Project Report on Ethereum Tokens Analysis

Submitted by:

Yashwanth Yella (YXY180024)

Vyom Gupta (FXV180000)

Ethereum and ERC20 Tokens:

Ethereum is a platform used for creating any randomly selected smart contract that represent digital assets called Ethereum tokens. Tokens can have a fixed supply, constant inflation rate, or even a supply determined by a sophisticated monetary policy. Tokens can be used for a variety of purposes such as paying to access a network or for decentralized governance over an organization.

Primary tokens used in this project

EOS token, Tronix token and Omisego token

Description about Tronix token:

Tronix is designed to ultimately make entertainment content both easier to sell and cheaper to consume. We achieve this by putting the content on a blockchain and the creators and consumers in a network of peers, eliminating the middleman.

Project Goal

Our goal is to extract the data from the tokens and remove the outliers if any present. The outliers are those on which the transaction amount is greater than the total circulating amount of the token. We also need to remove the transactions in which the buyer id is equal to the seller id.

Next, is to find the best distribution between number of sells and buys between the pair of users. This can be done by fitting all types of distributions on the processed data and finding which one has maximum accuracy.

We will find the most active buyers and sellers in one token and track them in another tokens by fitting the linear regression model with the buys of top k buyers as regressors and token price as the output value.

Project: Part 1

The sum of our UTD ids (2021440884 and 2021434517) when divided by 20 leaves remainder as 1 so we have selected the 1st, 2nd and 3rd tokens.

```
library(plyr)
library(Hmisc)
library(fitdistrplus)
```

```
library(magrittr)

library(dplyr)

library(sqldf)
```

fitdistrplus package functions are fitdist for fit on non-censored data and fitdistcens for fit on censored data.

Magrittr package function is structuring sequences of data operations left-to-right, avoiding nested function calls, minimizing the need for local variables and function definitions.

Dplyr package is grammar of data manipulation, providing a consistent set of verbs that help you solve the most common data manipulation challenges. It also contains summarise() function.

Sqldf package is an R package for running SQL statements on R data frames, optimized for convenience.

```
#Load Data

token_eos = read.table(file="D:/sem2/stats for DS/project/networkeosTX.txt"
, header=F,sep=" ")

token_sego = read.table(file="D:/sem2/stats for DS/project/networkomisegoTX
.txt", header=F,sep=" ")

token_onix = read.table(file="D:/sem2/stats for DS/project/networktronixTX.
txt", header=F,sep=" ")

#outliers

outlier_eos <- token_eos [which (token_eos$V4 >= 1.0e+27), ]

outlier_sego <- token_sego [which (token_sego$V4>=1.4e+26), ]

outlier_onix <- token_onix [which (token_onix$V4 >= 1.0e+17), ]

#outliers print

head(outlier_eos)

head(outlier_onix)

head(outlier_sego)

#remove the rows in which seller_id equals buyer_id

token_eos = token_eos [which(token_eos$V1!=token_eos$V2),]

token_sego = token_sego [which(token_sego$V1!=token_sego$V2),]

token_onix = token_onix [which(token_onix$V1!=token_onix$V2),]

#removing the outliers

eos_token <- token_eos [which (token_eos$V4 < 1.0e+27), ]

sego_token <- token_sego [which (token_sego$V4 < 1.4e+26), ]

onix_token <- token_onix [which (token_onix$V4 < 1.0e+17), ]

# filtered tokens

print("The tokens after removing outliers are:")

## [1] "The tokens after removing outliers are:"
```

```
print("eos token")
```

