Report

pushpita panigrahi(pxp171530), akash chand(axc173730), siddharth swarup panda(ssp171730)

Read the token data

We chose networkbnbTX token as our dataset.

```
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
```

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 that attempts to log a value greater than the product of total supply and subunits is deemed as fraudulent.

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 outliers : ',outliers)
```

```
## Number of outliers : 0
```

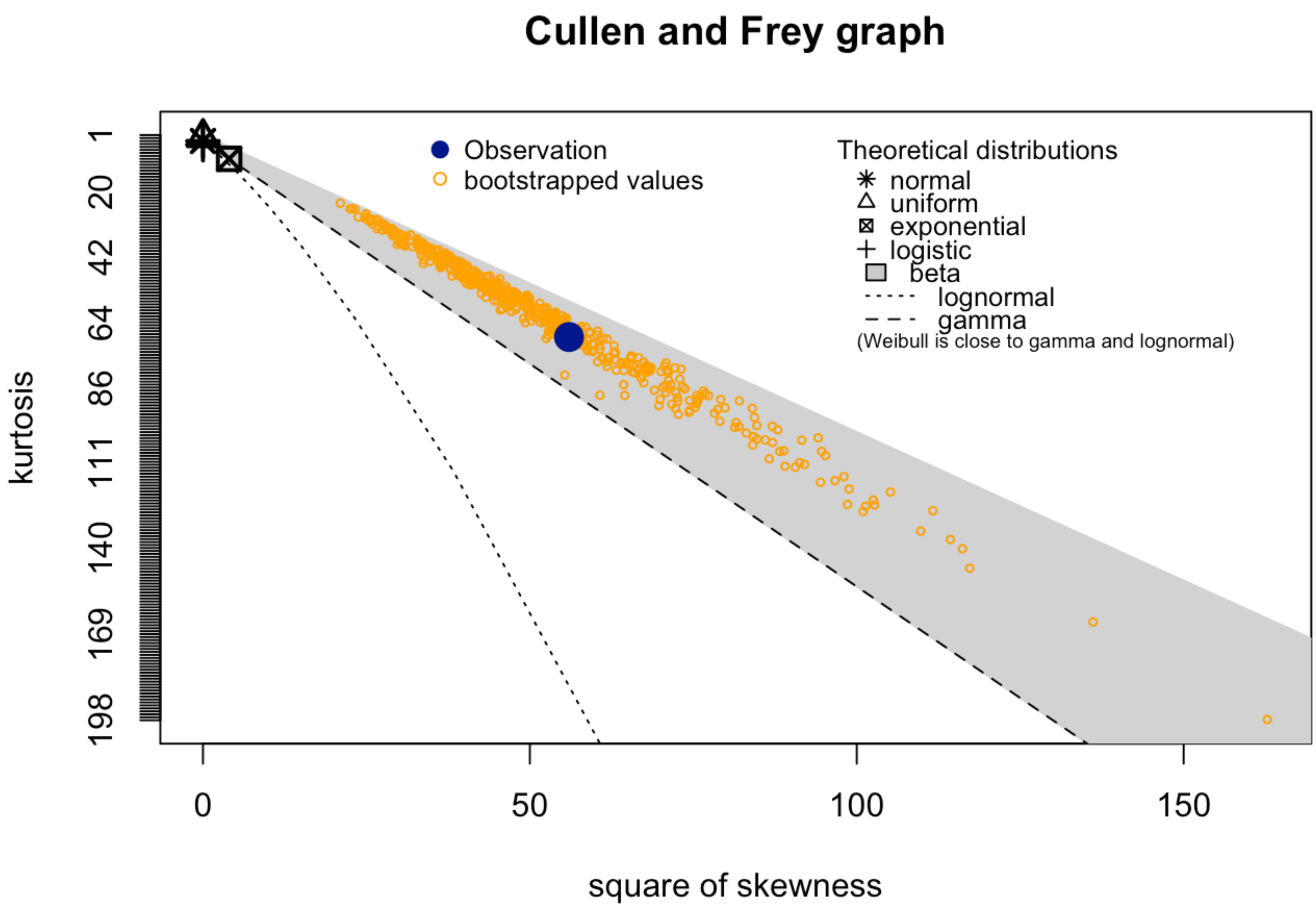
```
message('Number of valid amounts : ',count)
```

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

Calculating and plotting selling frequency

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

```
##   Users_Count Sell_Count
## 1           1      16575
## 2           2       3962
## 3           3       2115
## 4           4       1284
## 5           5        870
## 6           6        702
```




```
## summary statistics
## ------
## min:  65   max:  34809
## median:  399
## mean:  994.8245
## estimated sd:  2931.883
## estimated skewness:  7.48285
## estimated kurtosis:  68.98622
```

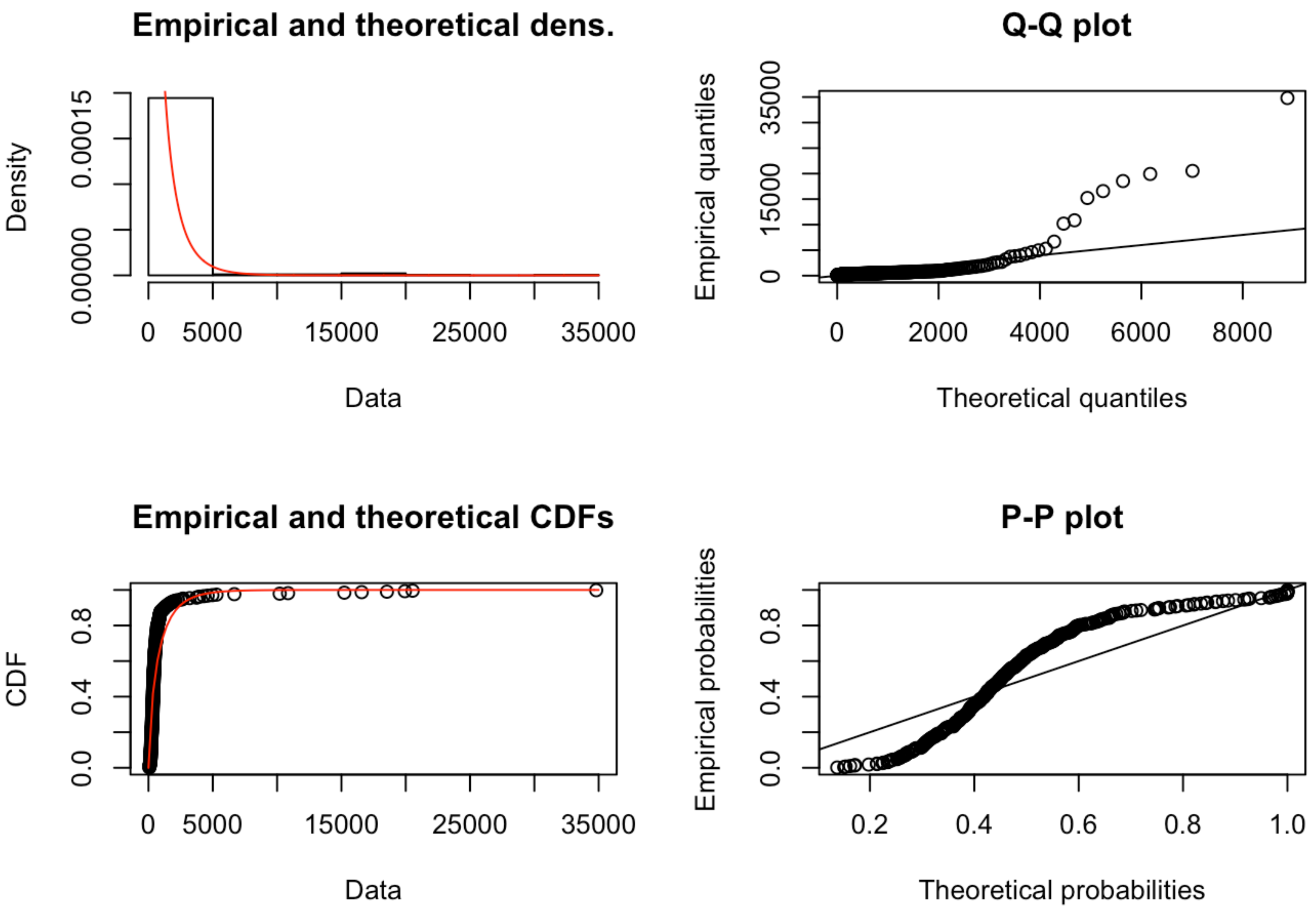
Approximating the selling distributions

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
```

```
## Fitting of the distribution ' weibull ' by maximum likelihood
## Parameters:
##          estimate  Std. Error
## shape   0.7719378  0.02515483
## scale 774.1209761 56.42896598
```

