Ethereum

Ethereum is an open platform that enables developers to build and deploy decentralized applications such as smart contracts and other complex legal and financial applications. One can think of Ethereum as a programmable Bitcoin where developers can use the underlying blockchain to create markets, shared ledgers, digital organizations, and other endless possibilities that need immutable data and agreements, all without the need for a middleman.

Smart contract is just a phrase used to describe computer code that can facilitate the exchange of money, content, property, shares, or anything of value. Smart contracts are created by developers and put to a blockchain address. When running on the blockchain a smart contract becomes like a self-operating computer program that automatically executes when specific conditions are met. Some smart contracts implement mechanisms that allow trading or sharing digital assets, known as crypto-tokens, on the blockchain.

Ethereum ERC-20 Tokens

ERC-20 defines a common list of rules for all Ethereum tokens to follow, meaning that this token empowers developers of all types to accurately predict how new tokens will function within the larger Ethereum system. The impact on developers is on next level as developers no need to redesign projects when each new token is released.

Primary tokens

1. Tenxpay Token
   It is a platform for payment that focuses on enabling users to use cryptocurrency in the day-to-day transactions. The company aims to accelerate crypto adoption for mainstream consumers

2. Bat Token
   It aims to correctly price user attention within the platform. Advertisers pay ads. It seeks to address fraud and opaqueness in digital advertising. Advertisers pay BAT to website publishers for the attention of users.

3. Storj Token
   It provides decentralized cloud storage to its community. It works on the premise that each user is entitled to the same amount of space on the network as they make available to the community. Users can also receive payment for renting their extra disk space.

Project Goal

There are two goals of this project. The first goal is to Find the distribution of how many times a pair user (i.e., address1 and address2) 1 - buys, 2 - sells a token with each other. Which distribution type fits these distributions best? Estimate population distribution parameters. And the second goal is to develop a regression model where “buys” of the top K buyers (by number of buys or amount of buys) are regressors, and token price is the outcome. And determine a K value to have the best regression results.

Part 1: Finding the distribution

1) Loading the files

```
tenxpay <- read_delim('networktenxpayTX.txt', delim = " ", col_names = F)
```

2) Removing the impossible values

If any of the transaction amount exceeds the total amount of tokens available in the network then those transactions are impossible, such transactions can be filtered out as follows

```
filteredtenxpay <- filter(tenxpay,tokenAmount < totalSupply)
```

Number of impossible values were 1, 6 and 53 in Texpay, Bat and Storj respectively.

3) Finding the outliers and removing them

All those transactions where the buyer and seller ids are same are considered as outliers. They can be removed as follows:

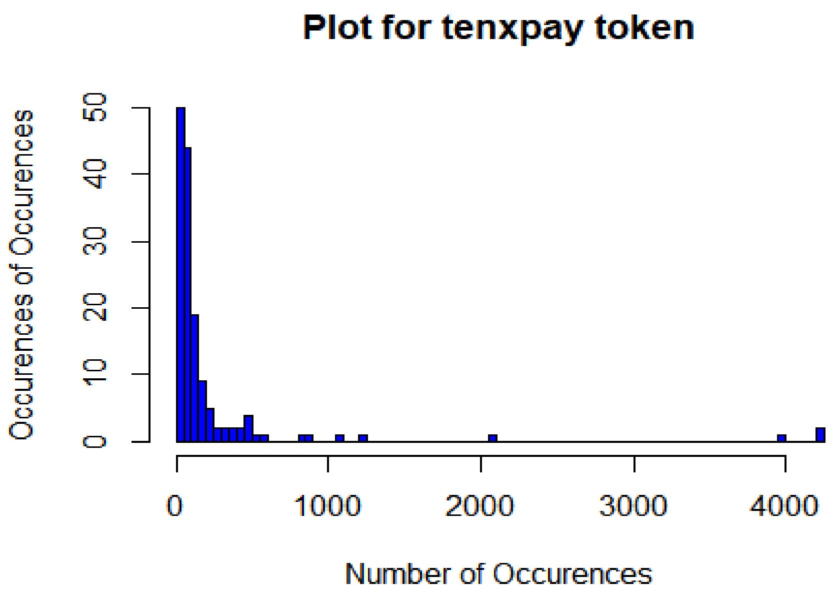
```
filteredtenxpay <- filter(filteredtenxpay,tokenAmount < totalSupply)
```

