CS 6313.002 - Statistical Methods for Data Science

Ethereum project

Maryam Bahojb Imani (mxb154330)
Azadeh Samadian (axs168931)

Fall 2018

Repo: https://github.com/Maryam-Imani/BlockChain-Regression
Repo: https://github.com/AzadehS/BlockChain

Introduction

The Ethereum project is a blockchain platform. Based on our UTD IDs we chose Omisego dataset. The data file contains two primary groups: token network edge files, and token price files.
Token edge files have this row structure: fromNodeID\ttoNodeID\tunixTime\ttokenAmount\r\n

This row implies that fromNodeID sold tokenAmount of the token to toNodeID at time unixTime. fromNodeID and toNodeID are people who invest in the token in real life; each investor can also use multiple addresses. Two addresses can sell/buy tokens multiple times with multiple amounts. For this reason, the network is considered a weighted, directed multi(edge) graph. Each token has a maximum token count maxt; you can think of maxt as the total circulating token amount.

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.

Data pre-processing:

The first step in data preprocessing is removing the outliers. For Omisego dataset we had get maximum total supply and sub units and on them outcome we find our data outliers. Based on outlier value we find the Number of users that included in outliers' transactions and then remove outliers.

Implementation and Results:

The first step for finding the distribution of buyers and sellers, we found their frequency and based on that we tried different distribution that some of them are mentioned below:

- Normal Distribution
- Exponential Distribution
- Testing Exponential Distribution
- Log-Normal Distribution
- Testing Poisson Distribution
- Testing Negative Binomial Distribution
- Testing Weibull Distribution
- Comparison between different distributions

Finally, we conclude that both of buyers and seller's frequency can fit with **Exponential** distributions.

Question 2:

How can we create layers of transactions with increasing amounts? Find an algorithm to compute the correlation of price data with each of the layers?

For question 2 we used token network edge files, and token price files. Same as question 1

We defined limits for outliers and find number of users that included in outliers' transactions:

Finding layers and correlations

we normalize the data by dividing them to the minimum. Initially we assumed that we have pre-defined number of iterations (or layers) and then during the procedure this dynamically is changing until getting the best correlation. Number of layers become cumulative. Then by merging two datasets together we find closing amount in a given date and later on finding the correlation between closing amount and frequency of transaction per date. Number of layers is changing in the code to get the highest correlation.

Project1- Question 1 Implementation and Results:

Reading data from the file

```
mydata <- read.table("token/networkomisegoTX.txt", header=FALSE,
    sep=" ",  col.names= c("fromNodeID", "toNodeID", "unixTime",
"tokenAmount"), colClasses=c('factor', 'factor', 'numeric', 'numeric') )
summary(mydata)
```

```
##    fromNodeID        toNodeID         unixTime          tokenAmount
##  17     :204064   297278 : 64690   Min.   :1.499e+09   Min.   :1.000e+00
##  297278 : 52534   5      : 57315   1st Qu.:1.506e+09   1st Qu.:1.100e+19
##  5      : 48528   311608 : 33408   Median :1.512e+09   Median :5.982e+19
##  311608 : 38011   36161  : 23013   Mean   :1.512e+09   Mean   :3.522e+71
##  36161  : 32247   75994  : 11173   3rd Qu.:1.517e+09   3rd Qu.:2.666e+20
##  13     : 21818   297094 :  7951   Max.   :1.526e+09   Max.   :1.158e+77
##  (Other):753535   (Other):953187
```

Outlier

```
maxTotalSupply<-140245398
subUnit<-10^18
outlier_value <- maxTotalSupply*subUnit
outlierData <- mydata [which(mydata$tokenAmount > outlier_value),]
#head(outlierData)
#library("dplyr")
#filter(mydata, tokenAmount < outlier_value)
message("Total Number of Outliers: ", length(outlierData$tokenAmount))
```

```
## Total Number of Outliers: 11
```

Number of users that included in outliers' transactions:

```
users <- c(outlierData$fromNodeID, outlierData$toNodeID)
uniqueUsers <- unique(users)
message(length(uniqueUsers), " users are included in outliers transactions.")
```

```
## 11 users are included in outliers transactions.
```

Remove outliers:

```
withoutOutlierData <- mydata [which(mydata$tokenAmount < outlier_value),]
```

Sellers:

Finding the frequency of each seller

```
w <- table(withoutOutlierData$fromNodeID)

Seller <- as.data.frame(w)

names(Seller)[1] <- 'Seller'
```

Finding the frequency of frequency of sellers

```
w2 <- table(Seller$Freq)

FreqSeller <-as.data.frame(w2)

colnames(FreqSeller) <- c("NumSeller", "FreqNumSeller")

head(FreqSeller)
```

```
##   NumSeller FreqNumSeller
## 1         0             1
## 2         1        194204
## 3         2         31348
## 4         3         11834
## 5         4          5924
## 6         5          3342
```

```
barplot(FreqSeller$FreqNumSeller, names.arg = FreqSeller$NumSeller, ylab =
"frequency of Number of Seller", xlab="Frequency", xlim = c(0, 50))
```

