A Time Series Analysis of Gold Prices

Abstract: This project analyzes the time series behavior of gold prices using advanced statistical modeling techniques. The goal is to model and forecast gold price movements, accounting for both the mean and volatility dynamics. The analysis begins with an ARIMA (Autoregressive Integrated Moving Average) model to capture the linear dependencies and trends in the gold price series. After identifying the optimal ARIMA model using autocorrelation and partial autocorrelation functions, the residuals are examined for volatility clustering, a common characteristic of financial time series. To model this volatility, a GARCH (Generalized Autoregressive Conditional Heteroskedasticity) model is fitted to the residuals of the ARIMA model. The GARCH model captures the time-varying volatility and provides insights into the risk associated with gold price fluctuations. The performance of the models is evaluated using diagnostic checks, including the Ljung-Box test for autocorrelation and the McLeod-Li test for ARCH effects. Finally, the fitted models are used to generate forecasts for future gold prices, along with confidence intervals to quantify uncertainty. The results demonstrate the effectiveness of combining ARIMA and GARCH models for analyzing and forecasting gold prices, providing valuable insights for investors and policymakers.

Nabeel Vakil

PSTAT 174 – Final Project

2025-03-21

1. Introduction

In this project, I will be working with a gold price dataset that contains information about gold prices from 2000 to 2021. I will be fitting the data to an (S)ARIMA model and then a GARCH model since GARCH models are generally effective at working with financial data, which proved to be the case here as well.

The purpose of this project is to better understand gold price dynamics, helping to identify possible trends, patterns, and behaviors in the gold market, since gold prices tend to be influenced by factors such as global economic conditions, inflation, currency fluctuations, and geopolitical events.

This project is important for me because it allows me to analyze the price of gold, considered to be a "safe-haven" asset, during different times in recent history. From this, we can evaluate how its value increases or decreases depending on economic conditions, for example, and also predict future prices of gold.

After doing some research, it appears that past studies on gold price datasets (including the 2000–2021 period) have provided valuable insights into price dynamics, volatility, and the role of gold in financial markets. Some key findings are that gold prices are influenced by a combination of macroeconomic factors, investor behavior, and geopolitical events; volatility in gold prices exhibits clustering and asymmetry, making GARCH-family models particularly useful; and gold serves as a safe-haven asset and a hedge against inflation, but its effectiveness can vary depending on the economic context.

An important discovery from the specific dataset I am working with is that gold prices seemed to follow a roughly positive linear trend over time up until about 2013 when they seemed to plummet for the next several years. After doing some research, I found that after a 12-year bull run, gold prices experienced a significant drop in 2013, particularly in April during the second quarter. This was primarily due to the anticipation of Fed tapering, meaning that it would cut back on its bond purchases and hence inject less liquidity into the financial system; expectations of higher interest rates, the U.S. dollar becoming stronger, which made gold less attractive and more expensive for holders of other currencies; and an improving global economic outlook, which reduced the appeal of gold as a safe-haven asset. Furthermore, reduced investment demand and speculative selling exacerbated this decline.

I came across this discovery after viewing the dataset in R and doing some further research online. With this information, though, I decided to safely cut the data so that it went until February 7th, 2013. I chose this date because that is where I got to 3100 observations in the time series, which seemed like a good time index to slice the dataset. This is because after this point roughly, the crash in gold prices took away the fairly positive linear trend that gold prices experienced before then.

2. Data

As aforementioned, the dataset I have chosen for this final project is one of gold prices, containing its open price, high price, low price, close price (adjusted for splits), adjusted close price (adjusted for splits and dividend and/or capital gain distributions), and volume. The dataset starts on August 30, 2000, and ends on July 16, 2021, but as mentioned above, I have filtered the data so that it goes from August 30, 2000, to February 7th, 2013. Furthermore, the frequency of the dataset is daily, with the units being U.S. dollars and the values being non-negative. The sample size of the entire time series dataset was 5317, but it became 3100 after slicing it.

The dataset is important to me because it can help me reach the goals of the project, which are to fit models to analyze the dynamics of price of gold, helping to identify possible trends, patterns, and behaviors in the gold market, and evaluate how these models do at forecasting future gold price data.

The purpose of studying this dataset is to use it to come up with my own findings for gold prices and see if they agree with findings from past research that has been done. Additionally, I hope to gain some valuable time series analysis skills by utilizing this dataset.

I obtained the data from Kaggle. Here is a link to the web page: https://www.kaggle.com/datasets/komalkhetlani/gold-prices-historical-data. As can be seen, the data was originally collected from Yahoo Finance by a user named Komal Khetlani.

Important to note is that as part of cleaning the dataset, I came up with an appropriate univariate time series to analyze since the original data consisted of open, high, low, close, and adjusted close prices. I did this by taking a midpoint of the open, high, low, and close prices to get an average price variable.

The first 10 rows of the cleaned dataset can be seen below. This was before I eliminated the other price columns and converted the dataset into a time series object.

A tibble: 10 × 8

date <date>	open <dbl>	high <dbl>	low <dbl>	close <dbl>	adj_close <dbl>	volume <dbl>	avg_price <dbl>
2000-08-30	273.9	273.9	273.9	273.9	273.9	0	273.90
2000-08-31	274.8	278.3	274.8	278.3	278.3	0	276.55
2000-09-01	277.0	277.0	277.0	277.0	277.0	0	277.00
2000-09-05	275.8	275.8	275.8	275.8	275.8	2	275.80
2000-09-06	274.2	274.2	274.2	274.2	274.2	0	274.20
2000-09-07	274.0	274.0	274.0	274.0	274.0	125	274.00
2000-09-08	273.3	273.3	273.3	273.3	273.3	0	273.30
2000-09-11	273.1	273.1	273.1	273.1	273.1	0	273.10
2000-09-12	272.9	272.9	272.9	272.9	272.9	0	272.90
2000-09-13	272.8	272.8	272.8	272.8	272.8	0	272.80

10 rows

3. Methodology

3.1 SARIMA (p, d, q) x (P, D, Q) model

The first model I tried fitting to the data was a SARIMA (p, d, q) x (P, D, Q) model. A SARIMA (Seasonal Autoregressive Integrated Moving Average) model is an extension of the ARIMA (Autoregressive Integrated Moving Average) model that explicitly incorporates seasonality in time series data. However, the gold price data did not exhibit any seasonality, so I just ended up fitting an ARIMA(p, d, q) model. ARIMA models combine three key components to model and predict time series data: Autoregression (AR), Differencing (I), and Moving Average (MA), where p denotes the order of the autoregressive term (number of lagged observations), d denotes the degree of differencing (number of times the data is differenced to achieve stationarity), and q denotes the order of the moving average (number of lagged forecast errors or white noise terms). ARIMA modeling requires six steps: 1) plot the time series data, 2) transform the data if needed (typically log and Box-Cox power transformations), 3) identify the difference d, 4) estimate the parameters, 5) perform diagnostics, and 6) select the best model.

3.2 GARCH(p, q) model

After fitting the ARIMA model, I fit a GARCH (Generalized Autoregressive Conditional Heteroskedasticity) model since it is known to work well with financial data. GARCH (p, q) models are used to model and forecast volatility in time series data. They assume that the log returns are conditionally normally distributed given the time-varying volatility, where the volatility depends on past squared residuals (ARCH terms) and past conditional variances (GARCH terms). It is an extension of the ARCH (Autoregressive Conditional Heteroskedasticity) model and is widely used to capture the clustering of volatility (i.e., periods of high volatility followed by high volatility and low volatility followed by low volatility). In the GARCH model, p denotes the order of the ARCH terms (lagged squared residuals), and q denotes the order of the GARCH terms (lagged conditional variances). The steps to GARCH modeling are to 1) fit a mean model (e.g., ARIMA) to the time series (to remove any trends or dependencies in the mean and extract the residuals), 2) plot the residuals (to visually inspect for volatility clustering) and perform statistical tests (to verify heteroskedasticity), 3) examine the ACF (Autocorrelation Function) and PACF (Partial Autocorrelation Function) of the squared residuals (to identify potential values for p and q), 4) estimate the parameters, 5) perform diagnostics, and 6) select the best model.

4. Results

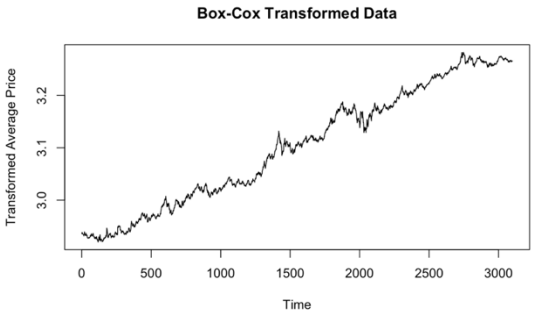
4.1 Results from SARIMA (p, d, q) x (P, D, Q) model

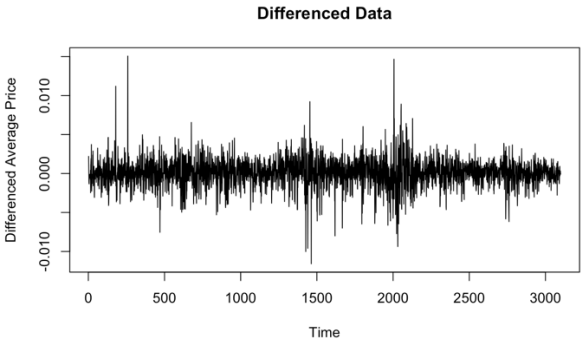
As aforementioned, I decided to go for an ARIMA(p, d, q) model since the gold price data did not exhibit seasonality. After going through the steps mentioned above, I arrived at the optimal choice of an ARIMA(1, 1, 0) model, which would just be an AR(1) model with a difference of 1. I will also delineate the result of each of these steps below.

Step 1 involved plotting the time series data to understand its underlying structure and help decide on appropriate modeling techniques. After cleaning the data, I created an xts (Extensible Time Series) object and obtained an initial plot of the data, which is shown to the right.

As seen to the left, I did the same for a regular ts (Time Series) object as well to get the x-axis indexed in terms of a variable t starting at 1 for the first observation and ending at 5317 for the last observation. These graphs are what initially indicated that I should cut the data up to about 2013 or the 3100th observation.

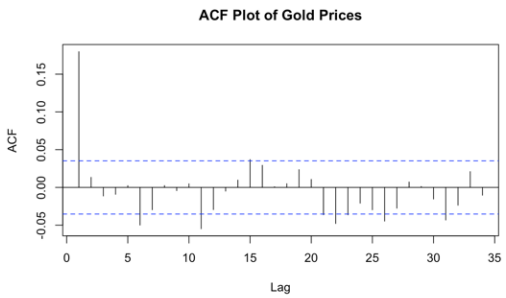
I then did a Box-Cox transformation, which included finding the optimal lambda. This was done to help stabilize the variance and make the data more suitable for modeling. For my data, after reducing it to 3100 observations, my optimal lambda came out to be approximately -0.2626. To the right is how the data looked after this optimal Box-Cox transformation.





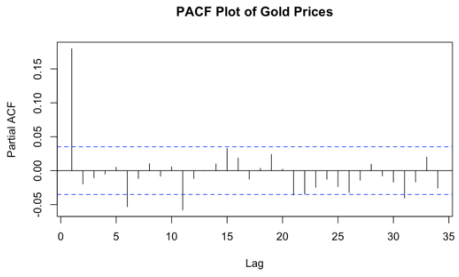
The third step involved differencing the data. First-order differencing removes the linear trend while higher-order differencing can remove more complex trends. For my data, it was just essential to do first-order differencing, which can be calculated as $\Delta y_t = y_t - y_{t-1}$, where y_t is the value at time t and y_{t-1} is the value at the previous time step $t - 1$. The data then looked like the picture to the left.

After this step, I checked the ACF and PACF plots, which are essential tools in time series analysis since they are used for selecting and fitting appropriate models. Autocorrelation measures the correlation between a time series and its lagged values, with ACF plots and PACF plots helping to visualize these correlations at different lags. ACF plots show the correlation between different lags for all the lags, and they help identify the moving average (MA) components in the data. On the other hand, PACF plots show the correlation between different lags after removing the influence of intermediate lags for all the lags, and they help identify the autoregressive (AR) components in the data. For the AR(p) model, the PACF plot cuts off after lag p, indicating the order of the autoregressive component, and for the MA(q) model, the ACF plot cuts off after lag q, indicating the order of the moving average component. Additionally, these plots can check for stationarity and seasonality. Here is how the ACF and PACF plots looked for my data.



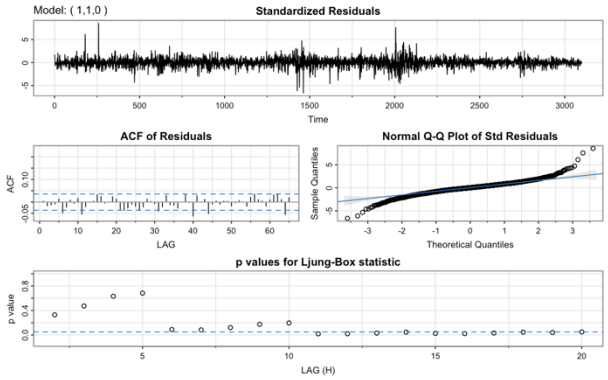
In the ACF plot to the left, the first lag shows a significant spike, which means the values at lag 1 are correlated with the current values. After lag 1 however, a good amount of the correlations fall within the confidence interval, suggesting that past gold prices have little to no predictable pattern. This also suggests an order of 1 for the moving average component (MA) or that an MA(1) model could be appropriate for modeling the series, assuming the data is stationary.

In the PACF plot to the right, the first lag is significant, confirming a direct relationship of the values at lag 1 with the current values. Unlike the ACF, the PACF plot shows how much of the correlation is not accounted for by previous lags. Beyond the first lag, the correlations appear largely insignificant. This significant spike at lag 1 and mostly none thereafter suggests an order of 1 for the autoregressive (AR) component or that an AR(1) model could be appropriate for modeling the series, assuming the data is stationary.



With this information, it was time for me to fit different models for model selection, estimate the parameters for each model, perform diagnostics for each model, and ultimately select the optimal model. For me, I decided to fit three models: an ARIMA(1, 1, 1) model, an ARIMA(0, 1, 1) model, and an ARIMA (1, 1, 0) model. I decided to fit the ARIMA(0, 1, 1) and ARIMA(1, 1, 0) models after I looked at the parameter estimates for the ARIMA (1, 1, 1) model, where I found none of my coefficients to be significant. After fitting the ARIMA(0, 1, 1) model, it provided a slightly higher AICc but with significant components. It was the same result for the ARIMA(1, 1, 0) model. In the end, however, I decided to go for the ARIMA(1, 1, 0) model because it had a slightly better AICc than the ARIMA(0, 1, 1) model. Their diagnostic plots were essentially the same though, so I feel like choosing one model over the other for the ARIMA(1, 1, 0) and ARIMA(0, 1, 1) models would not have changed much. The

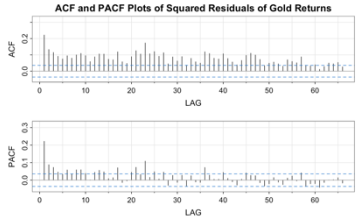
formula for my ARIMA(1, 1, 0) model is given by: $(1 - \phi B)(1 - B)^1 x_t = z_t$. Here is how the diagnostic plots looked for the ARIMA(1, 1, 0) model that I selected.



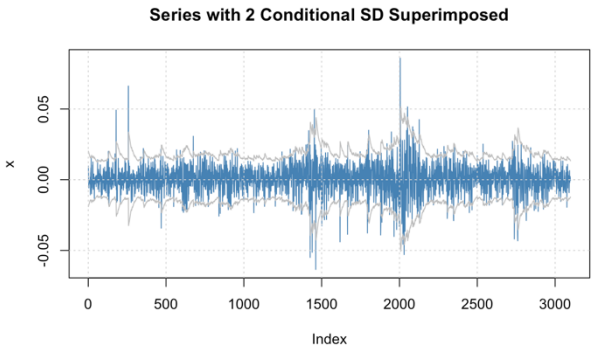
These residual diagnostics plots look decent but hint that an ARIMA model is not the best fit. In the graph of standardized residuals, we can see them fairly scattered around zero but with a changing spread at times, possibly indicating heteroscedasticity (non-constant variance). The ACF of the residuals plot also looks alright for the most part but has a few random significant spikes that could signal some autocorrelation among the residuals. The normal Q-Q plot of standard residuals is also fairly on the straight blue line but has some outliers towards the ends that show deviations from normality. Lastly, the p values for the Ljung Box statistic appear above 0.05 for the first 10 lags, failing to reject the null hypothesis and indicating that the residuals are uncorrelated for these lags, but fall below or are around 0.05 for the rest of the lags, rejecting the null hypothesis and indicating that the residuals exhibit significant autocorrelation at those lags.

