Project 1 & 2

Shurong Tian(sxt151030), Pratik Patel(pnp180002), Vishal Shah(vjs180000)

December 1st, 2018

Introduction

Ethereum is a decentralized, self-contained software platform that provides smart contracts based on blockchain technology and cryptocurrency concepts. It is a virtual machine that records every transaction between users. Smart contracts are self-executing contracts that are written into code in terms of agreement between buyers and sellers. Compare to the Bitcoin that most people are known of, Ethereum allows users to create any decentralized application over a peer-to-peer network whereas Bitcoin tracks ownership of Bitcoins. Vitalik Buterin, a programmer from Toronto, came up with the idea of Ethereum.

ERC-20 tokens are the current industry standard tokens used with smart contracts on the Ethereum platform. ERC-20 stands for Ethereum Request for Comments 20. The Ethereum community created standards for ERC-20 tokens so that they can be shared, transferred, and exchanged.

For this project, we choose **MCAP** (networkmcapTX) as Our primary token. MCAP was created by Bitcoin Growth Fund (BGF) as a mining and ICO fund token. MCAP token uses the counterparty protocol for peer-to-peer transactions. Market cap of MCAP is $84,707 USD (source:coinmarketcap.com).

The packages we are using in this project:

1. **fitdistrplus**: Help to Fit of a Parametric Distribution to Non-Censored or Censored Data. The two main functions are fitdist for fit on non-censored data and fitdistcens for fit on censored data. extends fitdist() function that we have used to find which distribution fits best with our data. Extends qqcomp() that we have used to plots theoretical quantiles against empirical ones.

2. **formatR:** Formate R code automatically Spaces and indent will be added to the code automatically, and comments will be preserved under certain conditions, so that R code will be more human-readable. We used format() function to chabge the formation of date.

3. **Data.table**: data.table inherits from data.frame. It offers fast subset, fast grouping, fast update, fast ordered joins and list columns in a short and flexible syntax, for faster development. It extends data.table() function that we have used to count the frequencies of buyer and seller for their transaction and mapping in other tokens.

Project 1

Question 1

We will be estimating distribution parameters and looking for a best-fitted distribution of how many times a user buys and sells a token.

```
# import the data
networkmcapTX <- read.table("F:/UTD/stats/Ethereum Dataset/Ethereum token graphs/networkmcapTX.txt", quote="\"", comment.char="")
mcap <- read.delim("F:/UTD/stats/Ethereum Dataset/tokenprice/mcap")
# change colomn names
colnames(networkmcapTX) <-c("fromNodeID","toNodeID","unixTime","tokenAmount")
attach(networkmcapTX)
attach(mcap)
```

The outliers of our data are the ones that are bigger than the total amount of the token, which is 10^16. To find the outliers, we simply created a subset for the values that are greater than 10^16. We found two transactions that are outliers.

Preprocessing

```
# total amount of the token is total supply(a) times decimals(b)
a <- 10^8
b <- 10^8
totalAmount <- a*b
message("Total amount of the token is: ", totalAmount)

## Total amount of the token is: 1e+16

outliers <- subset(networkmcapTX,networkmcapTX$tokenAmount > totalAmount)
outliers

##        fromNodeID toNodeID   unixTime  tokenAmount
## 637            81       81 1524669934 5.789604e+76
## 734009    4918508  4943289 1500028528 1.844674e+19
## 771403    4918508  4897478 1499244469 1.844674e+19

# remove the outliers
networkmcapT <- subset(networkmcapTX,networkmcapTX$tokenAmount <= totalAmount)
```

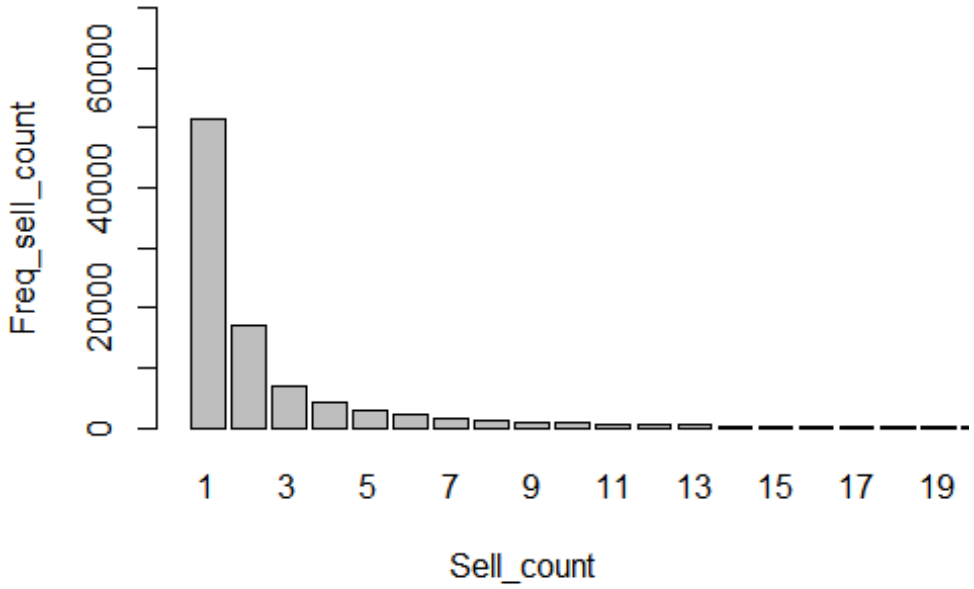
After filtering out the outliers, we looked at how many times a user buys and sells a token and found the numbers of buys/sells with corresponding numbers of users. We plotted the data with buys/sells counts being x-axis and user counts being y-axis.

