CS6313.002 Statistical Project

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 4.1, we found how many times a user buys or sells the STORJ token, then fit a distribution and estimate the best distribution.In 4.2, we find the correlation between the STORJ Token price data and layer features of the STORJ Token.In 4.3, we find the most active buyer and seller of our STORJ token and then track them in all other tokens, plot how many unique tokens have they invested in the provided time frame then fit and estimate distributions as part 1. In 4.4, we created a multiple linear regression model to explain price return on day t.

source code: https://github.com/vineetamonkar/r_project_2/blob/master/StatFinalReport.Rmd

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.

(a) from_node : Id which sells the token in the transaction

(b) to_node : Id which buys the token in the transaction

(c) unixtime : Unix time of the transaction

(d) totalamount : Total amount of the storj token involved in the transaction 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: Price dataset for storj token it contains 379 rows and 7 columns as follows:

(a) Date

(b) Open : Opening price of the token on that day

(c) High : Max price of the token on that day

(d) Low : Min price of the token on that day

(e) Close : Closing price of the token on that day

(f) Volume : Volume of the token on that day

(g) Market Cap: Market Cap of that token on that day We use price data in part 2 and 4.

3. Preprocessing

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 Finad and Fit Distribution

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

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

```
# fit distribution

# Poisson Distribution
library(fitdistrplus)
```

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

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

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

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

```
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)
```

The sells part is similiar to buys part, so we didn't show the code in this report, the follow is the plot of sells:

4.2 Calulate Correlations of Token Data and Price Data

```
# 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")
```