Here we are loading the data, finding the outliers and printing them and removing them.

```
#create data frame for buyers and sellers pairs
df_eos <- as.data.frame(cbind(eos_token$V1, eos_token$V2))
df_sego <- as.data.frame(cbind(sego_token$V1,sego_token$V2))
df_onix <- as.data.frame(cbind(onix_token$V1, onix_token$V2))
#find frequency of each pair
counts_eos <- ddply(df_eos,.(df_eos$V1,df_eos$V2),nrow)
counts_sego <- ddply(df_sego,.(df_sego$V1,df_sego$V2),nrow)
counts_onix <- ddply(df_onix,.(df_onix$V1,df_onix$V2),nrow)
#print frequencies
head(counts_eos)
```

```
##   df_eos$V1 df_eos$V2  V1
## 1   3116916     40044   1
## 2   3116917     40044   1
## 3   3101799   3116918   1
## 4   3091415    145709 137
## 5   3116919     40190   1
## 6   3098886   3116920   1
```

We are creating a data frame for the tokens for easy access and finding the frequencies of transactions of each pair.

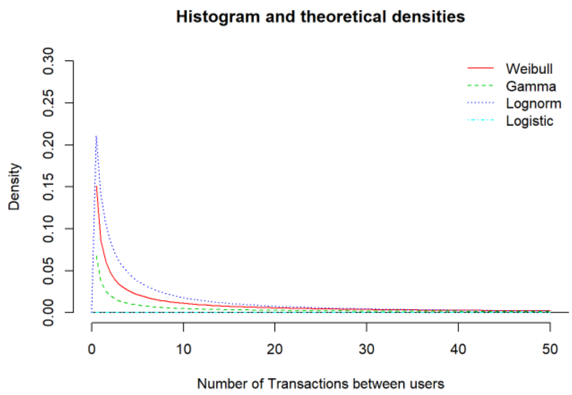
```
#Create data frame for ID and frequency
x_eos <- data.frame(counts_eos$V1)
x_sego <- data.frame(counts_sego$V1)
x_onix <- data.frame(counts_onix$V1)
#Print ID and frequency
head(x_eos)
```

We are creating a data frame for the id and frequency and printing them.

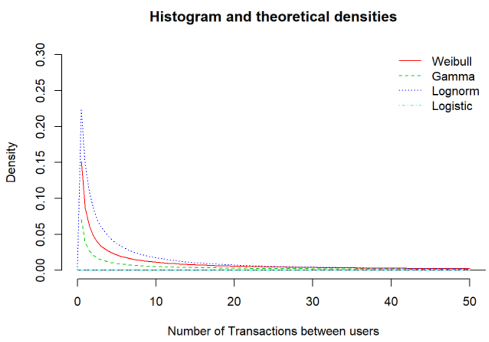
```
#Calculate Frequency of Counts
FrequencyOfFrequency_eos<-as.data.frame(table(x_eos$pairCount))
colnames(FrequencyOfFrequency_eos)<-c("No_of_Transactions","Frequency")
#FrequencyOfFrequency_eos
FrequencyOfFrequency_sego<-as.data.frame(table(x_sego$pairCount))
colnames(FrequencyOfFrequency_sego)<-c("No_of_Transactions","Frequency")
```

```
FrequencyOfFrequency_onix<-as.data.frame(table(x_onix$pairCount))
colnames(FrequencyOfFrequency_onix)<-c("No_of_Transactions","Frequency")
#Fit Distribution of eos
pois_eos<-fitdist(FrequencyOfFrequency_eos[,2], distr = "pois")
weibull_eos<-fitdist(FrequencyOfFrequency_eos[,2], distr = "weibull")
lnorm_eos<-fitdist(FrequencyOfFrequency_eos[,2], distr = "lnorm")
log_eos<-fitdist(FrequencyOfFrequency_eos[,2], distr = "logis")
gamma_eos<-fitdist(FrequencyOfFrequency_eos[,2],"gamma",lower = c(0, 0), st
art = list(scale = 1, shape = 1))
gofstat(list(weibull_eos,gamma_eos,lnorm_eos,log_eos))
```

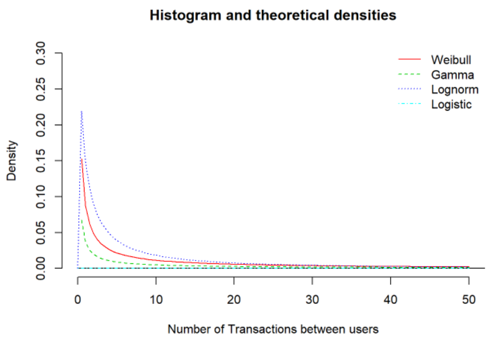
Here we are calculating the frequency of each counts and storing them. Then we are fitting the various types of distributions for that frequency.