Assumption - By looking at our data, we assume that our data is going to give us an exponential distribution - most likely a gamma distribution.

Find Distribution for Sells

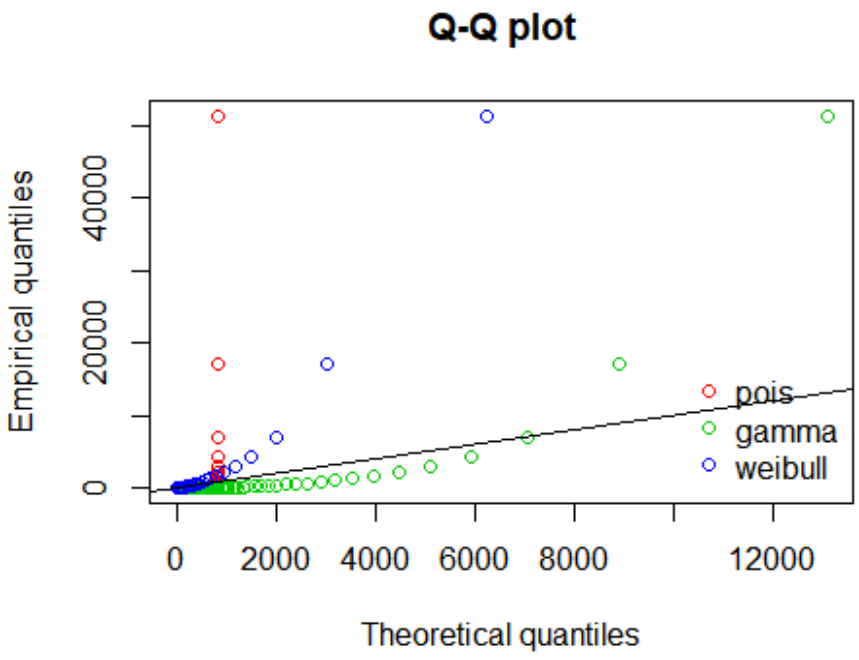
```
#Find Distribution - Sells
f<-as.vector(table(networkmcapT$fromNodeID))
user <- as.vector(unique(networkmcapT$fromNodeID))
user <- sort(user)
tab <- data.frame(user,f)
d <- data.table(from=c(tab$user),x=c(tab$f))
counts<- d[,.(rowCount= .N), by=x]
counts <- counts[order(counts$x),]
```

```
# Plotting data of sells
barplot(counts$rowCount,names.arg =counts$x ,ylab = "Freq_sell_count", xlab = "Sell_count", x
lim = c(0,20),ylim = c(0,70000))
```

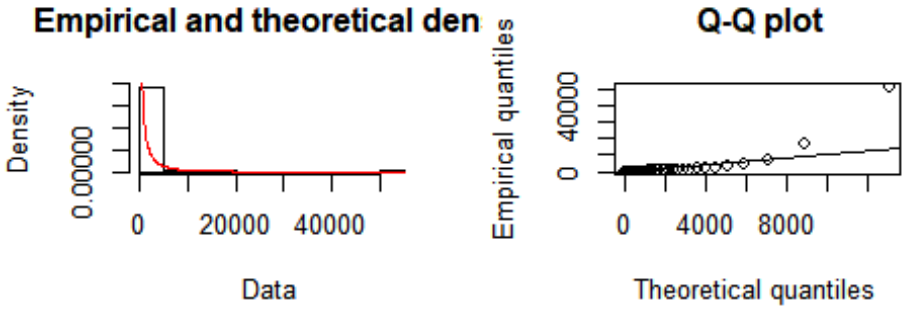


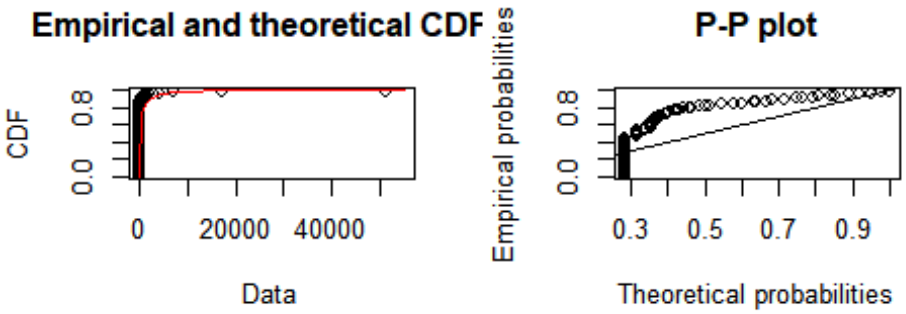
Distribution Fitting

```
poisfit  <-  fitdist(counts$rowCount, "pois")
gammafit <- fitdist(counts$rowCount, "gamma", method = "mle", lower = c(0, 0), start = list(s
cale = 1, shape = 1))
weibullfit  <-  fitdist(counts$rowCount, "weibull")

qqcomp(list(poisfit, gammafit, weibullfit),
       legendtext=c("pois", "gamma", "weibull") )
```




```
gammafit

## Fitting of the distribution ' gamma ' by maximum likelihood
## Parameters:
##          estimate Std. Error
## scale 4943.535552         NA
## shape    0.157015         NA

plot(gammafit)
```