4.2 Results from GARCH(p, q) model

Following this analysis, I decided to transform the price series into a returns series, which I got by simply taking the log of the differences in prices, and assess whether a GARCH-type fit is appropriate as a refined model to the ARIMA one. I decided to take the log returns rather than use the optimal lambda from our Box-Cox transformation from earlier since that lambda was close to zero anyway, which would have indicated a log transformation. It is simpler to use the log function because that is what is generally used for financial returns data, and log returns are particularly useful due to their additive property and symmetry. To check for GARCH behavior, I looked at the ACF and PACF plots of the gold returns, examined the ACF and PACF plots of the squared residuals (which I have shown to the right), and conducted a McLeod-Li test, which detects significant autocorrelation in the squared residuals via a statistical test similar to the Ljung-Box test. All of these indicated significant autocorrelation among the squared residuals, suggesting that a model accounting for changing volatility like an ARCH or GARCH model may be more suitable than one that assumes that the residuals are homoscedastic (constant variance) like an ARIMA model and many other time series models.

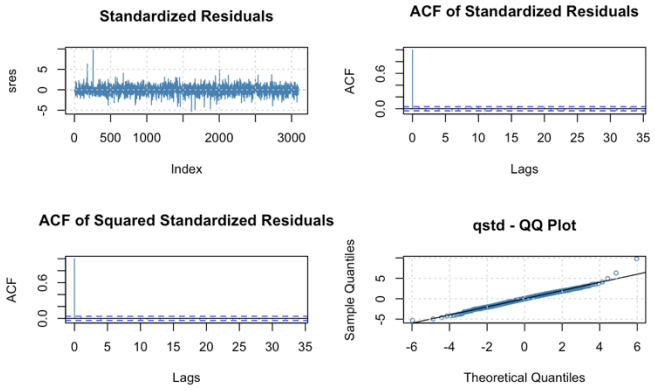


As can be seen, the autocorrelation structure in the squared residuals suggests GARCH behavior. Additionally, it suggests that a GARCH model would be a better fit than a simple ARCH model because, based on the ACF plot, for an ARCH(q) model, we would need a very large value of q for it to work well. Regardless, I tried fitting both a GARCH(1, 0) model (which would be an ARCH(1) model) and a GARCH(1, 1) model on top of our



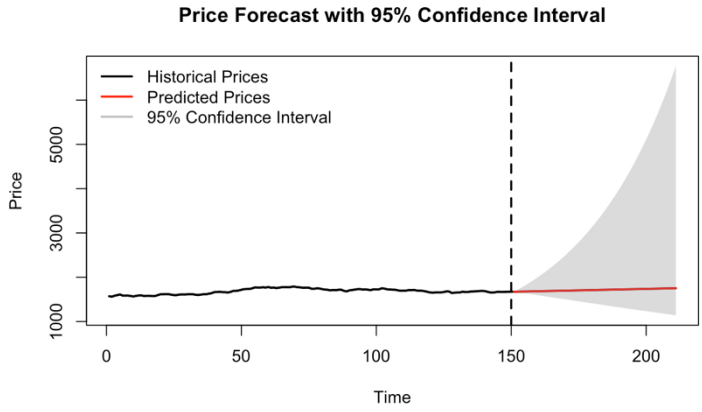
optimal AR(1) model we obtained from ARIMA modeling and found the GARCH (1, 1) model to be the best fit. On the top right is how the time series looked with 2 conditional standard deviations superimposed on it by the GARCH(1, 1) model. The conditional standard deviation bands are calculated by taking the square root of the conditional variance, where the conditional variance is estimated by the GARCH model. The bands are meant to widen during high-volatility periods and narrow during low-volatility periods, and they provide a visual representation of the time-varying volatility estimated by the GARCH model.

From the plot above to the right, the GARCH(1, 1) model seems effective at modeling the time-varying volatility in the gold price data. This plot looked much better than the one for the ARCH(1) model. Now all that was left was to check the lower information criteria (e.g., AIC, BIC) and analyze the standardized residuals to evaluate the model's performance. After calculating the AIC and BIC for the ARCH and GARCH models, I actually found that the ARCH model had slightly lower AIC and BIC values at -6.582282 and -6.572539 respectively compared to the GARCH model at -6.642045 and -6.630353 respectively. However, the residual plots for the GARCH model also looked much better, which further confirms the superiority of the GARCH model over the ARCH in this context. Below are plots of the standardized residuals, ACF of standardized residuals, ACF of squared standardized residuals, and QQ-Plot of standardized residuals for the GARCH fit.



These residual plots for the GARCH(1, 1) model indicate a solid fit overall for our gold data! Lastly, I used the McLeod-Li test to check for remaining ARCH effects in the standardized residuals. A p-value greater than 0.05 would be ideal because it indicates no significant remaining ARCH effects, meaning the GARCH model has adequately captured the volatility clustering. For my ARCH model, I got a p-value of 2.331e-15 while for my GARCH model, I got a p-value of 0.9889, also showing that the GARCH model was adequate!

With these optimal model fits, I also tried out some forecasting, trying to predict 60 days ahead with both my ARIMA and GARCH models. All of the models seemed to yield similar forecasts (excluding confidence intervals because those differed quite a bit), but it should be noted that I only forecasted the next two months of gold prices for each model. Nonetheless, even with only two months of forecasting, it seems like the GARCH model did slightly better than the ARIMA model when comparing the forecasts to the actual data. To the right is how my forecast looked with the GARCH(1, 1) model.



5. Conclusion and Future Study

To summarize, I ended up fitting both an ARIMA model and a GARCH model to the gold price data and found that the ARIMA model did okay but exhibited some clustered volatility and autocorrelation in its squared residuals, while the GARCH model did reasonably better. Ultimately, I would say that the optimal model I obtained out of all the models I fitted to the data was a GARCH(1, 1) model. In terms of forecasting, I tried to predict the prices of gold for the next two months with my data and the models I had and found that the GARCH model also did slightly better than the ARIMA model when comparing their forecasts to the actual data.

For future study, I would like to delve more into this forecasting aspect and see how the forecasts change for periods into the more distant future, for both the ARIMA and the GARCH models I fitted. I would also like to explore other models and see how they compare, such as state space or machine learning models.

Another interesting analysis would be fitting these models with all of the original data in the dataset rather than on a truncation of the data and seeing how they compare. My assumption is that all the models would perform worse because of the large fluctuations in the price in some of the areas that I cut out.

6. References and Appendix

6.1 References

Ash, A. (2023). *Gold's Big Price Crash, 10 Year On*. https://www.bullionvault.com/gold-news/opinion-analysis/golds-big-price-crash-10-year-07072023

Collinson, P. (2013). *Gold Price Slumps as Traders Face Global Metals Market Freefall.* https://www.theguardian.com/money/2013/apr/20/gold-price-slumps

Khetlani, K. (2021). *Gold Prices Historical Data.* https://www.kaggle.com/datasets/komalkhetlani/gold-prices-historical-data

Shumway, R. H., & Stoffer, D. S. (2025). *Time Series Analysis and its Applications: With R Examples* (Fifth). Springer Nature Switzerland Springer.

6.2 Appendix

Here are some useful formulas for some of the terms mentioned above:

Box-Cox transformation formula:

$$y_i^{(\lambda)} = \begin{cases} \dfrac{y_i^{\lambda} - 1}{\lambda} & \text{if } \lambda \neq 0, \\ \ln(y_i) & \text{if } \lambda = 0, \end{cases}$$

Ljung-Box test statistic formula:

$$Q = n(n+2) * \sum_{i=1}^{k} \frac{r_i^2}{n-k}$$

General SARIMA (p, d, q) x (P, D, Q) model formula:

$$\phi_p(B)\,\Phi_P(B^s)\,(1-B)^d(1-B^s)^D x_t = \theta_q(B)\,\Theta_Q(B^s)\,z_t$$

Here are the graphs we already displayed along with some additional graphs that provide more context to our analysis:

Cleaned Data:

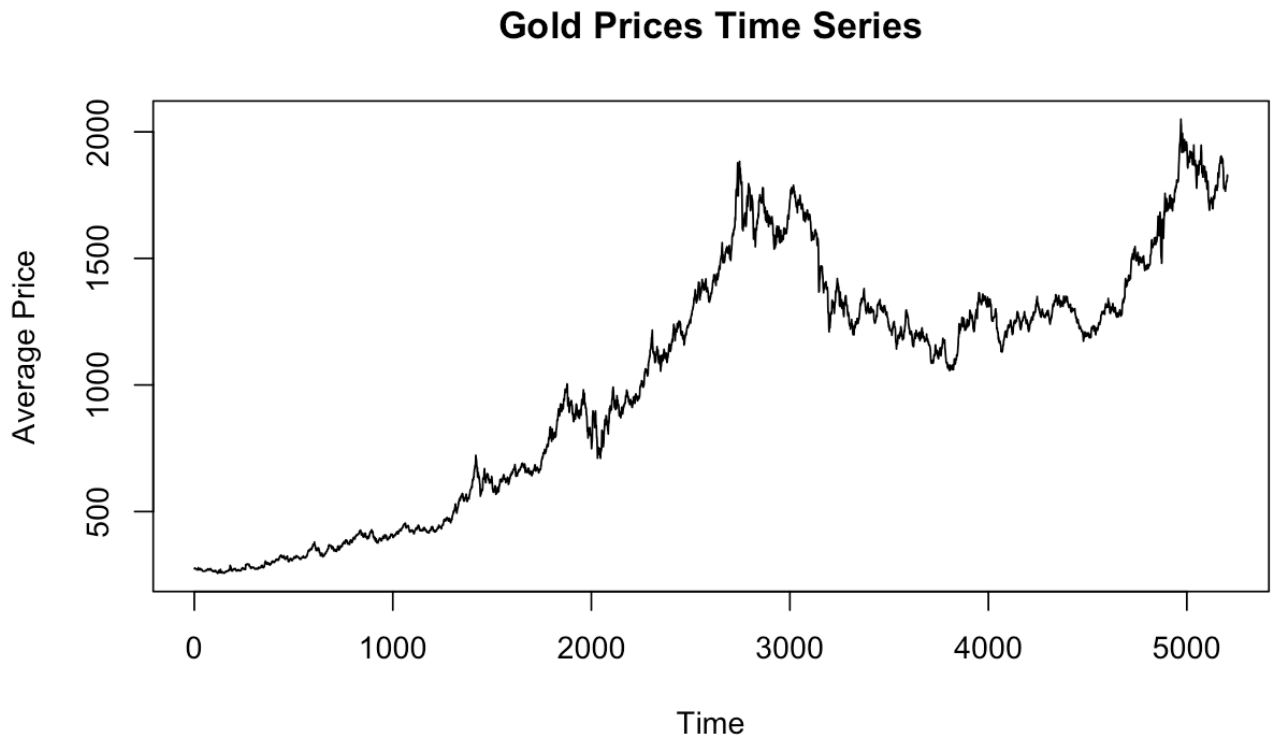
A tibble: 10 × 8

date	open	high	low	close	adj_close	volume	avg_price
<date>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
2000-08-30	273.9	273.9	273.9	273.9	273.9	0	273.90
2000-08-31	274.8	278.3	274.8	278.3	278.3	0	276.55
2000-09-01	277.0	277.0	277.0	277.0	277.0	0	277.00
2000-09-05	275.8	275.8	275.8	275.8	275.8	2	275.80
2000-09-06	274.2	274.2	274.2	274.2	274.2	0	274.20
2000-09-07	274.0	274.0	274.0	274.0	274.0	125	274.00
2000-09-08	273.3	273.3	273.3	273.3	273.3	0	273.30
2000-09-11	273.1	273.1	273.1	273.1	273.1	0	273.10
2000-09-12	272.9	272.9	272.9	272.9	272.9	0	272.90
2000-09-13	272.8	272.8	272.8	272.8	272.8	0	272.80

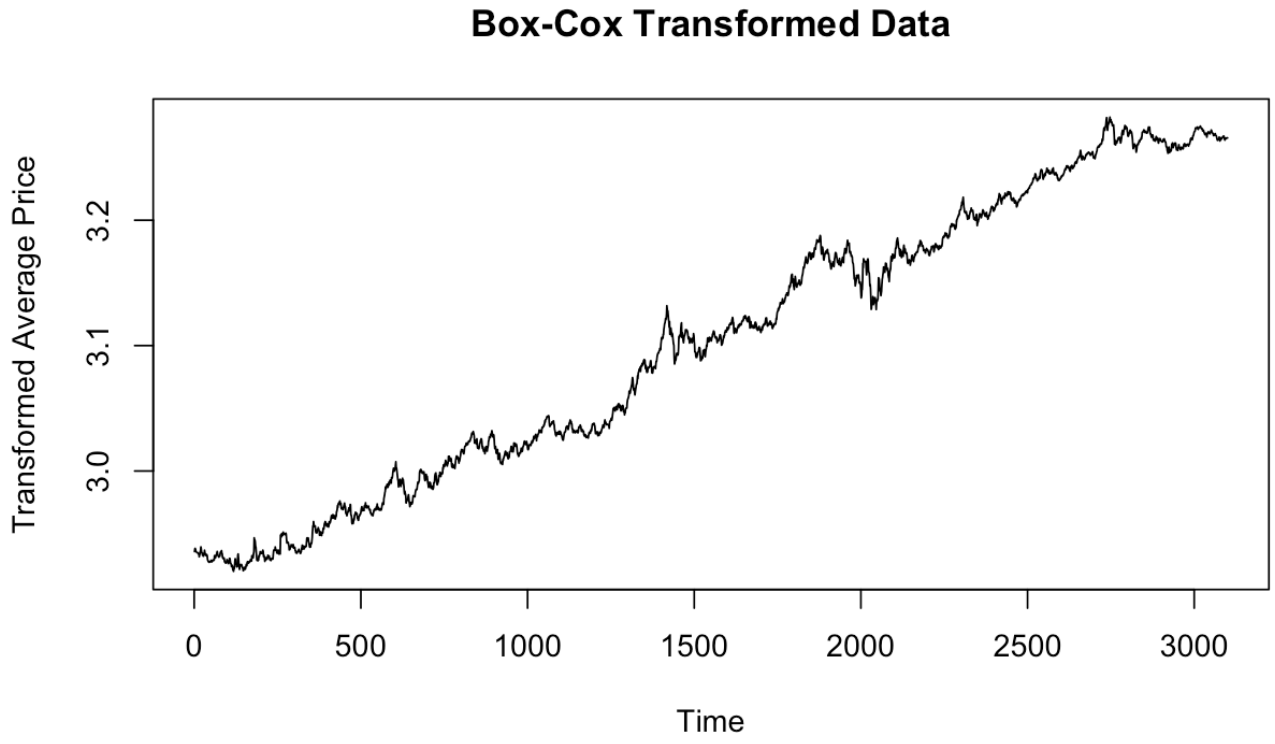
10 rows

Gold Prices Extensible Time Series:

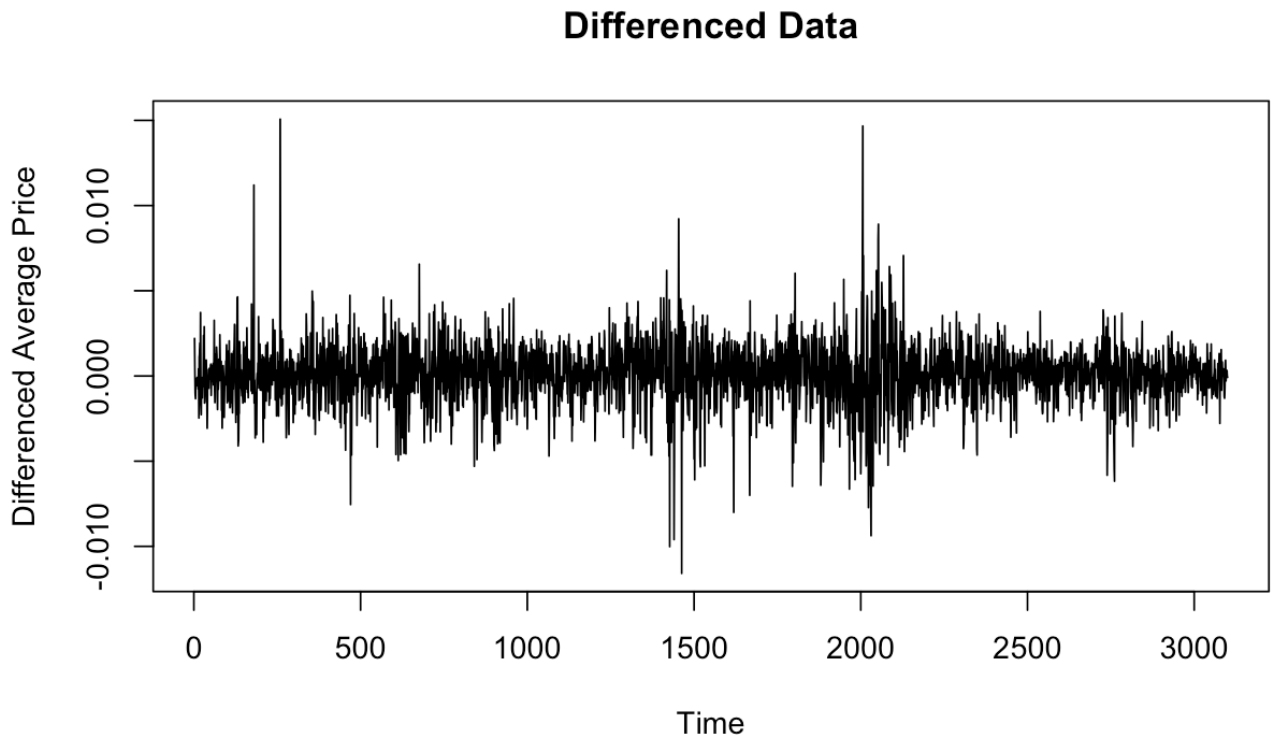
Gold Prices Time Series:



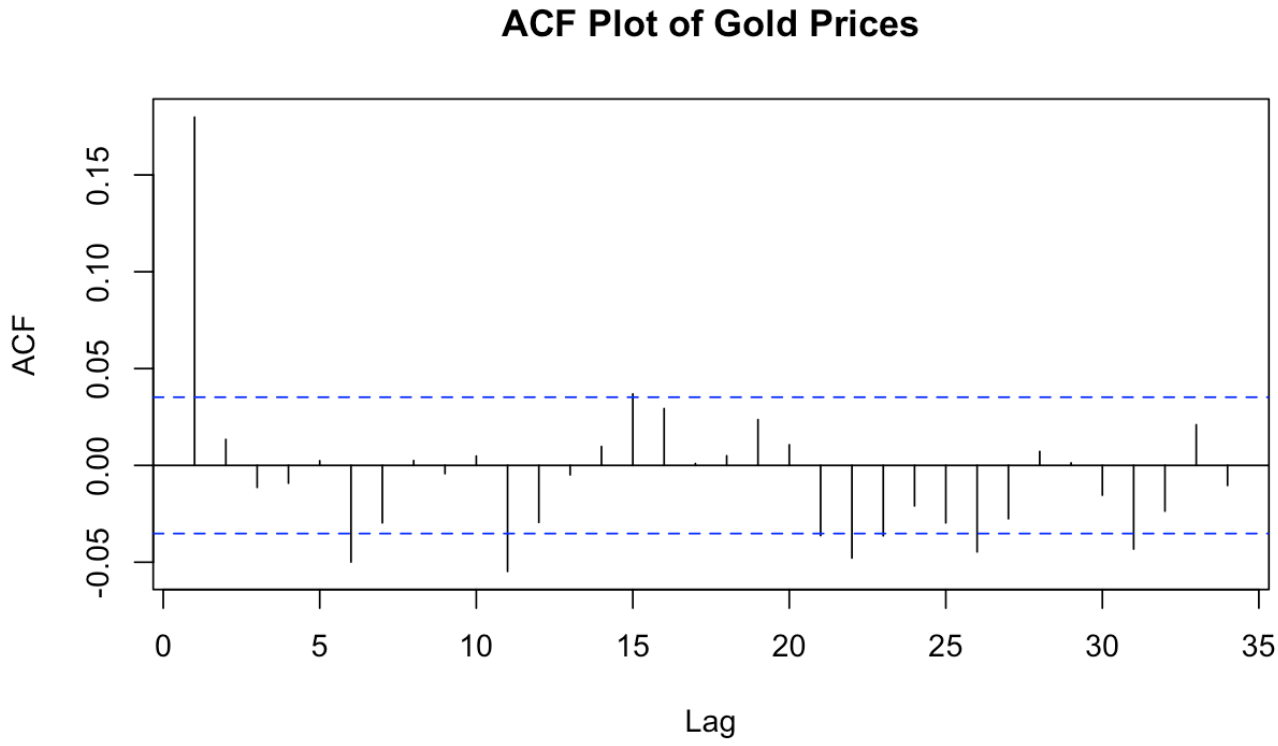
Box-Cox Transformed Data:



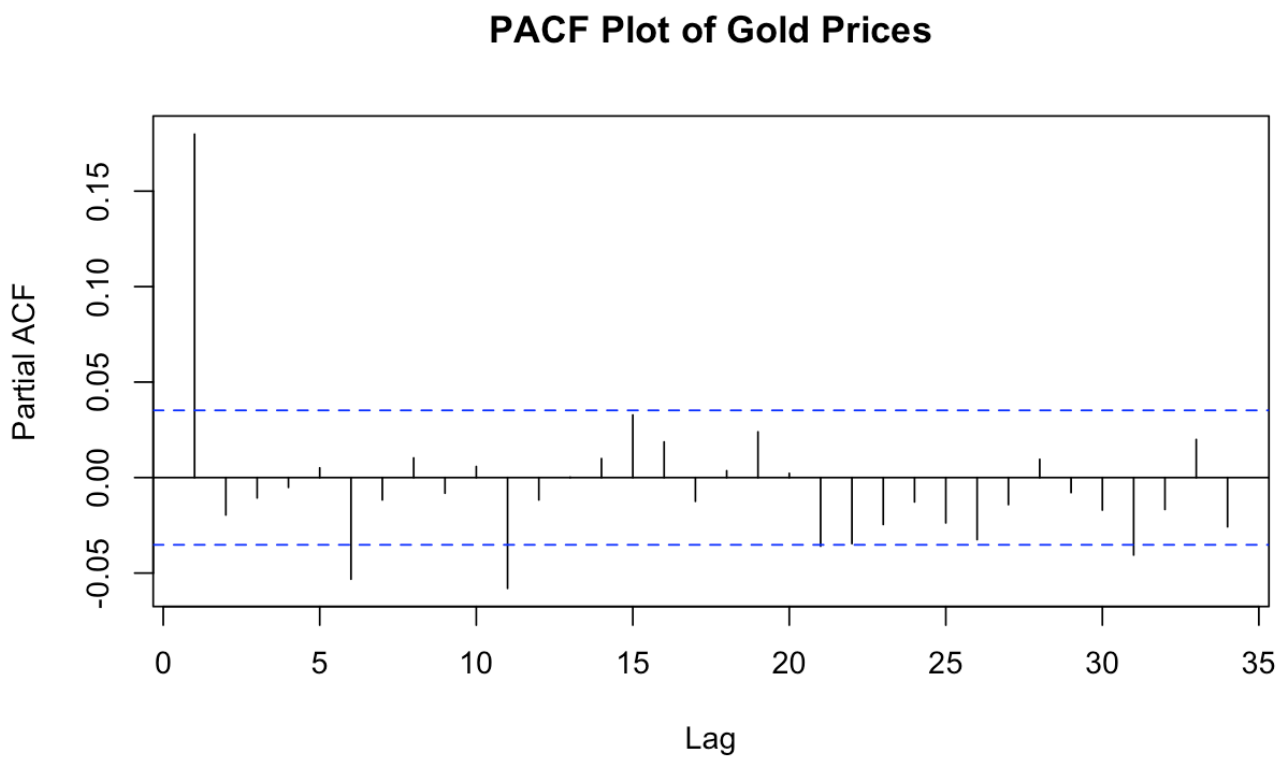
Differenced Data:



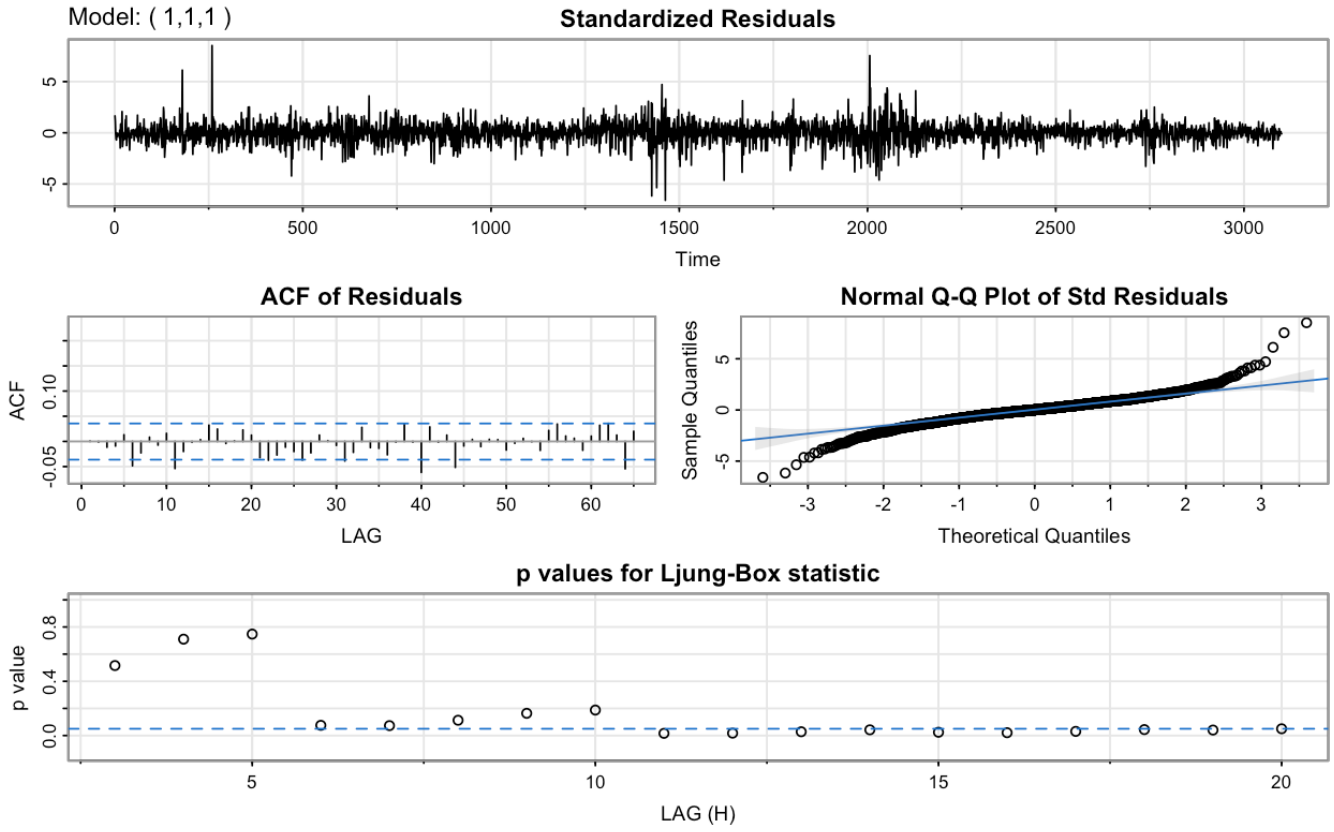
ACF Plot of Gold Prices:



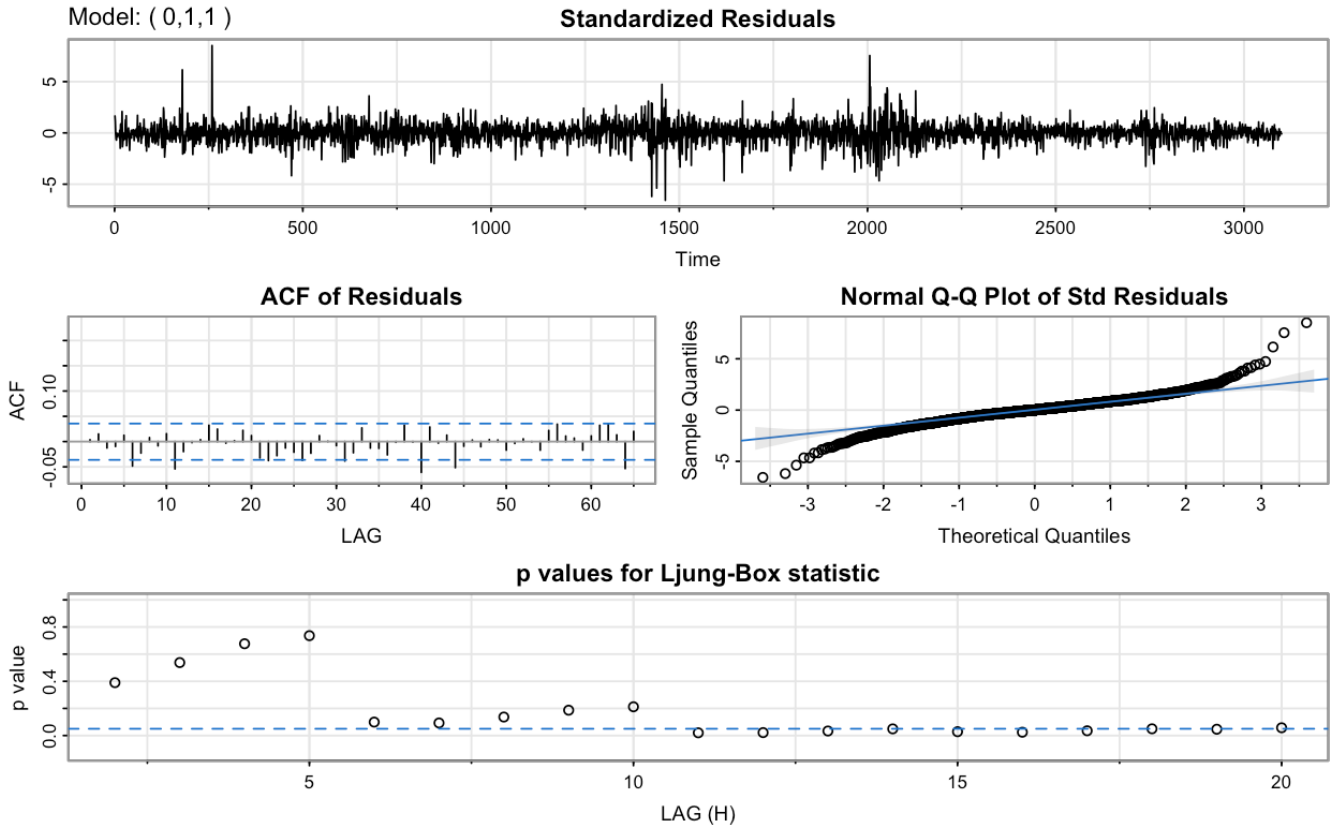
PACF Plot of Gold Prices:



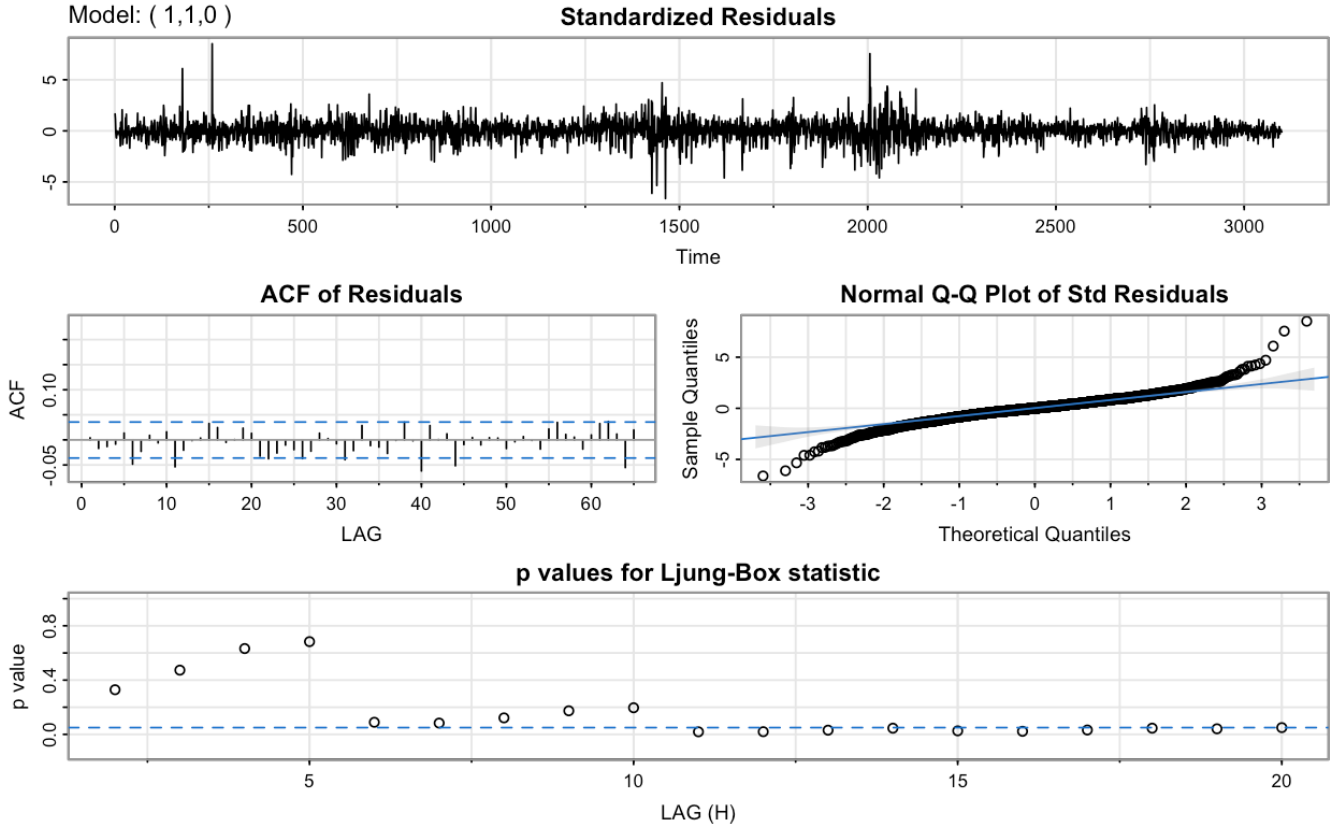
ARIMA(1, 1, 1) Residual Diagnostics Plots:



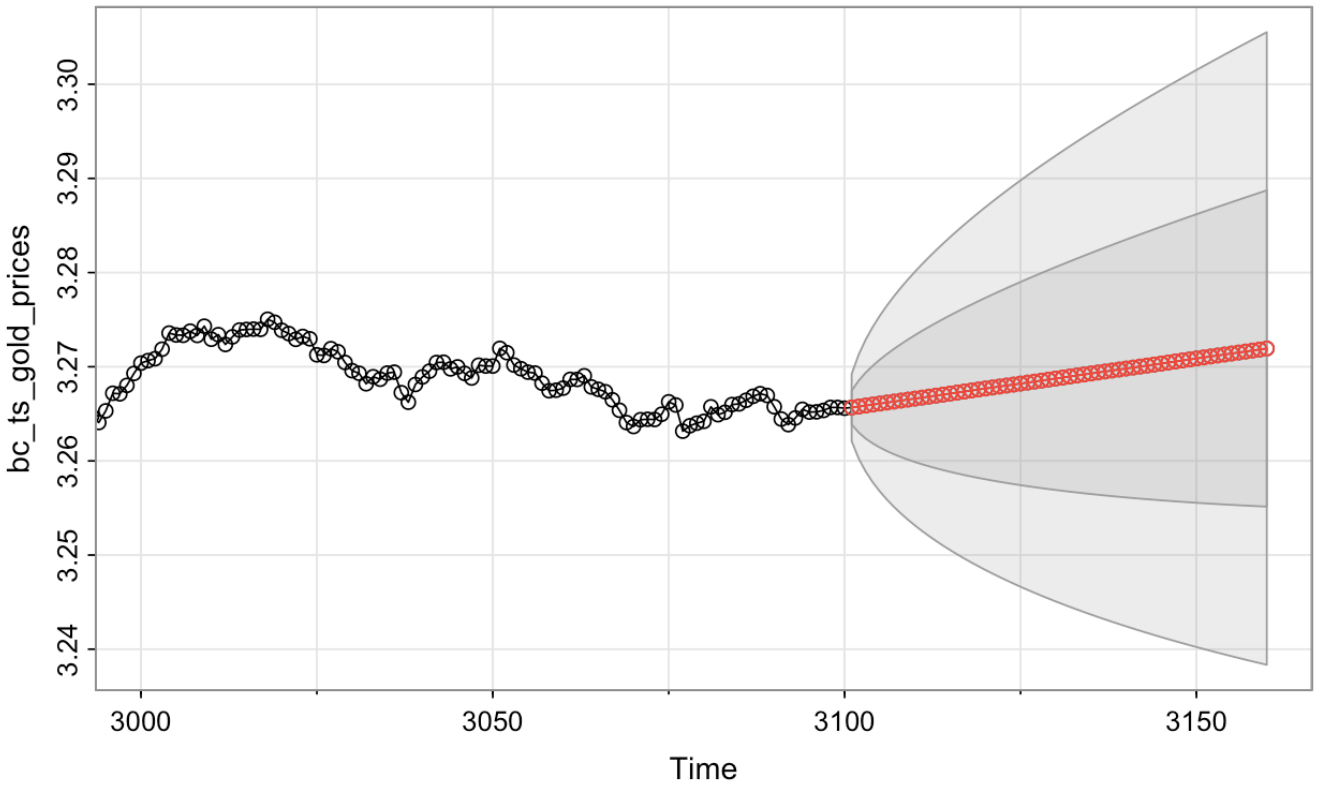
ARIMA(0, 1, 1) Residual Diagnostics Plots:



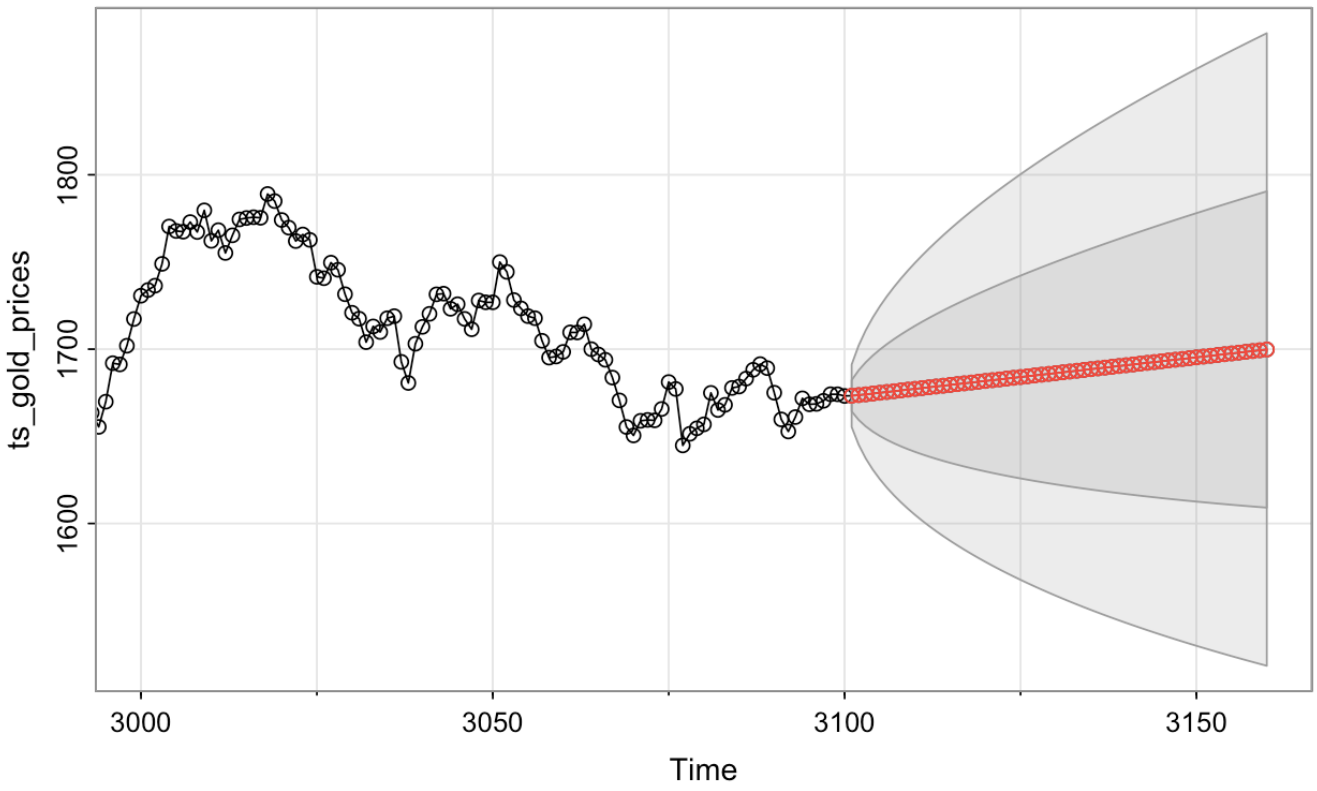
ARIMA(1, 1, 0) Residual Diagnostics Plots:



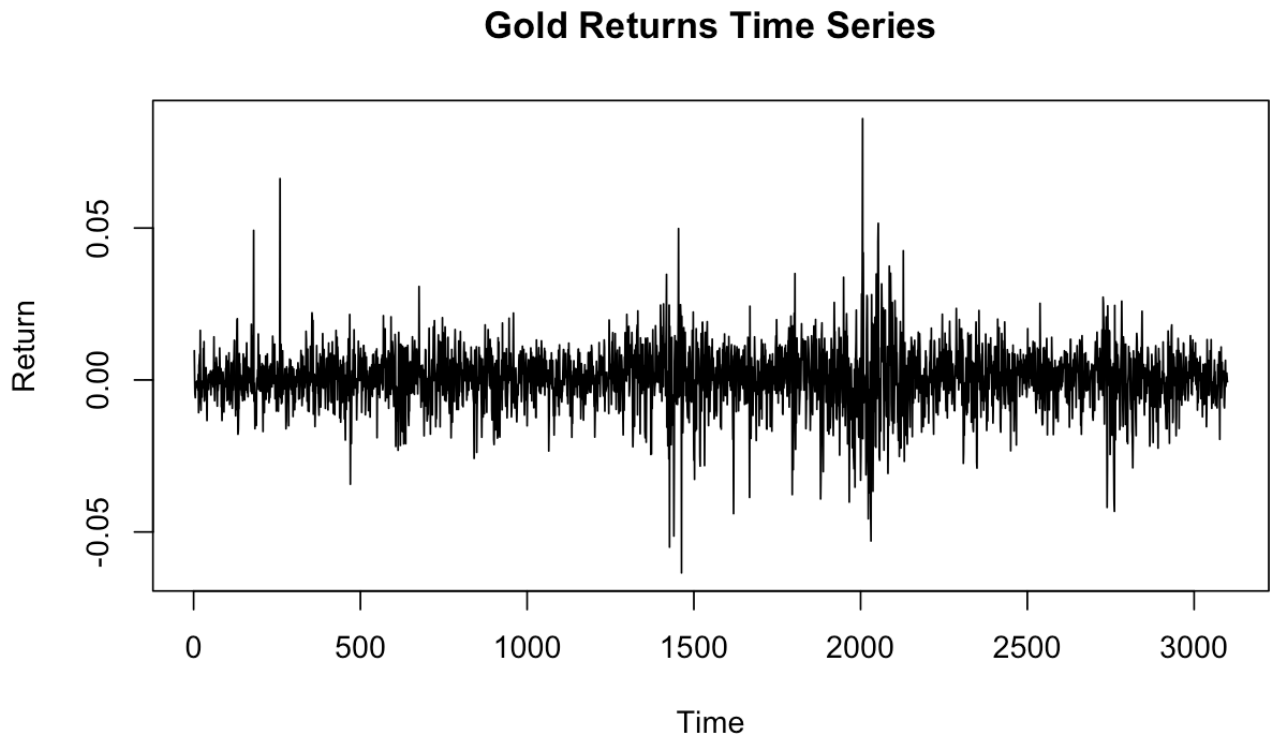
ARIMA(1, 1, 0) Return Forecast:



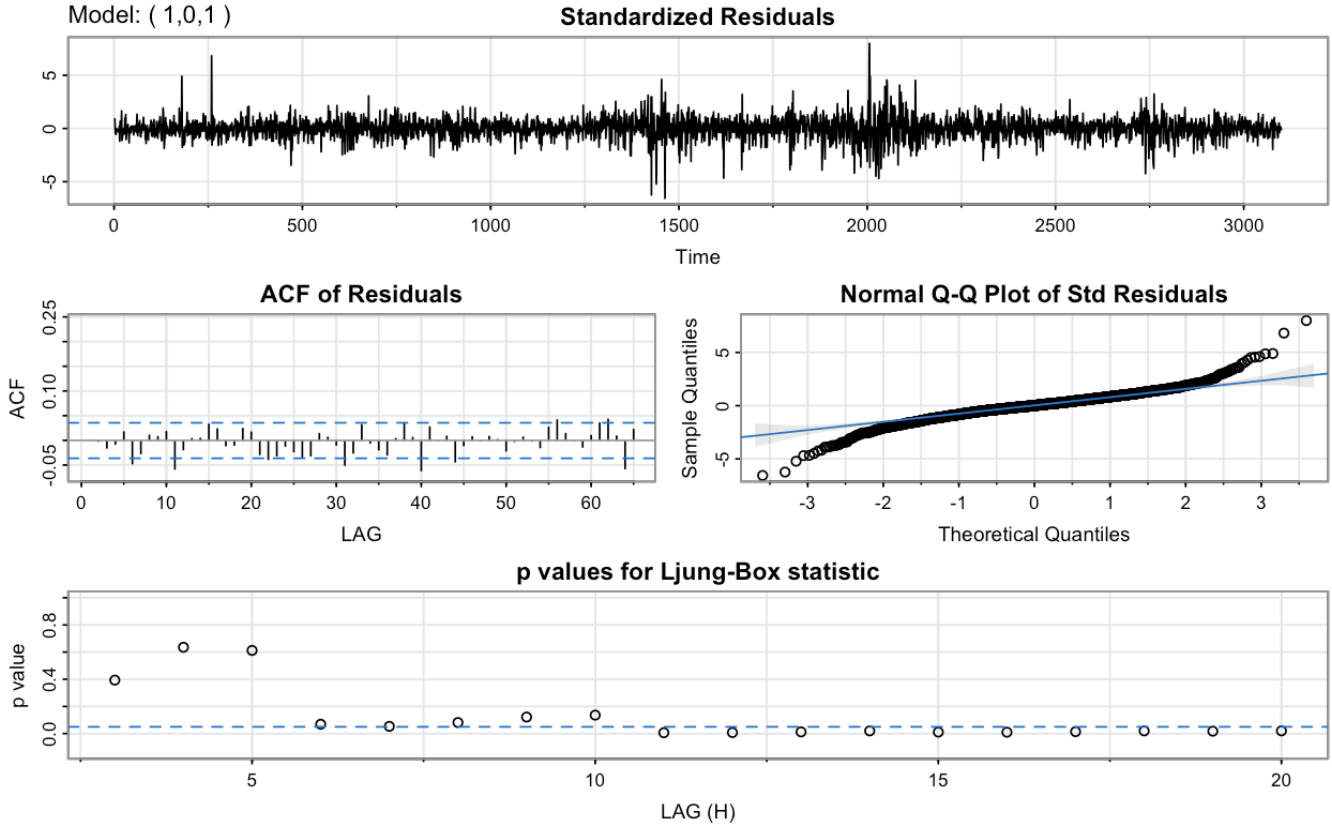
ARIMA(1, 1, 0) Price Forecast:



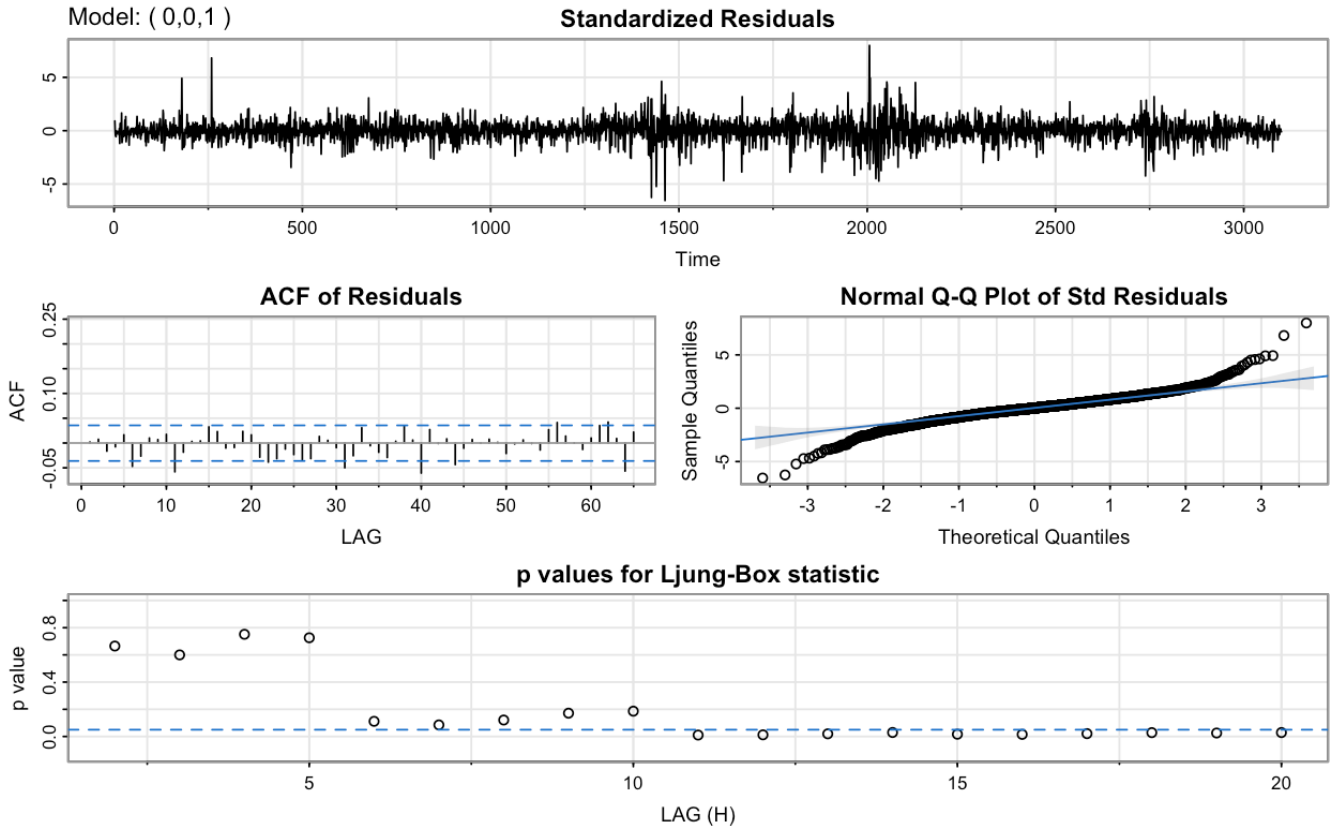
Gold Returns Time Series:



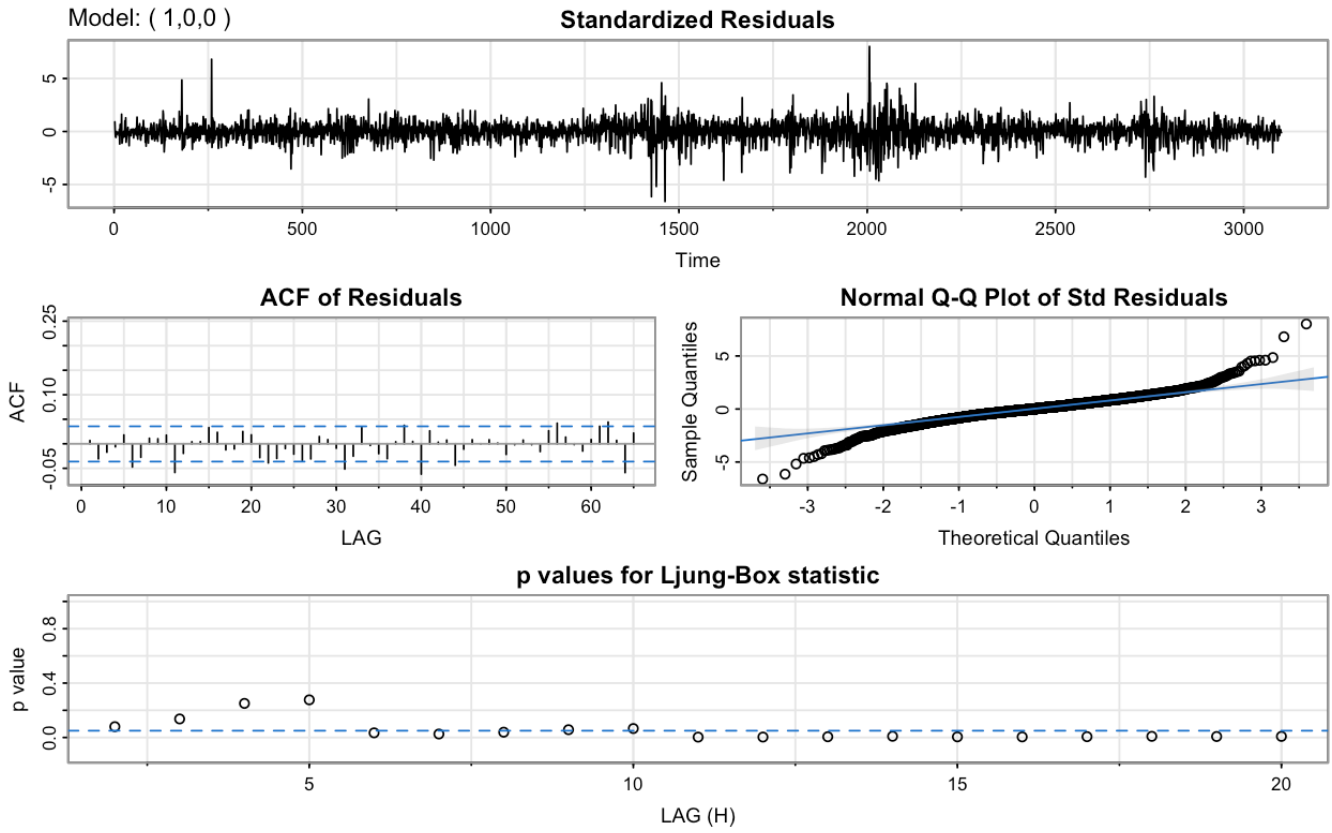
Gold Returns ARIMA(1, 1, 1) Residual Diagnostics Plots:



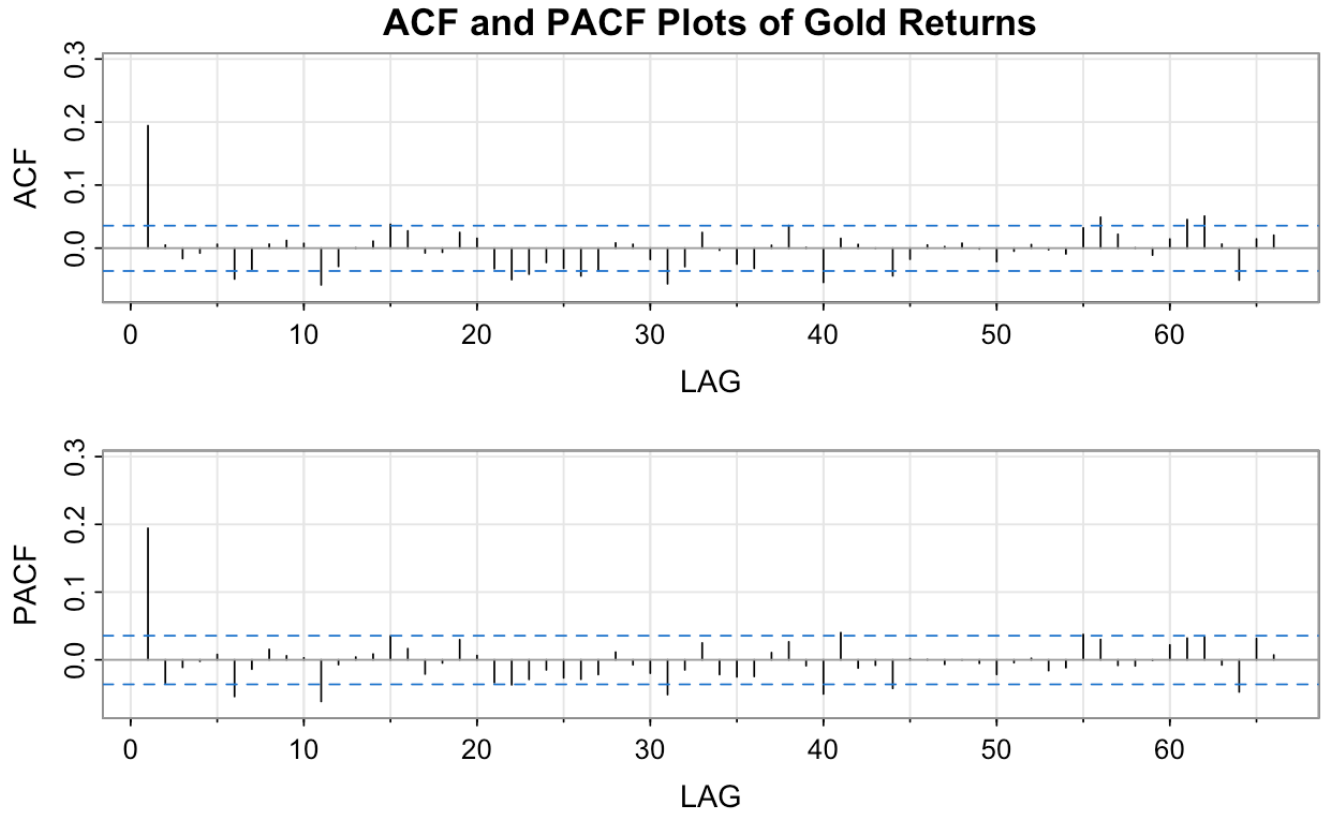
Gold Returns ARIMA(0, 1, 1) Residual Diagnostics Plots:



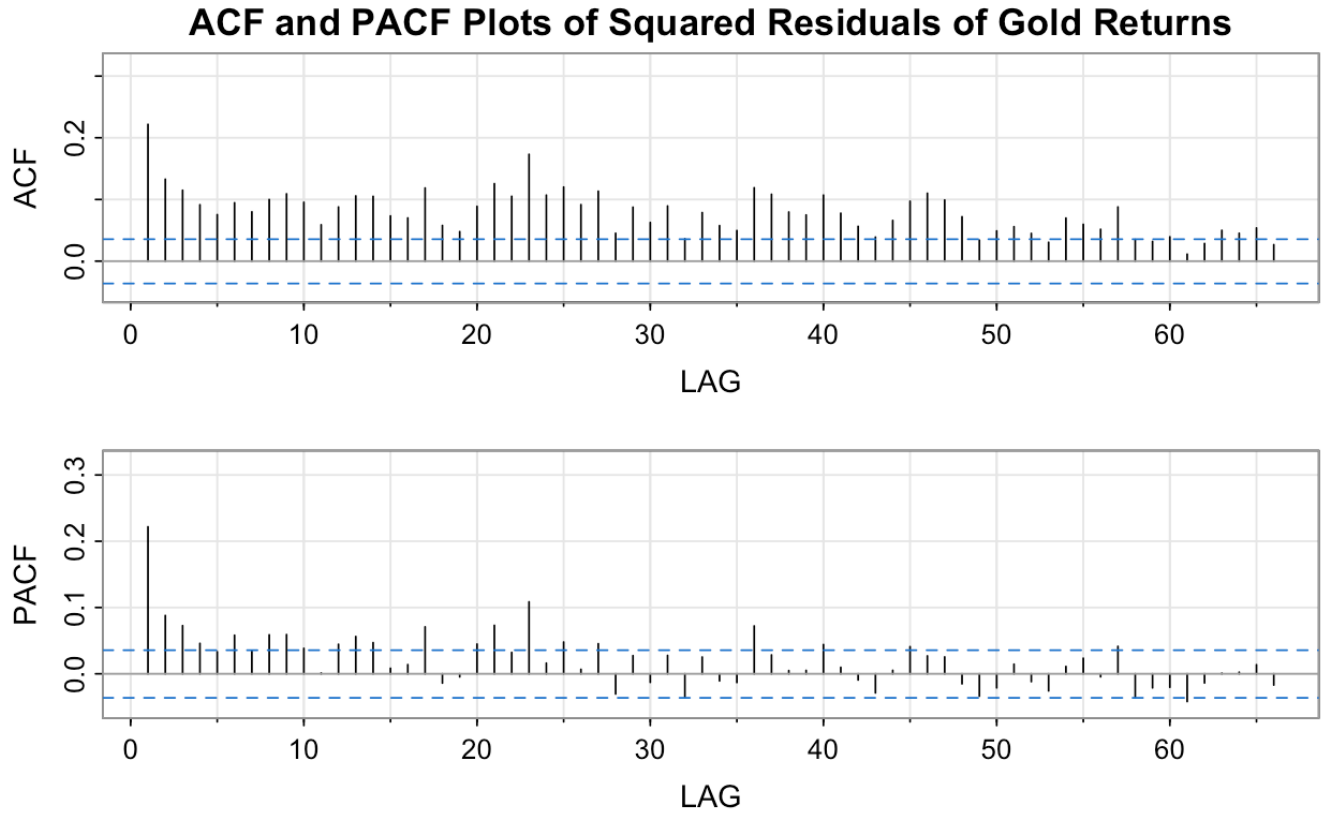
Gold Returns ARIMA(1, 1, 0) Residual Diagnostics Plots:



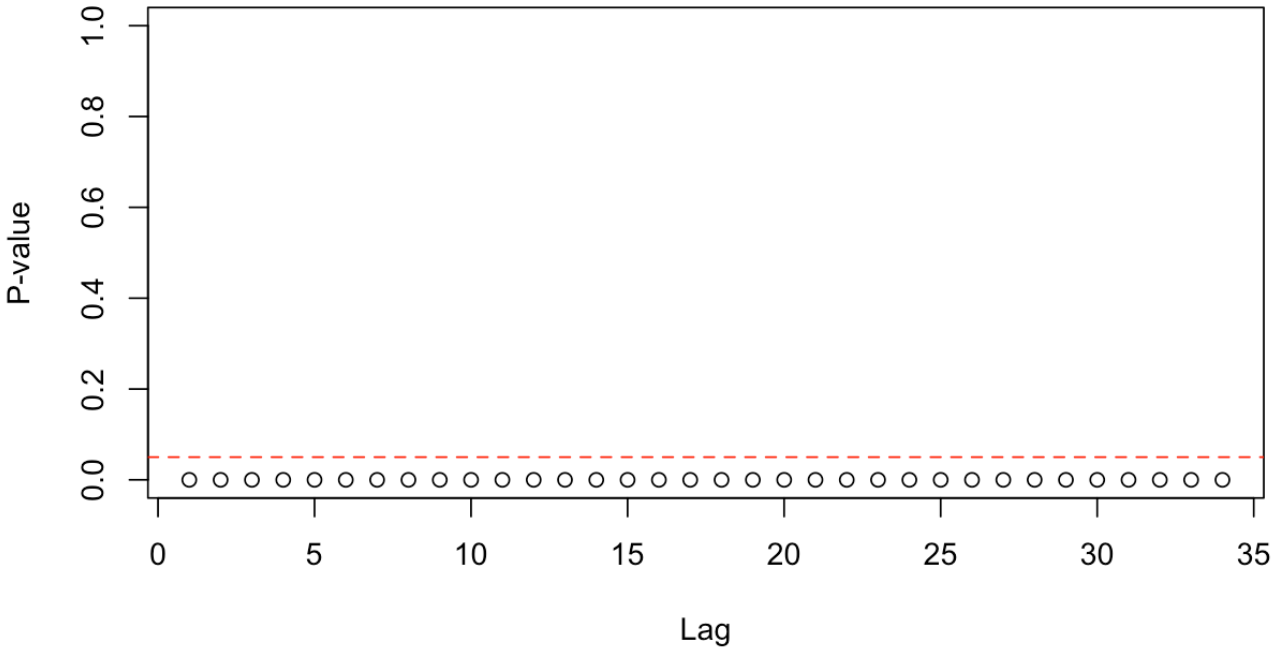
ACF and PACF Plots of Gold Returns:



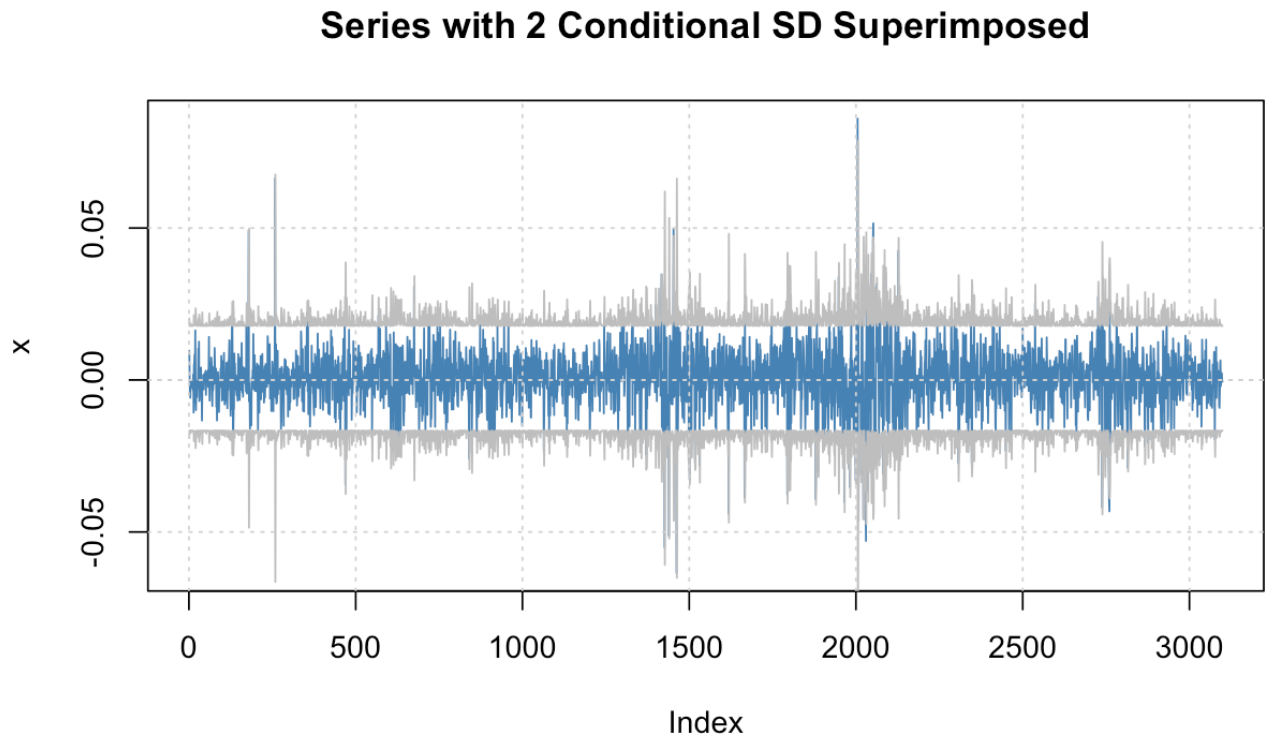
ACF and PACF Plots of Squared Residuals of Gold Returns:



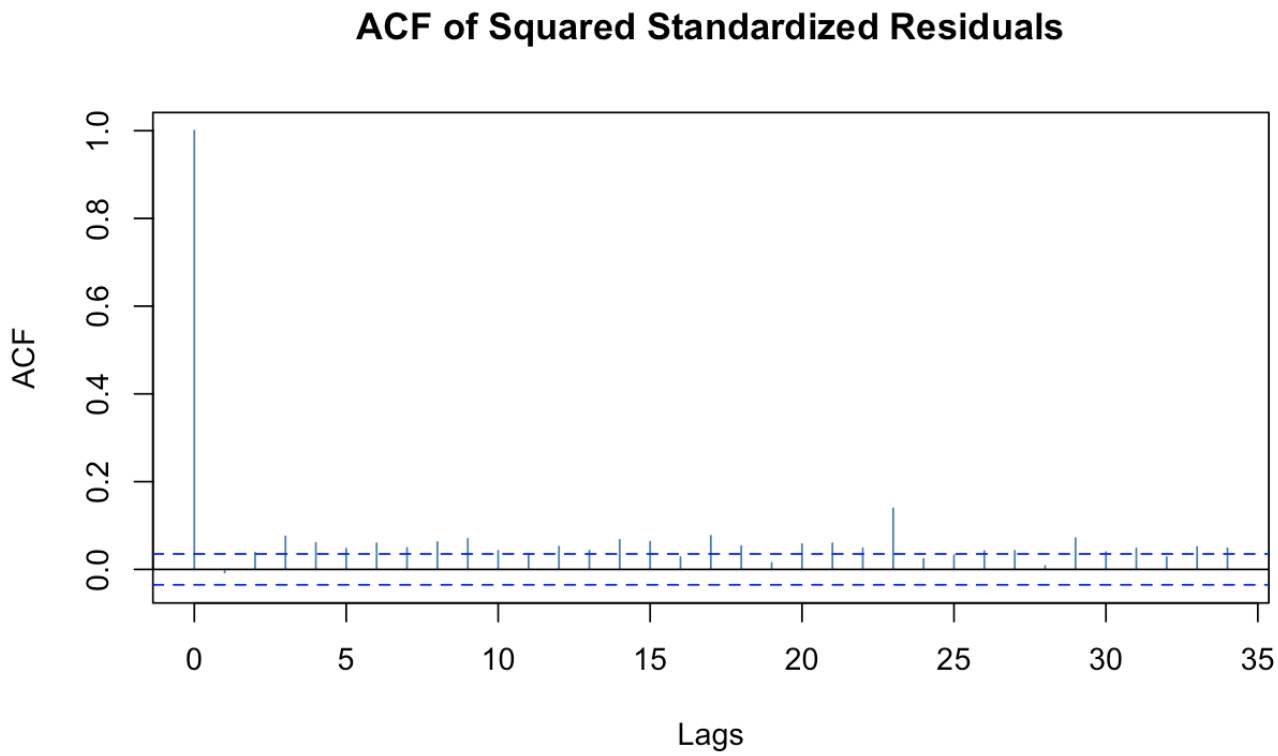
Plot of p-values for Mc-Leod-Li Test Statistic of Squared Residuals of Gold Returns:



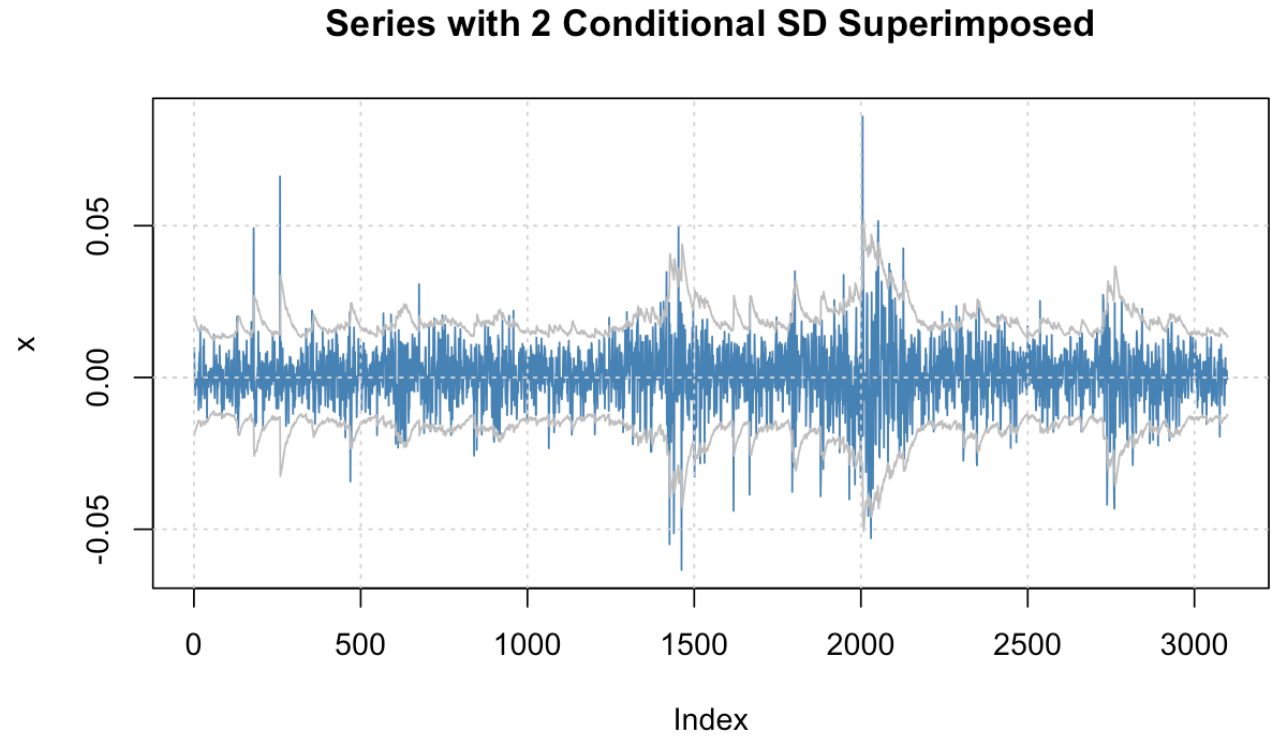
Series with 2 Conditional SD Superimposed for GARCH(1, 0):



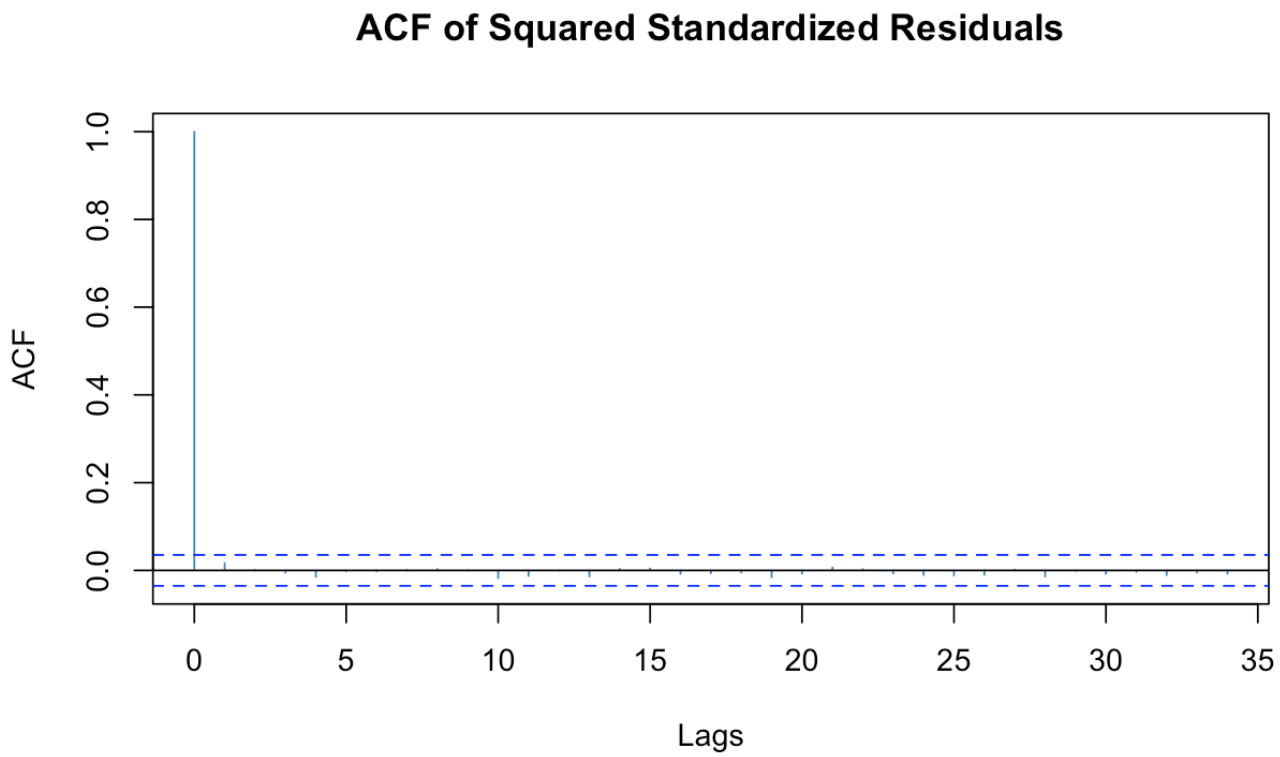
ACF of Squared Standardized Residuals for GARCH(1,0):



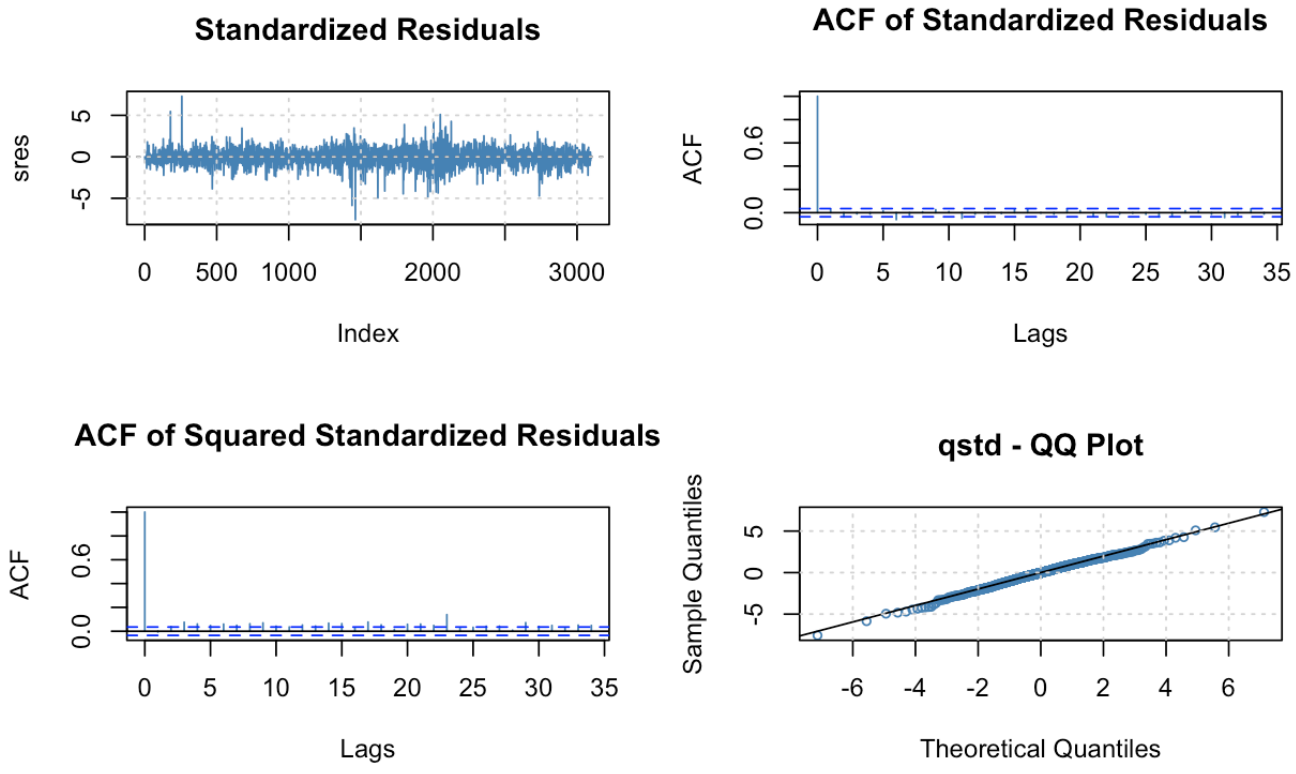
Series with 2 Conditional SD Superimposed for GARCH(1, 1):