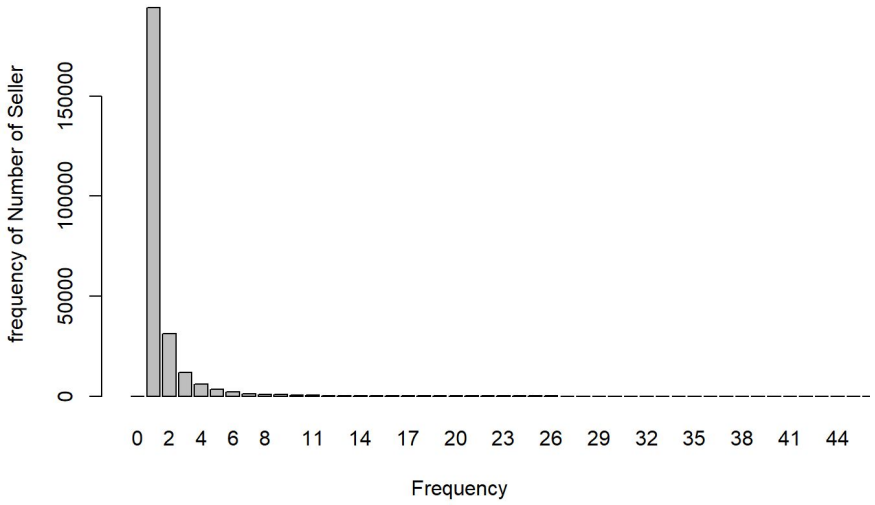

Fig 1. Frequency of number of sellers

Evaluating different distributions for sellers:

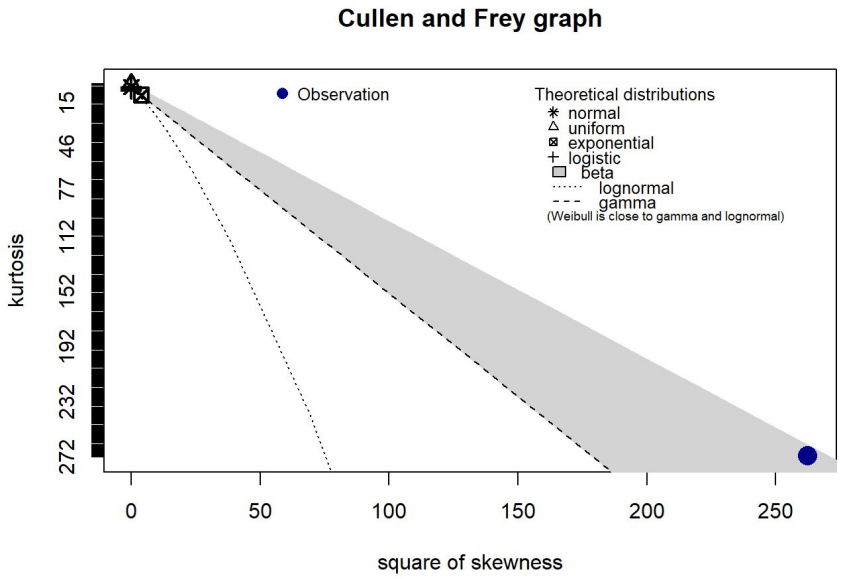

Fig 2. Evaluating different distribution for sellers

Testing Exponential Distribution

```
fit.exp <- fitdist(FreqSeller$FreqNumSeller, "exp", method="mme")
plot(fit.exp)
```

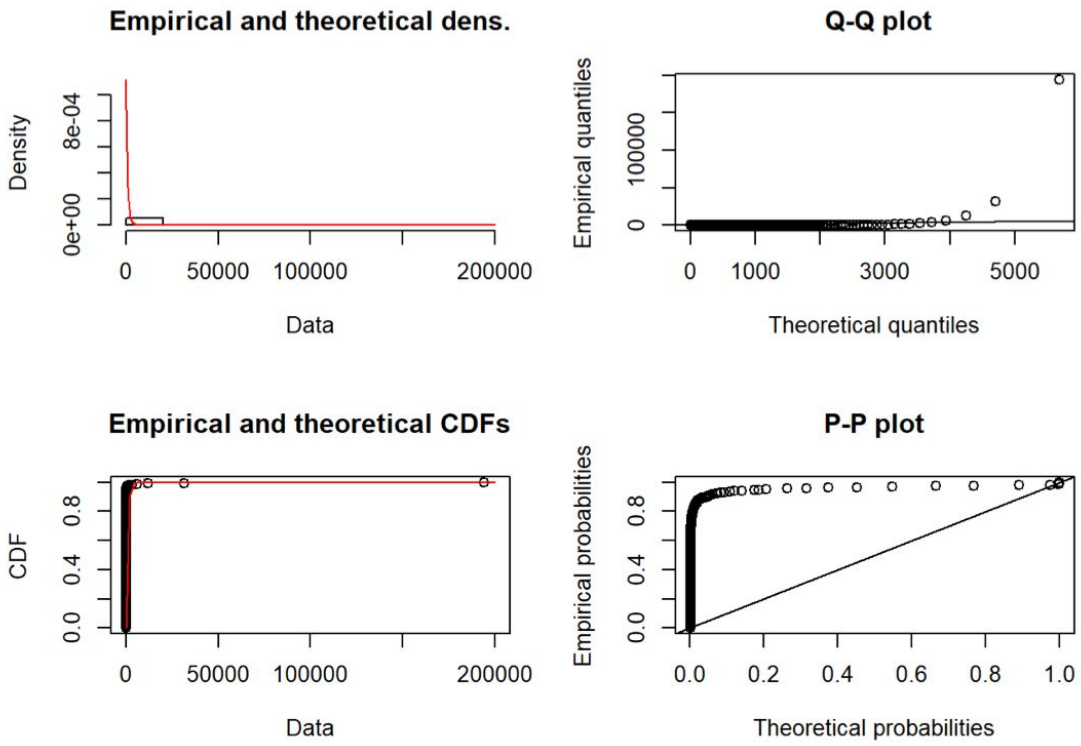

Fig 3. Exponential distribution for sellers

```
fit.exp
```

```
## Fitting of the distribution ' exp ' by matching moments
## Parameters:
##          estimate
## rate 0.001115434
```

