Time series modeling and forecasting of gold price

Ilaria Ferrero, Fabiana Pagliuca, Chiara Sangalli

2024-02-13

Contents

Introduction

In this report, we delve into the analysis of monthly gold price data. Gold has always attracted significant interest as a safe-haven asset and a barometer of economic stability. By examining historical price data, we aim to uncover trends, seasonal fluctuations, and potential anomalies that shape the gold market; moreover we will determine what SARIMA model best fits the data we collected and we will predict, using the previously estimated model, the future average monthly gold price.

We obtained our dataset from DataHub.io (here you can find the link) and we decided to focus on monthly average gold prices spanning a period of 20 years. The dataset comprehends observations from January 2000 to July 2020, for a total of 247 data points. *Figure 1* showcases the time plot of the gold price series: by observing it it's clear that the series is not stationary, because the mean is increasing over time, following an upward trend.

```
df <- read_excel("C:/Users/Lucia/Desktop/times series/monthly_csv.xls")
df_ts<- ts(df$Price,start=c(2000,1),frequency=12)
```

Preliminary Analysis

From the plot in *Figure 1* it appears clearly that the series is not stationary; this assumption is confirmed by the Augmented Dickey-Fuller test, that tests the null hypothesis that the time series has a unit root; the alternative hypothesis is that the series is stationary. In this case the test retrieves a **p-value** of 0.667, leading us to conclude that the series is non stationary.

```
# test to check the stationrity of the series:
adf.test(df_ts)
```

```
##
```

Figure 1: Time Plot of gold price

```
##  Augmented Dickey-Fuller Test
##
## data:  df_ts
## Dickey-Fuller = -1.7824, Lag order = 6, p-value = 0.6674
## alternative hypothesis: stationary
```

Figure 2 showcases a decomposition of the gold price data by breaking the series into four components. From top to bottom, the *observed* section shows the original gold price data; the *trend* component shows there was a general increase of the gold price over the period taken into consideration: more specifically, the price had been growing from 2000 to 2013, when it started to decrease briefly, remained constant for a while and eventually begun growing again; the last observed price, in July 2020 is the highest number registered in our series, meaning that the growth in the last period has surpassed the previous peak price registered in September 2011. The *seasonal* part of the decomposition shows a seasonal pattern, consisting of a regular rise and fall every year. The *random* component of the decomposition proves one again that the series is not a stationary process.

The last graphical evidence of the series's non stationarity is offered by the ACF and PACF plots, displayed in *figure 3*: as the we can see the auto correlations are way above the dotted threshold present in the plot, meaning that autocorrelations between observations are very high at every lag; on the contrary, the plot of the partial autocorrelation function shows that only the first value, corresponding to lag 0, is significant while the others are below the threshold: this suggests that there might be a first-order autocorrelation effect in the time series. In other words, there could be a direct dependency between an observation and the immediately succeeding one.

First-order difference

To transform our series in a stationary one, we can apply firstly a **first-order difference**: this means subtracting to each value in our series its previous one. The resulting series is displayed in *figure 4*: as we can see, the series does not have a constant mean over time and thus is not stationary in terms of the mean.

