Ethereum Token Analysis

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

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

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:

```
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)
```

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_names1 <- c("Date","Open","High","Low","Close","Volume","MarketCap")
myPrices <- read.csv( pricefile , header = TRUE, sep = "\t", dec = ".", col.names = c
ol_names1)
myPrices$Date <- format(as.Date(myPrices$Date, format = "%m/%d/%Y"), "%Y-%m-%d")
```

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.

```
message('Maximum allowed amount : ', totalAmount)
```

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

```
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

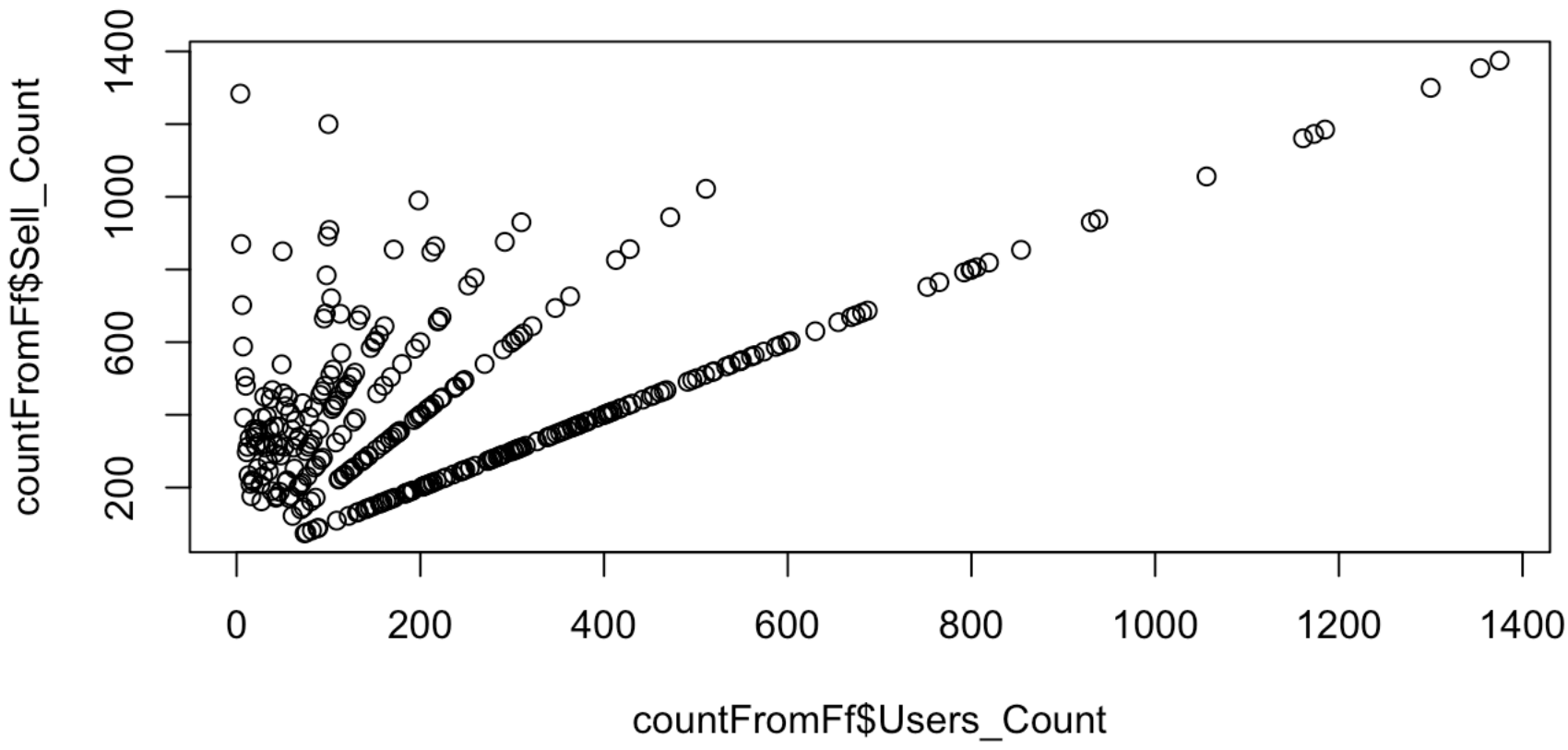
```
countFromDf <- count(mydata, "FROMNODE"); countFromFf <- count(countFromDf, "freq")
```

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

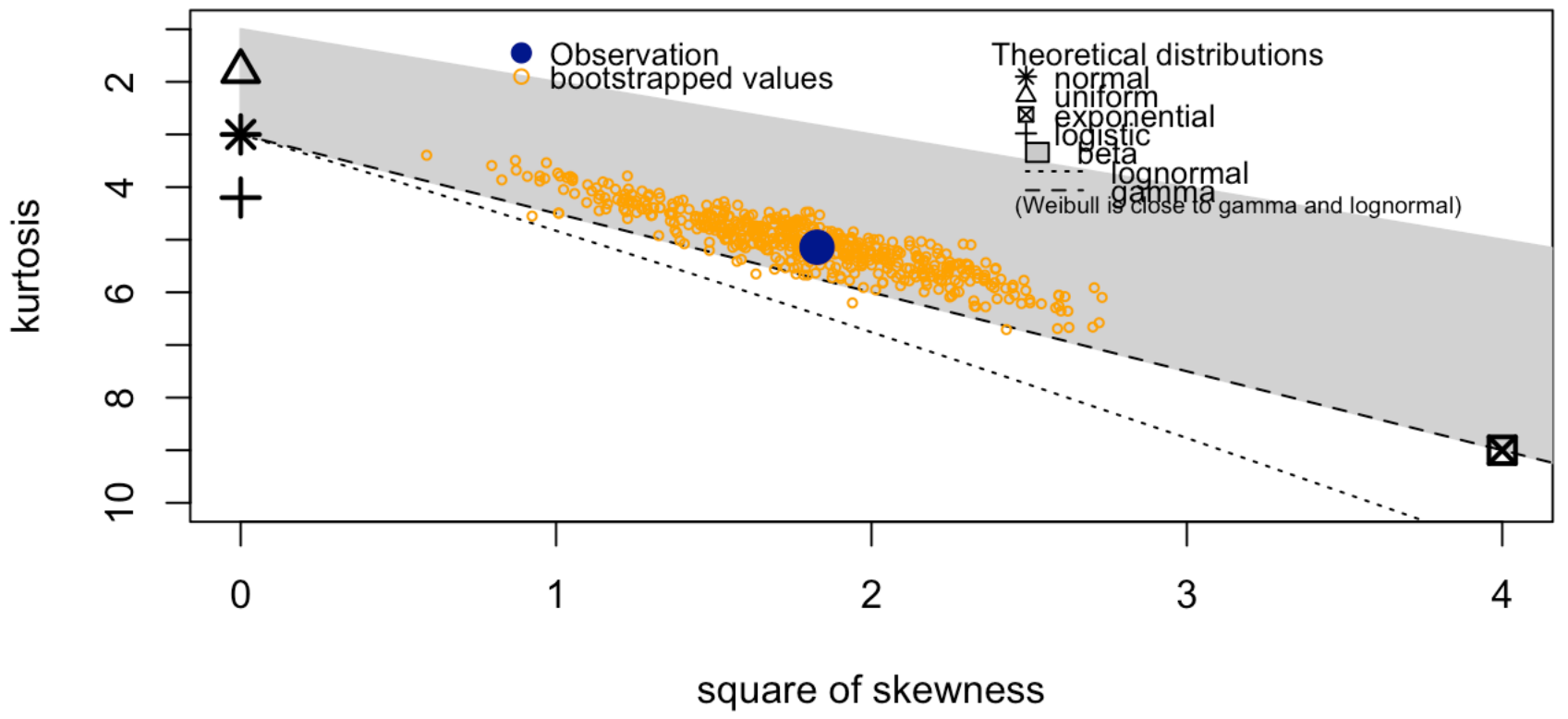
```
colnames(countFromFf) <- c("Users_Count", "Sell_Count")
```