EOS token



OMISEGO token



TRONIX token

Here we have observed that the log normal distribution fits the best for all the 3 tokens and we can see from the statistics table shown below:

```
## Goodness-of-fit statistics
##                              1-mle-weibull 2-mle-gamma 3-mle-lnorm
## Kolmogorov-Smirnov statistic     0.3499491   0.3989422   0.2267018
## Cramer-von Mises statistic       8.3984724  22.0297745   4.5676141
## Anderson-Darling statistic      45.0317290         Inf  26.3560421
##                              4-mle-logis
## Kolmogorov-Smirnov statistic   0.4882434
## Cramer-von Mises statistic    28.5020881
## Anderson-Darling statistic           Inf
## Goodness-of-fit criteria
##                                1-mle-weibull 2-mle-gamma 3-mle-lnorm
## Akaike's Information Criterion      3112.329    3613.667    2822.884
## Bayesian Information Criterion      3120.183    3621.521    2830.738
##                                4-mle-logis
## Akaike's Information Criterion    7879.676
## Bayesian Information Criterion    7887.530
```

Same process is performed for remaining 2 tokens and we can observe the similar table in which we can see the log normal has the best fit.

Project: Part 2

```
#eoscoin data imports


eosPrice = read.table(file="D:/sem2/stats for DS/project/eos.txt", header=F
,sep="\t")
colnames(eosPrice) <- c("Date","Open_Price","High_Price","Low_Price", "Clos
e_Price" ,"Volume","Market Cap")
colnames(eos_token)<-c("FromNodeId", "ToNodeId", "Unixdate", "TokenAmount")
#Change Date Formats
eosPrice$Date<-as.Date(eosPrice$Date,format= "%m/%d/%Y")
eosPrice$Date<- as.Date(as.POSIXct(eosPrice$Date, origin="1970-01-01"))
eos_token$Unixdate<- as.Date(as.POSIXct(eos_token$Unixdate, origin="1970-01
-01"))
#Print eoscoin Data
head(eosPrice)
```

Here we are loading he price tokens and changing the date format present in it.

```
#eos

buys.distribution.eos  <- eos_token %>% group_by(eos_token[, 2]) %>% summar
ise(n = n()) %>% ungroup

colnames(buys.distribution.eos) <- c("BuyerId","Frequency_of_buys")

sortedeos_Buyer <- buys.distribution.eos[order(-buys.distribution.eos$Frequ
ency_of_buys),]

#most active buyerId and no of times the buyer bought the token

head(sortedeos_Buyer,1)
```

Here we are grouping the tokens based on receiver addresses and number of counts. Then we are sorting the list based on frequency of buys.

Same step is performed for remaining 2 tokens.

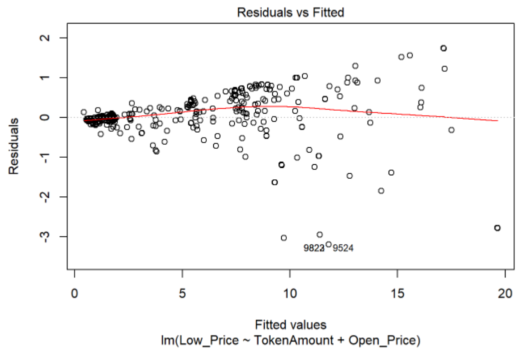
```
library(sqldf)

eos_token <- eos_token[order(-eos_token$TokenAmount),]

eosRegression<-sqldf("SELECT d.Unixdate,p.High_Price, p.Low_Price, p.Open_P
rice, p.Close_Price, d.TokenAmount FROM eos_token d NATURAL JOIN eosPrice p
WHERE d.Unixdate = p.Date LIMIT 10000")

eosRegression <- eosRegression[order(-eosRegression$TokenAmount, -eosRegres
sion$Open_Price),]

lm.fit.eos= lm(Low_Price ~ TokenAmount+Open_Price, data= eosRegression)

summary(lm.fit.eos)
```