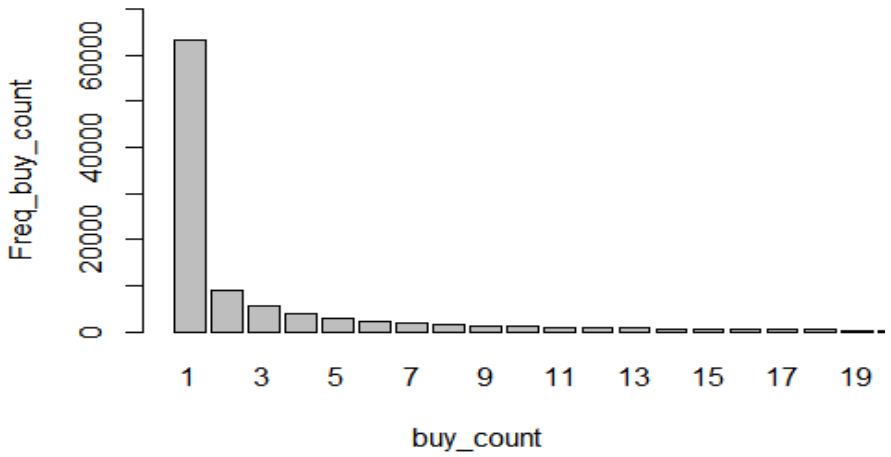




After comparing different distribution fitting, Distribution of Sells data is best fits with “gamma” Distribution

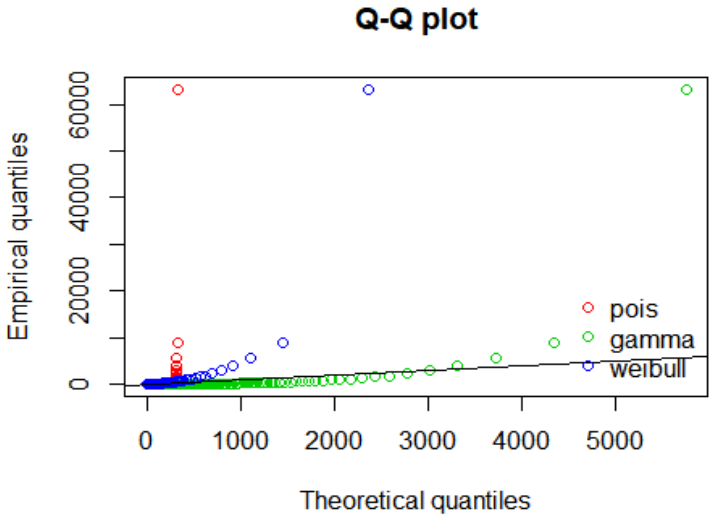
Find Distribution for Buys

```
f <- as.vector(table(networkmcapT$toNodeID))
user <- as.vector(unique(networkmcapT$toNodeID))
user <- sort(user)
tab <- data.frame(user,f)
d <- data.table(from=c(tab$user),x=c(tab$f))
counts<- d[,.(rowCount= .N), by=x]
counts <- counts[order(counts$x),]

# Plotting data of buys
barplot(counts$rowCount,names.arg =counts$x ,ylab = "Freq_buy_count", xlab = "buy_count", xli
m = c(0,20),ylim = c(0,70000))
```

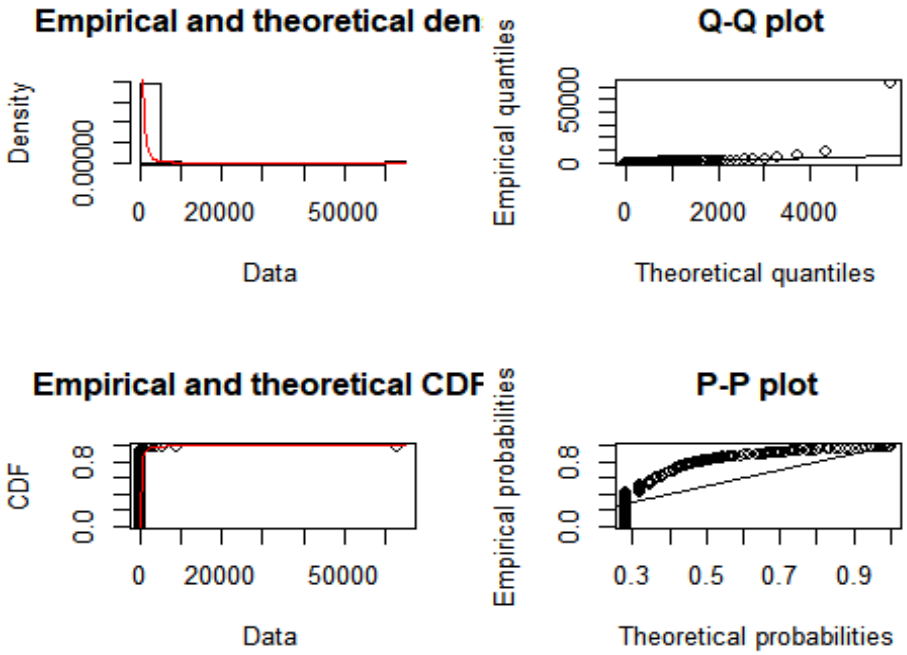


Distribution Fitting

```
poisfit  <-  fitdist(counts$rowCount, "pois")
gammafit <- fitdist(counts$rowCount, "gamma", method = "mle", lower = c(0, 0), start = list(s
cale = 1, shape = 1))
weibullfit  <-  fitdist(counts$rowCount, "weibull")

qqcomp(list(poisfit, gammafit, weibullfit),
       legendtext=c("pois", "gamma", "weibull") )
```



```
gammafit

## Fitting of the distribution ' gamma ' by maximum likelihood
## Parameters:
##           estimate Std. Error
## scale 1540.4842014         NA
## shape    0.1836044         NA

plot(gammafit)
```



After plotting and observing different distributions of discrete data for sell and buy, we have come to a conclusion that the gamma distribution fits best with the data for the token MCAP, which means that our assumption was correct.

