storjToken

Question 1:

library(readr)

## Warning: package 'readr' was built under R version 3.5.3

library(dplyr)

##   
## Attaching package: 'dplyr'

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

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

library(ggplot2)  
library(fitdistrplus)

## Warning: package 'fitdistrplus' was built under R version 3.5.3

## Loading required package: MASS

##   
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':  
##   
## select

## Loading required package: survival

## Loading required package: npsurv

## Loading required package: lsei

storj <- read\_delim('networkstorjTX.txt', delim = " ", col\_names = F)

## Parsed with column specification:  
## cols(  
## X1 = col\_double(),  
## X2 = col\_double(),  
## X3 = col\_double(),  
## X4 = col\_double()  
## )

print(storj)

## # A tibble: 406,131 x 4  
## X1 X2 X3 X4  
## <dbl> <dbl> <dbl> <dbl>  
## 1 6103619 5607448 1524611290 230769000000  
## 2 17 930403 1524611565 98700000000  
## 3 930403 5 1524612137 98700000000  
## 4 148332 1757147 1524612282 80312639339  
## 5 17 573 1524612421 212877929273  
## 6 573 5 1524612737 212877929273  
## 7 17 87160 1524613395 99700000000  
## 8 5 39994 1524613542 20341300000  
## 9 82 46 1524614390 206106470000  
## 10 171 573 1524614582 202600000000  
## # ... with 406,121 more rows

names(storj) <- c('fromID', 'toID', 'unixTime', 'tokenAmount')  
print(storj)

## # A tibble: 406,131 x 4  
## fromID toID unixTime tokenAmount  
## <dbl> <dbl> <dbl> <dbl>  
## 1 6103619 5607448 1524611290 230769000000  
## 2 17 930403 1524611565 98700000000  
## 3 930403 5 1524612137 98700000000  
## 4 148332 1757147 1524612282 80312639339  
## 5 17 573 1524612421 212877929273  
## 6 573 5 1524612737 212877929273  
## 7 17 87160 1524613395 99700000000  
## 8 5 39994 1524613542 20341300000  
## 9 82 46 1524614390 206106470000  
## 10 171 573 1524614582 202600000000  
## # ... with 406,121 more rows

decimals<-10^8  
supply<- 424999998.00001337  
totalSupply<- decimals \* supply  
print(totalSupply)

## [1] 4.25e+16

filteredstorj <- filter(storj,tokenAmount < totalSupply)  
print(filteredstorj)

## # A tibble: 406,078 x 4  
## fromID toID unixTime tokenAmount  
## <dbl> <dbl> <dbl> <dbl>  
## 1 6103619 5607448 1524611290 230769000000  
## 2 17 930403 1524611565 98700000000  
## 3 930403 5 1524612137 98700000000  
## 4 148332 1757147 1524612282 80312639339  
## 5 17 573 1524612421 212877929273  
## 6 573 5 1524612737 212877929273  
## 7 17 87160 1524613395 99700000000  
## 8 5 39994 1524613542 20341300000  
## 9 82 46 1524614390 206106470000  
## 10 171 573 1524614582 202600000000  
## # ... with 406,068 more rows

filteredstorj <- filter(storj,fromID != toID)  
print(filteredstorj)

## # A tibble: 402,453 x 4  
## fromID toID unixTime tokenAmount  
## <dbl> <dbl> <dbl> <dbl>  
## 1 6103619 5607448 1524611290 230769000000  
## 2 17 930403 1524611565 98700000000  
## 3 930403 5 1524612137 98700000000  
## 4 148332 1757147 1524612282 80312639339  
## 5 17 573 1524612421 212877929273  
## 6 573 5 1524612737 212877929273  
## 7 17 87160 1524613395 99700000000  
## 8 5 39994 1524613542 20341300000  
## 9 82 46 1524614390 206106470000  
## 10 171 573 1524614582 202600000000  
## # ... with 402,443 more rows

NoOfOutliers <- count(storj)-count(filteredstorj);  
print(NoOfOutliers)

## n  
## 1 3678

result <-filteredstorj %>% count(fromID,toID, sort = FALSE)  
names(result) <- c('fromID', 'toID', 'Occurences')  
names(result)