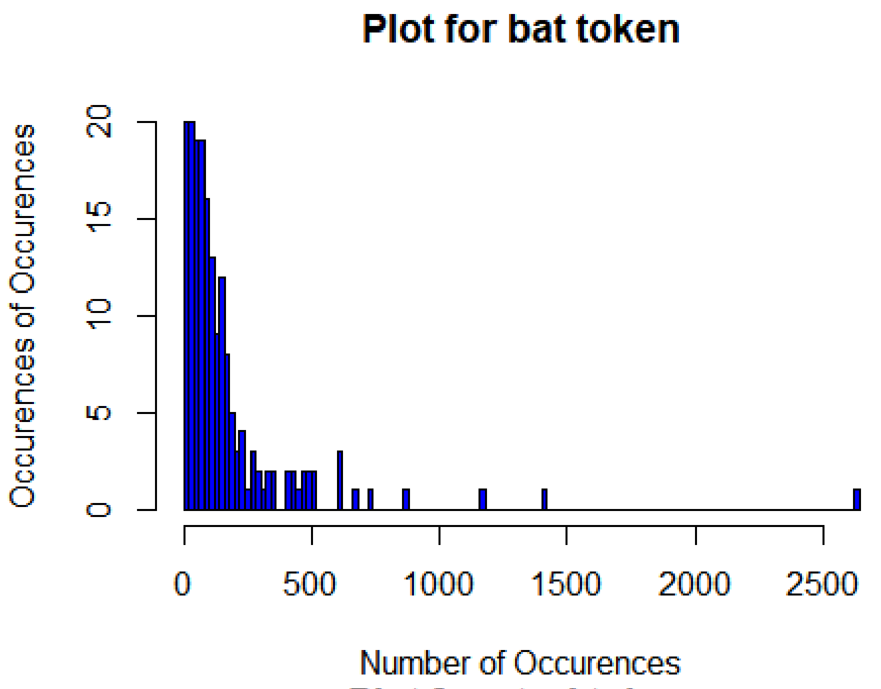
Number of outliers were 10079, 10709, 3572 in Texpay, Bat and Storj respectivel.y

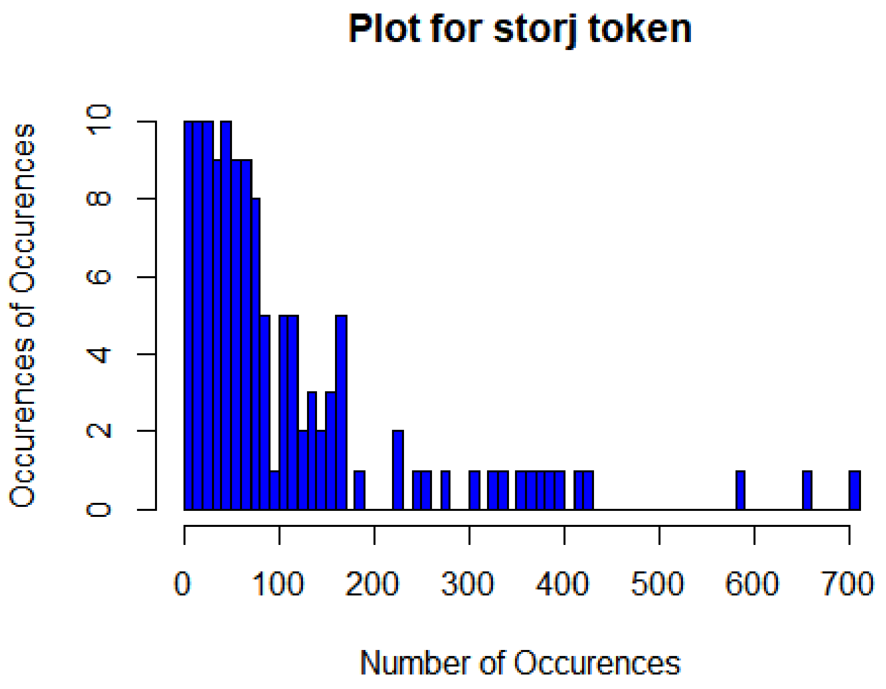
4) Visualizing the data

After removing the outliers and the impossible values we fetch the unique pairs of buyers and sellers along with their number of transactions.

```
result <-filteredtenxpay %>% count(fromID,toID, sort = FALSE)
result$Occ = 1
result_new <- aggregate(result$Occ, by=list(result$Occurences), FUN=sum)
##   Number Occurences
## 1      1     131029
## 2      2      27246
## 3      3       8532
## 4      4       3875
## 5      5       2066
## 6      6       1258
```







