Bitcoin Volatility Project

Contents

1 Introduction

1.1 Objective

Cryptocurrency is one of the fastest growing and most disruptive financial systems of the 21st century. A system developed entirely on code and using block-chain technology, Cryptocurrency's unconventional position as a decentralised digital market enables traders to operate outside of the traditional banking system that's subject to regulation, inside information and established players. Indeed, crypto's liberated position has drawn both praise and criticism from commentators, politicians, traders, and spectators alike. U.S senator Elizabeth Warren has branded the new system as a financial 'Wild-West'. Conversely, high-profile technology entrepreneur Elon Musk has validated Cryptocurrency by investing 1.5 billion into Bitcoin (a crypto currency) via his electric-car company Tesla, which has also started accepting the currency as payment for its vehicles – although even Elon Musk's position on Cryptocurrency is consistent with Elizabeth Warrens "Wild-West" branding. Nonetheless, Cryptocurrency's offering has ignited a flurry of opportunistic traders passionate about the currencies innovative structure and excited by its dramatic growth since inception in 2009. At its peak in April 2021, one Bitcoin was worth $63 729.5.

Cryptocurrency is a unique financial system offering traders an exciting return on investment, however, alike any other financial market, where there is potential for growth there is also potential for losses. Cryptocurrency's high-growth potential for return on investment is evenly matched by significant risk of loss. In 2018, the annual growth rate of Bitcoin was -72.6%. In 2019, Bitcoin's growth rate was 87.2%. Crypto-currencies such as Bitcoin, Ethereum, Litecoin and Dogecoin, are characterised by wide-market fluctuation and heavy trading - this is partly due to the system's unconventional structure. Compared to currencies backed by governments, the factors that influence a Bitcoin traders confidence varies from traditional indicators. Although, alike all other currencies, to mitigate the potential of loss due to market unpredictability, crypto traders must evaluate risk.

Financial risk is defined by the uncertainty of return on investment, which is derived by the value of an asset at various time intervals. One measure of evaluating risk is by calculating volatility - the rate at which a commodity is likely to rise or fall at any point in time. Given the barriers to accurately predicting price, traders use statistical calculations based upon historical data, to predict the likely range of fluctuation. This allows a trader to assess the level of associated risk at any-point in time, which enables optimisation of strategy for maximum return. For example, if a model predicts Bitcoin's value to be highly volatility for the next 7 days, one strategy for a risk-averse trader would be to mitigate risk of significant loss by reducing their holdings. Conversely, a trader with a large appetite for risk may maintain or increase their holdings on the chance that if the value rises they will achieve a significant gain.

It is important to note, that typically a high-risk market is a highly volatile market.Predicting the volatility of an asset is an important indication of the associated risk at various time intervals. Indeed, highly volatile assets represent high growth opportunities, they also pose great potential for loss. The goal of this project is to create a model that predicts the future conditional volatility of Bitcoin for the next 30 days. In this case, volatility is considered a statistical measure that indicates the dispersion of daily returns at time t - daily returns is defined as the money made or lost at the end of the day as a result of your Bitcoin holdings.

We will estimate the volatility at time intervals of one day. This will be achieved by calculating the conditional standard deviation at time t from Bitcoins daily returns using the GARCH(1,1) model.

GARCH stands for Generalized AutoRegressive Conditional Heteroskedasticity; it is a statistical model that was created by economics Nobelist Robert F. Engle to help estimate volatility. Using this model, we will then apply different machine learning models (ML) to our predictions to improve the accuracy of the estimated volatility.

To calculate the accuracy of our algorithm, we will use a loss function called the residual mean squared error or RMSE on a test set. The RMSE is defined as follow:

$$RMSE = \sqrt{\frac{1}{N}\sum_{t}(y_t - \hat{y}_t)^2}$$

Where y_t represents the true daily volatility for bitcoin at day t, and $\hat{y}_t$ denotes the predicted daily volatility for bitcoin at day t.

```
#RMSE function
RMSE <- function(true_volatility, predicted_volatility){
  sqrt(mean((true_volatility - predicted_volatility)^2))
}
```

Our aim is to develop an algorithm with the smallest RMSE possible. In order to create this algorithm, we will need to first separate the data into two different sets - one for training and one for testing. Then, we will explore and analyse the data, and develop methods to predict volatility. Finally, we will compare the different methods using the results and conclude which algorithm is most accurate.