Comparison between different distributions

```
par(mfrow=c(2,2))
plot.legend <- c("Weibull", "lognormal", "poisson")
denscomp(list(fit.weibull, fit.lnorm, fit.pois), legendtext = plot.legend)
cdfcomp (list(fit.weibull, fit.lnorm, fit.pois), legendtext = plot.legend)
qqcomp  (list(fit.weibull, fit.lnorm, fit.pois), legendtext = plot.legend)
ppcomp  (list(fit.weibull, fit.lnorm, fit.pois), legendtext = plot.legend)
```

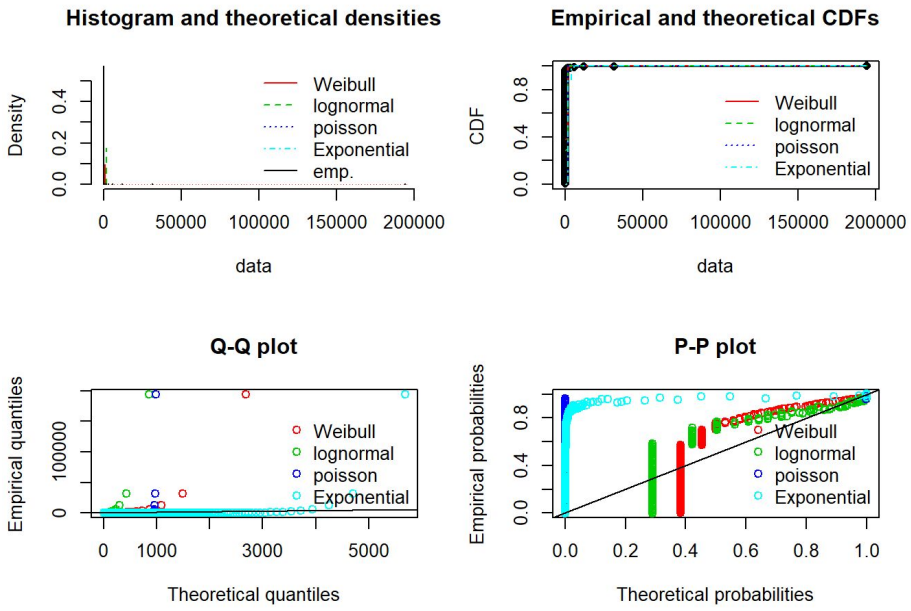

Fig 4.Comparison between different distributions for sellers

Buyers:

Finding the frequency of each buyer

For Buyer, we used the same code that we wrote for Seller but we replace “fromNodeID” with “toNodeID”.

Next plot shows the “Finding the frequency of frequency of buyers”.

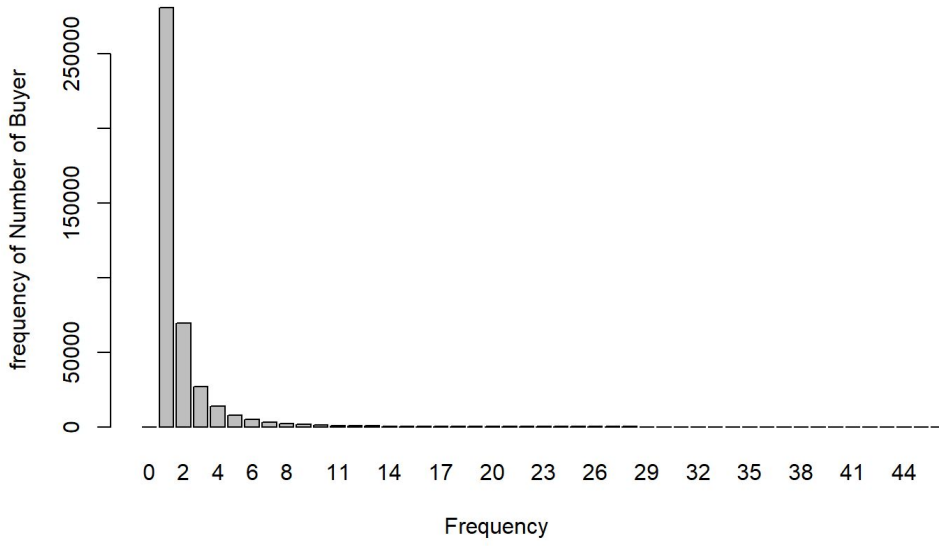

Fig 5. Frequency of number of buyers

Next plot present the evaluation of different distribution on the Buyer data.

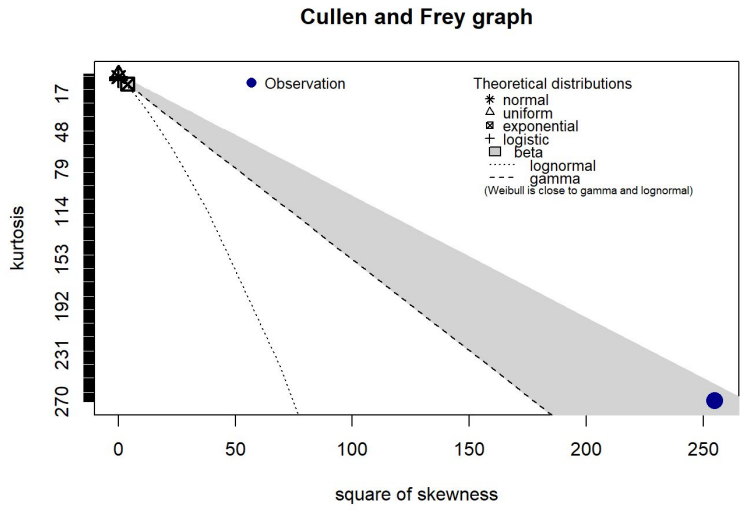

Fig 6. Evaluating different distribution for buyers

Testing Exponential Distribution for Buyers:

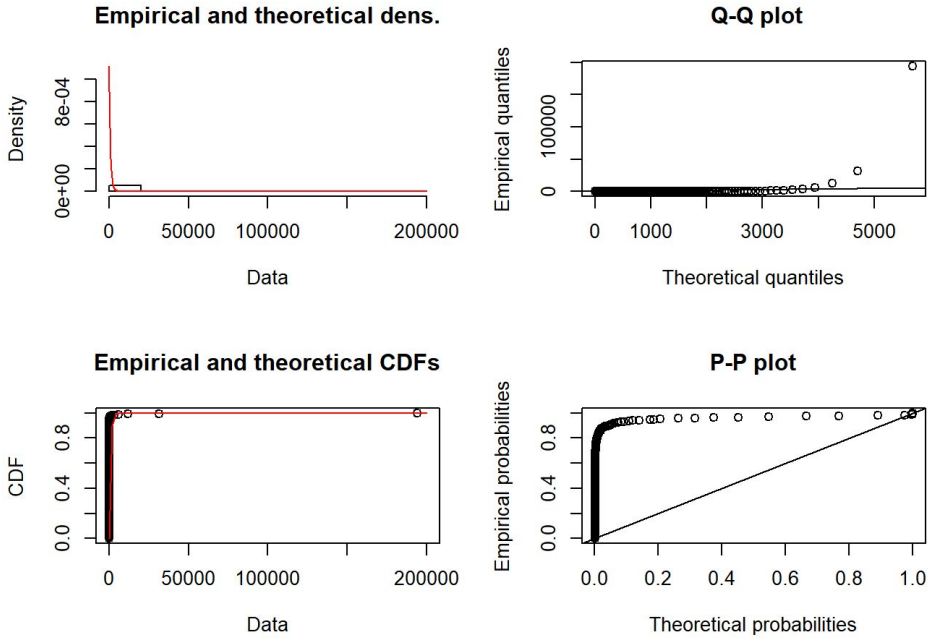

Fig 7. Exponential distribution for buyers

```
fit.exp
## Fitting of the distribution ' exp ' by matching moments
## Parameters:
##          estimate
## rate 0.000730197
```

We are going to compare different distributions and plot them in the following figure.

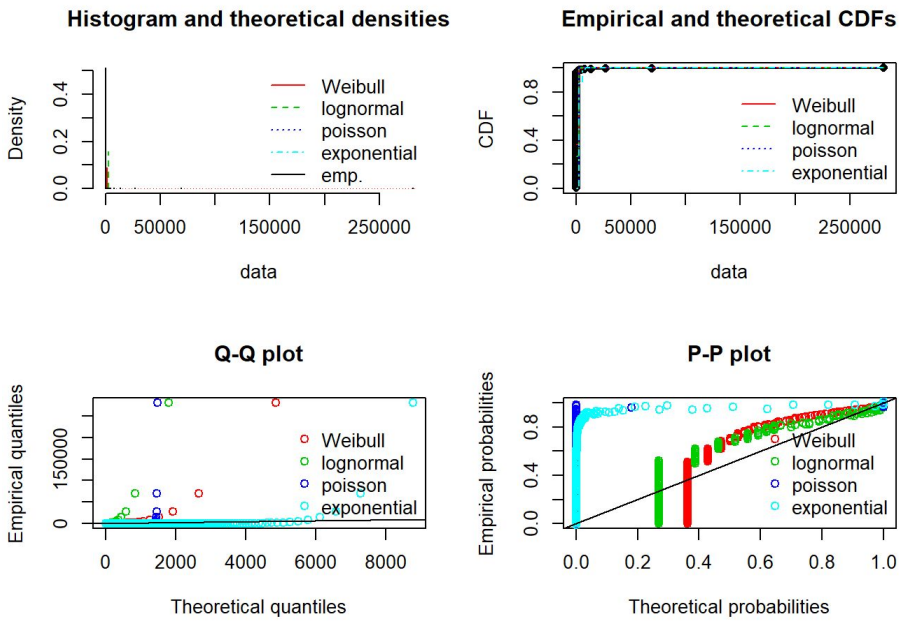

Fig 8.Comparison between different distributions for buyers

Based on the above plots, we believe both of buyers and seller's frequency can fit with **Exponential** distributions.

Project1- Question 2 Implementation and Results:

Finding layers and correlation

```
# In this part, we used frequncy of transaction per date as a feature. Then,
we found the corrolation of this feature and the amount of close date.

minimum <- min(cleanDF$tokenAmount, na.rm=TRUE)
maximum<-max(cleanDF$tokenAmount, na.rm=TRUE)

x2 <- minimum
layerNum <- 0
resultVec <- c()
```

```
xlabel <-c()

while ( x2*2<maximum) {

  x1 <- x2
  x2 <- x2 * 2

  df2 <- subset(cleanDF, cleanDF$tokenAmount >= x1 & cleanDF$tokenAmount < x2
, select=c(newDate))
  freqCountDF <- data.frame(table(df2$newDate))

  if(nrow(freqCountDF) > 2){

    layerNum <- layerNum +1

    # Counting the frequency of dates in the first dataset
    colnames(freqCountDF) <- c("newDate", "Freq")

    # merging two datasets together to find Closing amount in a given date
    mergeDF <- subset(merge(freqCountDF,df_set2,by="newDate"), select=c(Freq,
Close))

    # Counting the corrolation between closing amount and frequency of
transaction per date
    resultVec[layerNum] <- cor(mergeDF$Freq, mergeDF$Close, method = "pearson")

    xlabel[layerNum] <- paste0(layerNum , ": ", x1, "< tokenAmount < " , x2 , "
Cor=", resultVec[layerNum] )
  }
}
barplot( resultVec, names.arg=c(1:length(resultVec)), xlab = "Layer", ylab =
"Correlation")
```

To split data to layers, we exponentially increased the tokenAmount (base 2 and 10). Then, we found the correlation between number of transactions and close price for each layer.

