Bitcoin - EDA, Feature Engineering and Model Engineering

Muhammad Ali Ahmad, Haider Waseem, Niroshan Srishan, Renato Tizon

5/23/2022

1. Introduction

Within this script we attempt to identify the most appropriate models to forecast the closing price of Ethereum. We use EDA, feature engineering and subsequently evaluate multiple models to attain the appropriate forecasts. A domain of ARIMA models were selected through the ACF, PACF, the extended Autocorrelation Function (EACF), and Bayesian Information Criterion (BIC).

2. Setting up Environment

```
library(TSA)
```

```
## Warning: package 'TSA' was built under R version 4.1.3
```

```
##
## Attaching package: 'TSA'
```

```
## The following objects are masked from 'package:stats':
##
##     acf, arima
```

```
## The following object is masked from 'package:utils':
##
##     tar
```

```
library(fUnitRoots)
```

```
## Warning: package 'fUnitRoots' was built under R version 4.1.3
```

```
## Loading required package: timeDate
```

```
## Warning: package 'timeDate' was built under R version 4.1.2
```

```
##
## Attaching package: 'timeDate'
```

```
## The following objects are masked from 'package:TSA':
##
##     kurtosis, skewness
```

```
## Loading required package: timeSeries

## No methods found in package 'timeDate' for request: '[<-' when loading 'timeSeries'

## Loading required package: fBasics

## Warning: package 'fBasics' was built under R version 4.1.2
```

```
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 4.1.2

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

## Registered S3 methods overwritten by 'forecast':
##   method       from
##   fitted.Arima TSA
##   plot.Arima   TSA
```

```
library(lmtest)
```

```
## Warning: package 'lmtest' was built under R version 4.1.2

## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following object is masked from 'package:timeSeries':
##
##     time<-

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(fGarch)
```

```
## Warning: package 'fGarch' was built under R version 4.1.3
```

```
library(rugarch)
```

```
## Warning: package 'rugarch' was built under R version 4.1.3

## Loading required package: parallel
```

```
##
## Attaching package: 'rugarch'

## The following objects are masked from 'package:fBasics':
##
##     qgh, qnig

## The following object is masked from 'package:stats':
##
##     sigma
```

```
library(tseries)
```

```
## Warning: package 'tseries' was built under R version 4.1.2
```

```
library(readxl)
library(ggplot2)
library(Metrics)
```

```
## Warning: package 'Metrics' was built under R version 4.1.3

##
## Attaching package: 'Metrics'

## The following object is masked from 'package:forecast':
##
##     accuracy
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:timeSeries':
##
##     filter, lag

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)

#setwd('/Users/ali/Documents/University of Chicago/Time Series Analytics/Project')
btc <- read_excel('Bitcoin.xlsx', col_types = c("date", "numeric", "numeric",
                                                "numeric", "numeric", "numeric",
                                                "text"))
```

3. Exploratory Data Analysis

3.1 Data Visualizations

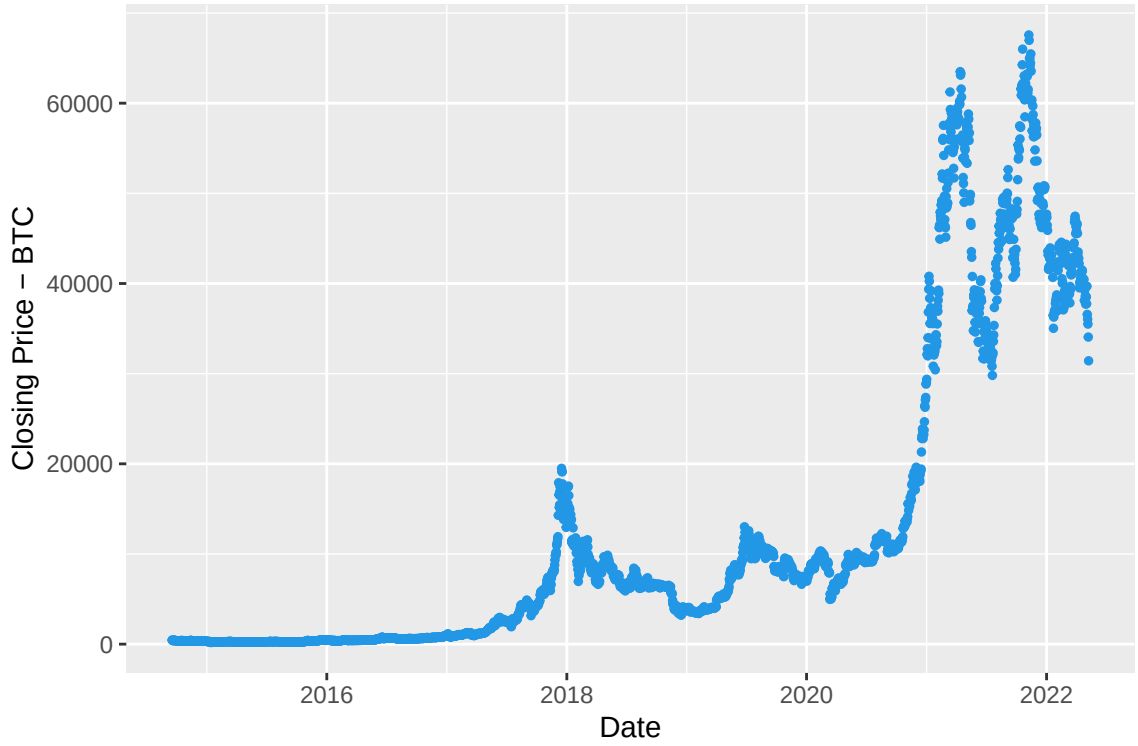
We can draw some conclusions from the graphs plotted below:

