CS6313.002 Statistical Project 1 - Part 1

Ruolan Zeng(rxz171630), Vineet Vilas Amonkar(vva180000)

Nov 30 2018

Abstract

The aim of the project is to implement different statistical methods on the Storj Token transaction data.The results of the methods are analyzed to derive inferences about the token data.In the first part, we find how many times a user buys or sells the STORJ token, then fit a distribution and estimate the best distribution.In the second part we find the correlation between the STORJ Token price data and layer features of the STORJ Token.In the third part of we find the most active buyer and seller of our STORJ token and then track them in all other tokens. We find their plot of how many unique tokens have they invested in the provided time frame then fit and estimate distributions as part 1.

1. Introduction:

1.1 Important Concepts

Before we get into the project details it is important to note few concepts. Below we summarise these terms:

1.1.1 Blockchain:

It is a public ledger formed of multiple blocks.Each block contains crytographic hash of the previous block, transaction data, timestamp and other metadata.The blockchain is a chain of these blocks linked together cryptographically and stored on a distributed peer-to-peer network.

1.1.2 Ethereum:

Ethereum is a blockchain platform for digital currency and for builiding decentralized applications using smart contracts. Smart Contract is a set of instructions of the format if-this-then-that, it is formed when someone needs to performing particular task involving one or more than one entities of the blockchain. The contract is a code which executes itself on occurence of a triggering event such as expiration date.The smart contracts can be written with different languages such as solidity The EVM is a runtime environment for smart contracts in Ethereum. Every Ethereum node in the network runs an EVM implementation and executes the same instructions. Ether is the digital currency(cryptocurrency) of Ethereum. Every individual transaction or step in a contract requires some computation. To perform any computation user has to pay a cost calulated in terms of 'Gas' and paid in terms of 'Ether'. The Gas consist of two parts:

Gas limit: It is the amount of units of gas Gas price: It is the amount paid per unit of gas

1.1.3 Ethereum ERC-20 Tokens:

If a user needs some service provided by the DAPPS, then he has to pay for that service in terms of 'token' associated with the DAPPS. These Ethereum tokens can be bought using Ether or other cryptocurrencies and can serve the following two purposes:

1) Usage Token: These tokens are used to pay for the services of the Dapp
2) Work Token: These tokens identify you as a shareholder in the DAPP

ERC-20 is a technical standard used for smart contracts on the Ethereum Blockchain for implementing tokens. It is a common list of rules for Ethereum token regarding interactions between tokens,transferring tokens between addresses and how data within th e token is accessed

1.2 Project Primary Token: Storj

1.2.1 Storj:

Storj is a decentralized cloud storage network. It uses a blockchain based hash table to store files on a peer-to-peer network. It consists of two entities farmers and tenants:

Tenants: They are the users who upload their data to cloud storage. Farmers: They are the different nodes in the network who host the data.

Characteristics of Storj:
1) Sharding : data is split into different parts and no one farmer has the entire data stored on his node.
2) End-to-End encryption: The files are encrypted before uploading.
3)File Verification: Regular Audits are performed to verify the files
4)Redundancy

Distributed Hash table (DHT): It consist of key-value pairs. The keyspace is split among the participating nodes. One method of DHT implementation: The name of the file is hashed to a key. The ownership of this key lies with one of the nodes. The network is traversed for that node which then stores that data and key. While retrieval the filename is hashed again to give the same key and that node is searched again which provides the corresponding value.

1.2.2 Storj Token:

STORJ tokens are utilized as the payment method on the STORJ network. Details (11/2/2018):

1 STORJ = $0.334425 USD
Market Cap = $45,114,748 USD
Circulating Supply = 135,787,439 STORJ
Total Supply = 424,999,998 STORJ
Subunits = 10^8

2. Data Description

The Data used for the project is divided in two parts:

1) Token network edge files:

There are 40 Token network edge files.Token edge files have 4 columns: from_node, to_node, unix-time, total-amount.For each row it implies that from_node sold total-amount of the token to to_nodeid at time unix-time.For Part 1,2 and 4 of the project we will only use the STORJ token network edge file. For part 3 we will use the token network edge files of all 40 tokens.

2) Token price files:

Structure of price data: Date, Open, High, Low, Close, Volume, MarketCap.Open and Close are the prices of the specific token at a given date. Volume and MarketCap give total bought/sold tokens and market valuation at the date. We use price data in part 2 and 4.

3. Preprocessing

3.1 Removing Outliers

There could be some records in the transactions where total amount is very large. Some of this records could be due to some bug or glitch(integer overflow problem). These values can be separated from data using a threshold value.

