TENXPAY

YESHWANT RANKA YVR180000
SAURABH MALKAR SXM170231

Ethereum

Ethereum is a decentralized platform that runs smart applications that run exactly as programmed without any possibility of downtime, censorship, fraud or third-party interference.These apps run on a custom built blockchain, an enormously powerful shared global infrastructure that can move value around and represent the ownership of property.

In the Ethereum blockchain, miners work to earn Ether, a type of crypto token that fuels the network. Beyond a tradeable cryptocurrency, Ether is also used by application developers to pay for transaction fees and services on the Ethereum network.While all blockchains have the ability to process code, most have their limitations. On the contrary rather than giving a set of limited operations, Ethereum allows developers to create whatever operations they want. This means developers can build thousands of different applications that go way beyond anything we have seen before.

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

One of the most significant token standards of all for Ethereum is called ERC-20, which was developed about a year and a half ago.

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.

ERC-20 defines six different functions for the benefit of other tokens within the Ethereum system. These are generally basic functionality issues, including how tokens are transferred and how users can access data about a token.

TenxPay Token

Tenx make cryptocurrencies spendable anytime, anywhere wallet app and debit card.TenX connects your blockchain assets for everyday use. TenX's debit card and banking licence will allow us to be a hub for the blockchain ecosystem to connect for real-world use cases.

Token TenXPay

TenxPay is sponsored by Crypto.com - Crypto Invest:

- Token: TenXPay
- Total Supply: 205,218,255.948577763364408207 PAY ($136,450,821.45)
- Decimals:18

Market

Volume (24H)	:	$522,372.89
Market Capitalization	:	$63,748,505.00
Circulating Supply	:	109,347,861.00 PAY
Market Data Source	:	Coinmarketcap

ICO Information

ICO Start Date	:	Jun 24, 2017
Total Raised	:	$80,000,000
ICO Price	:	$0.87 | 0.00285714 ETH
Country	:	China

- Date of token dataset: From 06/24/2017 To 05/06/2018 (mm/dd/yyyy)
- Columns in dataset:
  - fromNodeID
  - toNodeID
  - tunixTime
  - tokenAmount
- Total Tuples(rows) in the dataset: 329737

Outliers in TenxPay Token Dataset

The impact of the batchOverflow exploit was first made public by blockchain security startup PeckShield, who detected an unusual MESH token transaction on the 25th of April. According to PeckShield, the smart contract used in the exploit possesses a classic integer overflow problem.

- Outliers: Token Amount > Total Supply * 10^18
- Number of outliers found: 1 at row number 569 by UserId:- 81

Question 1: Find the distribution of how many times a user 1 - buys, 2 - sells a token. Which discrete distribution type fits these distributions best? Estimate distribution parameters.

Reading TenxPay Dataset

```
networktenx<-read.table(file="networktenxpayTX.txt",header=F,sep=" ")
```

Date of TenxPay Token dataset

```
max_date<-as.Date(as.POSIXct(max(networktenx$V3), origin="1970-01-01"))
```

```
min_date<-as.Date(as.POSIXct(min(networktenx$V3), origin="1970-01-01"))

max_date
```

Date of dataset:
Time period: 06-24-2017 to 05-06-2018

1]Buyers

In this step we are calculating the frequency of times a buyer has bought tokens in the time period.

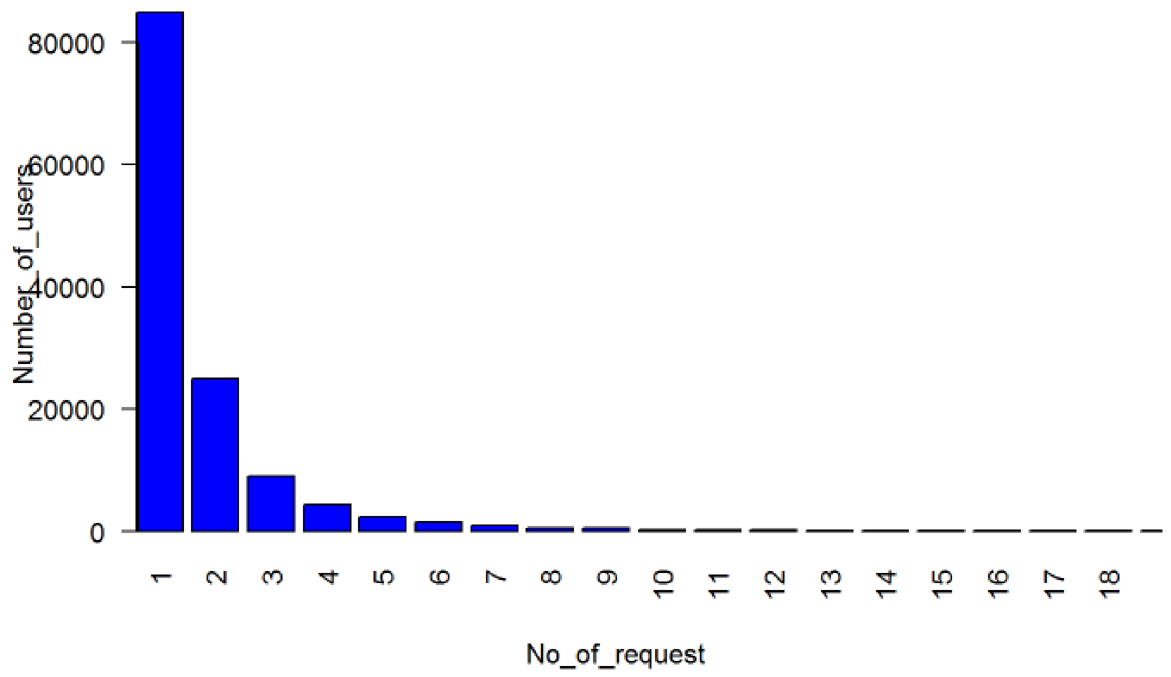
Step 1: Count how many times a user has bought tokens in the given time period (mentioned in date)

```
Buyer <-c(networktenx[,2])

count_buys<-as.data.frame(table(Buyer))

colnames(count_buys)<-c("UserId","Frequency")

summary(count_buys)
```

Step 2: Calculate how many times users have put a request to buy token in the given time period.

```
no_count_buys<-as.data.frame(table(count_buys$Frequency))

colnames(no_count_buys)<-c("No_of_request","Number_of_users")


summary(no_count_buys)
```

The Graph is for Number of requests which are less than 20.



