Application for Quantitative Analyst at Brave New Coin

Tushar Sawhney

31 October 2018

Question Two

How would you approach a price prediction model for Litecoin?

For this question, I have conducted some analysis in R using the `tidyverse` packages, among others, and also built a small reproducible implementation. All code is included in the document itself aswell as my GitHub repo.

```
library(tidyverse)
library(janitor)
library(stringr)
library(tidyquant)
library(knitr)
```

Extract Data

Before we can proceed with building a model, it is necessary for us to get data on the price of litecoin itself. To do this I have used the AlphaVantage service (found on https://www.alphavantage.co) to get free pricing data on Crypto-currency.

I am able to get a free API key by registering on their website (https://www.alphavantage.co/support/#api-key) and then we can use the `alphavantager` R library to interact with their API. For the purposes of security I have created a hidden folder `.credentials` in which I have stored my API key.

Let's also write it into a data folder in our project so we don't need to always call the API for retrieval. To make my results reproducible, I have commented the code above, and the data for Litecoin is present in this Git repo.

```
# library(alphavantager)
# # Read in credentials for alphavantage
# creds <- readLines(".credentials/alphavantage")
#
# # Authenticate Credentials
# av_api_key(creds)
#
# # Extract Data from AlphaVantage
# d_ltc <- av_get(symbol = "LTC", av_fun = "DIGITAL_CURRENCY_DAILY", market = "USD")
# d_ltc %>% write_csv("data/ltc.csv")
```

Load Data

```
d_ltc <- read_csv("data/ltc.csv") %>%
  clean_names() %>%
  select(-ends_with("1"))
```

We have now loaded in our data and removed some duplicate columns.

This data was from April 1st, 2014 and contains important information about LTC (the altacoin we are considering), which may be used for a price prediction model.

Within the dataset we have the following columns:

```
- timestamp: Trading day, in YYYY-MM-DD
- open_usd: Opening price of the coin in \$ USD
- high_usd: High Price of the coin for the trading day in \$ USD
- low_usd: Low Price of the coin for the trading day in \$ USD
- volume: It's a broad and largely unadjusted measure of the total value of outputs on the blockchain,
- market_cap_usd: This is the `close_usd` column multiplied by the `volume` (Unit price * Number of uni
```

We can have a look at what the data looks like in the table below and also check if there are any missing values in any of the columns.

```
d_ltc %>%
  head() %>%
  kable(caption = "First 6 rows in our LTC data")
```

Table 1: First 6 rows in our LTC data

timestamp	open_usd	high_usd	low_usd	close_usd	volume	market_cap_usd
2014-04-01	13.52934	13.64550	13.15001	13.23114	367733.0	4865526
2014-04-02	13.18309	13.44053	10.96663	11.31391	893306.1	10106782
2014-04-03	11.29754	11.33187	10.26154	11.11005	658963.2	7321116
2014-04-04	11.09974	11.44777	10.73905	11.09453	335313.1	3720140
2014-04-05	11.04711	11.30656	10.83038	11.25180	151144.9	1700652
2014-04-06	11.26076	11.97720	11.08925	11.65064	263571.0	3070771

```
d_ltc %>%
  # Find number of missing values
  summarise_all(funs(missing = sum(is.na(.)))) %>%
  gather(Variable, `Number Missing`) %>%
  mutate(Variable = str_replace(Variable,pattern = "_missing", "")) %>%
  kable(caption = "Missing Data")
```