1. We observe that within the price, there is no obvious seasonality or intervention point.
2. Compared to the year 2016, there has been a significant upward trend within the price of Bitcoin.
3. Within the last one year, the price has been more erratic with a low closing price of $29796.29 on 21st July, 2021 for the specified time period.
4. As anticipated, there is very strong correlation between the closing price of bitcoin and the previous day closing price.

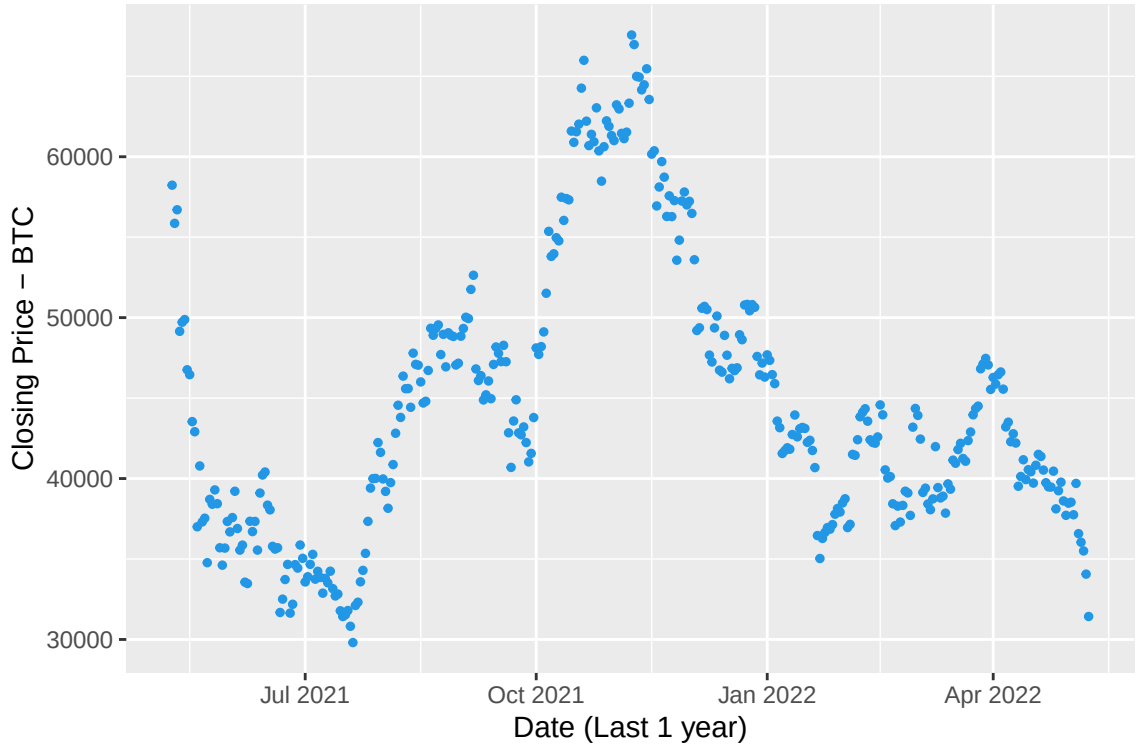
```
nrow(btc)
```

```
## [1] 2792
```

```
btc$year <- as.numeric(format(btc$Date,'%Y'))

ggplot(btc, aes(x = Date, y = Close)) +
  geom_point(size = 1.0, colour = 4) +
  labs(x ="Date", y = "Closing Price - BTC")
```




```
ggplot(btc[2427:2792, ], aes(x = Date, y = Close)) +
  geom_point(size = 1.0, colour = 4) +
  labs(x = "Date (Last 1 year)", y = "Closing Price - BTC")
```




```
ggplot(btc, aes(x = zlag(btc$Close), y = btc$Close)) +
  geom_point(size = 1.0, colour = 4) +
  labs(x = "Previous Day Closing Price", y = "Daily Closing Price")
```

```
## Warning: Use of ‘btc$Close‘ is discouraged. Use ‘Close‘ instead.
```

```
## Warning: Use of ‘btc$Close‘ is discouraged. Use ‘Close‘ instead.
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```


Daily Closing Price
60000
40000
20000
0
0
20000
40000
60000
Previous Day Closing Price

3.2 Finding Autocorrelation at Lag 1

The autocorrelation at Lag 1 is very strong: 0.9988481