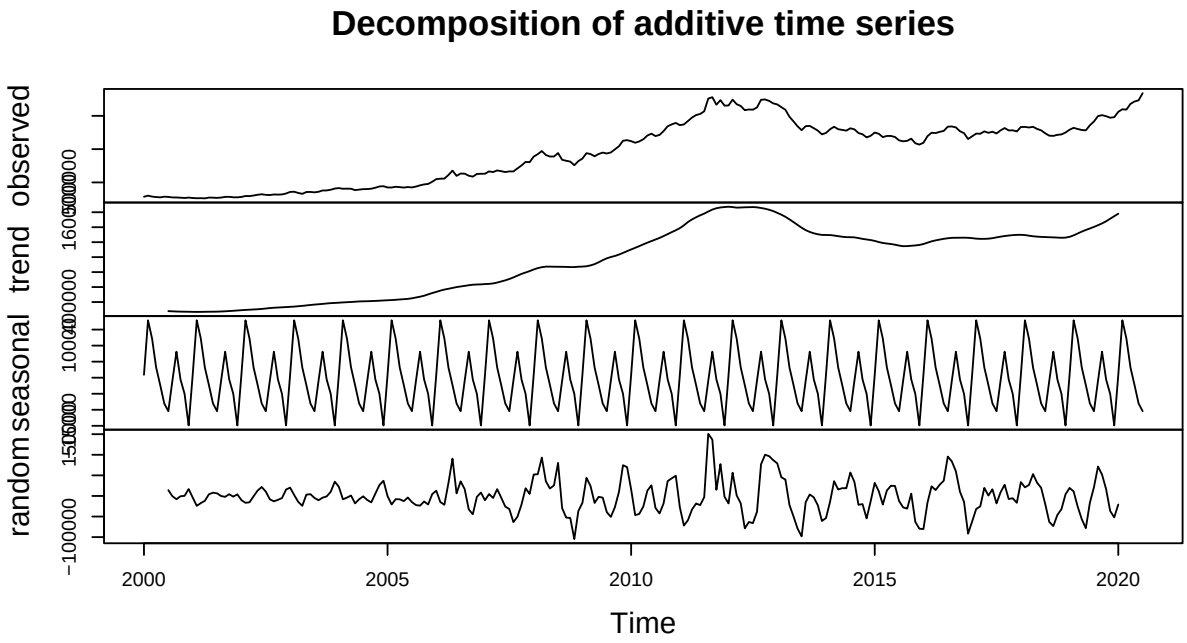

Figure 2: Time series decomposition

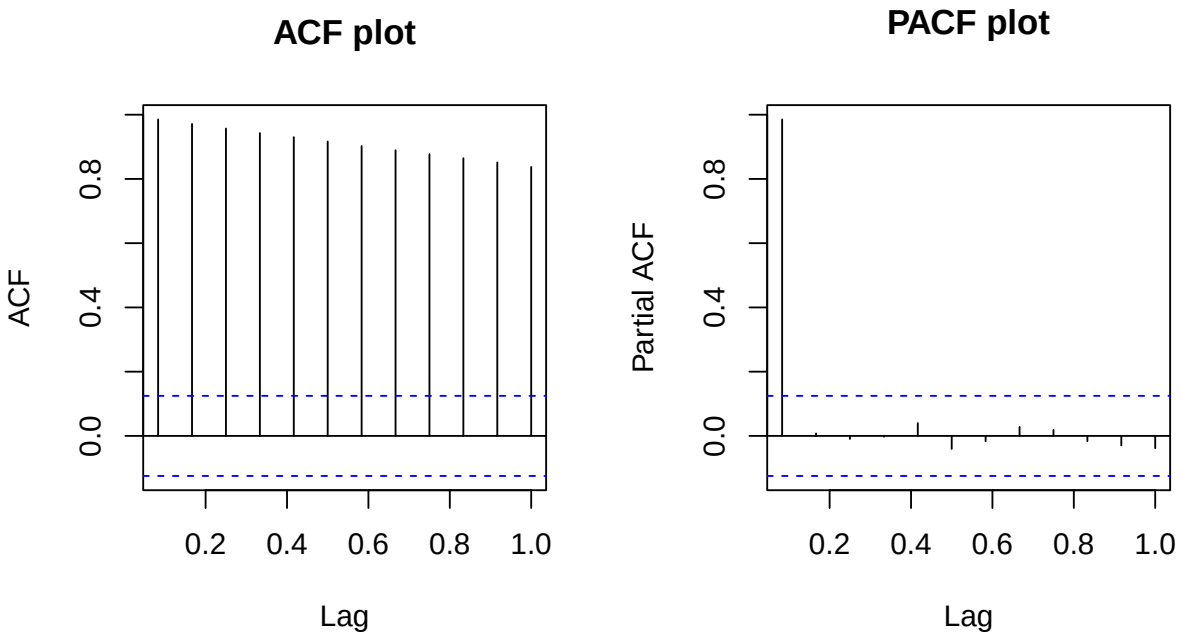

Figure 3: ACF and PACF plots of the original series

```r
#Compute the first-order difference
diff_ts <- diff(df_ts, lag = 12)
```

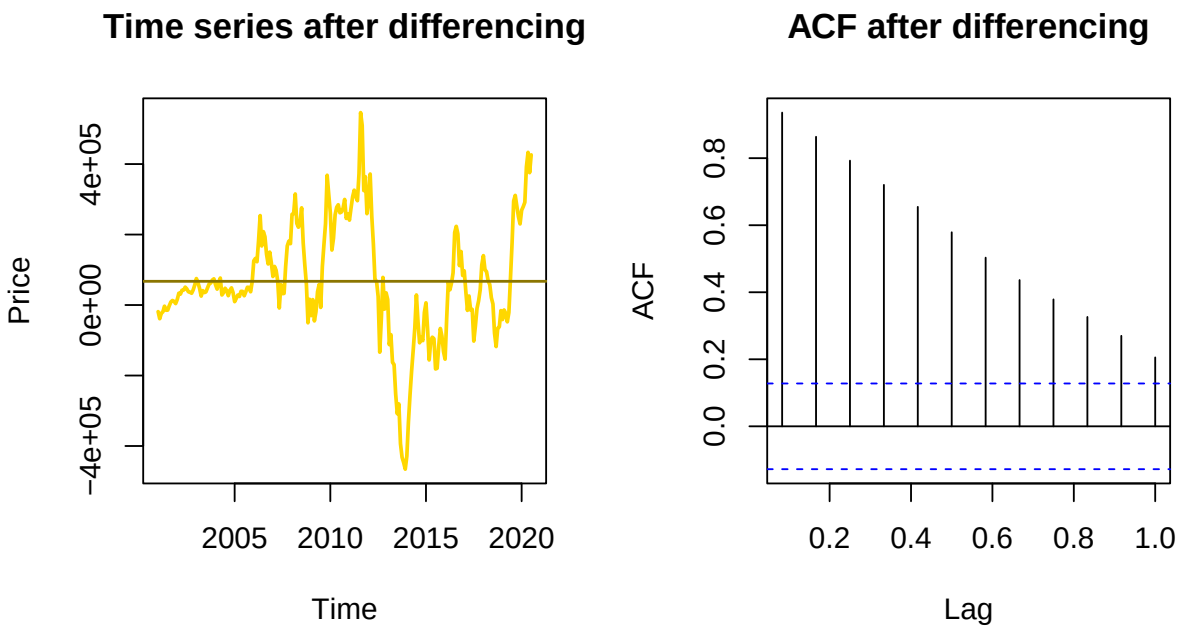

Figure 4: Gold price series and ACF after differencing

Figure 4 illustrates also the ACF plot with the first differencing on our time series, revealing autocorrelations that decrease slowly. When autocorrelations decrease slowly, it implies that successive observations in the series are still somewhat dependent on each other, violating the assumption of **stationarity**. Therefore, another differencing may be necessary to eliminate the residual autocorrelation and achieve stationarity.