Table 2: Missing Data

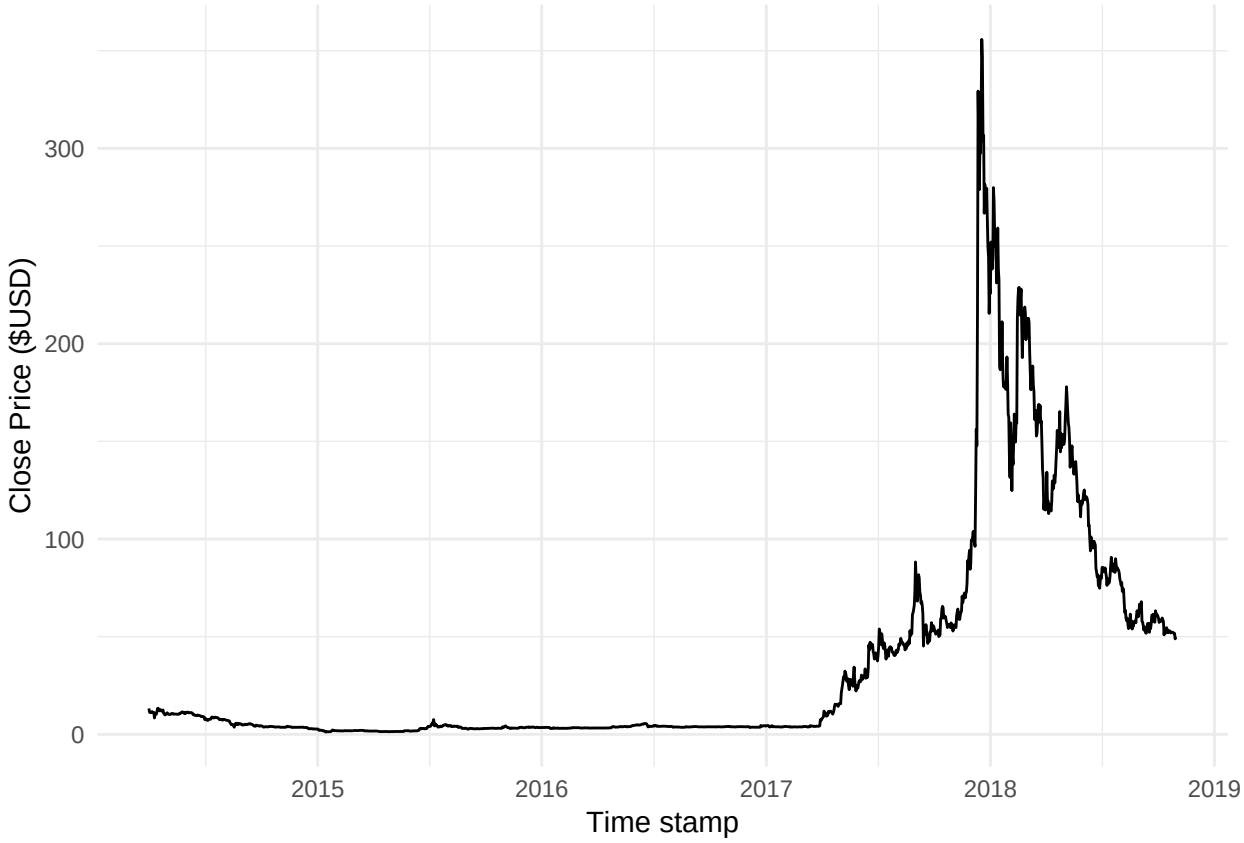
Variable	Number Missing
timestamp	0
open_usd	0
high_usd	0
low_usd	0
close_usd	0
volume	0
market_cap_usd	0

Awesome, our data looks clean and has no missing data points (not suprising as from an API) and we can begin with some exploratory data analysis.

Explore Data

Now we have seen our data is clean, lets plot it. In this piece of work, I have made the target the variable `close_usd` and as such our goal is to predict the Closing Price in ($USD)

```
d_ltc %>%
  ggplot(aes(x = timestamp, y = close_usd)) +
  geom_line() +
  labs(
    x = "Time stamp",
    y = "Close Price ($USD)"
  ) +
  theme_minimal()
```

