Report

Pushpita Panigrahi(pxp171530), Akash Chand(axc173730), Siddharth Swarup Panda(ssp171730)

Introduction

Blockchain Technology

Block chain is a growing list of records, called blocks, which are linked using cryptography (cryptography is the practice and study of techniques for secure communication in the presence of third parties) .Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data. *"The blockchain is an incorruptible digital ledger of economic transactions that can be programmed to record not just financial transactions but virtually everything of value."* It is an open, distributed ledger (a consensus of replicated, shared, and synchronized digital data geographically spread across multiple sites) that can record transactions between two parties efficiently and in a verifiable and permanent way". A blockchain is typically managed by a peer-to-peer network collectively adhering to a protocol for inter-node communication and validating new blocks. Once recorded, the data in any given block cannot be altered without alterating all its subsequent blocks, which requires consensus of the network majority.

Ethereum

Ethereum is a distributed public block chain network that focuses on running programming code of any decentralized application. More simply, it is a platform for sharing information across the globe that cannot be manipulated or changed. Ether, a decentralized digital currency, also known as ETH is a cryptocurrency whose blockchain is generated by the Ethereum platform. In addition to being a tradeable cryptocurrency, ether powers the Ethereum network by paying for transaction fees and computational services. As with other cryptocurrencies, the validity of each ether is provided by a blockchain, which is a continuously growing list of records, called blocks, which are linked and secured using cryptography. Unlike Bitcoin, Ethereum operates using accounts and balances in a manner called state transitions.

ERC-20 is a technical standard used for smart contracts on the Ethereum blockchain for implementing tokens. ERC stands for Ethereum Request for Comment, and 20 is the number that was assigned to this request.ERC-20 defines a common list of rules for Ethereum tokens to follow within the larger Ethereum ecosystem, allowing developers to accurately predict interaction between tokens. These rules include how the tokens are transferred between addresses and how data within each token is accessed

BNB Token

We chose Binance coin(networkbnbTX) token as our dataset. There are typically two different types of exchanges: the ones that deal with fiat currency and the ones that deal purely in crypto. The latter one even though they are small now, it is expected that pure crypto exchanges will have a bigger impact than fiat based exchanges in the near future. They will play an ever more important role in world finance and this new paradigm is called as Binance; Binary Finance. Features of Binance include : 1) Safety Stability - Multi-tier & multi-cluster system architecture 2) High Performance - Capable of processing 1,400,000 orders per second 3) High Liquidity - Abundant resources and partners 4) All Devices Covered - Web, Android, iOS, Mobile Web, Windows, macOS 5) Multilingual Support - Support and FAQs available in multiple languages 6) Multiple-Coin Support - BTC, ETH, LTC, BNB

Steps

Data preparation and preprocessing

The networkbnbTX file is read into a dataframe. We get the total supply and possible sub units of BNB token in market from www.coinmarketcap.com. Token edge files have following row structure: **fromNodeID,**

toNodeID, unixTime, tokenAmount. Each row implies that *fromNodeID* sold tokenAmount of the token to *toNodeID* at time *unixTime*. *fromNodeID* and *toNodeID* are ids for people who invest in the token in real life. Each person can use multiple ids and 2 ids can sell/buy tokens multiple times with multiple amounts. This makes our network a weighted, directed, multiedge graph. Below is the sample data from the network file:

```
library(plyr)
library(ggplot2)
library(fitdistrplus)
```

```
## Loading required package: MASS

## Loading required package: survival

## Loading required package: npsurv

## Loading required package: lsei
```

```
require(plyr)
library(grid)
```

```
file <-'networkbnbTX.txt'
col_names <- c("FROMNODE","TONODE","DATE","TOKENAMOUNT")
mydata <- read.csv( file, header = FALSE, sep = " ", dec = ".", col.names = col_names)
mydata$DATE <- as.Date(as.POSIXct(as.numeric(mydata$DATE), origin = '1970-01-01', tz = 'GMT'))
amounts <- mydata[4]

totalSupply <- 192443301
subUnits <- 18
totalAmount <- totalSupply * (10 ^ subUnits)
head(mydata)
```

```
##   FROMNODE  TONODE       DATE  TOKENAMOUNT
## 1       82 1443996 2018-04-24 4.071000e+19
## 2       82 1443997 2018-04-24 2.291000e+19
## 3        5 1443998 2018-04-24 2.297303e+18
## 4  1443999 1444000 2018-04-24 8.740000e+18
## 5       44 1444001 2018-04-24 1.180000e+18
## 6        5 1444002 2018-04-24 3.276959e+20
```

The price file for BNB token is read into a dataframe which contains the open, clase, max and min price for the token foe each day. The row structure of the file is **Date, Open, High, Low, Close, Volume, MarketCap**. *Open* and *close* are the prices of the specific token at each day. *Volume* gives total bought/sold tokens and *MarketCap* gives the market valuation at each day. Below is a sample data of the price file:

```
pricefile <-'bnb.txt'
col_names <- c("Date","Open","High","Low","Close","Volume","MarketCap")
myPrices <- read.csv( pricefile , header = TRUE, sep = "\t", dec = ".", col.names = col_names)
myPrices$Date <- format(as.Date(myPrices$Date, format = "%m/%d/%Y"), "%Y-%m-%d")
head(myPrices)
```

```
##         Date  Open  High   Low Close     Volume     MarketCap
## 1 2018-07-04 14.23 14.33 13.91 14.01 37,043,700 1,622,370,000
## 2 2018-07-03 14.56 14.78 14.08 14.17 60,657,300 1,660,830,000
## 3 2018-07-02 14.40 14.82 14.06 14.57 55,614,000 1,641,930,000
## 4 2018-07-01 14.68 14.69 14.14 14.40 38,434,400 1,673,690,000
## 5 2018-06-30 14.55 15.18 14.29 14.66 59,676,900 1,659,200,000
## 6 2018-06-29 14.17 14.65 13.78 14.51 52,784,600 1,616,460,000
```

Preprocessing

The preprocessing step involves removal of fraudulent transactions which might affect the distribution estimate negatively. The total supply of the networkbnb token is 192443301 (quoted from etherscan.io) and the range of subunits for the token is 18 decimal units. Thus any transaction having that attempts to log a value, i.e, *tokenAmount*, greater than the product of total supply and subunits is deemed as fraudulent. **Result:** The token networkbnb does not have any fraudulent transactions.

```
temp <- which(mydata< totalAmount)
#print meta data
message('Maximum allowed amount : ', totalAmount)
```

```
## Maximum allowed amount : 1.92443301e+26
```

```
count <- 0
outliers <- 0
for( a in 1:nrow(amounts)){
  if( a > totalAmount){
    outliers <- outliers + 1
  }
  else{
    count <- count + 1
  }
}
message('Number of fradulent transactions : ',outliers)
```

```
## Number of fradulent transactions : 0
```

```
message('Number of valid amounts : ',count)
```

```
## Number of valid amounts : 357142
```

Package used to fit distributions

For approximating distributions, descdist R function is used which provides a Cullen and Frey skewness-kurtosis plot. Bootstrapped samples of the data have been used in order to consider the uncertainty of the estimated skewness and kurtosis values which are higher order moments. Skewness is a measure of symmetry while kurtosis is the measure of the combined weight of distribution's tails relative to the center of the distribution, which is why both the statistics are dependable to estimate the distribution of the data.

Function to remove outliers

The data points which fall outside 2.5 times the inter-quartile range are considered as outliers and are removed from the dataset. We tested our experiments with values 1, 1,5, 2, 2.5, 3 times the IQR and found the best results with 2.5 IQR value. We believe this is because we need a significant number of data points to fit our distributions.

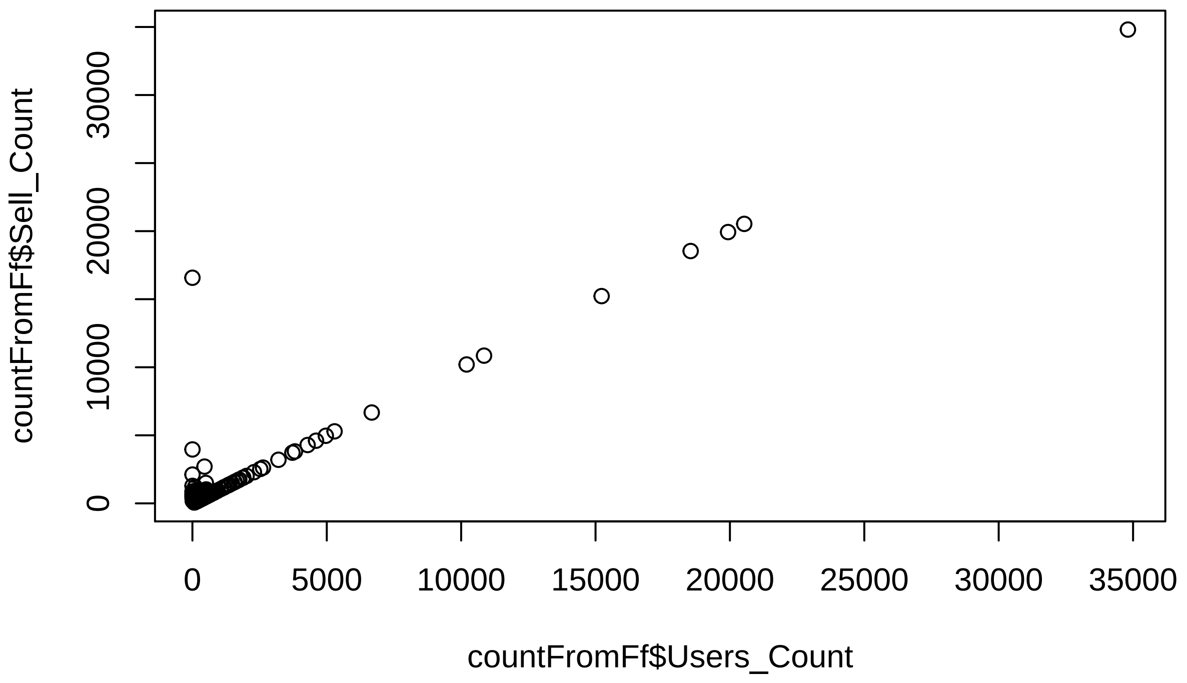
```
remove_outliers <- function(x, na.rm = TRUE, ...) {
  qnt <- quantile(x, probs=c(.25, .75), na.rm = na.rm, ...)
  H <- 2.5 * IQR(x, na.rm = na.rm)
  y <- x
  y[x < (qnt[1] - H)] <- NA
  y[x > (qnt[2] + H)] <- NA
  y
}
```

Study 1:

The aim of this experiment is to find a distribution of how many times the user id transacts in BNB token. We find the frequencies of buys and sells separately for each user and try to fit a distribution for the frequencies. This helps us understand the transaction patterns user wise for our token. Below is the statistics of the frequency distribution and visualisation of the count of frequencies. This helps us understand how likely it is to retain users using the token.