5) Possible distributions that could fit the data:
   Observing the histogram for all three tokens, distributions like Weibull, Gamma, Log Normal, Exponential, Poisson could possibly fit the data.
6) Functions and packages that are used for fitting the data:
   Package used: fitdistrplus
   Function useds:
   fitdist(data,distr)
   where, data is a numeric vector and
   distr is A character string "name" naming a distribution for which the corresponding density function dname, the corresponding distribution function pname and the

corresponding quantile function qname must be defined, or directly the density function.

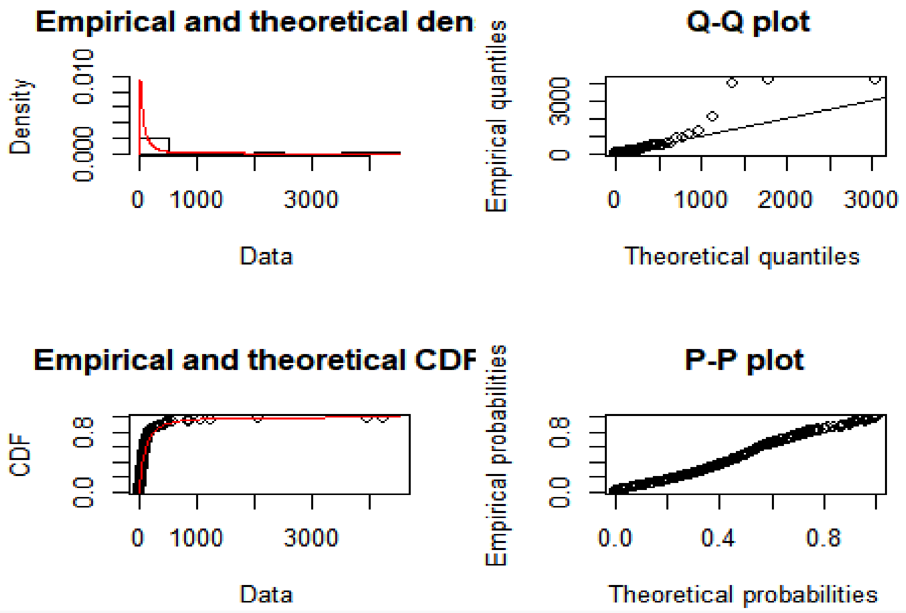
gofstat(f)

Where, f is An object of class "fitdist", output of the function fitdist, or a list of "fitdist" objects.

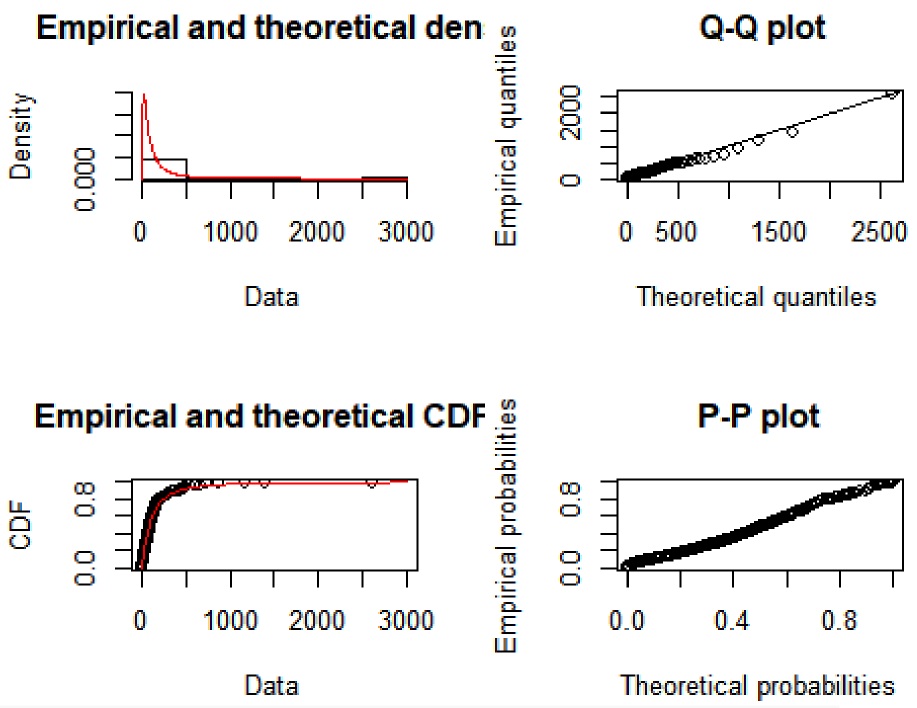
```
fit.lnorm.result <- fitdist(result_new$Number, 'lnorm')
gofstat(list(fit.weibull.result, fit.gamma.result, fit.lnorm.result, fit.exp.result, fit.log.result, fit.
pois.result))
```