The non-stationarity of this new series is confirmed by the high *p-value* of the **Dickey-Fuller test**, which leads us to accept the null hypothesis of non-stationarity.

```r
adf.test(diff_ts)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff_ts
## Dickey-Fuller = -2.8261, Lag order = 6, p-value = 0.2286
## alternative hypothesis: stationary
```

Second-order difference

Now, we performs a second differencing operation on the time series using a lag of 12 month. *Figure 4* shows that the series does not have a constant mean over time and thus is not stationary in terms of the mean: since our monthly dataset disoplayed a seasonal structure, we decided to apply a **seasonal difference of the first-order** with a lag of 12 months.

```r
# Compute the seasonal difference
seasonal_diff <- diff(diff_ts, lag = 12)
```

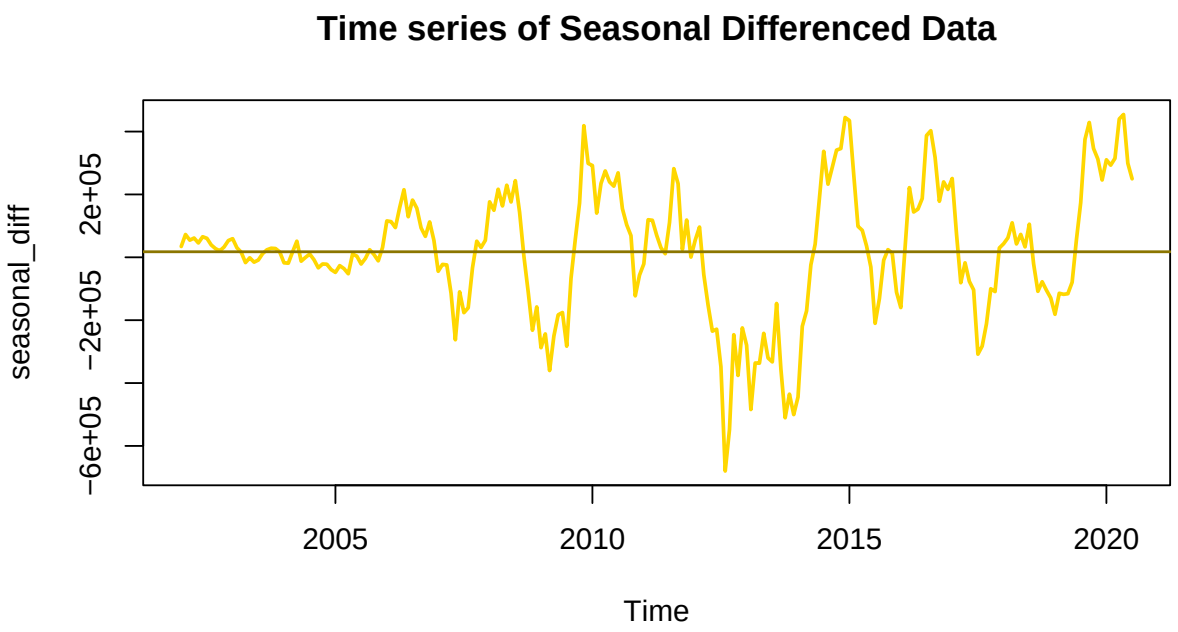

Figure 5: Time series of Seasonal Differenced Data

The **Augmented Dickey-Fuller** test performed on the seasonally differenced dataset shows a test statistic value of -4.8688, indicating a high degree of stationarity in the data. The associated p-value is 0.01, suggesting strong evidence against the null hypothesis of non-stationarity. Therefore, based on this test result, we can confidently conclude that the seasonally differenced dataset is stationary.

```
adf.test(seasonal_diff)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  seasonal_diff
## Dickey-Fuller = -4.8688, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

To support this conclusion, we can observe the **ACF** and the **PACF** plots of the seasonally differenced series in *Figure 6*. Looking at the seasonal peaks in the ACF plot, we can see that there is a high peak at lag 0, while at the subsequent seasonal lag the value is not equally high. As for the PACF plot, it is evident that there are two significant peaks at the seasonal lags. Regarding the non-seasonal lags, autocorrelation values are relevant for the first 6 lags, while partial autocorrelations are alway negilible. These observations will guide us in the choiche of the our SARIMA model's parameters **p, q, P** and **Q.**

SARIMA

Model selection and fitting

To effectively capture the patterns present in our observed time series, we will employ a **seasonal Autoregressive Integrated Moving Average** (**ARIMA**) model. Seasonal ARIMA models are well-suited for analyzing data exhibiting recurring patterns or seasonal fluctuations over time. By incorporating both autoregressive and moving average components along with differencing operations to stabilize non-stationary

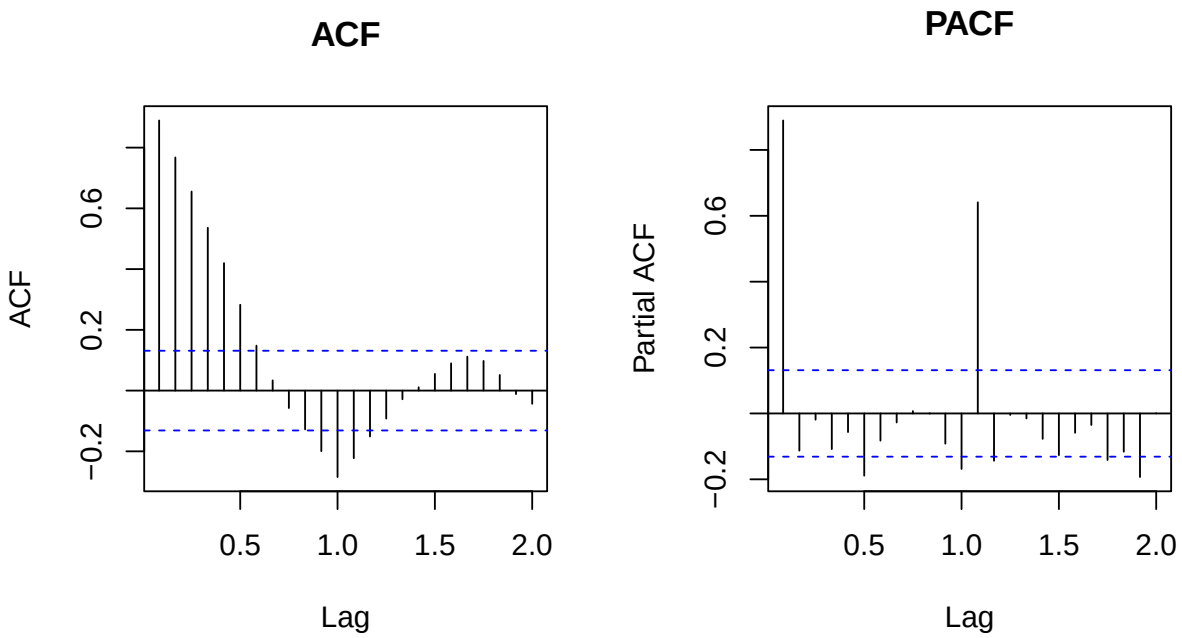

Figure 6: ACF of seasonally differenced data

data, this approach enables us to model and forecast the seasonal behavior present in our time series.

As previously observed, applying first-order differencing and seasonal first-order differencing effectively transforms our non-stationary series into a stationary one; for this reason we will fix both differencing parameters **d** and **D** equal to 1.

