BITCOIN PRICE PREDICTION

Forecasting Methods Final Project - Bitcoin Price Prediction

Prakhar Goyal, Hardik Gupta, Sidharth Gaur & Mudit Verma

Executive Summary

Problem Statement

Bitcoin is the longest running and most well-known cryptocurrency, first released as open source in 2009 by the anonymous Satoshi Nakamoto. Bitcoin serves as a decentralized medium of digital exchange, with transactions verified and recorded in a public distributed ledger (the blockchain) without the need for a trusted record keeping authority or central intermediary. The dataset includes historical bitcoin market data at 1-min intervals for select bitcoin exchanges where trading takes place.

Variable Names:

1. Timestamp: Start time of time window (60s window), in Unix time
2. Open: Open price at start time window
3. High: High price within time window
4. Low: Low price within time window
5. Close: Close price at end of time window
6. Volume_(BTC): Amount of BTC transacted in time window
7. Volume_(Currency): Amount of Currency transacted in time window
8. Weighted_Price: volume-weighted average price (VWAP)

We would be using time series forecasting methods to predict the price of Bitcoin for future dates.

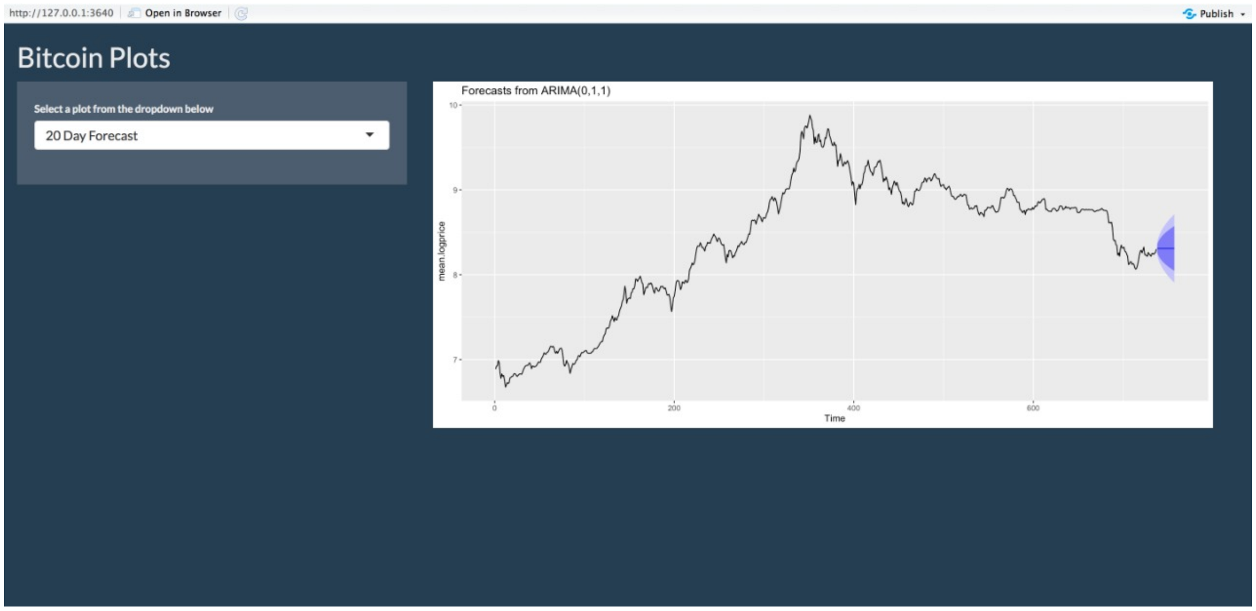
Approach

We have taken the following steps to reach the goal of the project: • Analyze the dataset to develop initial hypotheses • Understand which variables can help predict Bitcoin price • Data Cleaning and Exploratory Data Analysis • Check Stationarity using Dicky Fuller • Seasonal Decomposition of Data • Model Identification and comparison with similar models • Develop model to predict the price • Residual analysis • Forecast for future dates

Major Results

Based on the analysis of the dataset, we were able to build an **ARIMA (0,1,1)** model with a seasonal component which has the RMSE of **0.04**.