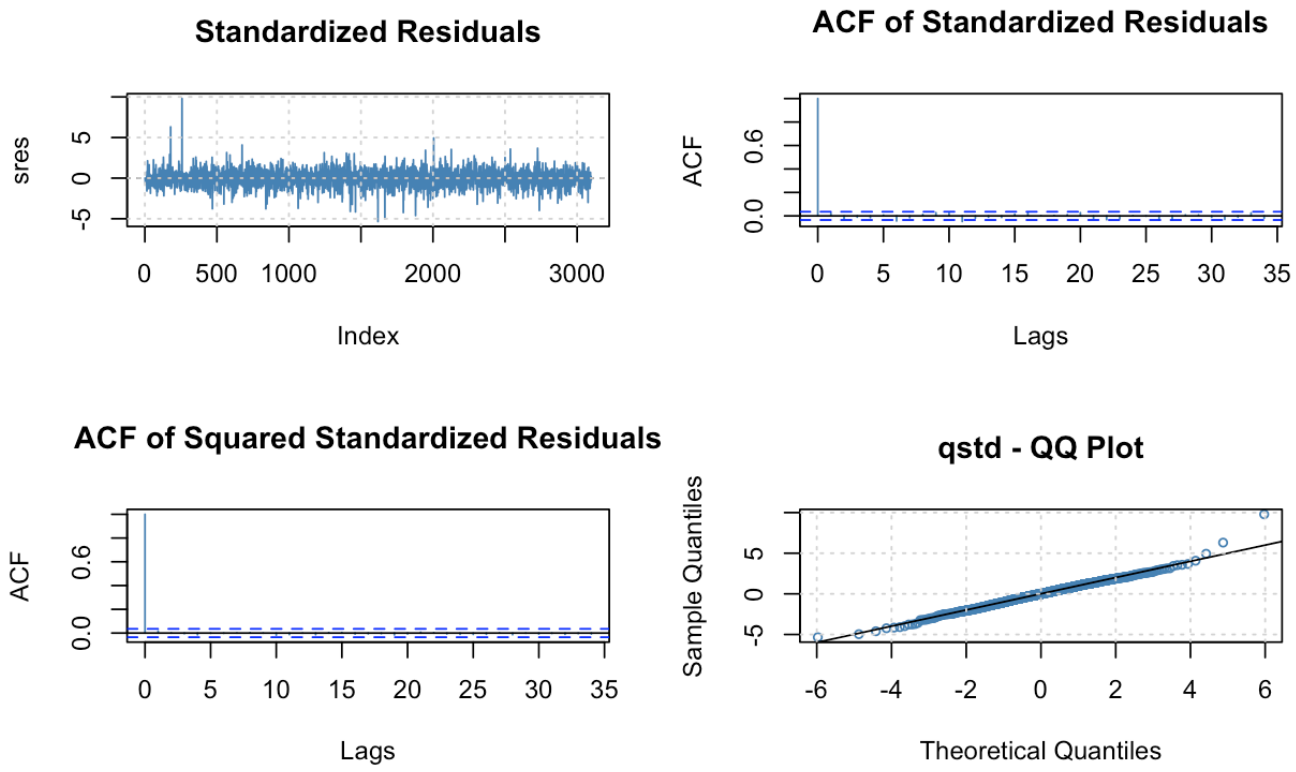
ACF of Squared Standardized Residuals for GARCH(1,1):



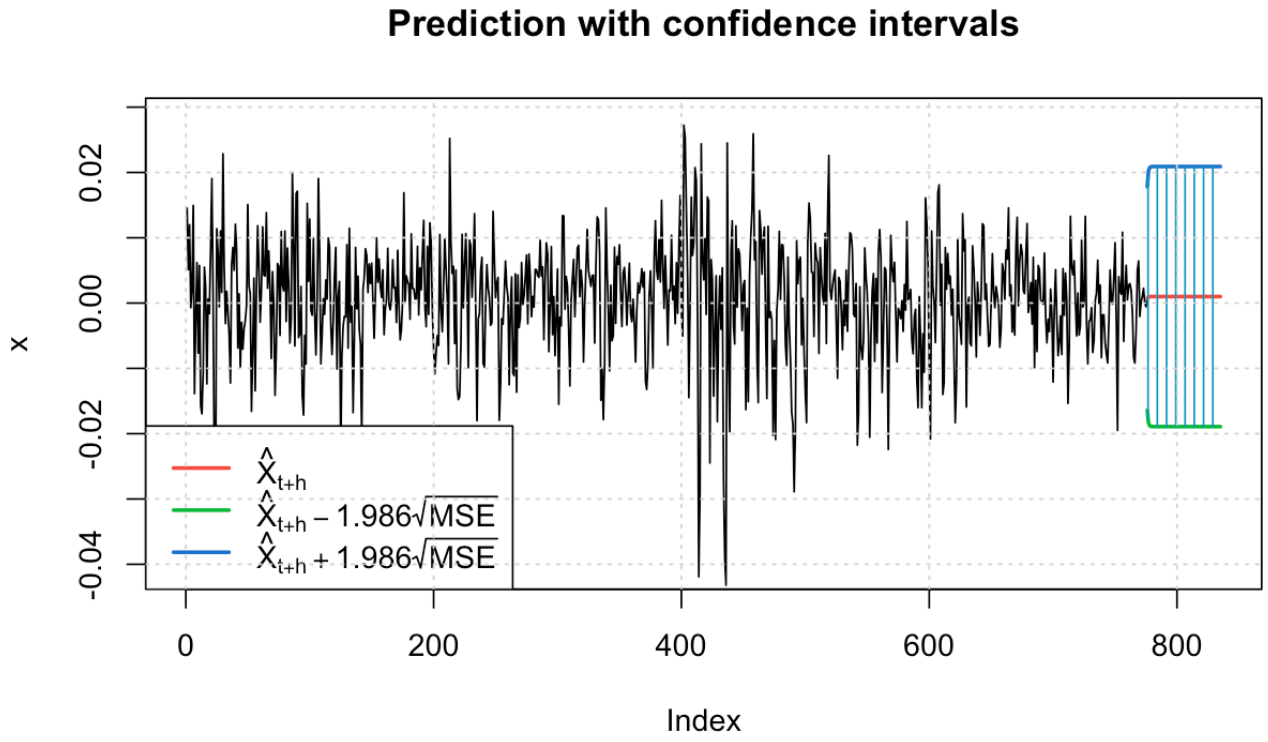
GARCH(1, 0) Residual Diagnostics Plots:



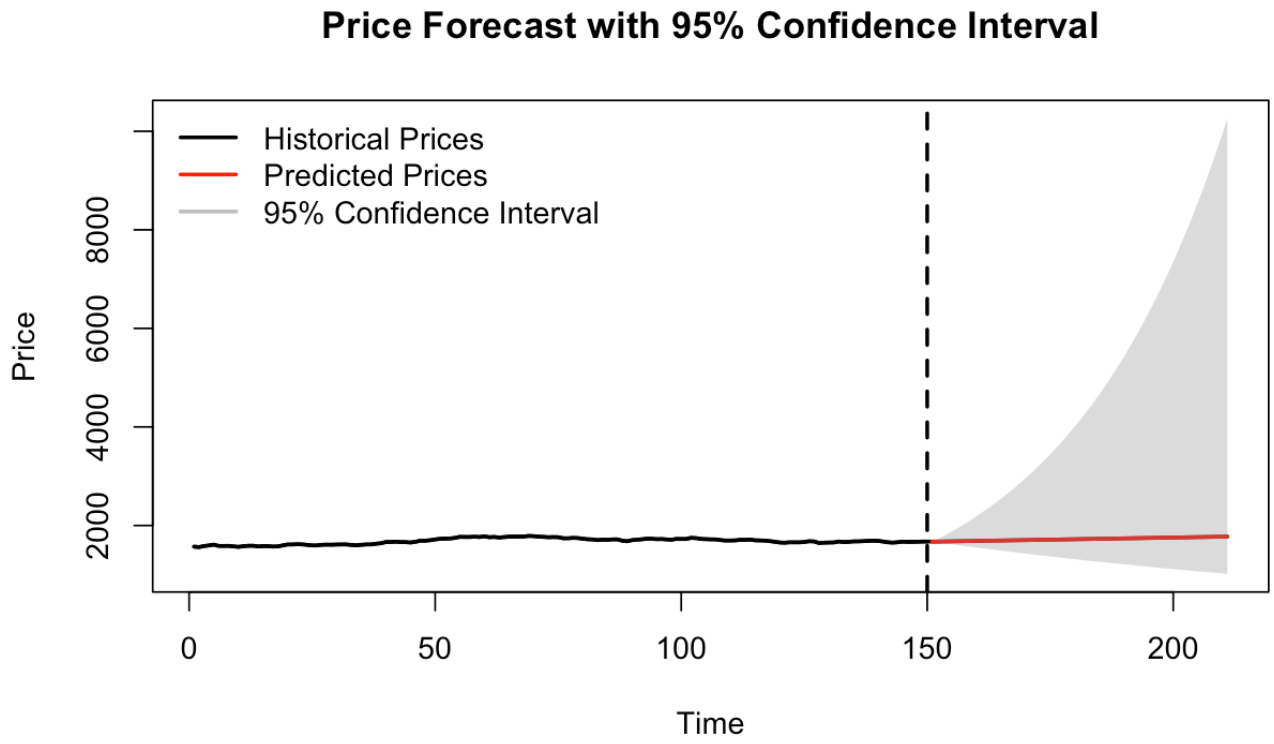
GARCH(1,1) Residual Diagnostics Plots:



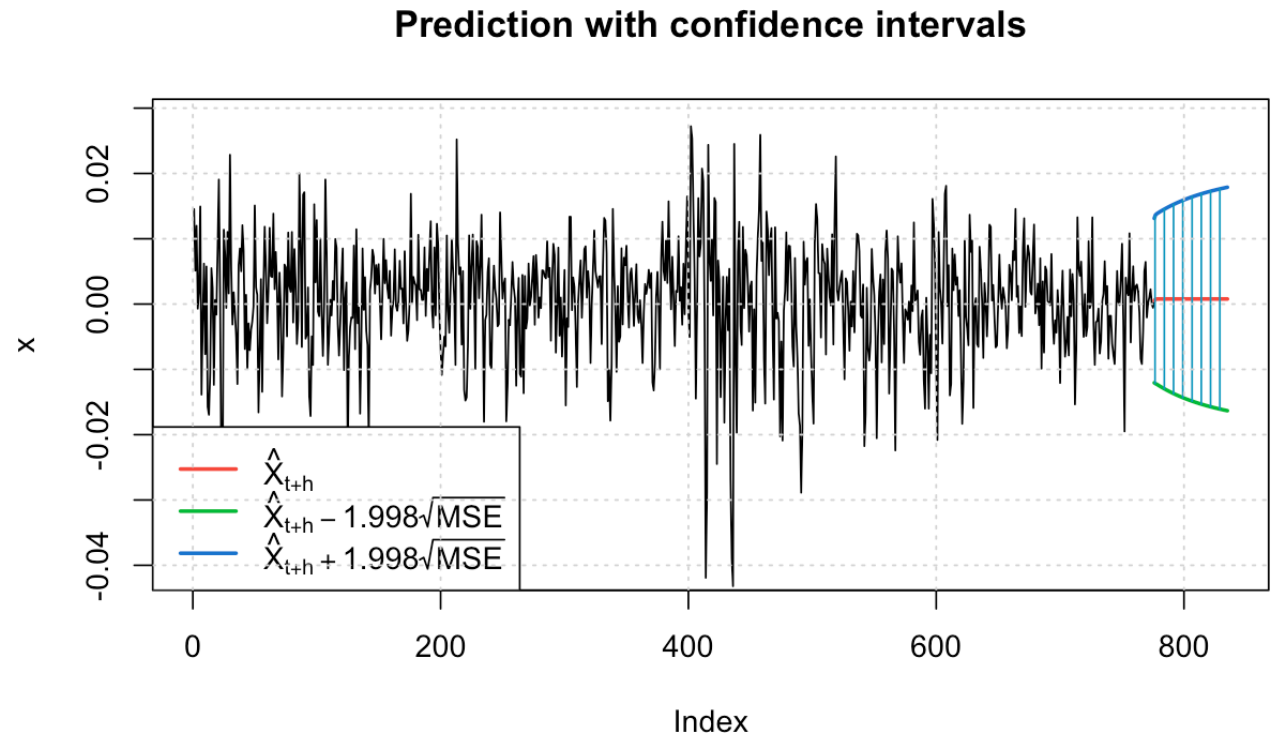
GARCH(1, 0) Return Forecast:



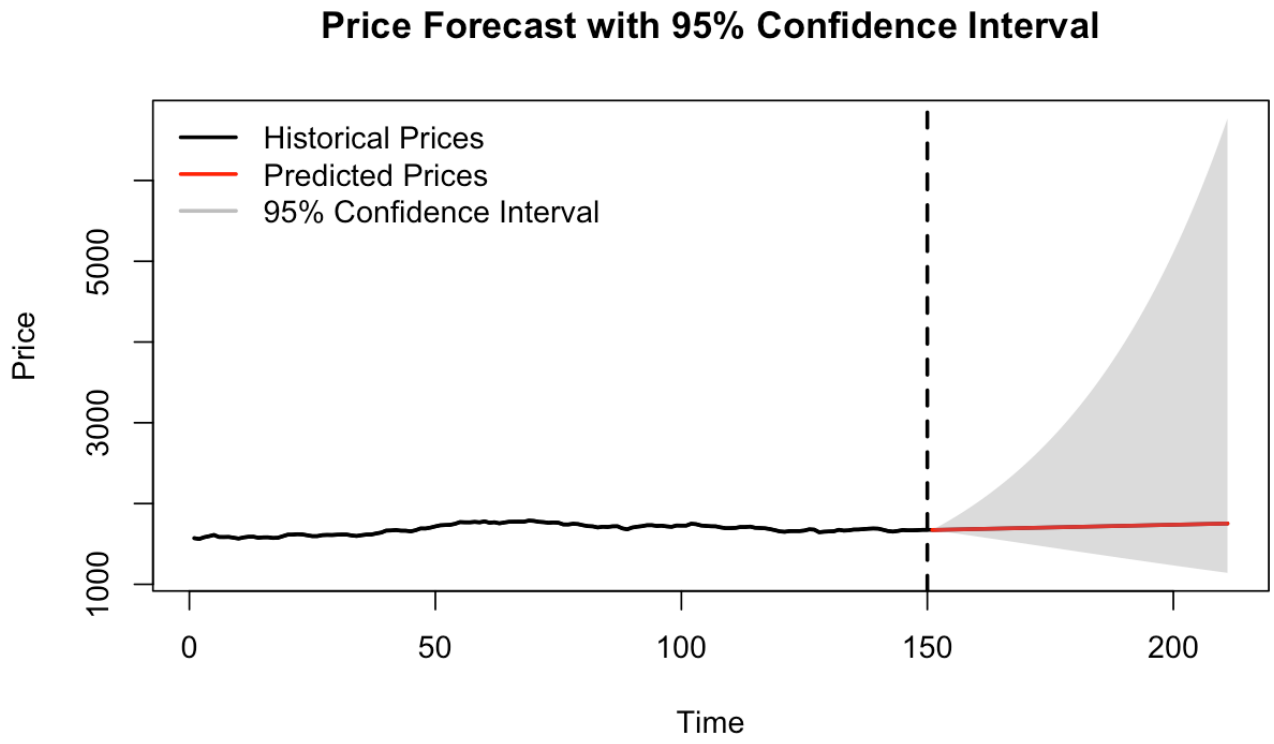
GARCH(1, 0) Price Forecast:



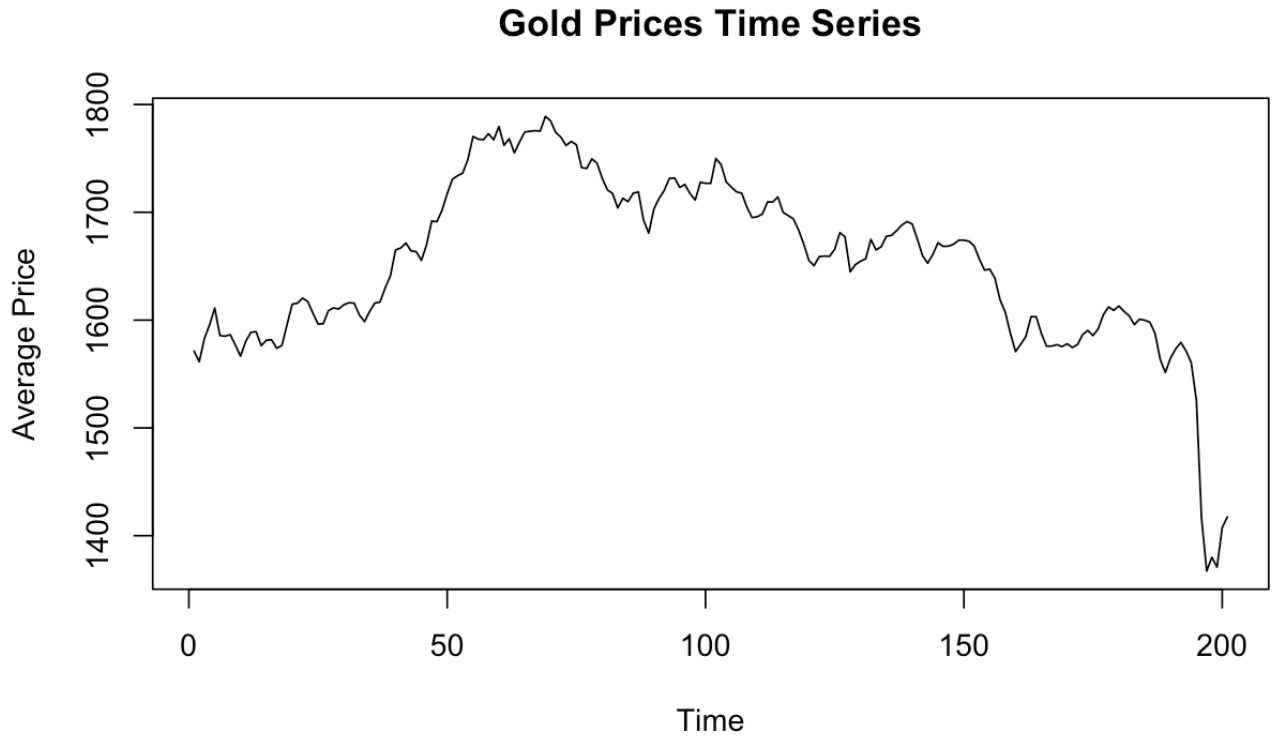
GARCH(1, 1) Return Forecast:



GARCH(1, 1) Price Forecast:



Actual Prices That Were Forecasted:



Here is the R code for our analysis:

---

title: "Final Project"

author: "Nabeel Vakil"

date: "2025-03-21"

output: pdf_document

```{r setup, include=FALSE}

knitr::opts_chunk$set(echo = TRUE)

```

```{r}

# loading the necessary packages

library(tidyverse)

library(janitor)

library(astsa)

library(MASS)

library(xts)

library(forecast)

library(TSA)

library(fGarch)

```

```{r}

# loading in the data

gold_prices <- read_csv("data/goldprices.csv")
```

```
# tidying up the column names
gold_prices <- gold_prices %>%
  clean_names()

# seeing the first 10 rows of the dataset
gold_prices %>%
  head(10)

# seeing the last 10 rows of the dataset
gold_prices %>%
  tail(10)

# converting the values specified as "null" to `NA` values
gold_prices <- gold_prices %>%
  mutate(open = case_when(
    open == "null" ~ NA,
    .default = open
  ),
  high = case_when(
    high == "null" ~ NA,
    .default = high
  ),
  low = case_when(
    low == "null" ~ NA,
    .default = low
  ),
  close = case_when(
    close == "null" ~ NA,
```

```
    .default = close
  ),
  adj_close = case_when(
    adj_close == "null" ~ NA,
    .default = adj_close
  ),
  volume = case_when(
    volume == "null" ~ NA,
    .default = volume
  ))

# dropping the NAs
gold_prices <- gold_prices %>%
  drop_na()

# making the price and volume variables numeric
gold_prices <- gold_prices %>%
  mutate(open = as.numeric(open),
         high = as.numeric(high),
         low = as.numeric(low),
         close = as.numeric(close),
         adj_close = as.numeric(adj_close),
         volume = as.numeric(volume))

# using a midpoint to come up with univariate price data
gold_prices <- gold_prices %>%
  mutate(avg_price = (open + high + low + close)/4)
```

```{r}
# seeing the first 10 rows of the cleaned dataset
gold_prices %>%
  head(10)
```

```{r}
# creating an xts object
xts_gold_prices <- xts(gold_prices$avg_price, order.by = gold_prices$date)

# verifying the number of rows in the dataset
nrow(gold_prices)

# verifying the number of observations in the xts object
nrow(xts_gold_prices)

# viewing the xts object
head(xts_gold_prices)

# plotting the xts object
plot(xts_gold_prices, main = "Gold Prices Extensible Time Series", ylab = "Average Price",
     xlab = "Date")
```

```{r}
# creating a time index column
gold_prices <- gold_prices %>%
```

```
  mutate(time_index = 1:nrow(gold_prices))

# creating a ts object
ts_gold_prices <- ts(gold_prices$avg_price, start = 1, frequency = 1)

# verifying the number of rows in the dataset
nrow(gold_prices)

# verifying the number of observations in the ts object
length(ts_gold_prices)

# viewing the ts object
head(ts_gold_prices)

# plotting the ts object
plot(ts_gold_prices, main = "Gold Prices Time Series", ylab = "Average Price",
     xlab = "Time")
```

```{r}
# getting actual prices to compare with forecasted prices from our models for later
actual_prices <- gold_prices[2950:3150, ]

# creating a ts object
ts_actual_prices <- ts(actual_prices$avg_price, start = 1, frequency = 1)

# plotting the ts object
plot(ts_actual_prices, main = "Gold Prices Time Series", ylab = "Average Price",
```

```
    xlab = "Time")
```

```{r}
gold_prices[3100, ]
# keeping only up to row 3100 in the dataset, which goes up to February 7th, 2013,
# because gold prices crashed in 2013
gold_prices <- gold_prices[1:3100, ]

# creating a new ts object for the reduced dataset
ts_gold_prices <- ts(gold_prices$avg_price, start = 1, frequency = 1)

# verifying the number of rows in the reduced dataset
nrow(gold_prices)

# verifying the number of observations in the new ts object
length(ts_gold_prices)

# viewing the new ts object
head(ts_gold_prices)

# plotting the new ts object
plot(ts_gold_prices, main = "Gold Prices Time Series", ylab = "Average Price",
    xlab = "Time")
```

```{r}
# finding the optimal lambda for the Box-Cox power transformation to stabilize variance
```

```
time_index <- 1:length(ts_gold_prices)
lambda <- boxcox(ts_gold_prices ~ time_index, plotit = T)

# extracting the optimal lambda
optimal_lambda <- lambda$x[which.max(lambda$y)]

# applying the Box-Cox transformation manually
bc_ts_gold_prices <- (ts_gold_prices^optimal_lambda - 1) / optimal_lambda

# viewing the transformed data
head(bc_ts_gold_prices)

# plotting the transformed data
plot(bc_ts_gold_prices, main = "Box-Cox Transformed Data",
    ylab = "Transformed Average Price", xlab = "Time")
```

```{r}
# differencing the data to remove the linear trend
d_bc_ts_gold_prices <- diff(bc_ts_gold_prices, 1)

# plotting the differenced data
plot(d_bc_ts_gold_prices, main = "Differenced Data", ylab = "Differenced Average Price",
    xlab = "Time")
```

```{r}
# checking the ACF and PACF plots
```

```
acf(d_bc_ts_gold_prices, main = "ACF Plot of Gold Prices")
pacf(d_bc_ts_gold_prices, main = "PACF Plot of Gold Prices")
```

```{r}
# fitting models for model selection
sarima(bc_ts_gold_prices, 1, 1, 1) # ARIMA(1, 1, 1), AICc = -9.817208,
# ^ AR(1) and MA(1) components are not significant
sarima(bc_ts_gold_prices, 0, 1, 1) # ARIMA (0, 1, 1), AICc = -9.817589
arima.fit <- sarima(bc_ts_gold_prices, 1, 1, 0) # ARIMA(1, 1, 0), AICc = -9.817515, optimal choice
```

```{r}
# forecasting the next 60 days in the future of my dataset with ARIMA
sarima.for(bc_ts_gold_prices, n.ahead = 60, 1, 1, 0)
sarima.for(ts_gold_prices, n.ahead = 60, 1, 1, 0)
```

```{r}
# getting the returns series from the price series
d_log_ts_gold_prices <- diff(log(ts_gold_prices))
plot(d_log_ts_gold_prices, main = "Gold Returns Time Series", ylab = "Return",
    xlab = "Time")

## checking for GARCH behavior
acf2(d_log_ts_gold_prices, main = "ACF and PACF Plots of Gold Returns")
sarima(d_log_ts_gold_prices, 1, 0, 1) # ARIMA(1, 1, 1), AICc = -6.410395
sarima(d_log_ts_gold_prices, 0, 0, 1) # ARIMA(0, 1, 1), AICc = -6.410944
```

```
d_log_ts_gold_prices_fit <- sarima(d_log_ts_gold_prices, 1, 0, 0) # ARIMA(1, 1, 0), AICc = -6.410004, optimal choice


# checking the squared residuals

acf2(resid(d_log_ts_gold_prices_fit$fit)^2, main = "ACF and PACF Plots of Squared Residuals of Gold Returns")

# ^ Autocorrelation structure in squared residuals suggests GARCH behavior


# checking McLeod-Li and the squared residual plots to assess whether a GARCH type fit is appropriate

McLeod.Li.test(y = d_log_ts_gold_prices)

# ^ McLeod-Li test suggests a GARCH type fit is appropriate
```


```{r}
# ARCH and GARCH fits

arch_fit <- garchFit(~arma(1, 0) + garch(1, 0), d_log_ts_gold_prices, cond.dist='std'); arch.fit

garch_fit <- garchFit(~arma(1, 0) + garch(1, 1), d_log_ts_gold_prices, cond.dist='std'); garch.fit
```


```{r}
# visualization of ARCH and GARCH fits

plot(arch_fit, which = c(3,11))

plot(garch_fit, which = c(3,11))
```


```{r}
# extracting AIC and BIC for ARCH fit

arch_aic <- arch_fit@fit$ics["AIC"]

arch_bic <- arch_fit@fit$ics["BIC"]
```

```
cat("AIC:", arch_aic, "\n")

cat("BIC:", arch_bic, "\n")


# plotting standardized residuals, ACF of standardized residuals, ACF of squared standardized residuals, and QQ-Plot of standardized residuals for ARCH fit

par(mfrow = c(2, 2))  # 2 rows, 2 columns

plot(arch_fit, which = c(9,10,11,13))


# extracting standardized residuals for ARCH fit

arch_std_residuals <- residuals(arch_fit, standardize = TRUE)


# McLeod-Li test for remaining ARCH effects

Box.test(arch_std_residuals^2, lag = 10, type = "Ljung-Box")
```


```{r}

# extracting AIC and BIC for GARCH fit

garch_aic <- garch_fit@fit$ics["AIC"]

garch_bic <- garch_fit@fit$ics["BIC"]

cat("AIC:", garch_aic, "\n")

cat("BIC:", garch_bic, "\n")


# plotting standardized residuals, ACF of standardized residuals, ACF of squared standardized residuals, and QQ-Plot of standardized residuals for GARCH fit

par(mfrow = c(2, 2))  # 2 rows, 2 columns

plot(garch_fit, which = c(9,10,11,13))


# extracting standardized residuals for GARCH fit

garch_std_residuals <- residuals(garch_fit, standardize = TRUE)
```

```
# McLeod-Li test for remaining ARCH effects
Box.test(garch_std_residuals^2, lag = 10, type = "Ljung-Box")
```

```{r}
# forecasting the next 60 days in the future of my dataset with ARCH:
predict(arch_fit, n.ahead = 60, plot = TRUE)
arch_pred <- predict(arch_fit, n.ahead = 60)

# getting the last 150 prices from our reduced data
original_prices <- gold_prices$avg_price[2950:3099]
last_price <- gold_prices$avg_price[3100]
original_time <- 1:length(original_prices)

# getting the predicted returns and confidence intervals
predicted_log_returns <- arch_pred$meanForecast  # predicted returns
lower_log_returns <- arch_pred$meanForecast + arch_pred$meanError - 1.96 *
arch_pred$standardDeviation  # lower bound returns
upper_log_returns <- arch_pred$meanForecast + arch_pred$meanError + 1.96 *
arch_pred$standardDeviation  # upper bound returns

# initializing vectors for predicted prices and intervals
predicted_prices <- numeric(length(predicted_log_returns) + 1)
lower_prices <- numeric(length(predicted_log_returns) + 1)
upper_prices <- numeric(length(predicted_log_returns) + 1)

predicted_prices[1] <- last_price
lower_prices[1] <- last_price
```

```
upper_prices[1] <- last_price

# iteratively calculating prices and intervals
for (t in 2:length(predicted_prices)) {
  predicted_prices[t] <- predicted_prices[t-1] * exp(predicted_log_returns[t-1])
  lower_prices[t] <- lower_prices[t-1] * exp(lower_log_returns[t-1])
  upper_prices[t] <- upper_prices[t-1] * exp(upper_log_returns[t-1])
}

# combining the original and predicted data
combined_time <- c(original_time, (max(original_time) + 1):(max(original_time) +
length(predicted_prices)))
combined_prices <- c(original_prices, predicted_prices)
combined_lower <- c(rep(NA, length(original_prices)), lower_prices)  # NA for original data
combined_upper <- c(rep(NA, length(original_prices)), upper_prices)  # NA for original data

# creating the plot
plot(combined_time, combined_prices, type = "l", col = "black", lwd = 2,
     xlab = "Time", ylab = "Price",
     main = "Price Forecast with 95% Confidence Interval",
     ylim = range(c(combined_prices, combined_lower, combined_upper), na.rm = TRUE))

# adding historical data in black
lines(combined_time[1:length(original_prices)], combined_prices[1:length(original_prices)], col =
"black", lwd = 2)

# adding predicted values in red
lines(combined_time[(length(original_prices) + 1):length(combined_time)],
      combined_prices[(length(original_prices) + 1):length(combined_time)], col = "red", lwd = 2)
```

```
# shading the area between the lower and upper bounds in gray
polygon(c(combined_time[!is.na(combined_lower)], rev(combined_time[!is.na(combined_lower)])),
        c(combined_lower[!is.na(combined_lower)], rev(combined_upper[!is.na(combined_lower)])),
        col = rgb(0.5, 0.5, 0.5, 0.3), border = NA)  # Gray shading with transparency

# adding a vertical line to separate historical and forecast periods
abline(v = max(original_time), col = "black", lty = 2, lwd = 2)

# adding a legend
legend("topleft", legend = c("Historical Prices", "Predicted Prices", "95% Confidence Interval"),
       col = c("black", "red", "gray"), lty = c(1, 1, 1), lwd = 2, bty = "n")
```

```{r}
# forecasting the next 60 days in the future of my dataset with GARCH:
predict(garch_fit, n.ahead = 60, plot = TRUE)
garch_pred <- predict(garch_fit, n.ahead = 60)

# getting the last 150 prices from our reduced data
original_prices <- gold_prices$avg_price[2950:3099]
last_price <- gold_prices$avg_price[3100]
original_time <- 1:length(original_prices)

# getting the predicted returns and confidence intervals
predicted_log_returns <- garch_pred$meanForecast  # predicted returns
lower_log_returns <- garch_pred$meanForecast + garch_pred$meanError - 1.96 *
garch_pred$standardDeviation  # lower bound returns
```

```
upper_log_returns <- garch_pred$meanForecast + garch_pred$meanError + 1.96 *
garch_pred$standardDeviation  # upper bound returns


# initializing vectors for predicted prices and intervals
predicted_prices <- numeric(length(predicted_log_returns) + 1)
lower_prices <- numeric(length(predicted_log_returns) + 1)
upper_prices <- numeric(length(predicted_log_returns) + 1)

predicted_prices[1] <- last_price
lower_prices[1] <- last_price
upper_prices[1] <- last_price

# iteratively calculating prices and intervals
for (t in 2:length(predicted_prices)) {
  predicted_prices[t] <- predicted_prices[t-1] * exp(predicted_log_returns[t-1])
  lower_prices[t] <- lower_prices[t-1] * exp(lower_log_returns[t-1])
  upper_prices[t] <- upper_prices[t-1] * exp(upper_log_returns[t-1])
}

# combining the original and predicted data
combined_time <- c(original_time, (max(original_time) + 1):(max(original_time) +
length(predicted_prices)))
combined_prices <- c(original_prices, predicted_prices)
combined_lower <- c(rep(NA, length(original_prices)), lower_prices)  # NA for original data
combined_upper <- c(rep(NA, length(original_prices)), upper_prices)  # NA for original data

# creating the plot
plot(combined_time, combined_prices, type = "l", col = "black", lwd = 2,
     xlab = "Time", ylab = "Price",
```

```
    main = "Price Forecast with 95% Confidence Interval",
    ylim = range(c(combined_prices, combined_lower, combined_upper), na.rm = TRUE))

# adding historical data in black
lines(combined_time[1:length(original_prices)], combined_prices[1:length(original_prices)], col =
"black", lwd = 2)

# adding predicted values in red
lines(combined_time[(length(original_prices) + 1):length(combined_time)],
     combined_prices[(length(original_prices) + 1):length(combined_time)], col = "red", lwd = 2)

# shading the area between the lower and upper bounds in gray
polygon(c(combined_time[!is.na(combined_lower)], rev(combined_time[!is.na(combined_lower)])),
       c(combined_lower[!is.na(combined_lower)], rev(combined_upper[!is.na(combined_lower)])),
       col = rgb(0.5, 0.5, 0.5, 0.3), border = NA)  # Gray shading with transparency

# adding a vertical line to separate historical and forecast periods
abline(v = max(original_time), col = "black", lty = 2, lwd = 2)

# adding a legend
legend("topleft", legend = c("Historical Prices", "Predicted Prices", "95% Confidence Interval"),
      col = c("black", "red", "gray"), lty = c(1, 1, 1), lwd = 2, bty = "n")
```