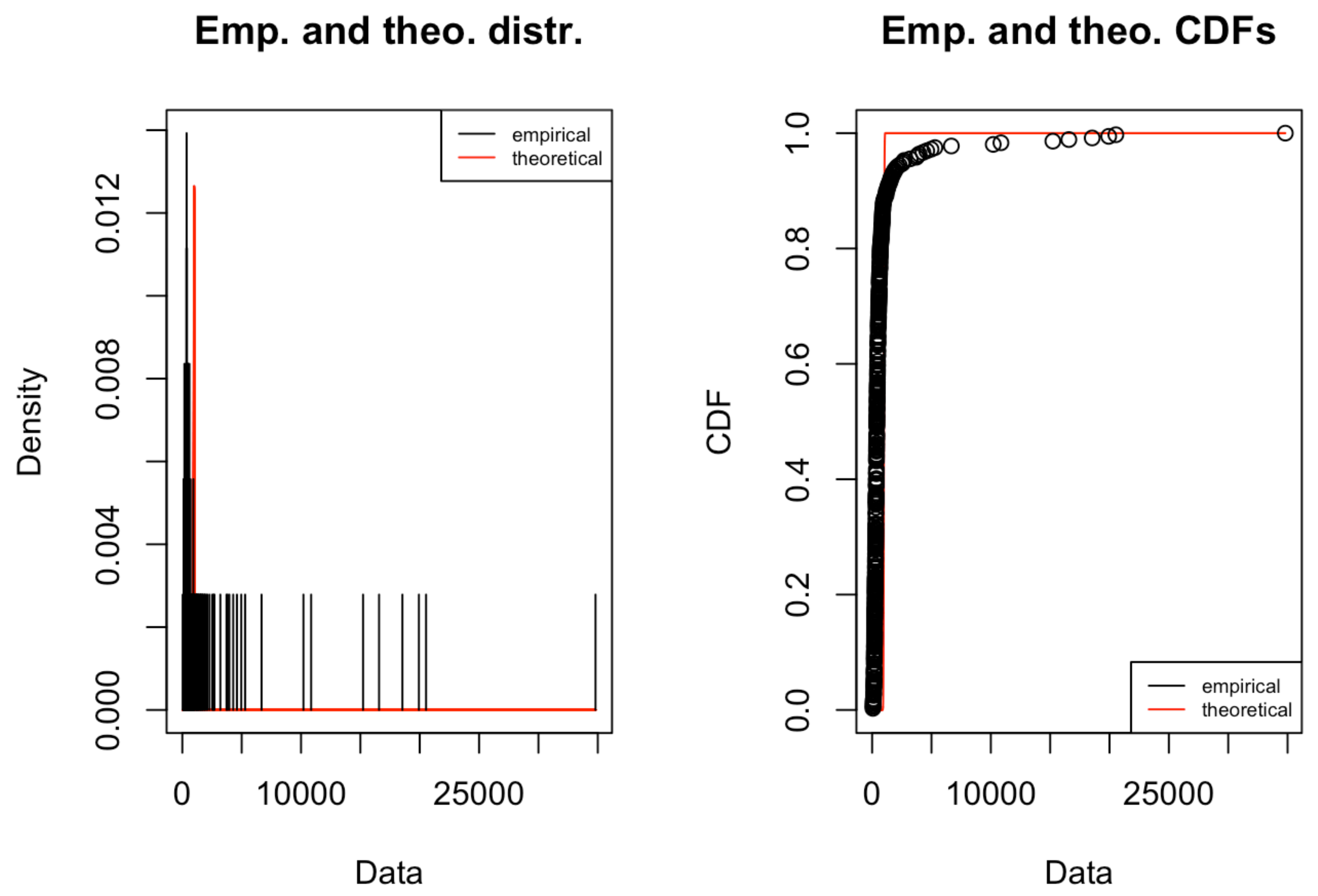
```
plot(distributionFit_Seller_wb)
```




```
distributionFit_Seller_pois
```

```
## Fitting of the distribution ' pois ' by maximum likelihood
## Parameters:
##        estimate Std. Error
## lambda 994.8245   1.664616
```

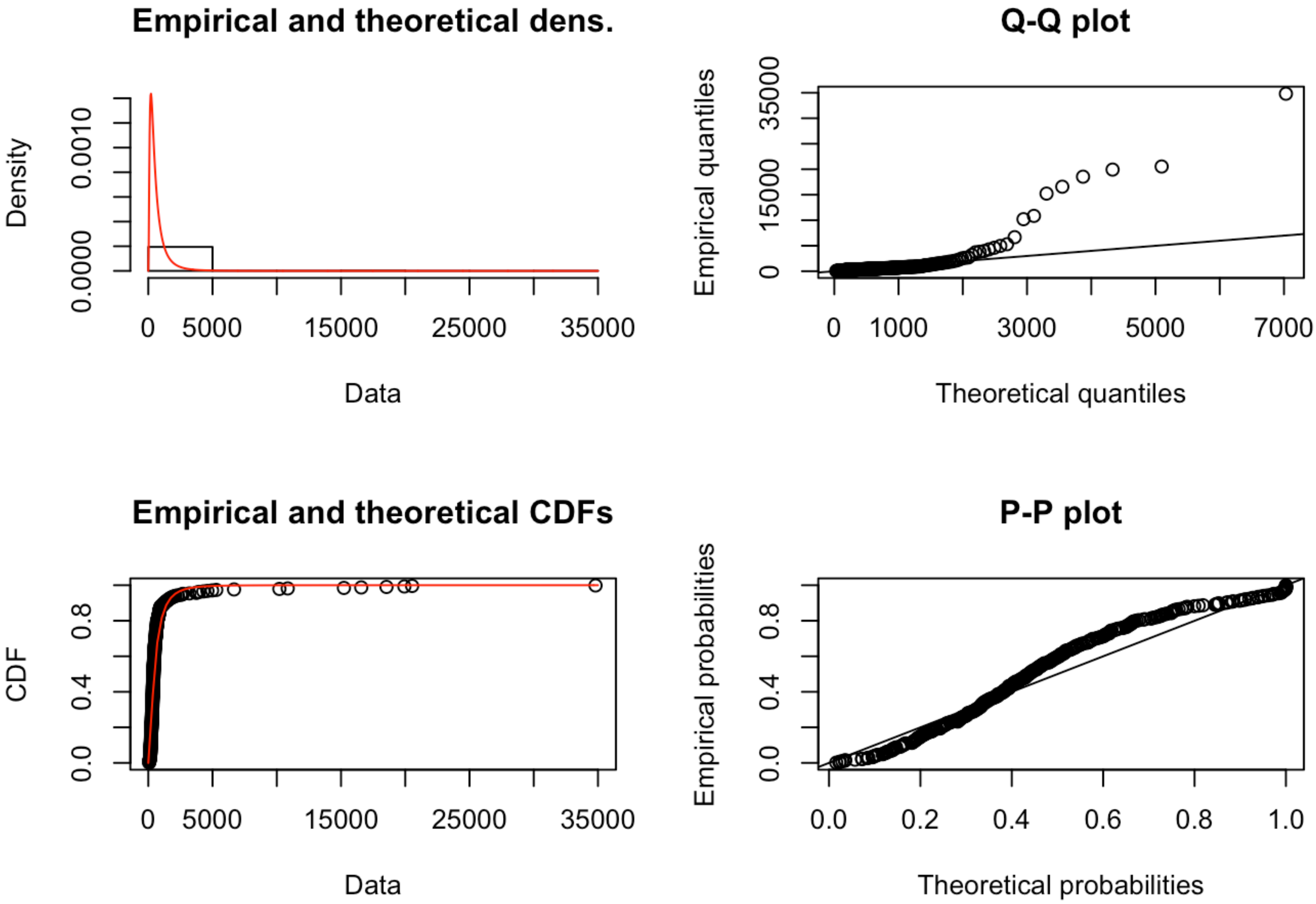
```
plot(distributionFit_Seller_pois)
```




```
distributionFit_Seller_ln
```

```
## Fitting of the distribution ' lnorm ' by maximum likelihood
## Parameters:
##          estimate Std. Error
## meanlog 6.1329835 0.04809244
## sdlog   0.9112216 0.03400630
```

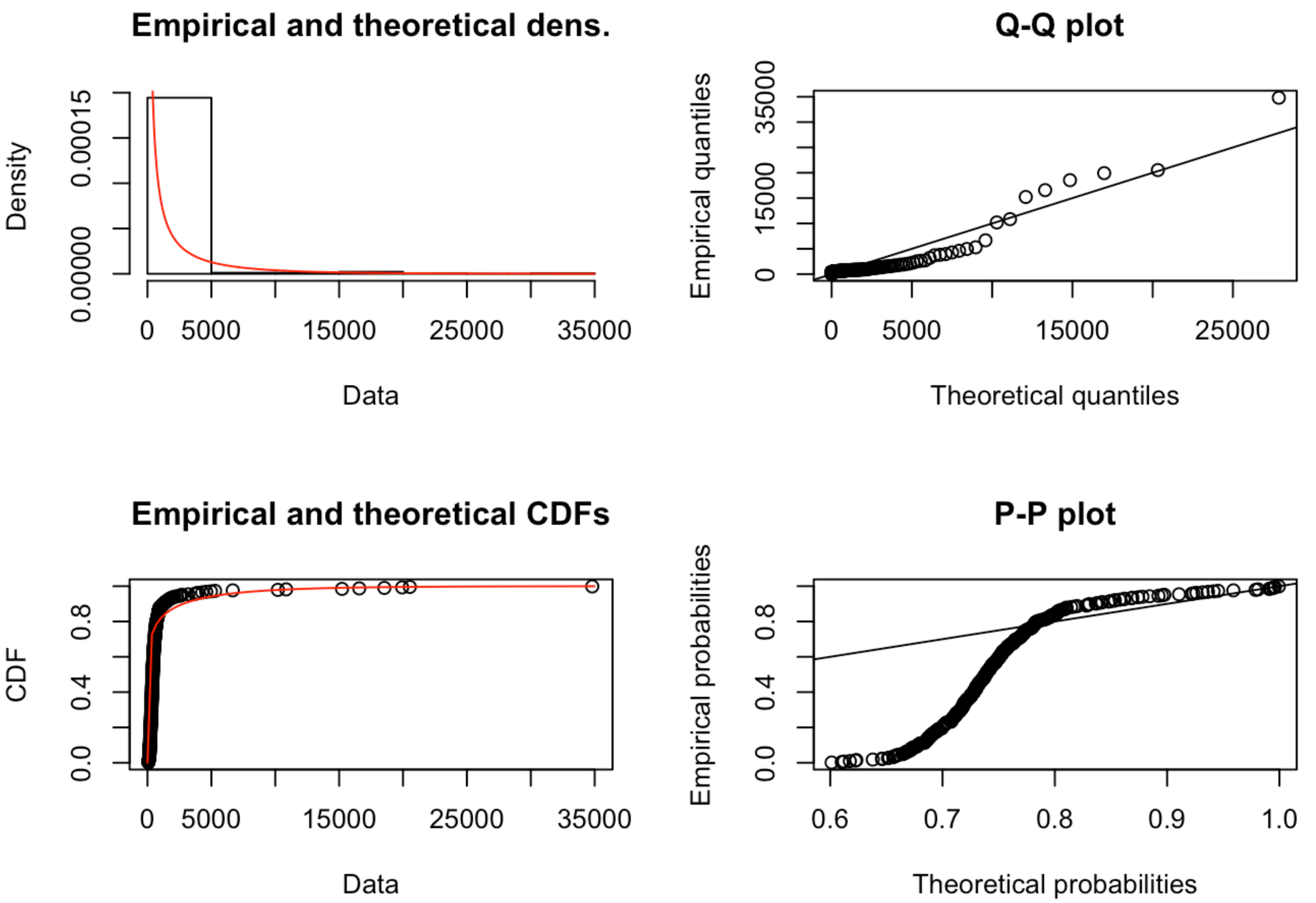
```
plot(distributionFit_Seller_ln)
```




```
distributionFit_Seller_gm
```

```
## Fitting of the distribution ' gamma ' by matching moments
## Parameters:
##           estimate
## shape 0.1154545321
## rate  0.0001160552
```

```
plot(distributionFit_Seller_gm)
```