Question 2

We will be creating layers of transactions to find the correlation of price data with each of the layers.

```
# Creating table with columns tokenamount and unixtime
table_of_amount <- subset(networkmcapT,select = c(unixTime,tokenAmount))
# Creating table with token daily closing price and date
dates <- mcap$Date
Date <- as.Date(dates, format = "%m/%d/%Y")
Date <- format(Date,"%Y-%m-%d")
Date <- as.Date(Date)
tablePrice <- data.table(Date,mcap$Close)
colnames(tablePrice) <-c("Date","Closing Price")


# Maping number of transaction with date
date <- as.Date(as.POSIXct(table_of_amount$unixTime , origin="1970-01-01"))
amount <- as.vector(table_of_amount$tokenAmount)
data <- data.frame(date,amount)
p <- data.table(from=c(data$amount),Date=c(data$date))
table_of_date<- p[,.(rowCount= .N), by=Date]
colnames(table_of_date) <-c("Date","Num of Transactions")

# Merging both table on Date column
table_of_date_transaction <- merge(table_of_date,tablePrice,by="Date")
table_of_date_transaction
```

```
##            Date Num of Transactions Closing Price
##   1: 2017-05-30                   1      6.540000
##   2: 2017-05-31                   3      5.940000
##   3: 2017-06-01                   4      6.130000
##   4: 2017-06-02                   9      6.550000
##   5: 2017-06-03                  23      6.880000
##  ---
## 338: 2018-05-02                 541      0.134011
## 339: 2018-05-03                 382      0.133013
## 340: 2018-05-04                 784      0.113962
## 341: 2018-05-05                 524      0.115260
## 342: 2018-05-06                 204      0.110585
```

```
# Creating 35 layers based on Dates and calculating correlation for each layer
p <- 1
nr <- 0
pr <- 0
ncr <- 0
pcr <- 0
for (x in 0:34){
    layer <- subset(table_of_date_transaction,
                    table_of_date_transaction$Date >
                      table_of_date_transaction$Date[p]
                    & table_of_date_transaction$Date <
                      table_of_date_transaction$Date[p + 5])

  cr <- cor(layer$`Num of Transactions`,layer$`Closing Price`)

  if(cr < 0){
   nr <- nr + cr
   ncr <- ncr+1
```

```
  }

  else{
    pr <- pr + cr
    pcr <- pcr+1
  }

  p <- p + 5
}

avg_negative_correlation <- nr/ncr
avg_positive_correlation <- pr/pcr
```

We are using “dates” to create layers. Reason for using this approach:

We used different methods like creating layers using tokenamount(0.01?maxtokenamount) and number of transaction but as our data does not follow any pattern or send signals thess approach are not feasible

Still creating layers using dates we got various correlation both positive and neagative but these are best among all other approaches.

here is average positive and neagative correlation

```
# Correlations

avg_positive_correlation

## [1] 0.6188896

avg_negative_correlation

## [1] -0.5486701
```

According to Pearson correlation, the correlation coefficient is between -1 and 1. A good correlation coefficient should be 0.7 or higher. By looking at our correlation for each layer, there are some layers with high correlations. For example, layer 0 has around 0.93 correlation and layer 3 has around -0.96 correlation. But our overall average is not as close. 1/3 of the layers’ data don’t follow the pattern so that they have very low correlations.

Question 3

We will find the most active buyers and sellers in our primary token and track them in other tokens. Then, we will find a distribution for the number of unique tokens they invest in.

```
#Creaating table of active buyers using frequency greater than Mean of frequencies of transac
tion for buyers
p <- data.table(from=c(networkmcapT$fromNodeID),buyers=networkmcapT$fromNodeID)
table_of_buyers<- p[,.(rowCount= .N), by=buyers]
table_of_buyers <- table_of_buyers[order(-table_of_buyers$rowCount),]
Active_buyers <- subset(table_of_buyers,table_of_buyers$rowCount > mean(table_of_buyers$rowCo
unt),select = buyers)

#Creaating table of active sellers using frequency greater than Mean of frequencies of transa
ction for sellers
p <- data.table(from=c(networkmcapT$toNodeID),sellers=networkmcapT$toNodeID)
table_of_sellers<- p[,.(rowCount= .N), by=sellers]
table_of_sellers <- table_of_sellers[order(-table_of_sellers$rowCount),]

Active_sellers <- subset(table_of_sellers,table_of_sellers$rowCount > mean(table_of_sellers$r
owCount),select = sellers)



#importing different tokens and change the column names for the table
#to track buyers and sellers in other tokens, we simply merged buyers column in Active-buyers
and networkaragonTX

networkaragonTX <- read.table("F:/UTD/stats/Ethereum Dataset/Ethereum token graphs/networkara
gonTX.txt", quote="\"", comment.char="")
colnames(networkaragonTX) <-c("buyers","sellers","unixTime","tokenAmount")
buyers_networkaragonTX <- merge(Active_buyers,networkaragonTX,by="buyers")
sellers_networkaragonTX <- merge(Active_sellers,networkaragonTX,by="sellers")
buyers_of_networkaragonTX <- unique(buyers_networkaragonTX$buyers)
sellers_of_networkaragonTX <- unique(sellers_networkaragonTX$sellers)

#importing other tokens as above
```