Calculating this value for the STORJ token:The value of transaction amount can't be greater than the max value where, max value = total supply of tokens * subunits.subsituting the values from above.

```
# Load Data
tokenData <- read.delim("networkstorjTX.txt", header = FALSE, sep = " ")
tokenFrame<- as.data.frame(tokenData)
colnames(tokenFrame) <- c("fromNode","toNode","Date","totalAmount")

# Find Outliers:
TotalSupply <- 424999998
Decimals <- 10^8
OutlierValue <- TotalSupply*Decimals
Outlierdata <- tokenFrame[ which(tokenFrame$totalAmount > OutlierValue),]

# Total Number Of Outliers:
message("Total number of outliers: ", length(Outlierdata$totalAmount))
```

```
## Total number of outliers: 53
```

```
# How Many Users Are Included In Outliers Transactions:
users <- c(Outlierdata$fromNode,Outlierdata$toNode)
uniqueUsers<- unique(users)
message(length(uniqueUsers), " users are includednin outliers transactions")
```

```
## 14 users are includednin outliers transactions
```

```
# Remove Outliers
WithoutOutlierdata <- tokenFrame[ which(tokenFrame$totalAmount < OutlierValue),]
```

4.Token Data Analysis

4.1 Q1

4.4.1 Fit Distribution – Buys

The package we use to fit distribution: fitdistrplus, provide the function fitdist() we can use to fit distribution of our data. **The function we use to fit distribution:** fitdist(). Fit of univariate distributions to different type of data with different estimate method we can choose: maximum likelihood estimation (mle), moment matching estimation (mme), quantile matching estimation (qme), maximizing goodness-of-fit estimation (mge), the default and we mostly used one is MLE. Output of this function is S3 object, we can use several methods like plot(), print(), summary() to visualize it or get more detailed information.We use plot in this project.

```
frequencyTable <- table(WithoutOutlierdata[2])
freq<- as.data.frame(frequencyTable)
colnames(freq) <- c("buyerID","frequency")
FrequencyBuyers = table(freq$frequency)
freqNoBuys = as.data.frame(FrequencyBuyers)
```

```r
colnames(freqNoBuys) <- c("NoBuys","freqNoBuys")

# barplot(freqNoBuys$freqNoBuys,names.arg = freqNoBuys$NoBuys,ylab = "Frequency of Number of Buyers", x
```

```r
# fit distribution

# Poisson Distribution
library(fitdistrplus)
```

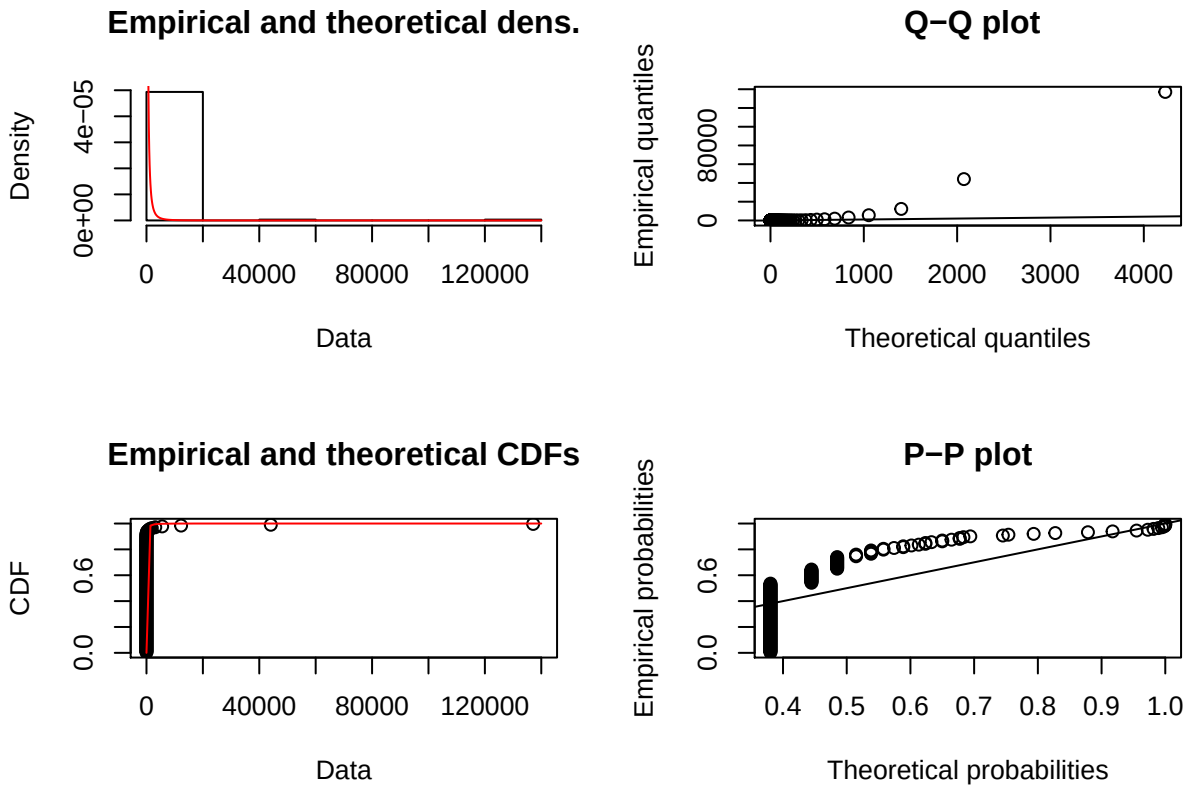
```
## Loading required package: MASS
```

```
## Loading required package: survival
```

```
## Loading required package: npsurv
```

```
## Loading required package: lsei
```