```
btc.ts <- ts(as.vector(btc$Close), start = c(2014, 260), frequency = 365)

x = zlag(btc.ts)
index = 2:length(x)
cor(btc.ts[index],x[index])
```

```
## [1] 0.9988481
```

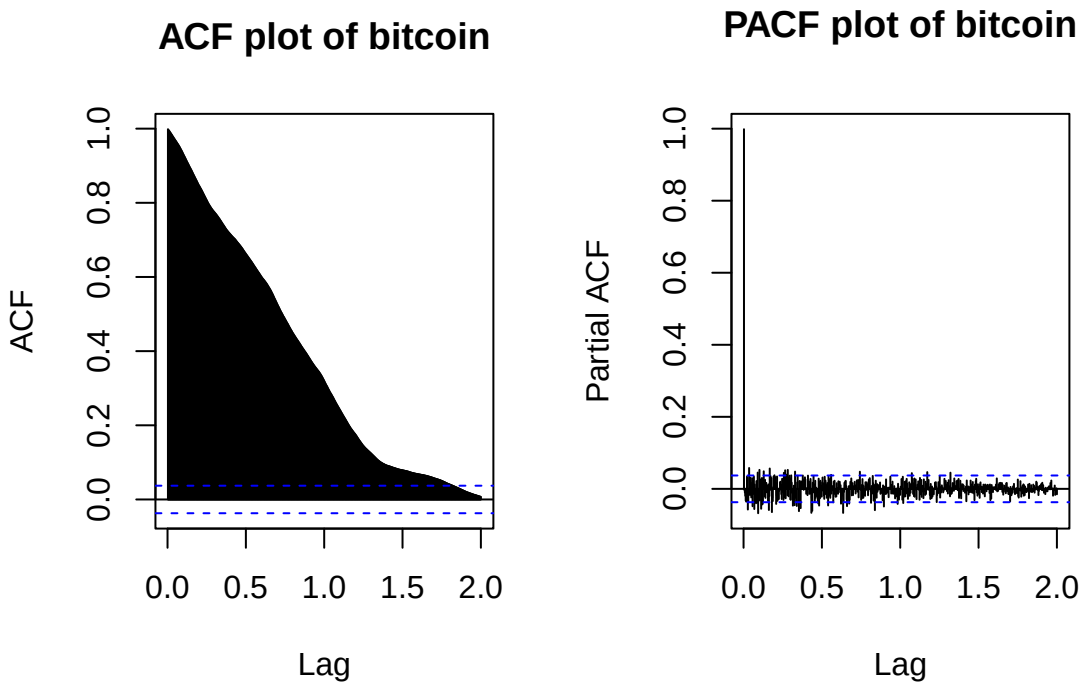
```
acf(btc.ts, lag = 5, pl = FALSE)
```