2]Sellers

In this step we are calculating the frequency of times a seller has sold tokens in the time period.

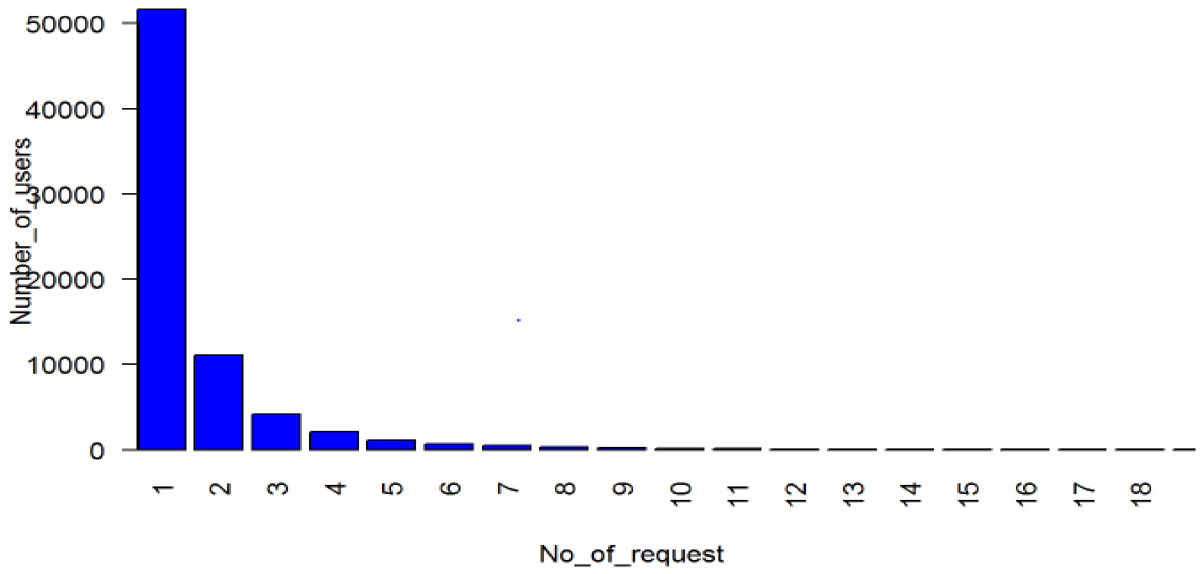
Step 1: Count how many times a user has sold tokens in the given time period (mentioned in date) Count the number of times a user node Id have appeared in ToID column of our dataset.

```
Seller<-c(networktenx[,1])

count_sells<-as.data.frame(table(Seller))

colnames(count_sells)<-c("UserId","Frequency")


summary(count_sells)
```

Step 2: Calculate how many times users have put a request to buy token in the given time period.

```
no_count_sells<-as.data.frame(table(count_sells$Frequency))

colnames(no_count_sells)<-c("No_of_request","Number_of_users")

summary(no_count_sells)
```

The Graph is for Number of requests which are less than 20.



Fitting Discrete Distribution

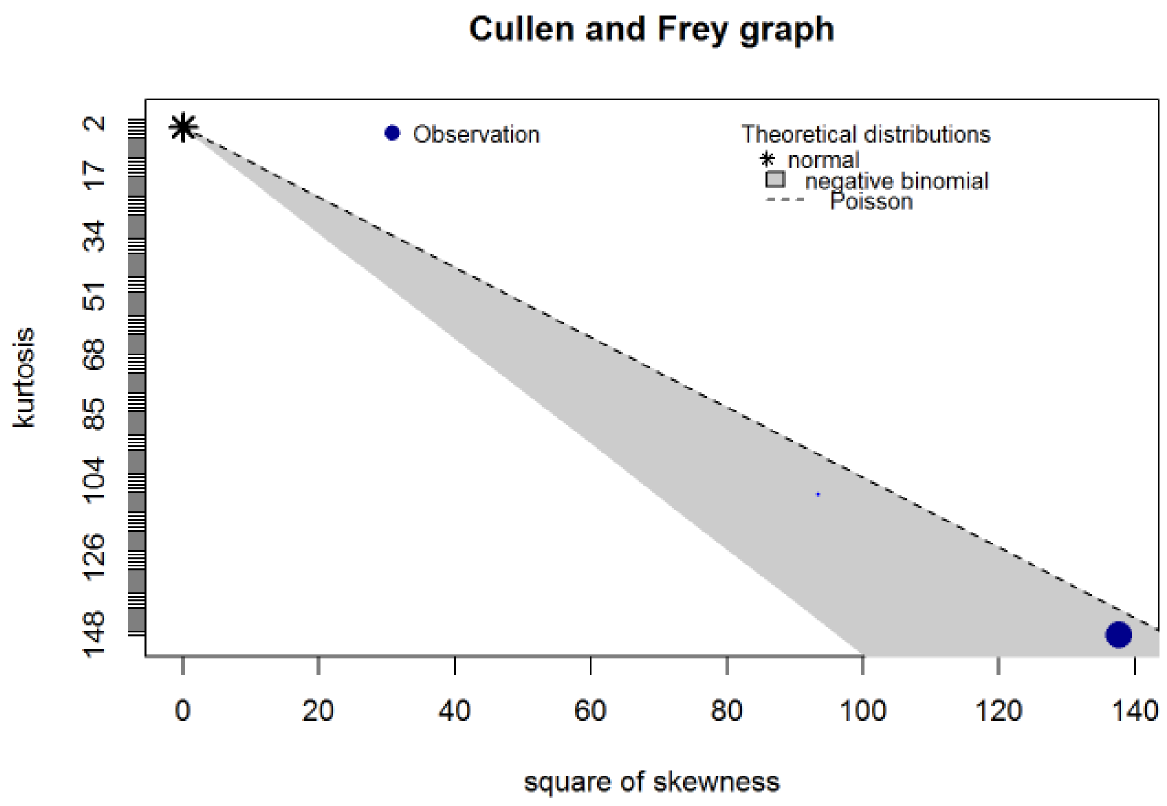
We have to find a discrete distribution that fits our distribution for Number of request vs Number of users for both buyers and sellers.