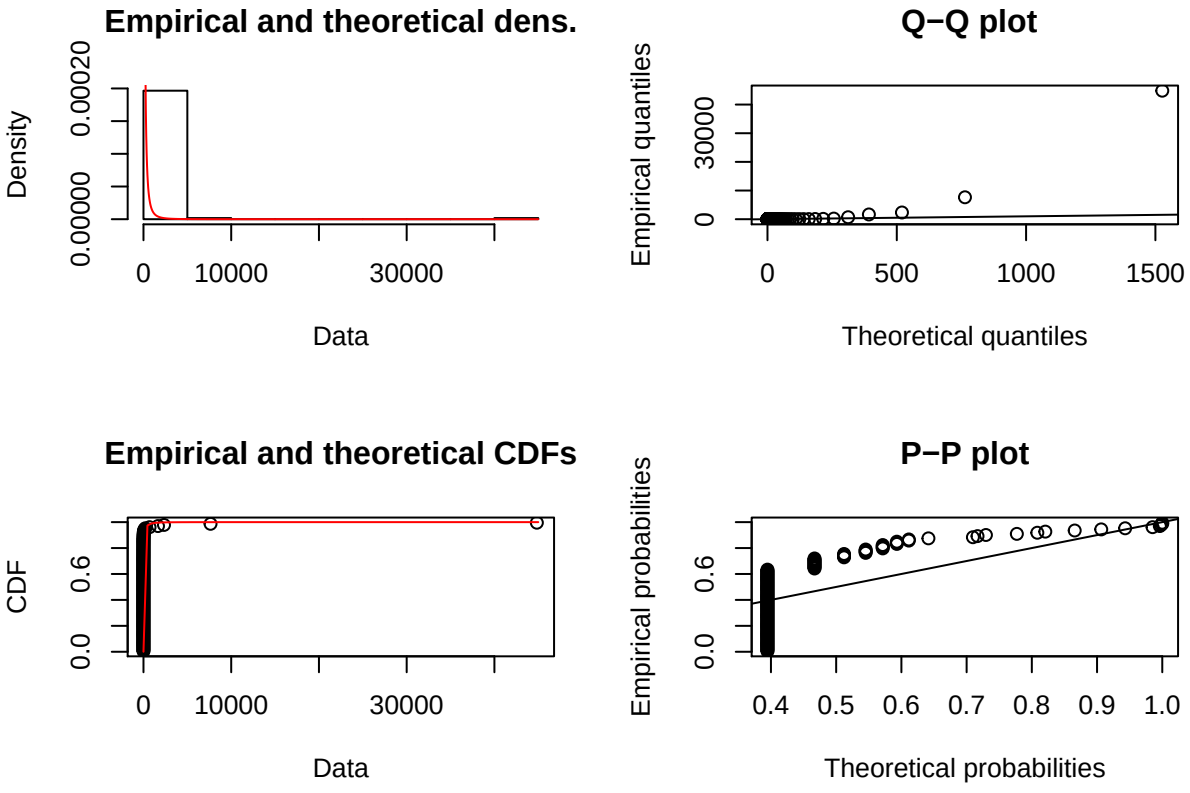
```r
fit <- fitdist(freqNoBuys$freqNoBuys, "pois", method="mle")
# Weibull Distribution
fit2 <- fitdist(freqNoBuys$freqNoBuys, "weibull",method = "mle")
# Exponential Distribution
fit3 <- fitdist(freqNoBuys$freqNoBuys, "exp",method = "mme")
# Geometric Distribution
fit4 <- fitdist(freqNoBuys$freqNoBuys, "geom",method = "mme")
# Normal Distribution
fit5 <- fitdist(freqNoBuys$freqNoBuys, "norm")

plot(fit2)
```