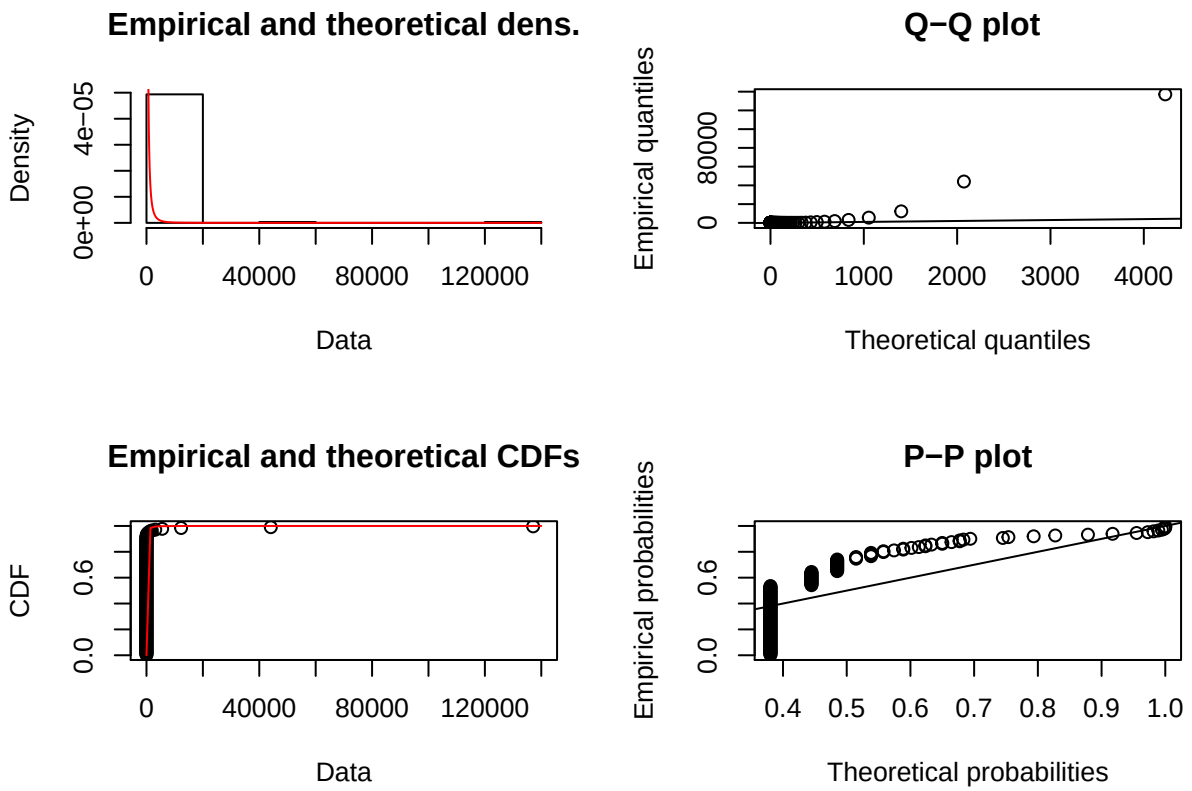

Figure 1: Fitting with Weibull Distribution-Buys

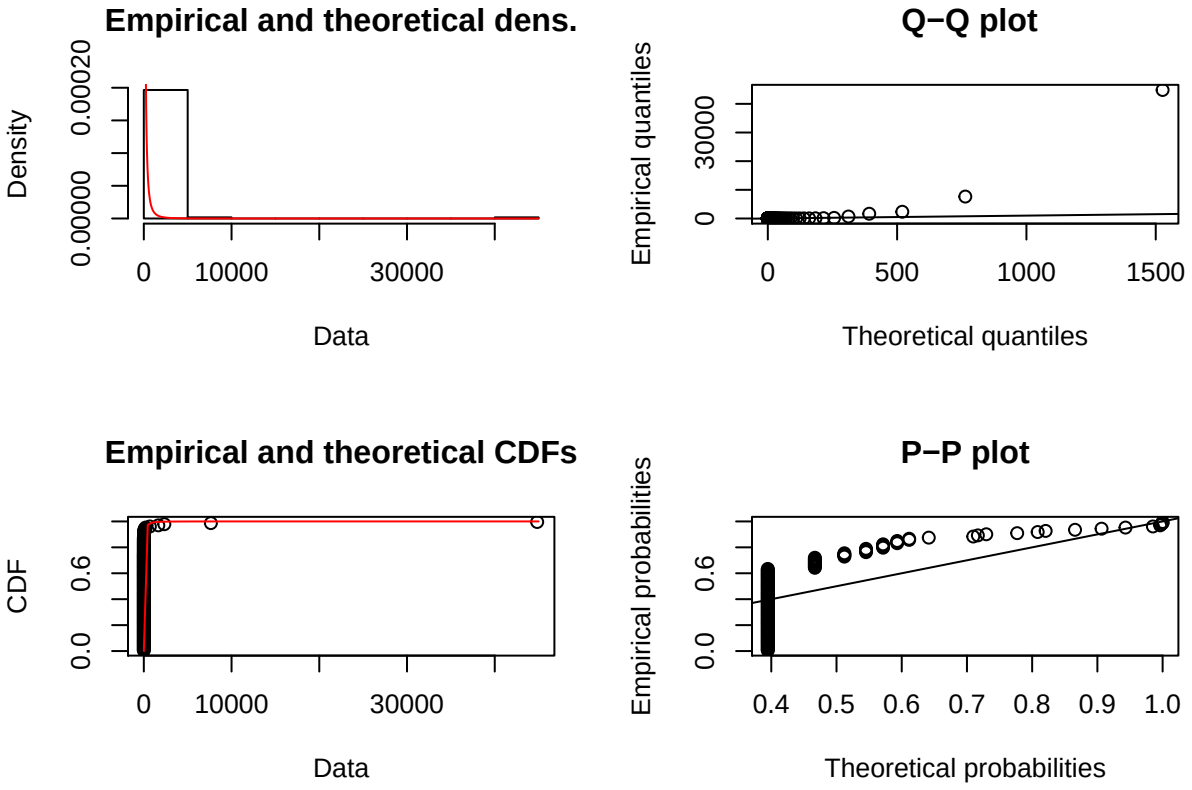

Figure 2: Fitting with Weibull Distribution-Selles

```r
# Join Two Data by Date
CombineData <- merge(x = WithoutOutlierdata, y = priceData, by = "Date")
buyerstable <- aggregate(data=CombineData, toNode ~ Date, function(x) length(unique(x)))
colnames(buyerstable) <- c("Date","unique_buyers")
CombineData <- merge(CombineData,buyerstable,by = "Date")
CombineData <- unique(CombineData[,c(1,4:11)])

## 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
```

```r
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$totalAmount < value),]
  correlation <- cor(layer$unique_buyers, layer$Open,use="complete.obs",method = "pearson")
  print(correlation)
}
```

```
## [1] -0.1193053
## [1] -0.1379293
## [1] -0.1628312
## [1] -0.0212934
## [1] 0.07903625
## [1] 0.2030551
## [1] 0.1708239
## [1] 0.2684059
## [1] 0.3257792
## [1] 0.4943405
## [1] 0.4705017
## [1] 0.4239833
## [1] 0.4032195
## [1] 0.4014254
## [1] 0.4011302
```

4.3 Most Active Buyer and Seller Analysis

```r
# 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
```

```r
# 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")
```

```r
# 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")
      }
    }
  }
}
```

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

```r
# 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")
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)
```

The sellers part is similiar to the above, we didn't show the code in this report.