Calculating the buying frequency

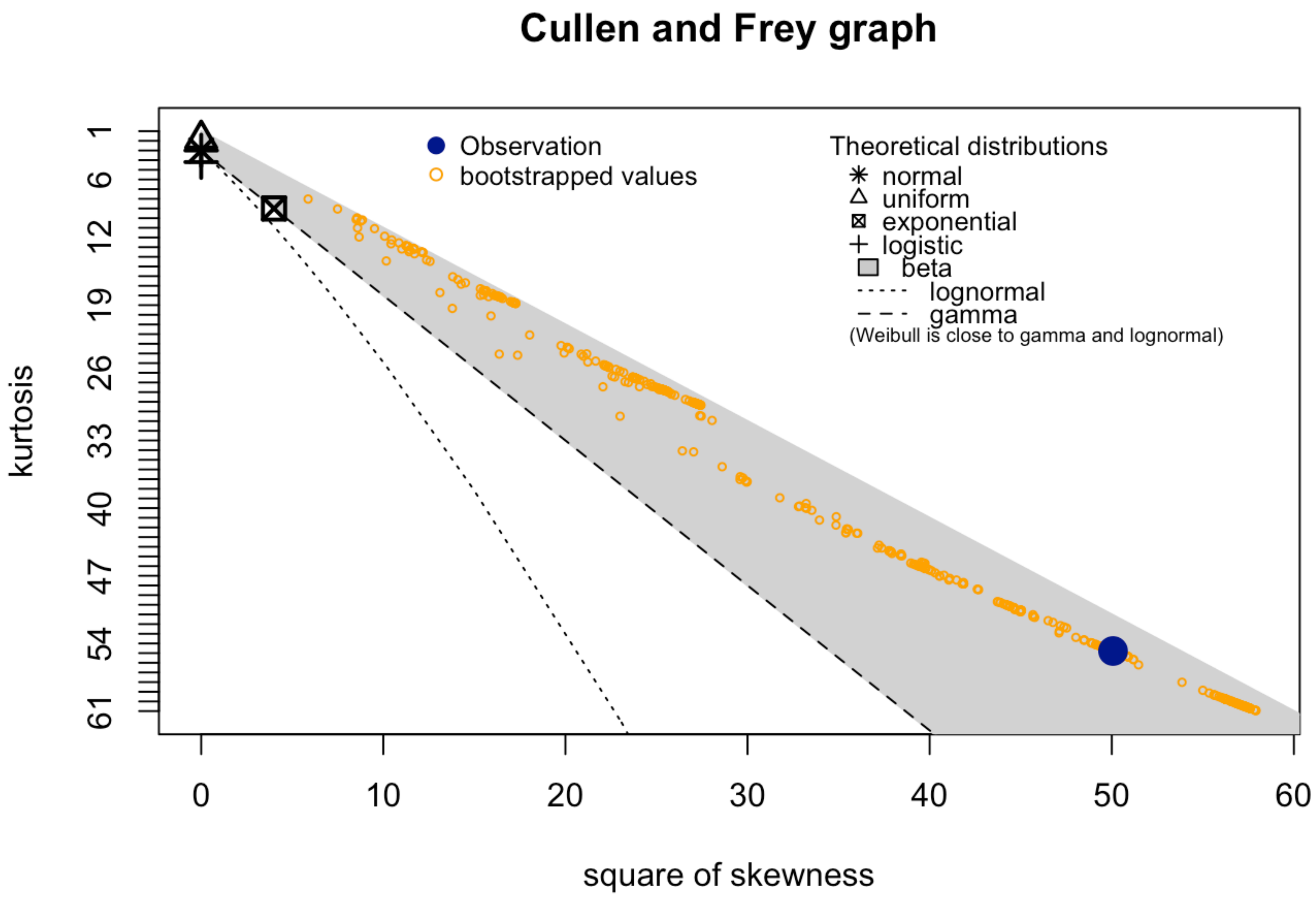
```
countToDf <- count(mydata, "TONODE")
countToFf <- count(countToDf, "freq")
```

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

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

```
##   Users_Count Buy_Count
## 1           1    252994
## 2           2     56706
## 3           3     16029
## 4           4      5184
## 5           5      2240
## 6           6      1452
```

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




```
## summary statistics
## ------
## min:  24   max:  252994
## median:  117.5
## mean:  6157.621
## estimated sd:  33890.53
## estimated skewness:  7.076031
## estimated kurtosis:  54.78311
```

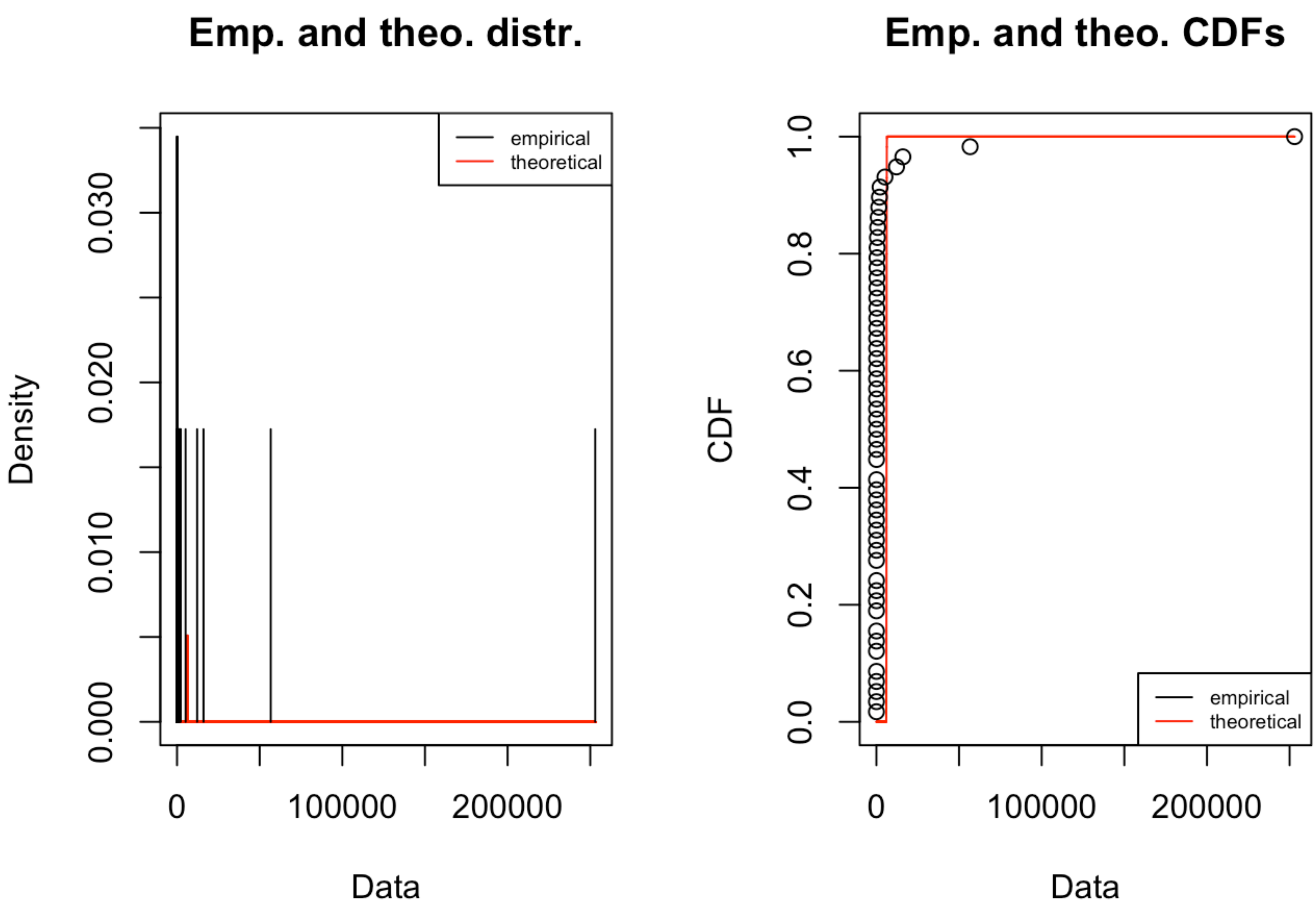
Approximating the buying distributions

```
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
```

```
## Fitting of the distribution ' pois ' by maximum likelihood
## Parameters:
##        estimate Std. Error
## lambda 6157.621   10.26635
```