Creating table for total number of active buyers linked with different tokens

```
##     Number_of_token Number_of_buyers
##  1:               1               48
##  2:               2               11
##  3:               3                5
##  4:               4                4
##  5:               5                3
##  6:               7                1
##  7:               8                3
##  8:              11                2
##  9:              12                2
## 10:              16                1
## 11:              17                1
## 12:              18                1
## 13:              26                1
```

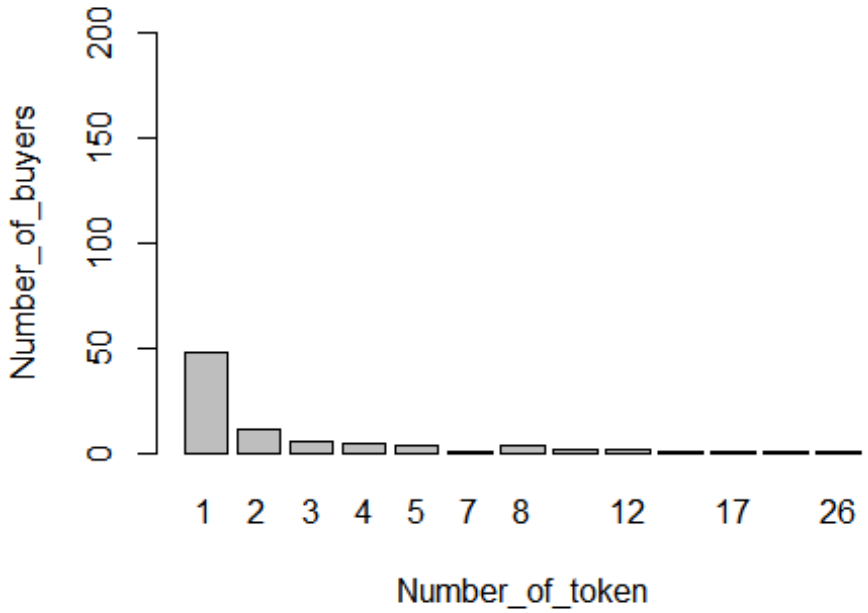
Creating table for total number of active sellers linked with different tokens

```
##     Number_of_token Number_of_sellers
##  1:               1              189
##  2:               2               48
```

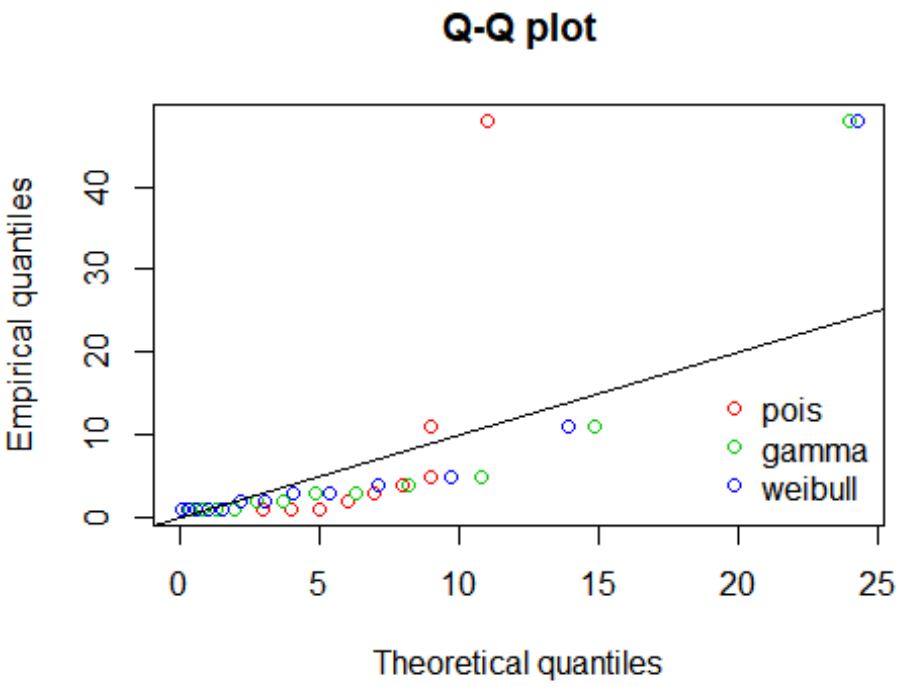
```
##  3:                3                17
##  4:                4                12
##  5:                5                 8
##  6:                6                 2
##  7:                7                 2
##  8:                9                 4
##  9:               10                 3
## 10:               11                 1
## 11:               14                 3
## 12:               17                 3
## 13:               19                 1
## 14:               20                 1
## 15:               21                 1
## 16:               31                 1
```