We use library fitdistrplus to find the discrete distribution that fits best to our distribution of dataset.

1. Buyers

We have used Cullen and Frey Graph to check which one of these distributions best describes our distribution for buyers. We have fit normal, negative binomial and Poisson distribution to our distribution of dataset.

```
descdist(no_count_buys[,2], discrete = TRUE)
```

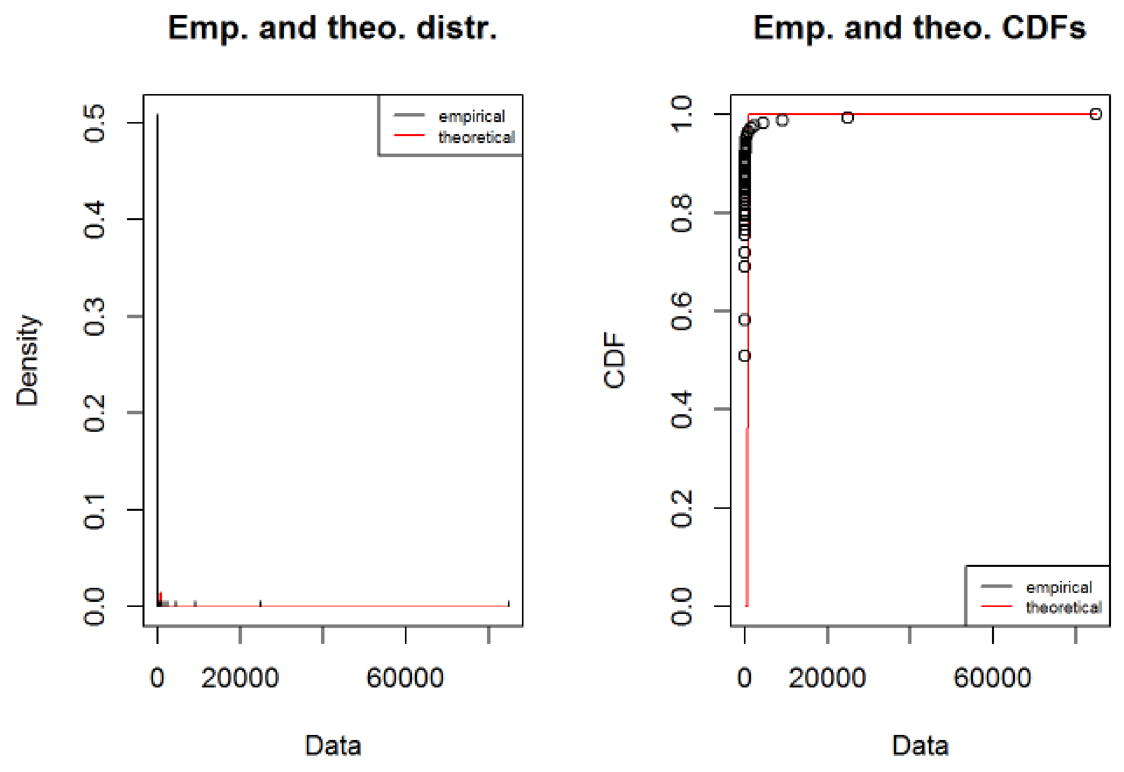


From the above graph we conclude that Poisson distribution is a best fit for our distribution.

Fitting Poisson Distribution

```
No.pois<-fitdist(no_count_buys[,2], distr = "pois")
plot(No.pois)
```