```
# fixed values for d and D
d <- 1
D <- 1
```

Regarding the seasonal parameters P and Q, we can conclude the following: given the two significant peaks in the PACF plot, we will choose $P = 2$, while considering the ambiguity of the ACF plot, we will have to decide whether the most appropriate value for Q will be 1 or 2. Moving on to the non-seasonal parameters, from the PACF plot, it is clear that $p = 0$, while once again we will need to determine the best value for q.

The seasonal period **s** of 12 is chosen to reflect the seasonality observed in *figure 2*, that suggests that there are seasonal effects that occur on a yearly basis.

```
# possible values for our parameters
p <- 0
q <- 0:5
P <- 2
Q <- 1:2

# seasonality
s <- 12
```

To test all the possible models, we will create a grid containing all the possible combination of values:

```
# grid of parameter combinations
param_grid <- expand.grid(p = p, d = d, q = q, P = P, D = D, Q = Q)
```

```r
# matrix to store results in
results_1 <- matrix(NA, nrow = nrow(param_grid), ncol = 2)
colnames(results_1) <- c("AIC", "BIC")
```

To determine the best model we will rely on two different indicators, the **Akaike Information Criterion** (AIC) and the **Bayesian Information Criterion** (BIC): both these criteria balance the goodness of fit of the model with the complexity of the model, penalizing overly complex models to prevent overfitting.

More specifically, the AIC can be computed as follows:

$$AIC = 2H - 2\,ln(\mathcal{L})$$

where H indicates the number of parameters in the model and $\mathcal{L}$ is the likelihood function, of which we take the logarithmic form.

The BIC is given by:

$$BIC = H\,ln(n) - 2\,ln(\mathcal{L})$$

In both cases, the best model is the one for which these quantities are smaller; for $n > 8$, the penalty introduced by the BIC ($H\,ln(n)$) is bigger than the one introduced by the AIC: for this reason the BIC is said to be more conservative than the AIC.

```r
# iterate through all the possible parameter combinations
for (i in 1:nrow(param_grid)) {

  # SARIMA model
  model <- Arima(df_ts,
                 order = c(p, d, param_grid$q[i]),
                 seasonal = list(order = c(P, D,
                 param_grid$Q[i]), period = s))

  # AIC and BIC for the model
  aic <- AIC(model)
  bic <- BIC(model)

  # store results
  results_1[i, 1] <- aic
  results_1[i, 2] <- bic
}

results_1 <- cbind(param_grid, results_1)
```

The AIC and the BIC obtained by every model are displayed in *table 1*: the next step is to select the best model, that will be the one with the lowest AIC and BIC.

```r
kable(results_1, caption = "AIC and BIC for every possible combination of parameters")
```

Table 1: AIC and BIC for every possible combination of parameters

p	d	q	P	D	Q	AIC	BIC
0	1	0	2	1	1	5652.987	5666.808
0	1	1	2	1	1	5645.586	5662.862
0	1	2	2	1	1	5647.583	5668.315
0	1	3	2	1	1	5648.359	5672.546

p	d	q	P	D	Q	AIC	BIC
0	1	4	2	1	1	5648.885	5676.528
0	1	5	2	1	1	5647.767	5678.865
0	1	0	2	1	2	5654.906	5672.182
0	1	1	2	1	2	5647.343	5668.075
0	1	2	2	1	2	5649.342	5673.529
0	1	3	2	1	2	5650.191	5677.833
0	1	4	2	1	2	5650.691	5681.789
0	1	5	2	1	2	5649.572	5684.125

```r
# best model AIC
model_AIC_id <- which.min(results_1$AIC)
model_AIC <- results_1[model_AIC_id, 1:6]
model_AIC
```

```
##   p d q P D Q
## 2 0 1 1 2 1 1
```

```r
# best model BIC
model_BIC_id <- which.min(results_1$BIC)
model_BIC <- results_1[model_BIC_id, 1:6]
model_BIC
```

```
##   p d q P D Q
## 2 0 1 1 2 1 1
```

According to both critera, the best model has $q = 1$ and $Q = 1$, so our final model will be a seasonal **ARIMA(0,1,1)(2,1,1)[12]**: it incorporates both seasonal and non-seasonal differencing of order 1, a moving average term of order 1 (**MA(1)**), a seasonal autoregressive term of order 2 (**SAR(2)**) and a seasonal moving average term of order 1 (**SMA(1)**).