```
##
## Autocorrelations of series 'btc.ts', by lag
##
## 0.00274 0.00548 0.00822 0.01096 0.01370
##   0.999   0.997   0.996   0.994   0.992
```

3.3 Plotting the ACF and PACF for our Data

We observe a decaying patter in the ACF plot and a very significant first lag, indicating the non-stationarity of our time-series data.

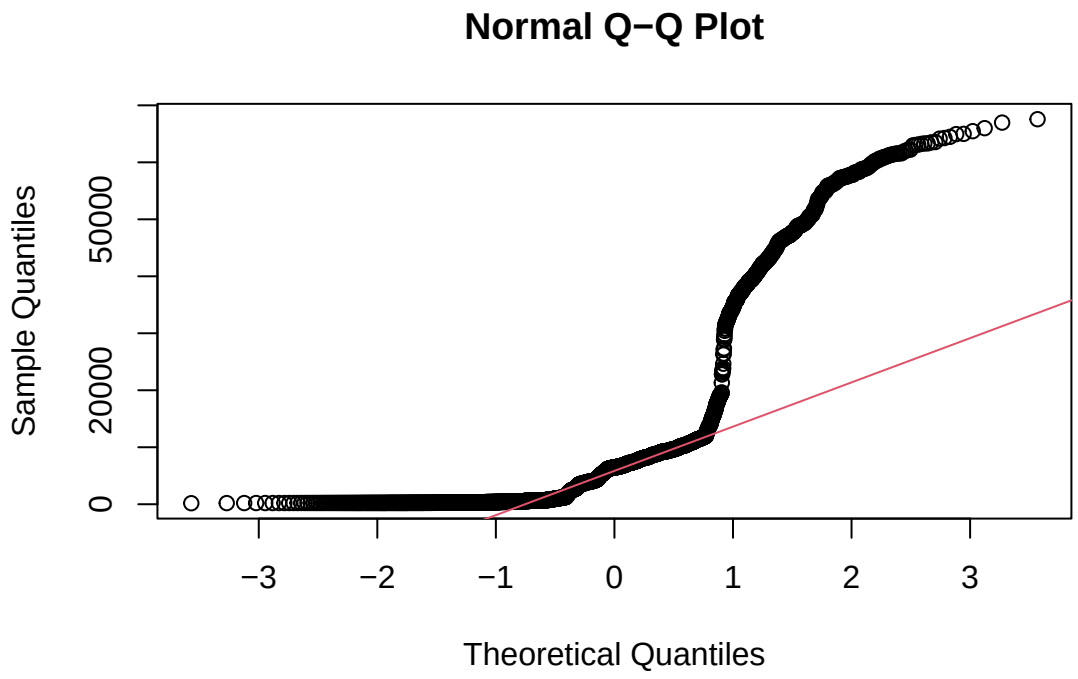
```
btc.log = log(btc.ts)
par(mfrow=c(1,2))
acf(btc.ts, lag.max = 730, main = "ACF plot of bitcoin")
pacf(btc.ts, lag.max = 730, main = "PACF plot of bitcoin")
```



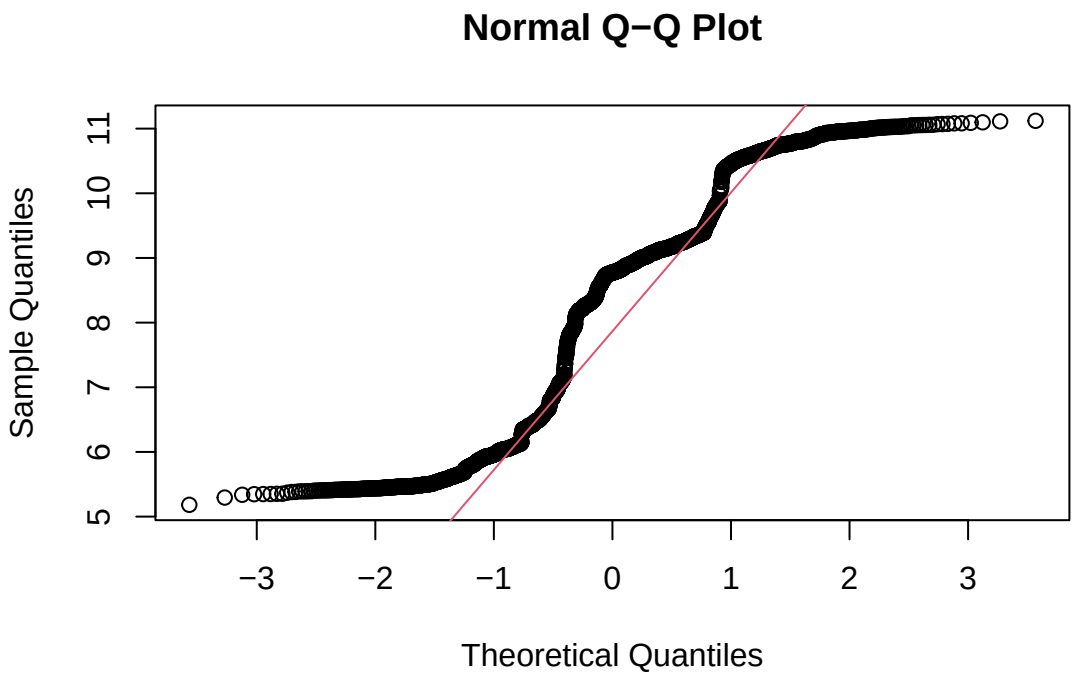
3.4 Data Engineering for Model Development

If we use the QQplot on closing prices of our dataset, we do not obtain a distribution following any obvious patterns. We can check the same assumptions while using a log transformation, and observe that the transformed data is closer to a normal distribution and could perhaps be better utilized for modeling. Lastly, looking at the ADF test, we are unable to reject the null hypothesis that the data is non-stationary.

```
qqnorm(btc.ts)
qqline(btc.ts, col = 2)
```




```
btc.log = log(btc.ts)
par(mfrow=c(1,1))
qqnorm(btc.log)
qqline(btc.log, col = 2)
```




```
diff.bitcoin.t = diff(btc.log, differences = 1)
adf.test(diff.bitcoin.t)
```

```
## Warning in adf.test(diff.bitcoin.t): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff.bitcoin.t
## Dickey-Fuller = -12.9, Lag order = 14, p-value = 0.01
## alternative hypothesis: stationary
```

4. Model Engineering

4.1 Splitting into Test and Train

```
btc$logprice <- log(btc$Close)
df_train = btc[965:2427, ]
df_test = btc[2754:2783, ]


library(dplyr)

var = df_train$Volume - lag(df_train$Volume)
df_train$Vol.Change = var

var_test = df_test$Volume - lag(df_test$Volume)
df_test$Vol.Change = var_test

df_test <- df_test %>%
  mutate(Vol.Change = Volume - lag(Volume))

df_train$Vol.Change[is.na(df_train$Vol.Change)] <-
  mean(df_train$Vol.Change, na.rm = TRUE)

df_test$Vol.Change[is.na(df_test$Vol.Change)] <-
  mean(df_test$Vol.Change, na.rm = TRUE)

train.xts <- ts(as.vector(df_train$logprice), start = c(2017, 128), frequency = 365)
test.xts <- ts(as.vector(df_test$logprice), start = c(2022, 91), frequency = 365)

train_vol.xts <- ts(as.vector(df_train$Vol.Change), start = c(2017, 128), frequency = 365)
test_vol.xts <- ts(as.vector(df_test$Vol.Change), start = c(2022, 91), frequency = 365)
```

4.2 Examining EACF Table

Description (obtained from google): The EACF allows for the identification of ARIMA models. We can summarize this table by understanding that we will observe a value of "x" in the Kth row and Jth column if the lag J+1 sample autocorrelation is significantly different from 0.