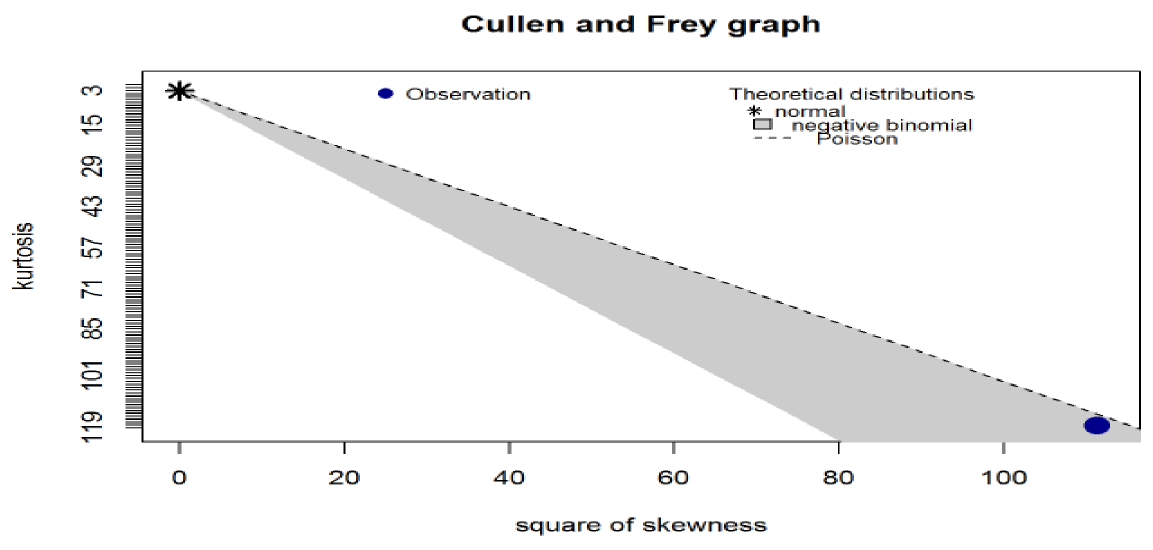
Token TenXPay



2. Sellers

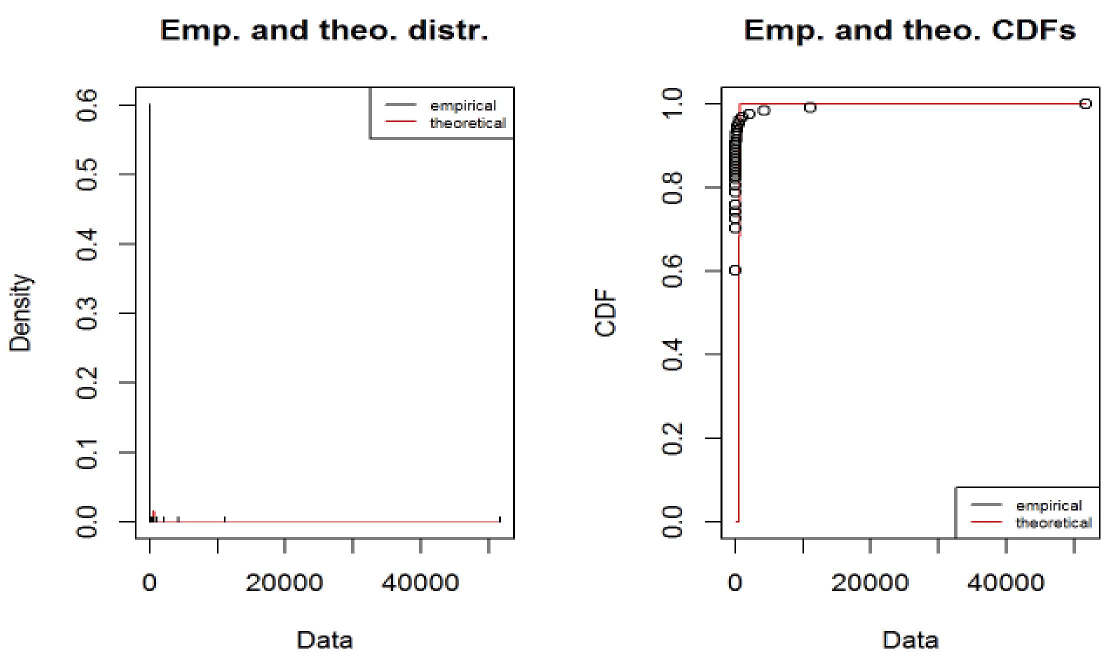
We have used Cullen and Frey Graph to check which one of these distributions best describes our distribution for sellers. We have fit normal, negative binomial and Poisson distribution to our distribution of dataset.

```
descdist(no_count_sells[,2], discrete = TRUE)
```



From the above graph we conclude that Poisson distribution is a best fit for our distribution.

```
No.pois<-fitdist(no_count_sells[,2], distr = "pois")

plot(No.pois)
```



The parameters for Poisson distribution are mean and variance.

In Poisson distribution mean is equal to variance.

1. Mean and Variance of Number of request for buys λ=748.2
2. Mean and Variance of Number of request for sells λ=567.2

Feature Analysis

In this part we are trying to find the features in the transaction data of all the user. The feature we selected is the amount of transaction that has taken place. To find a better correlation, we are dividing amount of transactions into layers.

Data Pre-processing:

- In the transactions data set we remove the Outliers that transactions involving more than $2.05218256 \times 10^{26}$ Tokens
- Convert Unix date format to 'CT' date format

- Takin only required price (Opening or Low) from the price Data
- Changing the format of price data from mm/dd/YYYY to YYYY-mm-dd

Steps:

1. Find the frequency of transaction based on the date.
2. Join Price table and Frequency table on Date.
3. Find and plot Pearson correlation.

```
networktenx<-read.table(file="networktenxpayTX.txt",header=F,sep=" ")

networktenx <- networktenx[ which(networktenx$V4<=2.05218256e+26), ]

colnames(networktenx)<-c("FromId","ToId","date","TotalAmount")

summary(networktenx)


networktenx$date<- as.Date(as.POSIXct(networktenx$date, origin="1970-01-01"))


pricetenx<-read.table(file="tenxpay.txt",header=T,sep="\t")

summary(pricetenx)


pricetenx<-pricetenx[c(1,2)]

pricetenx$Date<-as.Date(pricetenx$Date,format="%m/%d/%Y")

colnames(pricetenx)<-c("Date","Price")
```

Layers:

The transactions are divided into layers based on number of tokens that are involved in the transactions. For example, layer 1 involves transactions which has more that 10^{24} Tokens that are transferred from buyer to the seller.
In our project we are dividing the data into 7 layers as follows: x denotes the number of tokens involved in the transaction