Shiny App Snapshot:



Data Cleaning and Modelling

```
coin <- read.csv("C:/Users/mudit/Documents/Spring Sem/Time Series and
Forecasting/bitcoin-historical-data/coinbaseUSD_1-min_data_2014-12-
01_to_2019-01-09.csv")

str(coin)

## 'data.frame':    2099760 obs. of  8 variables:
##  $ Timestamp        : int  1417411980 1417412040 1417412100 1417412160
1417412220 1417412280 1417412340 1417412400 1417412460 1417412520 ...
##  $ Open             : num  300 NaN NaN NaN NaN NaN NaN 300 NaN NaN ...
##  $ High             : num  300 NaN NaN NaN NaN NaN NaN 300 NaN NaN ...
##  $ Low              : num  300 NaN NaN NaN NaN NaN NaN 300 NaN NaN ...
##  $ Close            : num  300 NaN NaN NaN NaN NaN NaN 300 NaN NaN ...
##  $ Volume_.BTC.     : num  0.01 NaN NaN NaN NaN NaN NaN 0.01 NaN NaN ...
##  $ Volume_.Currency.: num  3 NaN NaN NaN NaN NaN NaN 3 NaN NaN ...
##  $ Weighted_Price   : num  300 NaN NaN NaN NaN NaN NaN 300 NaN NaN ...

summary(coin)

##    Timestamp               Open               High
##  Min.   :1.417e+09   Min.   :    0.06   Min.   :    0.06
##  1st Qu.:1.452e+09   1st Qu.:  419.58   1st Qu.:  419.64
##  Median :1.484e+09   Median : 1014.58   Median : 1014.89
##  Mean   :1.484e+09   Mean   : 3246.40   Mean   : 3247.83
##  3rd Qu.:1.515e+09   3rd Qu.: 6322.63   3rd Qu.: 6324.01
##  Max.   :1.547e+09   Max.   :19891.99   Max.   :19891.99
##                      NA's   :109069     NA's   :109069
##       Low                Close            Volume_.BTC.
##  Min.   :    0.06   Min.   :    0.06   Min.   :   0.00
##  1st Qu.:  419.50   1st Qu.:  419.57   1st Qu.:   0.90
##  Median : 1014.15   Median : 1014.53   Median :   2.69
##  Mean   : 3244.86   Mean   : 3246.40   Mean   :   7.85
##  3rd Qu.: 6321.09   3rd Qu.: 6322.67   3rd Qu.:   7.60
##  Max.   :19891.98   Max.   :19891.99   Max.   :1563.27
##  NA's   :109069     NA's   :109069     NA's   :109069
##  Volume_.Currency.  Weighted_Price
##  Min.   :       0   Min.   :    0.06
##  1st Qu.:     644   1st Qu.:  419.56
##  Median :    3696   Median : 1014.51
##  Mean   :   36002   Mean   : 3246.34
##  3rd Qu.:   19724   3rd Qu.: 6322.55
##  Max.   :19970765   Max.   :19891.99
##  NA's   :109069     NA's   :109069
```

The Dataset consists of **2099760** observations under **8** variables. We have however, taken a daily average value of weighted price for our predictions. We can observe that there are many **missing values** in the dataset, so imputation or removal of them is required.

Also, we want to work on daily prices and currently prices for every minute are present in the dataset. So, in the next part we'll take care of the missing values and take average daily price.