4.4 Create Linear Regression Model

In this part we created a multiple linear regression model on the price data and our primary token data.Our response variable(y) is simple price return given by $p_t - p_{t-1}/p_{t-1}$ where, p_t is the token price in dollar for t_{th} day.

The regressor variables(x1...xn) are as follows: 1)No of token transactions per day 2)No of total token bought or sold per day 3)No of unique buyers per day 4)Percentage of investors who bought more than token per day.

Calculate response variable from the price data using open price. The first few records of the response variable as follows:

```r
return <- numeric(nrow(priceData))
return[379] <- priceData$Open[379]
```

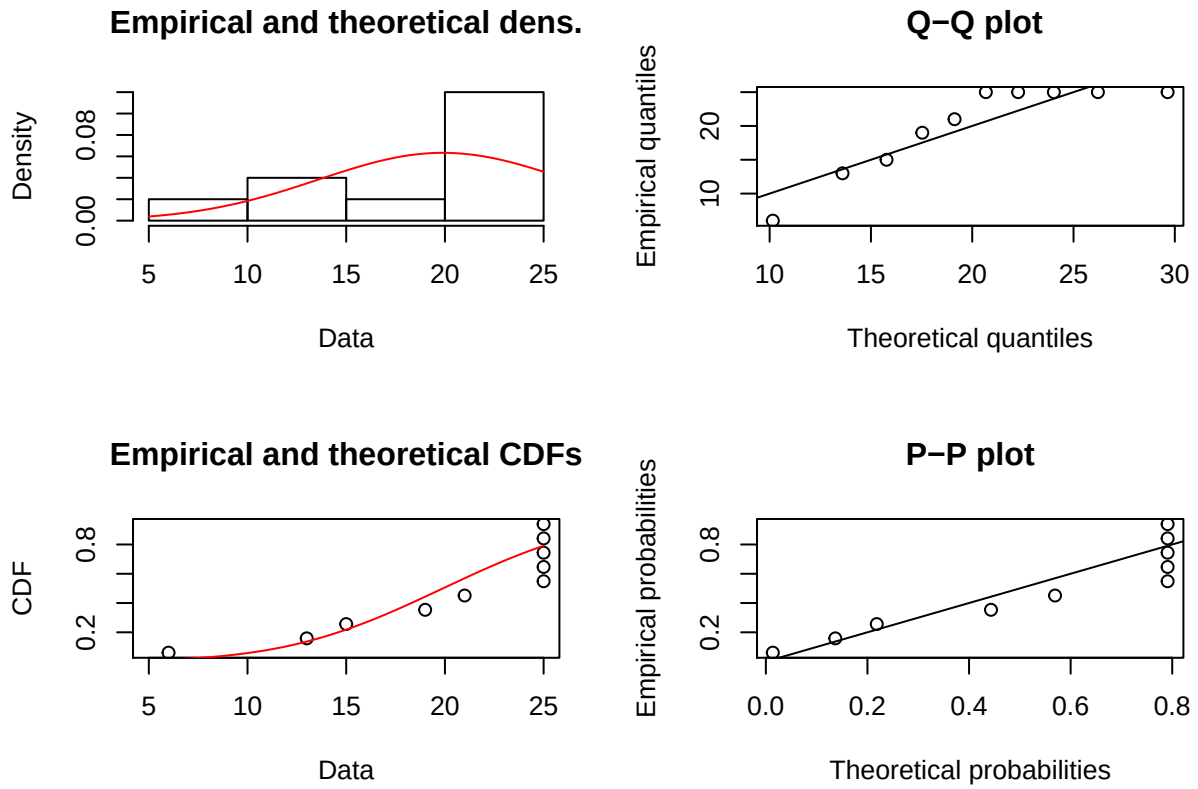

Figure 3: Fitting with Normal Distribution - Most Active Buyer

```
for (i in 378:1) {
  return[i] <- (priceData$Open[i]-priceData$Open[i+1])/priceData$Open[i+1]
}
priceData$return <- return
head(priceData$return)
```

```
## [1] -0.013908447  0.016677813 -0.064007249  0.008985674 -0.096438387
## [6] -0.013837618
```

Merge both the datasets according to date which would help us in calculating the regressor variables.

```
colnames(newwithoutoutlierdata)<-c('fromNode','toNode','Date','totalAmount')
newdataset <- merge(newwithoutoutlierdata,priceData,by = "Date")
```

4.4.1 Model 1:Simple Linear Regression

```
## [1] 1655  252  446  240  267  217
```

Data preparation: univariate and bivariate analysis. **Univariate Analysis**:

Check for outliers and then remove them by capping our data to the max value.To remove majority of the outlier we are capping the data 92% quantile. Then we will alsocheck the response variable.