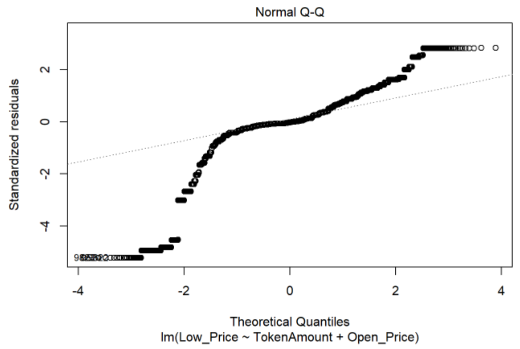
Now, we are trying to fit a linear regression model with regressors as EOS regression and with token amount as output. We can observe the summary of fit as below:

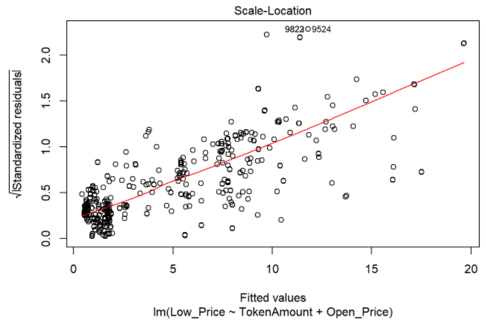
```
## Call:

## lm(formula = Low_Price ~ TokenAmount + Open_Price, data = eosRegression)

## Residuals:

##     Min       1Q   Median       3Q      Max

## -3.2015 -0.1100 -0.0136  0.2291  1.7389

## Coefficients:

##               Estimate Std. Error t value Pr(>|t|)

## (Intercept)  8.940e-02  9.581e-03   9.331   <2e-16 ***

## TokenAmount -2.123e-28  3.223e-28  -0.659     0.51

## Open_Price   9.037e-01  1.313e-03 688.022   <2e-16 ***

## ---

## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

## Residual standard error: 0.6135 on 9997 degrees of freedom

## Multiple R-squared:  0.9793, Adjusted R-squared:  0.9793
```

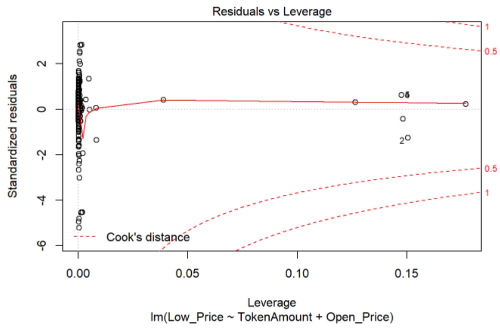
```
## F-statistic: 2.367e+05 on 2 and 9997 DF,  p-value: < 2.2e-16
```

The following graphs are for **EOS** token:





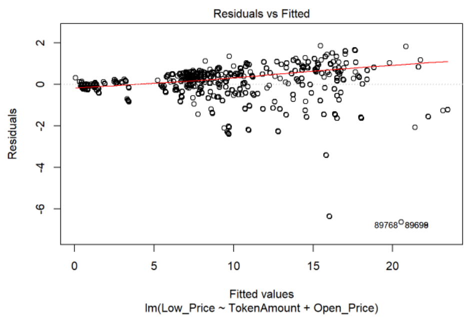


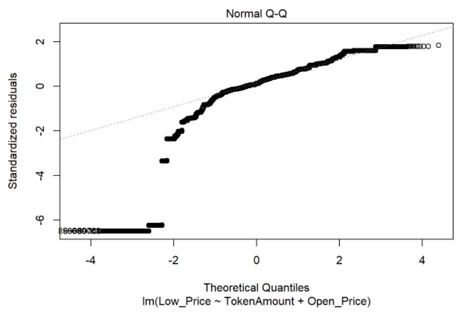


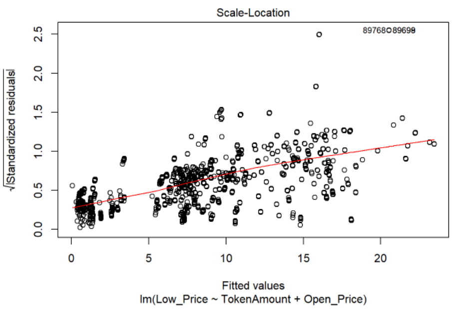
We got an adjusted R square value of 0.97 which is a good fit.

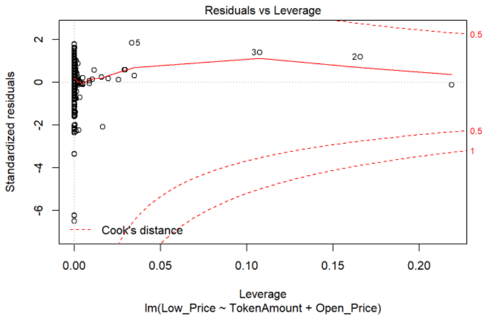
Repeat the same for the other 2 tokens and we get the following graphs:

For **OMISEGO** token:



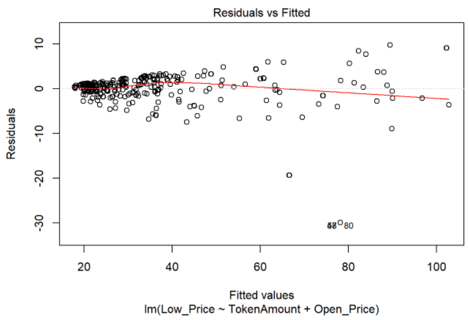


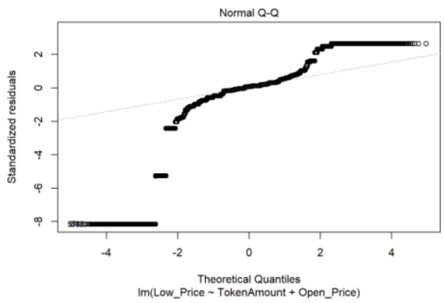


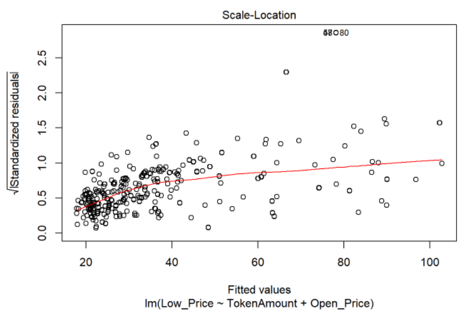


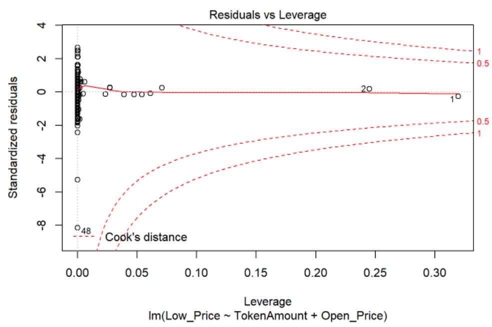
The adjusted R square value for this token is 0.96.

Graphs for **TRONIX** token are as follow:









The adjusted R square for this token is 0.978.

Conclusion

We have analysed the 3 tokens and observed that there were many outliers and illegal transactions present. We removed all those outliers and made a good distribution of the remaining data available. We also fit a linear regression model using the token prices as the regressors and token amount as the output. We got accurate results for model that we designed.