- Layer 1: $x \geq 10^{24}$
- Layer 2: $10^{20} \leq x < 10^{24}$
- Layer 3: $10^{18} \leq x < 10^{20}$
- Layer 4: $10^{15} \leq x < 10^{18}$
- Layer 5: $10^{10} \leq x < 10^{15}$
- Layer 6: $10^{3} \leq x < 10^{10}$
- Layer 7: $10^{0} \leq x < 10^{3}$

```
layer_1<-networktenx[which(networktenx$TotalAmount>=1e+24),]

layer_2<-networktenx[which(networktenx$TotalAmount>=1e+20 & networktenx$TotalAmount< 1e+24 )
,]

summary(layer_2)


layer_3<-networktenx[which(networktenx$TotalAmount>=1e+18 & networktenx$TotalAmount< 1e+20 )
,]

layer_4<-networktenx[which(networktenx$TotalAmount>=1e+15 & networktenx$TotalAmount< 1e+18 )
,]

layer_5<-networktenx[which(networktenx$TotalAmount>=1e+10 & networktenx$TotalAmount< 1e+15 )
,]

layer_6<-networktenx[which(networktenx$TotalAmount>=1e+3 & networktenx$TotalAmount< 1e+10 ),]

layer_7<-networktenx[which(networktenx$TotalAmount>=1e+0 & networktenx$TotalAmount< 1e+3),]
```

Pearson Correlation:

A Pearson correlation is a number between -1 and 1 that indicates the extent to which two variables are linearly related. The Pearson correlation is also known as the “product moment correlation coefficient” (**PMCC**) or simply “**correlation**”. In our project we are using Pearson correlation to find the correlation between the amount of transaction (frequency) on a particular day against the price of the token that day.

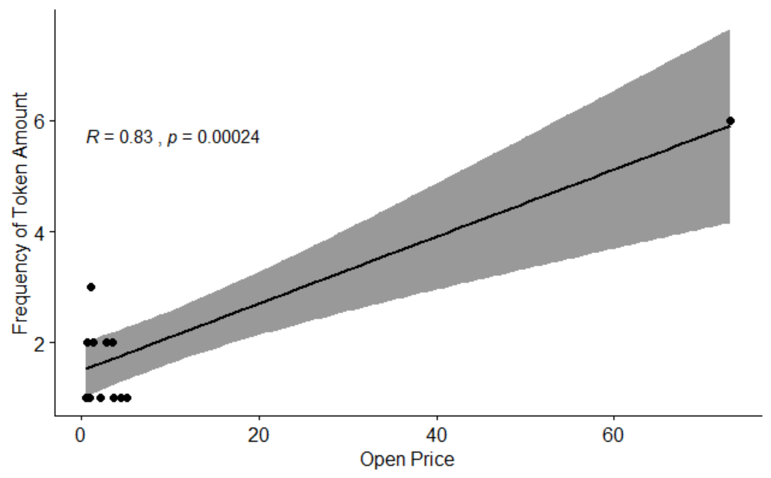
Results:

1. **Correlation for layer One using Opening price as price for that date:**

```
Pearson's product-moment correlation

data:  df3$Price and df3$Frequency
t = 5.1544, df = 12, p-value = 0.0002392
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
0.5349867 0.9445884
sample estimates:
cor 0.8299753
```

```
Flayer_1 <-c(layer_1[,3])

summary(Flayer_1)

count_layer1<-as.data.frame(table(Flayer_1))

colnames(count_layer1)<-c("Date","Frequency")

nrow(count_layer1)

library(anytime)

count_layer1$Date<- anydate(count_layer1$Date)

library(sqldf)

df3<-sqldf("SELECT f.Date, p.Price, f.Frequency FROM count_layer1 f INNER JOIN pricetenx p WHERE f.Da
te=p.Date")

cor.test(df3$Price, df3$Frequency,

                method = "pearson")
```

Plot of Pearson of Layer 1:



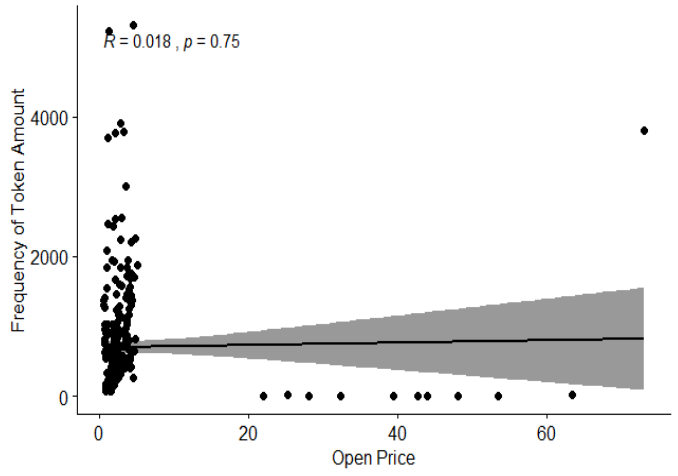
```
library("ggpubr")

ggscatter(df3, x = "Price", y = "Frequency",

        add = "reg.line", conf.int = TRUE,

        cor.coef = TRUE, cor.method = "pearson",

        xlab = "Open Price", ylab = "Frequency of Token Amount")
```

Correlation for rest of the layers:

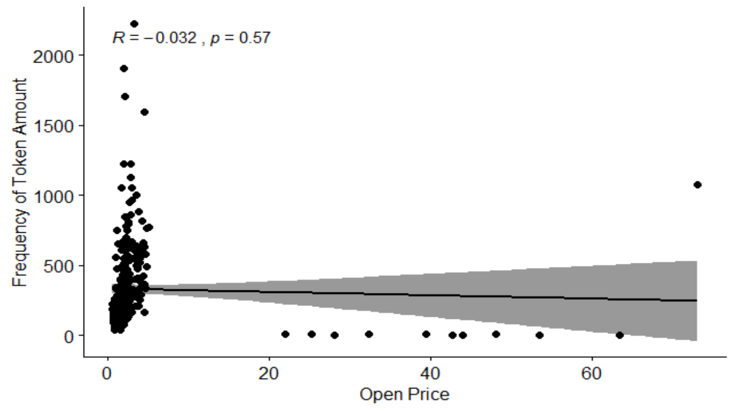
Layer	Correlation	P-value
1	0.83	0.00024
2	0.183	0.7466
3	-0.032	0.5676
4	0.18	0.0017
5	-0.206	0.072
6	-0.18	0.55
7	0.074	0.48