7) Findings:

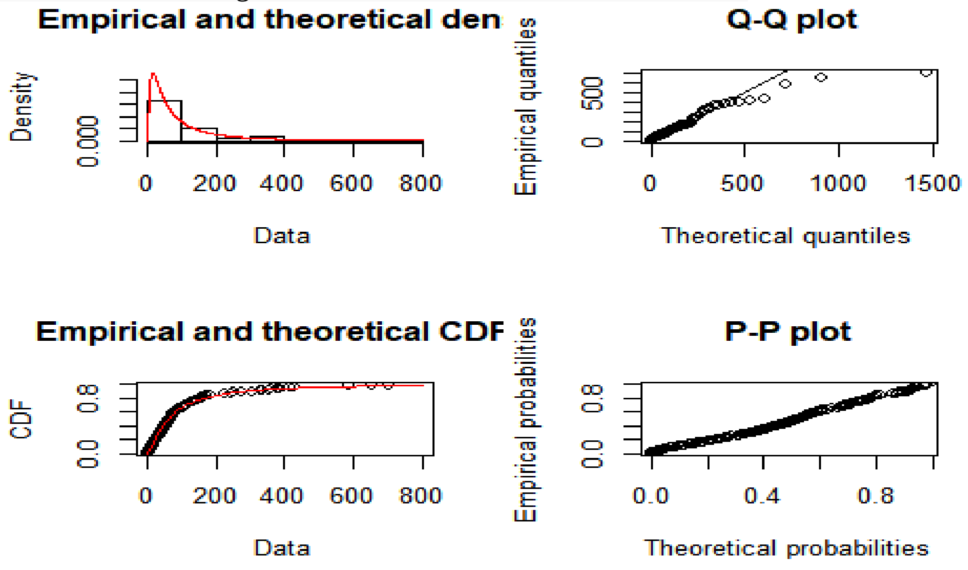
```
## Goodness-of-fit statistics
##                               1-mle-weibull 2-mle-gamma 3-mle-lnorm
## Kolmogorov-Smirnov statistic    0.1361148   0.1980425  0.07326387
## Cramer-von Mises statistic      0.9125615   1.7294835  0.23198072
## Anderson-Darling statistic      5.3549628   8.9203993  1.29183124
##                                 4-mle-exp 5-mle-logis 6-mle-pois
## Kolmogorov-Smirnov statistic  0.2872441   0.3147257  0.8109528
## Cramer-von Mises statistic    4.6133664   3.5188683 28.6724683
## Anderson-Darling statistic  22.0856395  20.1473062        Inf

plot(fit.lnorm.result)
```



```
## Goodness-of-fit statistics    1-mle-weibull 2-mle-gamma 3-mle-lnorm
## Kolmogorov-Smirnov statistic   0.09264762   0.1091445  0.05864597
## Cramer-von Mises statistic     0.33488301   0.4598649  0.16653606
## Anderson-Darling statistic     1.95215865   2.3737594  0.96295862
##                                4-mle-exp 5-mle-logis 6-mle-pois
## Kolmogorov-Smirnov statistic 0.1181292   0.2127198   0.646231
## Cramer-von Mises statistic   0.6465903   1.6148370  22.183829
## Anderson-Darling statistic   3.0716912  11.2372824        Inf
plot(fit.lnorm.result)
```



```
## Goodness-of-fit statistics  1-mle-weibull 2-mle-gamma 3-mle-lnorm
## Kolmogorov-Smirnov statistic   0.08070105  0.09181971  0.06026925
## Cramer-von Mises statistic     0.15736680  0.19431060  0.10223441
## Anderson-Darling statistic     0.93148358  1.07418524  0.63858344
##                               4-mle-exp 5-mle-logis 6-mle-pois
## Kolmogorov-Smirnov statistic 0.09605241   0.1917018   0.625439
## Cramer-von Mises statistic   0.21412425   0.8888001  12.701993
## Anderson-Darling statistic   1.15387288   7.1571758        Inf
```