Result for Correlation with Exponential Layering in base 2:

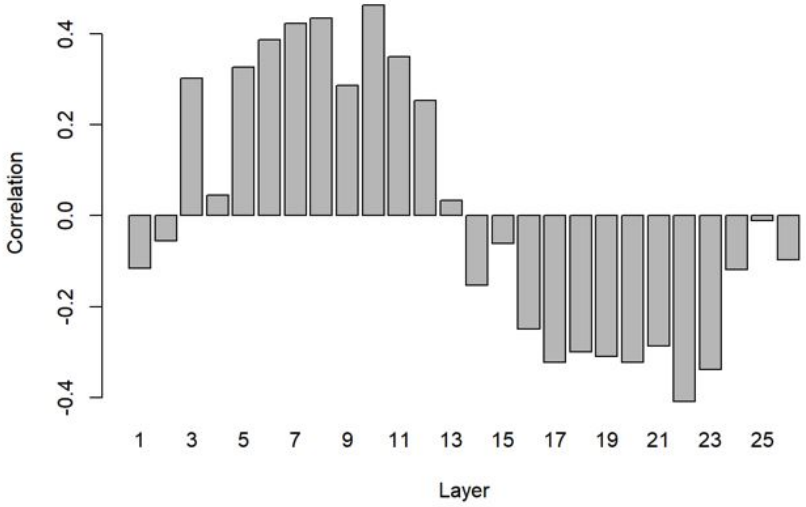

Fig 9.Correlation with Exponential layering in base 2

Layers:

```
##  [1] "1: 100000006845608944< tokenAmount < 200000013691217888  Cor=-0.116100863643567"
##  [2] "2: 200000013691217888< tokenAmount < 400000027382435776  Cor=-0.0557288774823974"
##  [3] "3: 400000027382435776< tokenAmount < 800000054764871552  Cor=0.302138235433703"
##  [4] "4: 800000054764871552< tokenAmount < 1600000109529743104  Cor=0.0456273227794848"
##  [5] "5: 1600000109529743104< tokenAmount < 3200000219059486208  Cor=0.326755681573848"
##  [6] "6: 3200000219059486208< tokenAmount < 6400000438118972416  Cor=0.38703462863633"
##  [7] "7: 6400000438118972416< tokenAmount < 12800000876237944832  Cor=0.422411645371166"
##  [8] "8: 12800000876237944832< tokenAmount < 25600001752475889664  Cor=0.433796847758056"
##  [9] "9: 25600001752475889664< tokenAmount < 51200003504951779338  Cor=0.285549696344644"
## [10] "10: 51200003504951779338< tokenAmount < 1.02400007009904e+20  Cor=0.462822532337046"
## [11] "11: 1.02400007009904e+20< tokenAmount < 2.04800014019807e+20  Cor=0.349847956963392"
## [12] "12: 2.04800014019807e+20< tokenAmount < 4.09600028039614e+20  Cor=0.253790716276259"
## [13] "13: 4.09600028039614e+20< tokenAmount < 8.19200056079228e+20  Cor=0.0339138122979369"
## [14] "14: 8.19200056079228e+20< tokenAmount < 1.63840011215846e+21  Cor=-0.152436779469996"
## [15] "15: 1.63840011215846e+21< tokenAmount < 3.27680022431691e+21  Cor=-0.0616918461148211"
## [16] "16: 3.27680022431691e+21< tokenAmount < 6.55360044863383e+21  Cor=-0.248742525921639"
## [17] "17: 6.55360044863383e+21< tokenAmount < 1.31072008972677e+22  Cor=-0.322478374376597"
## [18] "18: 1.31072008972677e+22< tokenAmount < 2.62144017945353e+22  Cor=-0.298807095923371"
## [19] "19: 2.62144017945353e+22< tokenAmount < 5.24288035890706e+22  Cor=-0.309385631722025"
## [20] "20: 5.24288035890706e+22< tokenAmount < 1.04857607178141e+23  Cor=-0.321971069245829"
## [21] "21: 1.04857607178141e+23< tokenAmount < 2.09715214356282e+23  Cor=-0.286866066848716"
## [22] "22: 2.09715214356282e+23< tokenAmount < 4.19430428712565e+23  Cor=-0.407786871633047"
## [23] "23: 4.19430428712565e+23< tokenAmount < 8.3886085742513e+23  Cor=-0.337914394891301"
```

[24] "24: 8.3886085742513e+23< tokenAmount < 1.67772171485026e+24 Cor=-0.118367678202037"
[25] "25: 1.67772171485026e+24< tokenAmount < 3.35544342970052e+24 Cor=-0.010770906533425"
[26] "26: 3.35544342970052e+24< tokenAmount < 6.71088685940104e+24 Cor=-0.0972982964469156"

Maximum Correlation : 0.46

Result for Exponential Layering in base 10:

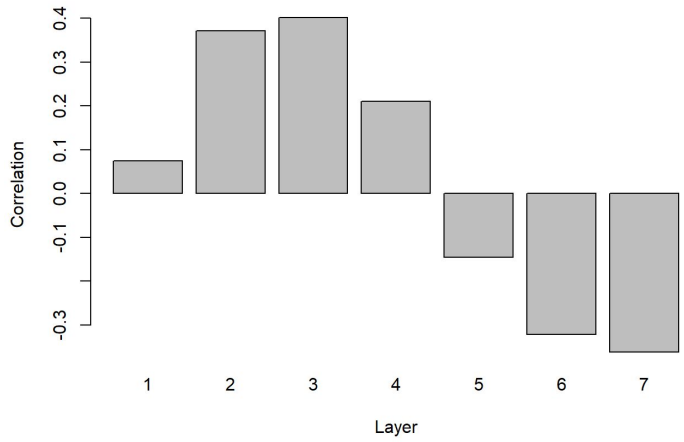

Fig 10.Correlation with Exponential layering in base 10

Layers:

[1] "1: 100000006845608944< tokenAmount < 1000000068456089472 Cor=0.075075650608101"
[2] "2: 1000000068456089472< tokenAmount < 10000000684560893952 Cor=0.371578156513923"
[3] "3: 10000000684560893952< tokenAmount < 1.00000006845609e+20 Cor=0.401935445687019"
[4] "4: 1.00000006845609e+20< tokenAmount < 1.00000006845609e+21 Cor=0.210405649338342"
[5] "5: 1.00000006845609e+21< tokenAmount < 1.00000006845609e+22 Cor=-0.145392515456215"
[6] "6: 1.00000006845609e+22< tokenAmount < 1.00000006845609e+23 Cor=-0.32138051807133"
[7] "7: 1.00000006845609e+23< tokenAmount < 1.00000006845609e+24 Cor=-0.361551696284235"

Maximum Correlation : 0.40

Cumulative Correlation

```
# In this part, we used frequncy of transaction per date as a feature. Then, we found the
corrolation of this feature and the amount of close date.

minimum <- min(cleanDF$tokenAmount, na.rm=TRUE)
maximum<-max(cleanDF$tokenAmount, na.rm=TRUE)
x2 <- minimum
layerNum <- 0
resultVec <- c()
xlabel <-c()

while ( x2*2<maximum) {

  x1 <- x2
  x2 <- x2 * 2

  df2 <- subset(cleanDF, cleanDF$tokenAmount < x2  , select=c(newDate))
  freqCountDF <- data.frame(table(df2$newDate))
```

```
  if(nrow(freqCountDF) > 2){
    layerNum <- layerNum + 1
    # Counting the frequency of dates in the first dataset
    colnames(freqCountDF) <- c("newDate", "Freq")
    # merging two datasets together to find Closing amount in a given date
    mergeDF <- subset(merge(freqCountDF,df_set2,by="newDate"), select=c(Freq, Close))
    # Counting the corrolation between closing amount and frequency of transaction per date
    resultVec[layerNum] <- cor(mergeDF$Freq, mergeDF$Close, method = "pearson")
    xlabel[layerNum] <- paste0(layerNum , ": tokenAmount < " , x2 , "  Cor=",
resultVec[layerNum] )
    print(xlabel[layerNum])
  }
}
barplot( resultVec, names.arg=c(1:length(resultVec)), xlab = "Layer", ylab = "Cumilitive
Correlation")
```

Result for Cumulative correlation with Exponential Layering in base 2:

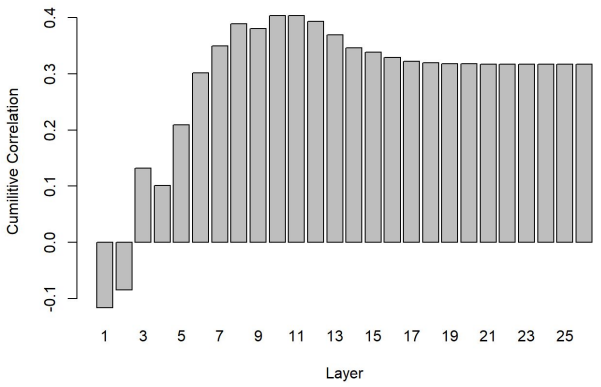

Fig 11.Cumulative correlation with Exponential layering in base 2

Layers:

```
##  [1] "1: tokenAmount < 200000013691217888  Cor=-0.116100863643567"
##  [2] "2: tokenAmount < 400000027382435776  Cor=-0.0850741308061474"
##  [3] "3: tokenAmount < 800000054764871552  Cor=0.131722028843304"
##  [4] "4: tokenAmount < 1600000109529743104  Cor=0.101066035649462"
##  [5] "5: tokenAmount < 3200000219059486208  Cor=0.208964968403616"
##  [6] "6: tokenAmount < 6400000438118972416  Cor=0.301620786708831"
##  [7] "7: tokenAmount < 12800000876237944832  Cor=0.349834229841956"
##  [8] "8: tokenAmount < 25600001752475889664  Cor=0.389484737786567"
##  [9] "9: tokenAmount < 51200003504951779338  Cor=0.380761766543603"
## [10] "10: tokenAmount < 1.02400007009904e+20  Cor=0.403701274191846"
## [11] "11: tokenAmount < 2.04800014019807e+20  Cor=0.403308291980739"
## [12] "12: tokenAmount < 4.09600028039614e+20  Cor=0.393680618802122"
## [13] "13: tokenAmount < 8.19200056079228e+20  Cor=0.369129463646885"
## [14] "14: tokenAmount < 1.63840011215846e+21  Cor=0.346626582964004"
## [15] "15: tokenAmount < 3.27680022431691e+21  Cor=0.338111286992914"
## [16] "16: tokenAmount < 6.55360044863383e+21  Cor=0.328992117375059"
## [17] "17: tokenAmount < 1.31072008972677e+22  Cor=0.322475017270155"
## [18] "18: tokenAmount < 2.62144017945353e+22  Cor=0.319926712185023"
## [19] "19: tokenAmount < 5.24288035890706e+22  Cor=0.318390176997383"
```