Calculating and plotting selling frequency

```
## Using freq as weighting variable
```

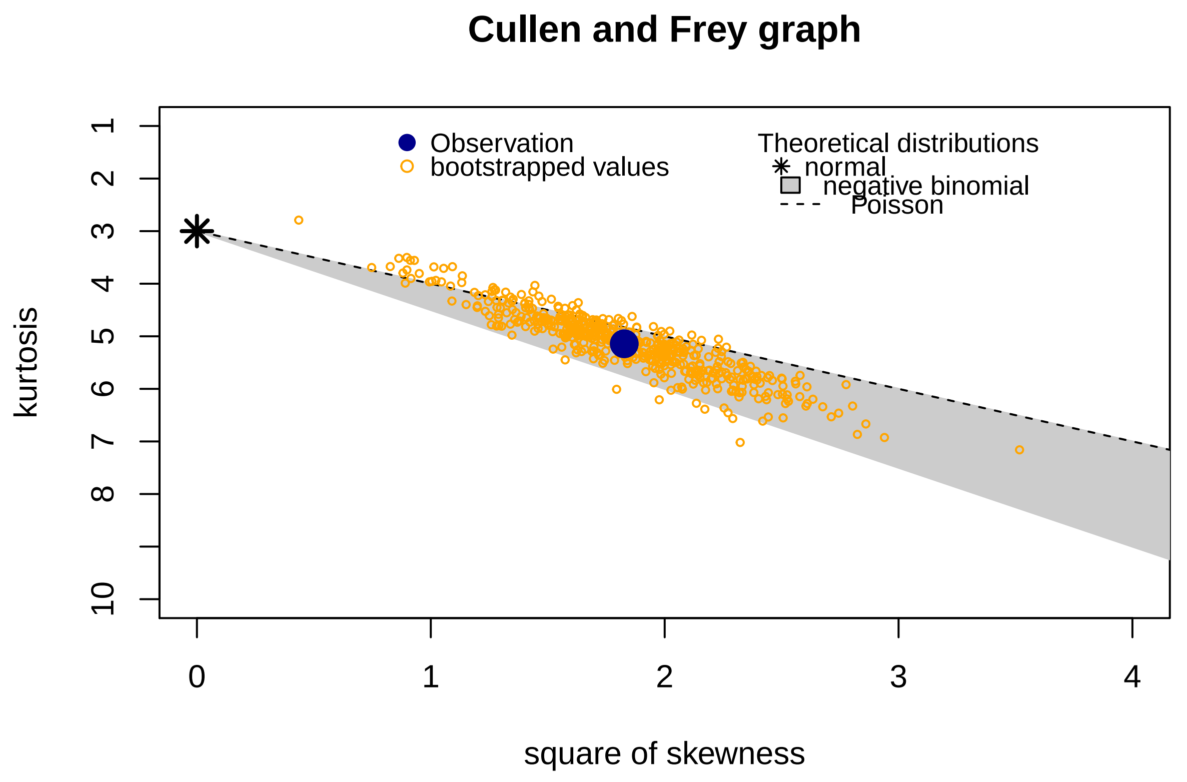


Removing outliers and summarizing data

```
newSet1 <- remove_outliers(countFromFf$Sell_Count)
maxCount1 = max(newSet1[complete.cases(newSet1)])
minCount1 = min(newSet1[complete.cases(newSet1)])
countFromFf <- subset(countFromFf, Sell_Count<maxCount1 & Sell_Count>minCount1)
head(countFromFf)
```

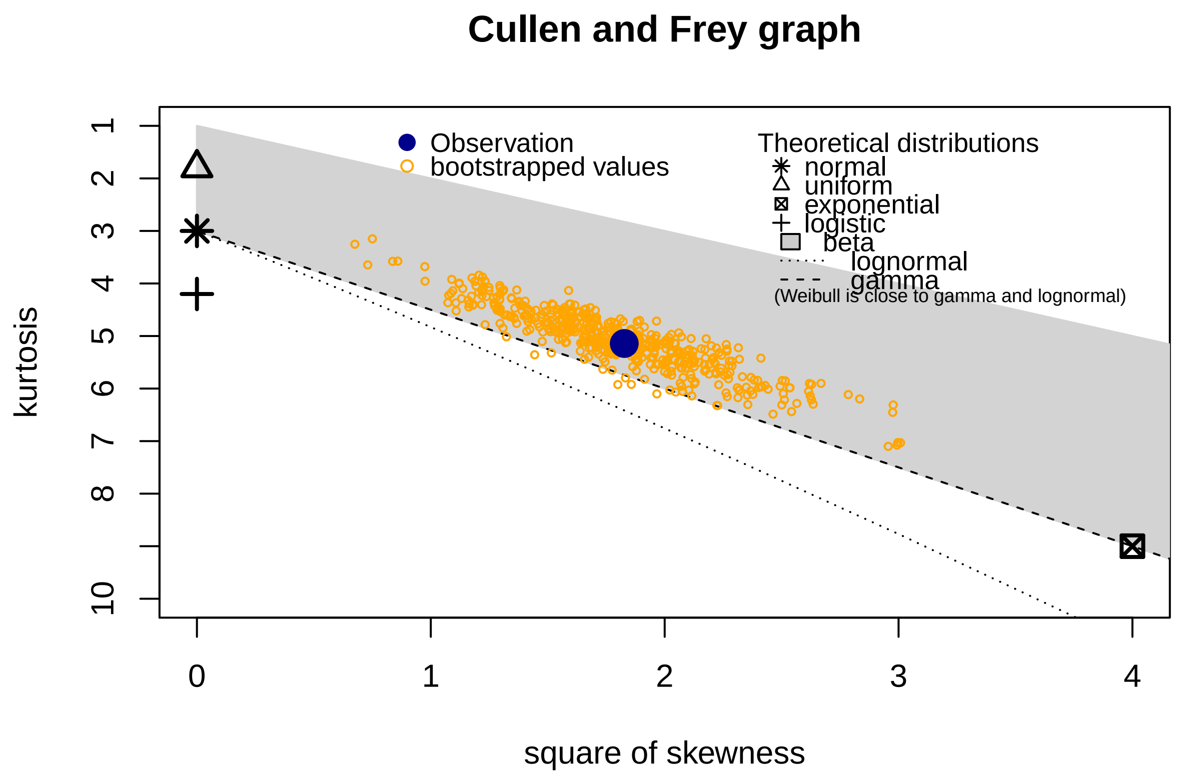
```
##   Users_Count Sell_Count
## 4           4       1284
## 5           5        870
## 6           6        702
## 7           7        588
## 8           8        392
## 9           9        504
```

```
descdist(countFromFf$Sell_Count, boot= 500, discrete=TRUE)
```



```
## summary statistics
## ------
## min:  74   max:  1375
## median:  373.5
## mean:  431.3773
## estimated sd:  240.2746
## estimated skewness:  1.351751
## estimated kurtosis:  5.142384
```

```
descdist(countFromFf$Sell_Count, boot= 500)
```



```
## summary statistics
## ------
## min:  74   max:  1375
## median:  373.5
## mean:  431.3773
## estimated sd:  240.2746
## estimated skewness:  1.351751
## estimated kurtosis:  5.142384
```

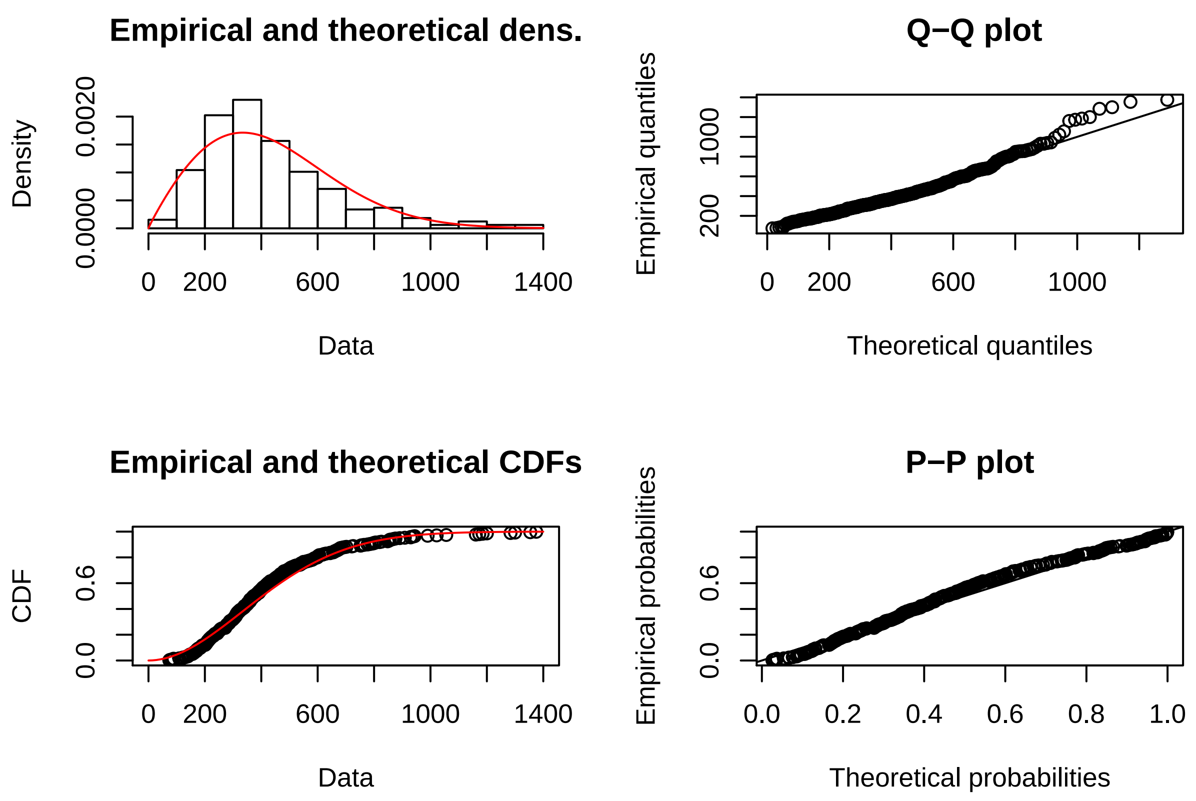
Approximating the selling distributions

From the above Cullen and Frey graph we could narrow down our distribution selection to Weibull, lognormal, gamma and poisson.

```
distributionFit_Seller_pois <- fitdist(countFromFf$Sell_Count, "pois", method ="mle")
distributionFit_Seller_wb <- fitdist(countFromFf$Sell_Count, "weibull", method ="mle")
distributionFit_Seller_ln <- fitdist(countFromFf$Sell_Count, "lnorm", method ="mle")
distributionFit_Seller_gm <- fitdist(countFromFf$Sell_Count, "gamma" ,method="mme")
distributionFit_Seller_wb$loglik
```

```
## [1] -2212.37
```

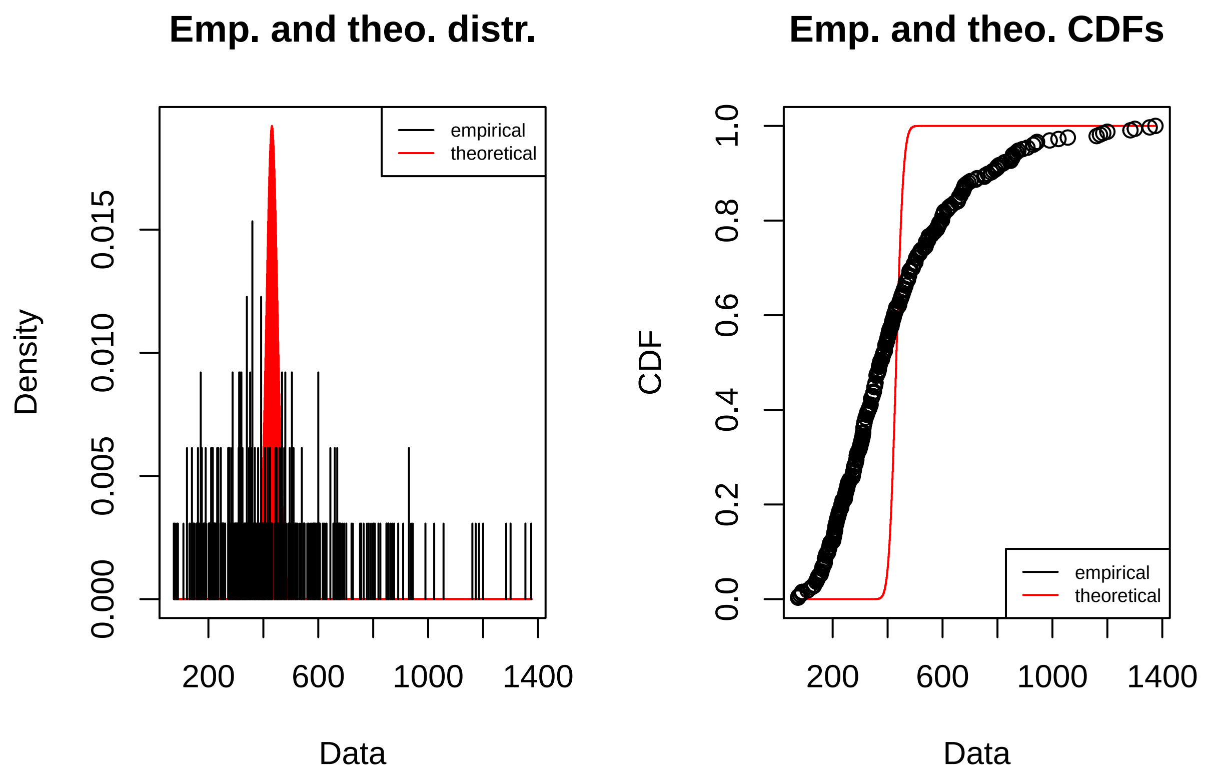
```
plot(distributionFit_Seller_wb)
```



```
distributionFit_Seller_pois$loglik
```

```
## [1] -20956.41
```