4.1.2 Fit Distribution – Sells

```r
frequencyTable <- table(WithoutOutlierdata[1])
freq<- as.data.frame(frequencyTable)
colnames(freq) <- c("sellerID","frequency")
FrequencySellers = table(freq$frequency)
freqNoSells = as.data.frame(FrequencySellers)
colnames(freqNoSells) <- c("NoSells","freqNoSells")

# barplot(freqNoSells$freqNoSells,names.arg = freqNoSells$NoSells,ylab = "Frequency of Number of Seller
```

```r
# fit distribution

# Poisson Distribution
library(fitdistrplus)
fitSells <- fitdist(freqNoSells$freqNoSells, "pois", method="mle")
# Weibull Distribution
fitSells2 <- fitdist(freqNoSells$freqNoSells, "weibull",method = "mle")
# Exponential Distribution
fitSells3 <- fitdist(freqNoSells$freqNoSells, "exp",method = "mme")
# Geometric Distribution
fitSells4 <- fitdist(freqNoSells$freqNoSells, "geom",method = "mme")
# Normal Distribution
fitSells5 <- fitdist(freqNoSells$freqNoSells, "norm")

plot(fitSells2)
```



4.1.3 Conclusion

We totally tried 5 distributions: Poisson Distribution, Weibull Distribution, Exponential Distribution,

Geometric Distribution, Normal Distribution. After compare with parameters and graphs of different distributions, we can find: The distribution of buyers and sellers are similiar. The estimate parameters of Normal Distribution are very close to the parameters of our token data, but the graphs don't fit well. The graphs of Poisson Distribution and Weibull Distribution both fit our token data well, but for the parameters, the mean(buys:1331.66, sells:499.26) is not equal to variance(buys:133341824.3, sells:17815729.69), so it should not be Poisson Distribution.

Hence, we finally conclude that **Weibull Distribution** fit our distribution best.

4.2 Q2

4.2.1 Calulate Correlations

```
# Change the Date Format
WithoutOutlierdata$Date <- as.Date(as.POSIXct(WithoutOutlierdata$Date,origin="1970-01-01",tz="GMT"))

# Load Price Date and Change the Date Format
priceData <- read.delim("storj", header = TRUE, sep = "\t")
priceData$Date <- as.Date(priceData$Date,"%m/%d/%Y")

# Join Two Data by Date
CombineData <- merge(x = WithoutOutlierdata, y = priceData, by = "Date")

## Create layers and calulate correlations
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:MASS':
##
##     select
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
alpha <- c(1e-14, 1e-12, 1e-10, 1e-9, 4e-9, 5e-9, 6e-9, 8e-9, 1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2)
maxValue <- max(WithoutOutlierdata$totalAmount)

for(a in alpha){
  value <- a*maxValue
  layer <- CombineData[ which(CombineData$totalAmount < value),]
  result<-layer %>% select(Date,fromNode,Open,High,Low,Close,Volume) %>%
    group_by(Date) %>%
    mutate(fromNode = n())
  result <- unique(result)
  colnames(result) <- c('Date','TransNo','Open','High','Low','Close','Volume')
  correlation <- cor(result$TransNo, result$Open,use="complete.obs",method = "pearson")
  correlation2 <- cor(result$TransNo, result$High,use="complete.obs",method = "pearson")
  correlation3 <- cor(result$TransNo, result$Low,use="complete.obs",method = "pearson")
  correlation4 <- cor(result$TransNo, result$Close,use="complete.obs",method = "pearson")
```

```
  correlation5 <- cor(result$TransNo, as.numeric(result$Volume),use="complete.obs",method = "pearson")
  print(correlation3)
}
```

```
## [1] -0.2476416
## [1] -0.215383
## [1] -0.107761
## [1] 0.009517341
## [1] 0.1329595
## [1] 0.1719883
## [1] 0.1904545
## [1] 0.1928889
## [1] 0.1864201
## [1] 0.1520662
## [1] 0.1568887
## [1] 0.1687585
## [1] 0.176851
## [1] 0.1782314
## [1] 0.178309
```

4.2.2 Conclusion

The number of layers is 15. The best correlation is 0.193.