## [1] "fromID" "toID" "Occurences"

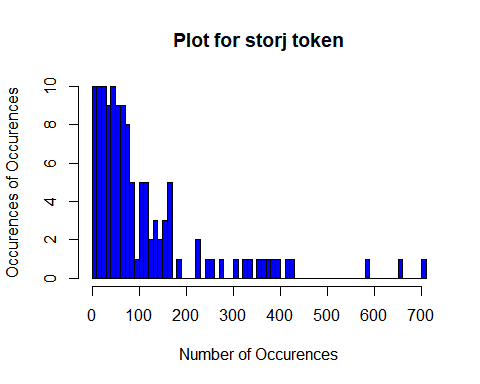
sum(result$Occurences)

## [1] 402453

result$Occ = 1  
result\_new <- aggregate(result$Occ, by=list(result$Occurences), FUN=sum)  
  
names(result\_new) <- c('Number','Occurences')  
head(result\_new)

## Number Occurences  
## 1 1 198834  
## 2 2 44017  
## 3 3 14089  
## 4 4 4207  
## 5 5 2216  
## 6 6 1396

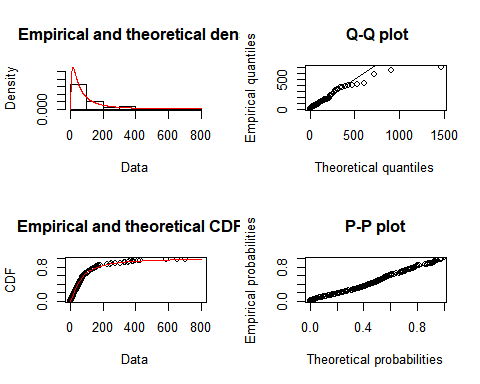
hist(result\_new$Number, breaks = 100, col = c("blue"), xlab = "Number of Occurences", ylab = "Occurences of Occurences",main = "Plot for storj token")



fit.exp.result <- fitdist(result\_new$Number, 'exp')  
fit.gamma.result <- fitdist(result\_new$Number, 'gamma',lower = c(0, 0), start = list(scale = 1, shape = 1))  
fit.geometric.result <- fitdist(result\_new$Number, 'geom')  
fit.log.result <- fitdist(result\_new$Number, 'logis')  
fit.lnorm.result <- fitdist(result\_new$Number, 'lnorm')  
fit.nbinom.result <- fitdist(result\_new$Number, 'nbinom')  
fit.norm.result <- fitdist(result\_new$Number, 'norm')  
fit.pois.result <- fitdist(result\_new$Number, 'pois')  
fit.unif.result <- fitdist(result\_new$Number, 'unif')  
fit.weibull.result <- fitdist(result\_new$Number, 'weibull')  
gofstat(list(fit.weibull.result, fit.gamma.result, fit.lnorm.result, fit.exp.result, fit.log.result, fit.pois.result))

## Goodness-of-fit statistics  
## 1-mle-weibull 2-mle-gamma 3-mle-lnorm  
## Kolmogorov-Smirnov statistic 0.08070105 0.09181971 0.06026925  
## Cramer-von Mises statistic 0.15736680 0.19431060 0.10223441  
## Anderson-Darling statistic 0.93148358 1.07418524 0.63858344  
## 4-mle-exp 5-mle-logis 6-mle-pois  
## Kolmogorov-Smirnov statistic 0.09605241 0.1917018 0.625439  
## Cramer-von Mises statistic 0.21412425 0.8888001 12.701993  
## Anderson-Darling statistic 1.15387288 7.1571758 Inf  
##   
## Goodness-of-fit criteria  
## 1-mle-weibull 2-mle-gamma 3-mle-lnorm  
## Akaike's Information Criterion 1429.939 1430.627 1430.977  
## Bayesian Information Criterion 1435.596 1436.283 1436.633  
## 4-mle-exp 5-mle-logis 6-mle-pois  
## Akaike's Information Criterion 1428.733 1540.632 14839.89  
## Bayesian Information Criterion 1431.561 1546.289 14842.72

plot(fit.lnorm.result)

 Question 2:

names(filteredstorj) <- c('fromID', 'toID', 'TimeStamp', 'TokenAmount')  
filteredstorj