1.2 Dataset

This research has been completed using multiple cryptocurrencies' financial datasets, which have been sourced from Yahoo and downloaded to my personal Github. The Cryptocurrencies include Bitcoin, Dogecoin, Ethereum and Litecoin. This code also adds libraries that we will be using to conduct analysis.

```
############################################################
# Create edx set, validation set (final hold-out test set)
############################################################

# Note: this process could take a couple of minutes

if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret",repos = "http://cran.r-project.org")
if(!require(PerformanceAnalytics)) install.packages("PerformanceAnalytics",
                                                    repos ="http://cran.us.r-project.org")
if(!require(xts)) install.packages("xts", repos = "http://cran.us.r-project.org")
if(!require(rugarch)) install.packages("rugarch", repos = "http://cran.us.r-project.org")
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")
if(!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(dplyr)
library(caret)
library(PerformanceAnalytics)
library(xts)
library(lubridate)
library(rugarch)
library(randomForest)

#downloading the cryptos data
#all datasets from
#https://github.com/MatthieuvdSlikke/DataScience_Crytpo_HarvardX-ML/tree/main/DataSets

#bitcoin
dl <- tempfile()
download.file("https://raw.githubusercontent.com/MatthieuvdSlikke/DataScience_Crytpo_HarvardX-ML/main/Da
              dl, method = "curl")
bitcoin <- read.csv(file = dl, header = TRUE, stringsAsFactors = FALSE)
bitcoin <- bitcoin %>% mutate(Date=as.Date(Date)) %>%
  filter(Date >= '2015-01-01') %>%
  filter(Open!=is.na(Open),Volume!=is.na(Volume)) %>%
  filter(Open!='null',Volume!='null') %>%
  mutate(Open= as.numeric(Open), Volume=as.numeric(Volume)) %>%
  select(Date,Open,Volume)

#dogecoin
dl <- tempfile()
download.file("https://raw.githubusercontent.com/MatthieuvdSlikke/DataScience_Crytpo_HarvardX-ML/main/Da
              dl, method = "curl")
doge <- read.csv(file = dl, header = TRUE, stringsAsFactors = FALSE)
doge <- doge %>%
  mutate(Date=as.Date(Date)) %>%
  filter(Date >= '2015-01-01') %>%
```

```
  filter(Open!=is.na(Open),Volume!=is.na(Volume)) %>%
  filter(Open!='null',Volume!='null') %>%
  mutate(Open_dgc= as.numeric(Open), Volume_dgc=as.numeric(Volume)) %>%
  select(Date,Open_dgc,Volume_dgc)

#ethereum
dl <- tempfile()
download.file("https://raw.githubusercontent.com/MatthieuvdSlikke/DataScience_Crytpo_HarvardX-ML/main/Da
              dl, method = "curl")
ethereum <- read.csv(file = dl, header = TRUE, stringsAsFactors = FALSE)
ethereum  <- ethereum %>% mutate(Date=as.Date(Date)) %>%
  filter(Date >= '2015-01-01') %>%
  filter(Open!=is.na(Open),Volume!=is.na(Volume)) %>%
  filter(Open!='null',Volume!='null') %>%
  mutate(Open_eth= as.numeric(Open), Volume_eth=as.numeric(Volume)) %>%
  select(Date,Open_eth,Volume_eth)

#litecoin
dl <- tempfile()
download.file("https://raw.githubusercontent.com/MatthieuvdSlikke/DataScience_Crytpo_HarvardX-ML/main/Da
              dl, method = "curl")
litecoin <- read.csv(file = dl, header = TRUE, stringsAsFactors = FALSE)
litecoin <- litecoin %>% mutate(Date=as.Date(Date)) %>%
  filter(Date >= '2015-01-01') %>%
  filter(Open!=is.na(Open),Volume!=is.na(Volume)) %>%
  filter(Open!='null',Volume!='null') %>%
  mutate(Open_ltc= as.numeric(Open), Volume_ltc=as.numeric(Volume)) %>%
  select(Date,Open_ltc,Volume_ltc)
```

2 Methods and Analysis

2.1 Data Exploration and Cleaning

2.1.1 Overview

Firstly, all data will be joined into one table and then separated into two sets: the test set and the validation set.

To predict the volatility for the next 30days, the test set will be defined as equal to the validation set minus the last 30 days.

For simplicity, the model will only use the open prices to calculate the daily returns and associated volatility; Open prices are defined by how much a crypto is worth in USD at day t.

In the joined table, I have also added a fourth feature: Open_all_coins, which represents the average open price of Dogecoin, Ethereum and Litecoin.

```
#table joined together by Date
crypto <- inner_join(doge, ethereum, by='Date') %>%
  inner_join(.,litecoin, by='Date') %>%
  inner_join(.,bitcoin, by='Date')

#selecting all open prices of coins and take the average price of all the other coins than bitcoin
validation_set <- crypto %>% select(Date,Open,Open_dgc,Open_eth,Open_ltc) %>%
  mutate(Open_all_coins =(Open_dgc+Open_eth+Open_ltc)/3)

#test set
test_set <- validation_set %>% slice_head(n=nrow(validation_set)-30)
```

Here is what the first 6 data points look like:

```
##         Date    Open Open_dgc Open_eth Open_ltc Open_all_coins
## 1 2015-08-07 278.741 0.000168 2.831620  4.06334       2.298376
## 2 2015-08-08 279.742 0.000169 2.793760  4.22099       2.338306
## 3 2015-08-09 261.116 0.000158 0.706136  3.84339       1.516561
## 4 2015-08-10 265.478 0.000162 0.713989  3.90080       1.538317
## 5 2015-08-11 264.342 0.000160 0.708087  3.94874       1.552329
## 6 2015-08-12 270.598 0.000164 1.058750  4.14334       1.734085
```