4.3 Q3

4.3.1 Buyer

```
# Find most active buyer in our token:
frequencyTable <- table(WithoutOutlierdata[2])
freq<- as.data.frame(frequencyTable)
colnames(freq) <- c("buyerID","frequency")
mostFreq <- freq[which(freq$frequency== max(freq$frequency)),]
buyer <- mostFreq$buyerID
message("Id of the most active buyer in our token: ", buyer)
```

```
## Id of the most active buyer in our token: 5
```

```
# Plot the number of transactions of each month of buyer5:
user5 <- WithoutOutlierdata[which(WithoutOutlierdata$toNode == buyer),]
user5$Date <- as.Date(as.POSIXct(user5$Date,origin="1970-01-01",tz="GMT"))
user5$Date <- cut(user5$Date, breaks = "month")
frequencyTable5 <- table(user5[3])
freq5 <-  as.data.frame(frequencyTable5)
colnames(freq5) <- c("time","frequency")
# barplot(freq5$frequency,names.arg = freq5$time,ylab = "number of transactions", xlab = "month")
```

```
# Load all token data
filenames <- list.files("tokens", pattern="*.txt", full.names=TRUE)

#find the minimum and max date of all token data
newfreq5 <- freq5[order(freq5$time),]
minDate <- as.Date(newfreq5$time[1])
maxDate <- as.Date(newfreq5$time[nrow(newfreq5)])
```

```
for (file in filenames){
  token <- read.delim(file, header = FALSE, sep = " ")
  Frame<- as.data.frame(token)
  colnames(Frame)<-c('fromNode','toNode','unixTime','totalAmount')
  user5_other <- Frame[which(Frame$toNode == buyer),]
  if (nrow(user5_other)>0){
    user5_other$unixTime <- as.Date(as.POSIXct(user5_other$unixTime,origin="1970-01-01",tz="GMT"))
    user5_other$unixTime <- cut(user5_other$unixTime, breaks = "month")
    frequencyTable5_other <- table(user5_other[3])
    freq5_other <-  as.data.frame(frequencyTable5_other)
    colnames(freq5_other) <- c("time","frequency")
    newfreq5_other <- freq5_other[order(freq5_other$time),]
    tempminDate <- as.Date(newfreq5_other$time[1])
    tempmaxDate <- as.Date(newfreq5_other$time[nrow(newfreq5_other)])
    if (tempminDate<minDate){
      minDate <- tempminDate
    }
    if (tempmaxDate>maxDate){
      maxDate <- tempmaxDate
    }
  }
}

# Base on the minmum and maxmum date, create an array of counter and date intervals
countArray <- c(0,0,0,0,0,0,0,0,0,0)
dateinterval <- c("2017-08-01","2017-09-01","2017-10-01","2017-11-01","2017-12-01","2018-01-01","2018-0
dateinterval <- as.Date(dateinterval)

# Count the number of unique tokens that buyer5 bought each month
for (file in filenames){
  token2 <- read.delim(file, header = FALSE, sep = " ")
  Frame2<- as.data.frame(token2)
  colnames(Frame2)<-c('fromNode','toNode','unixTime','totalAmount')
  user5_other2 <- Frame2[which(Frame2$toNode == buyer),]

  if (nrow(user5_other2)>0){
    user5_other2$unixTime <- as.Date(as.POSIXct(user5_other2$unixTime,origin="1970-01-01",tz="GMT"))
    user5_other2$unixTime <- cut(user5_other2$unixTime, breaks = "month")
    frequencyTable5_other2 <- table(user5_other2[3])
    freq5_other2 <-  as.data.frame(frequencyTable5_other2)
    colnames(freq5_other2) <- c("time","frequency")
    freq5_other2$time <- as.Date(freq5_other2$time)
    for (arow in 1:nrow(freq5_other2))
    {
      if (freq5_other2$time[arow]==dateinterval[1]){
        countArray[1] = countArray[1]+1

      }
      else if (freq5_other2$time[arow]==dateinterval[2]){
        countArray[2] = countArray[2]+1
      }
      else if (freq5_other2$time[arow]==dateinterval[3]){
        countArray[3] = countArray[3]+1
```

```
      }
      else if (freq5_other2$time[arow]==dateinterval[4]){
        countArray[4] = countArray[4]+1
      }
      else if (freq5_other2$time[arow]==dateinterval[5]){
        countArray[5] = countArray[5]+1
      }
      else if (freq5_other2$time[arow]==dateinterval[6]){
        countArray[6] = countArray[6]+1
      }
      else if (freq5_other2$time[arow]==dateinterval[7]){
        countArray[7] = countArray[7]+1
      }
      else if (freq5_other2$time[arow]==dateinterval[8]){
        countArray[8] = countArray[8]+1
      }
      else if (freq5_other2$time[arow]==dateinterval[9]){
        countArray[9] = countArray[9]+1
      }
      else if (freq5_other2$time[arow]==dateinterval[10]){
        countArray[10] = countArray[10]+1
      }
      else {
        print("error")
      }
    }
  }
}

# plot
dateinterval2 <- c("Aug-17","Sep-17","Oct-17","Nov-17","Dec-17","Jan-18","Feb-18","Mar-18","Apr-18","Ma
# barplot(countArray,names.arg = dateinterval2,ylab = "no of unique tokens", xlab = "month")
```