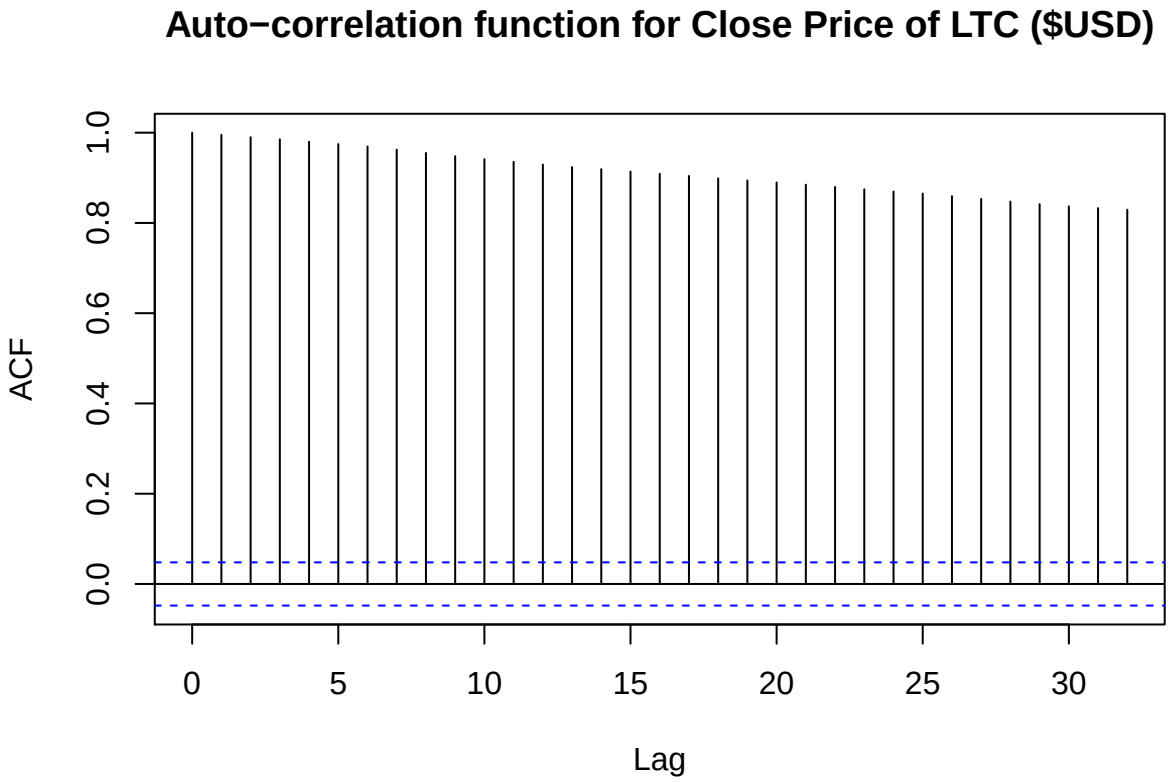


Since our data is time series, we can do some simple time series analysis on it to see whether or not it satisfies some stationarity assumptions:

- Constant Mean
- Constant Variance
- No Correlation between errors

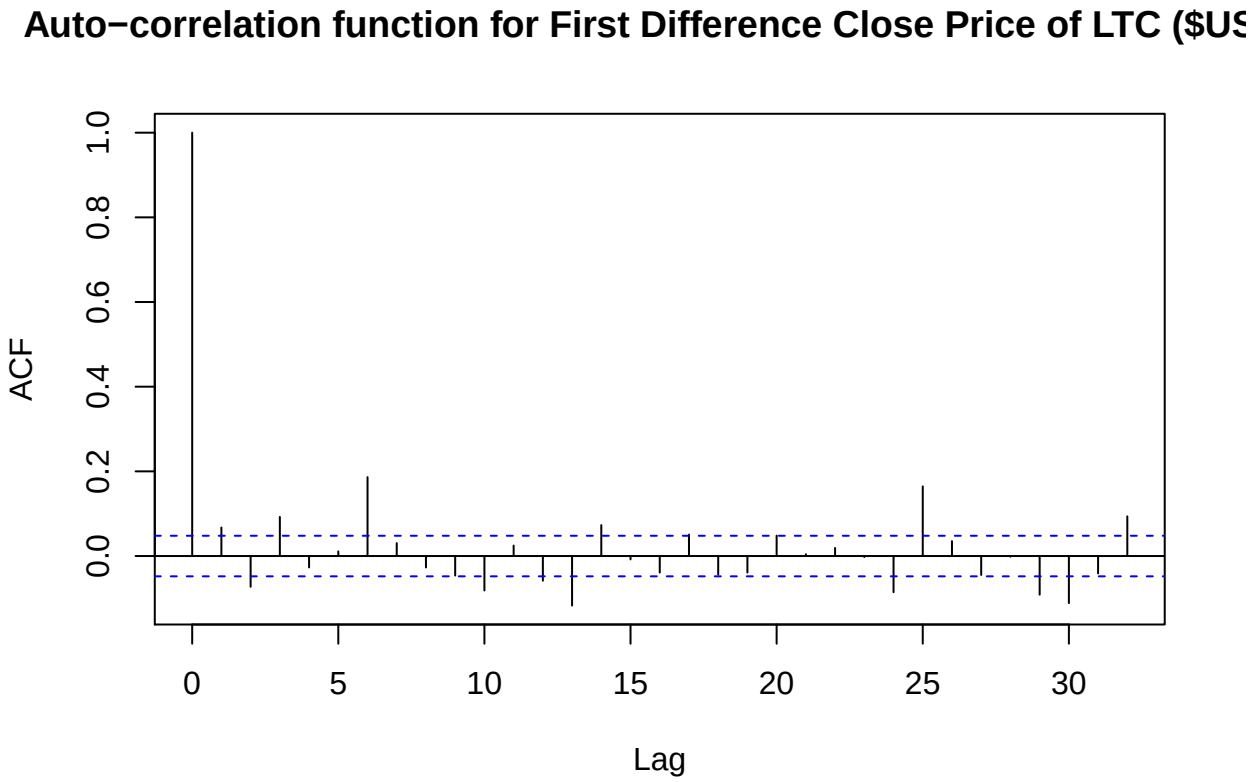
First let's have a look at the ACF (Autocorrelation Function for our data)

```
d_ltc %>%
  pull(close_usd) %>%
  acf(main = "Auto-correlation function for Close Price of LTC ($USD)")
```



We can see there is a high amount of auto-correlation in our data. Let's take the first difference and then investigate the ACF.

```
d_ltc %>%
  mutate(
    close_lag_usd = lag(close_usd),
    first_diff = close_usd - close_lag_usd
  ) %>%
  na.omit() %>%
  pull(first_diff) %>%
  acf(main = "Auto-correlation function for First Difference Close Price of LTC ($USD)")
```



