Ethereum Blockchain data analysis

Rajkumar Panneer Selvam

December 2, 2018

Abstract

Blockchain has recently gained momentum. As one of its first implementations, bitcoin as a cryptocurrency has gained a lot of attention. Together with Ethereum, blockchain implementation with focus on smart contracts, they represent the very core of modern cryptocurrency development. This project is focused on some data analysis in Ethereum and its price datasets like finding discrete distribution type of how many times a user buys/sells a token, extracting features on the ethereum transaction dataset and finding its correlation with price dataset, creating a multiple linear regression model to compute the price on day t by extracting some features on day t-1 from the datasets.

1 Introduction

Ethereum: Ethereum is a decentralized network of computers with two basic functions.They are: blockchain that can record transactions, and a virtual machine that can produce smart contracts.Because of these two functions,Ethereum is able to support decentralized applications (DApps). These DApps are built on the existing Ethereum blockchain, piggybacking off of its underlying technology. In return, Ethereum charges developers for the computing power in their network, which can only be paid in Ether, the only inter-platform currency.

ERC20 tokens: ERC-20 tokens are tokens designed and used solely on the Ethereum platform. They follow a list of standards so that they can be shared, exchanged for other tokens,or transferred to a crypto-wallet.The Ethereum community created these standards with three optional rules, and six mandatory. Optional: Token Name,Symbol,Decimal (up to 18). Mandatory: totalSupply, balanceOf, transfer, transferFrom, approve, allowance.

2 VeChain Ethereum token

The primary ethereum token I have used in this project is **VeChain token**.**VeChain Token (VET)** is the Smart Money or Smart Value in VeChainThor Ecosystem which is

programmable and executable in the smart contracts to carry on value transferring along with commercial activities running on the **VeChainThor Blockchain**. Besides that, VeChain Token (VET) can be discovered as a key element to build up the links among dots in the ecosystem.

The VeChainThor Blockchain is the platform to carry out this future ecosystem with robust blockchain core infrastructure, matching infrastructure services, proper governance and economic design, growing community and business engagement.

VeChain Token (VET) and **VeChainThor Energy a.k.a VeThor (VTHO)**, carrying the value transfers and executing transactions on the VeChainThor Blockchain network.

The vision of VeChain and the VeChainThor Blockchain is to build a trust-free and distributed business ecosystem platform to enable transparent information flow, efficient collaboration, and high-speed value transfers.

3 Dataset explanation and preprocessing

3.1 Ethereum transaction dataset

The data files contain two primary files: VeChain token network edge file, and VeChain price file.

Example of token edge data reference: see Table 1.

The row in the data explains a token has been sold by fromNodeID to toNodeId with tokenAmount value at unixTime date.

fromNodeID	toNodeID	tokenAmount	unixTime
9783305	5	1.438288e+21	1524611290
6	2924030	2.000000e+19	1524611320
82	39989	1.414000e+21	1524611320

Table 1: Ethereum transaction and price data

Example of price data reference: see Table 2.

The price data is taken from https://coinmarketcap.com/. Open and close are the prices of the specific token at the given date.Volume and MarketCap gives total bought/sold tokens and market valuation at the date.

Open	High	Low	Close	Volume	Market.Cap	Date
0.261962	0.265923	0.238242	0.238242	2,133,510	72,606,000	2017-11-23
0.237601	0.260191	0.228138	0.258151	1,884,340	65,854,200	2017-11-24
0.256155	0.262847	0.246314	0.256218	1,416,020	70,996,600	2017-11-25

Table 2: Ethereum price data

3.2 Preprocessing dataset

Finding and Removing Outliers:. We are finding the outlier amounts which are bigger than the total amount of the token in the network token edge file. For the vechain token, the maximum value possible is 86712634466 $\times(10^{18})$. This value is taken from **https://coinmarketcap.com/**. Hence, I used this value to find out the outliers. Total number of outliers is 4. Number of users involved int the transactions are 6.

see Table 3for outliers transactions.

fromNodeID	toNodeID	tokenAmount	unixTime
7960793	7960794	8.000100e+38	1525476070
7960793	7960794	8.000000e+36	1525476550
5442774	913046	1.157921e+77	1517552643

Table 3: Ethereum outlier transaction data

I have removed these transactions from the network token file and final dataset for analysis is obtained.

