0.258453
## 1 1 0 0 0 1 7 AIC= -14613.18  SSE= 0.09118186  p-VALUE= 0.2584664
## 1 1 0 0 0 2 7 AIC= -14611.2  SSE= 0.09118107  p-VALUE= 0.2590509
## 1 1 0 0 0 3 7 AIC= -14609.81  SSE= 0.09115376  p-VALUE= 0.2926626
## 1 1 0 0 0 4 7 AIC= -14611.88  SSE= 0.09097124  p-VALUE= 0.8552784
## 1 1 0 1 0 0 7 AIC= -14613.18  SSE= 0.09118185  p-VALUE= 0.2584609
## 1 1 0 1 0 1 7 AIC= -14611.18  SSE= 0.09118185  p-VALUE= 0.2584674
## 1 1 0 1 0 2 7 AIC= -14609.2  SSE= 0.09118108  p-VALUE= 0.2590283
## 1 1 0 1 0 3 7 AIC= -14607.81  SSE= 0.0911537  p-VALUE= 0.2923719
## 1 1 0 2 0 0 7 AIC= -14611.2  SSE= 0.09118114  p-VALUE= 0.2590442
## 1 1 0 2 0 1 7 AIC= -14609.2  SSE= 0.09118114  p-VALUE= 0.2590278
## 1 1 0 2 0 2 7 AIC= -14607.2  SSE= 0.0911811  p-VALUE= 0.259058
## 1 1 0 3 0 0 7 AIC= -14609.75  SSE= 0.0911563  p-VALUE= 0.2877021
## 1 1 0 3 0 1 7 AIC= -14611.58  SSE= 0.09098498  p-VALUE= 0.8699916
## 1 1 0 4 0 0 7 AIC= -14614.67  SSE= 0.09084671  p-VALUE= 0.9746426
## 1 1 1 0 0 0 7 AIC= -14613.18  SSE= 0.09118186  p-VALUE= 0.2584664
## 1 1 1 0 0 1 7 AIC= -14611.18  SSE= 0.09118186  p-VALUE= 0.2584693
## 1 1 1 0 0 2 7 AIC= -14609.2  SSE= 0.09118108  p-VALUE= 0.2590807
## 1 1 1 0 0 3 7 AIC= -14607.81  SSE= 0.09115369  p-VALUE= 0.2923721
## 1 1 1 1 0 0 7 AIC= -14611.18  SSE= 0.09118185  p-VALUE= 0.2584674
## 1 1 1 1 0 1 7 AIC= -14609.18  SSE= 0.09118186  p-VALUE= 0.2584718
## 1 1 1 1 0 2 7 AIC= -14607.2  SSE= 0.09118108  p-VALUE= 0.2590396
## 1 1 1 2 0 0 7 AIC= -14609.2  SSE= 0.09118114  p-VALUE= 0.2590278
## 1 1 1 2 0 1 7 AIC= -14607.2  SSE= 0.09118113  p-VALUE= 0.259003
## 1 1 1 3 0 0 7 AIC= -14611.58  SSE= 0.09098498  p-VALUE= 0.8699916
```

This time for loop provides ARIMA(0 1 0 0 0 0) as my best option with lowest AIC= -14617.12 and SSE= 0.09118448. And (0 1 0 2 0 3) as my second best option with AIC= -14615.66 SSE= 0.09080153.

I tried both to see if the assumptions are met.

```
sarima(log(df2$Ounce_Price), 0,1,0,0,0,0,7)
```

```
## initial  value -5.007464
## iter   1 value -5.007464
## final  value -5.007464
## converged
## initial  value -5.007464
## iter   1 value -5.007464
## final  value -5.007464
## converged
```



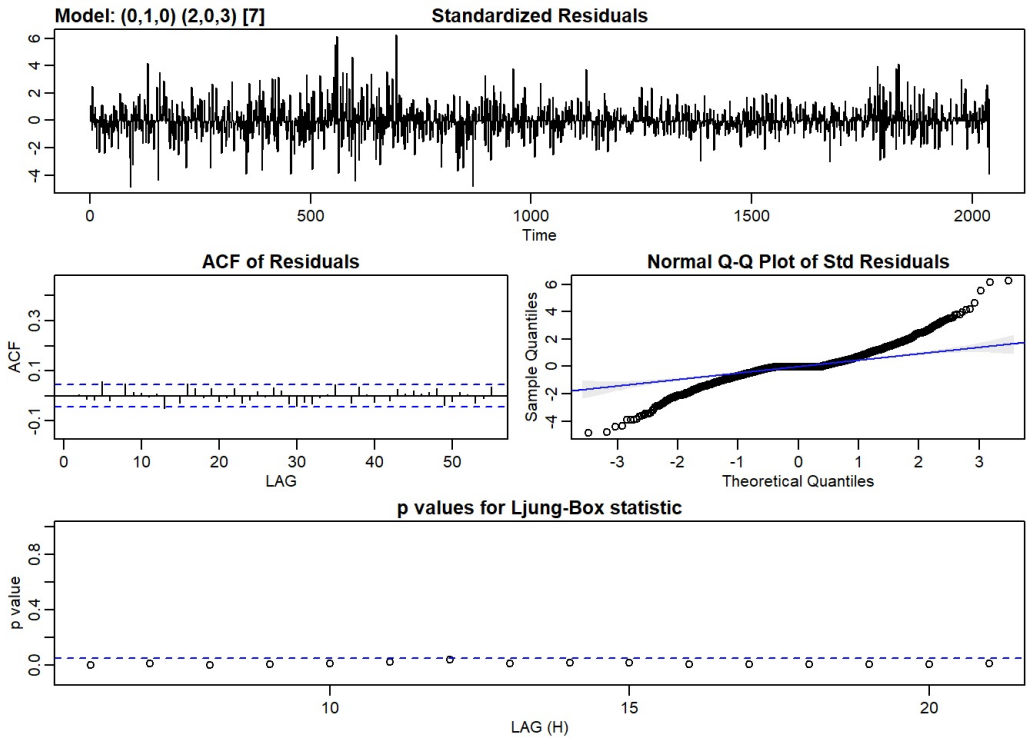

```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), xreg = constant, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##       constant
##          1e-04
## s.e.     1e-04
##
## sigma^2 estimated as 4.473e-05:  log likelihood = 7309.83,  aic = -14615.65
##
## $degrees_of_freedom
## [1] 2036
##
## $ttable
##          Estimate    SE t.value p.value
## constant    1e-04 1e-04  0.7226    0.47
##
## $AIC
## [1] -7.175088
##
## $AICc
## [1] -7.175087
##
## $BIC
## [1] -7.169571
```

From plots drawed residuals did not quite normally distributed, inspite of their distribution seems white. And also from p-values generated from Ljung-Box statistic there are still significant autocorrelation left in residuals.

```
sarima(log(df2$Ounce_Price), 0,1,0,2,0,3,7)
```

```
## initial  value -5.007039
## iter   2 value -5.008007
## iter   3 value -5.011251
## iter   4 value -5.011262
## iter   5 value -5.011274
## iter   6 value -5.011312
## iter   7 value -5.011394
## iter   8 value -5.011545
## iter   9 value -5.011738
## iter  10 value -5.011972
## iter  11 value -5.012013
## iter  12 value -5.012015
## iter  13 value -5.012016
## iter  14 value -5.012017
## iter  15 value -5.012017
```

```
## iter  16 value -5.012017
## iter  17 value -5.012017
## iter  18 value -5.012018
## iter  19 value -5.012019
## iter  20 value -5.012019
## iter  21 value -5.012019
## iter  22 value -5.012020
## iter  23 value -5.012020
## iter  24 value -5.012022
## iter  25 value -5.012027
## iter  26 value -5.012036
## iter  27 value -5.012044
## iter  28 value -5.012052
## iter  29 value -5.012056
## iter  30 value -5.012059
## iter  31 value -5.012060
## iter  32 value -5.012064
## iter  33 value -5.012071
## iter  34 value -5.012079
## iter  35 value -5.012084
## iter  36 value -5.012085
## iter  37 value -5.012086
## iter  38 value -5.012086
## iter  39 value -5.012087
## iter  40 value -5.012089
## iter  41 value -5.012090
## iter  42 value -5.012090
## iter  43 value -5.012090
## iter  44 value -5.012091
## iter  45 value -5.012093
## iter  46 value -5.012096
## iter  47 value -5.012100
## iter  48 value -5.012102
## iter  49 value -5.012104
## iter  50 value -5.012108
## iter  51 value -5.012116
## iter  52 value -5.012130
## iter  53 value -5.012144
## iter  54 value -5.012149
## iter  55 value -5.012151
## iter  56 value -5.012152
## iter  57 value -5.012154
## iter  58 value -5.012157
## iter  59 value -5.012160
## iter  60 value -5.012161
## iter  61 value -5.012161
## iter  61 value -5.012161
## iter  61 value -5.012161
## final  value -5.012161
## converged
## initial  value -5.011804
## iter   2 value -5.011816
## iter   3 value -5.011819
## iter   4 value -5.011819
## iter   5 value -5.011820
## iter   6 value -5.011823
## iter   7 value -5.011824
## iter   8 value -5.011824
## iter   9 value -5.011825
## iter  10 value -5.011826
## iter  11 value -5.011827
## iter  12 value -5.011829
## iter  13 value -5.011830
## iter  14 value -5.011830
## iter  14 value -5.011830
## final  value -5.011830
## converged
```

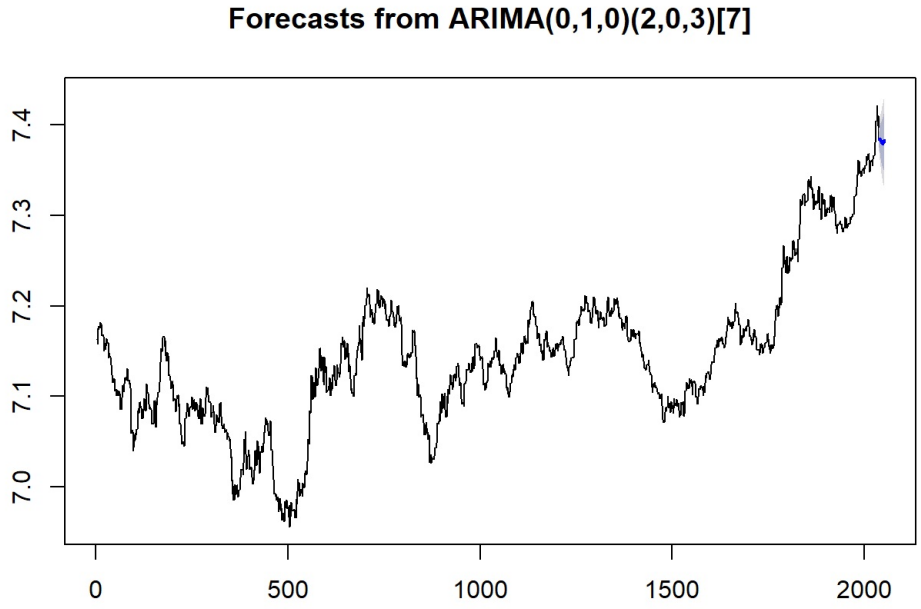