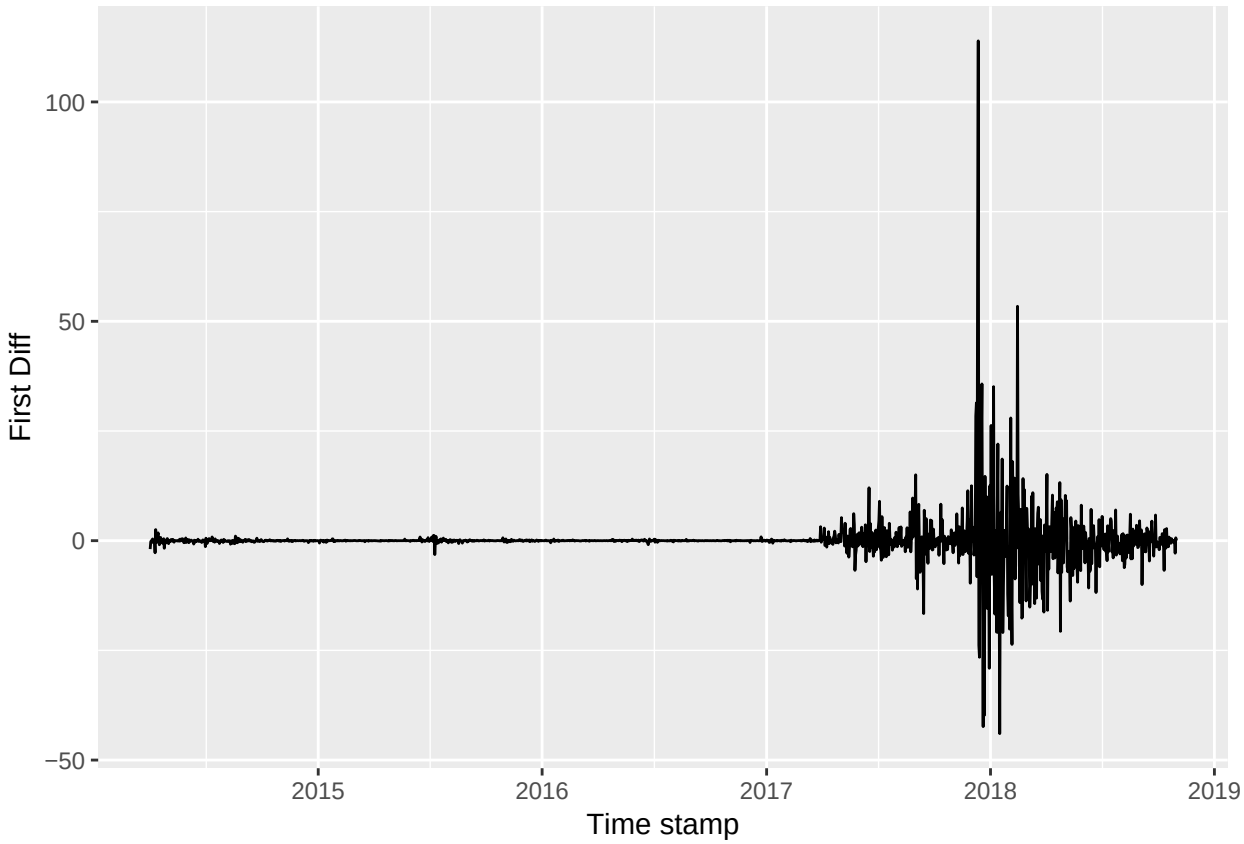
Ok we can don't see much auto-correlation when we difference our results and as such this indicates for our time series $\{x_t\}_{i=1}^{T}$ that x_t and x_{t-1} are correlated.

Let's tackle this by predicting the differenced values rather than the actual closing price. We can modify our data below.

```
d_diff <- d_ltc %>%
  mutate(
    close_lag_usd = lag(close_usd),
    first_diff = close_usd - close_lag_usd
  )
```