4 Data analysis section

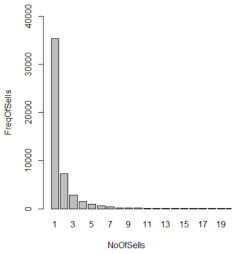
4.1 Find Sellers distribution

Here, I am trying to find the sells distribution from the preprocessed dataset. Hence, I am using fromNodeID column in the token network edge file to find the sells distribution.

see Table 4 for Sellers dataset from transaction data.

NoOfSells	FrequencyOfSells
1	35384
2	7271
3	2855
4	1559
5	936
6	594

Table 4: Sample Sellers distribution dataset derived from VeChain token network edge file.



(a) Seller distribution.

```
sell_fit_exp = fitdist(sell_count_df$FreqOfSells,"exp",method="mle"
sell_fit_exp

## Fitting of the distribution ' exp ' by maximum likelihood
## Parameters:
##          estimate   Std. Error
## rate 0.002554203 0.0001806976

Rate_exp = 0.002554203
Mean_exp = 1/Rate_exp
message("Mean from Exponential is ", Mean_exp)

## Mean from Exponential is 391.511559574552
```

(b) Seller exponential result.

```
sell_fit_pois = fitdist(sell_count_df$FreqOfSells,"pois",method="mle")
sell_fit_pois

## Fitting of the distribution ' pois ' by maximum likelihood
## Parameters:
##        estimate Std. Error
## lambda 391.5116   1.742099
```

(c) Seller poisson result.

Figure 1: Seller results.

In order to find the corresponding distribution, I am trying to fit the sellers distribution to Poisson and exponential, since the sellers distribution resembles to fit those distributions.**Mean of the sells distribution is 391.511627906977**.

I have used R library fitdistrplus to fit the sells distribution to Poisson and Exponential.**see figure 1b and 4.1** for results.

From the above results, we can derive the mean for both Poisson and exponential. Poisson mean is Lambda which results in 391.5116

Exponential mean is 1/Rate which results in 391.511559574552.

Mean of the actual Sells distribution is 391.511627906977.

Result: The mean of both exponential and Poisson matches the mean of sells distribution. I would result to exponential distribution.

4.2 Find Buyers distribution

Here, I am trying to find the buys distribution from the preprocessed dataset. Hence, I am using toNodeID column in the token network edge file to find the buys distribution.

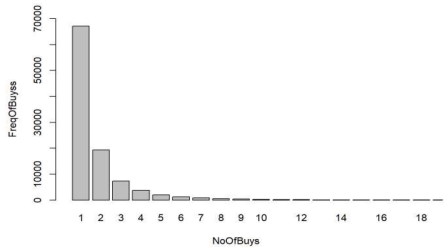
see Table 5 for buyers dataset from transaction data.

NoOfBuys	FrequencyOfBuys
1	67069
2	19346
3	7337
4	3822
5	2051
6	1274

Table 5: Sample Buyers distribution dataset derived from VeChain token network edge file.

In order to find the corresponding distribution, I am trying to fit the buys distribution to Poisson and exponential, since the buys distribution resembles to fit those distributions. Mean of the buys distribution is 777.298507462687

I have used R library fitdistrplus to fit the buys distribution to Poisson and Exponential. Below is Exponential results



(a) Buyers distribution.

```
#library(fitdistrplus)

buy_fit_exp = fitdist(buy_count_df$FreqOfBuys,"exp",method="mme")

buy_fit_exp
```

```
## Fitting of the distribution ' exp ' by matching moments
## Parameters:
##          estimate
## rate 0.001286507
```

(b) Buyers exponential result.

```
buy_fit_pois = fitdist(buy_count_df$FreqOfBuys,"pois",method="mme")

buy_fit_pois
```

```
## Fitting of the distribution ' pois ' by matching moments
## Parameters:
##        estimate
## lambda 777.2985
```

(c) Buyers poisson result.