```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), xreg = constant, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##          sar1    sar2    sma1     sma2     sma3  constant
##       -0.3802  0.1040  0.4046  -0.1807  -0.0109     1e-04
## s.e.   0.7035  0.2826  0.7034   0.2940   0.0638     1e-04
##
## sigma^2 estimated as 4.434e-05:  log likelihood = 7318.72,  aic = -14623.44
##
## $degrees_of_freedom
## [1] 2031
##
## $ttable
##          Estimate     SE t.value p.value
## sar1      -0.3802 0.7035 -0.5403  0.5890
## sar2       0.1040 0.2826  0.3681  0.7128
## sma1       0.4046 0.7034  0.5752  0.5652
## sma2      -0.1807 0.2940 -0.6147  0.5388
## sma3      -0.0109 0.0638 -0.1713  0.8640
## constant   0.0001 0.0001  0.7461  0.4557
##
## $AIC
## [1] -7.17891
##
## $AICc
## [1] -7.17889
##
## $BIC
## [1] -7.1596
```

Again from plots drawed, results seems not changed that much.

I also wanted to see their forecast capabilities, so I drawed plots below for second model. It has a little fluctiation on forecasted prices.

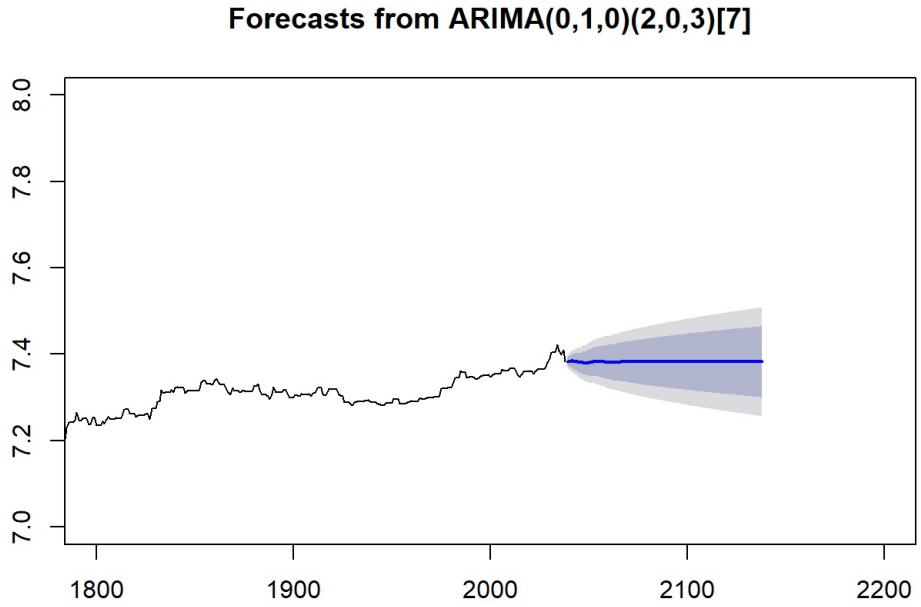
```
modelx <- arima(x=log(df2$Ounce_Price), order = c(0,1,0),
                seasonal = list(order=c(2,0,3), period=7))
plot(forecast(modelx))
```




```
forecast(modelx, 50)
```

```
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2039       7.383896 7.375361 7.392432 7.370843 7.396950
## 2040       7.383896 7.371825 7.395967 7.365435 7.402357
## 2041       7.384580 7.369796 7.399364 7.361970 7.407190
## 2042       7.384013 7.366943 7.401084 7.357906 7.410121
## 2043       7.382932 7.363846 7.402018 7.353743 7.412121
## 2044       7.382563 7.361656 7.403471 7.350588 7.414538
## 2045       7.380698 7.358116 7.403281 7.346161 7.415235
## 2046       7.380698 7.356482 7.404914 7.343663 7.417733
## 2047       7.380698 7.354953 7.406444 7.341324 7.420073
## 2048       7.379114 7.351925 7.406304 7.337532 7.420697
## 2049       7.380298 7.351737 7.408858 7.336618 7.423977
## 2050       7.381462 7.351593 7.411330 7.335781 7.427142
## 2051       7.380717 7.349595 7.411839 7.333120 7.428314
## 2052       7.383291 7.350965 7.415618 7.333852 7.432730
## 2053       7.383291 7.349983 7.416600 7.332350 7.434232
## 2054       7.383291 7.349029 7.417553 7.330892 7.435691
## 2055       7.383872 7.348682 7.419062 7.330054 7.437691
## 2056       7.383432 7.347338 7.419526 7.328231 7.438633
## 2057       7.382948 7.345972 7.419924 7.326398 7.439498
## 2058       7.383130 7.345293 7.420967 7.325263 7.440997
## 2059       7.382081 7.343401 7.420760 7.322925 7.441236
## 2060       7.382081 7.342524 7.421637 7.321584 7.442577
## 2061       7.382081 7.341667 7.422495 7.320273 7.443889
## 2062       7.381727 7.340473 7.422981 7.318634 7.444820
## 2063       7.381991 7.339914 7.424069 7.317640 7.446343
## 2064       7.382253 7.339368 7.425137 7.316666 7.447839
## 2065       7.382089 7.338412 7.425766 7.315290 7.448887
## 2066       7.382666 7.338211 7.427122 7.314677 7.450655
## 2067       7.382666 7.337475 7.427858 7.313552 7.451781
## 2068       7.382666 7.336750 7.428582 7.312444 7.452888
## 2069       7.382799 7.336171 7.429428 7.311487 7.454112
## 2070       7.382699 7.335368 7.430030 7.310312 7.455085
## 2071       7.382588 7.334565 7.430611 7.309144 7.456033
## 2072       7.382631 7.333926 7.431337 7.308143 7.457120
## 2073       7.382392 7.333015 7.431770 7.306875 7.457909
## 2074       7.382392 7.332341 7.432443 7.305846 7.458939
## 2075       7.382392 7.331677 7.433108 7.304830 7.459955
## 2076       7.382313 7.330942 7.433684 7.303748 7.460879
## 2077       7.382372 7.330354 7.434391 7.302817 7.461928
## 2078       7.382431 7.329773 7.435089 7.301898 7.462964
## 2079       7.382395 7.329105 7.435685 7.300895 7.463895
## 2080       7.382525 7.328610 7.436439 7.300070 7.464979
## 2081       7.382525 7.327999 7.437051 7.299134 7.465915
## 2082       7.382525 7.327393 7.437656 7.298209 7.466840
## 2083       7.382555 7.326825 7.438285 7.297324 7.467786
## 2084       7.382532 7.326210 7.438854 7.296395 7.468669
## 2085       7.382507 7.325599 7.439415 7.295474 7.469540
## 2086       7.382517 7.325029 7.440005 7.294597 7.470437
## 2087       7.382463 7.324400 7.440525 7.293664 7.471261
## 2088       7.382463 7.323830 7.441095 7.292792 7.472134
```

```
plot(forecast(modelx ,100),xlim=c(1800,2200),ylim=c(7,8))
```



I also tried auto arima function again, at first it provides ARIMA(0,1,0) with AIC=-14617.12.

```
auto.arima(log(df2$Ounce_Price), seasonal = TRUE)
```

```
## Series: log(df2$Ounce_Price)
## ARIMA(0,1,0)
##
## sigma^2 estimated as 4.476e-05:  log likelihood=7309.56
## AIC=-14617.12   AICc=-14617.12   BIC=-14611.5
```

Then I pushed it to see if there is a better option, so it provides ARIMA(4,1,1) with higher AIC=-14610.03 which is worse than its first production according to AIC levels.