We also observe that the price values are distributed over a large range, so we'll be using log transformation while building the models

```
coin$date <- as.Date(as.POSIXct(coin$Timestamp, origin="1970-01-01"))

library(lubridate)

##
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':
##
##     date

library(dplyr)

coin$year <- year(coin$date)

median_price_NAs <- coin %>%
  group_by(year) %>%
  summarise_each(funs(sum(is.nan(.))))

## Warning: funs() is soft deprecated as of dplyr 0.8.0
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
```

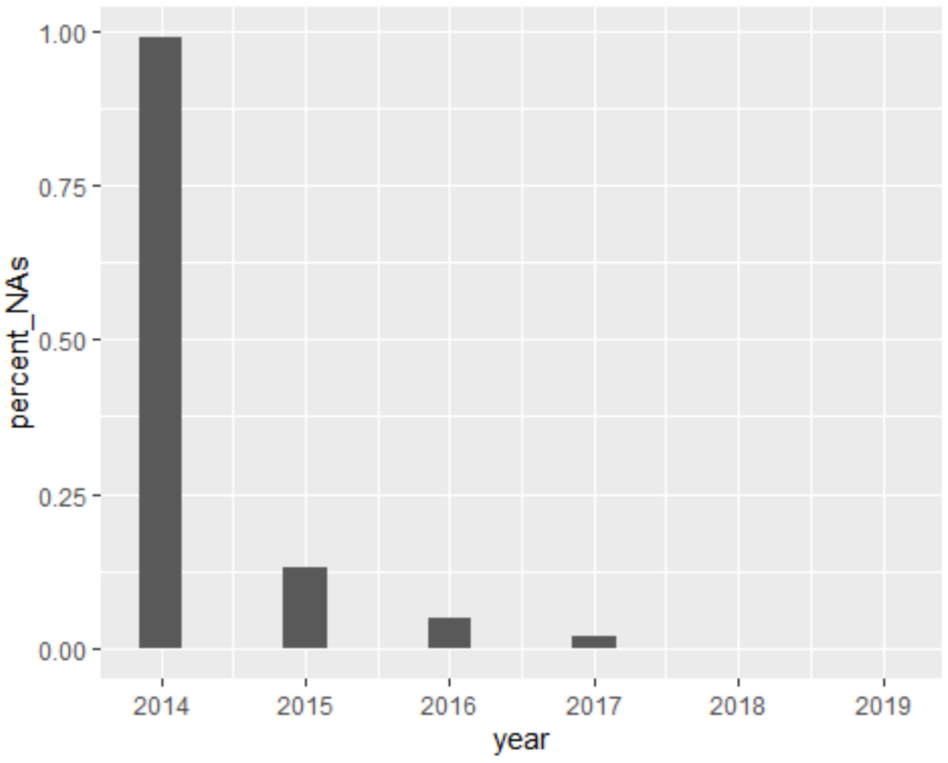
```
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once per session.
```

```
year_month_count <- coin %>%
  group_by(year) %>%
  tally(name = 'count')

price_with_time <- merge(median_price_NAs, year_month_count)

price_with_time$percent_NAs <-
  round(price_with_time$Weighted_Price / price_with_time$count, 2)

# visualising percentage NAs by year
ggplot(data = price_with_time, aes(x = year, y = percent_NAs)) +
  geom_bar(stat = "identity", width = 0.3)
```