Figure 2: Buyers results.

From the above 2 results, we can derive the mean for both Poisson and exponential.

Poisson mean is Lambda which results in 777.2985

Exponential mean is 1/Rate which results in 777.29853005075.

Mean of the actual Buys distribution is 777.298507462687.

Result: The mean of both exponential and Poisson matches the mean of buys distribution. I would result to exponential distribution.

4.3 Layer creation and finding correlation

Preprocessing steps: I need to change unix time to Dateformat in VeChain tokenData since the price data has Dateformat type (not in unix). Sample VeChain token data as below table 6.

fromNodeID	toNodeID	tokenAmount	datevalue
9783305	5	1.438288e+21	2018-04-24
6	2924030	2.000000e+19	2018-04-24
82	39989	1.414000e+21	2018-04-24

Table 6: Ethereum transaction data

Creating layers of transactions:

Here,layers are created based on the increasing order of transaction amounts and I choose 15 layers by ensuring at least 17250 transactions per layer. I used 15 layers since the correlation between token amount vs price decreases with increasing amount. 15 layers showed better result in the rate of decrease. First, we find the frequency of the transaction amounts in the tokenData and sort based on the transaction amount. Then we split the tokenData by ensuring atleast 17250 transactions per layer which yields 15 layers in total.

The table 9 shows the layers and theis corresponding minimum and maximum transaction amounts in that layer, also the number of transaction in that particular layer.

The feature we are interested is number of transactions in each layer for a given date. Hence, we are finding the frequency of transactions for a given date in each layer to find its correlation between the price data.

DateValue	number of transactions
2017-08-18	87
2017-08-19	100
2017-08-20	12
2017-08-21	87

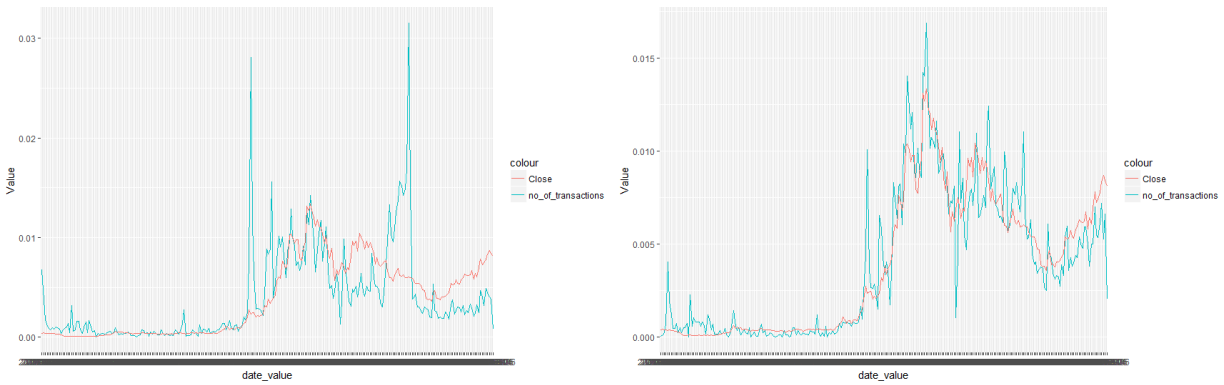
Table 7: Ethereum transaction data

Layer	Min transaction amount	Max transaction amount	No of transactions
1	1	1.07e+19	17254
2	1.07322e+19	4.601684525e+19	17250
3	4.6023e+19	9.3926e+19	17250
4	9.3949e+19	1.52846e+20	17254
5	1.5284600001e+20	2.68019e+20	17250
6	2.68084e+20	4.5e+20	17376
7	4.50013e+20	6.3570204e+20	17251
8	6.35788e+20	9.949e+20	17250
9	9.94944e+20	1.333851e+21	17251
10	1.333863e+21	2e+21	18091
11	2.00000000001e+21	3e+21	17795
12	3.00000168e+21	4.8099561e+21	17250
13	4.81e+21	9.3943824e+21	17250
14	9.395e+21	2.3697264e+22	17250
15	2.3703049e+22	1.071784e+28	15811