Removing outliers and summarizing data

```
newSet1 <- remove_outliers(countFromFf$Sell_Count)
countFromFf <- subset(countFromFf, Sell_Count<max(newSet1[complete.cases(newSet1)]) &
Sell_Count>min(newSet1[complete.cases(newSet1)]))
plot(countFromFf$Users_Count, countFromFf$Sell_Count); descdist(countFromFf$Sell_Coun
t, boot= 500)
```



Cullen and Frey graph



```
## 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_pois$loglik; distributionFit_Seller_wb$loglik; distributionFit
_Seller_ln$loglik; distributionFit_Seller_gm$loglik
```

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

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

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

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

Calculating and plotting buying frequency

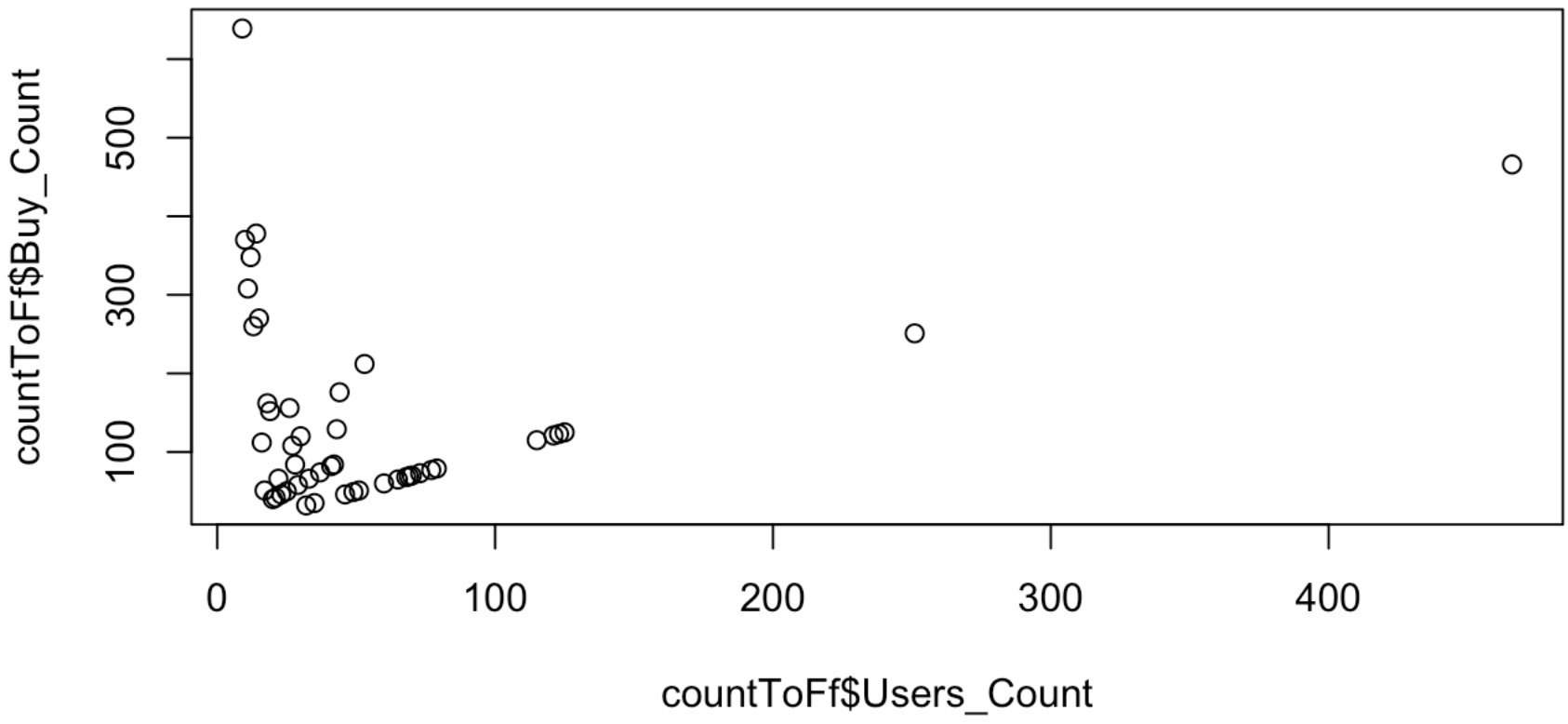
```
countToDf <- count(mydata, "TONODE"); countToFf <- count(countToDf, "freq")
```