```
eacf(diff.bitcoin.t)
```

```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 o o o o o x o o o x o  o  o  o
## 1 x o o o o x x o o x o  o  o  o
## 2 x x o o o x x o o x o  o  o  o
## 3 x x o o o x o o o x o  o  o  o
## 4 x x x x o o o o o o o  o  o  o
## 5 x x x x x x o o o o o  o  o  o
## 6 x x x x x x o o o o o  o  o  o
## 7 x x x x x o x o o o o  o  o  o
```

At the k=0, and J=0, the eacf table returns an O instead of an X and suggests that our TS distribution could be a white noise distribution. We will be selecting the ARMA models around (0, 0).

4.3 Building ARMA Models with differing values of p, d, q

```
model_011 = Arima(train.xts, order = c(0, 1, 1))
df_test$arma_011 = forecast(model_011, h = 30)$mean
smape(df_test$logprice, df_test$arma_011)
```

```
## [1] 0.03170673
```

```
model_012 = Arima(train.xts, order = c(0, 1, 2))
df_test$arma_012 = forecast(model_012, h = 30)$mean
smape(df_test$logprice, df_test$arma_012)
```

```
## [1] 0.03177797
```

```
model_112 = Arima(train.xts, order = c(1, 1, 2))
df_test$arma_112 = forecast(model_112, h = 30)$mean
smape(df_test$logprice, df_test$arma_112)
```

```
## [1] 0.03327167
```

```
df_test %>% select(Date, logprice, arma_011, arma_012, arma_112)
```

```
## # A tibble: 30 x 5
##    Date                logprice arma_011 arma_012 arma_112
##    <dttm>                 <dbl>    <dbl>    <dbl>    <dbl>
##  1 2022-04-01 00:00:00     10.7     11.0     11.0     11.0
##  2 2022-04-02 00:00:00     10.7     11.0     11.0     11.0
##  3 2022-04-03 00:00:00     10.7     11.0     11.0     11.0
##  4 2022-04-04 00:00:00     10.7     11.0     11.0     11.0
##  5 2022-04-05 00:00:00     10.7     11.0     11.0     11.0
##  6 2022-04-06 00:00:00     10.7     11.0     11.0     11.0
##  7 2022-04-07 00:00:00     10.7     11.0     11.0     11.0
##  8 2022-04-08 00:00:00     10.7     11.0     11.0     11.0
##  9 2022-04-09 00:00:00     10.7     11.0     11.0     11.0
## 10 2022-04-10 00:00:00     10.7     11.0     11.0     11.0
## # ... with 20 more rows
```

4.4. Using the Auto Arima function to build our model

```
# library(forecast)

Auto_arima = auto.arima(train.xts)
summary(Auto_arima)
```

```
## Series: train.xts
## ARIMA(0,1,0) with drift
##
## Coefficients:
##         drift
##       0.0024
## s.e.  0.0011
##
## sigma^2 = 0.001839:  log likelihood = 2530.32
## AIC=-5056.65   AICc=-5056.64   BIC=-5046.07
##
## Training set error measures:
##                        ME       RMSE        MAE          MPE      MAPE
## Training set 5.092012e-06 0.04285221 0.02838642 -0.001374662 0.3150767
##                     MASE        ACF1
## Training set 0.03966092 -0.03014191
```

```
df_test$auto_arima = forecast(Auto_arima, h = 30)$mean
smape(df_test$logprice, df_test$auto_arima)
```

```
## [1] 0.03507879
```

```
# AR_xreg = auto.arima(train.xts, xreg = train_vol.xts)
# summary(AR_xreg)
# df_test$AR_xreg = forecast(AR_xreg, xreg = test_vol.xts, h = 30)$mean
# smape(df_test$logprice, df_test$AR_xreg)
```

4.5 Seasonal ARIMA

```
ggplot(btc, aes(x = Date, y = logprice)) +
  geom_point(size = 1.0, colour = 4) +
  labs(x ="Date", y = "Closing Price - BTC")
```


```
Arima.monthly <- auto.arima(ts(df_train$logprice, frequency = 12, start=c(2017, 128)))
df_test$monthlyARIMA = forecast(Arima.monthly, h = 30)$mean
smape(df_test$logprice, df_test$monthlyARIMA)
```

```
## [1] 0.03507879
```

4.6 Comparing our Models

```
Model.Name <- c("Arima (0, 1, 1)", "Arima (0, 1, 2)", "Arima (1, 1, 2)",
                "Auto Arima",
                # "Auto Arima - Xreg",
                "Seasonal ARIMA")