Distribution Fitting for active buyers

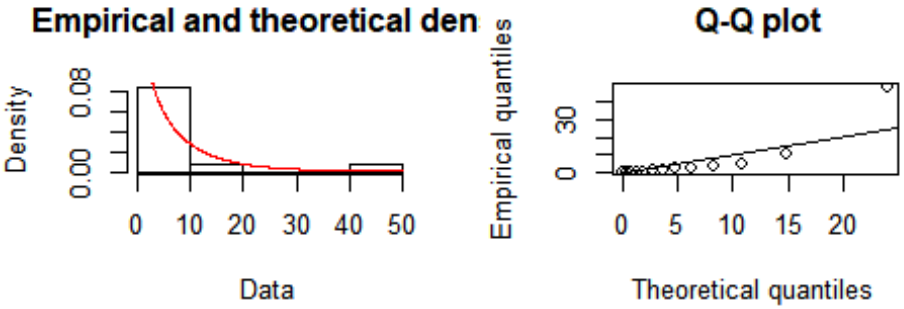
```
barplot(buyer_count$Number_of_buyers,names.arg =buyer_count$Number_of_token ,ylab = "Number_of_buyers", xlab = "Number_of_token",ylim = c(0,200))
```

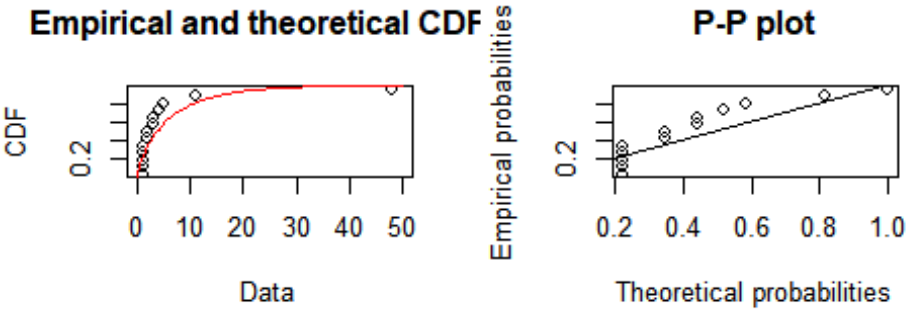



```
poisfit  <-  fitdist(buyer_count$Number_of_buyers, "pois")
gammafit <- fitdist(buyer_count$Number_of_buyers, "gamma", method = "mle", lower = c(0, 0), start = list(scale = 1, shape = 1))
weibullfit  <-  fitdist(buyer_count$Number_of_buyers, "weibull")

qqcomp(list(poisfit, gammafit, weibullfit),
       legendtext=c("pois", "gamma", "weibull") )
```



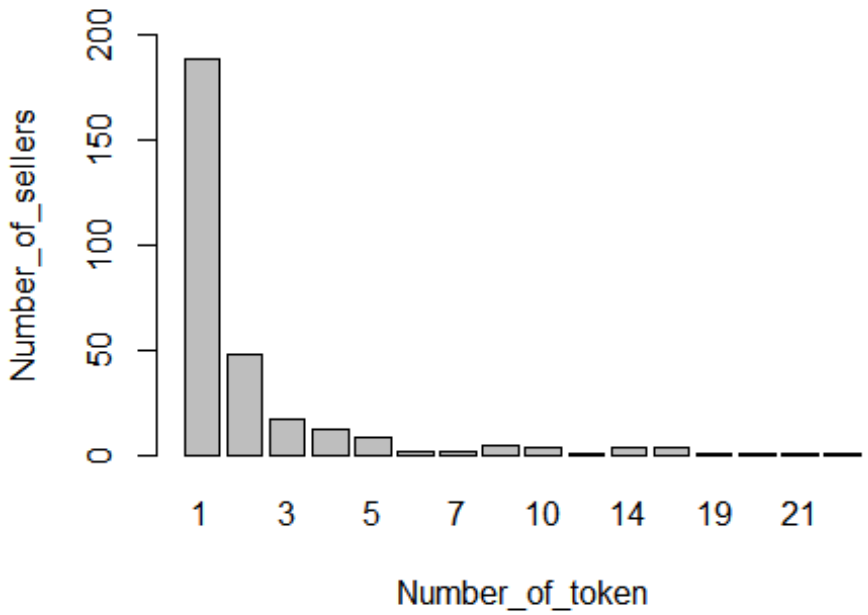

```
gammafit

## Fitting of the distribution ' gamma ' by maximum likelihood
## Parameters:
##        estimate Std. Error
## scale 9.1380718         NA
## shape 0.6984628         NA

plot(gammafit)
```