```
auto.arima(log(df2$Ounce_Price), start.P = 2, start.Q = 2, max.order = 6,
           d=1, ic="aic", trace=TRUE, stepwise = FALSE)
```

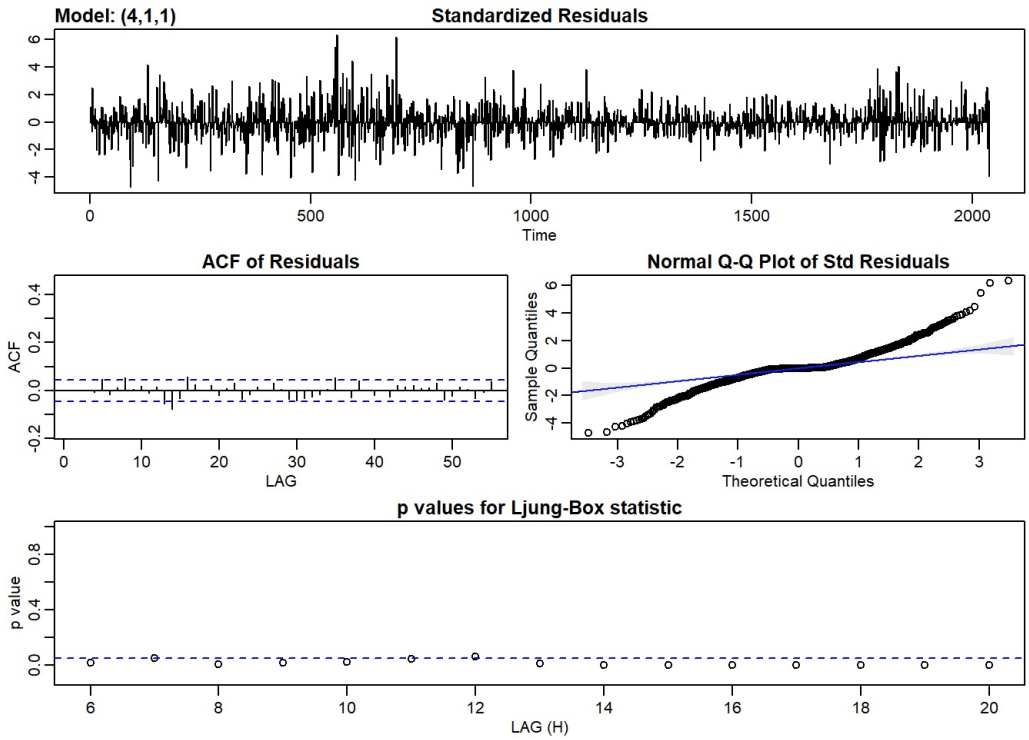
```
##
##  Fitting models using approximations to speed things up...
##
##  ARIMA(0,1,0)                    : -14607.11
##  ARIMA(0,1,0) with drift         : -14605.64
##  ARIMA(0,1,1)                    : -14605.16
##  ARIMA(0,1,1) with drift         : -14603.69
##  ARIMA(0,1,2)                    : -14603.18
##  ARIMA(0,1,2) with drift         : -14601.71
##  ARIMA(0,1,3)                    : -14601.79
##  ARIMA(0,1,3) with drift         : -14600.33
##  ARIMA(0,1,4)                    : -14600.29
##  ARIMA(0,1,4) with drift         : -14598.85
##  ARIMA(0,1,5)                    : -14605.07
##  ARIMA(0,1,5) with drift         : -14603.58
##  ARIMA(1,1,0)                    : -14604.16
##  ARIMA(1,1,0) with drift         : -14602.69
##  ARIMA(1,1,1)                    : -14602.16
##  ARIMA(1,1,1) with drift         : -14600.69
##  ARIMA(1,1,2)                    : -14600.18
##  ARIMA(1,1,2) with drift         : -14598.71
##  ARIMA(1,1,3)                    : -14598.79
##  ARIMA(1,1,3) with drift         : -14597.34
##  ARIMA(1,1,4)                    : -14600.86
##  ARIMA(1,1,4) with drift         : -14599.42
##  ARIMA(1,1,5)                    : -14603.07
##  ARIMA(1,1,5) with drift         : -14601.59
##  ARIMA(2,1,0)                    : -14601.18
##  ARIMA(2,1,0) with drift         : -14599.7
##  ARIMA(2,1,1)                    : -14599.18
##  ARIMA(2,1,1) with drift         : -14597.7
##  ARIMA(2,1,2)                    : -14597.18
##  ARIMA(2,1,2) with drift         : -14595.71
##  ARIMA(2,1,3)                    : -14603.66
##  ARIMA(2,1,3) with drift         : -14601.98
##  ARIMA(2,1,4)                    : -14603.74
##  ARIMA(2,1,4) with drift         : -14602.3
##  ARIMA(3,1,0)                    : -14598.74
##  ARIMA(3,1,0) with drift         : -14597.29
##  ARIMA(3,1,1)                    : -14596.74
##  ARIMA(3,1,1) with drift         : -14595.29
##  ARIMA(3,1,2)                    : -14603.05
##  ARIMA(3,1,2) with drift         : -14601.67
##  ARIMA(3,1,3)                    : -14600.81
##  ARIMA(3,1,3) with drift         : -14599.14
##  ARIMA(4,1,0)                    : -14596.74
##  ARIMA(4,1,0) with drift         : -14595.33
##  ARIMA(4,1,1)                    : -14607.79
##  ARIMA(4,1,1) with drift         : -14606.42
##  ARIMA(4,1,2)                    : -14606.14
##  ARIMA(4,1,2) with drift         : -14604.73
##  ARIMA(5,1,0)                    : -14606.45
##  ARIMA(5,1,0) with drift         : -14604.9
##  ARIMA(5,1,1)                    : -14605.57
##  ARIMA(5,1,1) with drift         : -14604.04
##
##  Now re-fitting the best model(s) without approximations...
##
##
##
##
##  Best model: ARIMA(4,1,1)
```

```
## Series: log(df2$Ounce_Price)
## ARIMA(4,1,1)
##
## Coefficients:
##           ar1     ar2      ar3      ar4     ma1
##       -0.9690  0.0081  -0.0137  -0.0241  0.9751
## s.e.   0.0295  0.0309   0.0309   0.0225  0.0194
##
## sigma^2 estimated as 4.481e-05:  log likelihood=7311.02
## AIC=-14610.03   AICc=-14609.99   BIC=-14576.32
```

I also checked its assumptions but still they did not met.

```
sarima(log(df2$Ounce_Price), 4,1,1)
```

```
## initial  value -5.006599
## iter   2 value -5.006889
## iter   3 value -5.006896
## iter   4 value -5.006906
## iter   5 value -5.006908
## iter   6 value -5.006940
## iter   7 value -5.007005
## iter   8 value -5.007269
## iter   9 value -5.007386
## iter  10 value -5.007516
## iter  11 value -5.007931
## iter  12 value -5.008030
## iter  13 value -5.008367
## iter  14 value -5.009311
## iter  15 value -5.009517
## iter  16 value -5.009796
## iter  17 value -5.009974
## iter  18 value -5.010028
## iter  19 value -5.010038
## iter  20 value -5.010070
## iter  21 value -5.010099
## iter  22 value -5.010106
## iter  23 value -5.010109
## iter  24 value -5.010109
## iter  25 value -5.010110
## iter  25 value -5.010110
## iter  25 value -5.010110
## final  value -5.010110
## converged
## initial  value -5.007928
## iter   2 value -5.008181
## iter   3 value -5.008184
## iter   4 value -5.008184
## iter   5 value -5.008184
## iter   6 value -5.008185
## iter   7 value -5.008185
## iter   8 value -5.008185
## iter   9 value -5.008185
## iter  10 value -5.008186
## iter  11 value -5.008186
## iter  12 value -5.008186
## iter  13 value -5.008187
## iter  14 value -5.008188
## iter  15 value -5.008188
## iter  16 value -5.008188
## iter  16 value -5.008188
## iter  16 value -5.008188
## final  value -5.008188
## converged
```

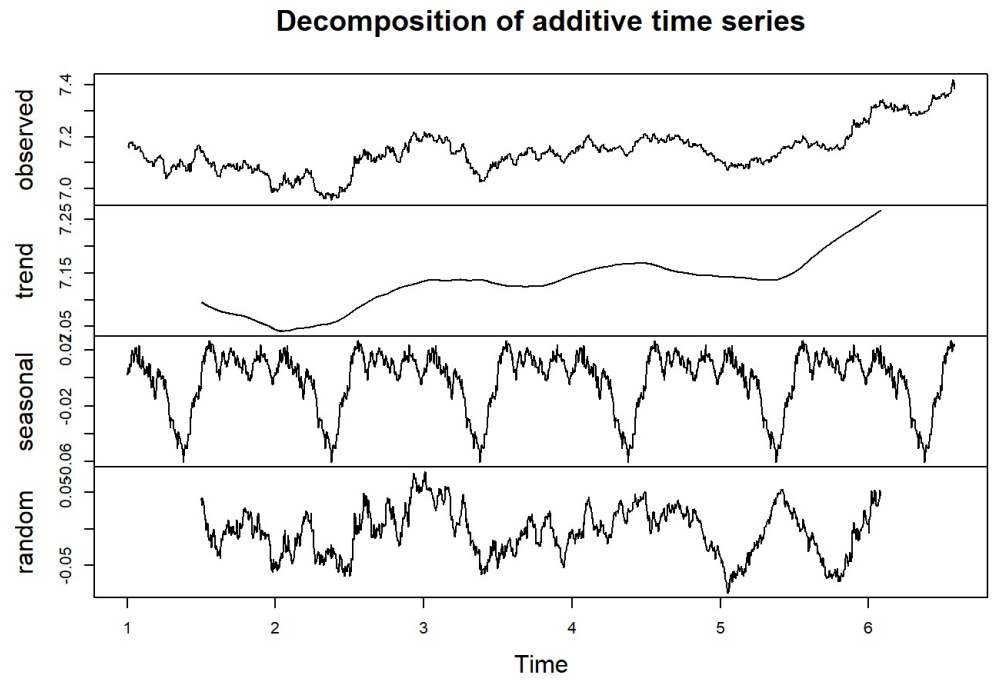



```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), xreg = constant, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##           ar1     ar2      ar3      ar4     ma1  constant
##       -0.9658  0.0075  -0.0143  -0.0248  0.9718     1e-04
## s.e.   0.0322  0.0309   0.0309   0.0225  0.0234     1e-04
##
## sigma^2 estimated as 4.466e-05:  log likelihood = 7311.3,  aic = -14608.6
##
## $degrees_of_freedom
## [1] 2031
##
## $ttable
##          Estimate     SE  t.value p.value
## ar1       -0.9658 0.0322 -30.0001  0.0000
## ar2        0.0075 0.0309   0.2445  0.8069
## ar3       -0.0143 0.0309  -0.4633  0.6432
## ar4       -0.0248 0.0225  -1.1053  0.2692
## ma1        0.9718 0.0234  41.5568  0.0000
## constant   0.0001 0.0001   0.7272  0.4672
##
## $AIC
## [1] -7.171627
##
## $AICc
## [1] -7.171607
##
## $BIC
## [1] -7.152317
```

ARIMA & SARIMA after decomposition

I thought there is still something is missing or wrong in my model trials. So I decided to decompose the dataset to see a better picture about trend and seasonality.

```
comps <- decompose(ts(log(df2$Ounce_Price), deltat= 1/365))
plot(comps)
```

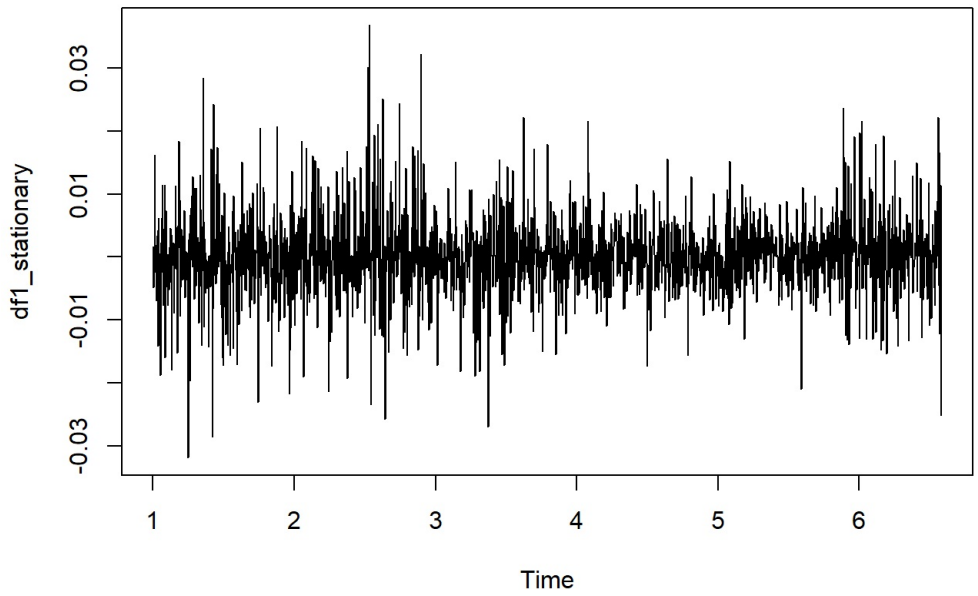