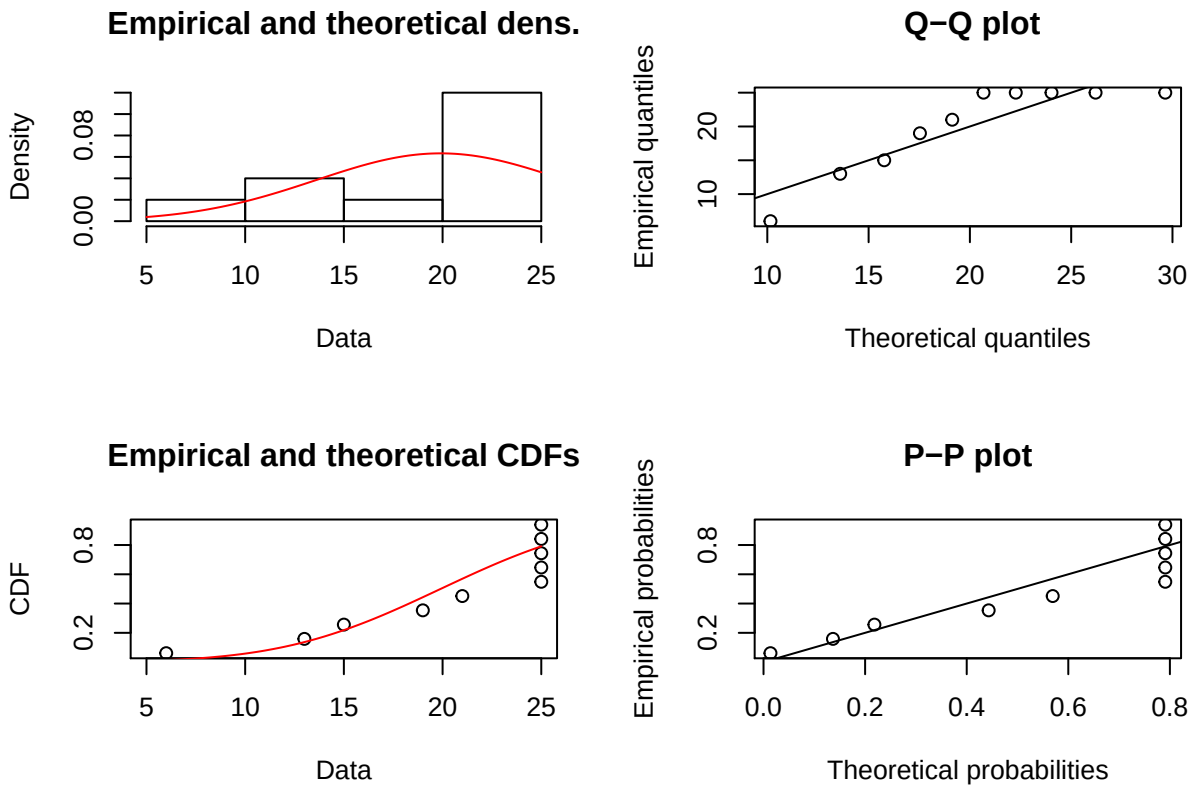
```
# fir distribution

# 1.Exponential Distribution
library(fitdistrplus)
fit <- fitdist(countArray, "exp", method="mle")
# ks.test(countArray,"pexp",0.05,alternative = c("two.sided", "less", "greater"), exact = NULL)

# 2.Weibull Distribution
fit2 <- fitdist(countArray, "weibull",method = "mle")
# ks.test(countArray,"pweibull",3.938559,alternative = c("two.sided", "less", "greater"), exact = NULL)

# 3.Poisson Distribution
fit3 <- fitdist(countArray, "pois", method="mle")
# ks.test(countArray,"ppois",19.9,alternative = c("two.sided", "less", "greater"), exact = NULL)

# 4.Geometric Distribution
fit4 <- fitdist(countArray, "geom",method = "mme")
# ks.test(countArray,"pgeom",0.04784689,alternative = c("two.sided", "less", "greater"), exact = NULL)

# 5. Normal Distribution
fit5 <- fitdist(countArray, "norm")
```

```r
ks.test(countArray,"pnorm",19.9,6.3,alternative = c("two.sided", "less", "greater"), exact = NULL)
```

```
## Warning in ks.test(countArray, "pnorm", 19.9, 6.3, alternative =
## c("two.sided", : ties should not be present for the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  countArray
## D = 0.29089, p-value = 0.3659
## alternative hypothesis: two-sided
```

```r
plot(fit5)
```