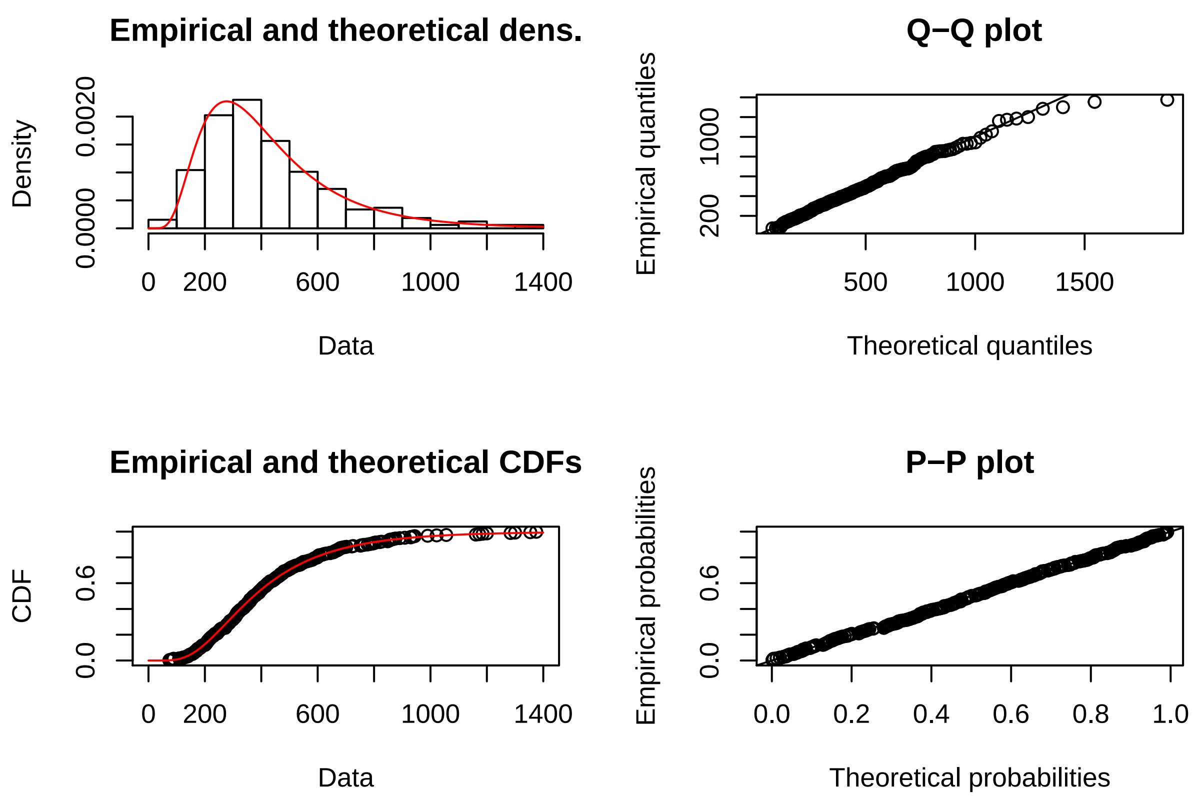
```
plot(distributionFit_Seller_pois)
```



```
distributionFit_Seller_ln$loglik
```

```
## [1] -2195.726
```

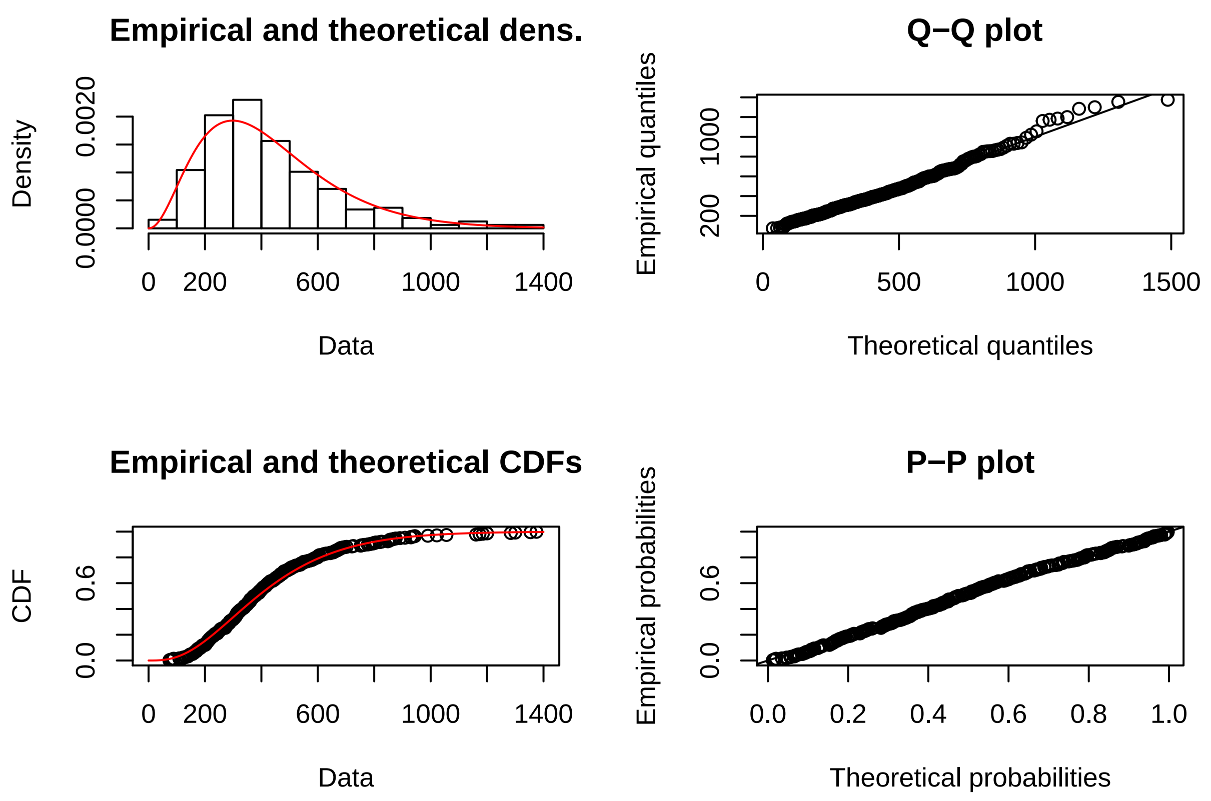
```
plot(distributionFit_Seller_ln)
```



```
distributionFit_Seller_gm$loglik
```

```
## [1] -2199.702
```

```
plot(distributionFit_Seller_gm)
```

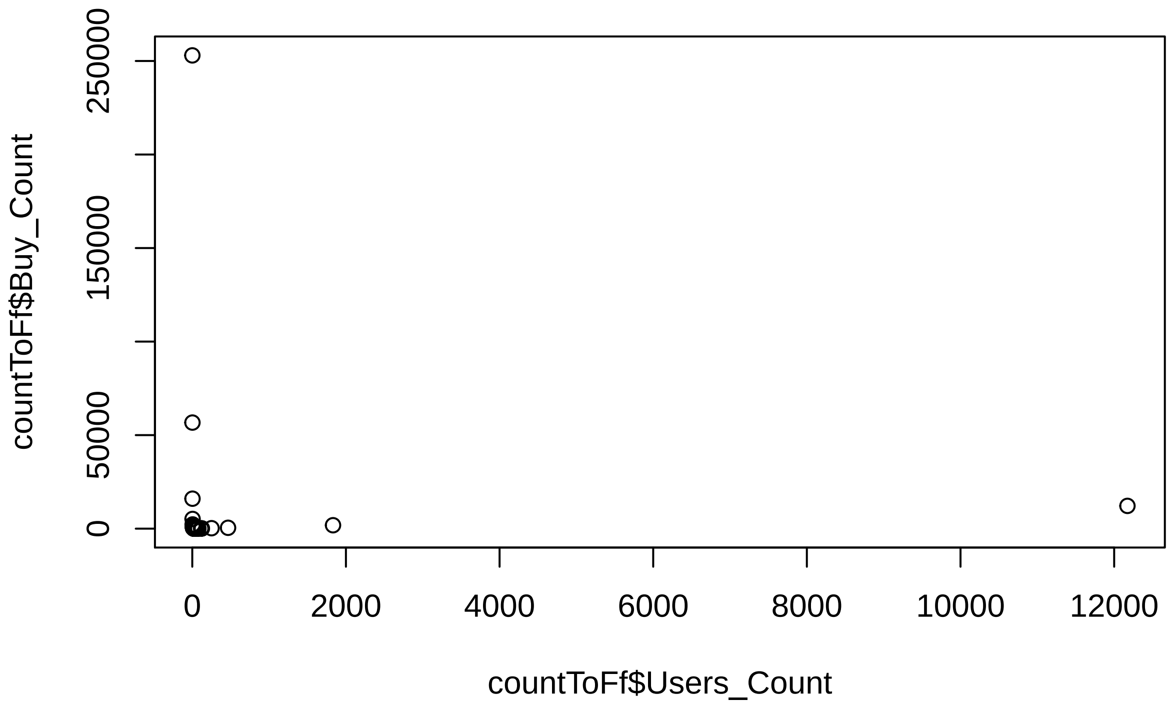


Calculating and plotting buying frequency

```
countToDf <- count(mydata, "TONODE")
countToFf <- count(countToDf, "freq")
```

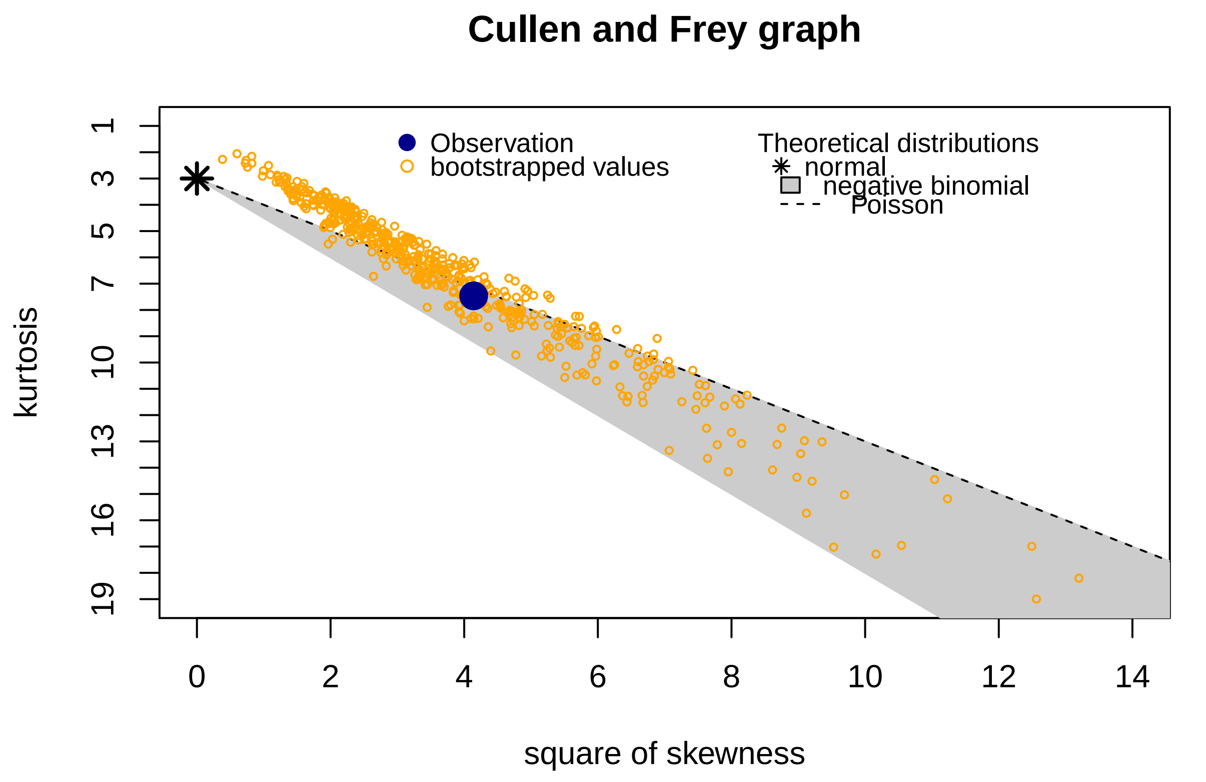
```
## Using freq as weighting variable
```

```
colnames(countToFf) <- c("Users_Count", "Buy_Count")
plot(countToFf$Users_Count, countToFf$Buy_Count)
```