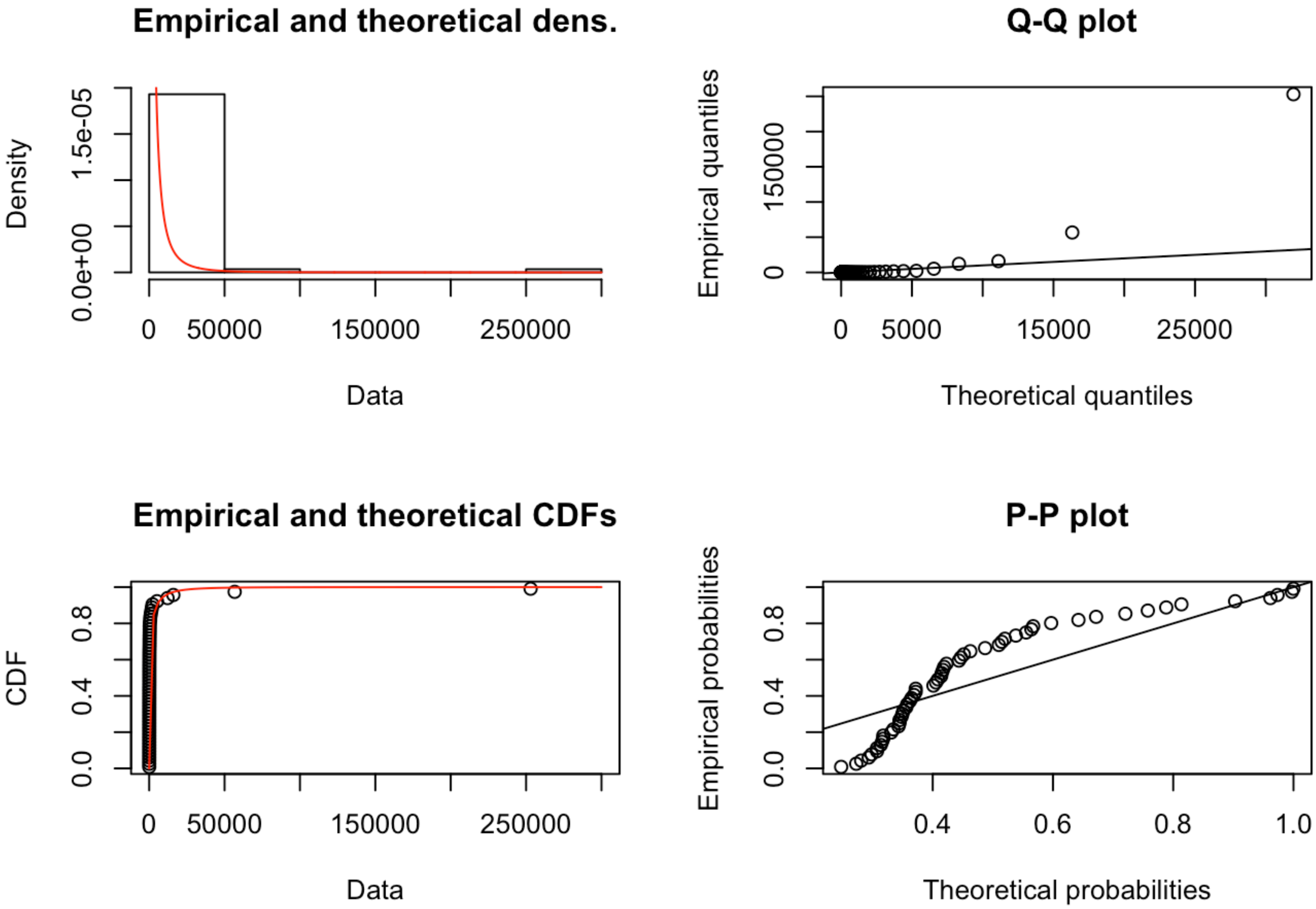
```
plot(distributionFit_Buyer_pois)
```




```
distributionFit_Buyer_wb
```

```
## Fitting of the distribution ' weibull ' by maximum likelihood
## Parameters:
##          estimate   Std. Error
## shape   0.3913615   0.03285488
## scale 595.1275712 213.11455385
```

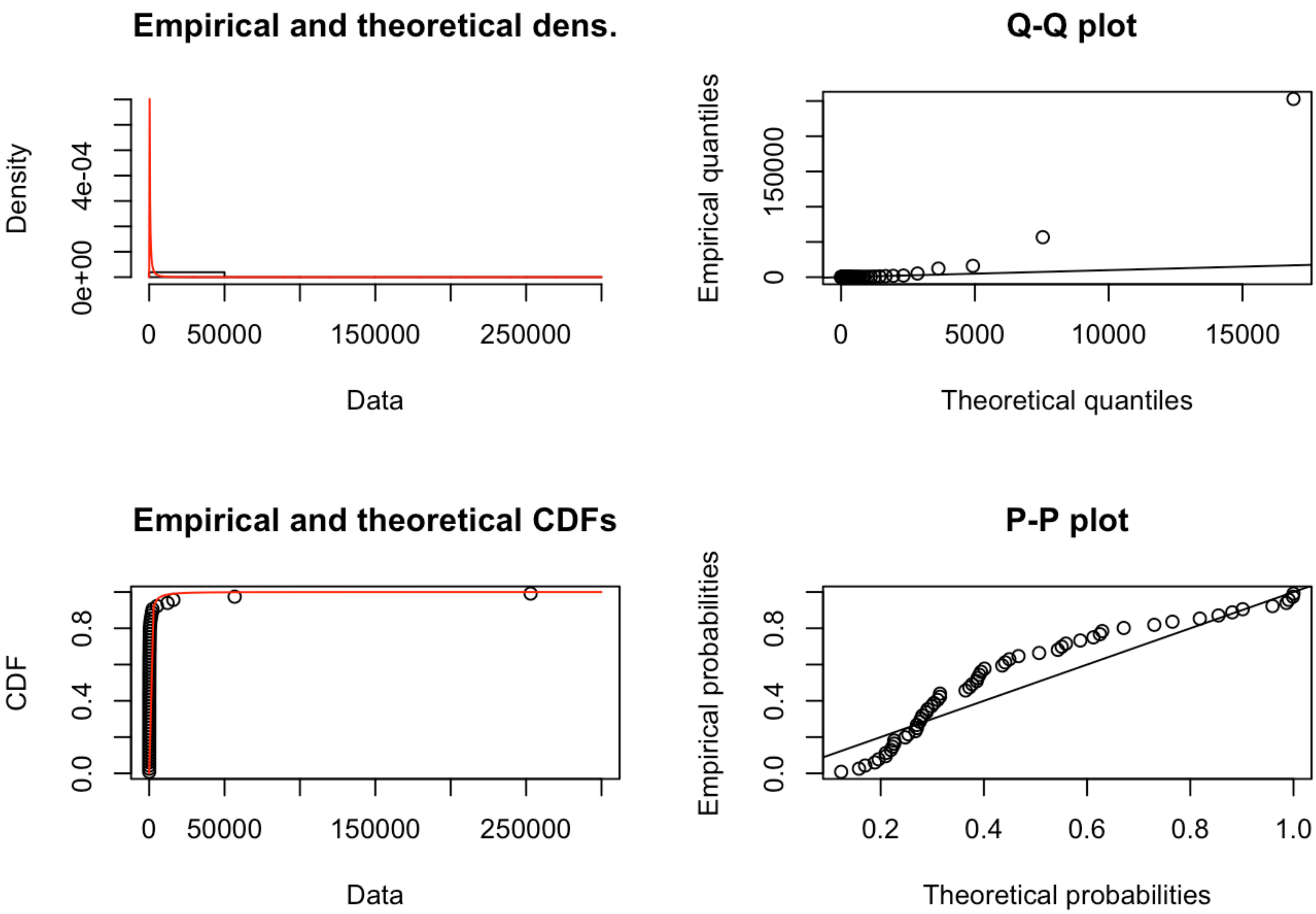
```
plot(distributionFit_Buyer_wb)
```




```
distributionFit_Buyer_ln
```

```
## Fitting of the distribution ' lnorm ' by maximum likelihood
## Parameters:
##          estimate Std. Error
## meanlog 5.323868  0.2432331
## sdlog   1.852408  0.1719915
```

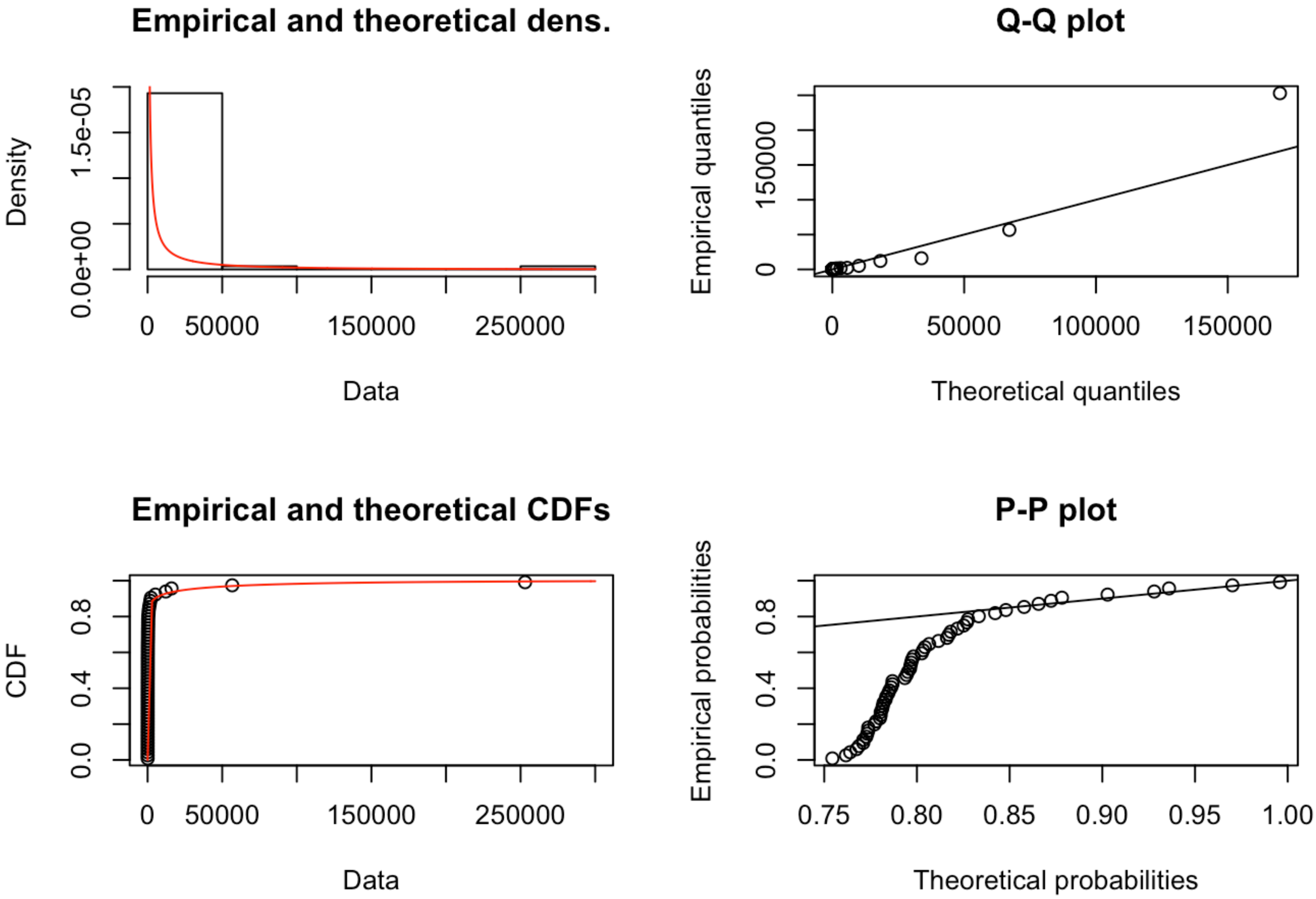
```
plot(distributionFit_Buyer_ln)
```




```
distributionFit_Buyer_gm
```

```
## Fitting of the distribution ' gamma ' by matching moments
## Parameters:
##           estimate
## shape 3.359095e-02
## rate  5.455183e-06
```

```
plot(distributionFit_Buyer_gm)
```