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

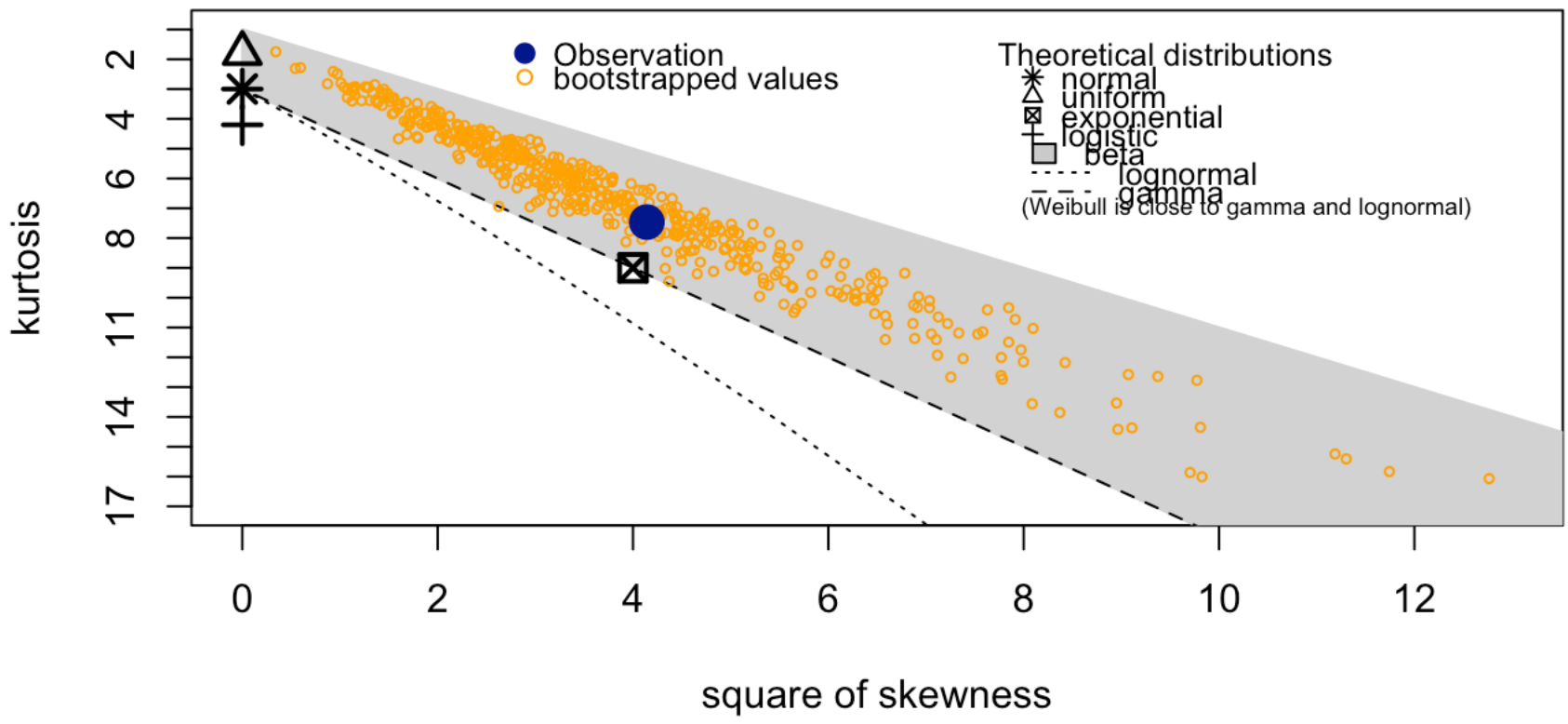
```
colnames(countToFf) <- c("Users_Count", "Buy_Count")
```

Removing outliers and summarizing data

```
newSet2 <- remove_outliers(countToFf$Buy_Count)
countToFf <- subset(countToFf, Buy_Count<max(newSet2[complete.cases(newSet2)]) & Buy_
Count>min(newSet2[complete.cases(newSet2)]))
plot(countToFf$Users_Count, countToFf$Buy_Count); descdist(countToFf$Buy_Count, boot=
500)
```



Cullen and Frey graph



```
## 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$loglik; distributionFit_Buyer_wb$loglik; distributionFit_B
uyer_ln$loglik; distributionFit_Buyer_gm$loglik
```

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

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

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

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

Observations

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:

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.

```
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

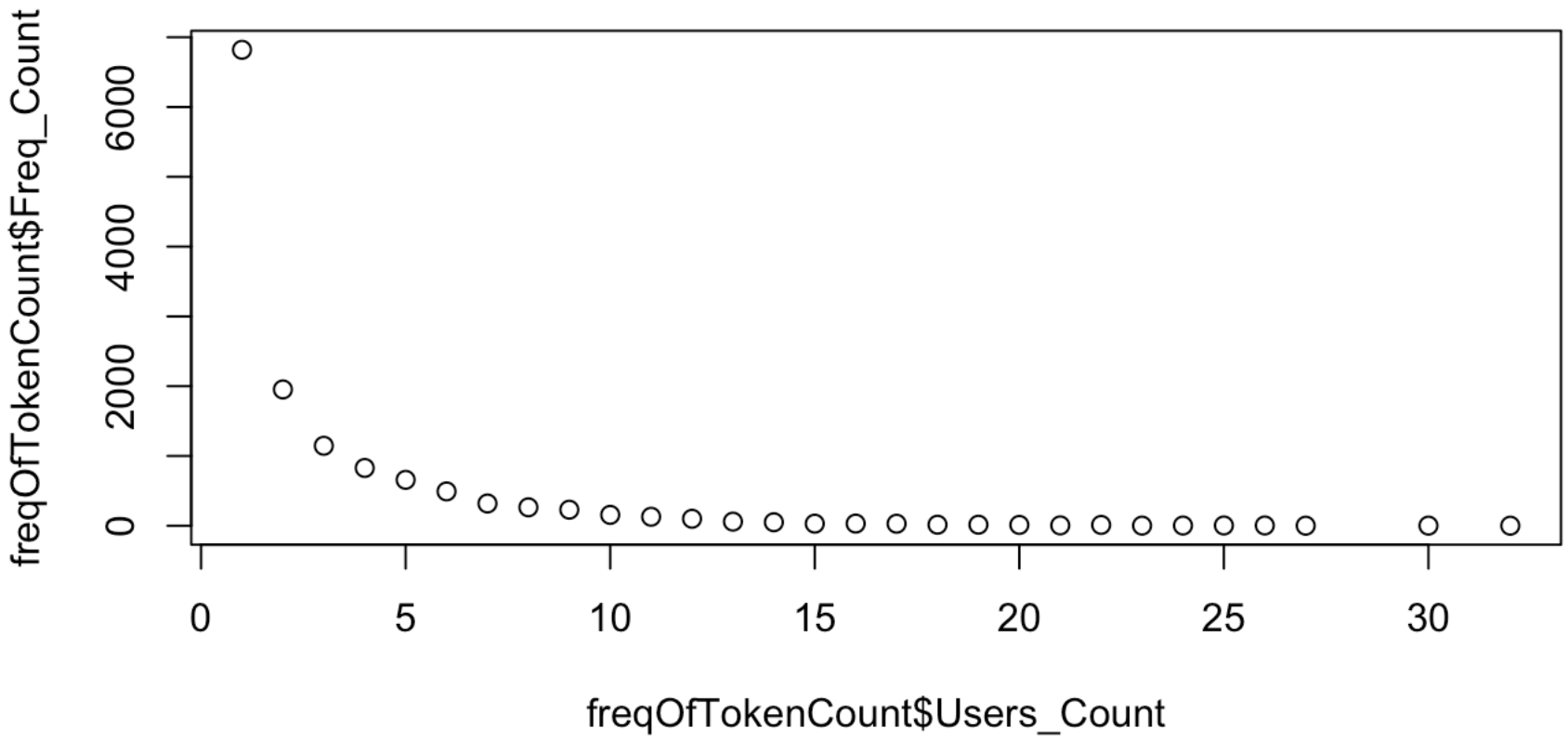
We go through all the other tokens in given dataset and find out how many tokens each user buys/sells