Table 8: Ethereum transaction data

Finding the correlation between the price and no of transactions in each layer:

First, I am doing inner join between VeChain price data with each layer by date to create the dateframe for finding the correlation. Then, we normalize the **Close price** and **no of transactions** columns to easily plot the correlation between them. We used **spearman correlation algorithm** to find the correlation which yields better result than other algorithms like Kendall and Pearson. Below are the correlation plot for some layers with respect to Close price.



(a) Layer 1 (b) Layer 2

Figure 3: Correlation plots Layer 1 and Layer 2.

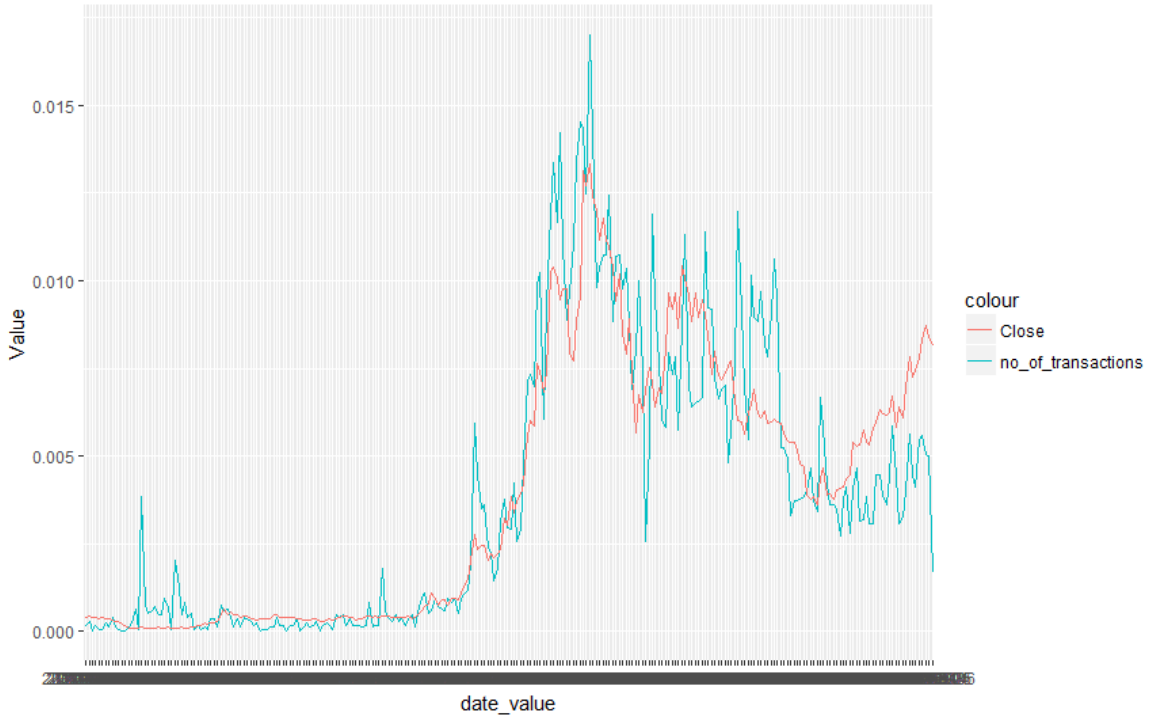
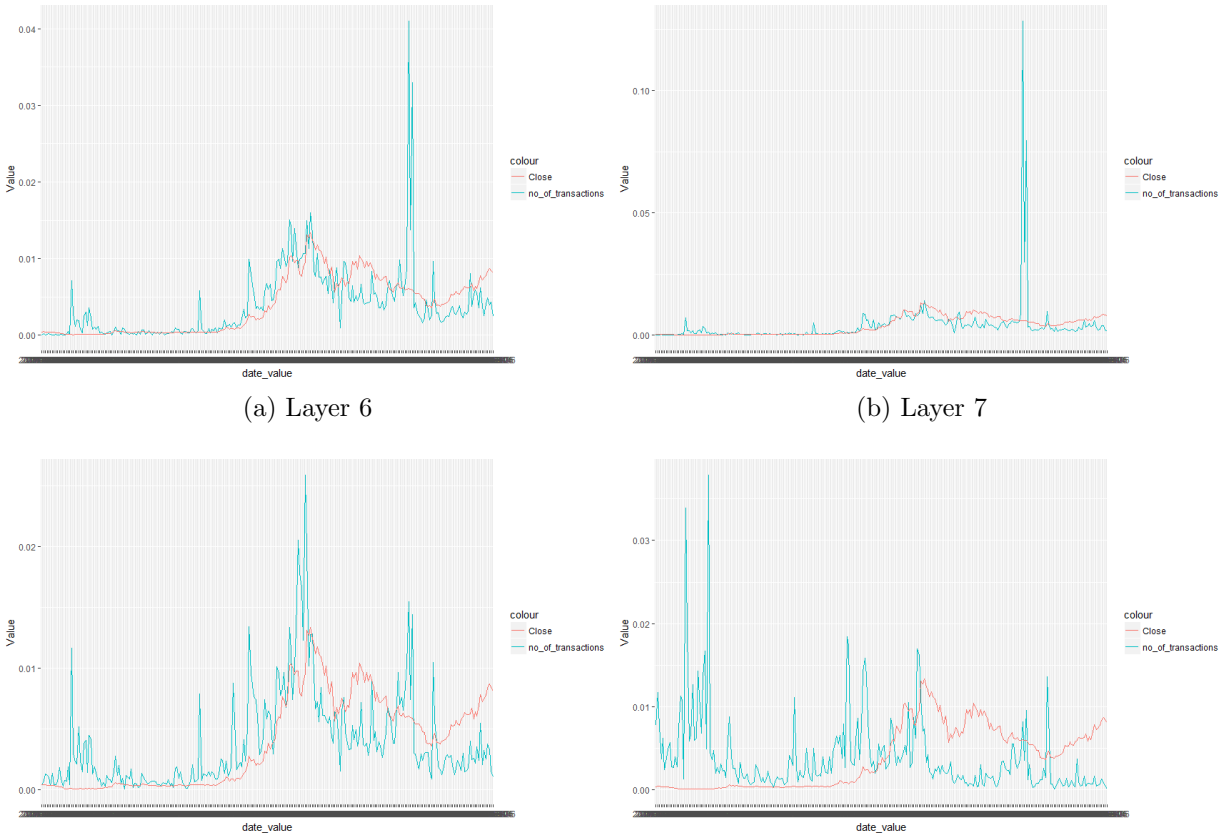


Figure 4: Correlation plots Layer 3

Table below shows the correlation co-efficient for each layer between No of transactions and Close Price data.

Layer	**Correlation Value**
1	0.80577881839797
2	0.876665933792494
3	0.884761927827058
4	0.866706530030129
5	0.867942503109674
6	0.807705244016247
7	0.790804898175551
8	0.720135621446249
9	0.703813426238188
10	0.699120921406116
11	0.66695016013608
12	0.480012763468364
13	0.332736280839345
14	0.0985150470835267
15	-0.226270425352802

Table 9: Correlation value for each layer



(a) Layer 6

(b) Layer 7

(c) Layer 9

(d) Layer 15

Figure 5: Layers correlation plots

Result: From the above table 9, we can find that the correlation is decreasing with increase in the layer.