Conclusion

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

Study 2 :

We are trying to find the correlation between the unique number if buyers each day to the token opening price for the day.

Read the price file

Price file contains details of the open, clase, max and min price for the token foe each day

```
pricefile <-'bnb.txt'
col_names <- c("Date","Open","High","Low","Close","Volume","MarketCap")
myPrices <- read.csv( pricefile , header = TRUE, sep = "\t", dec = ".", col.names = c
ol_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
```

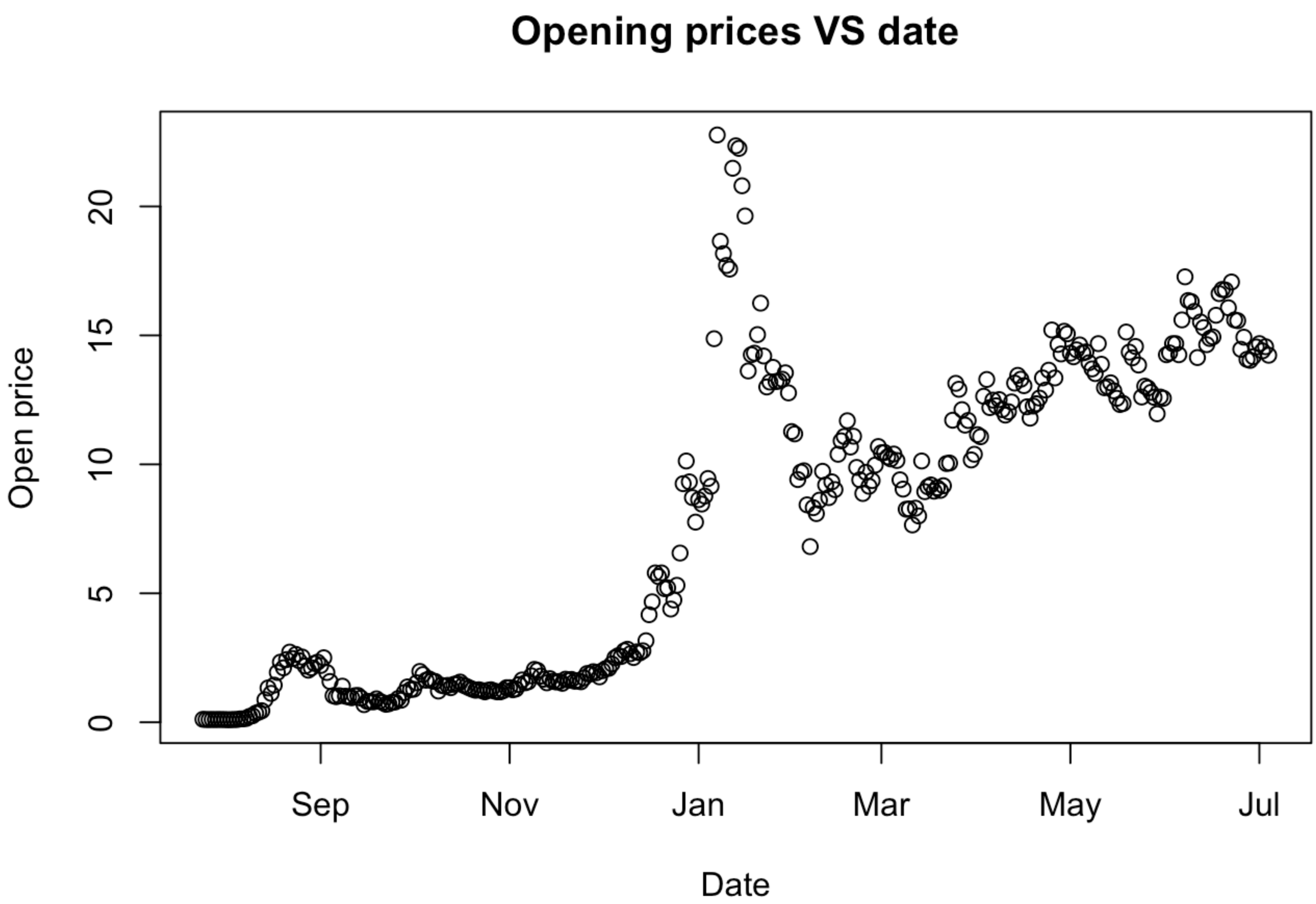
Studying distribution of the opening price .

The see the pattern for opening price values 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")
```



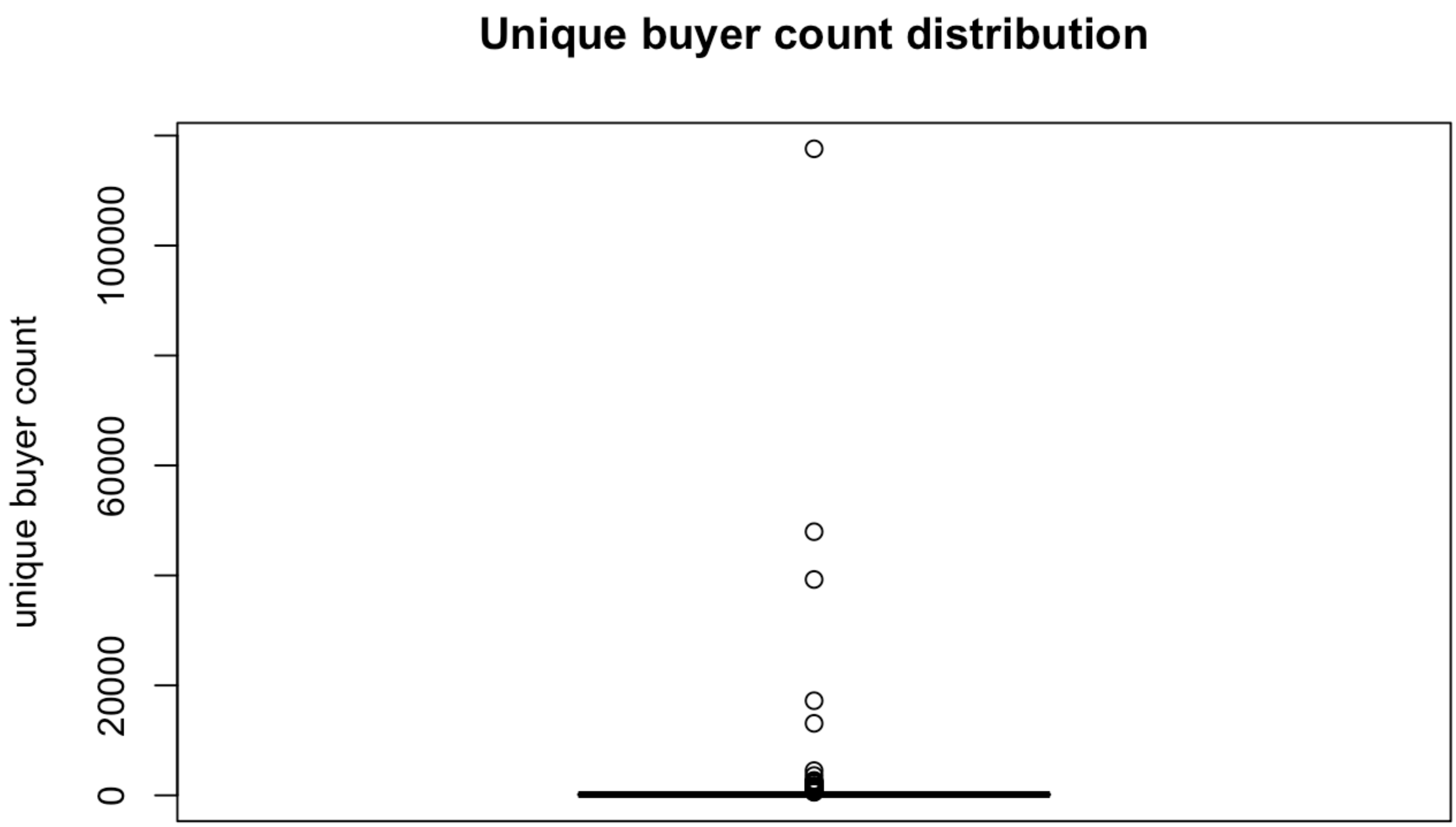
Studying the distribution of number of unique buyers each day.

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$count, main="Unique buyer count distribution", ylab="
unique buyer count")$out
```