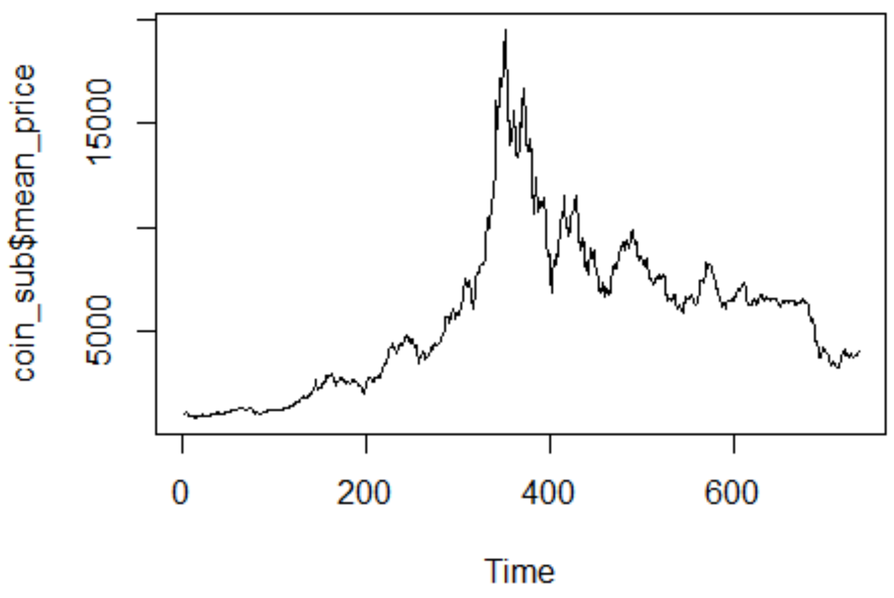
We can see from the above graph that the missing values are very high until 2016. So we'll remove that part of data and use price data 2017 onwards for prediction.

```
coin_clean <- na.omit(coin)
coin_clean$date <- as.Date(as.POSIXct(coin_clean$Timestamp, origin="1970-01-
01"))
```

```r
coin_daily_price <- coin_clean %>% group_by(date) %>% summarise(mean_price =
mean(Weighted_Price))

coin_sub <- subset(coin_daily_price, date > "2016-12-31")

plot.ts(coin_sub$mean_price)
```



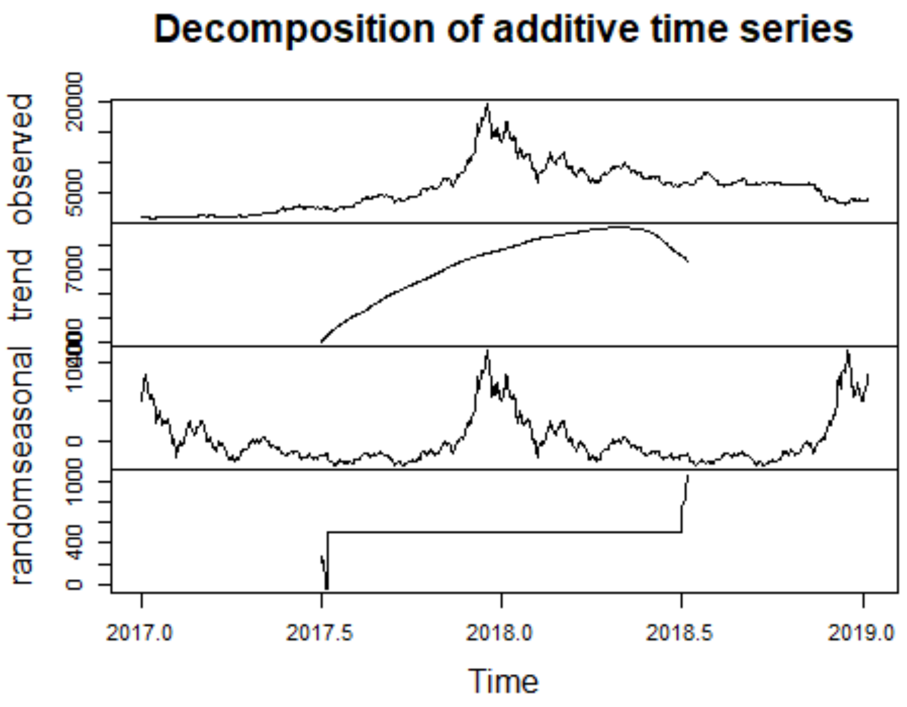
The above plot shows the variation of price at daily level from 2017.

In the next part, we'll check the seasonality of data by decomposing the time series.

```r
library(forecast)

ts_train <- ts(coin_sub$mean_price, frequency = 365, start = c(2017,1))

coin_decomp <- decompose(ts_train)
plot(coin_decomp)
```



From the above plot, we can see that time series doesn't look stationary and there might be a seasonal trend.