8) **Conclusion**: Based on the goodness of fit statistics & criteria of all the tokens Log normal fits the buyer-seller distribution the best. Moreover, Log normal is used to model human behaviors like user's dwell time on an online article or post. Hence, it makes sense that pairs of buyer and seller distribution also follows Log Normal distribution.

Part 2: Finding the top K buyers:

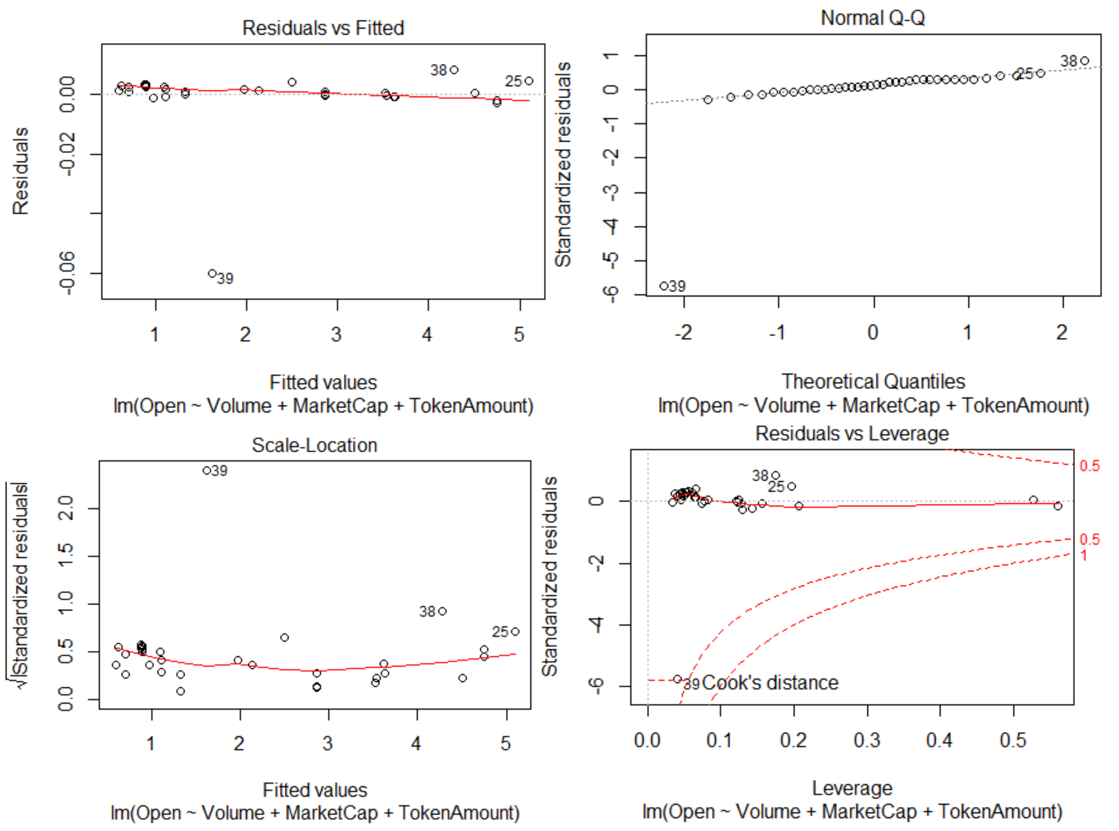
1) Since, we need to predict the price (present in token price file) using the values available in the network file we first need to merge those files together.
2) In order to merge network files with token price files we need to make sure both of those files are representing the data in the same format. For this we need to change the format of tokenAmount and timestamp
3) After, merging the files together, we take the top K buyers from the list and calculate the collinearity of Open price & Token Amount, Open price & Market Cap, Open Price & Volume.
4) ## K=50

```
cor.test(TopUniqueBuyers$Open,TopUniqueBuyers$Volume)

cor.test(TopUniqueBuyers$Open,TopUniqueBuyers$MarketCap)

cor.test(TopUniqueBuyers$Open,TopUniqueBuyers$TokenAmount)
```