```
col_names2 <- 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_names2))
  tusers <- unique(append(t$FROMNODE, t$TONODE))
  uniqueUsersForAllTokens <- append(uniqueUsersForAllTokens,tusers)
}
usersFromAllTokens <- do.call(rbind.data.frame, uniqueUsersForAllTokens); colnames(us
ersFromAllTokens)<-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$USERID[usersFromAllTokens$USERI
D %in% activeUsers$USERS]))
colnames(userTokenCount) <-  c("USERID", "COUNT")
freqOfTokenCount <- count(userTokenCount, "COUNT"); colnames(freqOfTokenCount) <- c("
Users_Count", "Freq_Count")
plot(freqOfTokenCount$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$Freq_Count)
freqOfTokenCount <- subset(freqOfTokenCount, Freq_Count<max(newSet3[complete.cases(ne
wSet3)]) & Freq_Count>min(newSet3[complete.cases(newSet3)]))
descdist(freqOfTokenCount$Freq_Count, boot= 500, graph=FALSE)
```

```
## 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$Freq_Count, "pois", method ="m
le")
distributionFit_Count_wb <- fitdist(freqOfTokenCount$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_pois$loglik; distributionFit_Count_wb$loglik; distributionFit_C
ount_ln$loglik; distributionFit_Count_gm$loglik
```

```
## [1] -2313.547
```

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

```
## [1] -125.3001
```

```
## [1] -126.8819
```

Observations

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.

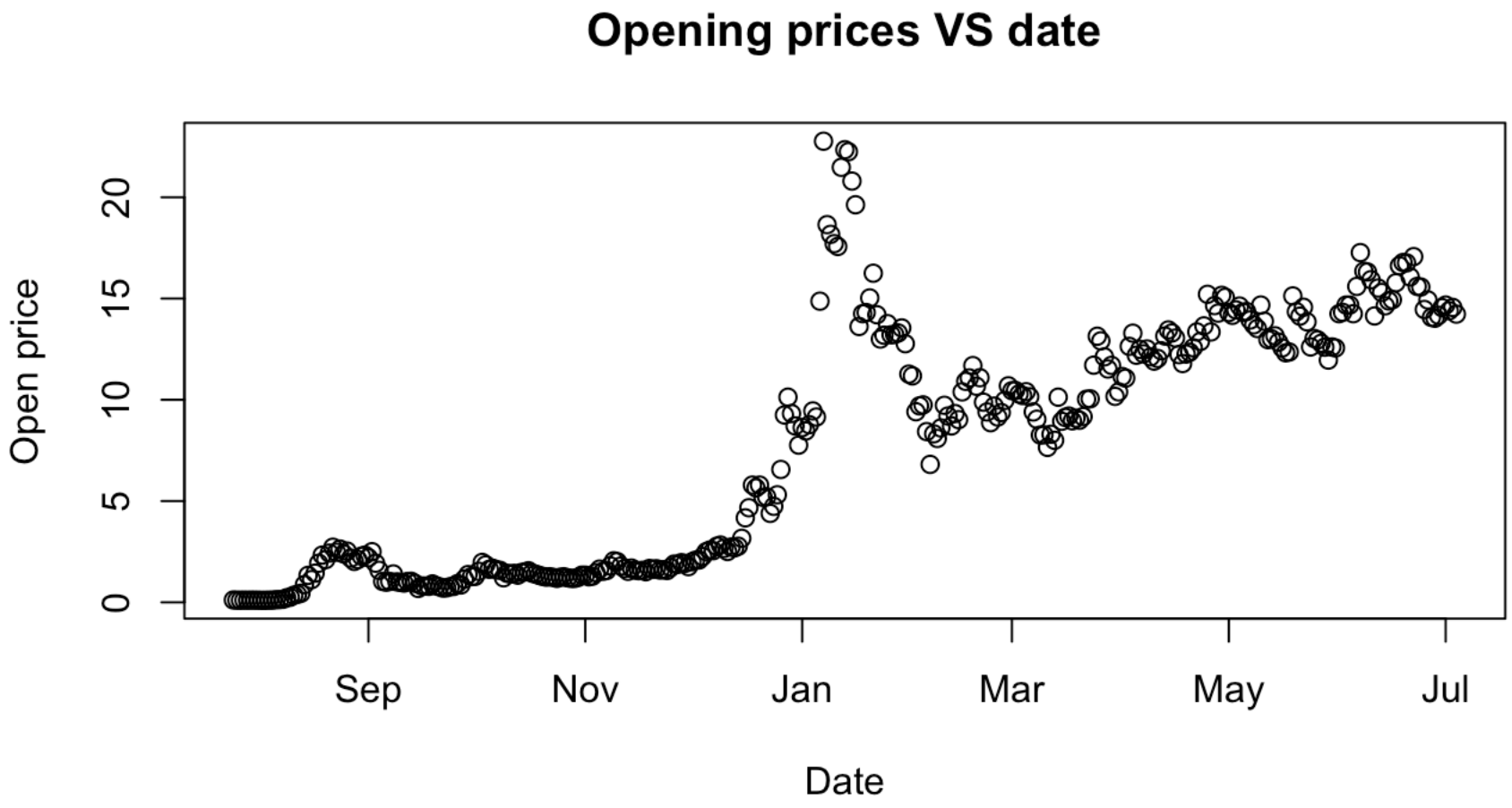
Study 3:

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.

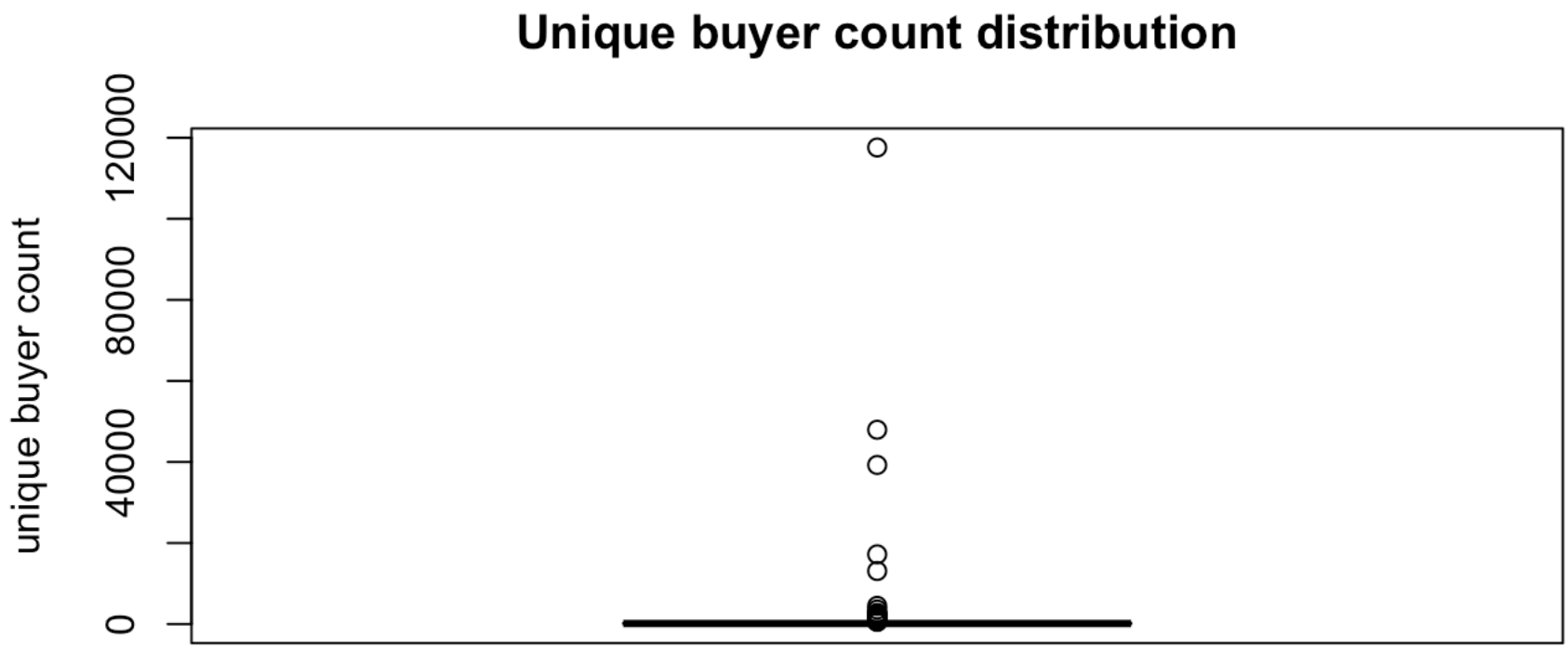
```
timePrices <- subset(myPrices, select=c("Date","Open", "Close"))
timePrices$Date <- as.Date(timePrices$Date, "%Y-%m-%d"); timePrices <- unique(timePri
ces)
plot(timePrices$Date, timePrices$Open, main = "Opening prices VS date", xlab = "Date"
, ylab="Open price")
```



Studying the distribution of number of unique buyers each day.

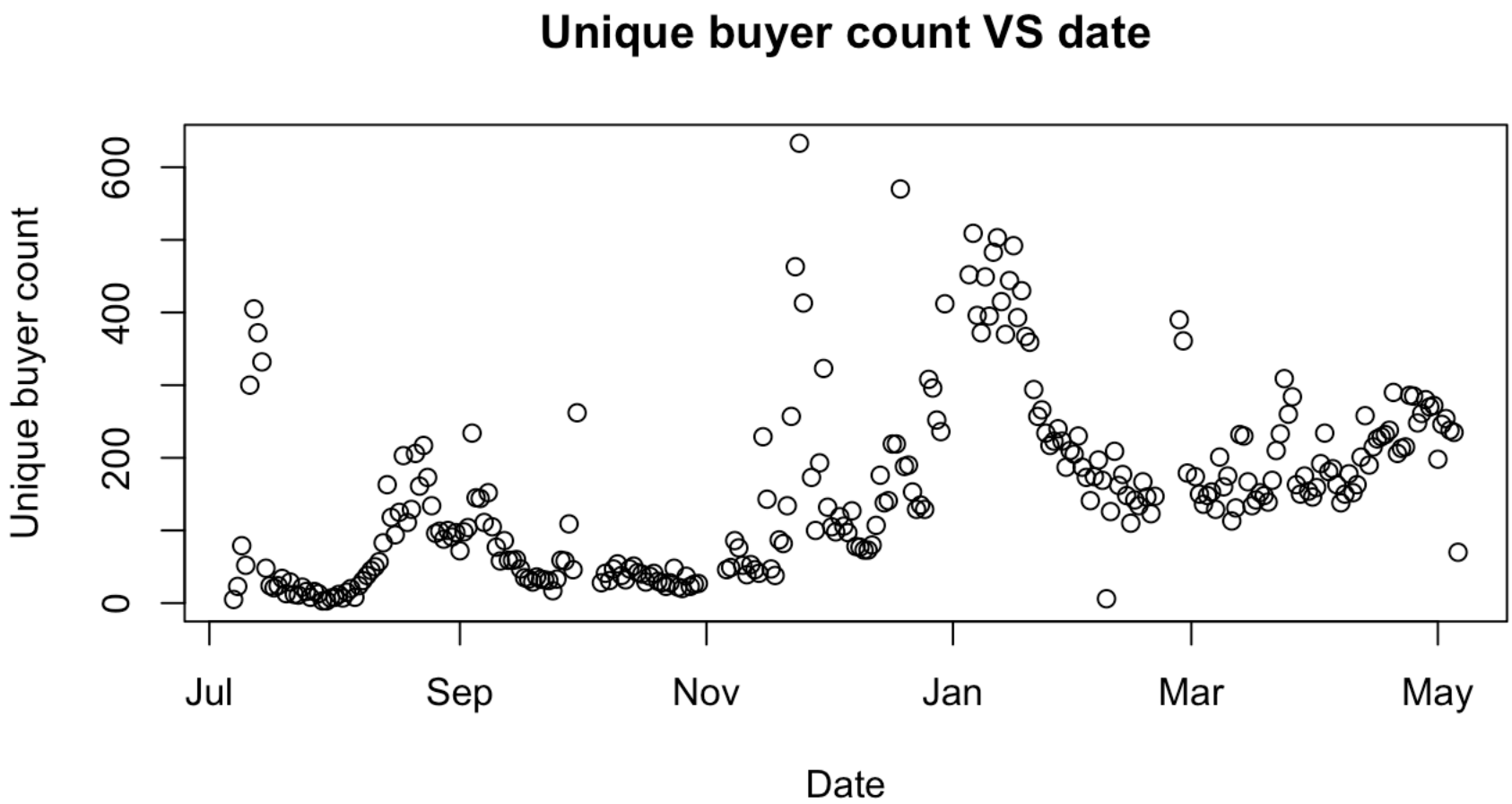
Study the pattern of how many unique buyers are there for our token each day. We see outliers in this data.

