tenxpayToken

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

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

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

print(tenxpay)

## # A tibble: 329,737 x 4  
## X1 X2 X3 X4  
## <dbl> <dbl> <dbl> <dbl>  
## 1 560 1452 1524611450 1.73e20  
## 2 2011173 2011174 1524611865 4.56e20  
## 3 75989 1822217 1524612292 5.80e20  
## 4 40002 6382858 1524612655 4.48e20  
## 5 17 2029263 1524612851 5.00e21  
## 6 222770 4848204 1524612957 3.28e20  
## 7 17 1148 1524613473 4.81e21  
## 8 76011 76012 1524613896 5.20e19  
## 9 6382859 104531 1524614072 6.08e19  
## 10 187 3169275 1524614361 5.55e19  
## # ... with 329,727 more rows

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

## # A tibble: 329,737 x 4  
## fromID toID unixTime tokenAmount  
## <dbl> <dbl> <dbl> <dbl>  
## 1 560 1452 1524611450 1.73e20  
## 2 2011173 2011174 1524611865 4.56e20  
## 3 75989 1822217 1524612292 5.80e20  
## 4 40002 6382858 1524612655 4.48e20  
## 5 17 2029263 1524612851 5.00e21  
## 6 222770 4848204 1524612957 3.28e20  
## 7 17 1148 1524613473 4.81e21  
## 8 76011 76012 1524613896 5.20e19  
## 9 6382859 104531 1524614072 6.08e19  
## 10 187 3169275 1524614361 5.55e19  
## # ... with 329,727 more rows

decimals<-10^18  
supply<- 205218255.948577763364408207  
totalSupply<- decimals \* supply  
print(totalSupply)

## [1] 2.052183e+26

filteredtenxpay <- filter(tenxpay,tokenAmount < totalSupply)  
print(filteredtenxpay)

## # A tibble: 329,736 x 4  
## fromID toID unixTime tokenAmount  
## <dbl> <dbl> <dbl> <dbl>  
## 1 560 1452 1524611450 1.73e20  
## 2 2011173 2011174 1524611865 4.56e20  
## 3 75989 1822217 1524612292 5.80e20  
## 4 40002 6382858 1524612655 4.48e20  
## 5 17 2029263 1524612851 5.00e21  
## 6 222770 4848204 1524612957 3.28e20  
## 7 17 1148 1524613473 4.81e21  
## 8 76011 76012 1524613896 5.20e19  
## 9 6382859 104531 1524614072 6.08e19  
## 10 187 3169275 1524614361 5.55e19  
## # ... with 329,726 more rows

filteredtenxpay <- filter(tenxpay,fromID != toID)  
print(filteredtenxpay)

## # A tibble: 319,656 x 4  
## fromID toID unixTime tokenAmount  
## <dbl> <dbl> <dbl> <dbl>  
## 1 560 1452 1524611450 1.73e20  
## 2 2011173 2011174 1524611865 4.56e20  
## 3 75989 1822217 1524612292 5.80e20  
## 4 40002 6382858 1524612655 4.48e20  
## 5 17 2029263 1524612851 5.00e21  
## 6 222770 4848204 1524612957 3.28e20  
## 7 17 1148 1524613473 4.81e21  
## 8 76011 76012 1524613896 5.20e19  
## 9 6382859 104531 1524614072 6.08e19  
## 10 187 3169275 1524614361 5.55e19  
## # ... with 319,646 more rows

NoOfOutliers <- count(tenxpay)-count(filteredtenxpay);  
print(NoOfOutliers)

## n  
## 1 10081

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

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

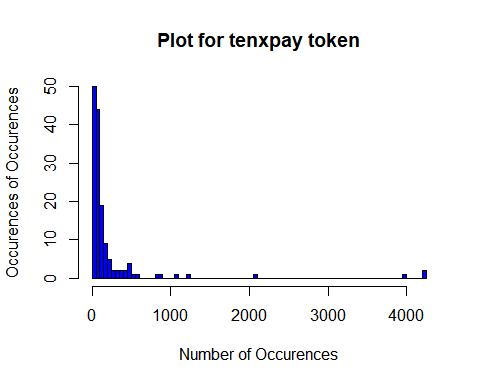
sum(result$Occurences)

## [1] 319656

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 131029  
## 2 2 27246  
## 3 3 8532  
## 4 4 3875  
## 5 5 2066  
## 6 6 1258

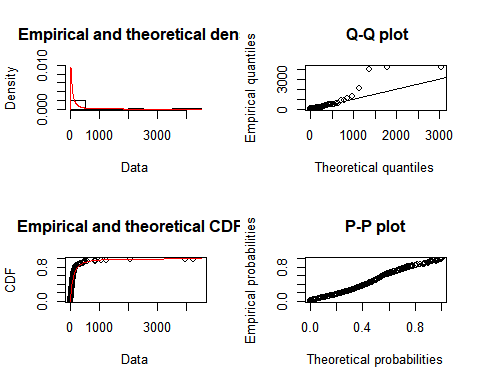
hist(result\_new$Number, breaks = 100, col = c("blue"), xlab = "Number of Occurences", ylab = "Occurences of Occurences",main = "Plot for tenxpay token")



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 0.1361148 0.1980425 0.07326387  
## Cramer-von Mises statistic 0.9125615 1.7294835 0.23198072  
## Anderson-Darling statistic 5.3549628 8.9203993 1.29183124  
## 4-mle-exp 5-mle-logis 6-mle-pois  
## Kolmogorov-Smirnov statistic 0.2872441 0.3147257 0.8109528  
## Cramer-von Mises statistic 4.6133664 3.5188683 28.6724683  
## Anderson-Darling statistic 22.0856395 20.1473062 Inf  
##   
## Goodness-of-fit criteria  
## 1-mle-weibull 2-mle-gamma 3-mle-lnorm  
## Akaike's Information Criterion 1850.388 1879.366 1809.956  
## Bayesian Information Criterion 1856.396 1885.374 1815.964  
## 4-mle-exp 5-mle-logis 6-mle-pois  
## Akaike's Information Criterion 1916.776 2160.562 85409.95  
## Bayesian Information Criterion 1919.780 2166.570 85412.95

plot(fit.lnorm.result)