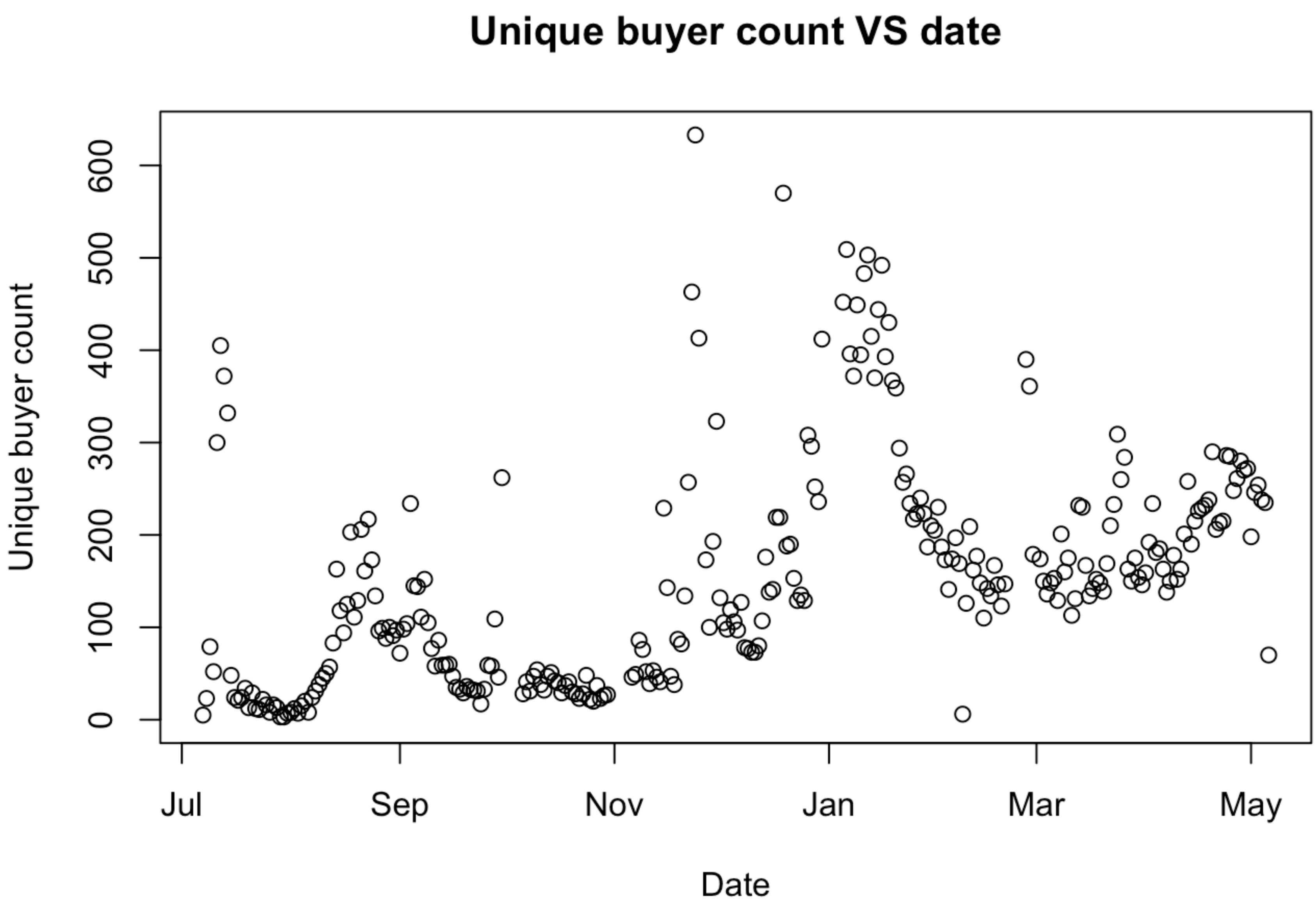


We see the summary of the outliers and plot the data with and without the outliers.

```
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 = "Date", ylab="Unique buyer count")
```



Combine opening price and unique buyer count for each day

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

```
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
}

priceSellForEachDay <- merge(x=timePrices, y=timeBuyFreq, by.x=c("Date"), by.y = c("D
ATE"))
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
## 5 2017-07-29 0.104782     3
## 6 2017-07-30 0.107935     3
```

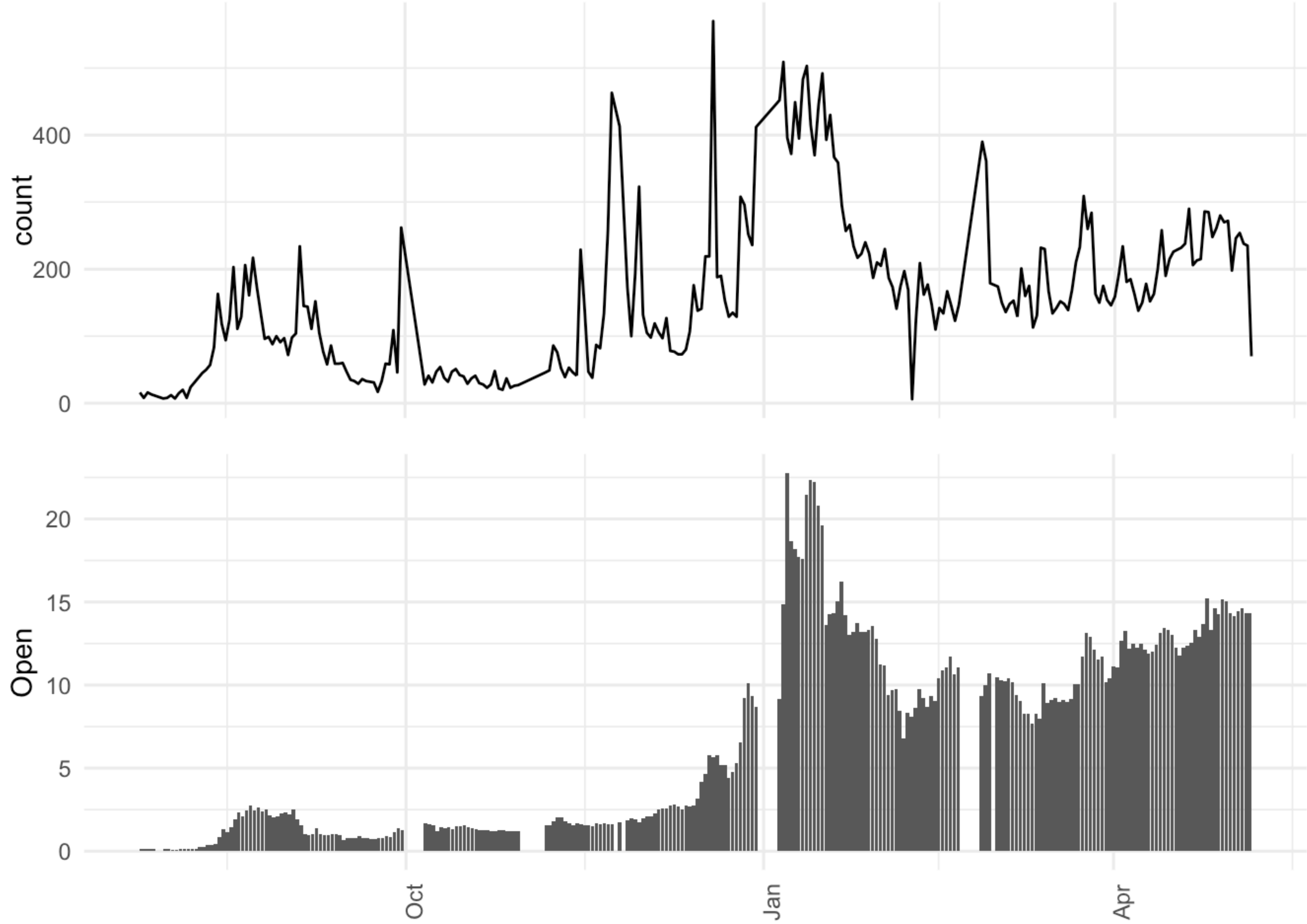
```
newSet <- remove_outliers(priceSellForEachDay$count)
maxCount = max(newSet[complete.cases(newSet)])
minCount = min(newSet[complete.cases(newSet)])
priceSellForEachDay <- subset(priceSellForEachDay, count<maxCount & count>minCount)
cor(priceSellForEachDay$Open, priceSellForEachDay$count, method=c("pearson"))
```

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

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. So we combine both plots to visualize the correlation.

```
#' 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 all the tokens all throughout the dataset

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 teh distrbution later on

```
#getting the active users in the current token
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 the token data

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

```
col_names <- c("FROMNODE","TONODE","DATE","TOKENAMOUNT")
fpath<-"/Users/pushpitapanigrahi/Desktop/PushpitaFiles/Study/4.StatsForDS/Proj1/Ether
eum 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 userId in this list
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")
head(usersFromAllTokens)
```