## # A tibble: 402,453 x 4  
## fromID toID TimeStamp TokenAmount  
## <dbl> <dbl> <dbl> <dbl>  
## 1 6103619 5607448 1524611290 230769000000  
## 2 17 930403 1524611565 98700000000  
## 3 930403 5 1524612137 98700000000  
## 4 148332 1757147 1524612282 80312639339  
## 5 17 573 1524612421 212877929273  
## 6 573 5 1524612737 212877929273  
## 7 17 87160 1524613395 99700000000  
## 8 5 39994 1524613542 20341300000  
## 9 82 46 1524614390 206106470000  
## 10 171 573 1524614582 202600000000  
## # ... with 402,443 more rows

filteredstorj$TokenAmount<-(filteredstorj$TokenAmount)/decimals  
Time<-as.Date(as.POSIXct(filteredstorj$TimeStamp, origin="1970-01-01"))  
filteredstorj$TimeStamp<-Time  
filteredstorj

## # A tibble: 402,453 x 4  
## fromID toID TimeStamp TokenAmount  
## <dbl> <dbl> <date> <dbl>  
## 1 6103619 5607448 2018-04-24 2308.  
## 2 17 930403 2018-04-24 987   
## 3 930403 5 2018-04-24 987   
## 4 148332 1757147 2018-04-24 803.  
## 5 17 573 2018-04-24 2129.  
## 6 573 5 2018-04-24 2129.  
## 7 17 87160 2018-04-24 997   
## 8 5 39994 2018-04-24 203.  
## 9 82 46 2018-04-24 2061.  
## 10 171 573 2018-04-25 2026   
## # ... with 402,443 more rows

storjTokenData <- read\_delim('storj', delim = "\t", col\_names = T)

## Parsed with column specification:  
## cols(  
## Date = col\_character(),  
## Open = col\_double(),  
## High = col\_double(),  
## Low = col\_double(),  
## Close = col\_double(),  
## Volume = col\_number(),  
## `Market Cap` = col\_character()  
## )

storjTokenData

## # A tibble: 379 x 7  
## Date Open High Low Close Volume `Market Cap`  
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>   
## 1 07/15/2018 0.415 0.432 0.415 0.429 985751 56,386,300   
## 2 07/14/2018 0.421 0.430 0.415 0.416 834294 57,181,600   
## 3 07/13/2018 0.414 0.428 0.414 0.421 2140660 56,243,600   
## 4 07/12/2018 0.443 0.443 0.408 0.415 1080950 60,089,800   
## 5 07/11/2018 0.439 0.448 0.432 0.442 1159100 59,554,700   
## 6 07/10/2018 0.485 0.486 0.437 0.439 1161000 65,911,000   
## 7 07/09/2018 0.492 0.500 0.480 0.486 976432 66,835,800   
## 8 07/08/2018 0.513 0.514 0.491 0.492 924071 69,697,400   
## 9 07/07/2018 0.520 0.520 0.485 0.514 1440160 70,652,300   
## 10 07/06/2018 0.501 0.543 0.475 0.520 2973900 68,046,800   
## # ... with 369 more rows

names(storjTokenData)<-c("TimeStamp","Open","High", "Low", "Close","Volume","MarketCap")  
storjTokenData$TimeStamp<-as.Date(storjTokenData$TimeStamp,"%m/%d/%Y")  
storjTokenData$MarketCap <- as.numeric(gsub(",","",storjTokenData$MarketCap))

## Warning: NAs introduced by coercion

storjTokenData$MarketCap<-as.double(storjTokenData$MarketCap)  
storjTokenData

## # A tibble: 379 x 7  
## TimeStamp Open High Low Close Volume MarketCap  
## <date> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 2018-07-15 0.415 0.432 0.415 0.429 985751 56386300  
## 2 2018-07-14 0.421 0.430 0.415 0.416 834294 57181600  
## 3 2018-07-13 0.414 0.428 0.414 0.421 2140660 56243600  
## 4 2018-07-12 0.443 0.443 0.408 0.415 1080950 60089800  
## 5 2018-07-11 0.439 0.448 0.432 0.442 1159100 59554700  
## 6 2018-07-10 0.485 0.486 0.437 0.439 1161000 65911000  
## 7 2018-07-09 0.492 0.500 0.480 0.486 976432 66835800  
## 8 2018-07-08 0.513 0.514 0.491 0.492 924071 69697400  
## 9 2018-07-07 0.520 0.520 0.485 0.514 1440160 70652300  
## 10 2018-07-06 0.501 0.543 0.475 0.520 2973900 68046800  
## # ... with 369 more rows

library(plyr)