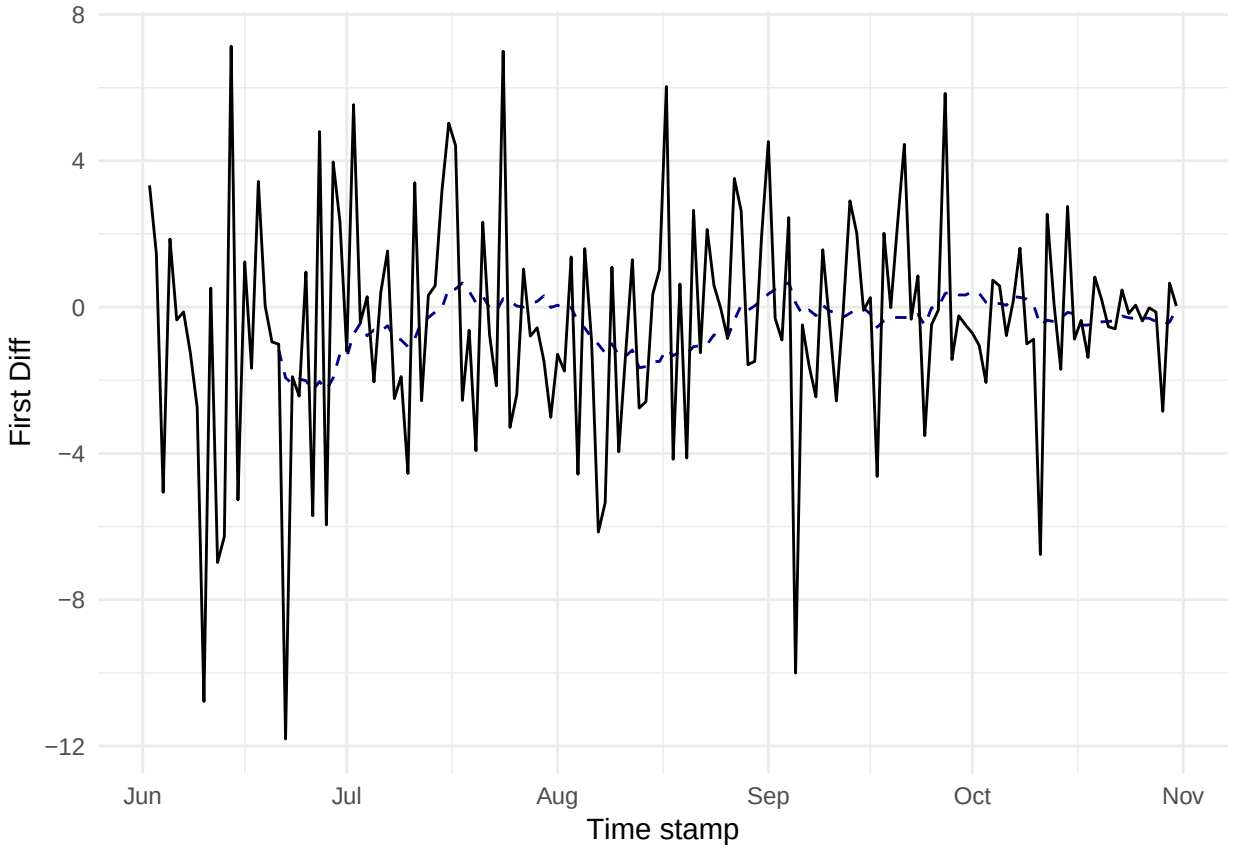
Let's plot our differenced series.

```
d_diff %>%
  ggplot(aes(x = timestamp, y = first_diff)) +
  geom_line()+
  labs(
    x = "Time stamp",
    y = "First Diff"
  )
```



Due to the fact we're looking at such a large time series, the series is clearly not stationary. Let's get rid of all the points except those since June 2018. This makes sense as it is unlikely the price of LTC from last year will effect tomorrows price.

```
d_diff %>%
  filter(timestamp > as.Date("2018-06-01")) %>%
  ggplot(aes(x = timestamp, y = first_diff)) +
  # Plot moving average
  geom_ma(ma_fun = SMA, n = 20) +
  geom_line() +
  labs(
    x = "Time stamp",
    y = "First Diff"
  )+
  theme_minimal()
```