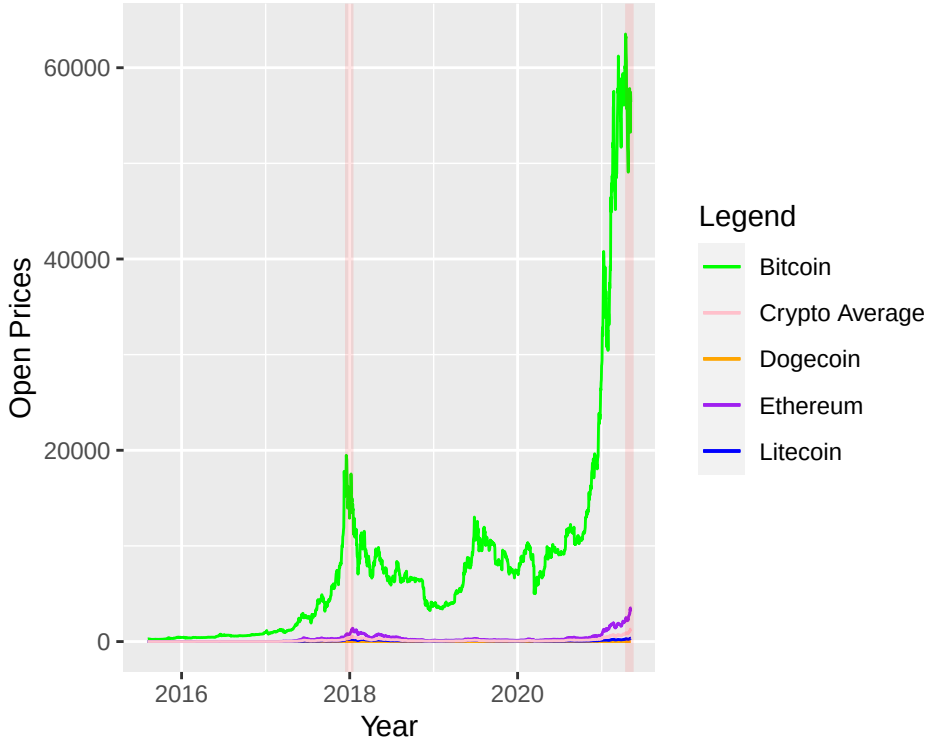
```
dim(test_set)
```

```
## [1] 2097    6
```

There are 2097 days of data organised into 6 columns. Each column is labelled according to its currency, except Bitcoin is labelled as "Open". Each column represents the open price of each coin at date t. The final column is the average price of all coins except Bitcoin.

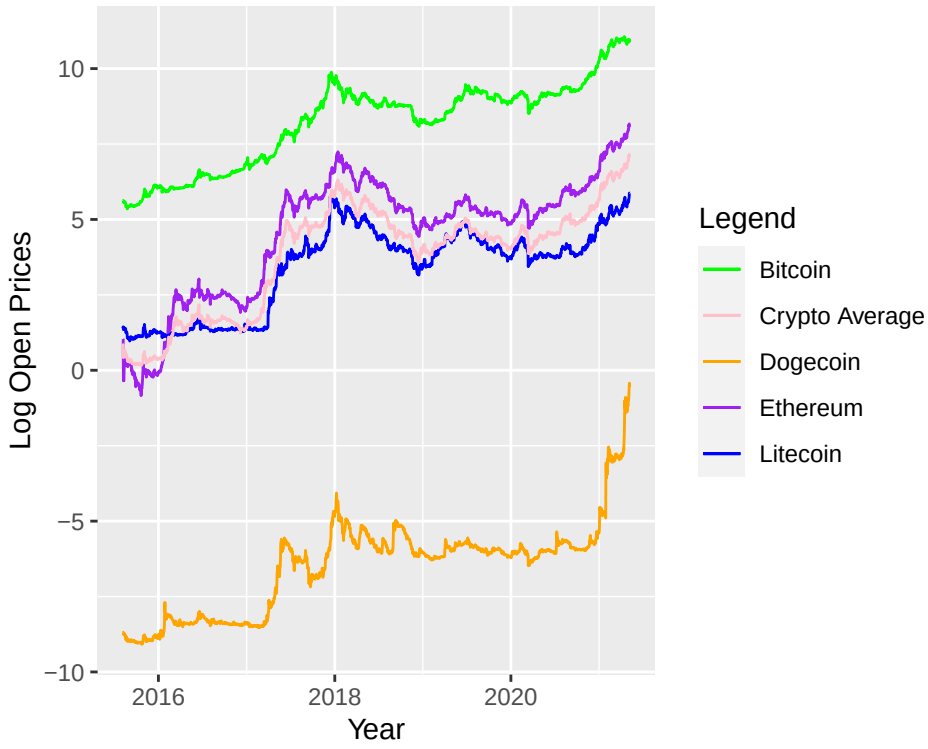
2.1.2 Exploration

The graph below represents the evolution each cryptocurrencies price.



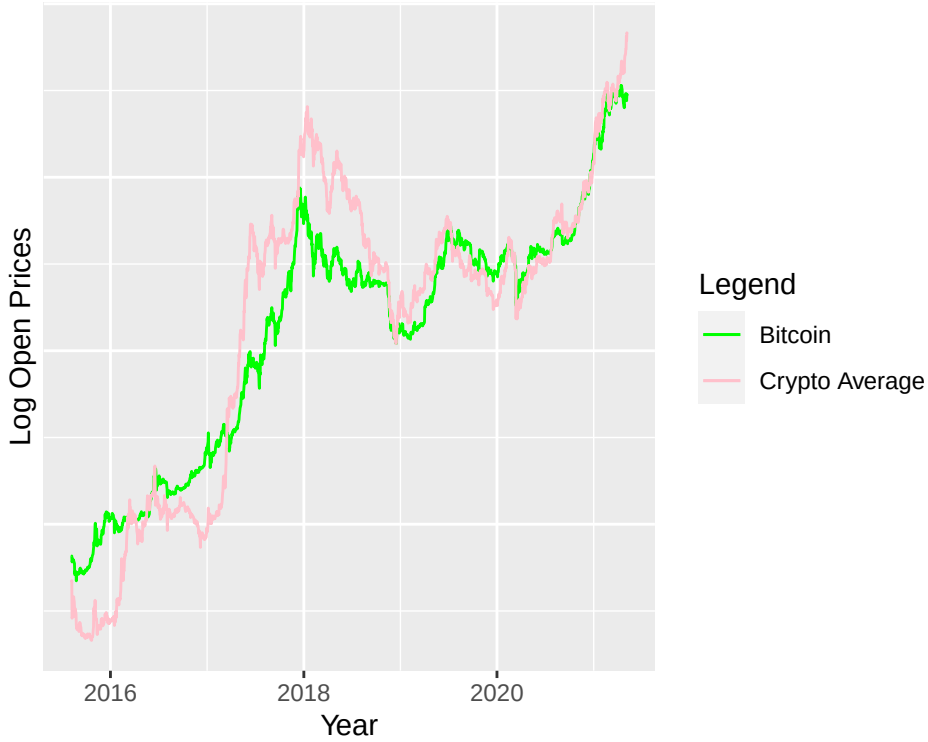
This graph represents the extreme disparity between Bitcoin's price compared to all other cryptocurrencies. Bitcoin is so much more valuable that the evolution of all other Cryptocurrencies seems negligeable. However, upon closer examination, in early 2018 and mid 2021 Bitcoin and the other cryptocurrencies seem to peak at the same time. This indicates a possible correlation.