Checking the Stationarity using Dicky Fuller Test

```
# Stationarity Check

log(1+coin_sub$mean_price) %>% adf.test()

##
##  Augmented Dickey-Fuller Test
##
## data:  .
## Dickey-Fuller = -0.78174, Lag order = 9, p-value = 0.9632
## alternative hypothesis: stationary

# Price is distributed across a wide range
```

The p-value is greater than 0.05 which means that we fail to reject the null hypothesis and time series is not stationary. So we use differencing to make it stationary.

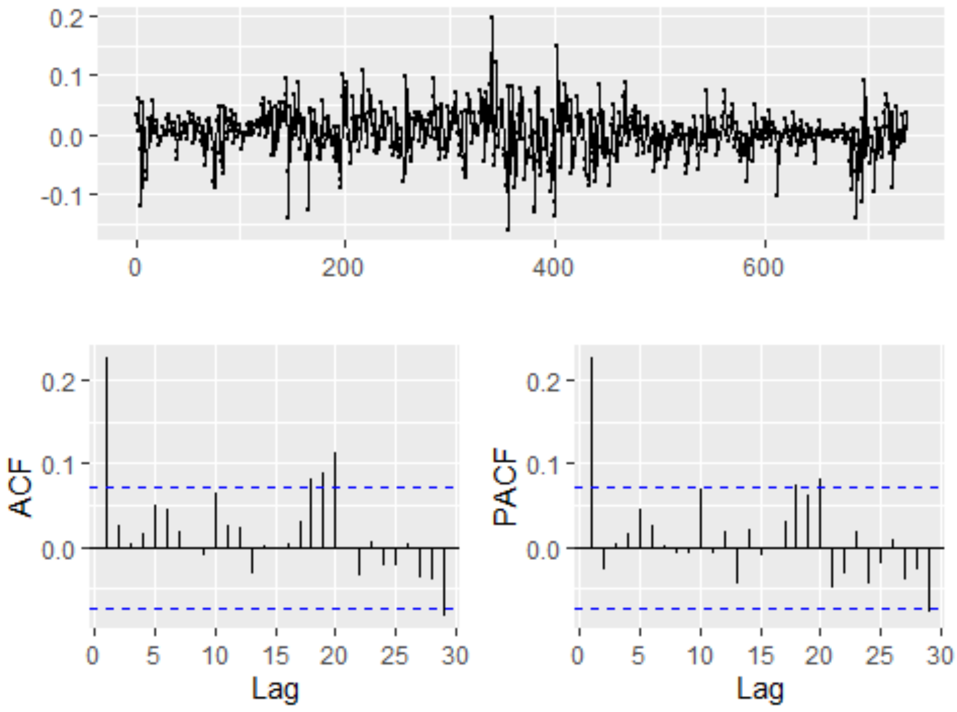
```
##Proof of stationarity using dicky fuller
log(1+coin_sub$mean_price) %>% diff() %>% diff(lag=1) %>% adf.test()

##
##  Augmented Dickey-Fuller Test
##
## data:  .
## Dickey-Fuller = -14.459, Lag order = 9, p-value = 0.01
## alternative hypothesis: stationary
```

Now, the p-value is less than 0.05 and we have successfully converted our data to a stationary time series.