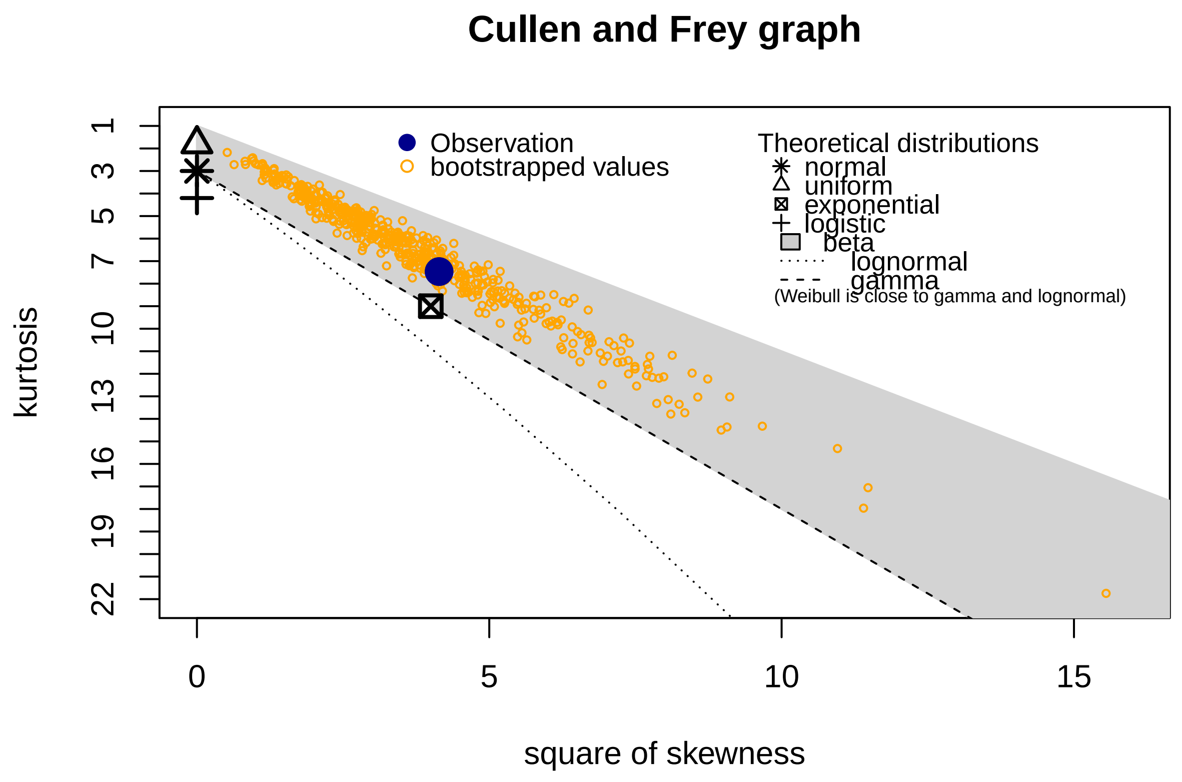
Removing outliers and summarizing data

```
newSet2 <- remove_outliers(countToFf$Buy_Count)
maxCount2 = max(newSet2[complete.cases(newSet2)])
minCount2 = min(newSet2[complete.cases(newSet2)])
countToFf <- subset(countToFf, Buy_Count<maxCount2 & Buy_Count>minCount2)
descdist(countToFf$Buy_Count, boot= 500, discrete=TRUE)
```



```
## summary statistics
## ------
## min:  32   max:  639
## median:  84
## mean:  140.8085
## estimated sd:  128.0521
## estimated skewness:  2.035078
## estimated kurtosis:  7.463705
```

```
descdist(countToFf$Buy_Count, boot=500)
```



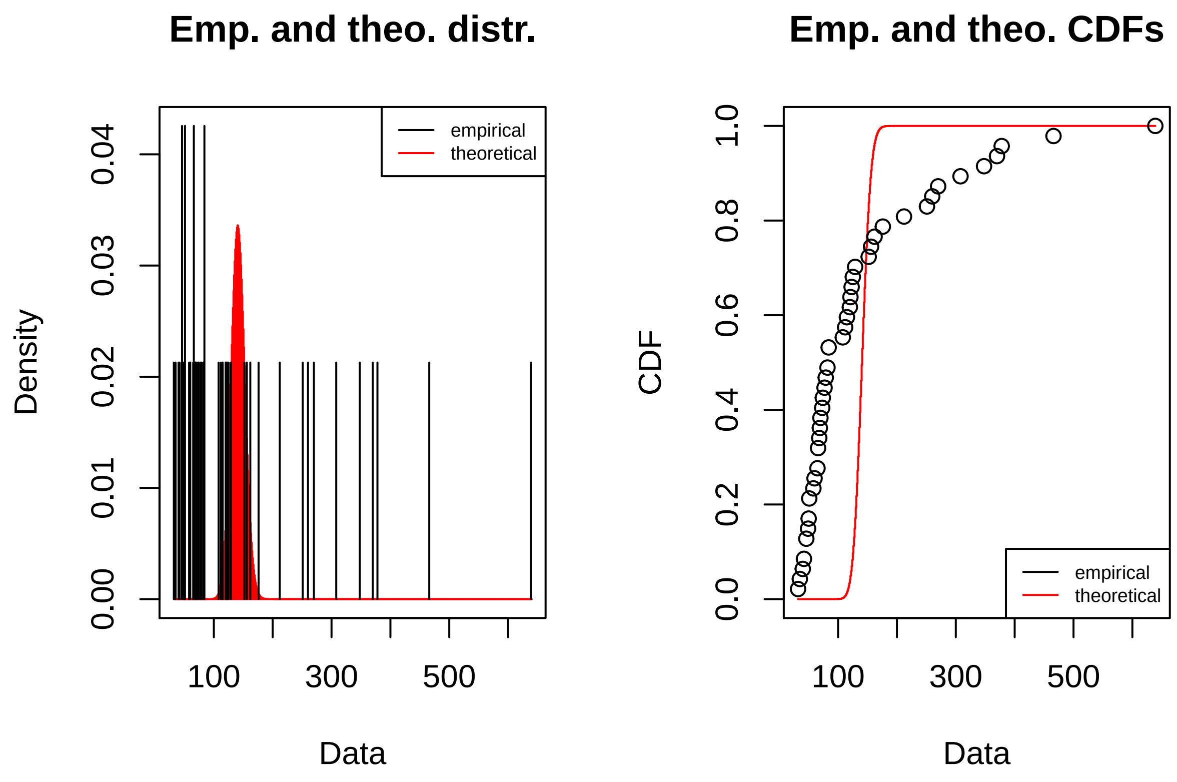
```
## summary statistics
## ------
## min:  32   max:  639
## median:  84
## mean:  140.8085
## estimated sd:  128.0521
## estimated skewness:  2.035078
## estimated kurtosis:  7.463705
```

Approximating the buying distributions

```
distributionFit_Buyer_pois <- fitdist(countToFf$Buy_Count, "pois", method ="mle")
distributionFit_Buyer_wb <- fitdist(countToFf$Buy_Count, "weibull", method ="mle")
distributionFit_Buyer_ln <- fitdist(countToFf$Buy_Count, "lnorm", method ="mle")
distributionFit_Buyer_gm <- fitdist(countToFf$Buy_Count, "gamma", method ="mme")
distributionFit_Buyer_pois$loglik
```

```
## [1] -2235.383
```

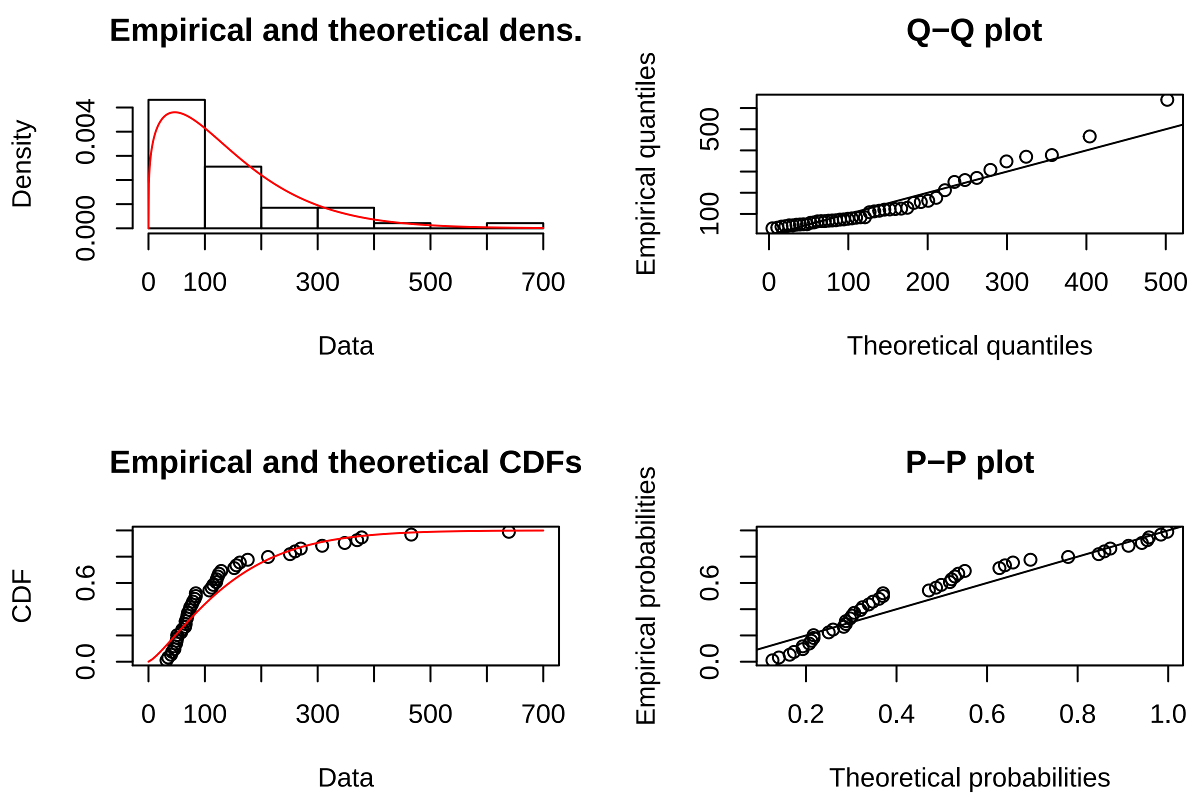
```
plot(distributionFit_Buyer_pois)
```



```r
distributionFit_Buyer_wb$loglik
```

```
## [1] -277.1455
```