In the next steps, we will analyse the evolution of the currencies by plotting their log transformations. This will allow us to visualise the evolution of cryptocurrencies proportionally.



By plotting the log transformations of each coin during the above specified timeframe, it can be observed that Litecoin, Dogecoin and Ethereum follow approximately the same trend as Bitcoin.

By overlaying the two time-series below - Bitcoin and Crypto Average – we can observe that they follow a similar evolution – sharing the same peaks.



This is reflected below in the table as the numbers represent a strong correlation between Bitcoins and the

other cryptocurrencies'.

method	correlations
BTC-DGC	0.6110214
BTC-LTC	0.7540934
BTC-ALL	0.9171386
BTC-ETH	0.9233657

2.1.3 Data preparation and cleaning

In order to apply the GARCH(1,1) model, the average daily returns R_t at day t must first be calculated:

$$R_t = \frac{P_t - P_{t-1}}{P_t}$$

where P_t is the price P at day t, and P_{t-1} the price P from at day $t-1$ (ie: or the previous day).

We will use a function to calculate the average daily returns CalculateReturns() from the PerformanceAnalytics package, but in order to use that function the dataframes must be converted into timeseries.

```
#transform into a time series
validation_xts <- as.xts(validation_set[, -1], order.by = validation_set$Date, dateFormat="POSIXct")
test_set_xts <- as.xts(test_set[, -1], order.by = test_set$Date, dateFormat="POSIXct")

#calculate all the returns
Returns_validation <- CalculateReturns(validation_xts)
Return_test <- CalculateReturns(test_set_xts)

#remove the first line as the first entry does not exist.
Returns_validation <- Returns_validation[-1,]
Return_test <- Return_test[-1,]
head(Return_test)
```

```
##                    Open     Open_dgc     Open_eth    Open_ltc Open_all_coins
## 2015-08-08  0.003591172  0.005952381 -0.013370438  0.03879813    0.017373282
## 2015-08-09 -0.066582804 -0.065088757 -0.747245290 -0.08945769   -0.351427436
## 2015-08-10  0.016705219  0.025316456  0.011121087  0.01493733    0.014345392
## 2015-08-11 -0.004279025 -0.012345679 -0.008266234  0.01228979    0.009108656
## 2015-08-12  0.023666242  0.025000000  0.495225869  0.04928154    0.117085790
## 2015-08-13 -0.016315635 -0.024390244  0.154417946 -0.04128795   -0.001457830
```

Next, the data is then split into time series according to their currency, which will help us later in applying our GARCH (1,1) model.

```
#seperate data
bitcoin_xts <- Returns_validation$Open
dogecoin_xts <- Returns_validation$Open_dgc
ethereum_xts <- Returns_validation$Open_eth
litecoin_xts <- Returns_validation$Open_ltc
all_coins_xts <- Returns_validation$Open_all_coins

bitcoin_test_xts <- Return_test$Open
dogecoin_test_xts <- Return_test$Open_dgc
ethereum_test_xts <- Return_test$Open_eth
litecoin_test_xts <- Return_test$Open_ltc
all_coins_test_xts <- Return_test$Open_all_coins
```

2.2 Models

As stated in the introduction, the accuracy of the models will be calculated using the RMSE function and by graphing their results. We can interpret the RMSE similarly to a standard deviation: which is the typical error made when predicting volatility.

In the following section, the GARCH(1,1) machine learning model will be applied to predict the volatility of Bitcoin for the next 30 days. The resulting predictions will be refined using linear regression and random forest.

2.2.1 GARCH(1,1)

As previously defined GARCH(1,1) model is a a machine learning algorithm that helps us predict the future volatility. It assumes that $R_t = \mu + \epsilon_t$ for t day where ϵ_t is independent errors sampled from the same distribution centered at 0. We also assume that $\epsilon_t \sim N(0, \sigma_t^2)$ meaning that returns are normally distributed.

The GARCH(1,1) variance σ_t^2 at day t is calculated as followed:

$$\sigma_t^2 = \omega + \alpha\epsilon_{t-1} + \beta\sigma_{t-1}^2$$

ω,α β must all be positive ie: > 0, and $\alpha + \beta < 1$ must be satisfied. This ensures that $\sigma_t^2 > 0$ at all times. To find the conditional volatility per day, which is the square root of σ_t^2, we need to estimate four parameters μ, ω, α and β by maximum likelihood.

Although this model only allow us to calculate the variance at time t, we can repeat the process as many times as we we like as long as we use the previously calculated estimate. Since $\alpha + \beta < 1$, the garch variance σ_t^2 is mean reverting, meaning that it will return to it's long run variance equal to $\frac{\omega}{(1-\alpha-\beta)}$.