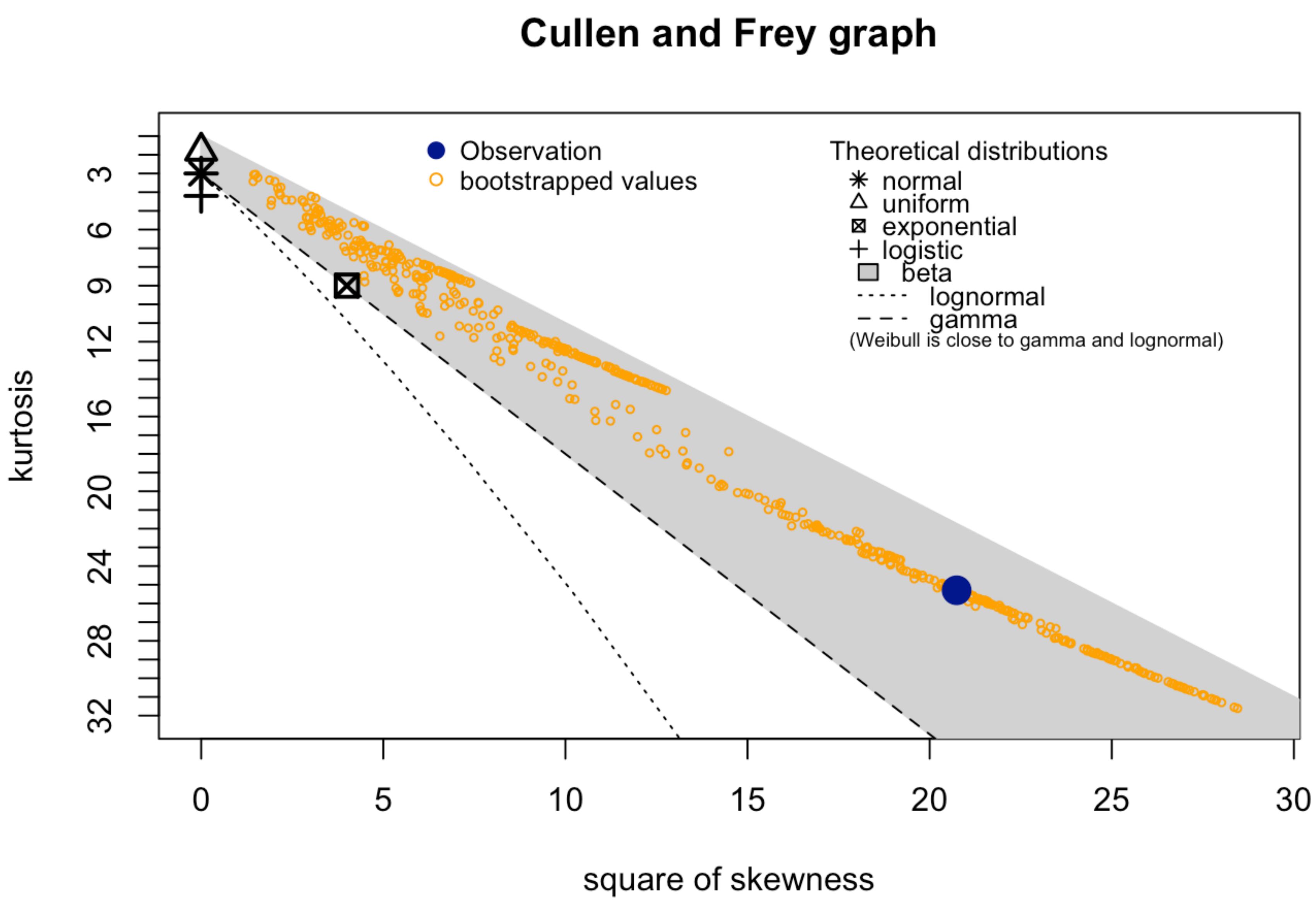
```
##   USERID
## 1 194317
## 2 194318
## 3 194320
## 4     17
## 5 194322
## 6 194324
```

```
# counting the number of tokens per userid
userTokenCount <- data.frame(table(usersFromAllTokens$USERID[usersFromAllTokens$USERI
D %in% activeUsers$USERS]))
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
```

Getting the distribution of unique token counts for the active users

```
freqOfTokenCount <- count(userTokenCount, "COUNT")
colnames(freqOfTokenCount) <- c("Users_Count", "Freq_Count")
descdist(freqOfTokenCount$Freq_Count, boot= 500)
```




```
## summary statistics
## ------
## min:  1   max:  6820
## median:  31
## mean:  461.0345
## estimated sd:  1297.268
## estimated skewness:  4.55422
## estimated kurtosis:  25.29544
```

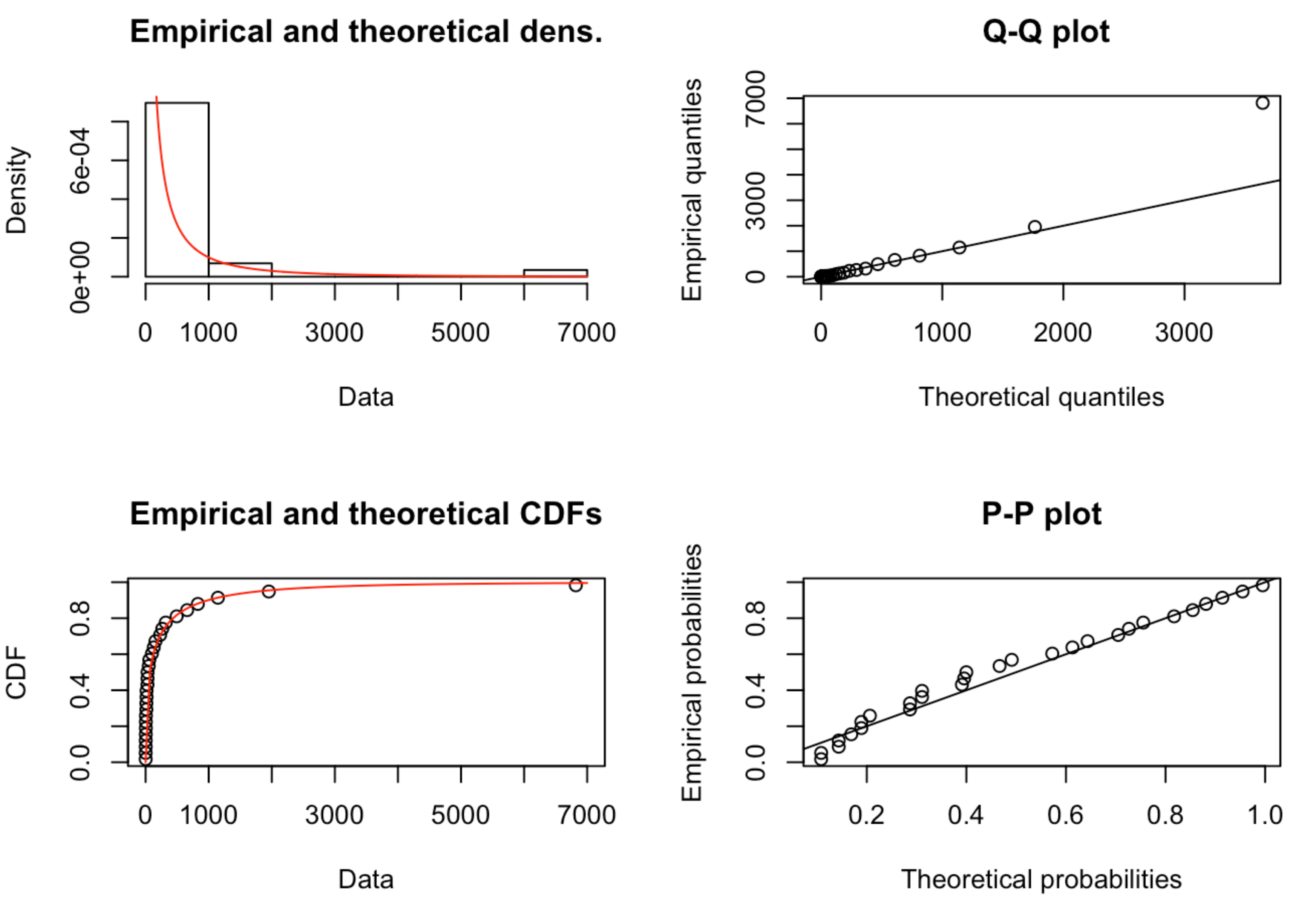
Fitting the distrubution to find the closest fit

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

distributionFit_Count_wb
```

```
## Fitting of the distribution ' weibull ' by maximum likelihood
## Parameters:
##          estimate  Std. Error
## shape   0.4348907  0.05987174
## scale 145.3452110 65.82282776
```

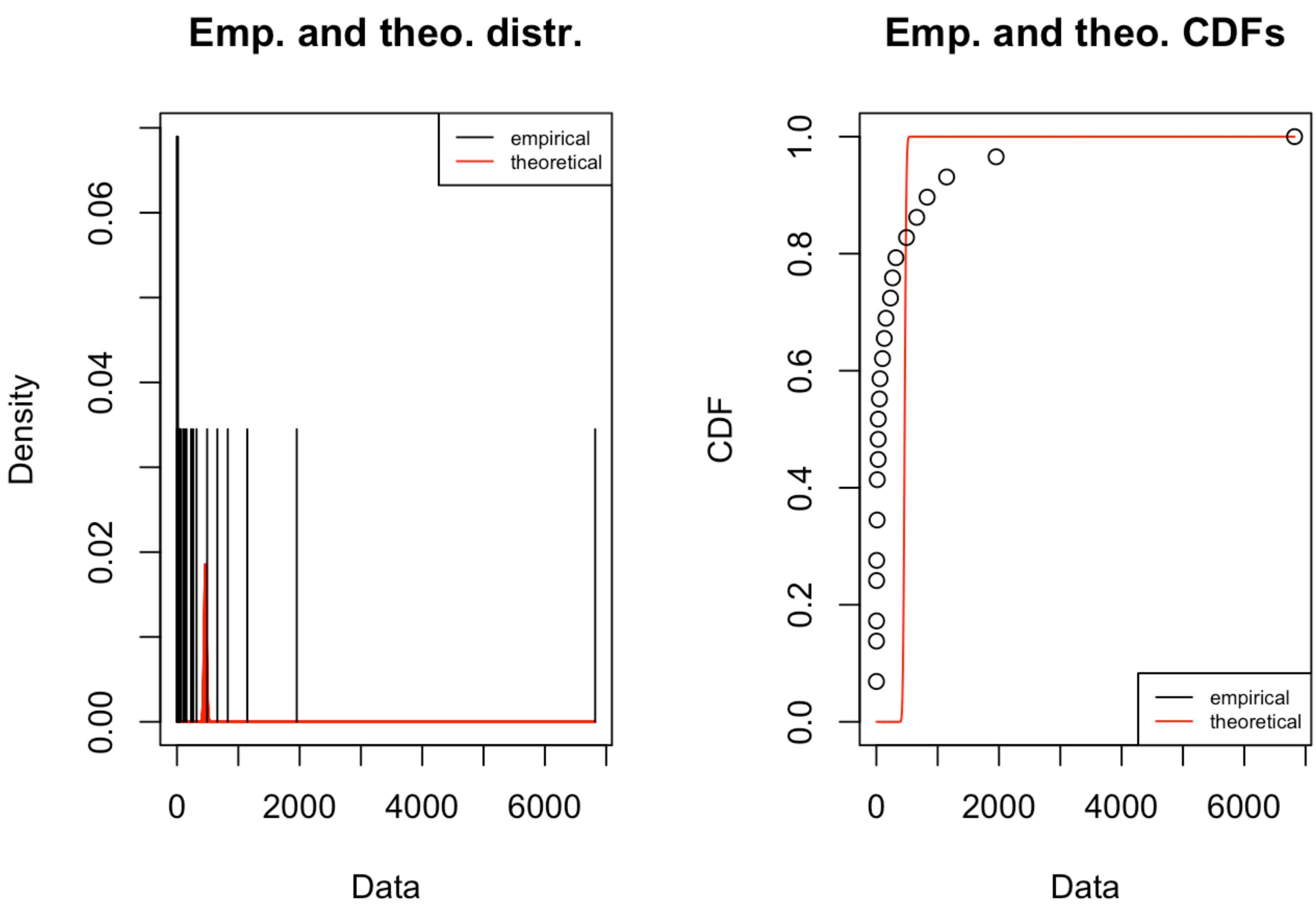
```
plot(distributionFit_Count_wb)
```