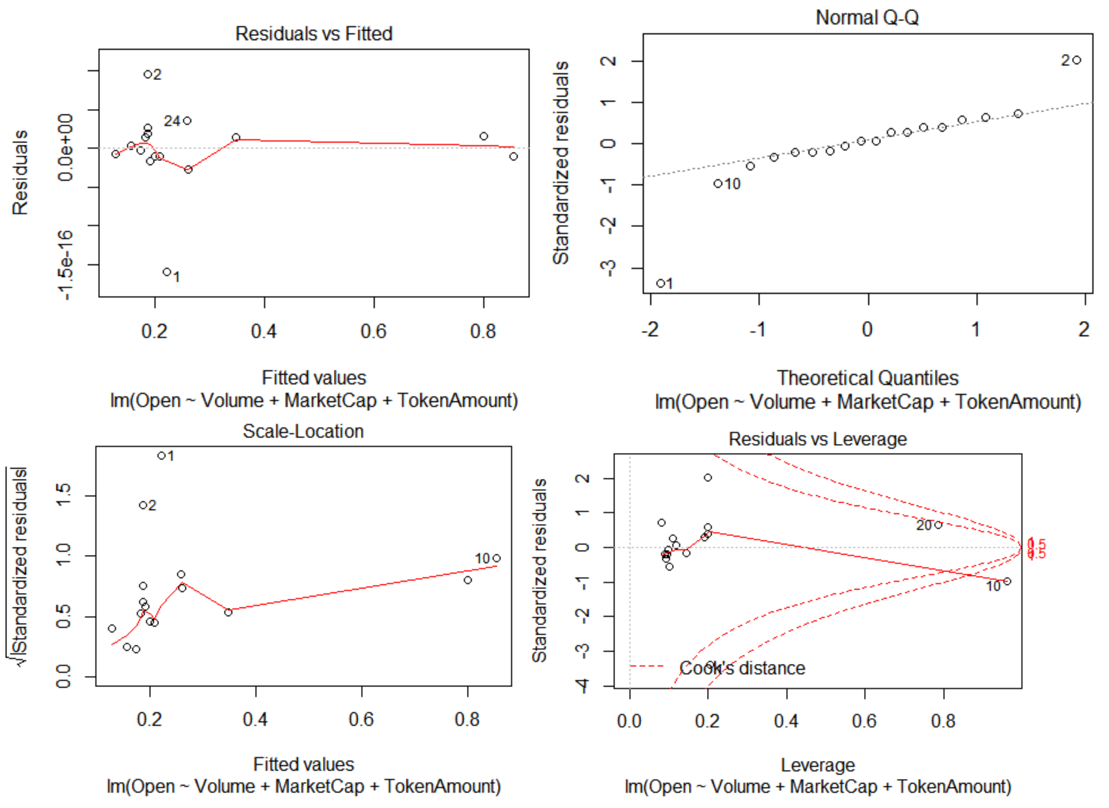
   Values of K are 50,25,5 for Tenxpay, Bat and Storj respectively