4.3.2 Seller

```r
# Find most active seller in our token
frequencyTableSeller <- table(WithoutOutlierdata[1])
freqseller<- as.data.frame(frequencyTableSeller)
colnames(freqseller) <- c("sllerID","frequency")
mostFreqSeller <-freqseller[which(freqseller$frequency == max(freqseller$frequency)),]
seller <- mostFreqSeller$sllerID
message("Id of the most active seller in our token: ", seller)
```

```
## Id of the most active seller in our token: 6253227
```

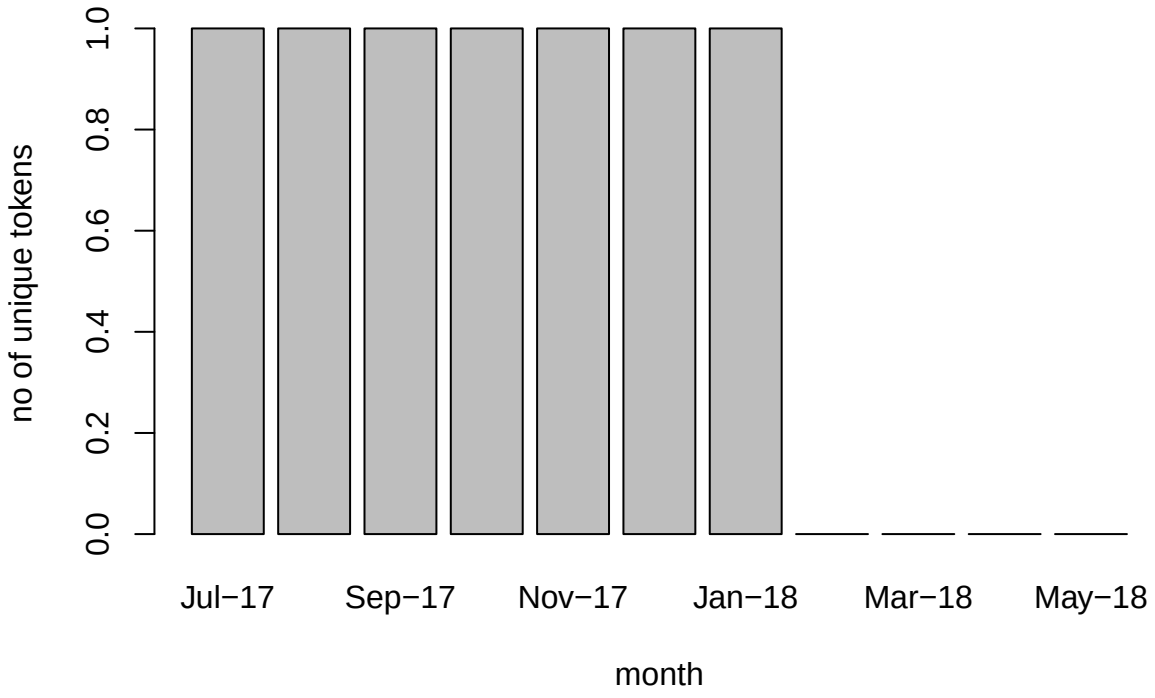
```r
# Plot the number of transactions of each month of seller6253227(we call it user2)
user2 <- WithoutOutlierdata[which(WithoutOutlierdata$fromNode == seller),]
user2$Date <- as.Date(as.POSIXct(user2$Date,origin="1970-01-01",tz="GMT"))
user2$Date <- cut(user2$Date, breaks = "month")
```

```r
frequencyTable2 <- table(user2[3])
freq2 <-  as.data.frame(frequencyTable2)
colnames(freq2) <- c("time","frequency")
# barplot(freq2$frequency,names.arg = freq2$time,ylab = "number of transactions", xlab = "month",ylim =
```

```r
# Read all token data
filenames <- list.files("tokens", pattern="*.txt", full.names=TRUE)

# find the minimum and max date of all token data

# initial minDate and maxDate
newfreq2 <- freq2[order(freq2$time),]
minDate <- as.Date(newfreq2$time[1])
maxDate <- as.Date(newfreq2$time[nrow(newfreq2)])

for (file in filenames){
  token <- read.delim(file, header = FALSE, sep = " ")
  Frame<- as.data.frame(token)
  colnames(Frame)<-c('fromNode','toNode','unixTime','totalAmount')
  user2_other <- Frame[which(Frame$fromNode == seller),]
  if (nrow(user2_other)>0){
    user2_other$unixTime <- as.Date(as.POSIXct(user2_other$unixTime,origin="1970-01-01",tz="GMT"))
    user2_other$unixTime <- cut(user2_other$unixTime, breaks = "month")
    frequencyTable2_other <- table(user2_other[3])
    freq2_other <-  as.data.frame(frequencyTable2_other)
    colnames(freq2_other) <- c("time","frequency")
    newfreq2_other <- freq2_other[order(freq2_other$time),]
    tempminDate2 <- as.Date(newfreq2_other$time[1])
    tempmaxDate2 <- as.Date(newfreq2_other$time[nrow(newfreq2_other)])
    if (tempminDate<minDate){
      minDate <- tempminDate
    }
    if (tempmaxDate>maxDate){
      maxDate <- tempmaxDate
    }
  }
}
```

```r
# Base on the minmum and maxmum date, create an array of counter and date intervals
countArraySeller <- c(0,0,0,0,0,0,0,0,0,0,0,0)
dateinterval <- c("2017-07-01","2017-08-01","2017-09-01","2017-10-01","2017-11-01","2017-12-01","2018-0
dateinterval <- as.Date(dateinterval)

# Count the number of unique tokens that the most active seller bought each month
for (file in filenames){
  token2 <- read.delim(file, header = FALSE, sep = " ")
  Frame2<- as.data.frame(token2)
  colnames(Frame2)<-c('fromNode','toNode','unixTime','totalAmount')