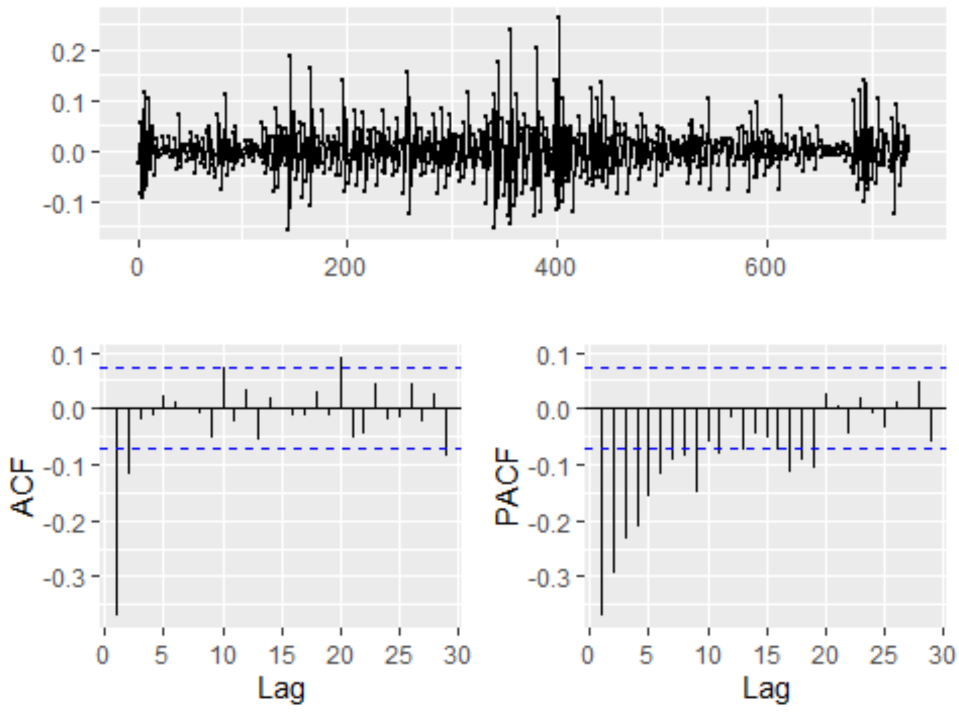
Let us now look at the seasonal component: First we find seasonal difference , which looks good at lag=1.

```
diff(log(1+coin_sub$mean_price),lag=1) %>% ggtsdisplay()
```



Then we come to non seasonal differencing

```
log(1+coin_sub$mean_price) %>% diff() %>% diff(lag=1) %>% ggtsdisplay()
#ordinary diff=1
```



There are significant spikes at lag 1, and lag 2 in the ACF which suggests a non-seasonal MA(2) component.

Fitting the MA component

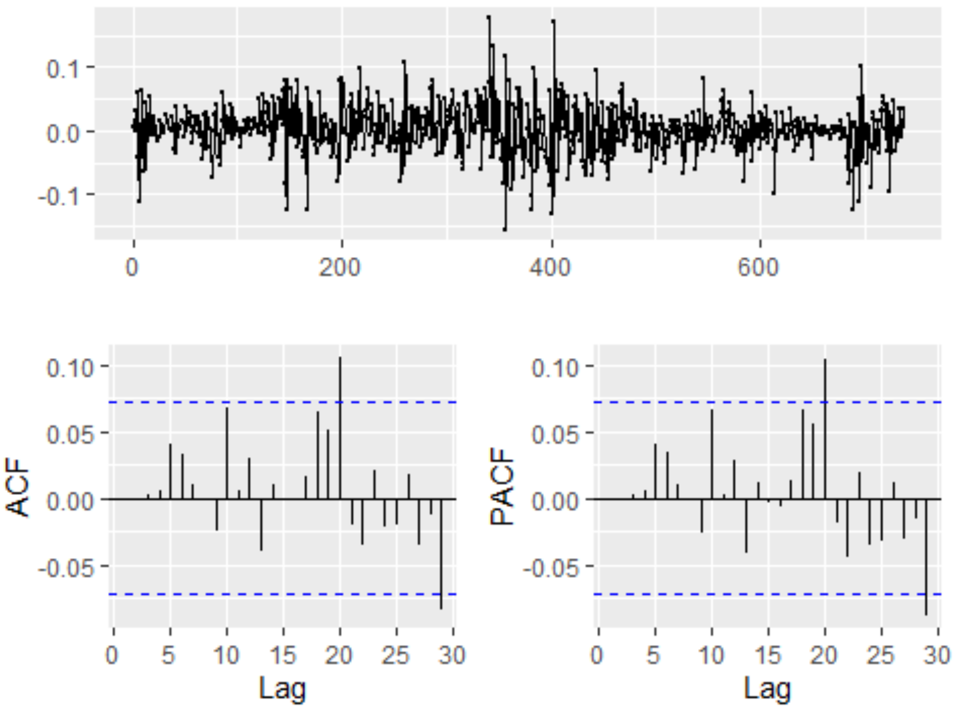
```
mean.logprice <- log(1+coin_sub$mean_price)

(inital_fit <- Arima(mean.logprice, order=c(0,1,2), seasonal=c(0,1,0)))

## Series: mean.logprice
## ARIMA(0,1,2)
##
## Coefficients:
##          ma1     ma2
##       0.2349  0.0275
## s.e.  0.0368  0.0371
##
## sigma^2 estimated as 0.001423:  log likelihood=1368.87
## AIC=-2731.75   AICc=-2731.71   BIC=-2717.94
```