But for simple return response varaible case we won't be removing any outliers as we don't want to have any impact on the response variable.

Bivariate Analysis: Draw the scatterplot between the regressor and the response variable.Before we do that it is important to shift our response variable up by one row because we will be using yesterday to model

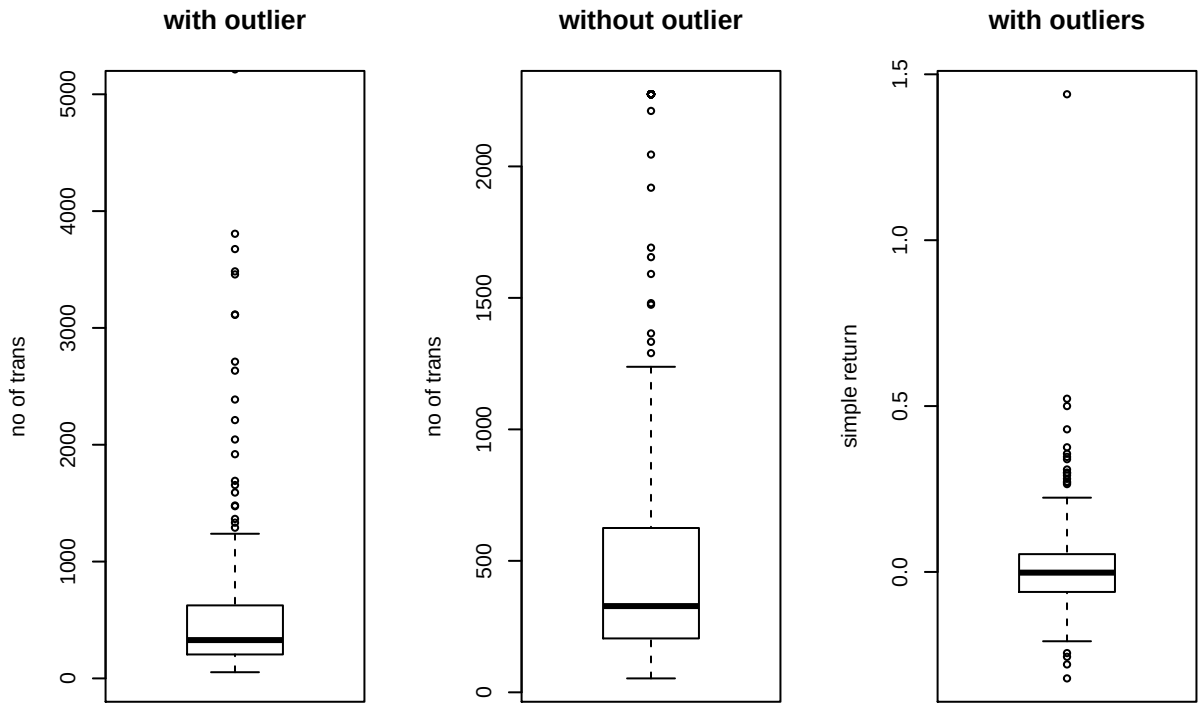

Figure 4: Boxplots of variables for Univariate analysis in simple regression

today's response.

```
shift <- function(x, n){
  c(x[-(seq(n))], rep(NA, n))
}
newsimpletable$return <- shift(newsimpletable$return, 1)
newsimpletable <- newsimpletable[1:(nrow(newsimpletable)-1),]
head(newsimpletable$return)
```

```
## [1]  0.04521051  0.49999359 -0.02718869 -0.17318319 -0.02977607 -0.18217782
```

```
#plot(newsimpletable$no_of_trans,newsimpletable$return)
```

We can see that there isn't much relation between no of transactions each day and the simple return variable. We can confirm this by finding the correlation between the two

```
cor(newsimpletable[,c(6,7)])
```

```
##                 return no_of_trans
## return       1.0000000   0.1422191
## no_of_trans 0.1422191   1.0000000
```

After completing the analysis we move ahead to fit the linear model.

```
final_data <- lm(return~no_of_trans,data=newsimpletable)
summary(final_data)
```

```
##
## Call:
## lm(formula = return ~ no_of_trans, data = newsimpletable)
##
## Residuals:
```