Fortunately for us, the rugarch package written by Alexios Ghalanos allow us to simply apply the GARCH(1,1) model. Firstly, we need to define which GARCH model we will use.

```
#let's define our GARCH model settings
garchspec <- ugarchspec(mean.model=list(armaOrder=c(0,0)),
                        variance.model = list(model="sGARCH",garchOrder=c(1,1)),
                        distribution.model = "norm")
garchspec
```

```
##
## *----------------------------------*
## *       GARCH Model Spec           *
## *----------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model      : sGARCH(1,1)
## Variance Targeting   : FALSE
##
## Conditional Mean Dynamics
## -----------------------------------
## Mean Model       : ARFIMA(0,0,0)
## Include Mean     : TRUE
## GARCH-in-Mean        : FALSE
##
## Conditional Distribution
## -----------------------------------
## Distribution :  norm
## Includes Skew    :  FALSE
## Includes Shape   :  FALSE
```

```
## Includes Lambda  :  FALSE
```

Now, we can estimate the GARCH(1,1) model to all of our cryptocurrencies.

```
#Apply the GARCH model to our cryptocurrencies'daily returns
garchfit_bitcoin_test <- ugarchfit(data=bitcoin_test_xts,spec=garchspec)
garchfit_dogecoin_test <- ugarchfit(data=dogecoin_test_xts,spec=garchspec)
garchfit_ethereum_test <- ugarchfit(data=ethereum_test_xts,spec=garchspec)
garchfit_litecoin_test <- ugarchfit(data=litecoin_test_xts,spec=garchspec)
garchfit_all_coins_test <- ugarchfit(data=all_coins_test_xts,spec=garchspec)
```

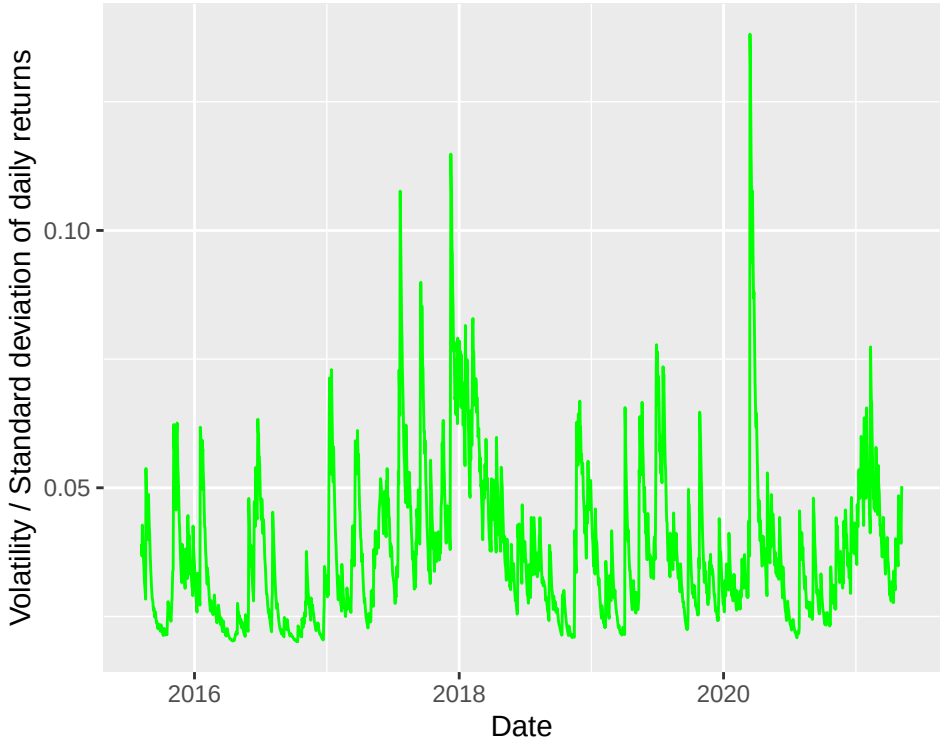
Our GARCH(1,1) model predicts our four parameters as follow:

```
##           mu        omega       alpha1        beta1
## 2.807516e-03 6.117748e-05 1.315522e-01 8.414913e-01
```

Let's now retrieve conditional volatility from our calculated garch model:

```
garchvol_bitcoin_test <- sigma(garchfit_bitcoin_test)
garchvol_dogecoin_test <- sigma(garchfit_dogecoin_test)
garchvol_ethereum_test <- sigma(garchfit_ethereum_test)
garchvol_litecoin_test <- sigma(garchfit_litecoin_test)
garchvol_all_coins_test <- sigma(garchfit_all_coins_test)
volatility_test <- data.frame(Date=index(garchvol_bitcoin_test),
                              vol_bitcoin=coredata(garchvol_bitcoin_test),
                              vol_dogecoin= coredata(garchvol_dogecoin_test),
                              vol_ethereum= coredata(garchvol_ethereum_test),
                              vol_litecoin= coredata(garchvol_litecoin_test),
                              vol_all_coins= coredata(garchvol_all_coins_test))
```

Our volatility for Bitcoin for the testing period using using the GARCH(1,1) looks like this:



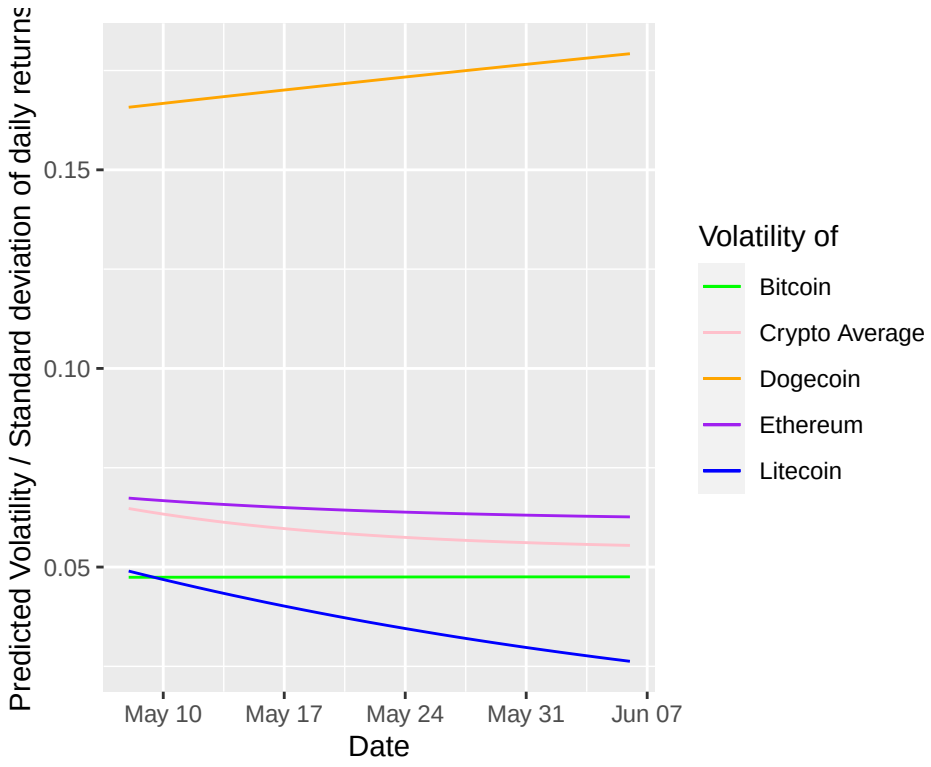
As we have observed a moderate correlation between the volatility of Bitcoin and the other cryptocurrencies' volatility, we can perform a linear regression to our model. Our formula to estimate the volatility of Bitcoin

method	correlations
VOL BTC-DGC	0.2271322
VOL BTC-ETH	0.3543413
VOL BTC-LTC	0.6262597
VOL BTC-ALL	0.6221566

Now let's predict the volatility for the next 30 days. The ugarchforecast() function allow us to predict the volatility for the next t days.

```
# forecast for the next 30 days based
garchforecast_bitcoin_test <- ugarchforecast(fitORspec = garchfit_bitcoin_test,n.ahead = 30 )
garchforecast_dogecoin_test <- ugarchforecast(fitORspec = garchfit_dogecoin_test,n.ahead = 30 )
garchforecast_ethereum_test <- ugarchforecast(fitORspec = garchfit_ethereum_test,n.ahead = 30 )
garchforecast_litecoin_test <- ugarchforecast(fitORspec = garchfit_litecoin_test,n.ahead = 30 )
garchforecast_all_coins_test <- ugarchforecast(fitORspec = garchfit_all_coins_test,n.ahead = 30 )

#retrieving the volatility
forecast_bitcoin <- sigma(garchforecast_bitcoin_test)
forecast_dogecoin <- sigma(garchforecast_dogecoin_test)
forecast_ethereum <- sigma(garchforecast_ethereum_test)
forecast_litecoin <- sigma(garchforecast_litecoin_test)
forecast_all_coins <- sigma(garchforecast_all_coins_test)

# change format from timeseries to dataframe
volatility_bitcoin_forecast <- data.frame(index=index(forecast_bitcoin),
                                          coredata(forecast_bitcoin)) %>%
  mutate(vol_bitcoin=X2021.05.07) %>% select(index,vol_bitcoin)
volatility_dogecoin_forecast <- data.frame(index=index(forecast_dogecoin),
                                           coredata(forecast_dogecoin)) %>%
  mutate(vol_dogecoin=X2021.05.07) %>% select(index,vol_dogecoin)
volatility_ethereum_forecast <- data.frame(index=index(forecast_ethereum),
                                           coredata(forecast_ethereum)) %>%
  mutate(vol_ethereum=X2021.05.07) %>% select(index,vol_ethereum)
volatility_litecoin_forecast <- data.frame(index=index(forecast_litecoin),
                                           coredata(forecast_litecoin)) %>%
  mutate(vol_litecoin=X2021.05.07) %>% select(index,vol_litecoin)
volatility_all_coins_forecast <- data.frame(index=index(forecast_all_coins),
                                            coredata(forecast_all_coins)) %>%
  mutate(vol_all_coins=X2021.05.07) %>% select(index,vol_all_coins)

#getting the correct dates and joining all the predicted volatilities together
tail_dates <- validation_set %>% slice_tail(n=30) %>% select(Date)

#store all predicted volatilities in a common dataframe
volatility_30days_forcecast <- data.frame(Date=tail_dates$Date,
                                          vol_bitcoin=volatility_bitcoin_forecast$vol_bitcoin,
                                          vol_dogecoin=volatility_dogecoin_forecast$vol_dogecoin,
                                          vol_ethereum=volatility_ethereum_forecast$vol_ethereum,
                                          vol_litecoin=volatility_litecoin_forecast$vol_litecoin,
                                          vol_all_coins=volatility_all_coins_forecast$vol_all_coins)
```

Here is a plot of our predictions. All predictions seem to flatten in time, which is expected because our GARCH(1,1) model is mean reverting, as we have mentioned previously, their long run variance is equal to $\frac{\omega}{(1-\alpha-\beta)}$.