```
summary(comps)
```

```
##          Length Class  Mode
## x        2038   ts     numeric
## seasonal 2038   ts     numeric
## trend    2038   ts     numeric
## random   2038   ts     numeric
## figure    365   -none- numeric
## type        1   -none- character
```

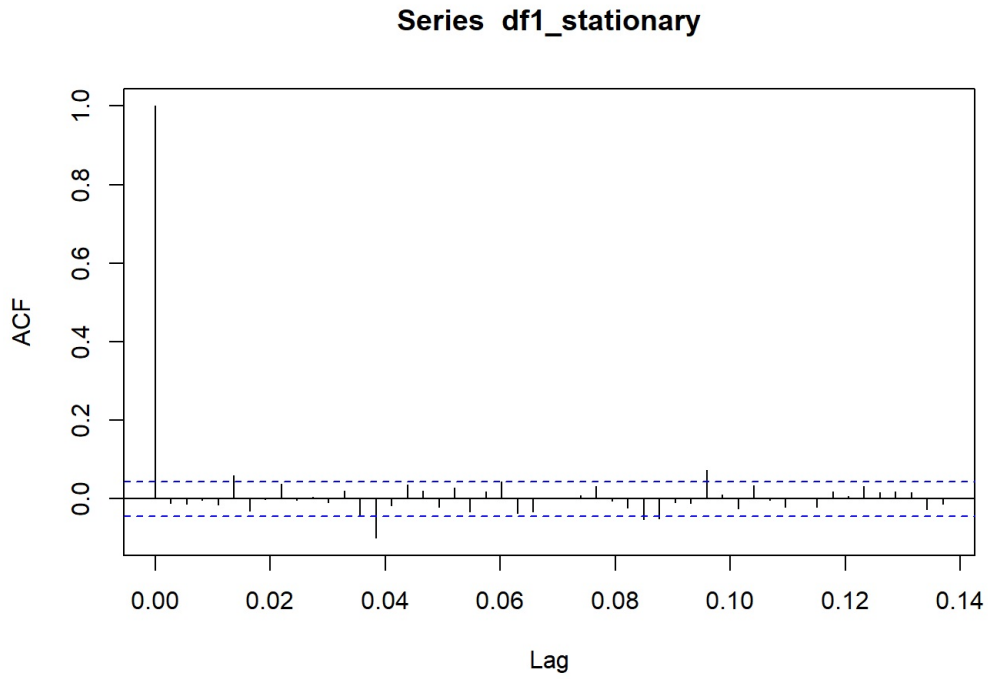
I removed seasonality and took the difference to make my dataset stationary with the codes below.

```
x <- log(df2$Ounce_Price) - comps$seasonal
df1_stationary <- diff(x, differences = 1)
plot(df1_stationary)
```

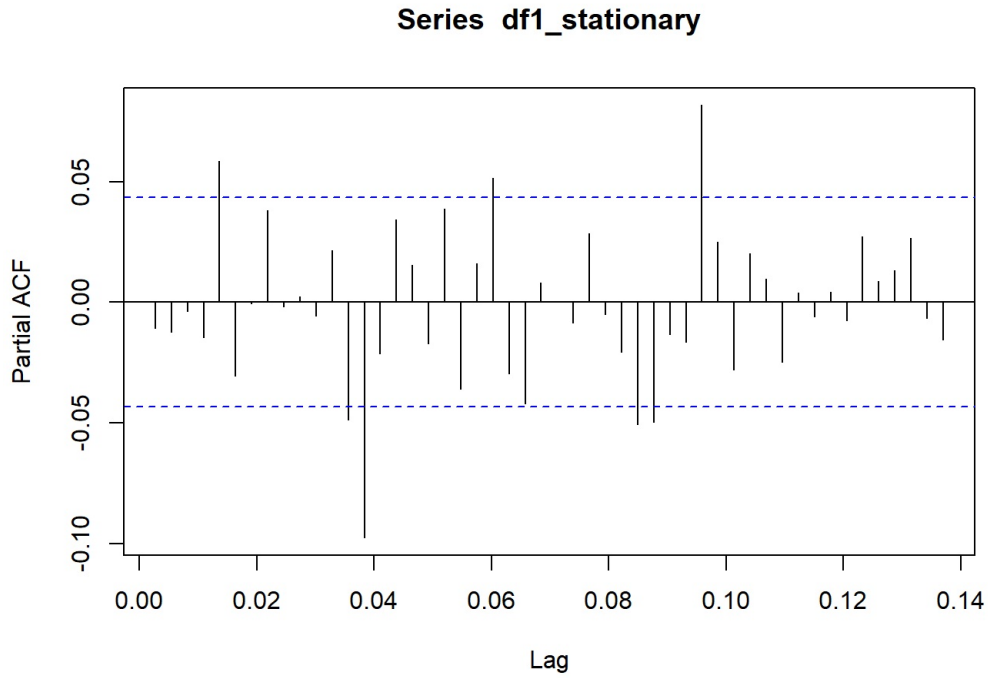


It is ready to fit a model now, but first I will check ACF and PACF to determine AR and MA processes.

```
acf(df1_stationary, lag.max = 50)
```




```
pacf(df1_stationary, lag.max = 50)
```



As it can be seen, it is clear that there are more than 1 seasonal spikes in both ACF and PACF plots.

SO I again generated a for loop with this time frequency level 5 and for p <= 1 q<= 1 and P <= 5 Q<=4 with if condition to keep their sum below 7.

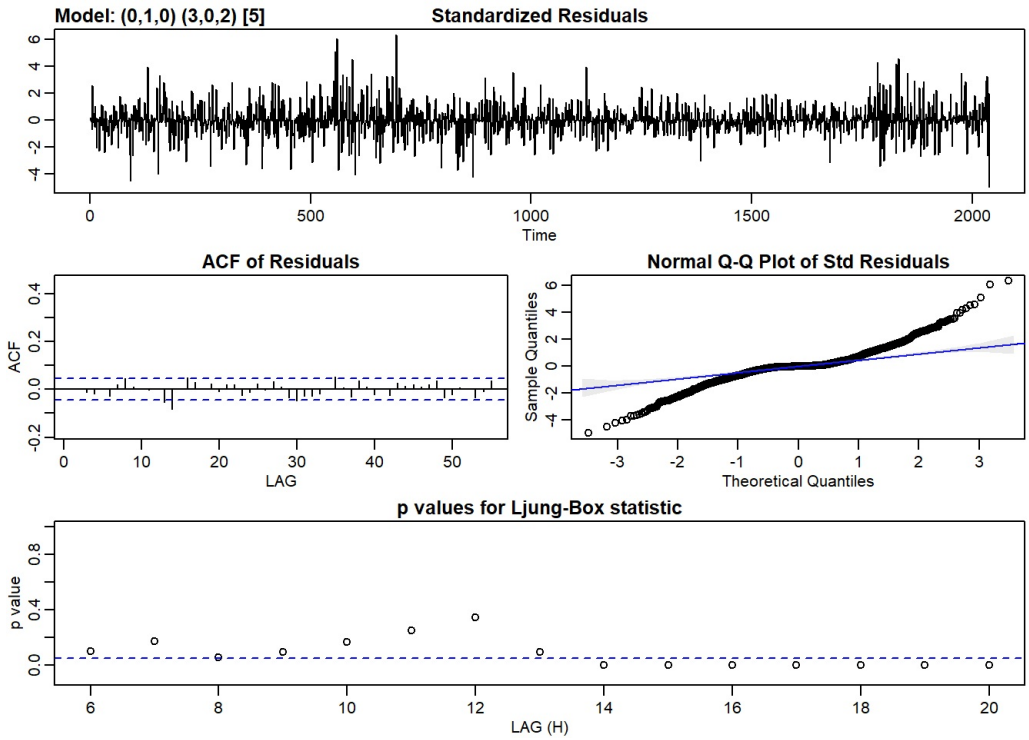
```
d=1
D=0

for(p in 0:1){
  for(q in 0:1){
    for(P in 0:5){
      for(Q in 0:4){
        if( p+q+P+Q+d+D <= 6){ # parcimony principle
          model <- arima(x= log(df2$Ounce_Price), order=c(p,d,q),
                         seasonal = list(order=c(P,D,Q), frequency = 5))
          sse <- sum(resid(model)^2)
          order <- paste(p,d,q,P,D,Q)
          pval<-Box.test(model$residuals, lag=log(length(model$residuals)))
          cat(p,d,q,P,D,Q,5, 'AIC=', model$aic, ' SSE=',sse,' p-VALUE=',
              pval$p.value,'\n')
        }
      }
    }
  }
}
```