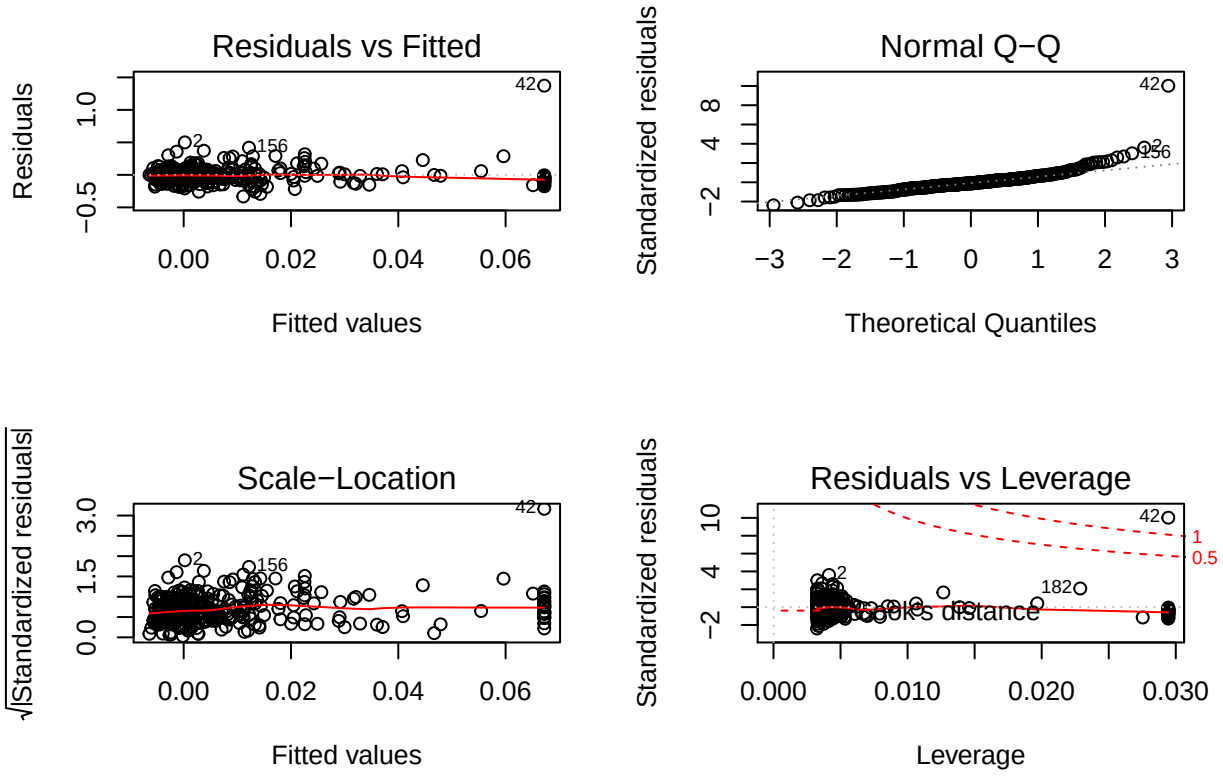

Figure 5: Residual Plot of the linear model

```
##      Min       1Q   Median       3Q      Max
## -0.33197 -0.07325 -0.01472  0.04961  1.37263
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -8.136e-03  1.089e-02  -0.747   0.4556
## no_of_trans  3.313e-05  1.318e-05   2.513   0.0125 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.139 on 306 degrees of freedom
## Multiple R-squared:  0.02023,    Adjusted R-squared:  0.01702
## F-statistic: 6.317 on 1 and 306 DF,  p-value: 0.01247
```

From the summary of the linear model we can clearly see that it is not a good model, the p-values of the intercept and the regressor are very high, also the r-square conveys only 2% of the response variable.

Testing

We can test this linear model using the plot function:

1. **Residuals vs fitted**:

Even though we see a horizontal line in this plot we can see that the residuals are not evenly spread across the line.It may not suggest a pattern but we can see most of the points in the left speculating a non-linear behaviour

2. **Normal Q-Q**:

The plot leads us to believe that the residuals follow normality as most of the points are seen to lie on the

line or near it although few observations such as 42,2 and 156 are significantly away from the line.

3. **Scale-Location**:

The horizontal line in this plot suggests that homoscedasticity is observed in the residuals.

4. **Residuals vs leverage**:

This plots finds the observations which have a high influence on the regression model according to the cooks distance. We can see that observation 42 is out of the region thus is an outlier which could have an impact on the model if we do remove or decide to keep it in the model.