SMAPE <- c(smape(df_test$logprice, df_test$arma_011),
           smape(df_test$logprice, df_test$arma_012),
           smape(df_test$logprice, df_test$arma_112),
           smape(df_test$logprice, df_test$auto_arima),
           # smape(df_test$logprice, df_test$AR_xreg),
           smape(df_test$logprice, df_test$monthlyARIMA))

AIC <- c(model_011$aic, model_012$aic,
         model_112$aic, Auto_arima$aic,
         # AR_xreg$aic,
         Arima.monthly$aic)

BIC <- c(model_011$bic, model_012$bic,
         model_112$bic, Auto_arima$bic,
         # AR_xreg$bic,
         Arima.monthly$bic)

RMSE <- c(rmse(df_test$logprice, df_test$arma_011),
          rmse(df_test$logprice, df_test$arma_012),
          rmse(df_test$logprice, df_test$arma_112),
          rmse(df_test$logprice, df_test$auto_arima),
          # rmse(df_test$logprice, df_test$AR_xreg),
          rmse(df_test$logprice, df_test$monthlyARIMA))
```

```
Rsquared <- c(cor(df_test$logprice, df_test$arma_011)^2,
              cor(df_test$logprice, df_test$arma_012)^2,
              cor(df_test$logprice, df_test$arma_112)^2,
              cor(df_test$logprice, df_test$auto_arima)^2,
              # cor(df_test$logprice, df_test$AR_xreg)^2,
              cor(df_test$logprice, df_test$monthlyARIMA)^2)
```

```
## Warning in cor(df_test$logprice, df_test$arma_011): the standard deviation is
## zero
```

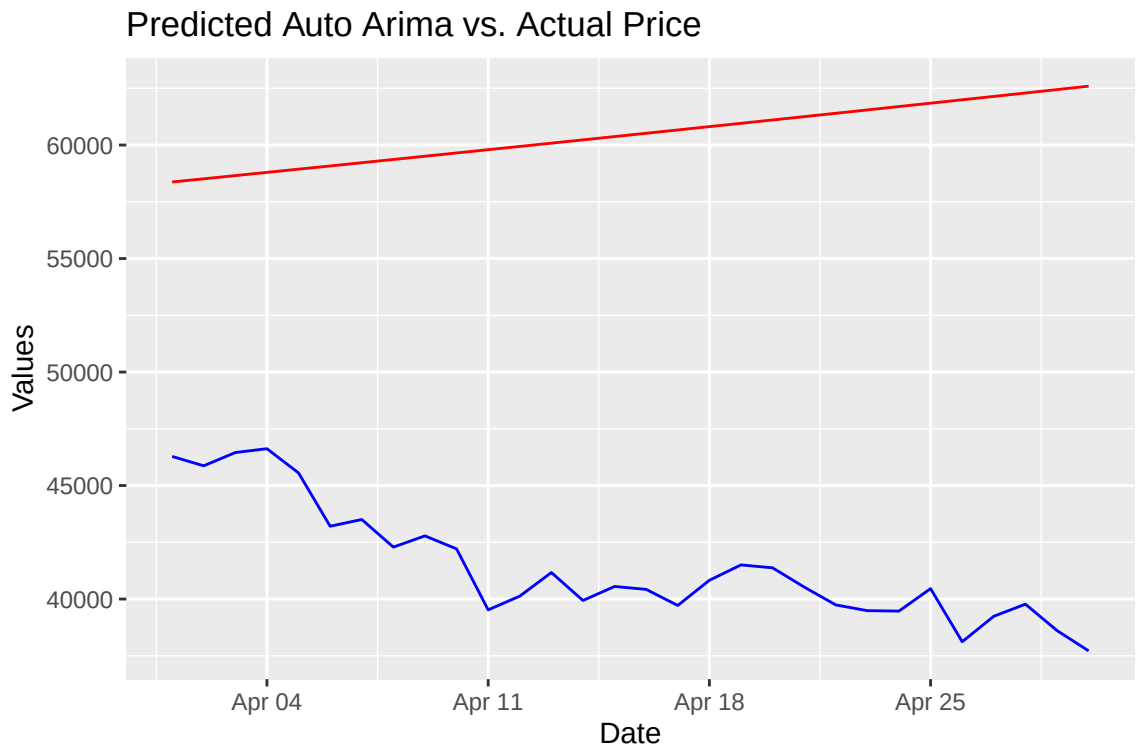
```
results <- data.frame(Model.Name, AIC, SMAPE, RMSE, Rsquared)
results
```