[20] "20: tokenAmount < 1.04857607178141e+23 Cor=0.317735338059152"
[21] "21: tokenAmount < 2.09715214356282e+23 Cor=0.317450542449664"
[22] "22: tokenAmount < 4.19430428712565e+23 Cor=0.31725811811073"
[23] "23: tokenAmount < 8.3886085742513e+23 Cor=0.317168616547979"
[24] "24: tokenAmount < 1.67772171485026e+24 Cor=0.317114640194778"
[25] "25: tokenAmount < 3.35544342970052e+24 Cor=0.317104642035045"
[26] "26: tokenAmount < 6.71088685940104e+24 Cor=0.317101357534143"

Maximum Correlation : 0.4

Result for Cumulative correlation with Exponential Layering in base 10:

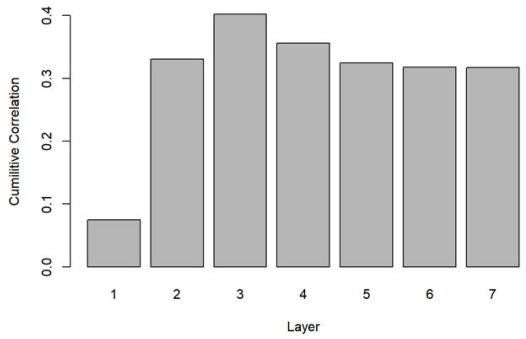

Fig 12.Cumulative correlation with Exponential layering in base 10

Layers:

[1] "1: tokenAmount < 1000000068456089472 Cor=0.075075650608101"
[2] "2: tokenAmount < 10000000684560893952 Cor=0.33030029435338"
[3] "3: tokenAmount < 1.00000006845609e+20 Cor=0.402203424106337"
[4] "4: tokenAmount < 1.00000006845609e+21 Cor=0.356031096201589"
[5] "5: tokenAmount < 1.00000006845609e+22 Cor=0.324207144929298"
[6] "6: tokenAmount < 1.00000006845609e+23 Cor=0.317715744168353"
[7] "7: tokenAmount < 1.00000006845609e+24 Cor=0.31713607735521"

Maximum Correlation : 0.4

Project 2 Implementation and Results:

We denote the token price in dollar as P_t for the tth day. Simple price return is given as $P_t - P_{t-1}/P_{t-1}$.

We extract features from the token network at day t-1. and create a multiple linear regression model to explain price return on day t.

The features that have been used are as below:

- Number of transactions
- Square root of number of transactions
- Percentage of investors that have more than 10 transaction
- Percentage of sellers that sell more than 10 transaction
- Unique number or investors

- Unique number or seller
- Ratio of number of buyer to seller

After feature engineering, we used Multiple Linear Regression to build a regression model.

```
#### Network Table
network <- read.table("token/networkomisegoTX.txt", header=FALSE,
    sep=" ",  col.names= c("fromNodeID", "toNodeID", "unixTime", "tokenAmount"),
colClasses=c('factor', 'factor', 'numeric', 'numeric') )

library(anytime)
network$newDate <- anydate(network$unixTime)
network$newDate <- as.double(network$newDate)
network <- network[order(network$newDate , decreasing = FALSE),]  # Sorting based on Date
networkDF <- as.data.frame.matrix(network)
head(network)
```

```
#### Price Table
Price <- read.table("OMISEGO", header=TRUE,
    sep="\t",  col.names= c("Date", "Open", "High"  ,"Low"  ,"Close",   "Volume",
"MarketCap"), colClasses=c('factor', 'numeric', 'numeric','numeric', 'numeric', 'numeric',
'numeric') )
Price$newDate <- as.Date(strptime(x = as.character(Price$Date), format = "%m/%d/%Y"))
Price <- Price[order(Price$newDate, decreasing = FALSE),  ]  # Sorting based on Date
priceDF <- as.data.frame.matrix(Price)
head(priceDF)
```

Feature Engineering

```
### Creating a data frame with date and price return values
priceReturn <- c()
priceReturn[1] <- 0
transNum <- c()
transNum[1] <- priceDF[1, 'newDate']
transFreqDF <- data.frame(table(cleanNetworkDF$newDate))
colnames(transFreqDF) <- c("newDate", "Freq")
for (t in 2:nrow(priceDF)) {
  # Finding price return in date t
  priceReturn[t] <- (priceDF[t, 'Close'] -  priceDF[t-1 ,'Close'] ) /  priceDF[t-1, 'Close']


  # Counting the number of date or #transaction per date
  transaction <- transFreqDF[transFreqDF$newDate == priceDF[t, 'newDate'], "Freq" ]

  if (is.na(transaction[1])) {
    transNum[t] <- 0
  } else {
    transNum[t] <- sqrt(transaction[1] )
  }
}
investorPerc <- c()
investorPerc[1] <- 0

sellerPerc <- c()
sellerPerc[1] <- 0

uniqueInvestor <- c()
uniqueInvestor[1] <- 0

uniqueSeller <- c()
uniqueSeller[1] <- 0

buyerSellerRatio <-c()
```

```
for (t in 2:nrow(priceDF) ){

    #### Counting the number of investors with more than 10 transaction
  df <- subset(cleanNetworkDF, cleanNetworkDF$newDate == priceDF[t, 'newDate'] ) #,
select=c(newDate)
  investorFreqDF <- data.frame(table(df$toNodeID))
  colnames(transFreqDF) <- c("Investor", "Freq")

  uniqueInvestor[t] = nrow(investorFreqDF)
  investorFreq <- investorFreqDF[investorFreqDF$Freq >10 , "Freq" ]

  if ( uniqueInvestor[t] == 0) {
    investorPerc[t] <- 0
  } else {
    investorPerc[t] <- length(investorFreq) / uniqueInvestor[t]
  }


  #### Counting the number of sellers with more than 10 transaction
  sellerFreqDF <- data.frame(table(df$fromNodeID))
  #colnames(sellerFreqDF) <- c("Seller", "Freq")

  uniqueSeller[t] = nrow(sellerFreqDF)
  sellerFreq <- sellerFreqDF[sellerFreqDF$Freq >20 , "Freq" ]

  if ( uniqueSeller[t] == 0) {
    sellerPerc[t] <- 0
  } else {
    sellerPerc[t] <- length(sellerFreq) / uniqueSeller[t]
  }

  buyerSellerRatio[t] <- uniqueInvestor[t] / uniqueSeller[t]
}

mergeDF = data.frame( priceDF$newDate , priceReturn, transNum, investorPerc, sellerPerc,
uniqueInvestor, uniqueSeller, buyerSellerRatio)
colnames(mergeDF) <- c("newDate", "priceReturn", "transNum", "investorPerc", "sellerPerc",
"uniqueInvestor", "uniqueSeller" , "buyerSellerRatio")

mergeDF <- mergeDF[-1, ]
head(mergeDF)
```