```r
mod_1 <- Arima(df_ts,
               order = c(0,1,1),
               seasonal = list(order = c(2,1,1), period = s))
summary(mod_1)
```

```
## Series: df_ts
## ARIMA(0,1,1)(2,1,1)[12]
##
## Coefficients:
##          ma1    sar1    sar2     sma1
##       0.2081  0.0175  0.0557  -1.0000
## s.e.  0.0668  0.0697  0.0685   0.1264
##
## sigma^2 = 1.479e+09:  log likelihood = -2817.79
## AIC=5645.59   AICc=5645.85   BIC=5662.86
##
## Training set error measures:
##                     ME    RMSE      MAE       MPE     MAPE      MASE
## Training set 1652.399 37113.3 26177.13 0.2361659 2.636456 0.1993544
##                      ACF1
## Training set -0.0004589513
```

Diagnostics

Once the model is fitted, we can conduct diagnostic checks on the fitted model to evaluate its adequacy: we will focus on residual analysis to assess whether the underlying assumptions of normality and independence holds; in other words we will study the residuals behavior to ensure that they follow a Gaussian distribution and exhibit no systematic patterns.

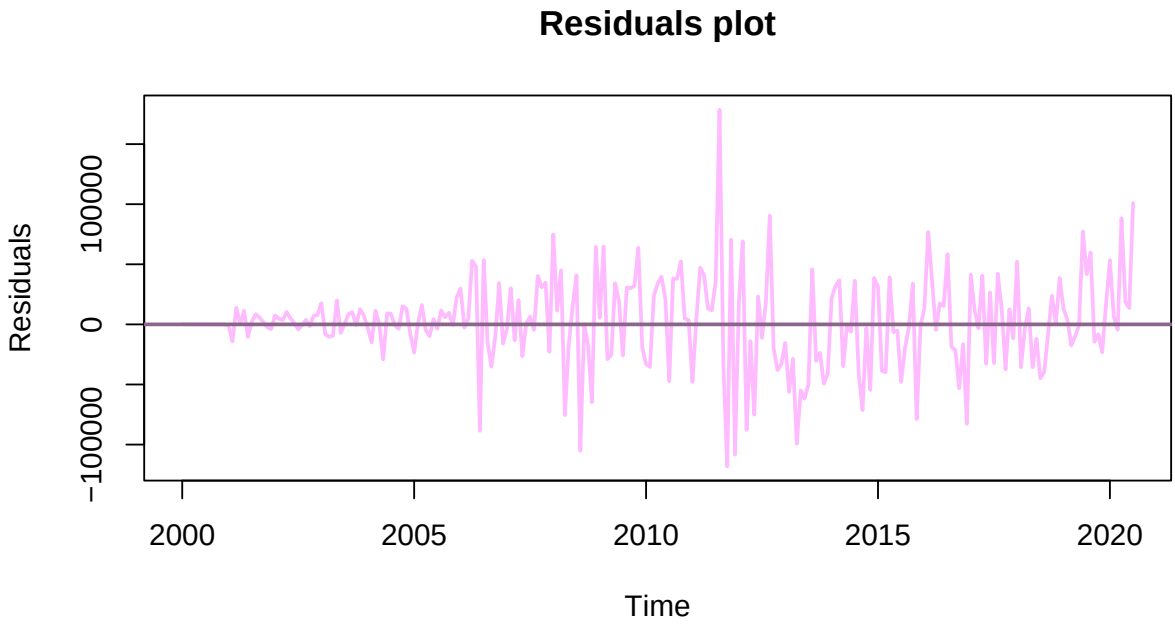

Figure 7: Plot of residuals against time

The plot of residuals against time represented in *figure 7* shows how our model's residuals move around 0 but with non-constant variance: the variance's increase appears clearly after 2005 and reaches its peak around 2012; moreover, a slight upward trend is clearly visible after 2020.

To test normality we will rely on both the Shapiro test and different graphical tools: The Shapiro test is used to assess whether a sample of data comes from a normally distributed population: its null hypothesis is that the data are normally distributed, so if the p-value associated with the test statistic is lower than the significance level 0.05, then the null hypothesis is rejected, suggesting that the data do not follow a normal distribution.

In this case, the low p-value obtained form the test and the plots in *figure 8* lead us to conclude that residuals are not normally distributed.

```
shapiro.test(mod_1$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  mod_1$residuals
## W = 0.96277, p-value = 5.017e-06
```

A possible way to solve the non-normality issue is to apply the logarithmic transformation to the time series before fitting the model:

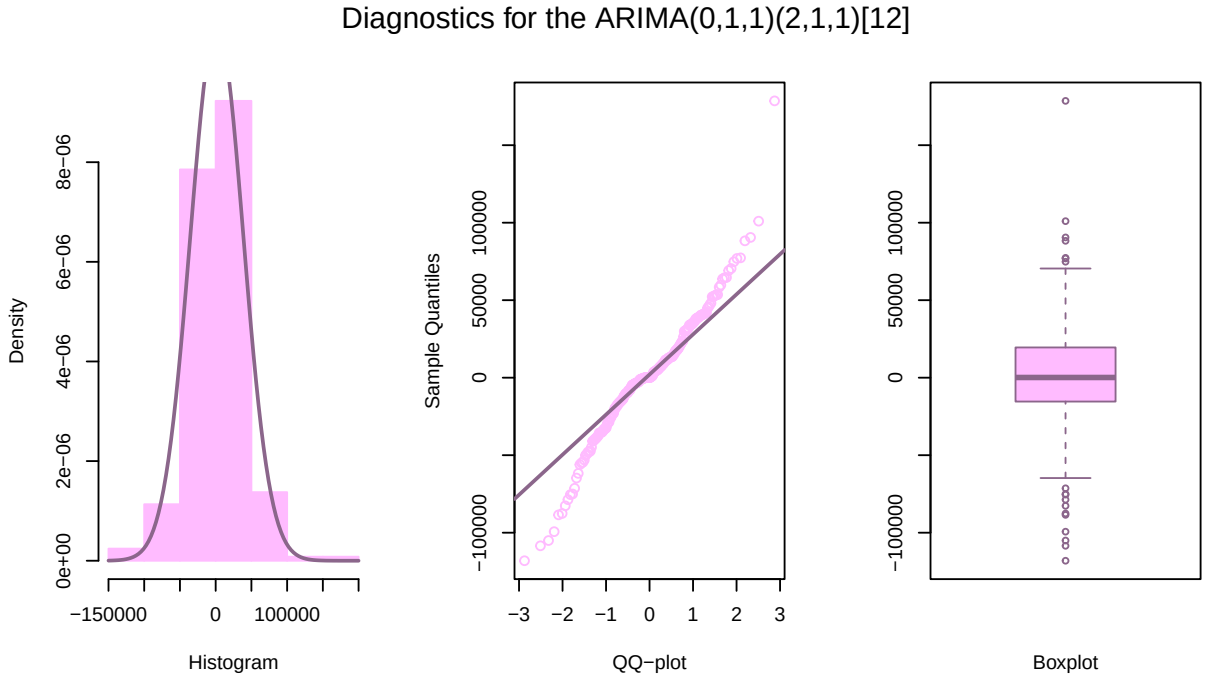

Figure 8: Histogram, QQ-plot, and boxplot to chech for residuals' normality