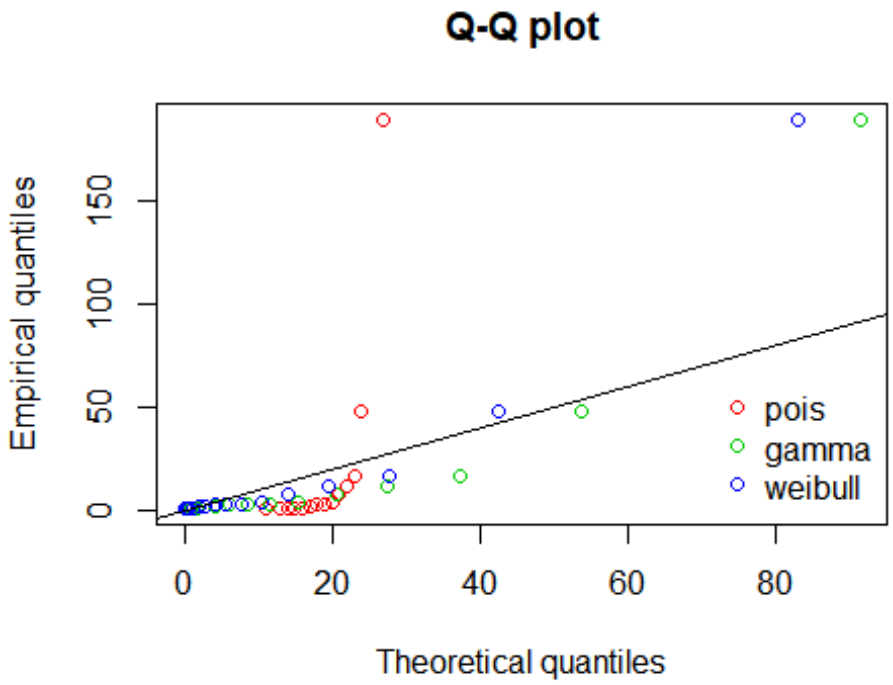


Distribution Fitting for active sellers

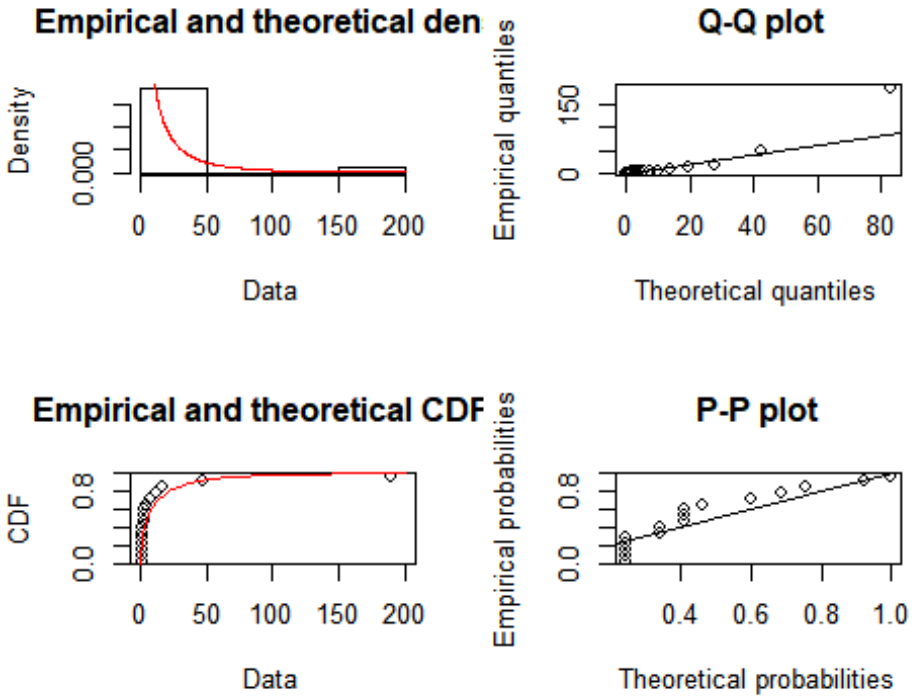
```
barplot(seller_count$Number_of_sellers,names.arg =seller_count$Number_of_token ,ylab = "Numbe
r_of_sellers", xlab = "Number_of_token",ylim = c(0,200))
```




```
poisfit  <-  fitdist(seller_count$Number_of_sellers, "pois")
gammafit <- fitdist(seller_count$Number_of_sellers, "gamma", method = "mle", lower = c(0, 0),
start = list(scale = 1, shape = 1))
weibullfit  <-  fitdist(seller_count$Number_of_sellers, "weibull")

qqcomp(list(poisfit, gammafit, weibullfit),
       legendtext=c("pois", "gamma", "weibull") )
```



```
weibullfit

## Fitting of the distribution ' weibull ' by maximum likelihood
## Parameters:
##        estimate Std. Error
## shape 0.5675289 0.09781738
## scale 9.2895167 4.36230402

plot(weibullfit)
```



After plotting and observing different distributions of discrete data for active user and their activity in other tokens, we have come to a conclusion that the weibull distribution fits best with the number of active users for both seller and buyers.

Conclusion

After completing this project, we learned more about Ethereum tokens and what token transactions look like. More importantly, we practiced to find and analyze relations from the data that was given to us. By making assumption first and fitting different distributions to our data, we finally concluded that our token MCAP has a gamma distribution. By creating layers using dates, we found that our correlation for each layer is not great. By tracking our most active users in all of the other tokens, we found that the Weibull distribution fits the best.

“All in all, we can sum up that our token mcap’s transaction were mostly influenced by top active buyers and sellers”.