```
## 0 1 0 0 0 0 5 AIC= -14617.12  SSE= 0.09118448  p-VALUE= 0.2608558
## 0 1 0 0 0 1 5 AIC= -14615.18  SSE= 0.09118186  p-VALUE= 0.2584731
## 0 1 0 0 0 2 5 AIC= -14613.2  SSE= 0.09118105  p-VALUE= 0.2590366
## 0 1 0 0 0 3 5 AIC= -14611.8  SSE= 0.09115392  p-VALUE= 0.2925342
## 0 1 0 0 0 4 5 AIC= -14610.3  SSE= 0.09113158  p-VALUE= 0.3348895
## 0 1 0 1 0 0 5 AIC= -14615.18  SSE= 0.09118185  p-VALUE= 0.258453
## 0 1 0 1 0 1 5 AIC= -14613.18  SSE= 0.09118186  p-VALUE= 0.2584664
## 0 1 0 1 0 2 5 AIC= -14611.2  SSE= 0.09118107  p-VALUE= 0.2590509
## 0 1 0 1 0 3 5 AIC= -14609.81  SSE= 0.09115376  p-VALUE= 0.2926626
## 0 1 0 1 0 4 5 AIC= -14611.88  SSE= 0.09097124  p-VALUE= 0.8552784
## 0 1 0 2 0 0 5 AIC= -14613.19  SSE= 0.09118117  p-VALUE= 0.25904
## 0 1 0 2 0 1 5 AIC= -14611.2  SSE= 0.09118115  p-VALUE= 0.2590383
## 0 1 0 2 0 2 5 AIC= -14609.2  SSE= 0.0911811  p-VALUE= 0.2590801
## 0 1 0 2 0 3 5 AIC= -14615.66  SSE= 0.09080153  p-VALUE= 0.5245891
## 0 1 0 3 0 0 5 AIC= -14611.75  SSE= 0.0911562  p-VALUE= 0.2876077
## 0 1 0 3 0 1 5 AIC= -14609.75  SSE= 0.0911563  p-VALUE= 0.2877375
## 0 1 0 3 0 2 5 AIC= -14615.68  SSE= 0.09080062  p-VALUE= 0.5201874
## 0 1 0 4 0 0 5 AIC= -14610.3  SSE= 0.09113154  p-VALUE= 0.3329942
## 0 1 0 4 0 1 5 AIC= -14610.03  SSE= 0.09105315  p-VALUE= 0.5986666
## 0 1 0 5 0 0 5 AIC= -14614.67  SSE= 0.09084665  p-VALUE= 0.9745169
## 0 1 1 0 0 0 5 AIC= -14615.18  SSE= 0.09118186  p-VALUE= 0.2584731
## 0 1 1 0 0 1 5 AIC= -14613.18  SSE= 0.09118186  p-VALUE= 0.2584658
## 0 1 1 0 0 2 5 AIC= -14611.2  SSE= 0.09118107  p-VALUE= 0.2590495
## 0 1 1 0 0 3 5 AIC= -14609.81  SSE= 0.09115376  p-VALUE= 0.2926666
## 0 1 1 0 0 4 5 AIC= -14609.62  SSE= 0.09107263  p-VALUE= 0.4582706
## 0 1 1 1 0 0 5 AIC= -14613.18  SSE= 0.09118186  p-VALUE= 0.2584664
## 0 1 1 1 0 1 5 AIC= -14611.18  SSE= 0.09118186  p-VALUE= 0.2584693
## 0 1 1 1 0 2 5 AIC= -14609.2  SSE= 0.09118108  p-VALUE= 0.2590807
## 0 1 1 1 0 3 5 AIC= -14607.81  SSE= 0.09115369  p-VALUE= 0.2923721
## 0 1 1 2 0 0 5 AIC= -14611.2  SSE= 0.09118115  p-VALUE= 0.2590383
## 0 1 1 2 0 1 5 AIC= -14609.2  SSE= 0.09118114  p-VALUE= 0.2590515
## 0 1 1 2 0 2 5 AIC= -14607.2  SSE= 0.0911811  p-VALUE= 0.2591639
## 0 1 1 3 0 0 5 AIC= -14609.75  SSE= 0.0911563  p-VALUE= 0.2877375
## 0 1 1 3 0 1 5 AIC= -14607.75  SSE= 0.09115633  p-VALUE= 0.287212
## 0 1 1 4 0 0 5 AIC= -14610.03  SSE= 0.09105315  p-VALUE= 0.5986665
## 1 1 0 0 0 0 5 AIC= -14615.18  SSE= 0.09118185  p-VALUE= 0.258453
## 1 1 0 0 0 1 5 AIC= -14613.18  SSE= 0.09118186  p-VALUE= 0.2584664
## 1 1 0 0 0 2 5 AIC= -14611.2  SSE= 0.09118107  p-VALUE= 0.2590509
## 1 1 0 0 0 3 5 AIC= -14609.81  SSE= 0.09115376  p-VALUE= 0.2926626
## 1 1 0 0 0 4 5 AIC= -14611.88  SSE= 0.09097124  p-VALUE= 0.8552784
## 1 1 0 1 0 0 5 AIC= -14613.18  SSE= 0.09118185  p-VALUE= 0.2584609
## 1 1 0 1 0 1 5 AIC= -14611.18  SSE= 0.09118185  p-VALUE= 0.2584674
## 1 1 0 1 0 2 5 AIC= -14609.2  SSE= 0.09118108  p-VALUE= 0.2590283
## 1 1 0 1 0 3 5 AIC= -14607.81  SSE= 0.0911537  p-VALUE= 0.2923719
## 1 1 0 2 0 0 5 AIC= -14611.2  SSE= 0.09118114  p-VALUE= 0.2590442
## 1 1 0 2 0 1 5 AIC= -14609.2  SSE= 0.09118114  p-VALUE= 0.2590278
## 1 1 0 2 0 2 5 AIC= -14607.2  SSE= 0.0911811  p-VALUE= 0.259058
## 1 1 0 3 0 0 5 AIC= -14609.75  SSE= 0.0911563  p-VALUE= 0.2877021
## 1 1 0 3 0 1 5 AIC= -14611.58  SSE= 0.09098498  p-VALUE= 0.8699916
## 1 1 0 4 0 0 5 AIC= -14614.67  SSE= 0.09084671  p-VALUE= 0.9746426
## 1 1 1 0 0 0 5 AIC= -14613.18  SSE= 0.09118186  p-VALUE= 0.2584664
## 1 1 1 0 0 1 5 AIC= -14611.18  SSE= 0.09118186  p-VALUE= 0.2584693
## 1 1 1 0 0 2 5 AIC= -14609.2  SSE= 0.09118108  p-VALUE= 0.2590807
## 1 1 1 0 0 3 5 AIC= -14607.81  SSE= 0.09115369  p-VALUE= 0.2923721
## 1 1 1 1 0 0 5 AIC= -14611.18  SSE= 0.09118185  p-VALUE= 0.2584674
## 1 1 1 1 0 1 5 AIC= -14609.18  SSE= 0.09118186  p-VALUE= 0.2584718
## 1 1 1 1 0 2 5 AIC= -14607.2  SSE= 0.09118108  p-VALUE= 0.2590396
## 1 1 1 2 0 0 5 AIC= -14609.2  SSE= 0.09118114  p-VALUE= 0.2590278
## 1 1 1 2 0 1 5 AIC= -14607.2  SSE= 0.09118113  p-VALUE= 0.259003
## 1 1 1 3 0 0 5 AIC= -14611.58  SSE= 0.09098498  p-VALUE= 0.8699916
```

Again best model occured in (0 1 0 0 0 0) with lowest AIC= -14617.12 and SSE= 0.09118448 and second best became (0 1 0 2 0 3) with AIC= -14615.68 lower SSE= 0.09080062.

```
sarima(df2$Ounce_Price, 0,1,0,3,0,2,5)
```

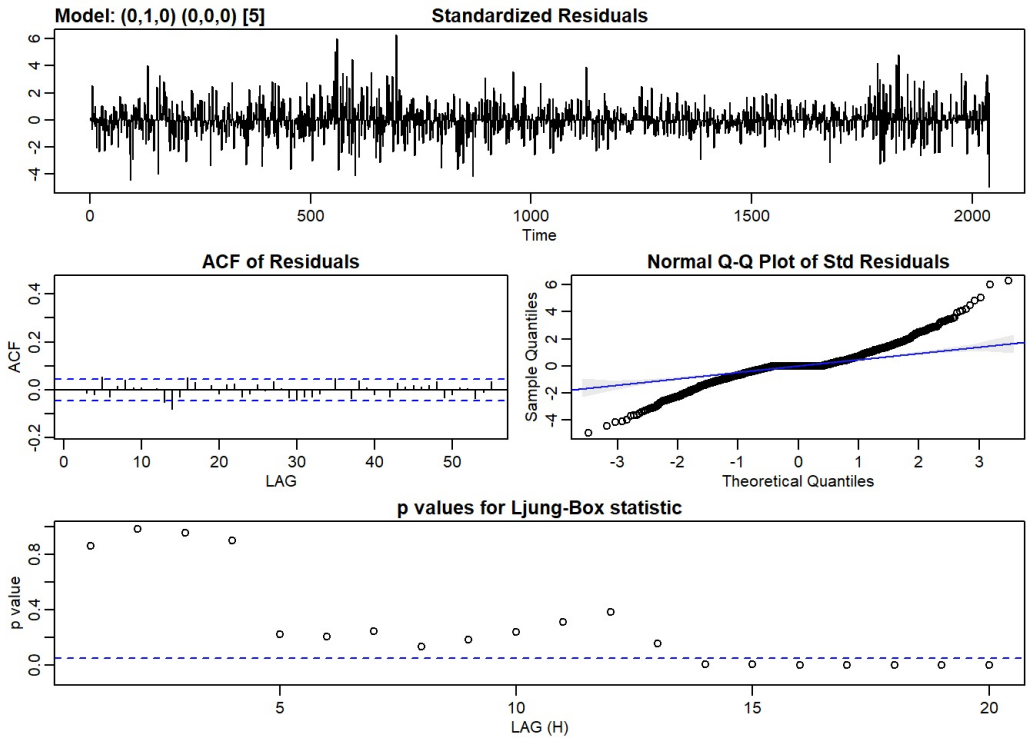
```
## initial  value 2.136140
## iter   2 value 2.135539
## iter   3 value 2.134132
## iter   4 value 2.134132
## iter   5 value 2.134131
## iter   6 value 2.134131
## iter   7 value 2.134128
## iter   8 value 2.134124
## iter   9 value 2.134115
## iter  10 value 2.134107
## iter  11 value 2.134102
## iter  12 value 2.134101
## iter  13 value 2.134101
## iter  14 value 2.134100
## iter  15 value 2.134099
## iter  16 value 2.134099
## iter  17 value 2.134099
## iter  18 value 2.134098
## iter  19 value 2.134097
## iter  20 value 2.134097
## iter  21 value 2.134097
## iter  22 value 2.134097
## iter  23 value 2.134097
## iter  24 value 2.134097
## iter  25 value 2.134096
## iter  26 value 2.134095
## iter  27 value 2.134094
## iter  28 value 2.134094
## iter  29 value 2.134094
## iter  29 value 2.134094
## iter  29 value 2.134094
## final  value 2.134094
## converged
## initial  value 2.133622
## iter   2 value 2.133622
## iter   3 value 2.133621
## iter   4 value 2.133621
## iter   5 value 2.133619
## iter   6 value 2.133616
## iter   7 value 2.133609
## iter   8 value 2.133600
## iter   9 value 2.133591
## iter  10 value 2.133589
## iter  11 value 2.133589
## iter  12 value 2.133589
## iter  13 value 2.133588
## iter  14 value 2.133588
## iter  15 value 2.133588
## iter  16 value 2.133588
## iter  16 value 2.133588
## iter  16 value 2.133588
## final  value 2.133588
## converged
```




```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), xreg = constant, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##         sar1     sar2     sar3     sma1    sma2  constant
##       0.1746  -0.1154  -0.0277  -0.1200  0.1181    0.1580
## s.e.  1.4105   0.4901   0.0413   1.4116  0.5535    0.1929
##
## sigma^2 estimated as 71.32:  log likelihood = -7236.5,  aic = 14486.99
##
## $degrees_of_freedom
## [1] 2031
##
## $ttable
##          Estimate     SE t.value p.value
## sar1       0.1746 1.4105  0.1237  0.9015
## sar2      -0.1154 0.4901 -0.2354  0.8139
## sar3      -0.0277 0.0413 -0.6707  0.5025
## sma1      -0.1200 1.4116 -0.0850  0.9323
## sma2       0.1181 0.5535  0.2134  0.8310
## constant   0.1580 0.1929  0.8195  0.4126
##
## $AIC
## [1] 7.111925
##
## $AICc
## [1] 7.111946
##
## $BIC
## [1] 7.131235
```