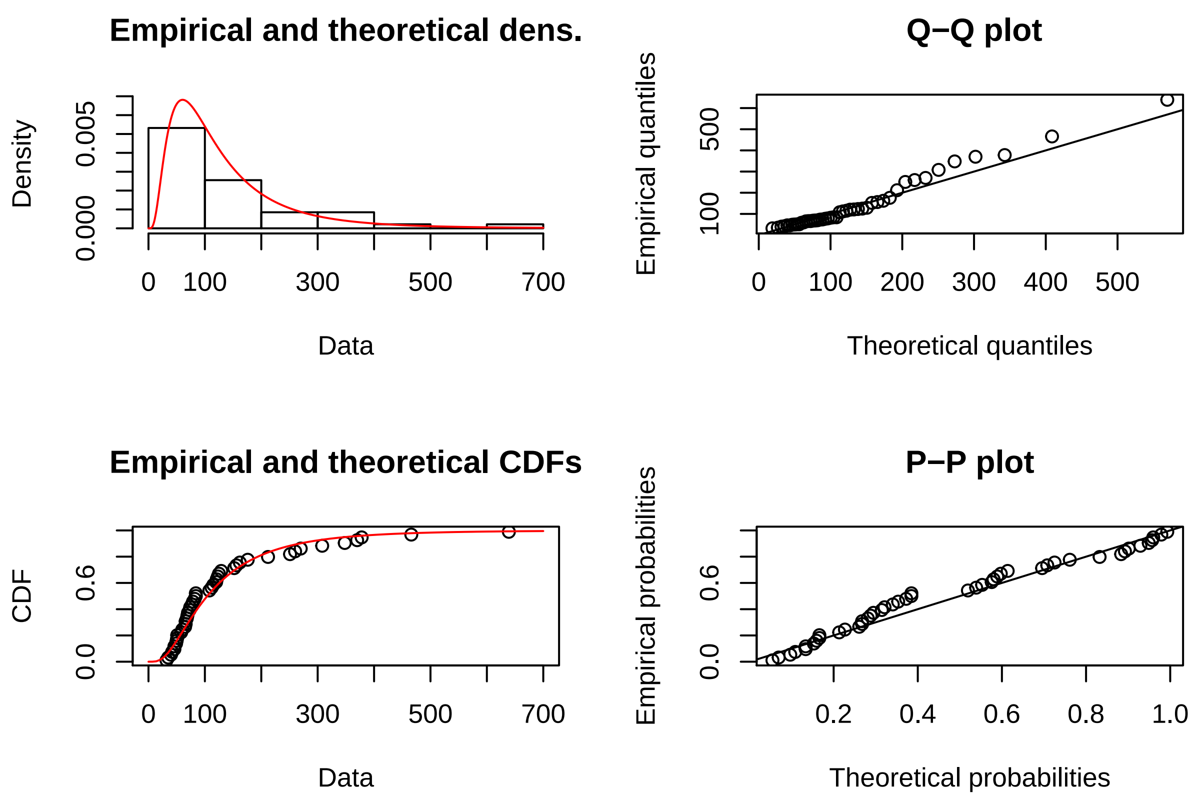
```r
plot(distributionFit_Buyer_wb)
```



```
distributionFit_Buyer_ln$loglik
```

```
## [1] -270.7538
```

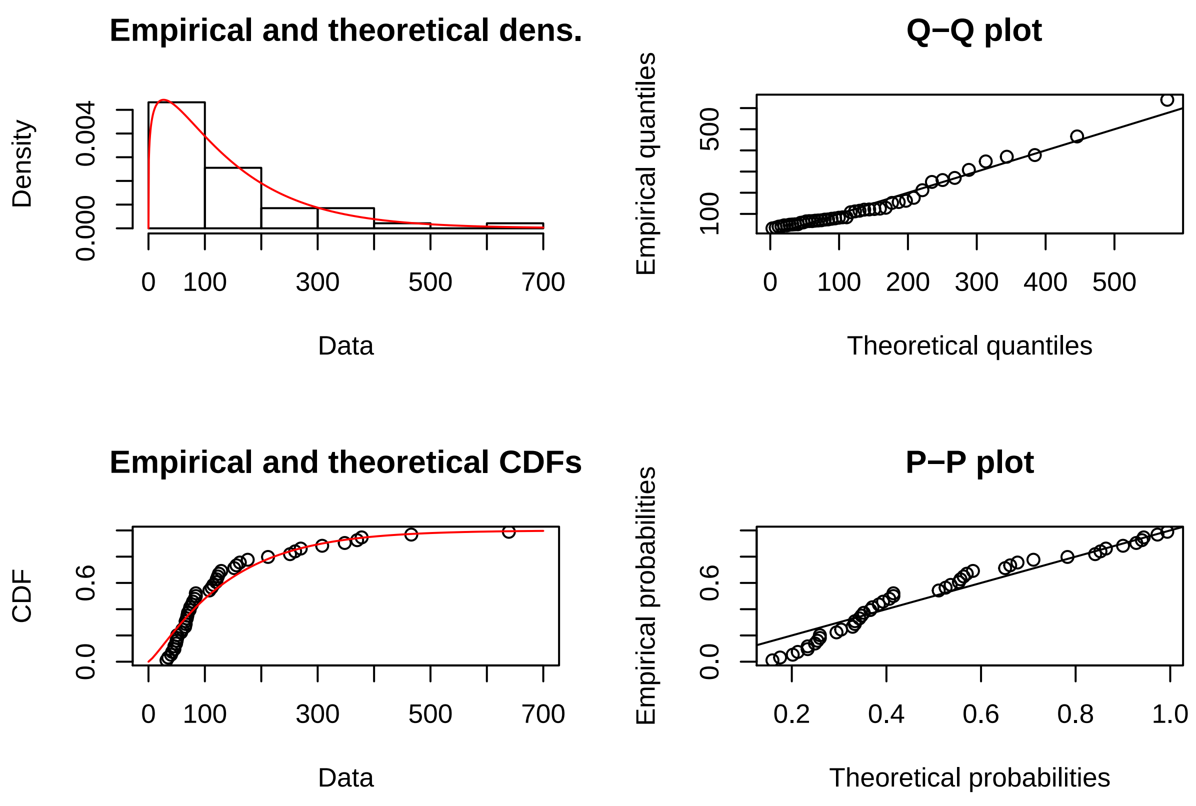
```
plot(distributionFit_Buyer_ln)
```



```
distributionFit_Buyer_gm$loglik
```

```
## [1] -277.1869
```

```
plot(distributionFit_Buyer_gm)
```



Study 1: Conclusion

From the above graph estimates, both buy and sell frequency for our dataset follows LOG-NORMAL distribution as the log likelihood value is maximum and standard error is least and the emperical distribution curve follows the theoritical distribution curve for the log-normal graph most accurately.

Study 2:

Here the aim is to select a feature and select a criteria for creating layers of transaction and finding the correlation of price data for BNB token among the layers. We find the correlation between the unique number of buyers(feature) in each day(layer) to the token opening price for the day. We select the layers to be a day as that is the minimum division available within the dataset. This study will help understanding the degree of correlation between how user activities model with respect to the opening price of the token in each day.

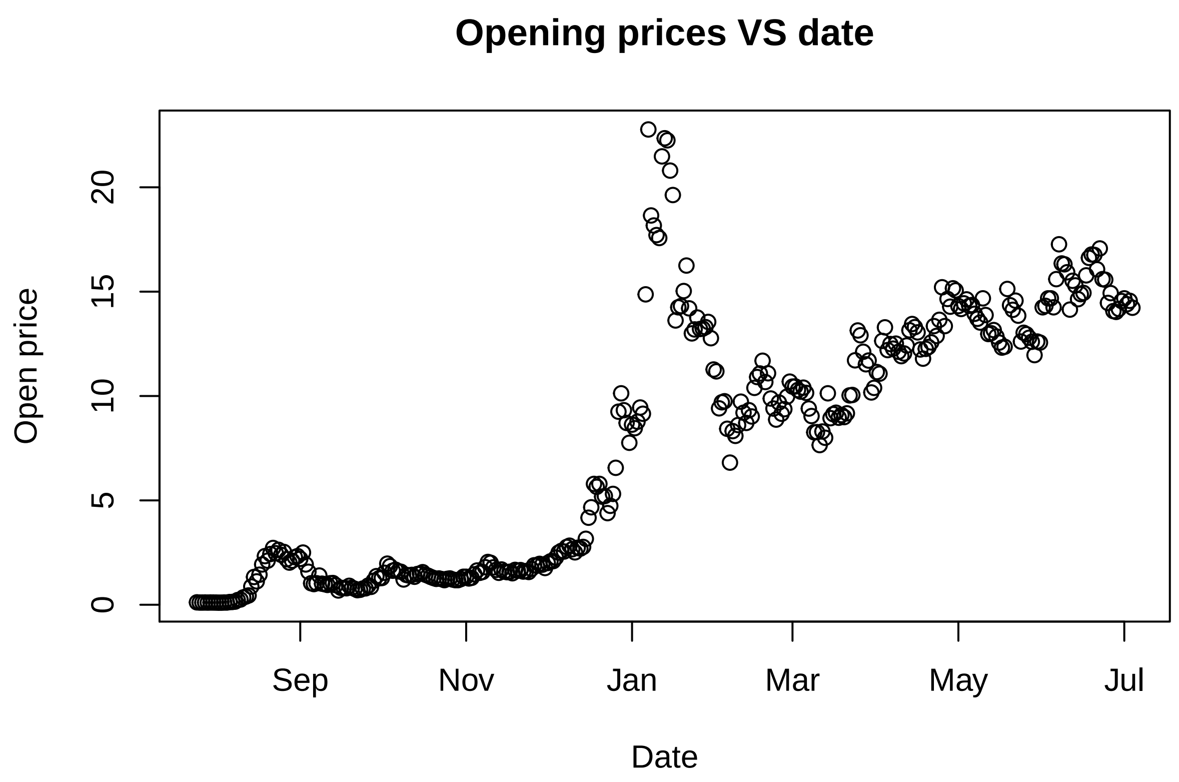
Studying distribution of the opening price .

Study the pattern for opening price values on each day for BNB token. We do not see any outliers in this data.

```
timePrices <- subset(myPrices, select=c("Date","Open"))
timePrices$Date <- as.Date(timePrices$Date, "%Y-%m-%d")
timePrices <- unique(timePrices)
summary(timePrices)
```

```
##       Date                  Open
##  Min.   :2017-07-25   Min.   : 0.09972
##  1st Qu.:2017-10-19   1st Qu.: 1.58000
##  Median :2018-01-13   Median : 8.94000
##  Mean   :2018-01-13   Mean   : 7.75033
##  3rd Qu.:2018-04-09   3rd Qu.:13.14000
##  Max.   :2018-07-04   Max.   :22.77000
```

```
plot(timePrices$Date, timePrices$Open, main = "Opening prices VS date", xlab = "Date", ylab="Open price
```



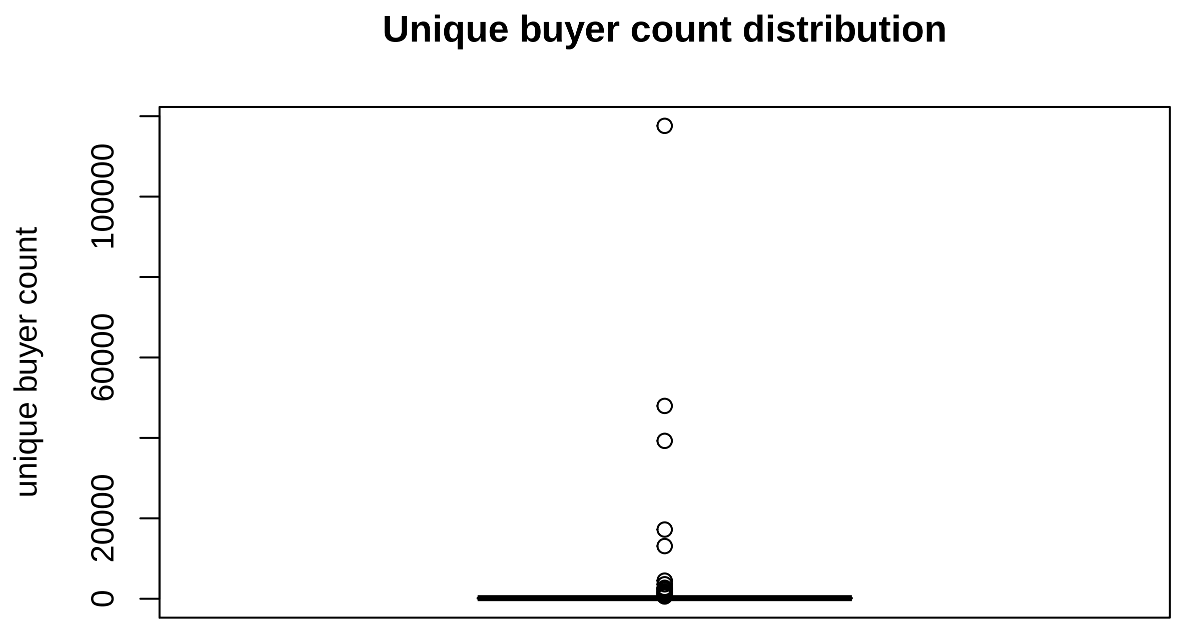
Studying the distribution of number of unique buyers each day.