4.4 Multiple Linear Regression

Here, we need to create a multiple linear regression model to explain the price return on day t. Price return is given by

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

where P_t is the token price in dollar for the t^{th} day.

By extracting at least three features from the token network at day t1, the objective is to create a multiple linear regression model to explain price return on day t.

I have taken layer 3 data for creating multiple linear regression since it has high correlation value than any other layers. To extract features from that layer, I have considered 3 features from ethereum dataset, percentage of difference in the number of transactions, percentage of difference in the number of unique buyers and percentage of difference in the number of unique seller on $t - 1^{th}$ day. Also, I have extracted 2 more features from price dataset, percentage of difference in the Open price and percentage of difference in Volume.

Percentage difference is calculated as follows,

$$\frac{Value_{t-1} - Value_{t-2}}{Value_{t-2}}$$

In order extract those features, first, I find out unique buyers and sellers on day t, and the number of transaction on day t. After I find those features for every date, I have merged all these 3 features by date column, and obtained a single data frame.

Date	**Number of transactions**
2017-08-18	14
2017-08-19	13
2017-08-20	1
2017-08-21	3
2017-08-22	3
2017-08-23	5

Table 10: Feature 1 from ethereum dataset

Date	**Number of unique buyers**
2017-08-18	10
2017-08-19	13
2017-08-20	1
2017-08-21	2
2017-08-22	3
2017-08-23	5

Table 11: Feature 2 from ethereum dataset

Date	Number of unique sellers
2017-08-18	7
2017-08-19	5
2017-08-20	1
2017-08-21	3
2017-08-22	3
2017-08-23	3

Table 12: Feature 3 from ethereum dataset

Date	percent diff in transactions	percent diff in unique buyers	percent diff in ur
2017-08-24	66.666667	66.666667	0
2017-08-25	-100.000000	-100.000000	-100.000000
2017-08-26	-66.666667	-66.666667	-66.666667
2017-08-27	0	0	0

Table 13: Features dataset from ethereum transaction

Now, I have chosen **Close price** value from price data, to evaluate **Price return** for every date. Also, extracted 2 featurs from price data, percentage increase in open price and volume as Table 14, and finally merged with the **features data** by date column to obtain final data frame as shown in Table 15 for multiple linear regression.

Date	Close price return	percent diff in Volume	percent diff in open price
2017-08-24	0.0485631442	-89.8734177	9.8514979
2017-08-25	-0.0841151262	754.1666667	4.85631442
2017-08-26	-0.0360896871	-2.4390244	-3.97948215
2017-08-27	0.0297554748	37.5000000	-3.60896871

Table 14: Close price return and Features from price data

Close price return	percent diff in transactions	percent diff in unique buyers	percent
0.0485631442	66.666667	66.666667	0
-0.0841151262	-100.000000	-100.000000	-100.00000
-0.0360896871	-66.666667	-66.666667	-66.66666
0.0297554748	0	0	0