By looking at the coefficients, MA2 doesn't look significant and we might have to remove it. Let us look at residual plots to confirm the same.

```
inital_fit %>%
  residuals() %>% ggtsdisplay()
```



We still see seasonal peacks in the ACF and PACF plot suggesting we need to try more seasonal orders. And we have seen earlier that ma2 was insignificant. So we can try MA(1) model.

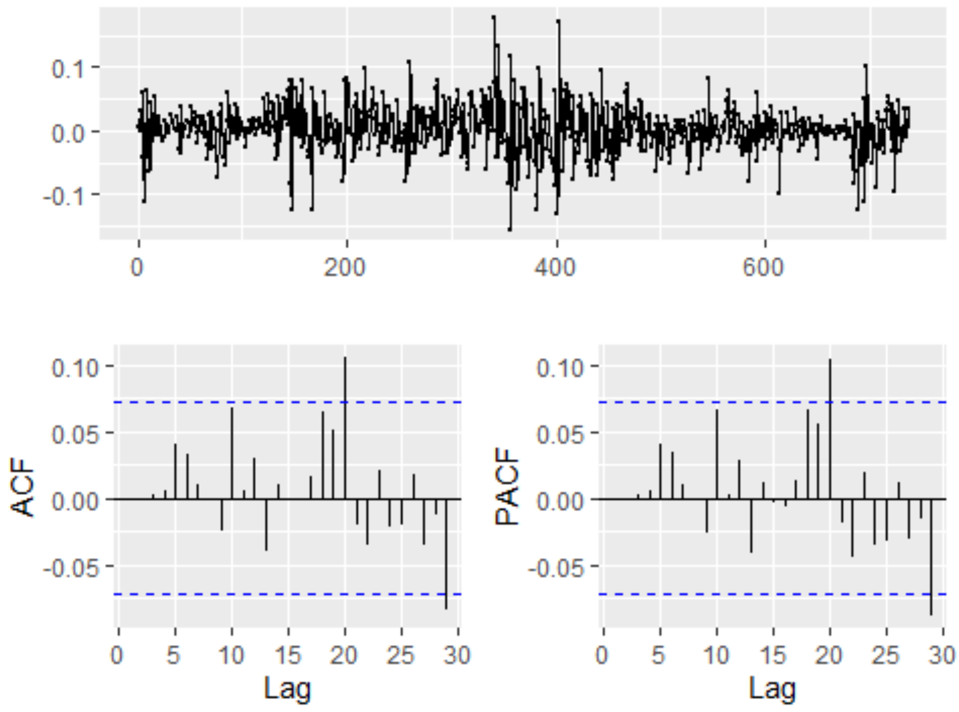
```
(updated_fit <- Arima(mean.logprice, order=c(0,1,1), seasonal=c(0,1,0)))

## Series: mean.logprice
## ARIMA(0,1,1)
##
## Coefficients:
##          ma1
##       0.2288
## s.e.  0.0350
##
## sigma^2 estimated as 0.001422:  log likelihood=1368.6
## AIC=-2733.2   AICc=-2733.18   BIC=-2724
```

Our coefficients of the MA1 is significant with lesser AIC, hence we would stick to this updated model.