Plot of all the layers:

Layer2:

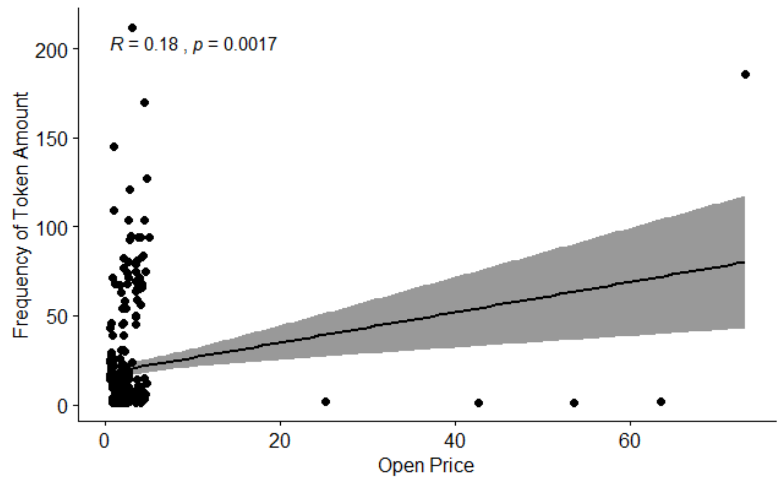


Layer 3:

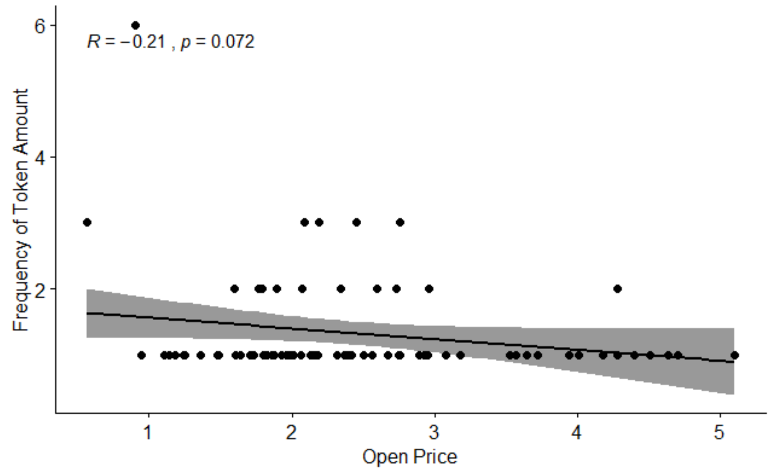


Token TenXPay

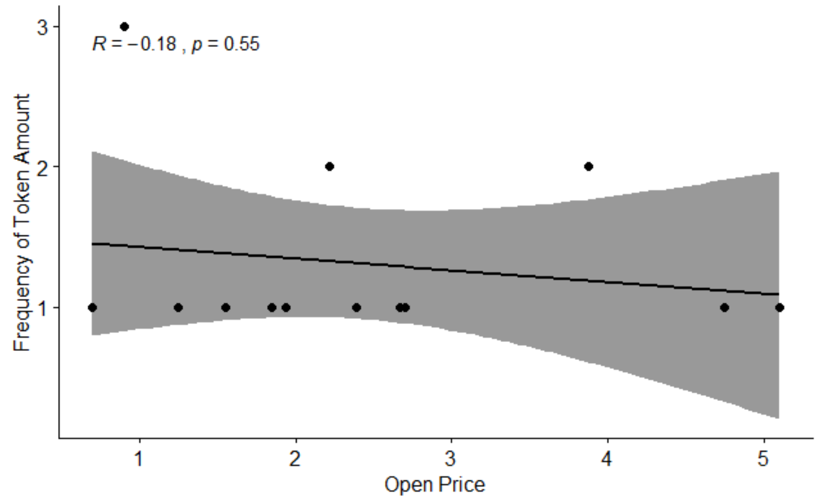
Layer 4:



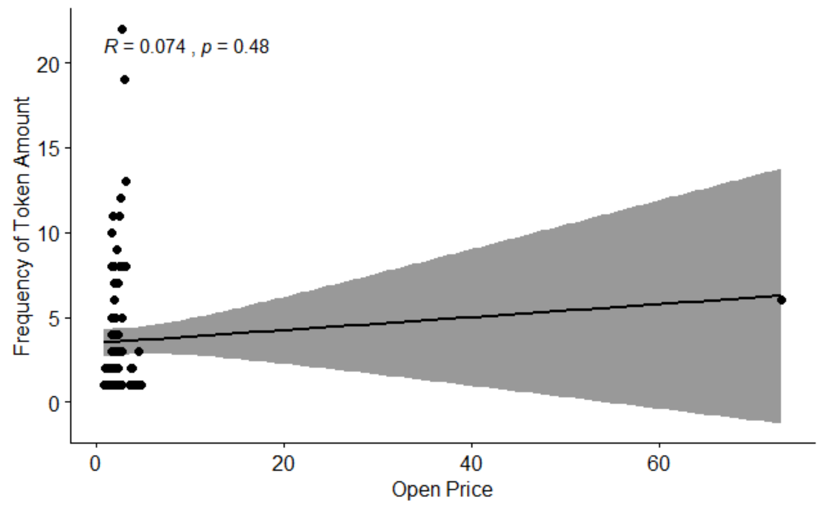
Layer 5:



Layer6:



Layer 7:



Next we take low as the price of the day:

```
lpricetenx<-read.table(file="tenxpay.txt",header=T,sep="\t")

summary(pricetenx)

pricetenx<-pricetenx[c(1,4)]

pricetenx$Date<-as.Date(pricetenx$Date,format="%m/%d/%Y")

colnames(pricetenx)<-c("Date","Price")
```

```
        Date            Open             High              Low        
 01/01/2018:  1   Min.   : 0.4862   Min.   : 0.5016   Min.   : 0.4667  
 01/02/2018:  1   1st Qu.: 1.1025   1st Qu.: 1.1875   1st Qu.: 1.0057  
 01/03/2018:  1   Median : 1.7100   Median : 1.8200   Median : 1.6000  
 01/04/2018:  1   Mean   : 3.0674   Mean   : 3.5908   Mean   : 2.4626  
 01/05/2018:  1   3rd Qu.: 2.4500   3rd Qu.: 2.6425   3rd Qu.: 2.2225  
 01/06/2018:  1   Max.   :73.0600   Max.   :86.2600   Max.   :49.0500  
 (Other)   :378                                                        
     Close               Volume          Market.Cap 
 Min.   : 0.4866   1,064,690:  1   -          : 12  
 1st Qu.: 1.0775   1,104,700:  1   100,064,000:  1  
 Median : 1.7100   1,105,650:  1   101,440,000:  1  
 Mean   : 2.9722   1,214,640:  1   101,797,000:  1  
 3rd Qu.: 2.4275   1,288,170:  1   102,011,000:  1  
 Max.   :63.8300   1,299,010:  1   102,404,000:  1  
                   (Other)  :378   (Other)    :367  
```

Correlation for All the layers with Low as the price of the day:

Correlation Table

Layer No	**Correlation**	**P-value**
1	-0.33	0.25
2	-0.1	0.071
3	-0.1	0.068
4	-0.0027	0.96
5	-0.24	0.035
6	-0.22	0.47
7	0.084	0.42

Result Discussion:

1. The layer 1 gives us the best correlation of all the layers for which the price is opening price. It has a correlation of 0.83 which tells us that the as the price increases more transactions are taking place and the transactions involves the highest number of tokens.
2. For all the other layer the correlation is significantly lower. which implies the price is not affecting the transactions as greatly as it does in layer one.
3. When we take low price as the price for the day, we are getting negative correlations in all the layers except the layer 7 which has very less correlation. We can conclude for this observation that as the price decreases the amount of transactions that are taking place increases. Hence, more people buy or sell at lower prices.

Multiple Regression

Regression is a technique in statistics which is used to predict the value of certain variable based on the Other variable. Multiple Regression is an extension of Linear Regression model. Instead on using just a single variable, in multiple regression we used more than two variables to predict a Variable. The variable we want to predict is called dependent variable and the variable which are used to predict are called Independent variable.

In regression we observe how the independent variables affect the depended variable and create a function for this. Then to predict the value of depended variable we use this function. We can only extrapolate the value of the dependent variable. That is the values can be predicted within the given only. If we try to used Regression to interpolate the values, we often end up getting values which cannot be guaranteed to be correct. This happens, since we do not know how the independent variable affects the dependent variable outside the observed range.

Token TenXPay

In this Project we are going to Multi Linear Regression to predict price of the TenxPay token.

Reading TenxPay Dataset

```
networktenx<-read.table(file="networktenxpayTX.txt",header=F,sep=" ")

networktenx <- networktenx[ which(networktenx$V4<=2.05218256e+26), ]

colnames(networktenx)<-c("FromId","ToId","Date","TotalAmount")

networktenx$Date<- as.Date(as.POSIXct(networktenx$Date, origin="1970-01-01"))

networktenx[order(networktenx$Date),]

summary(networktenx)
```

Output:

```
    FromId              ToId                 Date              TotalAmount
 Min.   :      5   Min.   :      5   Min.   :2017-06-24   Min.   :1.000e+00
 1st Qu.:     17   1st Qu.: 431373   1st Qu.:2017-08-15   1st Qu.:5.445e+19
 Median : 297031   Median :2062570   Median :2017-09-17   Median :2.656e+20
 Mean   :1644851   Mean   :3179541   Mean   :2017-10-16   Mean   :1.844e+21
 3rd Qu.:1943406   3rd Qu.:6393476   3rd Qu.:2017-12-15   3rd Qu.:1.005e+21
 Max.   :6438429   Max.   :6438432   Max.   :2018-05-06   Max.   :7.108e+24
```

Reading Price Data

```
pricetenx<-read.table(file="tenxpay.txt",header=T,sep="\t")

summary(pricetenx)



pricetenx$Date<-as.Date(pricetenx$Date,format="%m/%d/%Y")

pricetenx<-pricetenx[c(1,3,5)]

colnames(pricetenx)<-c("Date","High","Close")

pricetenx[order(pricetenx$Date),]

summary(pricetenx)
```