```
sarima(df2$Ounce_Price, 0,1,0,0,0,0,5)
```

```
## initial  value 2.135637
## iter   1 value 2.135637
## final  value 2.135637
## converged
## initial  value 2.135637
## iter   1 value 2.135637
## final  value 2.135637
## converged
```




```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), xreg = constant, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##       constant
##         0.1564
## s.e.    0.1875
##
## sigma^2 estimated as 71.61:  log likelihood = -7240.67,  aic = 14485.34
##
## $degrees_of_freedom
## [1] 2036
##
## $ttable
##          Estimate     SE t.value p.value
## constant   0.1564 0.1875  0.8342  0.4043
##
## $AIC
## [1] 7.111115
##
## $AICc
## [1] 7.111116
##
## $BIC
## [1] 7.116632
```

Well, as it can be seen from both models, there are still some autocorrelation remain in residuals according to Ljung-Box Statistic.

At last I generated a model manualy by only taking into account insights from ACF and PACF plots. I determined (0,1,0,4,0,3) for this model and surprisingly it provides better AIC score with AIC= -14637.75

```
model2 <- arima(x=log(df2$Ounce_Price), order = c(0,1,0),
                seasonal = list(order=c(4,0,3), period=5), method = "ML")
```

```
## Warning in arima(x = log(df2$Ounce_Price), order = c(0, 1, 0), seasonal =
## list(order = c(4, : possible convergence problem: optim gave code = 1
```

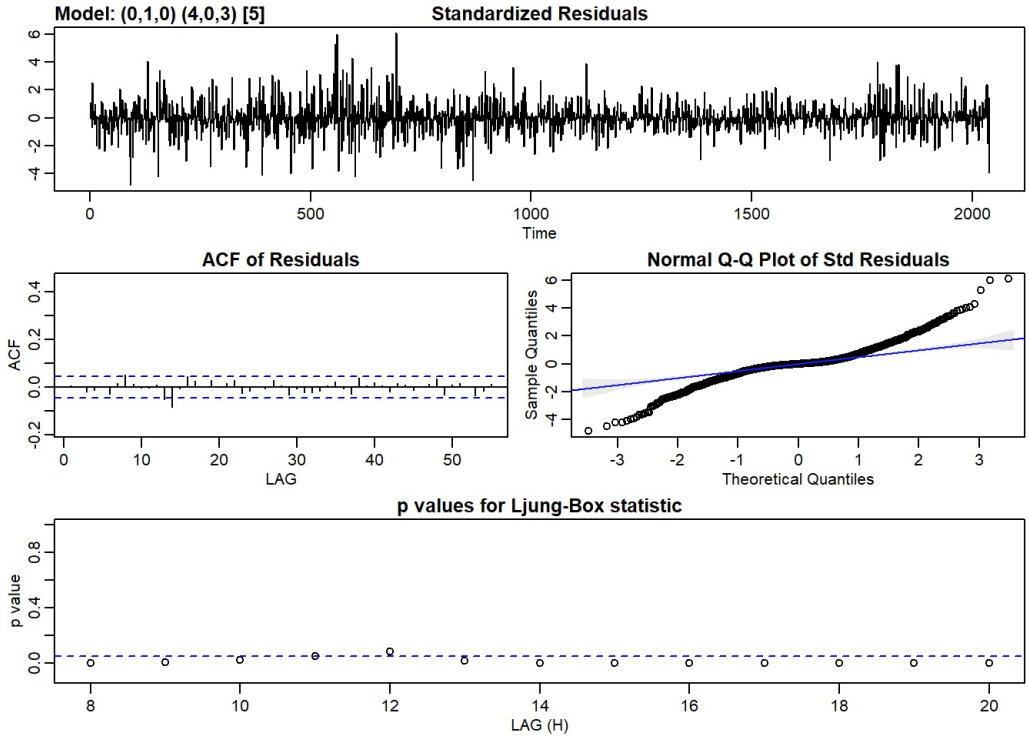
```
summary(model2)
```

```
##
## Call:
## arima(x = log(df2$Ounce_Price), order = c(0, 1, 0), seasonal = list(order = c(4,
##     0, 3), period = 5), method = "ML")
##
## Coefficients:
##          sar1    sar2    sar3     sar4    sma1     sma2     sma3
##       -0.6836  0.9956  0.7410  -0.0867  0.7442  -0.9451  -0.7986
## s.e.   0.0708  0.0241  0.0649   0.0248  0.0686   0.0202   0.0649
##
## sigma^2 estimated as 4.386e-05:  log likelihood = 7326.88,  aic = -14637.75
##
## Training set error measures:
##                        ME        RMSE         MAE         MPE       MAPE
## Training set 0.0001323388 0.006622678 0.004209449 0.001769646 0.05896055
##                   MASE        ACF1
## Training set 1.046328 0.007499686
```

```
sarima(log(df2$Ounce_Price), 0,1,0,4,0,3,5)
```

```
## initial  value -5.006889
## iter   2 value -5.007239
## iter   3 value -5.009099
## iter   4 value -5.009102
## iter   5 value -5.009105
## iter   6 value -5.009115
## iter   7 value -5.009150
## iter   8 value -5.009221
## iter   9 value -5.009411
## iter  10 value -5.009772
## iter  11 value -5.010773
## iter  12 value -5.010909
## iter  13 value -5.011299
## iter  14 value -5.011364
## iter  15 value -5.011919
## iter  16 value -5.012235
## iter  17 value -5.012385
## iter  18 value -5.012461
## iter  19 value -5.012494
## iter  20 value -5.012496
## iter  21 value -5.012498
## iter  22 value -5.012502
## iter  23 value -5.012508
## iter  24 value -5.012512
## iter  25 value -5.012514
## iter  26 value -5.012515
## iter  27 value -5.012518
## iter  28 value -5.012518
## iter  29 value -5.012521
## iter  30 value -5.012522
## iter  31 value -5.012524
## iter  32 value -5.012525
## iter  33 value -5.012529
## iter  34 value -5.012533
## iter  35 value -5.012536
## iter  36 value -5.012538
## iter  37 value -5.012539
## iter  38 value -5.012543
## iter  39 value -5.012550
## iter  40 value -5.012559
## iter  41 value -5.012566
## iter  42 value -5.012568
## iter  43 value -5.012568
## iter  43 value -5.012568
## iter  43 value -5.012568
## final  value -5.012568
## converged
## initial  value -5.013299
## iter   2 value -5.013312
## iter   3 value -5.013327
## iter   4 value -5.013373
## iter   5 value -5.013431
## iter   6 value -5.013500
## iter   7 value -5.013567
## iter   8 value -5.013586
## iter   9 value -5.013598
## iter  10 value -5.013631
```

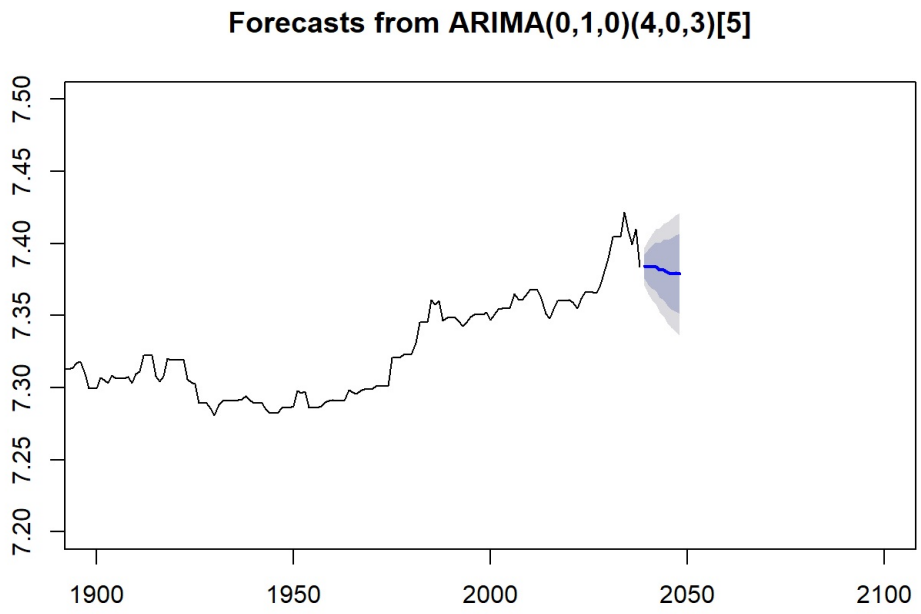
```
## iter  11 value -5.013706
## iter  12 value -5.013823
## iter  13 value -5.013945
## iter  14 value -5.014327
## iter  15 value -5.014762
## iter  16 value -5.014799
## iter  17 value -5.014858
## iter  18 value -5.014920
## iter  19 value -5.014923
## iter  20 value -5.014931
## iter  21 value -5.014933
## iter  22 value -5.014945
## iter  23 value -5.014953
## iter  24 value -5.014957
## iter  25 value -5.014958
## iter  26 value -5.014960
## iter  27 value -5.014966
## iter  28 value -5.014978
## iter  29 value -5.015003
## iter  30 value -5.015025
## iter  31 value -5.015033
## iter  32 value -5.015037
## iter  33 value -5.015038
## iter  34 value -5.015039
## iter  35 value -5.015041
## iter  35 value -5.015041
## final  value -5.015041
## converged
```



```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), xreg = constant, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##          sar1    sar2    sar3     sar4    sma1     sma2     sma3  constant
##       -0.9235  0.4855  0.4643  -0.0807  0.9904  -0.4151  -0.5015     1e-04
## s.e.   0.5389  1.0090  0.4956   0.0285  0.5401   1.0314   0.5260     2e-04
##
## sigma^2 estimated as 4.404e-05:  log likelihood = 7325.26,  aic = -14632.52
##
## $degrees_of_freedom
## [1] 2029
##
## $ttable
##          Estimate     SE t.value p.value
## sar1      -0.9235 0.5389 -1.7137  0.0867
## sar2       0.4855 1.0090  0.4812  0.6304
## sar3       0.4643 0.4956  0.9369  0.3489
## sar4      -0.0807 0.0285 -2.8364  0.0046
## sma1       0.9904 0.5401  1.8336  0.0669
## sma2      -0.4151 1.0314 -0.4024  0.6874
## sma3      -0.5015 0.5260 -0.9534  0.3405
## constant   0.0001 0.0002  0.7081  0.4790
##
## $AIC
## [1] -7.183368
##
## $AICc
## [1] -7.183333
##
## $BIC
## [1] -7.158541
```

But still I cannot drop autocorrelation in residuals.I did forecast by using this last model anyway.