2.2.2 Linear Regression

As we have observed a moderate correlation between the volatility of Bitcoin and the other cryptocurrencies' volatility, we can perform a linear regression to our model. Our formula to estimate the volatility of Bitcoin becomes:

$$\sigma_t = \sqrt{\omega + \alpha\epsilon_{t-1} + \beta\sigma_{t-1}^2} + \beta_0 + \beta_1\sigma_{l,t} + \beta_2\sigma_{a,t} + \beta_3\sigma_{e,t} + \beta_4\sigma_{d,t}$$

where $\sigma_{l,t}^2$ is the volatility of Litecoin at day t, $\sigma_{a,t}^2$ is the volatility of the average price all the cryptocurrencies at day t, $\sigma_{e,t}^2$ is the volatility of Ethereum at day t and $\sigma_{d,t}^2$ is the volatility of Dogecoin at day t.

```
#fit the data
#linear regression
fit <- volatility_test %>%
  lm(vol_bitcoin ~ vol_litecoin + vol_all_coins + vol_ethereum + vol_dogecoin, data = .)
```

We find the following coefficients for our model:

```
##   (Intercept)  vol_litecoin vol_all_coins  vol_ethereum  vol_dogecoin
##   0.007139871   0.261605841   0.573287259 -0.154589527   0.006805963
```

```
#predict using forecast data
predict <- volatility_30days_forcecast %>%
  mutate(vol_bitcoin_hat = predict(fit, newdata = .))
```

2.2.3 Random Forest

For our final prediction model, we will use random forest to estimate Bitcoin's volatility using regression.

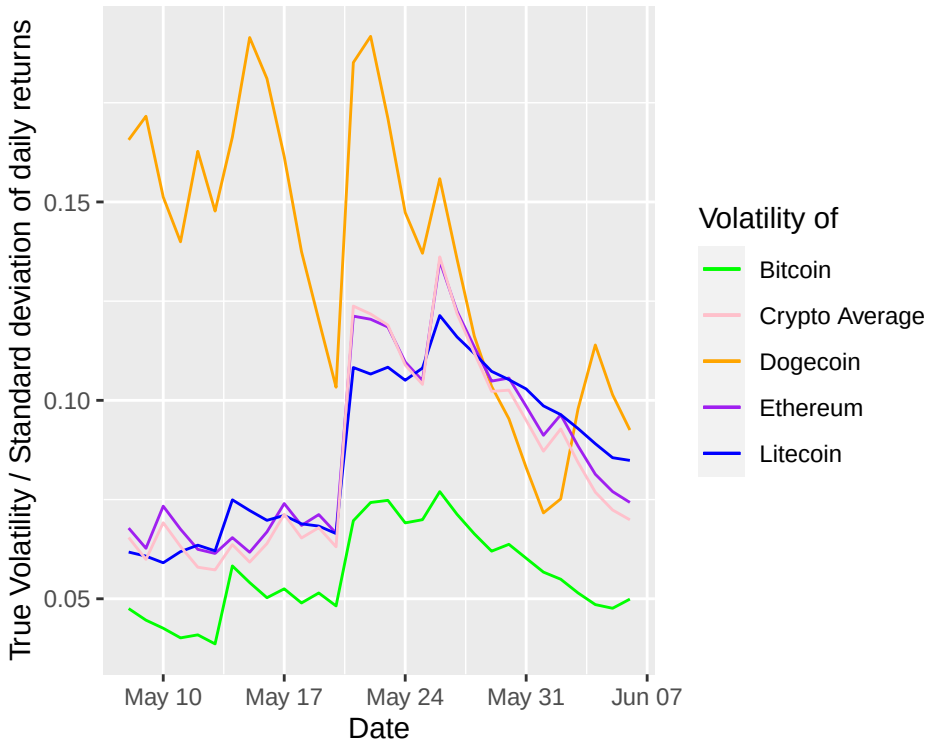
```
#random forest
fit_rf <- randomForest(vol_bitcoin ~ vol_litecoin +
                         vol_all_coins +
                         vol_ethereum +
                         vol_dogecoin,
                       data = volatility_test)

#predict using forecast data
predict_rf <- volatility_30days_forcecast %>%
  mutate(vol_bitcoin_hat = predict(fit_rf, newdata = .))
```