## -------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.  
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:  
## library(plyr); library(dplyr)

## -------------------------------------------------------------------------

##   
## Attaching package: 'plyr'

## The following objects are masked from 'package:dplyr':  
##   
## arrange, count, desc, failwith, id, mutate, rename, summarise,  
## summarize

filteredstorj <- filteredstorj[order(-filteredstorj$TokenAmount),]  
filteredstorj

## # A tibble: 402,453 x 4  
## fromID toID TimeStamp TokenAmount  
## <dbl> <dbl> <date> <dbl>  
## 1 63217 4631520 2018-01-29 1.84e19  
## 2 2612453 6289703 2017-11-03 4.00e16  
## 3 2612453 6289703 2017-11-03 4.00e16  
## 4 6248575 6248576 2018-02-27 1.98e13  
## 5 2612453 6289703 2017-11-03 4.00e10  
## 6 2612453 6289703 2017-11-03 4.00e10  
## 7 325265 104502 2017-11-23 2.00e10  
## 8 325265 104502 2017-11-23 2.00e10  
## 9 325265 104502 2017-11-23 2.00e10  
## 10 325265 104502 2017-11-23 2.00e10  
## # ... with 402,443 more rows

top\_buyers<-head(filteredstorj, 25)  
top\_buyers

## # A tibble: 25 x 4  
## fromID toID TimeStamp TokenAmount  
## <dbl> <dbl> <date> <dbl>  
## 1 63217 4631520 2018-01-29 1.84e19  
## 2 2612453 6289703 2017-11-03 4.00e16  
## 3 2612453 6289703 2017-11-03 4.00e16  
## 4 6248575 6248576 2018-02-27 1.98e13  
## 5 2612453 6289703 2017-11-03 4.00e10  
## 6 2612453 6289703 2017-11-03 4.00e10  
## 7 325265 104502 2017-11-23 2.00e10  
## 8 325265 104502 2017-11-23 2.00e10  
## 9 325265 104502 2017-11-23 2.00e10  
## 10 325265 104502 2017-11-23 2.00e10  
## # ... with 15 more rows

storjTokenData

## # A tibble: 379 x 7  
## TimeStamp Open High Low Close Volume MarketCap  
## <date> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 2018-07-15 0.415 0.432 0.415 0.429 985751 56386300  
## 2 2018-07-14 0.421 0.430 0.415 0.416 834294 57181600  
## 3 2018-07-13 0.414 0.428 0.414 0.421 2140660 56243600  
## 4 2018-07-12 0.443 0.443 0.408 0.415 1080950 60089800  
## 5 2018-07-11 0.439 0.448 0.432 0.442 1159100 59554700  
## 6 2018-07-10 0.485 0.486 0.437 0.439 1161000 65911000  
## 7 2018-07-09 0.492 0.500 0.480 0.486 976432 66835800  
## 8 2018-07-08 0.513 0.514 0.491 0.492 924071 69697400  
## 9 2018-07-07 0.520 0.520 0.485 0.514 1440160 70652300  
## 10 2018-07-06 0.501 0.543 0.475 0.520 2973900 68046800  
## # ... with 369 more rows

TopBuyerData<-join(top\_buyers, storjTokenData)