```
mod_2 <- Arima(log(df_ts),
               order = c(0,1,1),
               seasonal = list(order = c(2,1,1), period = s))
```

After the logarithmic transformation is applied, the plot showcasing residuals against time (*figure 9*) no longer presents the eteroschedasticity issue that was previously observed; the p-value of the Shapiro test being higher then 0.05 as well as the graphical representation of residuals' distribution in *figure 10* lead us to conclude that the model's residuals follows a Normal distribution. To sum up, the logarithmic transformation we applied managed to resolve both the eteroschedasticity and the non-normality issues.

```
shapiro.test(mod_2$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  mod_2$residuals
## W = 0.9886, p-value = 0.04811
```

Finally, we need to assess the independence of the residuals. *Figure 11* displays the autocorrelations of the residuals from our model, which are negligible at all lags. This indicates that our residuals meet the independence assumption, because there is no systematic relationship between the residuals at different time points.

We can finally fit the ARMA(0,1,1)(2,1,1)[12] with the logarithmic transformation:

```
summary(mod_2)
```

```
## Series: log(df_ts)
## ARIMA(0,1,1)(2,1,1)[12]
##
```

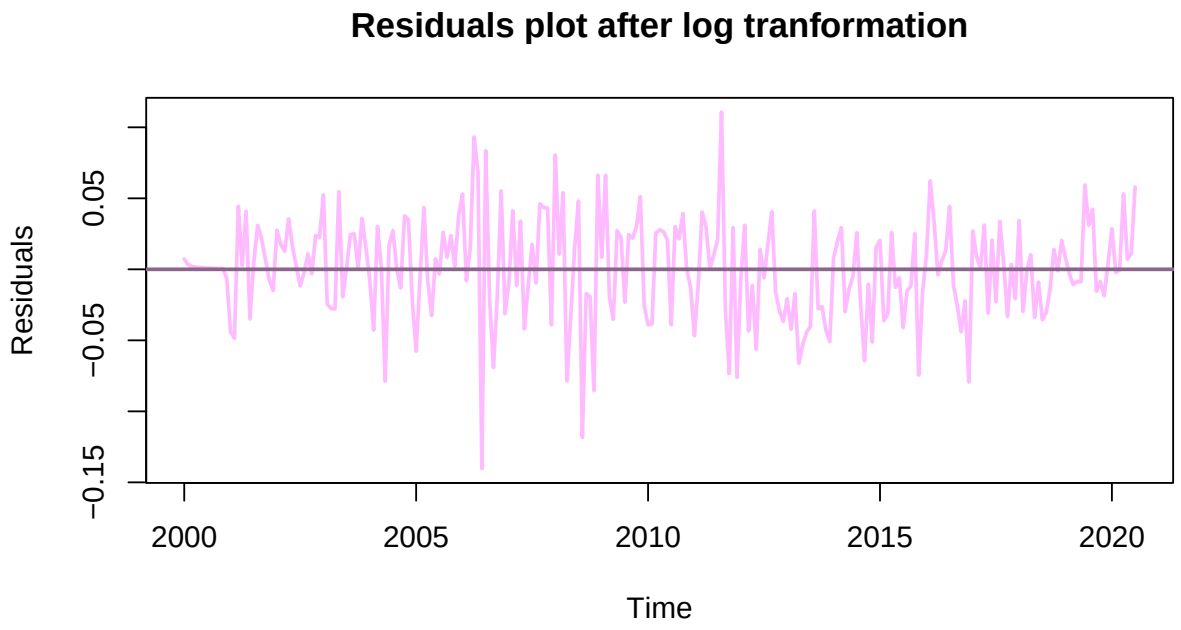

Figure 9: Plot of residuals against timme after the logarithmic transformation

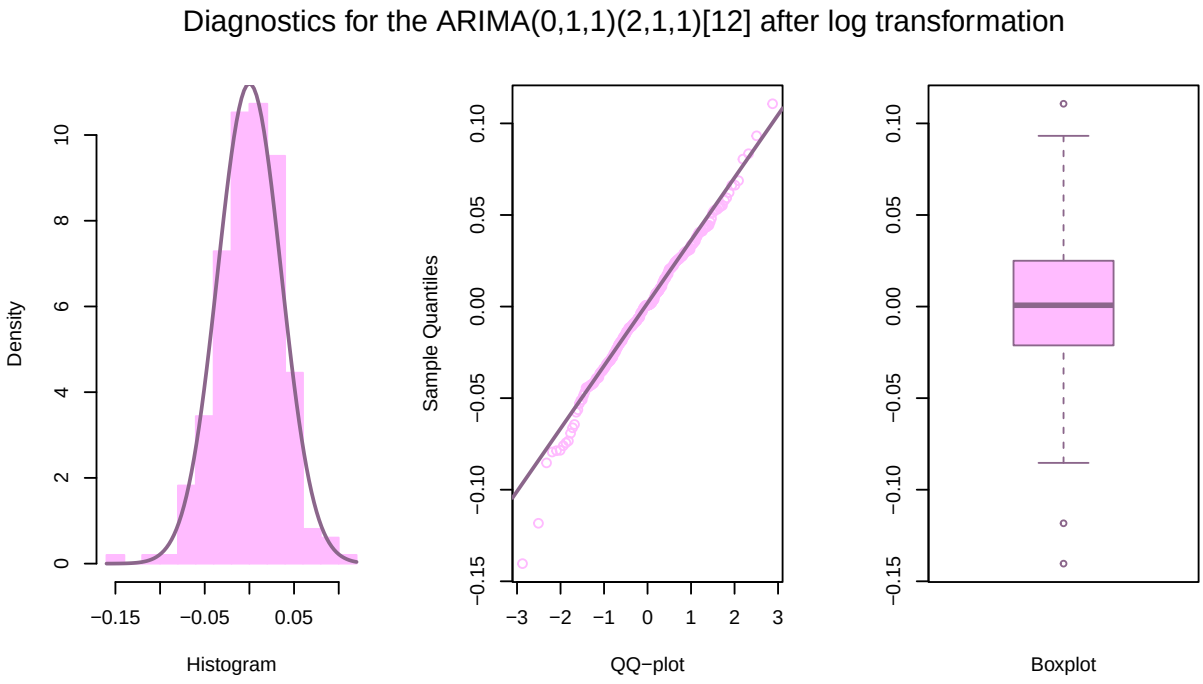

Figure 10: Histogram, QQ-plot, and boxplot to assess the normality of residuals after applying the log transformation

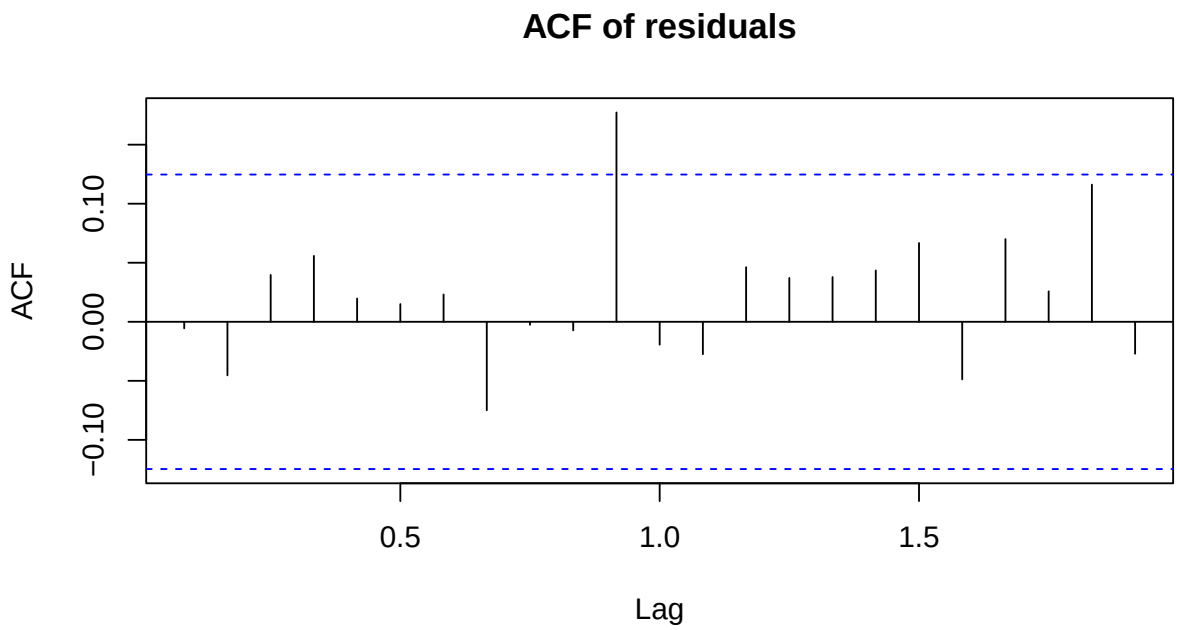