```
plot(forecast(model2),xlim=c(1900,2100),ylim=c(7.2,7.5))
```



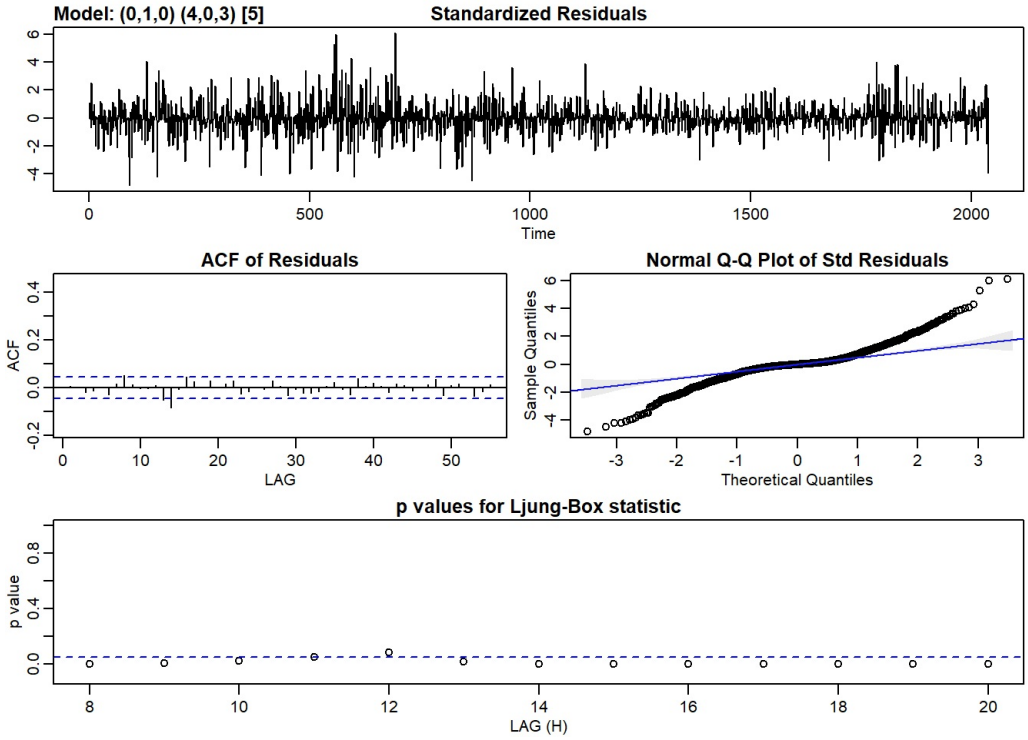

```
Box.test(model2$residuals)
```

```
##
##  Box-Pierce test
##
## data:  model2$residuals
## X-squared = 0.11463, df = 1, p-value = 0.7349
```

```
sarima(log(df2$Ounce_Price), 0,1,0,4,0,3,5)
```

```
## initial  value -5.006889
## iter   2 value -5.007239
## iter   3 value -5.009099
## iter   4 value -5.009102
## iter   5 value -5.009105
## iter   6 value -5.009115
## iter   7 value -5.009150
## iter   8 value -5.009221
## iter   9 value -5.009411
## iter  10 value -5.009772
## iter  11 value -5.010773
## iter  12 value -5.010909
## iter  13 value -5.011299
## iter  14 value -5.011364
## iter  15 value -5.011919
## iter  16 value -5.012235
## iter  17 value -5.012385
## iter  18 value -5.012461
## iter  19 value -5.012494
## iter  20 value -5.012496
## iter  21 value -5.012498
## iter  22 value -5.012502
## iter  23 value -5.012508
## iter  24 value -5.012512
## iter  25 value -5.012514
## iter  26 value -5.012515
## iter  27 value -5.012518
## iter  28 value -5.012518
## iter  29 value -5.012521
## iter  30 value -5.012522
## iter  31 value -5.012524
## iter  32 value -5.012525
## iter  33 value -5.012529
## iter  34 value -5.012533
## iter  35 value -5.012536
## iter  36 value -5.012538
## iter  37 value -5.012539
## iter  38 value -5.012543
## iter  39 value -5.012550
## iter  40 value -5.012559
## iter  41 value -5.012566
## iter  42 value -5.012568
## iter  43 value -5.012568
## iter  43 value -5.012568
## iter  43 value -5.012568
## final  value -5.012568
## converged
## initial  value -5.013299
## iter   2 value -5.013312
## iter   3 value -5.013327
## iter   4 value -5.013373
## iter   5 value -5.013431
## iter   6 value -5.013500
## iter   7 value -5.013567
## iter   8 value -5.013586
## iter   9 value -5.013598
## iter  10 value -5.013631
## iter  11 value -5.013706
## iter  12 value -5.013823
## iter  13 value -5.013945
## iter  14 value -5.014327
## iter  15 value -5.014762
## iter  16 value -5.014799
## iter  17 value -5.014858
## iter  18 value -5.014920
## iter  19 value -5.014923
## iter  20 value -5.014931
## iter  21 value -5.014933
## iter  22 value -5.014945
## iter  23 value -5.014953
## iter  24 value -5.014957
## iter  25 value -5.014958
## iter  26 value -5.014960
## iter  27 value -5.014966
## iter  28 value -5.014978
```

```
## iter  29 value -5.015003
## iter  30 value -5.015025
## iter  31 value -5.015033
## iter  32 value -5.015037
## iter  33 value -5.015038
## iter  34 value -5.015039
## iter  35 value -5.015041
## iter  35 value -5.015041
## final  value -5.015041
## converged
```




```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), xreg = constant, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##          sar1    sar2    sar3     sar4    sma1     sma2     sma3  constant
##       -0.9235  0.4855  0.4643  -0.0807  0.9904  -0.4151  -0.5015     1e-04
## s.e.   0.5389  1.0090  0.4956   0.0285  0.5401   1.0314   0.5260     2e-04
##
## sigma^2 estimated as 4.404e-05:  log likelihood = 7325.26,  aic = -14632.52
##
## $degrees_of_freedom
## [1] 2029
##
## $ttable
##          Estimate     SE t.value p.value
## sar1      -0.9235 0.5389 -1.7137  0.0867
## sar2       0.4855 1.0090  0.4812  0.6304
## sar3       0.4643 0.4956  0.9369  0.3489
## sar4      -0.0807 0.0285 -2.8364  0.0046
## sma1       0.9904 0.5401  1.8336  0.0669
## sma2      -0.4151 1.0314 -0.4024  0.6874
## sma3      -0.5015 0.5260 -0.9534  0.3405
## constant   0.0001 0.0002  0.7081  0.4790
##
## $AIC
## [1] -7.183368
##
## $AICc
## [1] -7.183333
##
## $BIC
## [1] -7.158541
```

Afterall I leave this approach in here and proceed to the next approach.

TES - Triple Exponential Smoothing

In my last approach I will use Exponential Smoothing method to forecast future gold prices. since the dataset has both trend and seasonality I used Triple Exponential Smoothing method to forecast. If the dataset have not any trend or seasonality it would be enough to use Sİmple Exponential Smoothing and use `HoltWİnters(beta=F, gama=F)` , but if the dataset has only trend but not seasonality I should use Double Exponential Smoothing and `HoltWİnters(`gama=F)` , but now I will determine these parameters to TRUE.

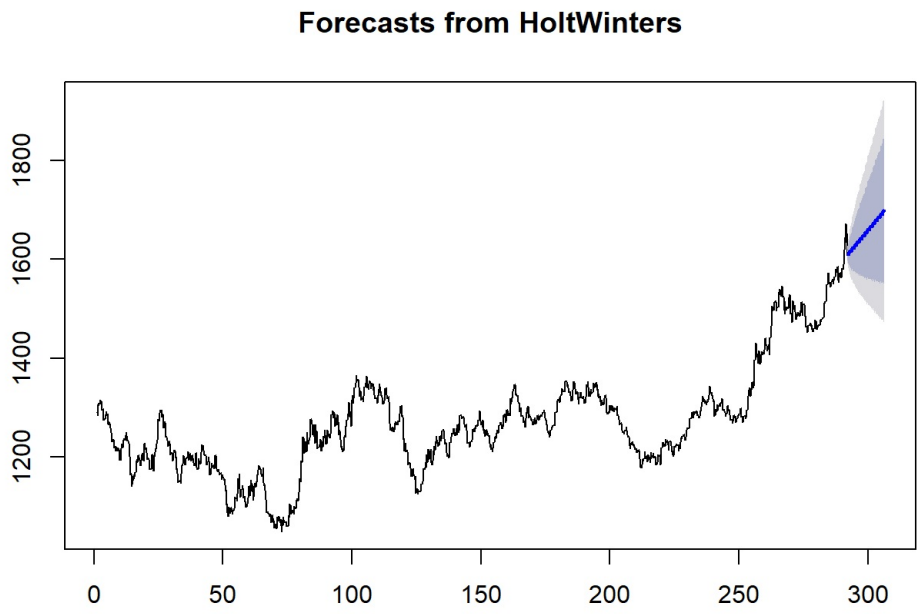
```
hw <- HoltWinters(ts(df2$Ounce_Price, frequency = 7))
hw
```

```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = ts(df2$Ounce_Price, frequency = 7))
##
## Smoothing parameters:
##  alpha: 0.9630005
##  beta : 0.006773294
##  gamma: 1
##
## Coefficients:
##            [,1]
## a  1610.2717875
## b     0.8984085
## s1   -1.6637352
## s2   -2.2344399
## s3   -0.6778132
## s4   -4.6473507
## s5   -1.2615616
## s6    1.6698324
## s7   -0.4217875
```

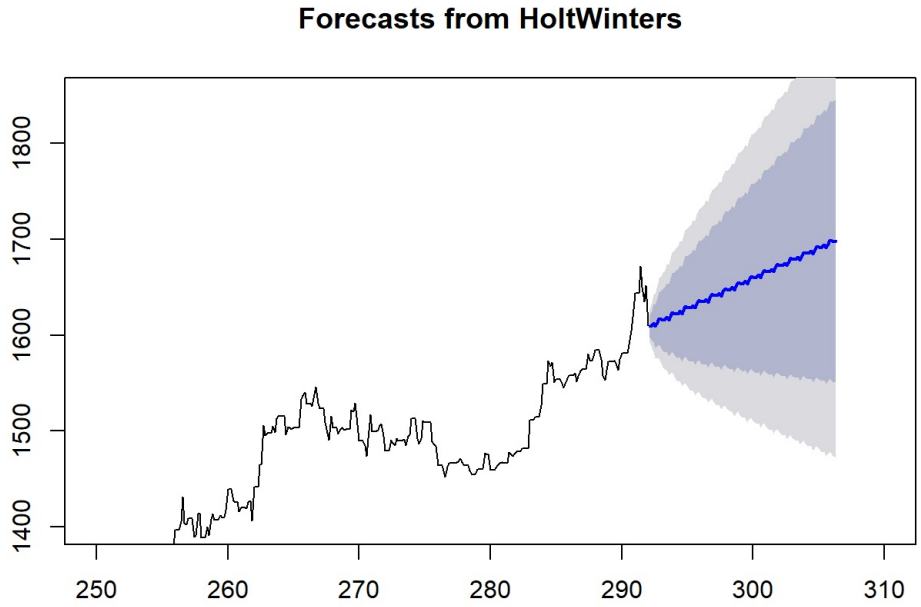
Well done! It fitted a model with beta, gama and 5 seasonal values as coefficients. I can see from this result that which month has greatest ridership by finding highest s level among periods.

Now I used this model to forecast prices of next 100 days.