4.4.2 Model 2:Multiple Linear Regression

Create three new features for our multiple regression model:

1. Total number of tokens involved per day:

First divide the total amount of each transaction with number of sub units of the storj token(10^8) to get the number of tokens, then we will group by and sum each amount according to the date

First few records are as follows

```
## [1]  820156.9  759898.0 1523895.9 1232037.0  777283.4 1355849.0
```

2. Number of Unique buyers per day:

Group the dataset according to the date and then count the number of unique buyers for each day.

First few records are as follows:

```
## [1] 1537  133  285  150  175  140
```

3. Percentage of investors who have bought more than 10 tokens:

Find the total number of investors per day and then find the total tokens bought by each investor per day. Then find the number of investors who bought more than 10 token that day.

First few records are as follows:

```
## [1] 33.37671 94.73684 94.73684 98.00000 94.28571 96.42857
```

Now that we have all the required features lets do the analysis as we did for the simple linear regression:

Univariate analysis:

Create a boxplot of unique buys regressor variable and then we check for the number of tokens(and capping at 94%) and percentage of investors variable:

After the univariate analysis lets do the **bivariate analysis**:

plot number of transactions vs return:
plot total token vs return:
plot unique buyers vs return:
plot percentage investors vs return:

After analysing lets create an initial multiple regression linear model, Use the car library to calculate variable inflation factor and find the multicollinearity.

```
final_data3 <- lm(return~no_of_trans+total_token+unique_buyers+percentage_investors,data = newsimpletabl
library(car)
vif(final_data3)
```