```
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(time
BuyFreq)
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.

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

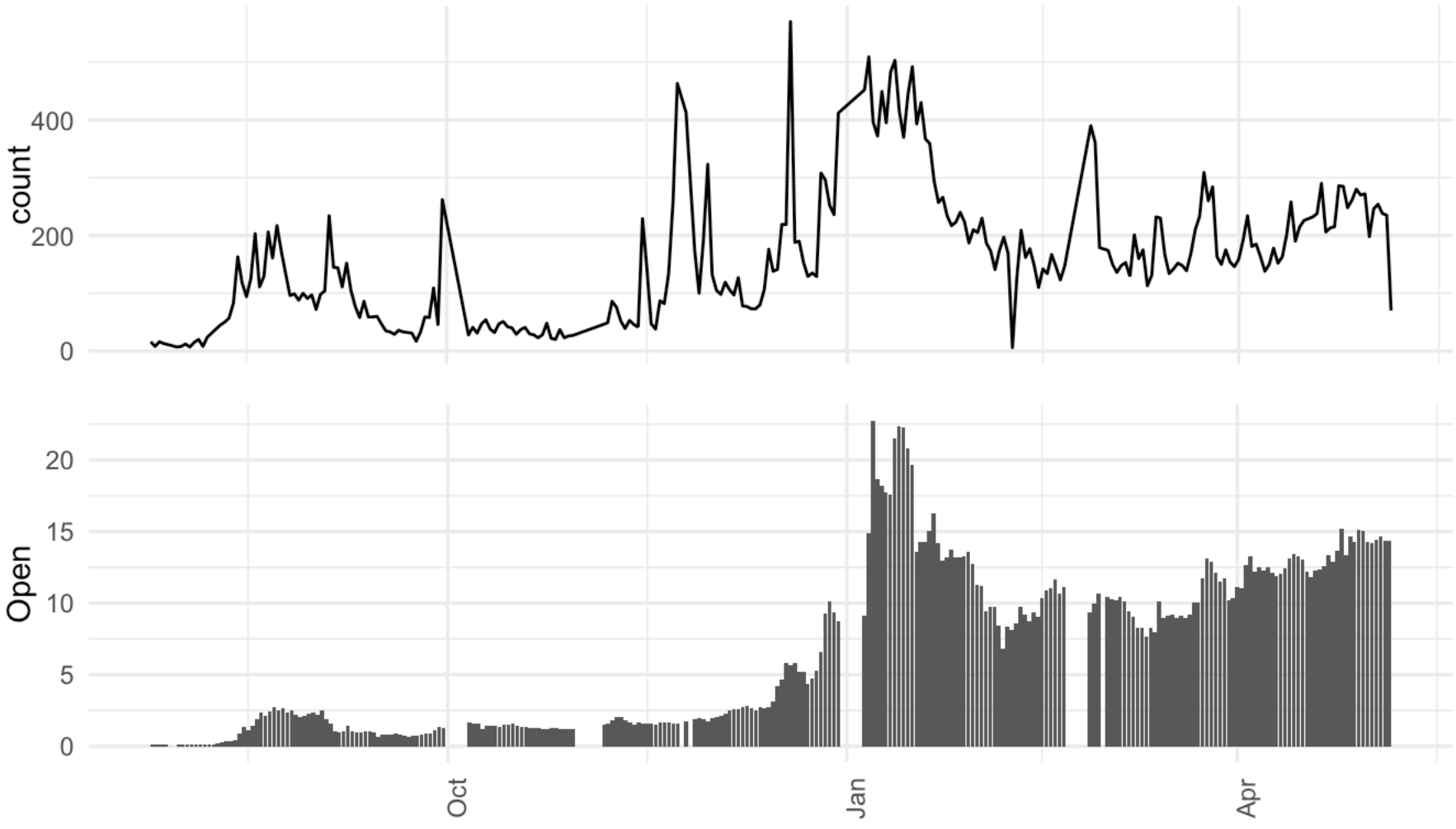
```
priceSellForEachDay <- merge(x=timePrices, y=timeBuyFreq, by.x=c("Date"), by.y = c("D
ATE"))
newSet <- remove_outliers(priceSellForEachDay$count)
priceSellForEachDay <- subset(priceSellForEachDay, count<max(newSet[complete.cases(ne
wSet)]) & count>min(newSet[complete.cases(newSet)]))
cor(priceSellForEachDay$Open, priceSellForEachDay$count, method=c("pearson"))
```

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

Observations

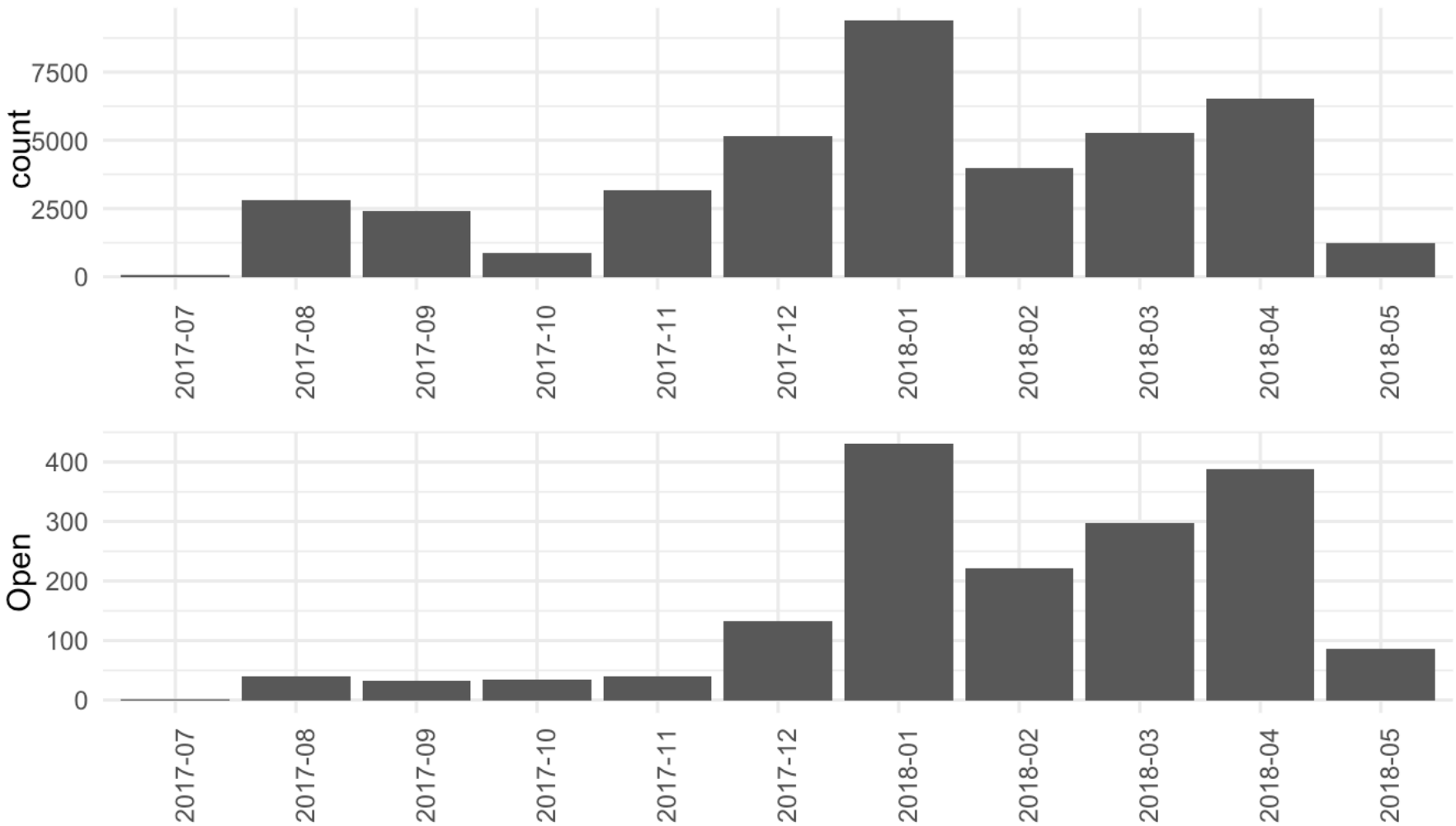
We find a very strong positive correlation between the number of people buying BNB token in a day to the price of the token that day. This means that user decisions get highly effected by the opening price of bnb token on that day. So we combine both plots to visualize the correlation.

```
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"))
```