5) We trained a multiple linear regressor using Volume, MarketCap and TokenAmout to predict the Opening Price of the token

```
linearModel <- lm(formula=Open ~ Volume+MarketCap+TokenAmount, data=TopUniqueB
uyers)
summary(linearModel)

plot(linearModel)

## K=50 for Texnpay

## lm(formula = Open ~ Volume + MarketCap + TokenAmount, data = TopUniqueBuyers)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.059979 -0.000080  0.001546  0.003078  0.008202
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.942e-03  3.847e-03  -1.025    0.313
## Volume       4.726e-11  7.910e-11   0.597    0.554
## MarketCap    9.549e-09  1.649e-11 578.950   <2e-16 ***
## TokenAmount  1.698e-09  1.994e-09   0.852    0.400
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01066 on 34 degrees of freedom
##   (12 observations deleted due to missingness)
## Multiple R-squared:  0.9999, Adjusted R-squared:  0.9999
## F-statistic: 2.147e+05 on 3 and 34 DF,  p-value: < 2.2e-16
```

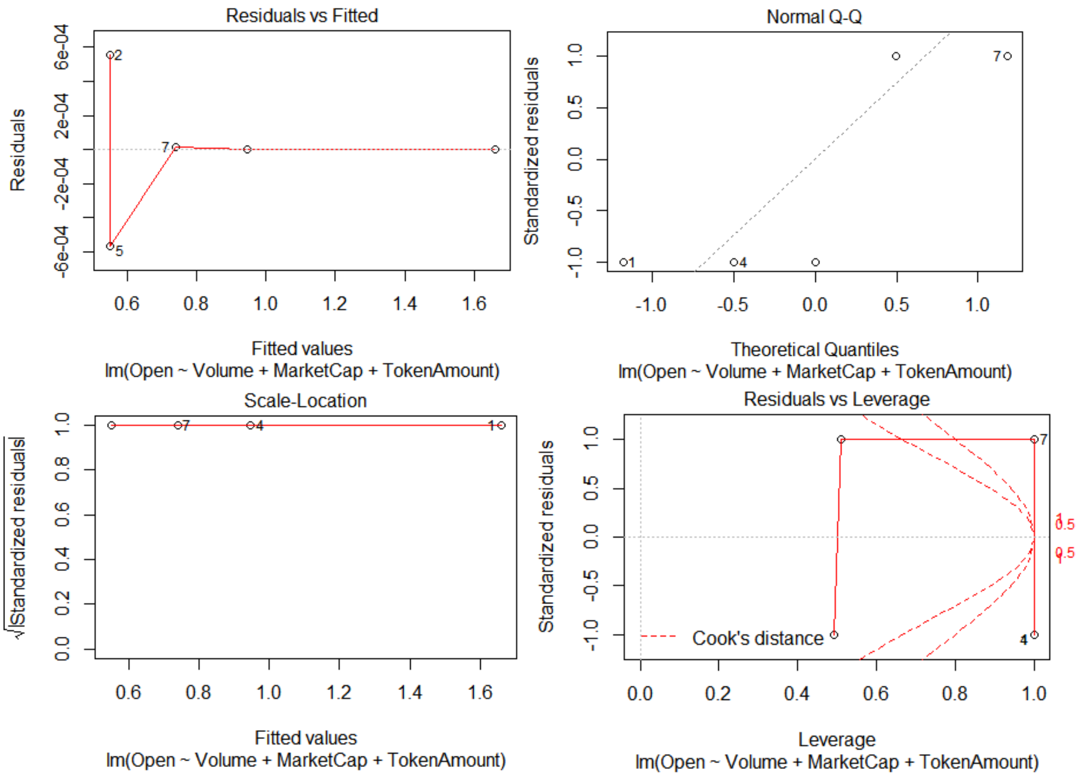


```
## K=25 for Bat

Residual standard error: 5.281e-17 on 14 degrees of freedom
##   (7 observations deleted due to missingness)
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 8.773e+31 on 3 and 14 DF,  p-value: < 2.2e-16
```



```
## K=5 for Storj
```

```
## Residual standard error: 0.0007915 on 1 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 4.53e+05 on 3 and 1 DF,  p-value: 0.001092
```



6) **Conclusion**: Values of K for Tenxpay is 50, Bat is 25, Storj K=5, Hence, by just considering the top 50,25 and 5 buyers we can predict the token price. This means that in order to predict token price, we don't need to considered all the transactions on the transaction from top K buyers are sufficient. We could just ignore the rest of the transactions. All these top K buyers are strongly influencing the token price.

References:

Coinmarketcap.com - https://coinmarketcap.com/
Etherscan - https://etherscan.io/