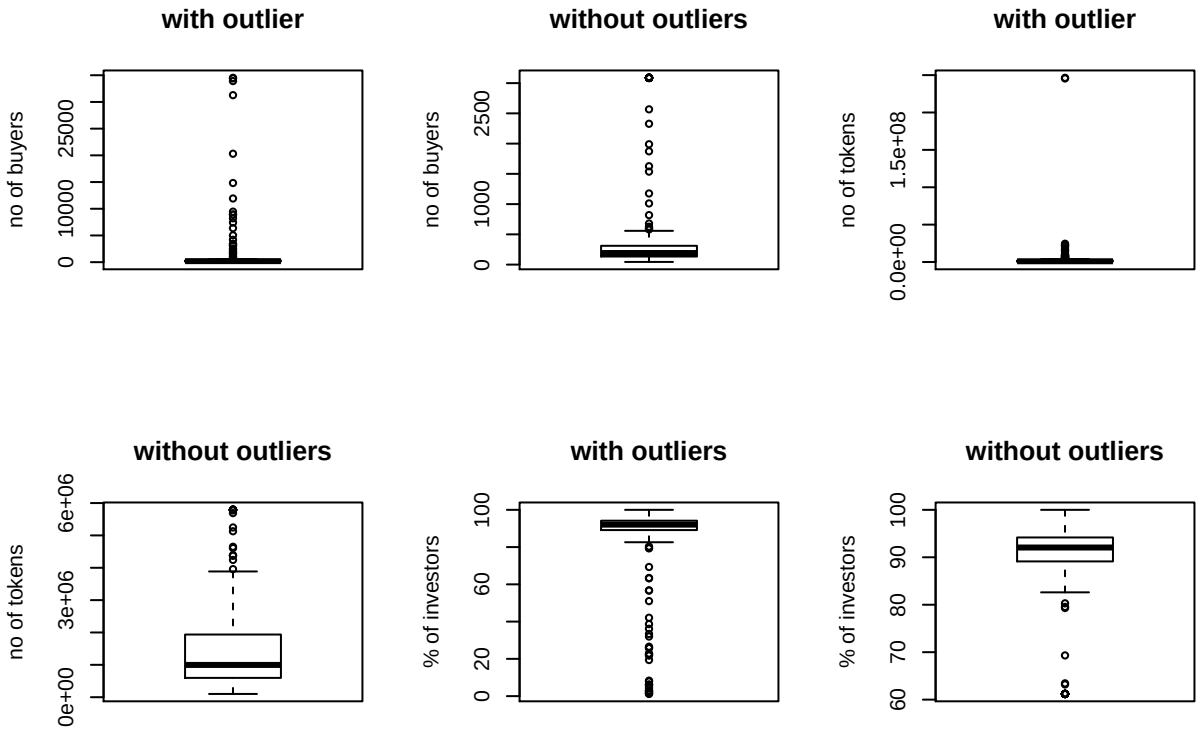

Figure 6: Boxplot if variables for univariate analysis in multiple regression

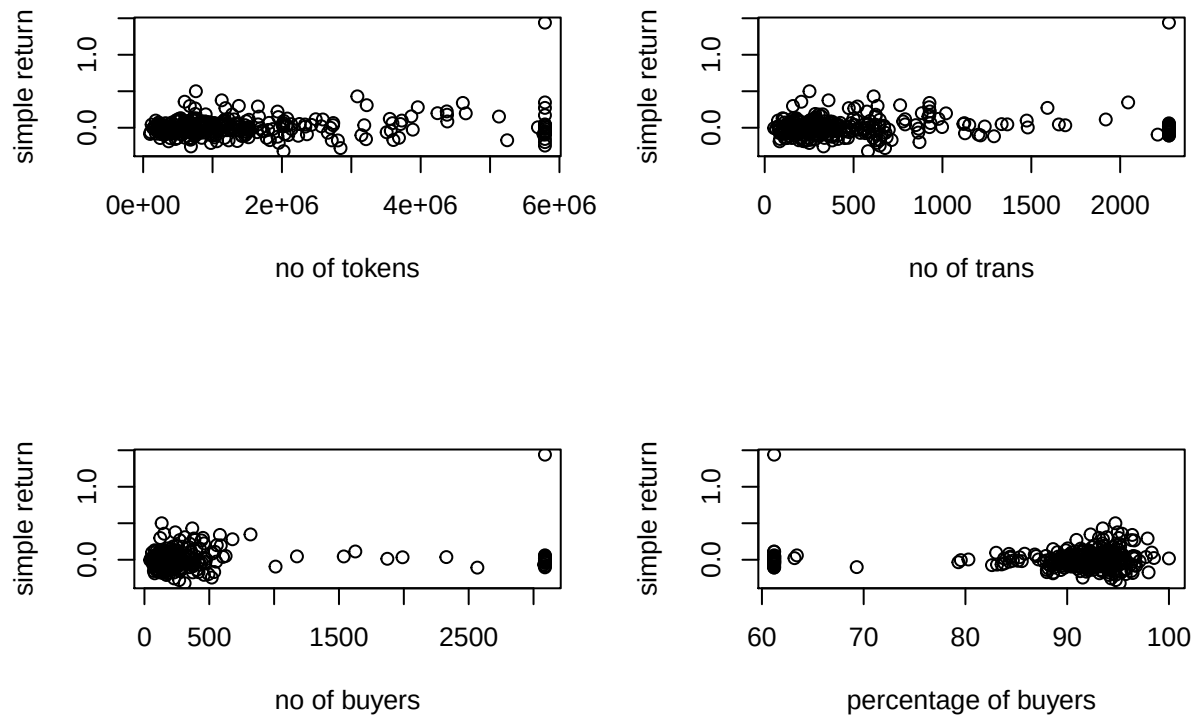

Figure 7: Scatter plot of variables for Bivariate analysis in multiple regression

```
##          no_of_trans          total_token        unique_buyers
##             5.226220             1.353319             6.124056
## percentage_investors
##             2.753391
```

Since the VIF for all the regressor variables is not much higher than 5 we will keep all of them in the initial model

```
summary(final_data3)
```

```
##
## Call:
## lm(formula = return ~ no_of_trans + total_token + unique_buyers +
##     percentage_investors, data = newsimpletable)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.34049 -0.06928 -0.00778  0.05409  1.32674
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -1.097e-01  1.202e-01  -0.913   0.3620
## no_of_trans           1.036e-05  3.001e-05   0.345   0.7302
## total_token           1.166e-08  6.087e-09   1.916   0.0564 .
## unique_buyers         2.340e-05  2.644e-05   0.885   0.3770
## percentage_investors  9.719e-04  1.302e-03   0.746   0.4560
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1385 on 303 degrees of freedom
## Multiple R-squared:  0.03767,    Adjusted R-squared:  0.02496
## F-statistic: 2.965 on 4 and 303 DF,  p-value: 0.01996
```

From the summary, p-value for the variables is very high suggesting that there is no linear relation between the regressor variable and the response variables.But before giving up on this we haven't yet used any features from the price data lets use the inherent features such as open, low, high etc to create a new linear model and see how we fare:

First find the multicollinearity and then use the step function to find the right combination of features:

```
##          no_of_trans          total_token        unique_buyers
##             5.520693             1.636274             6.273904
## percentage_investors                 Open                 High
##             2.805405            31.310888            18.042726
##                  Low
##            26.762959
```

Finally, Check the summary and plot of our final data model:

```
##
## Call:
## lm(formula = return ~ total_token + Open + High + Low, data = newsimpletable)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.43984 -0.03813  0.00046  0.03325  0.32659
##
## Coefficients:
```