```
##        Model.Name       AIC      SMAPE      RMSE  Rsquared
## 1 Arima (0, 1, 1) -5053.019 0.03170673 0.3473849        NA
## 2 Arima (0, 1, 2) -5054.001 0.03177797 0.3481507 0.1244026
## 3 Arima (1, 1, 2) -5054.128 0.03327167 0.3657190 0.7924382
## 4      Auto Arima -5056.649 0.03507879 0.3873542 0.7678484
## 5  Seasonal ARIMA -5056.649 0.03507879 0.3873542 0.7678484
```

4.7 Forecasting Chosen Model

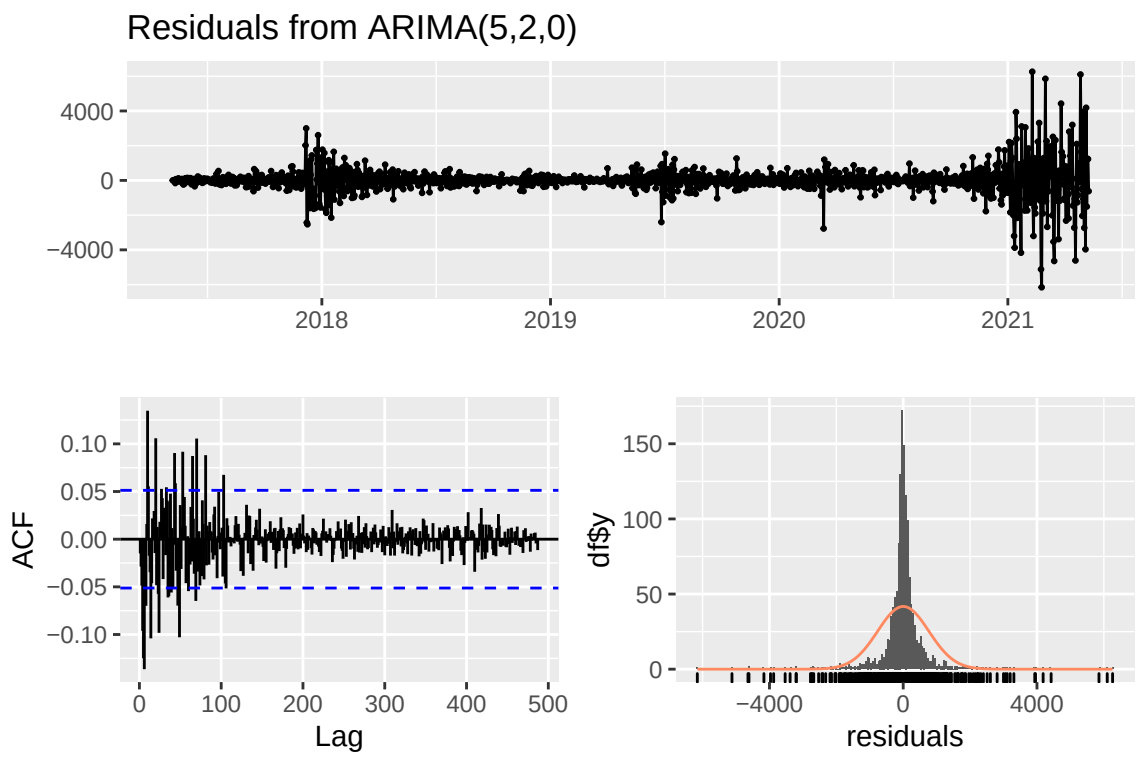
```
pred_auto_arima = exp(df_test$auto_arima)
df_test$pred_auto_arima = pred_auto_arima

ggplot() + geom_line(aes(x=df_test$Date, y=df_test$pred_auto_arima), group=1, colour = 'red') +
           geom_line(aes(x=df_test$Date, y=df_test$Close), group = 2, colour = 'blue') +
           ylab('Values')+xlab('Date') +
            labs(title="Predicted Auto Arima vs. Actual Price")
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```