Study the pattern of how many unique buyers are there for our token each day. We see outliers in this data.

```
timeBuyFreq <- ddply(mydata, .(DATE), mutate, count = length(unique(TONODE)))
timeBuyFreq <- subset(timeBuyFreq, select=c("DATE", "count"))
timeBuyFreq$DATE <- as.Date(timeBuyFreq$DATE, "%Y-%m-%d")
timeBuyFreq <- unique(timeBuyFreq)
summary(timeBuyFreq)
```

```
##       DATE                 count
##  Min.   :2017-07-07   Min.   :     3.00
##  1st Qu.:2017-09-20   1st Qu.:    53.75
##  Median :2017-12-05   Median :   148.00
##  Mean   :2017-12-05   Mean   :  1025.48
##  3rd Qu.:2018-02-19   3rd Qu.:   236.50
##  Max.   :2018-05-06   Max.   :117595.00
```

```
outliers <- boxplot(timeBuyFreq$count, main="Unique buyer count distribution", ylab="unique buyer count
```

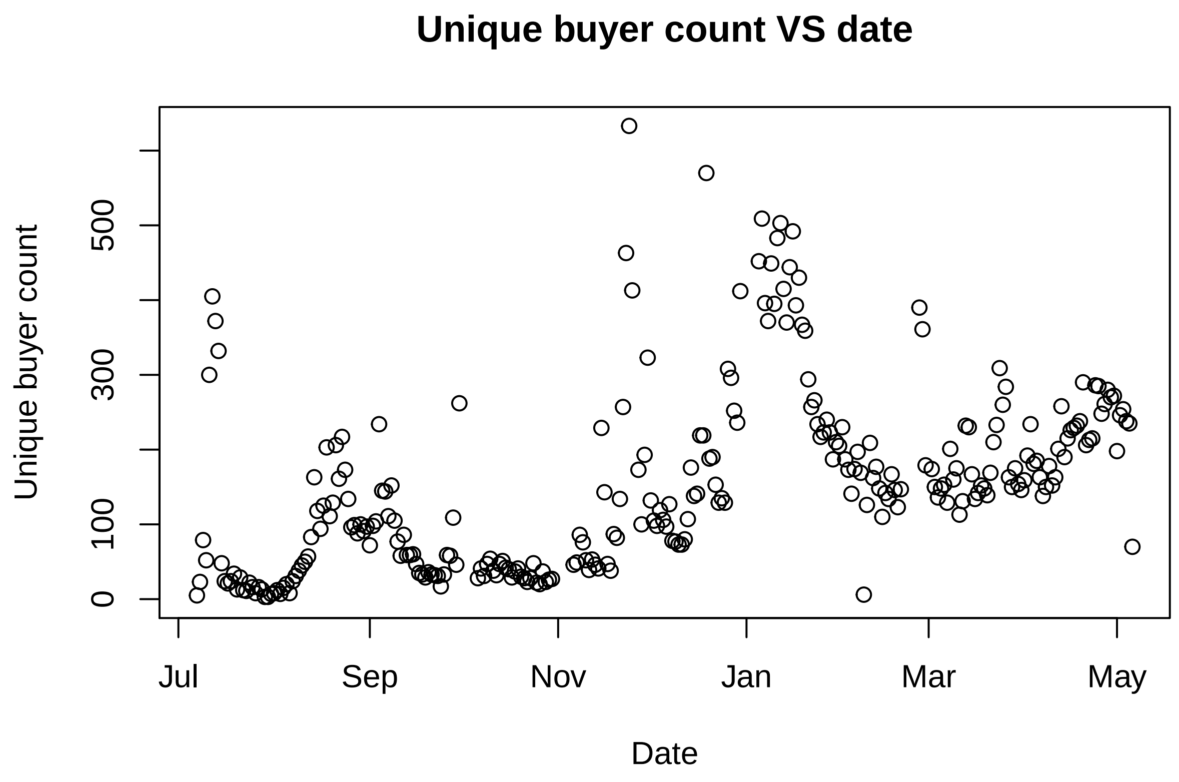


We see the summary of the outliers and plot the data with and without the outliers.

```
summary(outliers)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     570    1118    1746   10785    3600  117595
```

```
plot( timeBuyFreq$DATE, timeBuyFreq$count ,ylim=c(0, 633), main = "Unique buyer count VS date", xlab =
```



Combine opening price and unique buyer count for each day

We remove the outliers and merge the price and buyer counts to find the pearson correlation between the two fields with each day being a layer.

```r
priceSellForEachDay <- merge(x=timePrices, y=timeBuyFreq, by.x=c("Date"), by.y = c("DATE"))
newSet <- remove_outliers(priceSellForEachDay$count)
maxCount = max(newSet[complete.cases(newSet)])
minCount = min(newSet[complete.cases(newSet)])
priceSellForEachDay <- subset(priceSellForEachDay, count<maxCount & count>minCount)
head(priceSellForEachDay)
```

```
##         Date     Open count
## 1 2017-07-25 0.115203    16
## 2 2017-07-26 0.105893     8
## 3 2017-07-27 0.105108    16
## 4 2017-07-28 0.107632    13
## 7 2017-07-31 0.106828     7
## 8 2017-08-01 0.104595     8
```

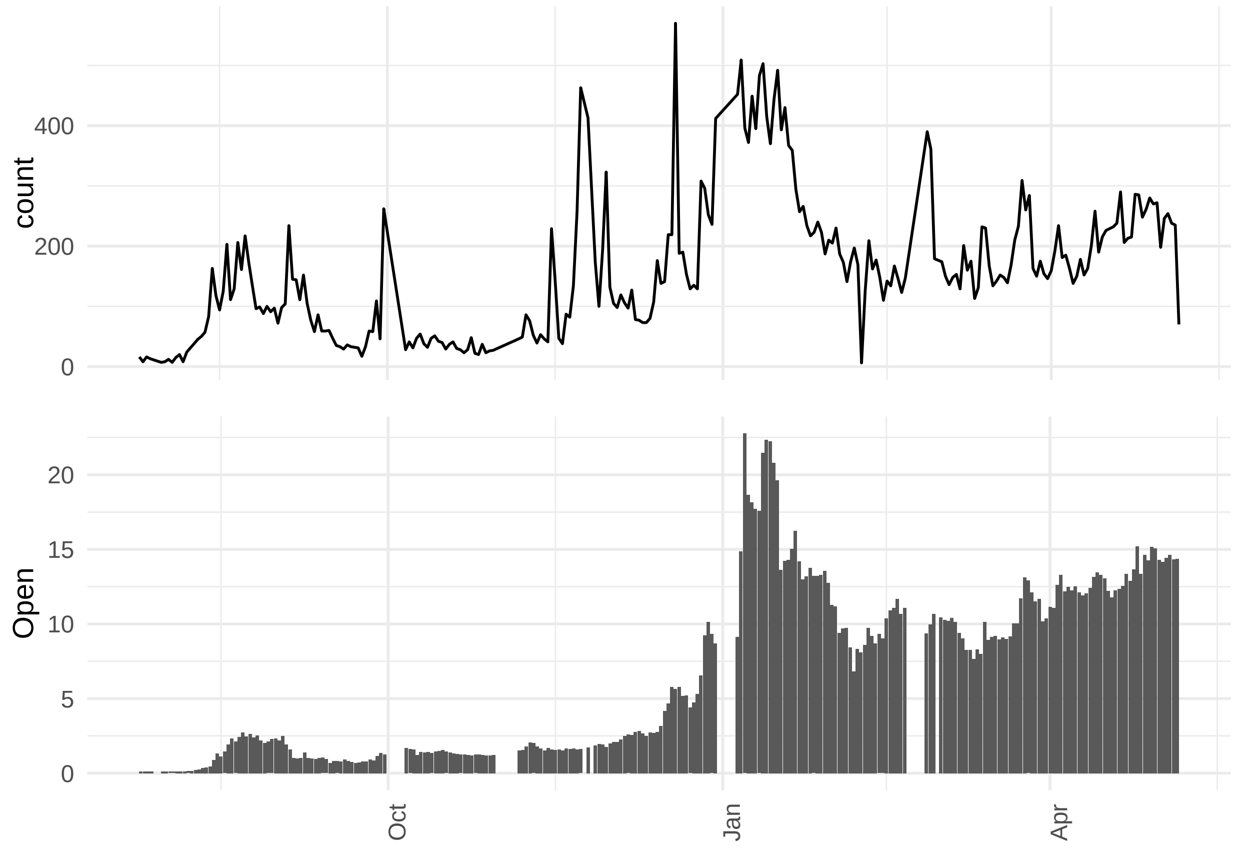
```r
cor(priceSellForEachDay$Open, priceSellForEachDay$count, method=c("pearson"))
```

```
## [1] 0.7335533
```

Study 2: Conclusion

We find a very strong positive correlation between the number of people buying BNB token in a day to the price of the token that day. This menas that user decisions get highly effected by the opening price of bnb token on that day. So we combine both plots to visualize the correlation.

```r
#' Create the two plots.
p1 <- ggplot(priceSellForEachDay, aes(Date, count)) + geom_line() + theme_minimal() +
      theme(axis.title.x = element_blank(), axis.text.x = element_blank())
p2 <- ggplot(priceSellForEachDay,aes(Date, Open)) + geom_bar(stat="identity") + theme_minimal() +
      theme(axis.title.x = element_blank(),axis.text.x = element_text(angle=90))
grid.newpage()
grid.draw(rbind(ggplotGrob(p1), ggplotGrob(p2), size = "last"))
```



Study 3:

We find the most active users in BNB token and try to fit a distribution for their activities among all the tokens throughout given dataset. We take into consideration anyone who is buying or selling the token as we are interested in all activities.