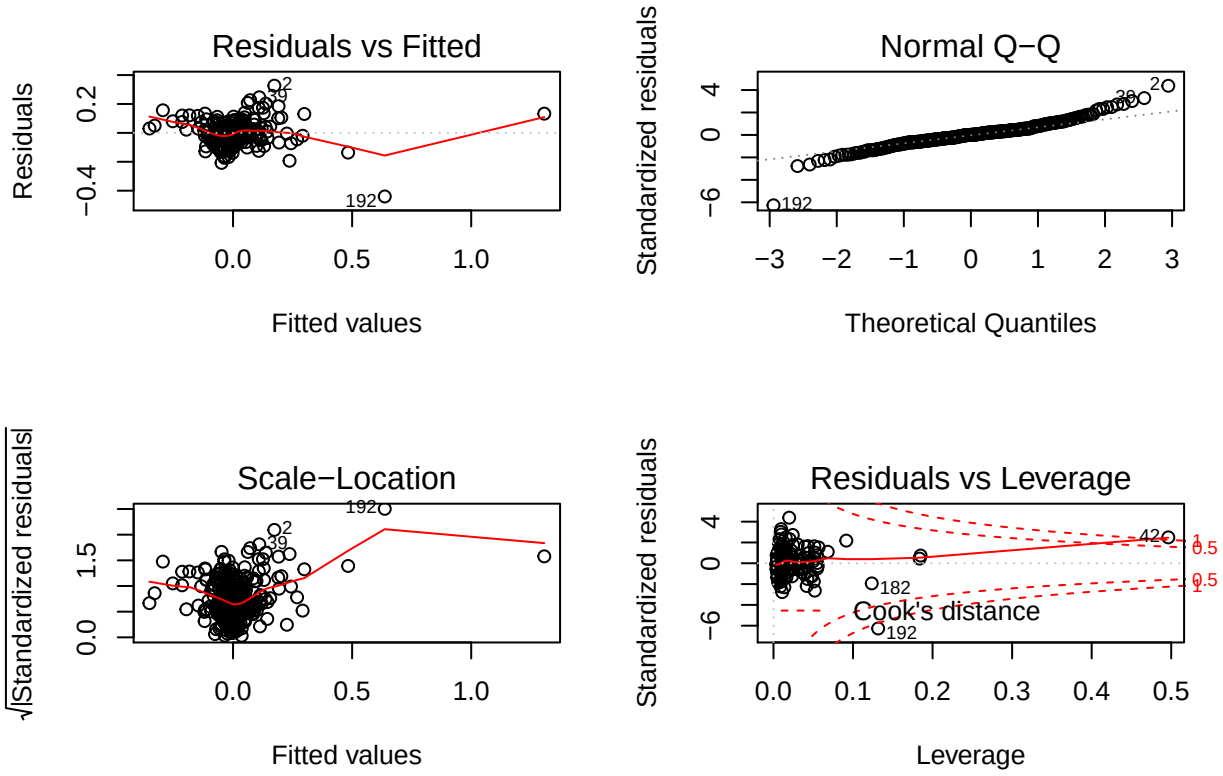

Figure 8: Residual plot of the linear model for multiple regression

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.630e-02  9.415e-03   1.731   0.0844 .
## total_token -6.124e-09  3.365e-09  -1.820   0.0697 .
## Open        -1.152e+00  4.946e-02 -23.300  < 2e-16 ***
## High         7.604e-01  3.364e-02  22.605  < 2e-16 ***
## Low          3.601e-01  5.156e-02   6.984 1.82e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0753 on 303 degrees of freedom
## Multiple R-squared:  0.7154, Adjusted R-squared:  0.7116
## F-statistic: 190.4 on 4 and 303 DF,  p-value: < 2.2e-16
```

5.Conclusion

5.1 Q1

We totally tried 5 distributions while fiting distributions with our data: Poisson Distribution, Weibull Distribution, Exponential Distribution, Geometric Distribution and 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.

5.2 Q2

The number of layers is 15. We tried several features of token data : (1) number of transactions(best correlation is 0.193) (2)Number of Unique buyers per day(best correlation is 0.14943) (3)Percentage of investors who have bought more than 10 tokens(best correlation is 0.1347)

Hence, the best correlation is 0.4943, when feature of token data is number of unique buyers per day.

5.3 Q3

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.

5.4 Q4

When we use the price data columns and remove the features which have high p-value, we get R^2 value of **71%**. We have analysed how we can use simple and multiple linear regression to create linear models by doing feature extraction on the storj token and price data. The model can be redeveloped and tested with different features to improve the R-squared value however the method of creation and analysis is mostly similar.