  user2_other2 <- Frame2[which(Frame2$fromNode == seller),]

  if (nrow(user2_other2)>0){
    user2_other2$unixTime <- as.Date(as.POSIXct(user2_other2$unixTime,origin="1970-01-01",tz="GMT"))
    user2_other2$unixTime <- cut(user2_other2$unixTime, breaks = "month")
    frequencyTable2_other2 <- table(user2_other2[3])
```

```
    freq2_other2 <-  as.data.frame(frequencyTable2_other2)
    colnames(freq2_other2) <- c("time","frequency")
    freq2_other2$time <- as.Date(freq2_other2$time)
    for (arow in 1:nrow(freq2_other2))
    {
      if (freq2_other2$time[arow]==dateinterval[1]){
        countArraySeller[1] = countArraySeller[1]+1

      }
      else if (freq2_other2$time[arow]==dateinterval[2]){
        countArraySeller[2] = countArraySeller[2]+1
      }
      else if (freq2_other2$time[arow]==dateinterval[3]){
        countArraySeller[3] = countArraySeller[3]+1
      }
      else if (freq2_other2$time[arow]==dateinterval[4]){
        countArraySeller[4] = countArraySeller[4]+1
      }
      else if (freq2_other2$time[arow]==dateinterval[5]){
        countArraySeller[5] = countArraySeller[5]+1
      }
      else if (freq2_other2$time[arow]==dateinterval[6]){
        countArraySeller[6] = countArraySeller[6]+1
      }
      else if (freq2_other2$time[arow]==dateinterval[7]){
        countArraySeller[7] = countArraySeller[7]+1
      }
      else if (freq2_other2$time[arow]==dateinterval[8]){
        countArraySeller[8] = countArraySeller[8]+1
      }
      else if (freq2_other2$time[arow]==dateinterval[9]){
        countArraySeller[9] = countArraySeller[9]+1
      }
      else if (freq2_other2$time[arow]==dateinterval[10]){
        countArraySeller[10] = countArraySeller[10]+1
      }
       else if (freq2_other2$time[arow]==dateinterval[11]){
        countArraySeller[11] = countArraySeller[11]+1
      }
      else {
        print("error")
      }
    }
  }
}

# plot
dateintervalSeller2 <- c("Jul-17","Aug-17","Sep-17","Oct-17","Nov-17","Dec-17","Jan-18","Feb-18","Mar-18
barplot(countArraySeller,names.arg = dateintervalSeller2,ylab = "no of unique tokens", xlab = "month")
```



4.3.3 Conclusion

For most active buyers, we tried 5 distributions: Poisson Distribution, Weibull Distribution, Exponential Distribution, Geometric Distribution, Normal Distribution.We also use ks-test to check which distribution fit our data best. After compare with parameters, graphs and p-value of different distributions, we can find: we get a bigest p-value = 0.3659 while fiting normal distribution, **normal distribution** fit our data best. From above results and plot, we can find that the most active seller in our token (seller 6253227) only bought our token, cannot find purchase record in other tokens, so the distribution is just a line between Jul 2017 and Jan 2018, unable to fit any distribution.