Getting the active users

We first find out the most active users for our token. Active users are selected as those users who buy/sell BNB token more than the average count of all users buying/selling BNB token. This is done to get enough data points for fitting the distrbution later on.

```r
allUsers <- append(mydata$TONODE, mydata$FROMNODE)
allUsers <- data.frame(allUsers)
colnames(allUsers) <- c("USERS")

usersFreq <- count(allUsers, "USERS")
```

```
meanFreq <- mean(usersFreq$freq)
activeUsers <- usersFreq[(usersFreq$freq>meanFreq),]
```

Reading all other token data

We go thorugh all the other tokens in given dataset and find out how many tokens each user buys/sells

```
col_names <- c("FROMNODE","TONODE","DATE","TOKENAMOUNT")
fpath<-"D:/GitHub Projects/statistics-for-DS/Token Graphs"
files <- list.files(path=fpath, pattern="*.txt", full.names=TRUE, recursive=FALSE)

uniqueUsersForAllTokens <- list() #For every token a user transacts in, there is one entry of the userI
for(i in 1:length(files)){
  t <- data.frame(read.csv( files[i], header = FALSE, sep = " ", dec = ".", col.names = col_names))
  tusers <- unique(append(t$FROMNODE, t$TONODE))
  uniqueUsersForAllTokens <- append(uniqueUsersForAllTokens,tusers)
}
usersFromAllTokens <- do.call(rbind.data.frame, uniqueUsersForAllTokens)
colnames(usersFromAllTokens)<-c("USERID")
```

Counting the number of tokens per userid

We now reverse the frequency count, to find how many unique tokens each user id is transacting in. Below is a snipet of resulting data:

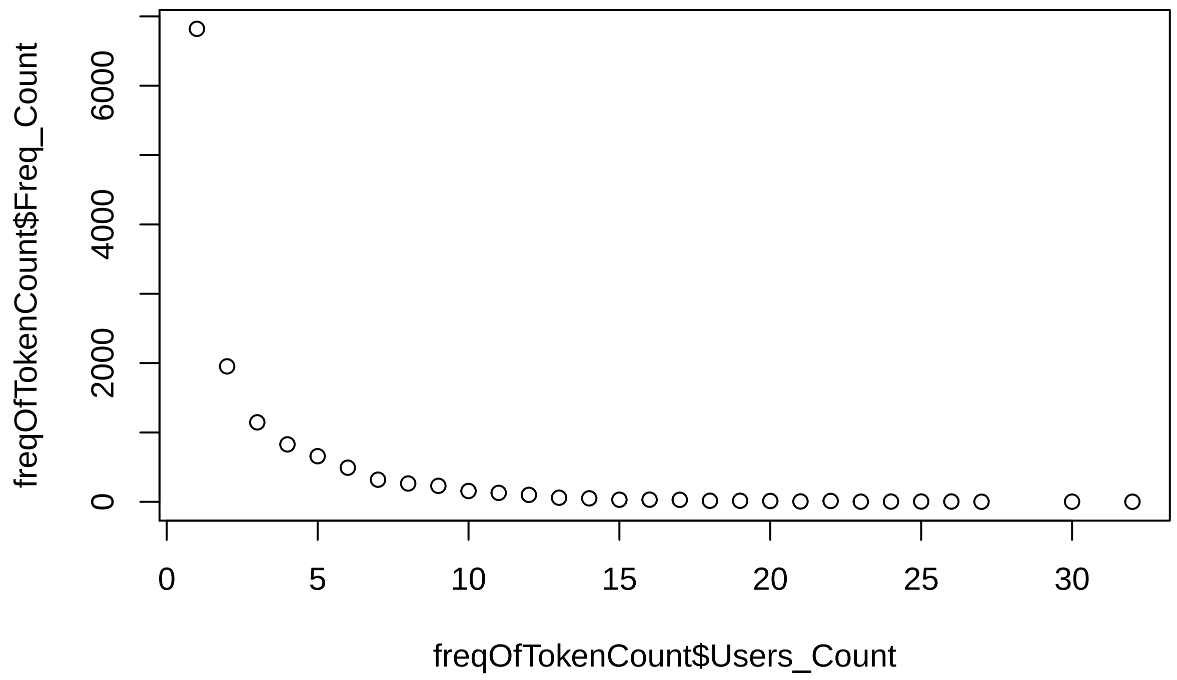
```
userTokenCount <- data.frame(table(usersFromAllTokens$USERID[usersFromAllTokens$USERID %in% activeUsers
colnames(userTokenCount) <-  c("USERID", "COUNT")
head(userTokenCount)
```

```
##   USERID COUNT
## 1      5    30
## 2      6    24
## 3      8    22
## 4     35    21
## 5     36     4
## 6     44    25
```

```
freqOfTokenCount <- count(userTokenCount, "COUNT")
colnames(freqOfTokenCount) <- c("Users_Count", "Freq_Count")
head(freqOfTokenCount)
```

```
##   Users_Count Freq_Count
## 1           1       6820
## 2           2       1953
## 3           3       1146
## 4           4        828
## 5           5        658
## 6           6        492
```

```
plot(freqOfTokenCount$Users_Count, freqOfTokenCount$Freq_Count)
```

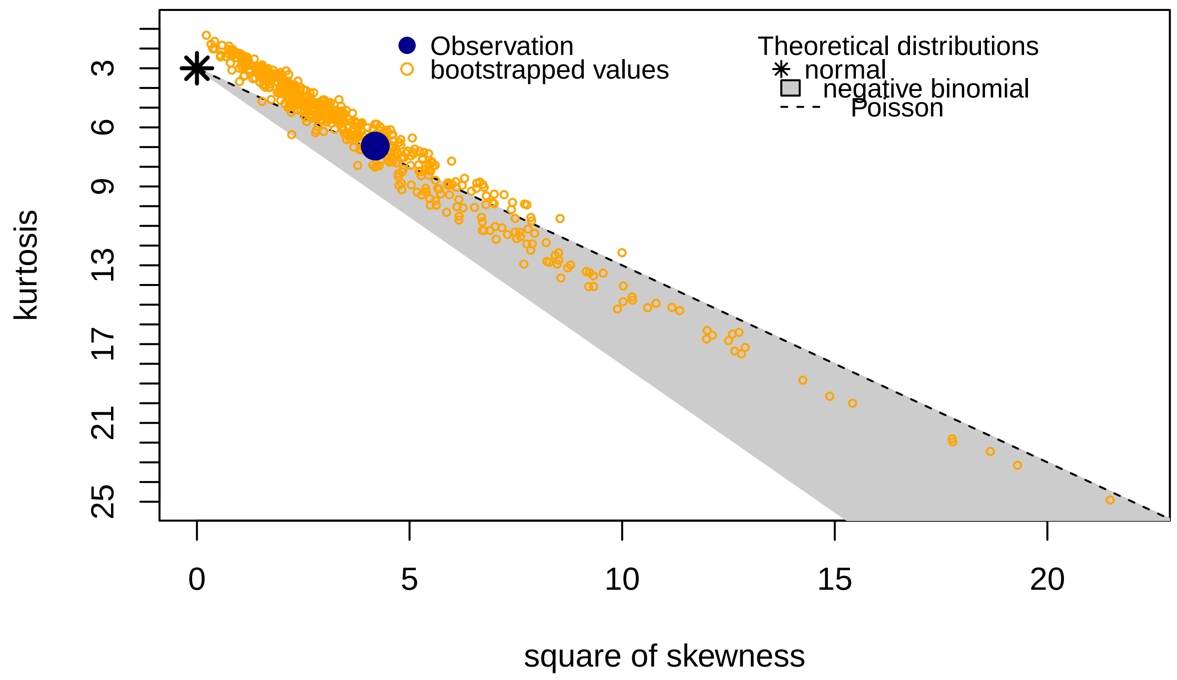


Removing outliers and summarizing unique token counts for the active users

We use the same Culley and Fray graph software to find the distributions our data approximately fits into.

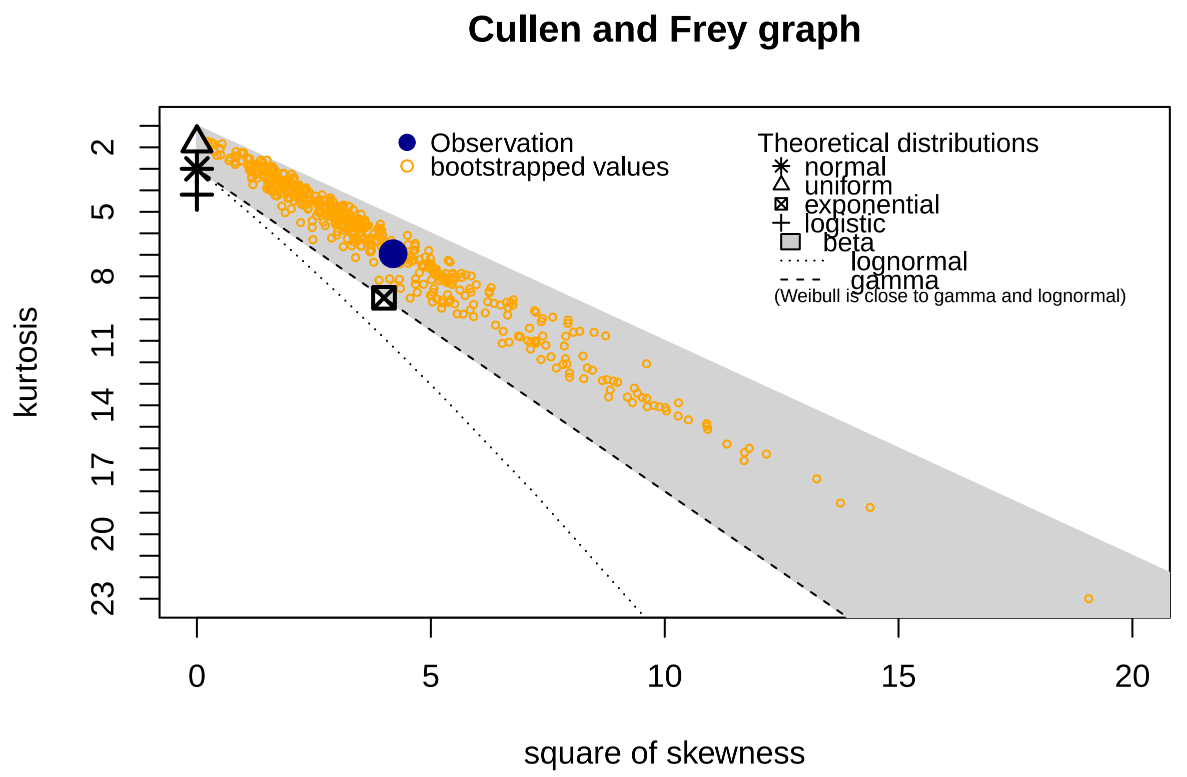
```
newSet3 <- remove_outliers(freqOfTokenCount$Freq_Count)
maxCount3 = max(newSet3[complete.cases(newSet3)])
minCount3 = min(newSet3[complete.cases(newSet3)])
freqOfTokenCount <- subset(freqOfTokenCount, Freq_Count<maxCount3 & Freq_Count>minCount3)
descdist(freqOfTokenCount$Freq_Count, boot= 500, discrete=TRUE)
```

Cullen and Frey graph



```
## summary statistics
## ------
## min:  2   max:  658
## median:  30
## mean:  113.9565
## estimated sd:  173.3594
## estimated skewness:  2.04776
## estimated kurtosis:  6.948924
```

```
descdist(freqOfTokenCount$Freq_Count, boot= 500)
```



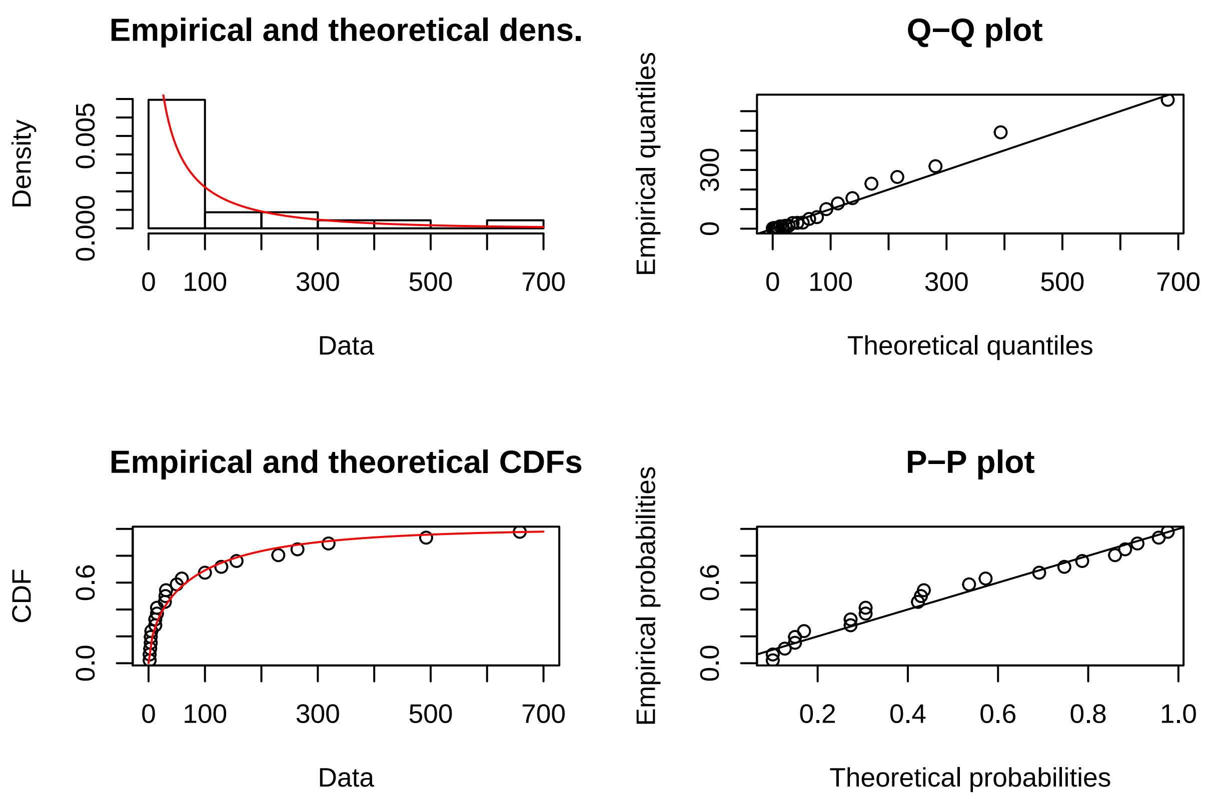
```
## summary statistics
## ------
## min:  2   max:  658
## median:  30
## mean:  113.9565
## estimated sd:  173.3594
## estimated skewness:  2.04776
## estimated kurtosis:  6.948924
```