Figure 11: ACF of residuals

```
## Coefficients:
##          ma1     sar1    sar2     sma1
##       0.1569  -0.0269  0.0325  -0.9996
## s.e.  0.0702   0.0690  0.0674   0.1647
##
## sigma^2 = 0.001357:  log likelihood = 425.18
## AIC=-840.35   AICc=-840.09   BIC=-823.07
##
## Training set error measures:
##                       ME       RMSE        MAE         MPE      MAPE      MASE
## Training set 6.331395e-05 0.03555013 0.02743969 0.001412522 0.2009565 0.1972498
##                      ACF1
## Training set -0.005562308
```

Forecasts

After fitting our SARIMA model on the historical time series data, the goal is to use this model to predict the values of the gold price at future points. So, in this step our aim is predicting future values based on past observations and historical patterns. We start with creating a training subset, that we can use to train our forecasting model.

```
train_ts <- window(df_ts, end=c(2019,07))
```

The `window()` function is used to subset the time series. In this context, it's creating a training subset (train_ts) that includes all observations up to June 2019. We decided to keep the entire series up to 2019, excluding the last year and reserving it for testing or validation purposes: this is because we will predict the last year based on the data up to 2019 so that we can compare the predicted data with the real data.

In the following code, a SARIMA(0,1,1)(2,1,1)[12] model is fitted on the logarithmically transformed training

data (train_ts), excluding the last 12 values. The subsequent 12-month ahead predictions are made.