## Joining by: TimeStamp

TopBuyerData

## fromID toID TimeStamp TokenAmount Open High Low  
## 1 63217 4631520 2018-01-29 1.840e+19 1.660000 1.660000 1.490000  
## 2 2612453 6289703 2017-11-03 4.000e+16 0.549204 0.550289 0.433011  
## 3 2612453 6289703 2017-11-03 4.000e+16 0.549204 0.550289 0.433011  
## 4 6248575 6248576 2018-02-27 1.981e+13 0.945856 0.956132 0.919801  
## 5 2612453 6289703 2017-11-03 4.000e+10 0.549204 0.550289 0.433011  
## 6 2612453 6289703 2017-11-03 4.000e+10 0.549204 0.550289 0.433011  
## 7 325265 104502 2017-11-23 2.000e+10 0.738191 0.742714 0.684710  
## 8 325265 104502 2017-11-23 2.000e+10 0.738191 0.742714 0.684710  
## 9 325265 104502 2017-11-23 2.000e+10 0.738191 0.742714 0.684710  
## 10 325265 104502 2017-11-23 2.000e+10 0.738191 0.742714 0.684710  
## 11 325265 104502 2017-11-23 2.000e+10 0.738191 0.742714 0.684710  
## 12 325265 104502 2017-11-23 2.000e+10 0.738191 0.742714 0.684710  
## 13 325265 104502 2017-11-23 2.000e+10 0.738191 0.742714 0.684710  
## 14 325265 104502 2017-11-23 2.000e+10 0.738191 0.742714 0.684710  
## 15 325265 104502 2017-11-23 2.000e+10 0.738191 0.742714 0.684710  
## 16 325265 104502 2017-11-23 2.000e+10 0.738191 0.742714 0.684710  
## 17 325265 104502 2017-11-23 2.000e+10 0.738191 0.742714 0.684710  
## 18 325265 104502 2017-11-23 2.000e+10 0.738191 0.742714 0.684710  
## 19 325265 104502 2017-11-23 2.000e+10 0.738191 0.742714 0.684710  
## 20 325265 104502 2017-11-23 2.000e+10 0.738191 0.742714 0.684710  
## 21 325265 104502 2017-11-23 2.000e+10 0.738191 0.742714 0.684710  
## 22 325265 104502 2017-11-23 2.000e+10 0.738191 0.742714 0.684710  
## 23 325265 104502 2017-11-23 2.000e+10 0.738191 0.742714 0.684710  
## 24 325265 104502 2017-11-23 2.000e+10 0.738191 0.742714 0.684710  
## 25 325265 104502 2017-11-23 2.000e+10 0.738191 0.742714 0.684710  
## Close Volume MarketCap  
## 1 1.500000 7986430 221932000  
## 2 0.515272 6272350 55679400  
## 3 0.515272 6272350 55679400  
## 4 0.940979 3356560 126201000  
## 5 0.515272 6272350 55679400  
## 6 0.515272 6272350 55679400  
## 7 0.703325 6182140 77555200  
## 8 0.703325 6182140 77555200  
## 9 0.703325 6182140 77555200  
## 10 0.703325 6182140 77555200  
## 11 0.703325 6182140 77555200  
## 12 0.703325 6182140 77555200  
## 13 0.703325 6182140 77555200  
## 14 0.703325 6182140 77555200  
## 15 0.703325 6182140 77555200  
## 16 0.703325 6182140 77555200  
## 17 0.703325 6182140 77555200  
## 18 0.703325 6182140 77555200  
## 19 0.703325 6182140 77555200  
## 20 0.703325 6182140 77555200  
## 21 0.703325 6182140 77555200  
## 22 0.703325 6182140 77555200  
## 23 0.703325 6182140 77555200  
## 24 0.703325 6182140 77555200  
## 25 0.703325 6182140 77555200

#TopBuyerData$percentage<-(TopBuyerData$TokenAmount/TopBuyerData$MarketCap)\*100  
#TopBuyerData<- subset(TopBuyerData,percentage<100)  
TopUniqueBuyers<-unique(TopBuyerData)  
TopUniqueBuyers

## fromID toID TimeStamp TokenAmount Open High Low  
## 1 63217 4631520 2018-01-29 1.840e+19 1.660000 1.660000 1.490000  
## 2 2612453 6289703 2017-11-03 4.000e+16 0.549204 0.550289 0.433011  
## 4 6248575 6248576 2018-02-27 1.981e+13 0.945856 0.956132 0.919801  
## 5 2612453 6289703 2017-11-03 4.000e+10 0.549204 0.550289 0.433011  
## 7 325265 104502 2017-11-23 2.000e+10 0.738191 0.742714 0.684710  
## Close Volume MarketCap  
## 1 1.500000 7986430 221932000  
## 2 0.515272 6272350 55679400  
## 4 0.940979 3356560 126201000  
## 5 0.515272 6272350 55679400  
## 7 0.703325 6182140 77555200

message("K=",nrow(TopUniqueBuyers))

## K=5

cor.test(TopUniqueBuyers$Open,TopUniqueBuyers$Volume)