```
plot(forecast(hw,100))
```




```
plot(forecast(hw,100),xlim=c(250,310),ylim=c(1400,1850))
```



From forecasts, it seems that the model predicts prices with an upward trend with small seasonality.

Now I can use this fitted model to forecast any specific time. Lets say, I want to forecast price of 1st of June 2020.

```
nrow(df2)
```

```
## [1] 2038
```

```
alpha <- hw$alpha
beta <- hw$beta
gamma <- hw$gamma
as.POSIXct(strptime(max(df2$dates), "%Y-%m-%d"))-as.POSIXct(strptime("2020-06-01", "%Y-%m-%d"))
```

```
## Time difference of -94 days
```

Now my dataset has prices for 2038 days. And I want to forecast 1st of June 2020, which is 94 days later from the last day of the dataset.

```
2038+94
```

```
## [1] 2132
```

```
94%%7
```

```
## [1] 3
```

So I can manually calculate it by using model coefficients above. Price at day 2132(2038+94) = Coef(a) + forecast_day*coef(b) + coef(s3)

I took into account coef of s3 by calculating the result of 94's modulo at base 7 that is 3. Since I determined 7 as the frequency of time series, it generates a circle of 7 day periods, so I just tried to find exact location of day 94 in that 7 day period process.

```
fc2132 <- 1610.2717875 + (94*0.8984085) + (-0.6778132)
```

Forecast of my fitted model will become 1694.044 USD for Ounce price at 1st of June 2020. I also checked it by running `forecast(hw,94)` and by looking the last prediction corresponds to day 94.

```
forecast(hw,94)
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 292.1429       1609.506 1598.202 1620.811 1592.218 1626.795
## 292.2857       1609.834 1594.090 1625.579 1585.755 1633.914
## 292.4286       1612.289 1593.064 1631.515 1582.886 1641.692
## 292.5714       1609.218 1587.015 1631.422 1575.261 1643.175
## 292.7143       1613.502 1588.643 1638.362 1575.483 1651.522
## 292.8571       1617.332 1590.044 1644.621 1575.598 1659.066
## 293.0000       1616.139 1586.592 1645.685 1570.951 1661.326
## 293.1429       1615.795 1583.972 1647.618 1567.126 1664.465
```

```
## 293.2857       1616.123 1582.294 1649.952 1564.386 1667.860
## 293.4286       1618.578 1582.832 1654.324 1563.909 1673.247
## 293.5714       1615.507 1577.919 1653.095 1558.020 1672.993
## 293.7143       1619.791 1580.425 1659.157 1559.586 1679.997
## 293.8571       1623.621 1582.533 1664.709 1560.782 1686.460
## 294.0000       1622.428 1579.666 1665.189 1557.030 1687.826
## 294.1429       1622.084 1577.579 1666.590 1554.019 1690.149
## 294.2857       1622.412 1576.319 1668.505 1551.919 1692.904
## 294.4286       1624.867 1577.221 1672.513 1551.999 1697.735
## 294.5714       1621.796 1572.628 1670.964 1546.600 1696.991
## 294.7143       1626.080 1575.418 1676.742 1548.599 1703.561
## 294.8571       1629.910 1577.779 1682.040 1550.183 1709.637
## 295.0000       1628.717 1575.141 1682.292 1546.779 1710.654
## 295.1429       1628.373 1573.277 1683.469 1544.111 1712.635
## 295.2857       1628.701 1572.203 1685.199 1542.295 1715.107
## 295.4286       1631.156 1573.274 1689.038 1542.633 1719.678
## 295.5714       1628.085 1568.835 1687.334 1537.471 1718.698
## 295.7143       1632.369 1571.768 1692.970 1539.687 1725.050
## 295.8571       1636.199 1574.260 1698.137 1541.472 1730.926
## 296.0000       1635.005 1571.742 1698.269 1538.253 1731.758
## 296.1429       1634.662 1570.002 1699.322 1535.773 1733.551
## 296.2857       1634.990 1569.031 1700.948 1534.114 1735.865
## 296.4286       1637.445 1570.198 1704.691 1534.600 1740.290
## 296.5714       1634.374 1565.849 1702.898 1529.574 1739.173
## 296.7143       1638.658 1568.865 1708.451 1531.919 1745.397
## 296.8571       1642.488 1571.435 1713.540 1533.822 1751.153
## 297.0000       1641.294 1568.991 1713.598 1530.716 1751.873
## 297.1429       1640.951 1567.326 1714.575 1528.352 1753.550
## 297.2857       1641.278 1566.419 1716.138 1526.791 1755.766
## 297.4286       1643.733 1567.646 1719.821 1527.368 1760.099
## 297.5714       1640.662 1563.354 1717.971 1522.429 1758.896
## 297.7143       1644.947 1566.423 1723.470 1524.855 1765.038
## 297.8571       1648.776 1569.043 1728.510 1526.835 1770.718
## 298.0000       1647.583 1566.646 1728.521 1523.800 1771.366
## 298.1429       1647.240 1565.031 1729.448 1521.513 1772.967
## 298.2857       1647.567 1564.166 1730.969 1520.016 1775.119
## 298.4286       1650.022 1565.432 1734.612 1520.653 1779.392
## 298.5714       1646.951 1561.177 1732.726 1515.771 1778.132
## 298.7143       1651.235 1564.281 1738.190 1518.250 1784.221
## 298.8571       1655.065 1566.934 1743.196 1520.281 1789.850
## 299.0000       1653.872 1564.569 1743.176 1517.294 1790.450
## 299.1429       1653.528 1562.988 1744.069 1515.058 1791.999
## 299.2857       1653.856 1562.150 1745.562 1513.604 1794.108
## 299.4286       1656.311 1563.443 1749.179 1514.282 1798.341
## 299.5714       1653.240 1559.213 1747.268 1509.437 1797.043
## 299.7143       1657.524 1562.340 1752.708 1511.953 1803.096
## 299.8571       1661.354 1565.016 1757.692 1514.017 1808.691
## 300.0000       1660.161 1562.671 1757.651 1511.063 1809.259
## 300.1429       1659.817 1561.113 1758.521 1508.863 1810.772
## 300.2857       1660.145 1560.294 1759.996 1507.437 1812.853
## 300.4286       1662.600 1561.605 1763.595 1508.141 1817.059
## 300.5714       1659.529 1557.391 1761.667 1503.322 1815.736
## 300.7143       1663.813 1560.534 1767.092 1505.861 1821.765
## 300.8571       1667.643 1563.224 1772.061 1507.949 1827.337
## 301.0000       1666.450 1560.893 1772.006 1505.015 1827.884
## 301.1429       1666.106 1559.352 1772.861 1502.839 1829.373
## 301.2857       1666.434 1558.545 1774.323 1501.432 1831.436
## 301.4286       1668.889 1559.867 1777.911 1502.154 1835.624
## 301.5714       1665.818 1555.663 1775.972 1497.351 1834.284
## 301.7143       1670.102 1558.817 1781.387 1499.906 1840.298
## 301.8571       1673.932 1561.516 1786.347 1502.007 1845.856
## 302.0000       1672.739 1559.194 1786.283 1499.087 1846.390
## 302.1429       1672.395 1557.663 1787.127 1496.927 1847.863
## 302.2857       1672.723 1556.863 1788.582 1495.531 1849.914
## 302.4286       1675.178 1558.192 1792.163 1496.264 1854.092
## 302.5714       1672.107 1553.995 1790.218 1491.471 1852.742
## 302.7143       1676.391 1557.154 1795.627 1494.034 1858.748
## 302.8571       1680.221 1559.859 1800.582 1496.144 1864.297
## 303.0000       1679.027 1557.542 1800.513 1493.231 1864.824
## 303.1429       1678.684 1556.017 1801.351 1491.081 1866.286
## 303.2857       1679.012 1555.222 1802.801 1489.691 1868.332
## 303.4286       1681.467 1556.554 1806.379 1490.429 1872.504
## 303.5714       1678.396 1552.360 1804.431 1485.641 1871.150
## 303.7143       1682.680 1555.522 1809.837 1488.209 1877.151
## 303.8571       1686.510 1558.230 1814.790 1490.322 1882.697
## 304.0000       1685.316 1555.914 1814.719 1487.413 1883.220
## 304.1429       1684.973 1554.393 1815.553 1485.268 1884.678
## 304.2857       1685.300 1553.599 1817.002 1483.880 1886.721
## 304.4286       1687.756 1554.932 1820.579 1484.619 1890.892
```

```
## 304.5714       1684.684 1550.739 1818.630 1479.832 1889.537
## 304.7143       1688.969 1553.901 1824.036 1482.400 1895.537
## 304.8571       1692.798 1556.608 1828.988 1484.514 1901.083
## 305.0000       1691.605 1554.293 1828.918 1481.604 1901.607
## 305.1429       1691.262 1552.773 1829.751 1479.461 1903.062
## 305.2857       1691.589 1551.978 1831.201 1478.072 1905.107
## 305.4286       1694.044 1553.310 1834.779 1478.809 1909.279
```

The result is same as calculated. This proves my manual calculation.

Lastly I would like to thank who reads this report and all kinds of suggestions/corrections/additions will be appreciated. And as Legal Disclaimer, this content is for informal purposes only, NO INVESTMENT ADVICE.

THANK YOU!