Regression Model (Multiple Linear Regression)

```
model <- lm(mergeDF$priceReturn ~ mergeDF$transNum + mergeDF$investorPerc +
mergeDF$sellerPerc + mergeDF$uniqueInvestor + mergeDF$uniqueSeller +
mergeDF$buyerSellerRatio , data = mergeDF)
summary(model)
##
## Call:
## lm(formula = mergeDF$priceReturn ~ mergeDF$transNum + mergeDF$investorPerc +
##     mergeDF$sellerPerc + mergeDF$uniqueInvestor + mergeDF$uniqueSeller +
##     mergeDF$buyerSellerRatio, data = mergeDF)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -0.30248 -0.05271 -0.00745  0.04578  0.72569
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                -1.233e-01  4.922e-02  -2.505   0.0128 *
## mergeDF$transNum            1.886e-03  1.046e-03   1.802   0.0725 .
## mergeDF$investorPerc        5.809e+00  1.940e+00   2.994   0.0030 **
## mergeDF$sellerPerc         -5.107e-02  1.073e+00  -0.048   0.9621
## mergeDF$uniqueInvestor     -9.510e-06  1.794e-05  -0.530   0.5965
## mergeDF$uniqueSeller        2.462e-06  1.871e-05   0.132   0.8954
## mergeDF$buyerSellerRatio    7.375e-03  2.081e-02   0.354   0.7233
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1178 on 287 degrees of freedom
##   (18 observations deleted due to missingness)
## Multiple R-squared:  0.04895,    Adjusted R-squared:  0.02907
## F-statistic: 2.462 on 6 and 287 DF,  p-value: 0.02448
```

Interpretation

The first step in interpreting the multiple regression analysis is to examine the F-statistic and the associated p-value, at the bottom of model summary.

In our example, it can be seen that p-value of the F-statistic is < 2.462. This means that, at least, one of the predictor variables is somehow related to the outcome variable.

To see which predictor variables are significant, you can examine the coefficients table, which shows the estimate of regression beta coefficients and the associated t-statitic p-values:

The above results shows that the "**investorPerc**" has the highest impact on the model. Besides, the other features does not have strong impact on the model. As a result, we used just one feature to build the regression model.

```
model <- lm(mergeDF$priceReturn ~ mergeDF$investorPerc , data = mergeDF)
## Call:
## lm(formula = mergeDF$priceReturn ~ mergeDF$investorPerc, data = mergeDF)
##
## Coefficients:
##         (Intercept)  mergeDF$investorPerc
##           -0.005883             3.952442
```

```
plot(mergeDF$priceReturn ~ mergeDF$investorPerc)
abline(-0.005883, 3.952442)
abline(lm(mergeDF$priceReturn ~ mergeDF$investorPerc))
```

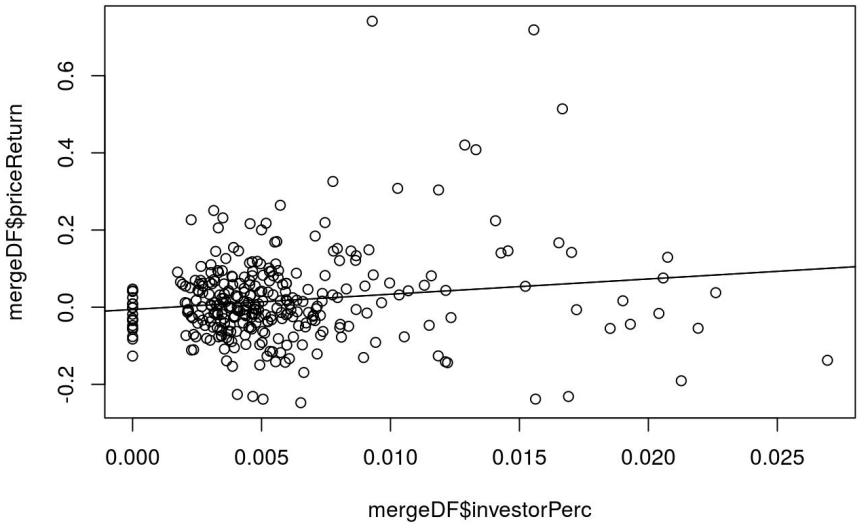

Fig 13. Using Linear Regression for predicting the price

Regression Equation:

priceReturn = -0.005883 + 3.952442 * Investor-Percentage

Reference:

[1] https://arxiv.org/abs/1708.08749