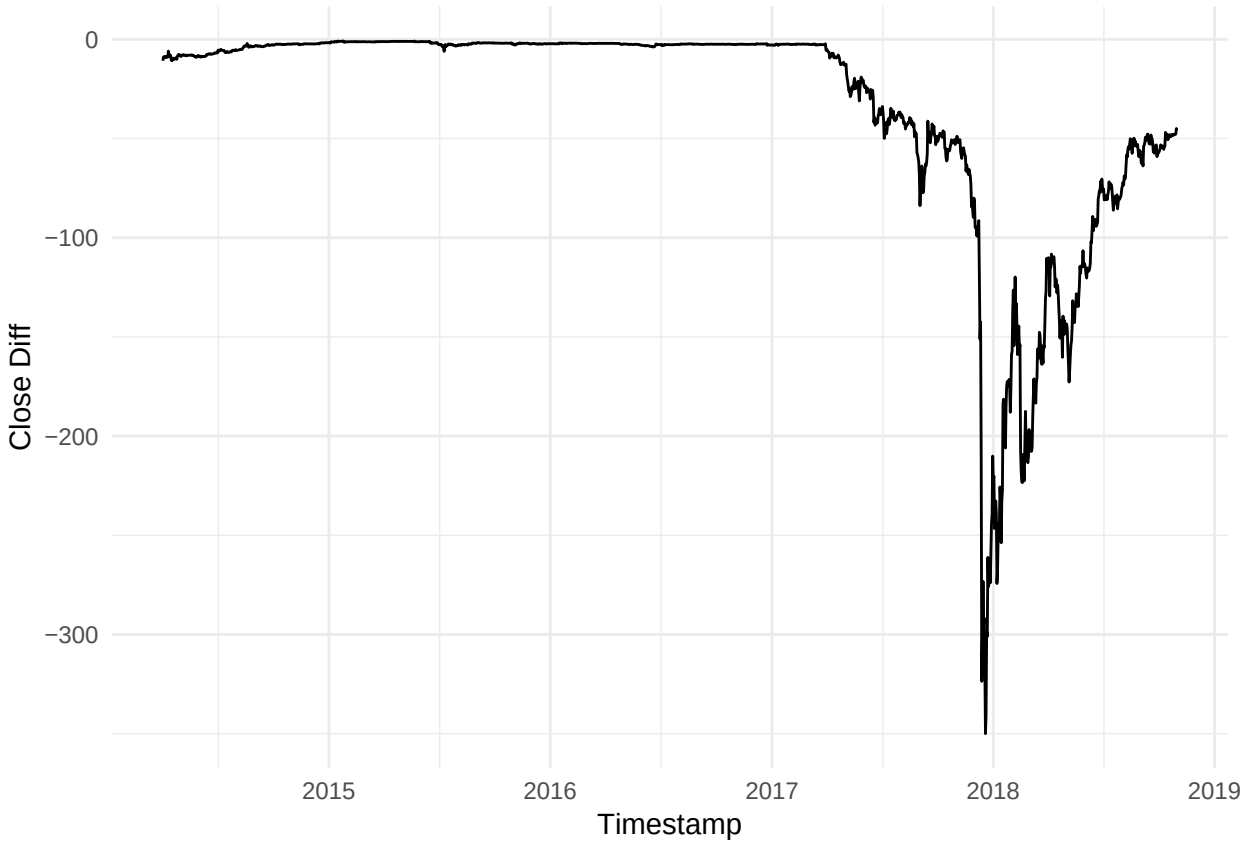


From the plot above, the data looks to have a decreasing variance and a constant mean, and the ACF also looked good. Thus since we violate the homoscedasity assumption.

Thus this indicates, we should log transform our close price, take the difference and then forecast differences.

We can also concur that since each observation depends on the previous one, linear models will not work and we will need to explore, either time series techniques, or non-linear regression models.

```
d_ltc %>%
  mutate(
    log_close = log(close_usd),
    log_close_lag = lag(close_usd),
    log_close_diff = log_close - log_close_lag
    ) %>%
  ggplot(aes(x = timestamp, y = log_close_diff)) +
  geom_line() +
  labs(
    x="Timestamp",
    y="Close Diff"
  )+
  theme_minimal()
```



Final Comments on Approach

In the above analysis we have explored the data from LTC and seen that the data looks to be non-stationary. If we are to predict the price of litecoin a two-pronged approach could be used:

1. Use additional features:
    - Use information such as high, low, open and previous days close, high, low and open price as input features
    - Engineer features from the time series (e.g. Seasonal Components)
    - Combine exogenous time series (e.g. Price of BTC) as useful features or other non-temporal exogenous features
2. Use more sophistacted models than those which are used for univariate time series forecasting
    - Use a non-linear model which doesn't require an independence assumption
    - We could also use a time series model to forecast the next point prediction and use a non-linear model to learn the residual series