Table 15: Multiple linear regression dataset

Now, the multiple linear regression is modelled using the extracted features and the result obtained as shown in Fig 6 and residuals plot in Fig 7

```
Call:
lm(formula = Return_Val ~ Merged_Features[, "Increase_In_Transaction"] +
    Merged_Features[, "Increase_In_Buyers"] + Merged_Features[,
    "Increase_In_Sellers"] + Merged_Features[, "Increase_In_Vol"] +
    (Merged_Features[, "Increase_In_Sellers"] * Merged_Features[,
        "Increase_In_Vol"]), data = Merged_Features)

Residuals:
     Min       1Q   Median       3Q      Max
-0.43364 -0.07666 -0.01158  0.06189  0.58039

Coefficients:
                                                                                Estimate Std. Error t value
(Intercept)                                                                    0.0157669  0.0085621   1.841
Merged_Features[, "Increase_In_Transaction"]                                  -0.0482479  0.0226373  -2.131
Merged_Features[, "Increase_In_Buyers"]                                        0.0507400  0.0228826   2.217
Merged_Features[, "Increase_In_Sellers"]                                       0.0323292  0.0098417   3.285
Merged_Features[, "Increase_In_Vol"]                                           0.0016581  0.0005730   2.894
Merged_Features[, "Increase_In_Sellers"]:Merged_Features[, "Increase_In_Vol"] -0.0005282  0.0002853  -1.851
                                                                              Pr(>|t|)
(Intercept)                                                                    0.06676 .
Merged_Features[, "Increase_In_Transaction"]                                   0.03406 *
Merged_Features[, "Increase_In_Buyers"]                                        0.02751 *
Merged_Features[, "Increase_In_Sellers"]                                       0.00117 **
Merged_Features[, "Increase_In_Vol"]                                           0.00415 **
Merged_Features[, "Increase_In_Sellers"]:Merged_Features[, "Increase_In_Vol"]  0.06532 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1312 on 245 degrees of freedom
Multiple R-squared:  0.09897,   Adjusted R-squared:  0.08058
F-statistic: 5.382 on 5 and 245 DF,  p-value: 0.0001026
```

Figure 6: Multiple Linear regression result

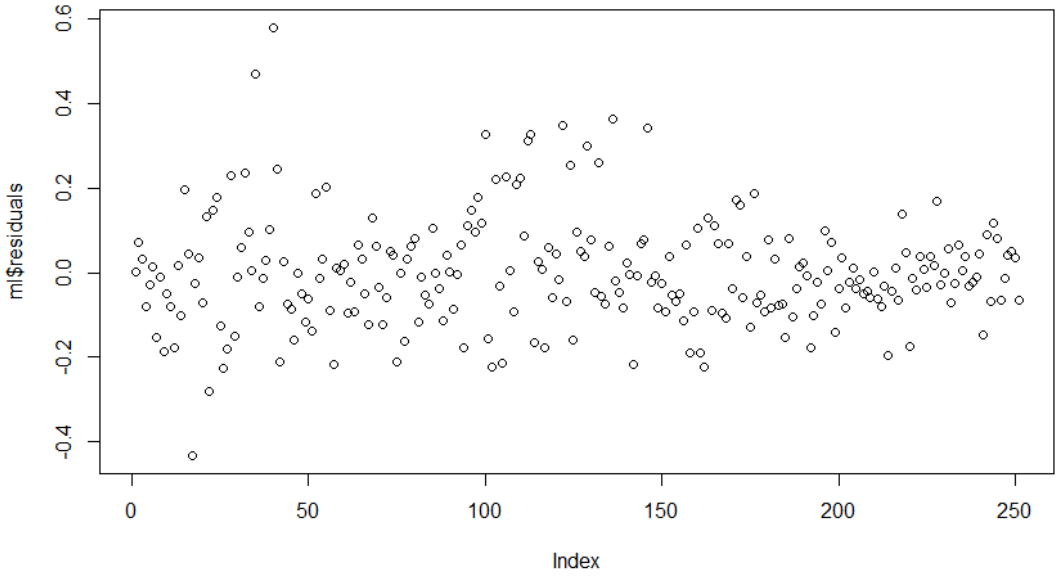


Figure 7: Multiple Linear regression residuals

Result:

From the above result, we can find that the model R^2 value is 0.08 which means the extracted feature doesn't yield good result. I have tried with the **previous day close price return** value as one of the features to improve model, but it also doesn't give better results.

I have understood that the correlation between combined values of the extracted features and price return should be high to get good model. Only percentage of difference in unique seller feature from ethereum dataset, and percentage of volume change feature in price data from price dataset has good impact on the overall model.

From the residual plot in Fig 7 , we can see that errors are in few fractions since the output value we are interested itself is in fractions.And most of the values are missed by the model in fractions.

5 References:

.

[1] https://cointelegraph.com/explained/erc-20-tokens-explained

[2] VeChain Development plan and White paper Version 1.0.0.0

[3] Online R materials.