We also see the month wise distribution of number of buyers and the price of token.From this plot we get see that in January 2018 the number of transactions is the highest and there it has the maximum positive correlation among the properties. Further we try to develop models based on data from the whole dataset and only january.

```
p1 <- ggplot(priceSellForEachDay, aes(factor(format(Date, "%Y-%m")), count)) + geom_b
ar(stat="identity") + theme_minimal() + theme(axis.title.x = element_blank(), axis.te
xt.x = element_text(angle=90))
p2 <- ggplot(priceSellForEachDay,aes(factor(format(Date, "%Y-%m")), 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 4

We try to create a linear regression model to predict the price return for the upcoming day. Simple price return is given as

$$P_t - (P_t - 1)/P_t - 1$$

We experiment on various features from our dataset, like number of users buying the token on a given day, closing price of the token on previous day, opening price of token on previous day, number of time the token in bought in a day, number of time the token is sold in a day. Three regressors are chosen from a series of experiments which give the highest value of r-squared for predicting price return of next day. The features we found best fitting are:

1.closing price of token
2.number of transactions
3.number of users buying the token

Compare opening price and number of transactions for each day

```
timeBuyFreq1 <- ddply(mydata, .(DATE), mutate, count = length(TONODE))
timeBuyFreq1 <- subset(timeBuyFreq1, select=c("DATE", "count"))
timeBuyFreq1$DATE <- as.Date(timeBuyFreq1$DATE, "%Y-%m-%d"); timeBuyFreq1 <- unique(t
imeBuyFreq1)
priceSellForEachDay1 <- merge(x=timePrices, y=timeBuyFreq1, by.x=c("Date"), by.y = c(
"DATE"))
newSet1 <- remove_outliers(priceSellForEachDay1$count)
priceSellForEachDay1 <- subset(priceSellForEachDay1, count<max(newSet1[complete.cases
(newSet1)]) & count>min(newSet1[complete.cases(newSet1)]))
```

Compare opening price and closing price of previous day for each day

```
y2=list(); y2 <- append(y2,0)
for(i in 2:nrow(priceSellForEachDay)){
  y2 <- append(y2,priceSellForEachDay$Close[i-1])
}
priceSellForEachDay$ClosePrev <- unlist(y2, use.names=FALSE)
```

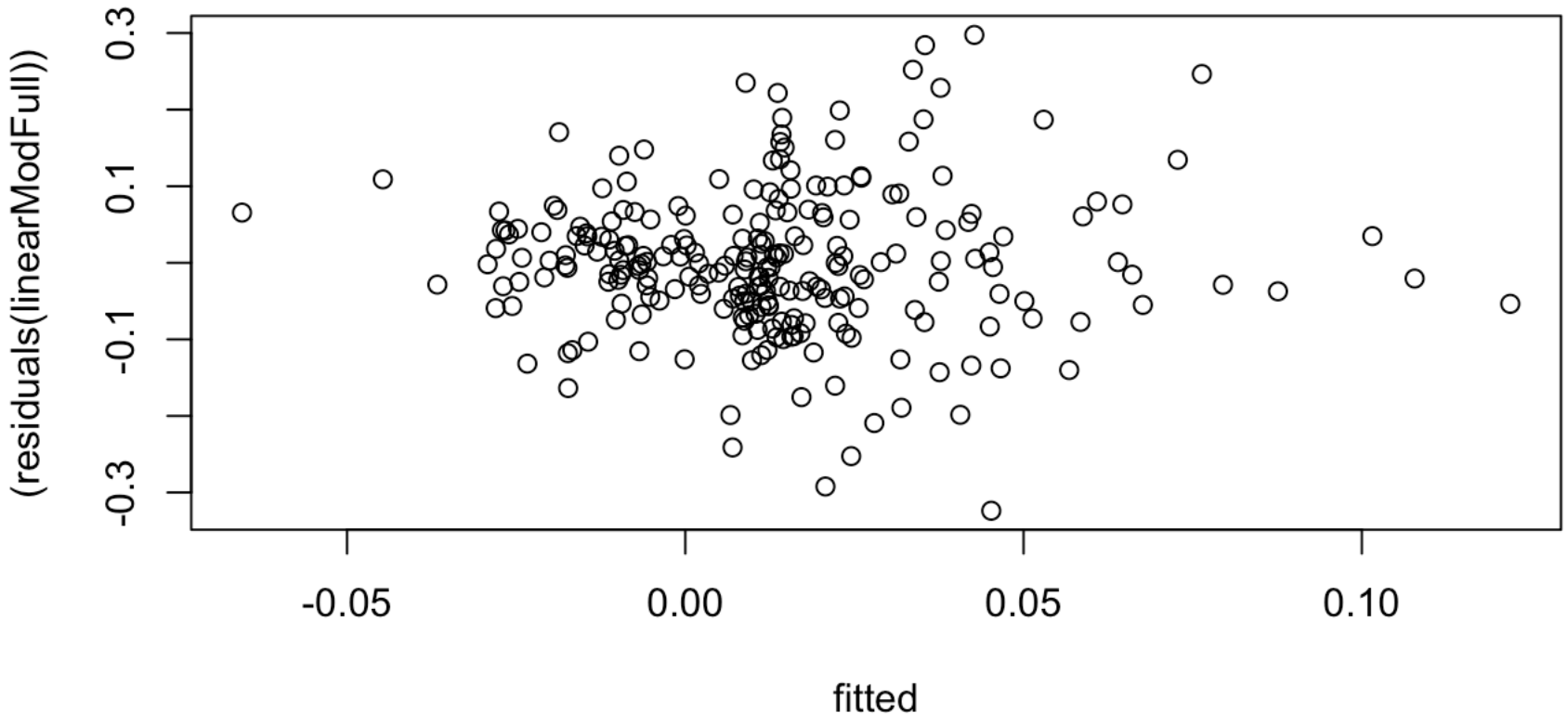
Merge all three regressors, and calculate price return