```
distributionFit_Seller_pois
```

```
## Fitting of the distribution ' pois ' by maximum likelihood
## Parameters:
##        estimate Std. Error
## lambda 994.8245   1.664616
```

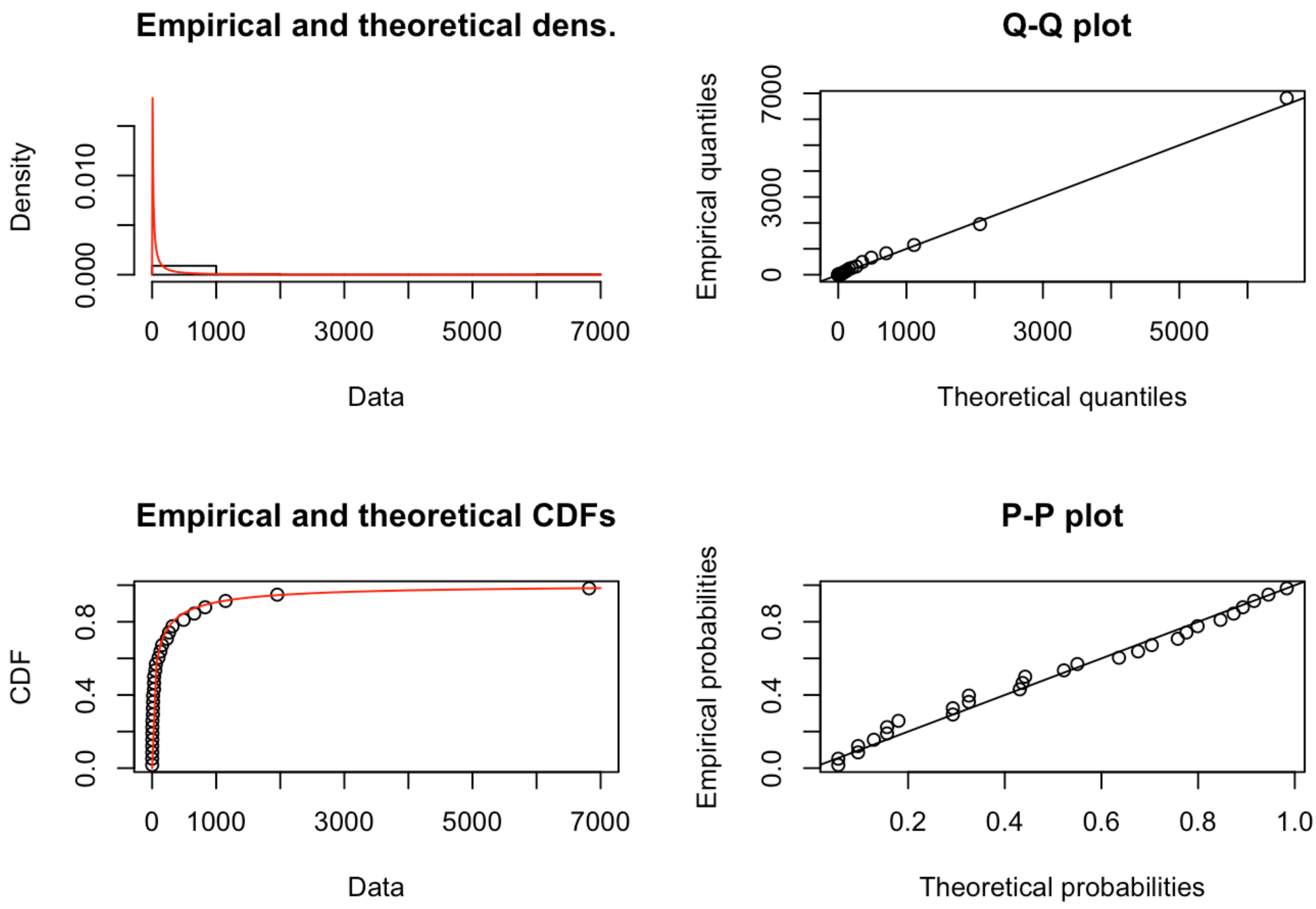
```
plot(distributionFit_Count_pois)
```




```
distributionFit_Seller_ln
```

```
## Fitting of the distribution ' lnorm ' by maximum likelihood
## Parameters:
##          estimate Std. Error
## meanlog 6.1329835 0.04809244
## sdlog   0.9112216 0.03400630
```

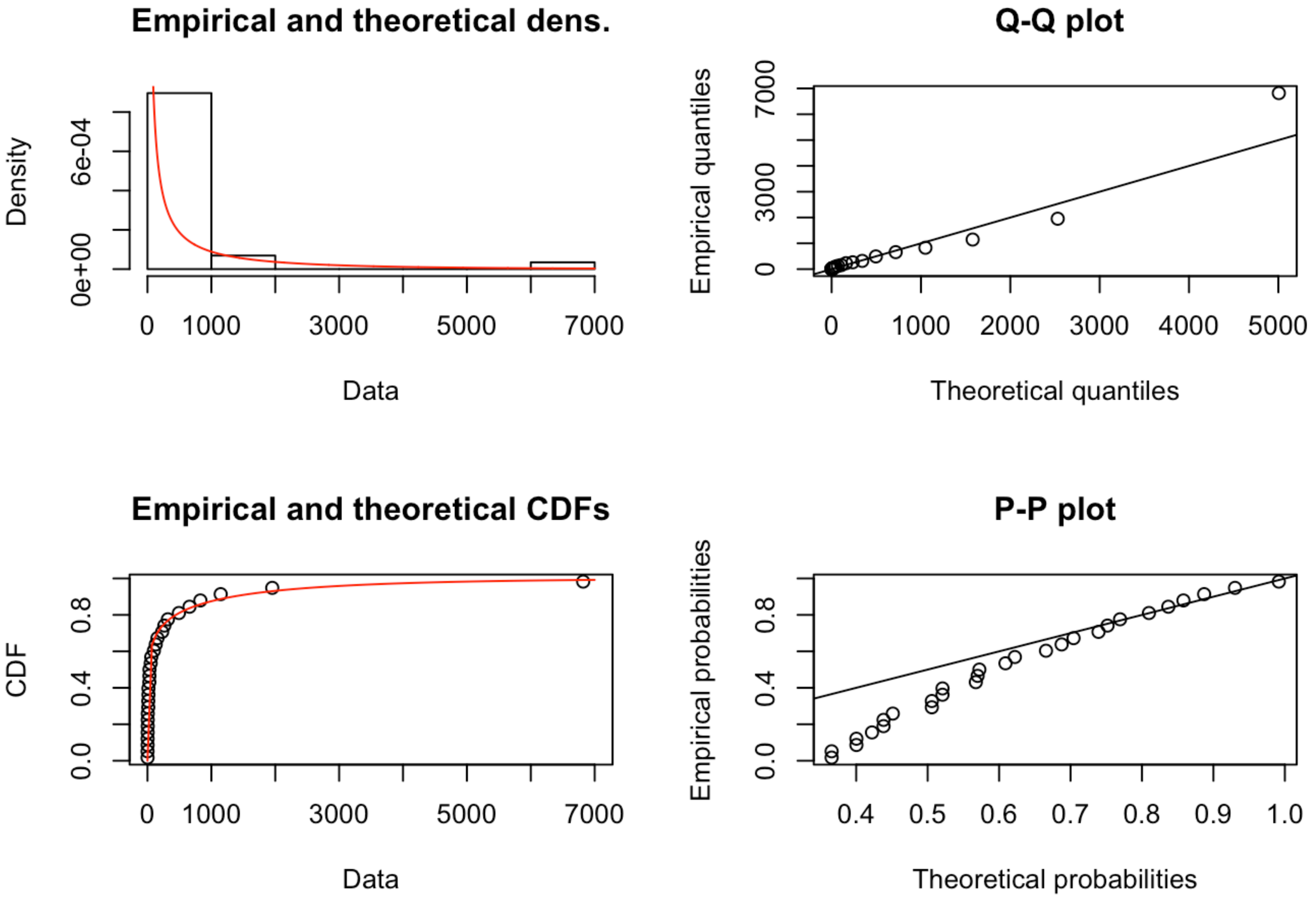
```
plot(distributionFit_Count_ln)
```




```
distributionFit_Seller_gm
```

```
## Fitting of the distribution ' gamma ' by matching moments
## Parameters:
##           estimate
## shape 0.1154545321
## rate  0.0001160552
```

```
plot(distributionFit_Count_gm)
```



Conclusion

The number of token is which the most active users appear, follows a poisson distribution in case of BNB token.