Fitting the distrubution to find the closest fit

```
distributionFit_Count_pois <- fitdist(freqOfTokenCount$Freq_Count, "pois", method ="mle")
distributionFit_Count_wb <- fitdist(freqOfTokenCount$Freq_Count, "weibull", method ="mle")
distributionFit_Count_ln <- fitdist(freqOfTokenCount$Freq_Count, "lnorm", method ="mle")
distributionFit_Count_gm <- fitdist(freqOfTokenCount$Freq_Count, "gamma" ,method="mme")
distributionFit_Count_wb$loglik
```

```
## [1] -126.1784
```

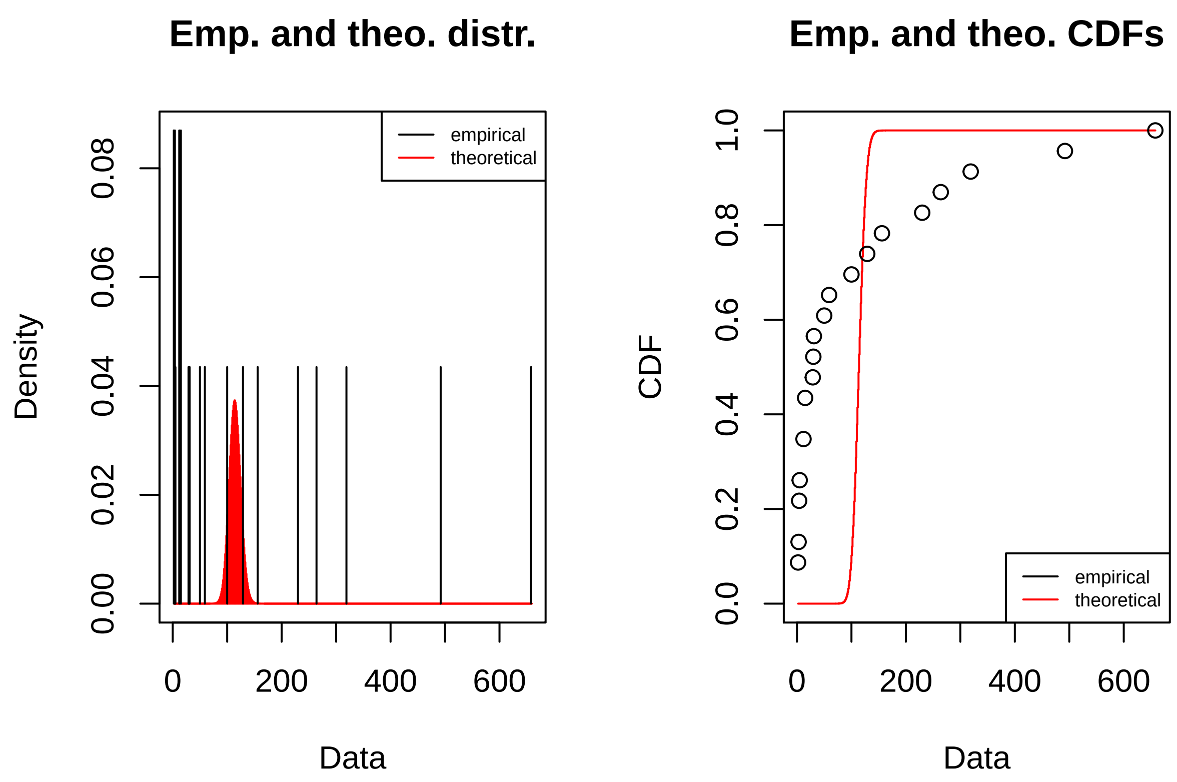
```
plot(distributionFit_Count_wb)
```



```
distributionFit_Seller_pois$loglik
```

```
## [1] -20956.41
```

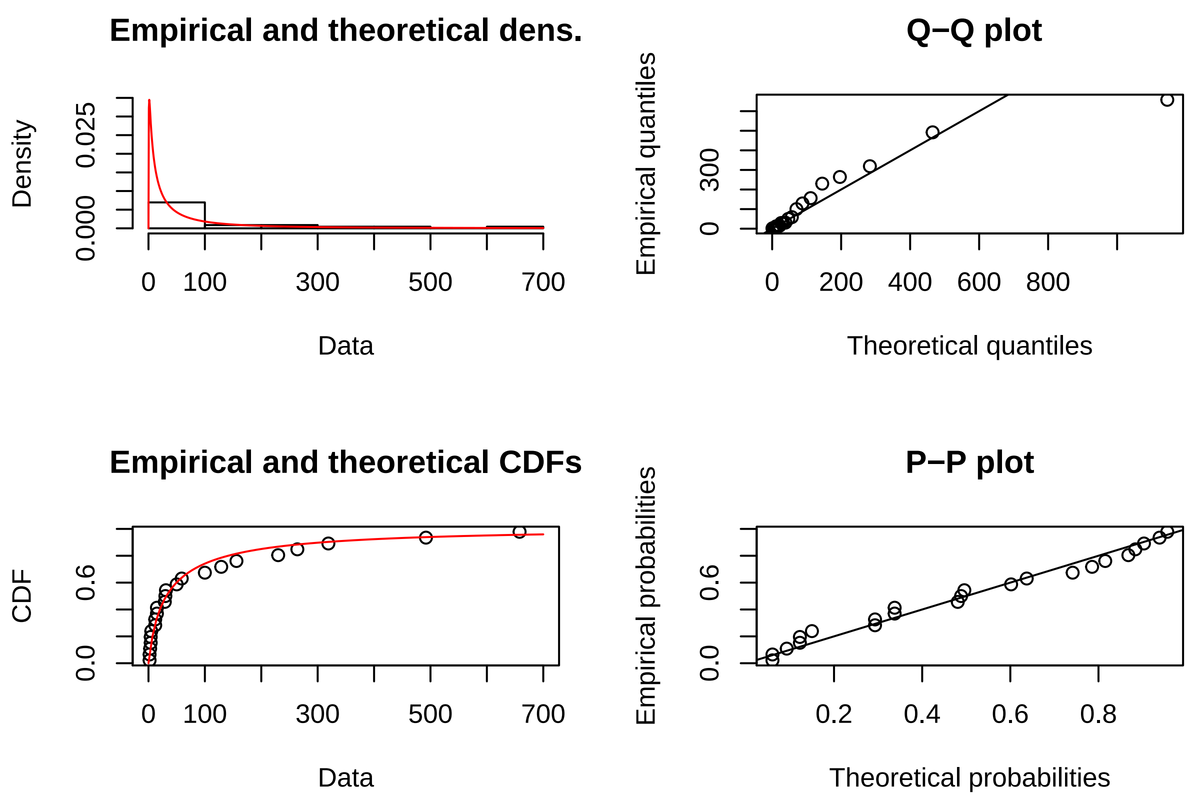
```
plot(distributionFit_Count_pois)
```



```r
distributionFit_Seller_ln$loglik
```

```
## [1] -2195.726
```

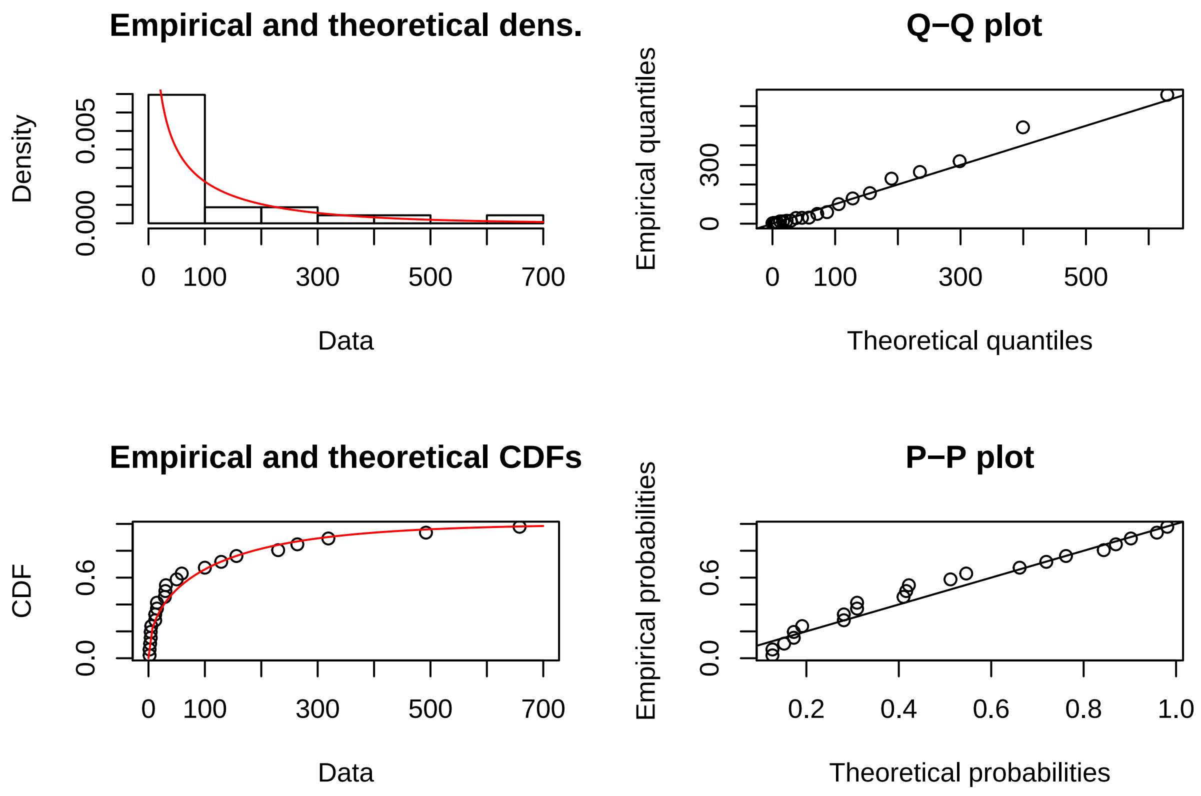
```r
plot(distributionFit_Count_ln)
```



```
distributionFit_Seller_gm$loglik
```

```
## [1] -2199.702
```

```
plot(distributionFit_Count_gm)
```



Study 3: Conclusion

From the above graph estimates, the number of unique tokens in our dataset in which the most active users of BNB token transact, follows a weibull distribution as the log likelihood value is maximum, also the emperical distribution curve follows the theoritical distribution curve for the log-normal graph most accurately.