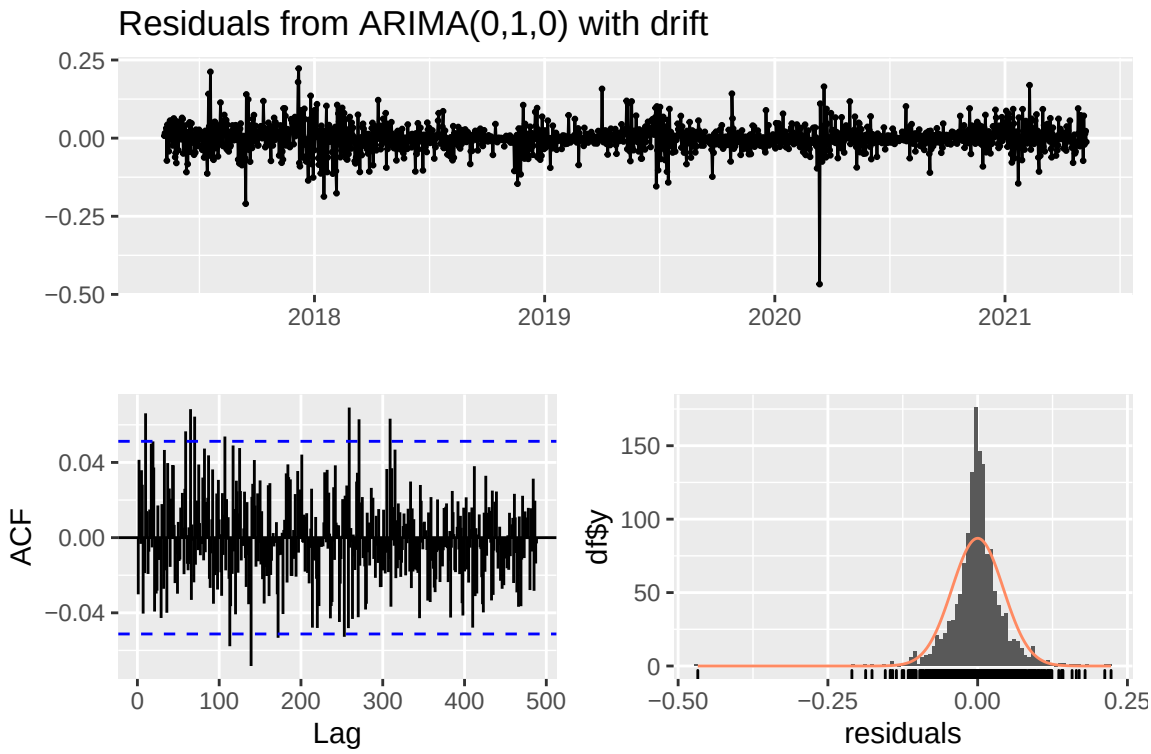
4.8 Residual Analysis

```
train.xts <- ts(as.vector(df_train$Close), start = c(2017, 128), frequency = 365)
Auto_arima_normal = auto.arima(train.xts)
#test.xts <- ts(as.vector(df_test$Close), start = c(2022, 91), frequency = 365)
checkresiduals(Auto_arima_normal)
```



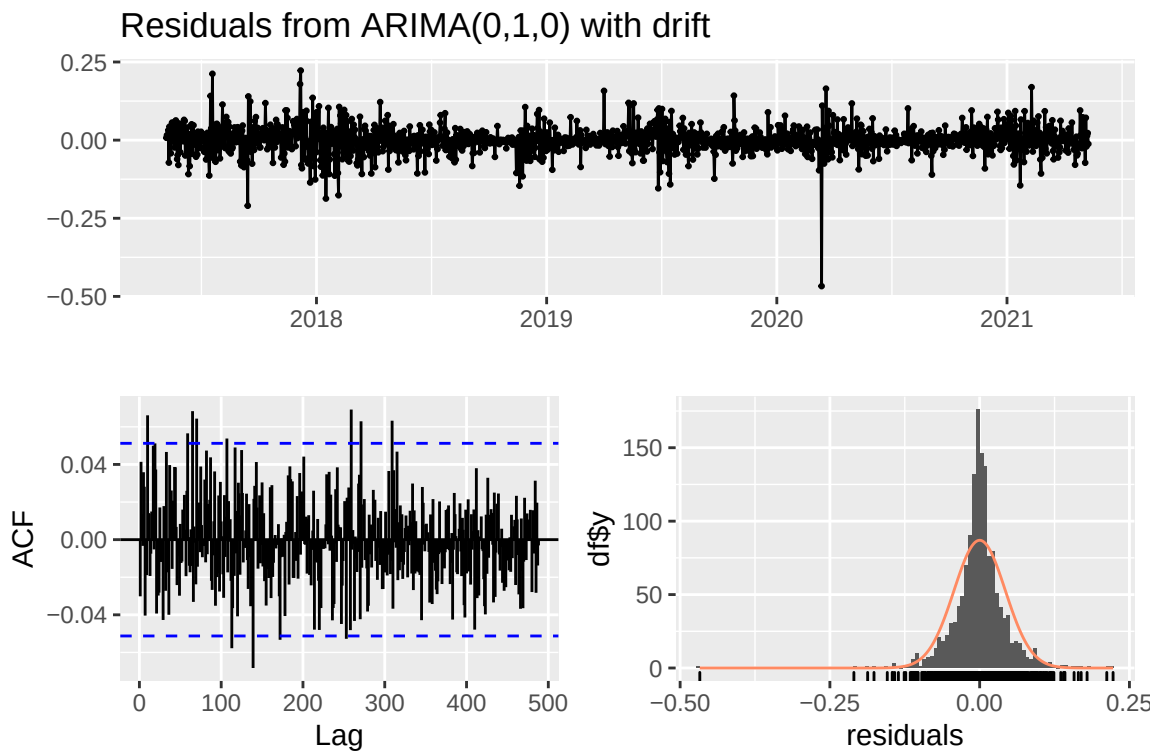

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(5,2,0)
## Q* = 447.21, df = 288, p-value = 5.169e-09
##
## Model df: 5.   Total lags used: 293
```

```
checkresiduals(Auto_arima)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,0) with drift
## Q* = 292.87, df = 292, p-value = 0.4746
##
## Model df: 1.   Total lags used: 293
```

```
checkresiduals(forecast(Auto_arima, h = 30))
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,0) with drift
## Q* = 292.87, df = 292, p-value = 0.4746
##
## Model df: 1.   Total lags used: 293
```