3 Results

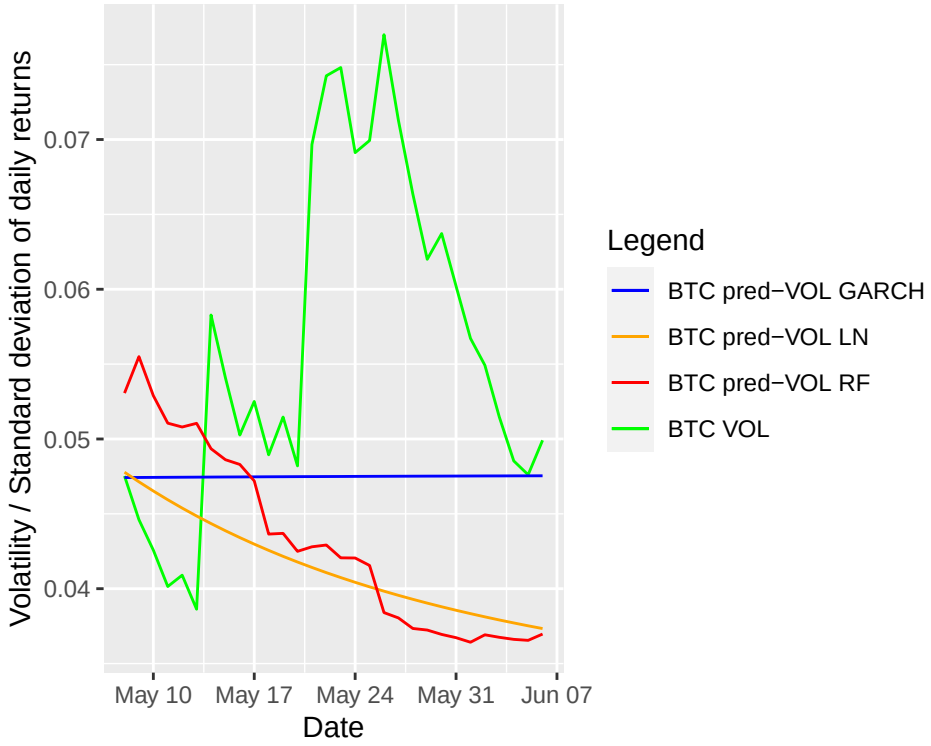
To analyse which algorithm most accurately predicts the Bitcoin volatility for the last 30 days, we must first calculate the true volatility for that period.

```
#applying the GARCH(1,1) model to the validation data
garchfit_bitcoin <- ugarchfit(data=bitcoin_xts,spec=garchspec)
garchfit_dogecoin <- ugarchfit(data=dogecoin_xts,spec=garchspec)
garchfit_ethereum <- ugarchfit(data=ethereum_xts,spec=garchspec)
garchfit_litecoin <- ugarchfit(data=litecoin_xts,spec=garchspec)
garchfit_all_coins <- ugarchfit(data=all_coins_xts,spec=garchspec)

#data volatility validation
garchvol_bitcoin <- sigma(garchfit_bitcoin)
garchvol_dogecoin <- sigma(garchfit_dogecoin)
garchvol_ethereum <- sigma(garchfit_ethereum)
garchvol_litecoin <- sigma(garchfit_litecoin)
garchvol_all_coins <- sigma(garchfit_all_coins)

#volatility in one table validation
volatility <- data.frame(Date=index(garchvol_bitcoin), vol_bitcoin=coredata(garchvol_bitcoin),
                         vol_dogecoin= coredata(garchvol_dogecoin),
                         vol_ethereum= coredata(garchvol_ethereum),
                         vol_litecoin= coredata(garchvol_litecoin),
                         vol_all_coins= coredata(garchvol_all_coins))

#retrieving the data for the last 30 days
validation_volatility <- volatility %>% slice_tail(n=30)
```

Below is a plot of the true volatilities. The true volatilities are much less linear than our GARCH(1,1) forecast predictions:



The below graph plots the true volatility of Bitcoin against our predicted volatilities for Bitcoin using our different models:



It can be observed that overall, the GARCH(1,1) predicted well for the last 30 days. However, the linear regression model was able to predict the volatility trend more accurately within the first 5 days. The random forest model is the worst performing model, even though it tends to follow the same trend as the linear regression model. These observations are confirmed in the following RMSE table:

method	RMSE
GARCH(1,1) model	0.0140564
Linear Regression + GARCH(1,1) model	0.0195019
Random Forest + GARCH(1,1) model	0.0199215

4 Conclusion

Although the GARCH(1,1) model performed with high accuracy – satisfying the objective of predicting the 30-day volatility of Bitcoin to an RMSE of 0.0140564 – the findings are simply an estimate on market fluctuation. The work demonstrated throughout this project does not anticipate whether a price would rise or fall, rather it enables a trader to strategise with a greater understanding of the associated risk. As the above results have only been calculated for the last 30 days on this test set, it is recommended that a trader perform a k cross validation calculation to improve the accuracy; this has not been done here. However, Linear regression should not be completely disregarded as it has demonstrated that it follows the same trend in the short term and in the first 5 days. K cross validation would enable us to confirm that.

5 Bibiography

Investopedia. 2021. Volatility. [online] Available at: https://www.investopedia.com/terms/v/volatility.asp [Accessed 23 June 2021].

Investopedia. 2021. Return. [online] Available at: https://www.investopedia.com/terms/r/return.asp [Accessed 23 June 2021].

Dr. Kris Boudt. Data Camp. 2021 Garch Models in R [online] Available at: https://campus.datacamp.com/courses/garch-models-in-r/ [Accessed 23 June 2021].

Engle, R., 2021. 8. [online] Stern.nyu.edu. Available at: https://www.stern.nyu.edu/rengle/GARCH101.PDF [Accessed 30 June 2021].

Elizabeth Warren story: Graig Graziosi,Thursday 10 June 2021 16:50 https://www.independent.co.uk/news/world/americas/us-politics/crypto-wild-west-elizabeth-warren-b1863530.html

Elon Musk Bitcoin: Steve Kovach, PUBLISHED MON, FEB 8 20217:48 AM ESTUPDATED MON, FEB 8 20211:43 PM EST https://www.cnbc.com/2021/02/08/tesla-buys-1point5-billion-in-bitcoin.html

Risk definiton: https://www.investor.gov/introduction-investing/investing-basics/what-risk?fbclid=IwAR2fEzNh4I3rx32gzZ9JMmr5ETV1Uk6VwZN4Gy9ruzp3tb7QIoP3vObl6E4

```
##
## When using rugarch in publications, please cite:
##
## To cite the rugarch package, please use:
##
##   Alexios Ghalanos (2020). rugarch: Univariate GARCH models. R package
##   version 1.4-4.
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {rugarch: Univariate GARCH models.},
##     author = {Alexios Ghalanos},
##     year = {2020},
##     note = {R package version 1.4-4.},
##   }
```

More information on Bitcoin volatility: https://www.buybitcoinworldwide.com/volatility-index/