```
# Fit SARIMA model excluding the last 12 values
mod_3 <- Arima(log(train_ts),
                order = c(0, 1, 1),
                seasonal = list(order = c(2, 1,
                1), period = s))

# Make 12-month ahead predictions
pred <- forecast(mod_3, h = 12)
```

We plot the predicted values along with the corresponding true values in *figure 12*:

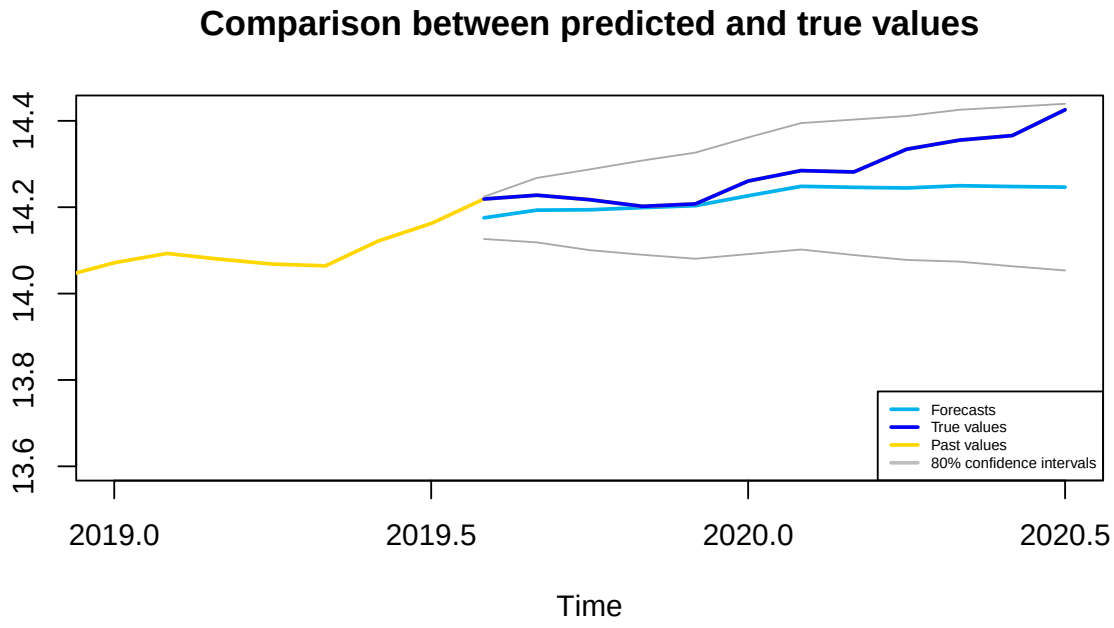

Figure 12: Forecasts from ARIMA (0,1,1)(0,1,1)[12]

The light blue line represents the 12-month predictions generated by the model. These predictions indicate the model's estimate for future values of the time series based on the training data. The blue line represents the actual values of the time series: these are the real data that the model hasn't seen during training and were excluded from the training period to evaluate the predictive capability of the model. The forecast lines (light blue) closely follow the real values (blue), so the model is making accurate predictions. The gray lines represents 80% confidence intervals.

Finally we can predict future values for the next 4 years: the predictions are displayed in *figure 13* and we can see that they reflect the upward trend that our series showcased in the last period, but with a smaller steapness.

Conclusions

The model that fits better the monthly gold prices is a seasonal **ARIMA(0,1,1)(2,1,1)[12]**: we have reached this result by examining the ACF and PACF plots of our differenced series and comparing various models with different parameter sets.

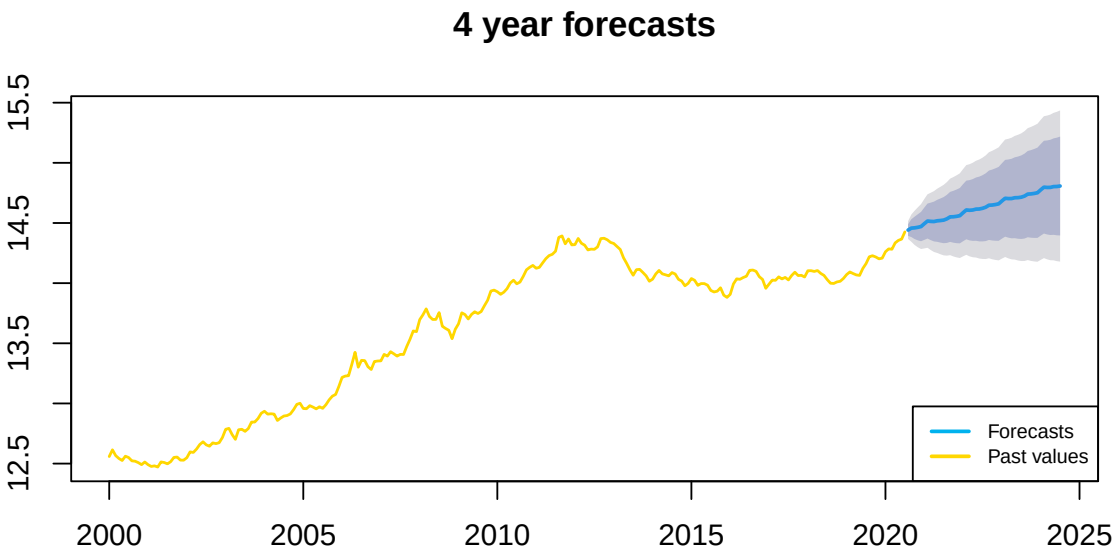

Figure 13: Predictions for future values

The model we fitted takes into account both the trend and seasonality in the time series data; in order to respect the normality and independence assumptions we transformed our series by applying the logarithmic transformation and we were able to predict future gold price values.