Token TenXPay

Output:

```
        Date           Open              High              Low
Close
 01/01/2018:  1   Min.   : 0.4862   Min.   : 0.5016   Min.   : 0.4667   Min.
: 0.4866
 01/02/2018:  1   1st Qu.: 1.1025   1st Qu.: 1.1875   1st Qu.: 1.0057   1st
Qu.: 1.0775
 01/03/2018:  1   Median : 1.7100   Median : 1.8200   Median : 1.6000
Median : 1.7100
 01/04/2018:  1   Mean   : 3.0674   Mean   : 3.5908   Mean   : 2.4626   Mean
: 2.9722
 01/05/2018:  1   3rd Qu.: 2.4500   3rd Qu.: 2.6425   3rd Qu.: 2.2225   3rd
Qu.: 2.4275
 01/06/2018:  1   Max.   :73.0600   Max.   :86.2600   Max.   :49.0500   Max.
:63.8300
 (Other)   :378
       Volume          Market.Cap
 1,064,690:  1   -           : 12
 1,104,700:  1   100,064,000:  1
 1,105,650:  1   101,440,000:  1
 1,214,640:  1   101,797,000:  1
 1,288,170:  1   102,011,000:  1
 1,299,010:  1   102,404,000:  1
 (Other)  :378   (Other)    :367
      Date                High             Close
 Min.   :2017-06-27   Min.   : 0.5016   Min.   : 0.4866
 1st Qu.:2017-09-30   1st Qu.: 1.1875   1st Qu.: 1.0775
 Median :2018-01-04   Median : 1.8200   Median : 1.7100
 Mean   :2018-01-04   Mean   : 3.5908   Mean   : 2.9722
 3rd Qu.:2018-04-10   3rd Qu.: 2.6425   3rd Qu.: 2.4275
 Max.   :2018-07-15   Max.   :86.2600   Max.   :63.8300
```

Normalizing High and Close from tenxPay

We are selecting High Price of day as the dependent variable and Close price of the day as a independent variable. In this step we are normalizing the variable values for simplicity purpose.

```
Buyer <-c(networktenx[,2])

library(dplyr)

 pricetenx<- pricetenx %>%

   mutate(norm_high = (High-lag(High))/lag(High)) %>% as.data.frame %>%

  mutate(norm_close = (Close-lag(Close))/lag(Close)) %>% as.data.frame %>%

setNames(c("Date","High","Close","norm_high","norm_close"))
```

Selecting Number of transactions as a feature

In this step we are finding the number of transactions per day and normalizing them to for using them in Regression model.

```
Flayer_1 <-c(networktenx[,3])

summary(Flayer_1)

count_layer<-as.data.frame(table(Flayer_1))

colnames(count_layer)<-c("Date","Frequency")


library(anytime)

count_layer$Date<- anydate(count_layer$Date)

count_layer<- count_layer %>%

   mutate(norm_frequency = (Frequency-lag(Frequency))/lag(Frequency)) %>% as.data.frame %>%

setNames(c("Date","Frequency","norm_frequency"))
```

Select Total amount of token for a date as the third Feature.

In this step we are finding the total amount of tokens involved in transaction for a day. We are normalizing this for using it in regression model.

```
Total_Amount<-aggregate(networktenx$TotalAmount, by=list(Date=networktenx$Date), FUN=sum)


Total_Amount

 Total_Amount<- Total_Amount %>%

   mutate(norm_frequency = (x-lag(x))/lag(x)) %>% as.data.frame %>%

setNames(c("Date","Total_Amount","norm_Total"))
```

Joining all the feature into a single data set.

```
library(sqldf)

df1<-sqldf("SELECT f.Date, p.norm_high,p.norm_close,f.norm_frequency FROM count_layer f INNER JOI
N pricetenx p WHERE f.Date=p.Date ")


df2<-sqldf("SELECT p.Date, p.norm_high,p.norm_close,p.norm_frequency,t.norm_Total FROM df1 p INN
ER JOIN Total_Amount t WHERE p.Date=t.Date ")
```

Token TenXPay

Creating a Regression model using Close price , Total number of transactions and Total amount of tokens as features and using them to predict dependent variable High price for the day.

```
Regression_tenxpay<-lm(norm_high ~ norm_close + norm_frequency + norm_Total, data=df2)

Regression_tenxpay

summary(Regression_tenxpay)
```

Output:

```
Call:
lm(formula = norm_high ~ norm_close + norm_frequency + norm_Total, 
    data = df2)

Coefficients:
   (Intercept)      norm_close  norm_frequency      norm_Total  
     -0.015992       -0.001742        0.153067        0.043051  
Call:
lm(formula = norm_high ~ norm_close + norm_frequency + norm_Total, 
    data = df2)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.83067 -0.05491  0.02094  0.07981  0.35694 

Coefficients:
                Estimate Std. Error t value Pr(>|t|)    
(Intercept)    -0.015992   0.008187  -1.953   0.0517 .  
norm_close     -0.001742   0.002540  -0.686   0.4933    
norm_frequency  0.153067   0.012096  12.655   <2e-16 ***
norm_Total      0.043051   0.004530   9.503   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1444 on 310 degrees of freedom
Multiple R-squared:  0.9981,	Adjusted R-squared:  0.9981 
F-statistic: 5.518e+04 on 3 and 310 DF,  p-value: < 2.2e-16
```

Result Discussion:

1. The intercept is -0.01592 , the co-efficient for Norm_close is -0.001742, Number of transactions(Norm_frequency) is 0.153067 and total amount of tokens per day (norm_total) 0.043051.
2. This gives the Formula “norm_High= (-0.01592) + (-0.001742)*norm_Close + 0.153067* norm_Frequnecy + 0.043051*norm_Total.
3. This show than the High price decreases by -0.001742 for one percent change in close price.
4. High price increases by 0.153067 for one percent increases in total number of transactions.

5. High price increases by 0.043051 for one percent change in total amount of tokens involved in transaction per day .
6. The R squared value is 0.9981. This shows that the variables are fitted very closely to the regression line.

Referrence:

- https://www.r-bloggers.com/r-tutorial-series-multiple-linear-regression/
- https://statistics.laerd.com/spss-tutorials/multiple-regression-using-spss-statistics.php
- https://etherscan.io/
- https://coinmarketcap.com/
- https://cointelegraph.com/explained/erc-20-tokens-explained
- https://www.investinblockchain.com/what-are-ethereum-tokens/
- https://arxiv.org/abs/1708.08749