```
corDataSet <- merge(x= priceSellForEachDay, y=priceSellForEachDay1, by.x=c("Date"), b
y.y=c("Date"))
y=list()
for(i in 1:nrow(corDataSet)-1){
  y <- append(y,(corDataSet$Open.x[i+1]-corDataSet$Open.x[i])/corDataSet$Open.x[i])
}
y <- append(y,0)
corDataSet$priceReturn <- unlist(y, use.names=FALSE)
```

Building the prediction model

After studying the linear relationship pictorially in the scatter plot and by computing correlations, we build the multiple linear model. Linear fit occurs by minimising the mean squared error between the actual and predicted values. We build a linear model using all 3 regressors and entire dataset considering each day as one layer. We remove ouliers from the price return, which had a proportionately very high value for one day. After removing the oulier, we found significant increase in the outcome of the linear regression model. There is no pattern found in the residuals

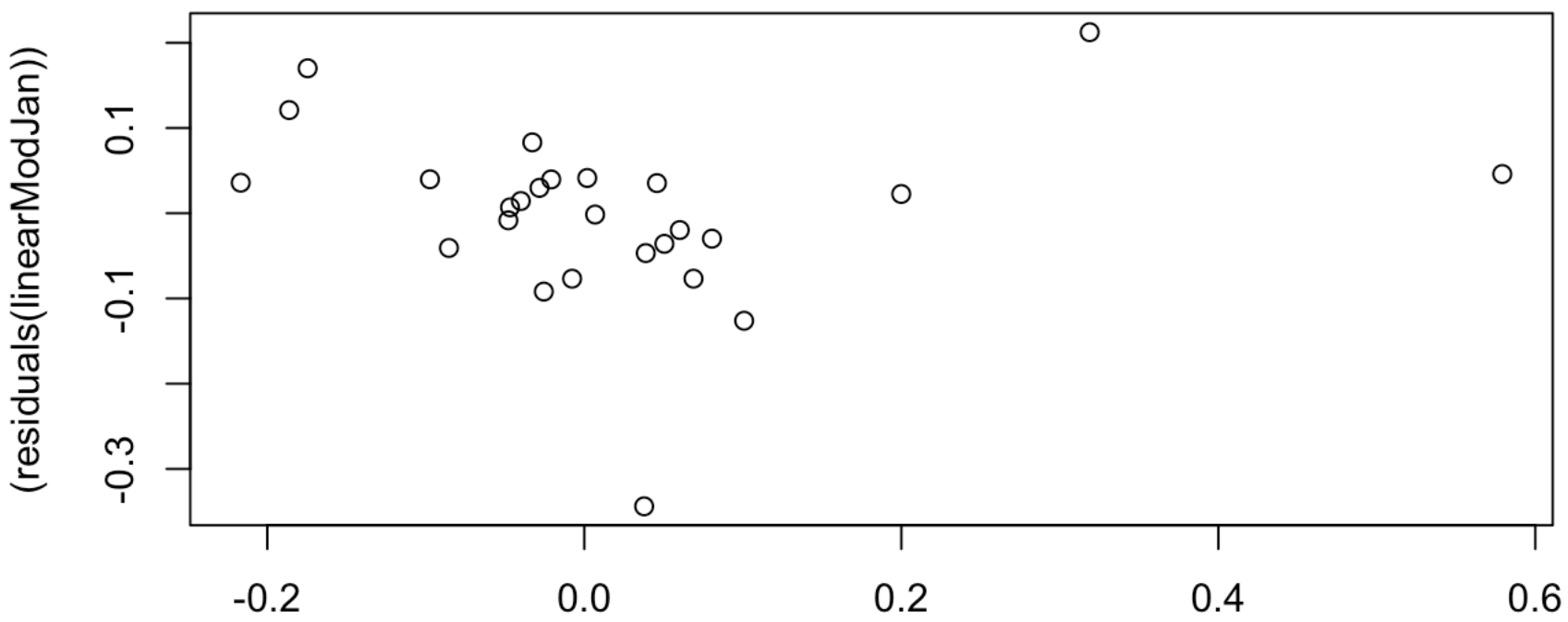
```
newSet11 <- remove_outliers(corDataSet$priceReturn)
corDataSet11 <- subset(corDataSet, priceReturn<max(newSet11[complete.cases(newSet11)]
) & priceReturn>min(newSet11[complete.cases(newSet11)]))
linearModFull <- lm( priceReturn ~  (ClosePrev)^2  + (count.y)^2 + (count.x)^3, data=
corDataSet11)
fitted = fitted(linearModFull); plot(fitted, (residuals(linearModFull)));  summary(li
nearModFull)
```




```
##
## Call:
## lm(formula = priceReturn ~ (ClosePrev)^2 + (count.y)^2 + (count.x)^3,
##     data = corDataSet11)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.32379 -0.05351 -0.00346  0.05370  0.29756
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.489e-03  1.079e-02   0.879    0.380
## ClosePrev   -6.908e-03  1.694e-03  -4.079 6.17e-05 ***
## count.y      2.062e-04  1.536e-04   1.343    0.181
## count.x      5.178e-05  2.106e-04   0.246    0.806
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09617 on 239 degrees of freedom
## Multiple R-squared:  0.06845,    Adjusted R-squared:  0.05675
## F-statistic: 5.854 on 3 and 239 DF,  p-value: 0.0007133
```

We also tried to form a model on a single layer, month of january 2018, having the highest activity as seen above. We see that our chosen features give a high r-squared value on the data of this layer. There is no pattern found in the residuals

```
highCorData <- subset(corDataSet, format.Date(Date, "%m")=="01")
linearModJan <- lm( priceReturn ~  (ClosePrev)^2 + (count.x)^3 + (count.y), data=high
CorData)
fitted = fitted(linearModJan); plot(fitted, (residuals(linearModJan))); summary(linea
rModJan)
```




```
##
## Call:
## lm(formula = priceReturn ~ (ClosePrev)^2 + (count.x)^3 + (count.y),
##     data = highCorData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.34391 -0.03953  0.01061  0.03979  0.21243
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.2278721  0.0975209   2.337  0.02897 *
## ClosePrev   -0.0459219  0.0069740  -6.585 1.27e-06 ***
## count.x      0.0023184  0.0005042   4.598  0.00014 ***
## count.y     -0.0005958  0.0002932  -2.032  0.05440 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1103 on 22 degrees of freedom
## Multiple R-squared:  0.7034, Adjusted R-squared:  0.6629
## F-statistic: 17.39 on 3 and 22 DF,  p-value: 5.162e-06
```

Summary and Conclusion

We found an average positive correlation among the number of users transaction in a particular day to the token price on next day. Also, the closing price of previous day had a positive correlation to the price return of next day and was a helpful feature in building the model. But the number of unique users transacting in a day is not a strong feature to reject our null hypothesis and declare that there is a relation between them. We conlude that the closing price and number of transactions happening in a day have a significant effect on the price return of next day and considering the high number of data points, we establish a co-relation among these features but a low r-sqaured value denoted that it is not sufficient information to build a strong predictive model.

Although, on considering monthly layers, there was not a consistent positive/negative correlation among the features as seen from a random scatter of residuals although the r-squared value is less. We concluded that the BNB token has a rather dynamic behaviour over the entirety of its lifetime. For a single layer, in the month of January 2018, there was a significant relation between our features and price return. We get a r-squared value of around 65% and a scattered residual plot, which makes for a decently accurate model linear predictive model.

References

1.*http://snap.stanford.edu/class/cs224w-2015/projects_2015/Using_the_Bitcoin_Transaction_Graph_to_Predict_the_Price_of_Bitcoin.pdf* *(http://snap.stanford.edu/class/cs224w-2015/projects_2015/Using_the_Bitcoin_Transaction_Graph_to_Predict_the_Price_of_Bitcoin.pdf*)*

*2.*https://coinmarketcap.com/* (https://coinmarketcap.com/*)