##   
## Pearson's product-moment correlation  
##   
## data: TopUniqueBuyers$Open and TopUniqueBuyers$Volume  
## t = 0.71144, df = 3, p-value = 0.5281  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## -0.7556123 0.9453261  
## sample estimates:  
## cor   
## 0.3799449

cor.test(TopUniqueBuyers$Open,TopUniqueBuyers$MarketCap)

##   
## Pearson's product-moment correlation  
##   
## data: TopUniqueBuyers$Open and TopUniqueBuyers$MarketCap  
## t = 19.269, df = 3, p-value = 0.0003053  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## 0.9376755 0.9997484  
## sample estimates:  
## cor   
## 0.9959844

cor.test(TopUniqueBuyers$Open,TopUniqueBuyers$TokenAmount)

##   
## Pearson's product-moment correlation  
##   
## data: TopUniqueBuyers$Open and TopUniqueBuyers$TokenAmount  
## t = 4.5459, df = 3, p-value = 0.01994  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## 0.2973654 0.9957712  
## sample estimates:  
## cor   
## 0.9344688

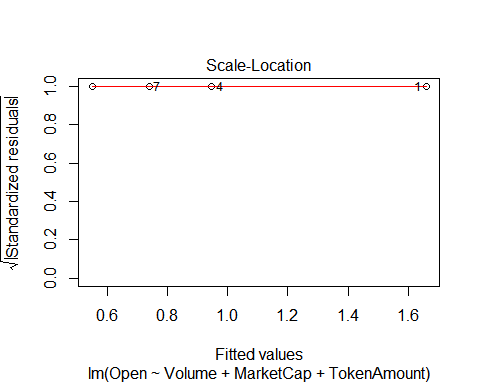
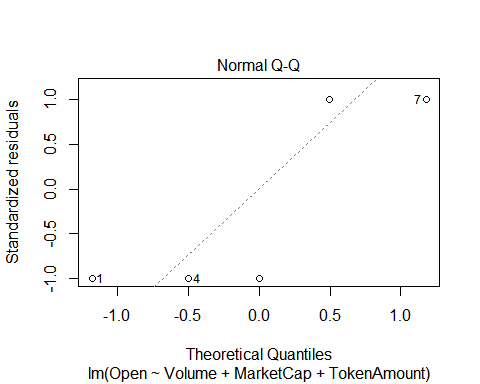
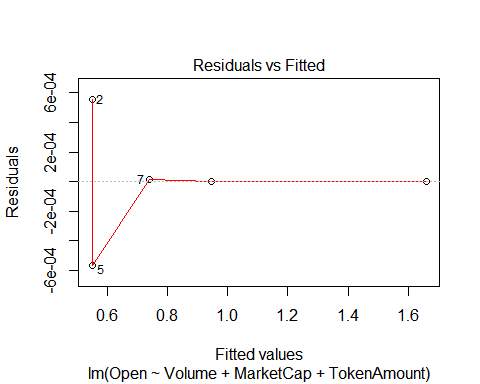
linearModel <- lm(formula=Open ~ Volume+MarketCap+TokenAmount, data=TopUniqueBuyers)   
linearModel

##   
## Call:  
## lm(formula = Open ~ Volume + MarketCap + TokenAmount, data = TopUniqueBuyers)  
##   
## Coefficients:  
## (Intercept) Volume MarketCap TokenAmount   
## -4.531e-01 8.049e-08 8.945e-09 -2.798e-20

summary(linearModel)

##   
## Call:  
## lm(formula = Open ~ Volume + MarketCap + TokenAmount, data = TopUniqueBuyers)  
##   
## Residuals:  
## 1 2 4 5 7   
## -1.205e-06 5.544e-04 -1.102e-06 -5.648e-04 1.271e-05   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -4.531e-01 9.854e-03 -45.99 0.01384 \*   
## Volume 8.049e-08 1.114e-09 72.26 0.00881 \*\*  
## MarketCap 8.945e-09 4.822e-11 185.51 0.00343 \*\*  
## TokenAmount -2.798e-20 5.224e-22 -53.55 0.01189 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.0007915 on 1 degrees of freedom  
## Multiple R-squared: 1, Adjusted R-squared: 1   
## F-statistic: 4.53e+05 on 3 and 1 DF, p-value: 0.001092

plot(linearModel)



## Warning in sqrt(crit \* p \* (1 - hh)/hh): NaNs produced

## Warning in sqrt(crit \* p \* (1 - hh)/hh): NaNs produced