Project 2

```r
# Creating table with columns tokenamount and unixtime
table_of_transaction <- subset(networkmcapT ,select=c(unixTime,tokenAmount))
# Creating table with token daily closing price and date
dates <- mcap$Date
Date <- as.Date(dates, format = "%m/%d/%Y")
Date <- format(Date,"%Y-%m-%d")
Date <- as.Date(Date)
tablePrice <- data.table(Date,mcap$Close)
colnames(tablePrice) <-c("Date","Closing Price")


# Maping number of transaction with date
date <- as.Date(as.POSIXct(table_of_transaction$unixTime , origin="1970-01-01"))
amount <- as.vector(table_of_transaction$tokenAmount)
data <- data.frame(date,amount)
p <- data.table(from=c(data$amount),Date=c(data$date))
table_of_date<- p[,.(rowCount= .N), by=Date]
colnames(table_of_date) <-c("Date","Num of transaction")


#calculating percent change in closing price
table_Price <- data.table(Date,mcap$Close)
n <- nrow(table_Price)
a <- mcap$Close[ 2 : n ]
b <- mcap$Close[ 1 : (n - 1)]
percent_change <- (a - b) * 100 / b
percent_change[n] <- mean(percent_change)

table_Price <- data.table(Date,mcap$Close,percent_change)
new_table <- merge(table_Price,table_of_date,by="Date")
colnames(new_table) <- c("Date","close","percent_change","Num of transaction")

#Creating Multiple linear regression model for each layer using features like number of trans
action and percentage change in closing price

# summary for first five layers
p <- 1
for (x in 0:4){
    layer <- subset(new_table,
                    new_table$Date >
                      new_table$Date[p]
                    & new_table$Date <=
                      new_table$Date[p + 5])
    p <- p + 5
fit <- lm(layer$close ~ layer$`Num of transaction` +
           layer$percent_change , data=layer)
print(summary(fit))
}
```

```
Layer 1
## Call:
## lm(formula = layer$close ~ layer$`Num of transaction` + layer$percent_change,
##     data = layer)
## Residuals:
##        1        2        3        4        5
##  0.07928 -0.26598 -0.01594  0.27363 -0.07099
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  6.242056   0.165847  37.638 0.000705 ***
## layer$`Num of transaction`   0.007502   0.004837   1.551 0.261032
## layer$percent_change        -0.039981   0.020223  -1.977 0.186667
## Residual standard error: 0.2804 on 2 degrees of freedom
## Multiple R-squared:  0.7128, Adjusted R-squared:  0.4257
## F-statistic: 2.482 on 2 and 2 DF,  p-value: 0.2872
##
Layer 2
##
## Residuals:
##        1        2        3        4        5
## -0.32507  0.32399  0.22575 -0.03317 -0.19150
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  7.2038583  0.3810996  18.903  0.00279 **
## layer$`Num of transaction` -0.0009729  0.0049314  -0.197  0.86183
## layer$percent_change       -0.0394825  0.0254676  -1.550  0.26121
## Residual standard error: 0.3869 on 2 degrees of freedom
## Multiple R-squared:  0.782,  Adjusted R-squared:  0.5641
## F-statistic: 3.588 on 2 and 2 DF,  p-value: 0.218
##
Layer 3
## Residuals:
##        1        2        3        4        5
##  0.32073 -0.08671  0.20537 -0.51472  0.07533
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  8.03731    0.72938  11.019  0.00814 **
## layer$`Num of transaction` -0.05969    0.02427  -2.460  0.13308
## layer$percent_change       -0.06489    0.01167  -5.559  0.03087 *
## Residual standard error: 0.46 on 2 degrees of freedom
## Multiple R-squared:  0.9399, Adjusted R-squared:  0.8799
## F-statistic: 15.65 on 2 and 2 DF,  p-value: 0.06007
##
Layer 4
## Residuals:
##        1        2        3        4        5
## -0.36082  0.02411 -0.07964  0.37534  0.04101
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  4.740599   0.642629   7.377   0.0179 *
## layer$`Num of transaction` -0.001585   0.009664  -0.164   0.8848
## layer$percent_change         0.036644   0.087070   0.421   0.7148
## Residual standard error: 0.3739 on 2 degrees of freedom
## Multiple R-squared:  0.147,  Adjusted R-squared:  -0.706
```

```
## F-statistic: 0.1724 on 2 and 2 DF,  p-value: 0.853
##
```

```
Layer 5
## Residuals:
##        1        2        3        4        5
## -0.16706  0.01217  0.02155  0.09773  0.03561
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  4.845076   0.117476  41.243 0.000587 ***
## layer$`Num of transaction`   0.006929   0.005812   1.192 0.355436
## layer$percent_change        -0.029642   0.019263  -1.539 0.263707
## Residual standard error: 0.1403 on 2 degrees of freedom
## Multiple R-squared:  0.5681, Adjusted R-squared:  0.1361
## F-statistic: 1.315 on 2 and 2 DF,  p-value: 0.4319
```

```
# Creating Multiple linear regression model for remaining layers
for (x in 5:34){
    layer <- subset(new_table,
                    new_table$Date >
                      new_table$Date[p]
                    & new_table$Date <=
                      new_table$Date[p + 5])
    p <- p + 5
fit <- lm(layer$close ~ layer$`Num of transaction` +
           layer$percent_change , data=layer)
summary(fit)
}
```

conclusion

We are using layers that we have created during Project 1 to create the multiple linear regression model. Reason for using layers is that our token MCAP's correlation is very low while for some of the layers correlation is around 0.6 which is helpful for creating proper multiple linear regression model.

The features that we used are "Number of transactions" for particular Date from blockchain data and the "percentage change in closing price" from price data.

Average Multiple R-squared value for the layers is around 0.65 and Average Adjusted R-squared value for the layers is around 0.57. We are getting better R-squared value for layers which has higher correlation.

We have also tried features from price data like "Market.cap" and "change in higher and lower price" of token on particular day but we ended up with overfitting issue as R-squared was around 0.99 and while we used data from blockchain data R-squared value was too low.

References

Akcora, et al. “Blockchain: A Graph Primer.” *[Astro-Ph/0005112] A Determination of the Hubble Constant from Cepheid Distances and a Model of the Local Peculiar Velocity Field*, American Physical Society, 10 Aug. 2017, arxiv.org/abs/1708.08749.

“R Packages.” *RStudio*, 11 Oct. 2018, www.rstudio.com/products/rpackages/.