Residual Diagnostics of Final Model

```
inital_fit %>%
  residuals() %>% ggtsdisplay()
```



We still see peaks and try models with different AR and MA orders, but we donot get better AIC than this.

Summary of Final Model

```
summary(updated_fit)
```

```
## Series: mean.logprice
## ARIMA(0,1,1)
##
## Coefficients:
##          ma1
##       0.2288
## s.e.  0.0350
##
## sigma^2 estimated as 0.001422:  log likelihood=1368.6
## AIC=-2733.2   AICc=-2733.18   BIC=-2724
##
## Training set error measures:
##                       ME       RMSE        MAE       MPE      MAPE
## Training set 0.001577124 0.03766058 0.02711671 0.0200653 0.3224285
##                   MASE        ACF1
## Training set 0.9735932 0.004808117
```

So, our model has a RMSE of **0.04** which is quite good.

Testing for Model Assumption - Ljung-Box Test

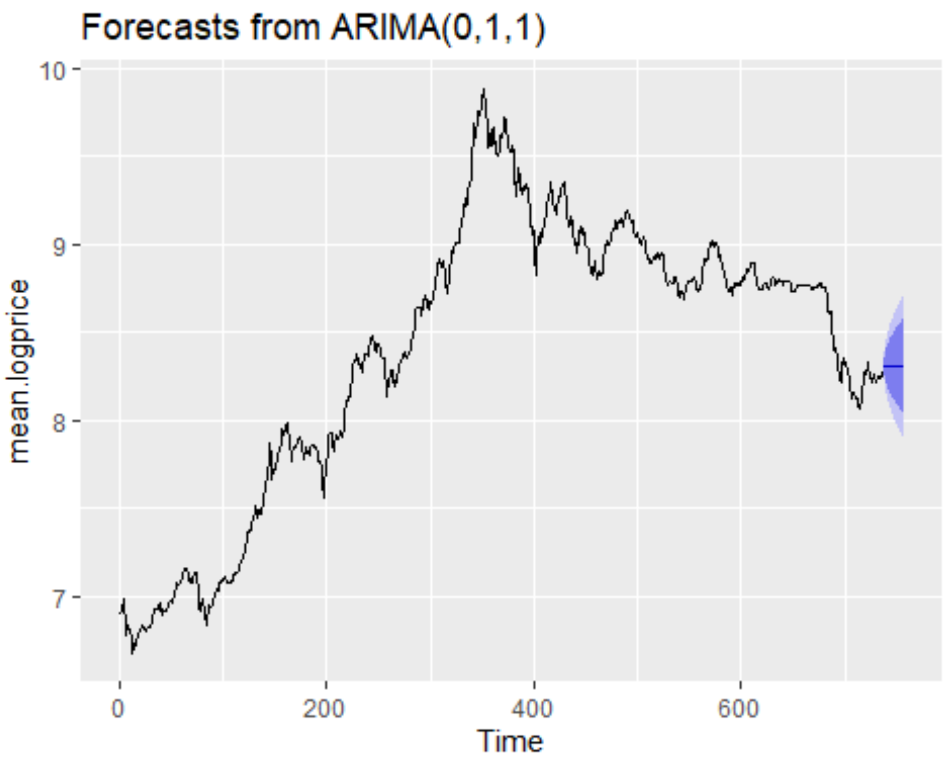
```r
checkresiduals(updated_fit, plot = F)
```

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,1)
## Q* = 6.7409, df = 9, p-value = 0.6641
##
## Model df: 1.   Total lags used: 10
```

As p-value>0.05 we do not have enough evidence to reject the null hypothesis, hence we can conclude that our mode assumptions are not being violated.

Forecasting Price using Final Model

```r
#Forecasting
updated_fit %>% forecast(h=20) %>% autoplot()
```


Forecasts from ARIMA(0,1,1)
mean.logprice
Time