 Question 2:

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

## # A tibble: 319,656 x 4  
## fromID toID TimeStamp TokenAmount  
## <dbl> <dbl> <dbl> <dbl>  
## 1 560 1452 1524611450 1.73e20  
## 2 2011173 2011174 1524611865 4.56e20  
## 3 75989 1822217 1524612292 5.80e20  
## 4 40002 6382858 1524612655 4.48e20  
## 5 17 2029263 1524612851 5.00e21  
## 6 222770 4848204 1524612957 3.28e20  
## 7 17 1148 1524613473 4.81e21  
## 8 76011 76012 1524613896 5.20e19  
## 9 6382859 104531 1524614072 6.08e19  
## 10 187 3169275 1524614361 5.55e19  
## # ... with 319,646 more rows

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

## # A tibble: 319,656 x 4  
## fromID toID TimeStamp TokenAmount  
## <dbl> <dbl> <date> <dbl>  
## 1 560 1452 2018-04-24 173.   
## 2 2011173 2011174 2018-04-24 456.   
## 3 75989 1822217 2018-04-24 580.   
## 4 40002 6382858 2018-04-24 448.   
## 5 17 2029263 2018-04-24 4998.   
## 6 222770 4848204 2018-04-24 328.   
## 7 17 1148 2018-04-24 4812   
## 8 76011 76012 2018-04-24 52   
## 9 6382859 104531 2018-04-24 60.8  
## 10 187 3169275 2018-04-24 55.5  
## # ... with 319,646 more rows

tenxpayTokenData <- read\_delim('tenxpay', 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()  
## )

tenxpayTokenData

## # A tibble: 384 x 7  
## Date Open High Low Close Volume `Market Cap`  
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>   
## 1 07/15/2018 0.746 0.748 0.722 0.730 4856290 81,509,100   
## 2 07/14/2018 0.698 0.801 0.696 0.746 7753930 76,261,300   
## 3 07/13/2018 0.600 0.845 0.600 0.702 12405200 65,521,200   
## 4 07/12/2018 0.599 0.617 0.569 0.588 4906510 65,393,900   
## 5 07/11/2018 0.593 0.656 0.568 0.600 5622960 64,739,100   
## 6 07/10/2018 0.629 0.666 0.592 0.592 4754660 68,694,000   
## 7 07/09/2018 0.644 0.646 0.616 0.629 3774090 70,287,200   
## 8 07/08/2018 0.655 0.667 0.642 0.643 3202230 71,505,900   
## 9 07/07/2018 0.631 0.655 0.626 0.654 3336940 68,905,100   
## 10 07/06/2018 0.685 0.685 0.614 0.634 4293480 74,808,700   
## # ... with 374 more rows

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

## Warning: NAs introduced by coercion

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

## # A tibble: 384 x 7  
## TimeStamp Open High Low Close Volume MarketCap  
## <date> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 2018-07-15 0.746 0.748 0.722 0.730 4856290 81509100  
## 2 2018-07-14 0.698 0.801 0.696 0.746 7753930 76261300  
## 3 2018-07-13 0.600 0.845 0.600 0.702 12405200 65521200  
## 4 2018-07-12 0.599 0.617 0.569 0.588 4906510 65393900  
## 5 2018-07-11 0.593 0.656 0.568 0.600 5622960 64739100  
## 6 2018-07-10 0.629 0.666 0.592 0.592 4754660 68694000  
## 7 2018-07-09 0.644 0.646 0.616 0.629 3774090 70287200  
## 8 2018-07-08 0.655 0.667 0.642 0.643 3202230 71505900  
## 9 2018-07-07 0.631 0.655 0.626 0.654 3336940 68905100  
## 10 2018-07-06 0.685 0.685 0.614 0.634 4293480 74808700  
## # ... with 374 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

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

## # A tibble: 319,656 x 4  
## fromID toID TimeStamp TokenAmount  
## <dbl> <dbl> <date> <dbl>  
## 1 74302 250834 2017-07-07 7108079   
## 2 1722175 1839267 2017-09-03 5000000   
## 3 74302 2476250 2017-07-08 3402000   
## 4 2476250 297243 2017-07-27 3359999   
## 5 326195 249800 2017-07-09 3027229.  
## 6 250834 1940497 2017-07-08 2500000   
## 7 6431626 2335406 2017-07-18 2130877.  
## 8 298944 296381 2017-08-10 2000000   
## 9 354421 6437070 2017-07-07 2000000   
## 10 74302 183229 2017-07-08 1988000   
## # ... with 319,646 more rows

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

## # A tibble: 25 x 4  
## fromID toID TimeStamp TokenAmount  
## <dbl> <dbl> <date> <dbl>  
## 1 74302 250834 2017-07-07 7108079   
## 2 1722175 1839267 2017-09-03 5000000   
## 3 74302 2476250 2017-07-08 3402000   
## 4 2476250 297243 2017-07-27 3359999   
## 5 326195 249800 2017-07-09 3027229.  
## 6 250834 1940497 2017-07-08 2500000   
## 7 6431626 2335406 2017-07-18 2130877.  
## 8 298944 296381 2017-08-10 2000000   
## 9 354421 6437070 2017-07-07 2000000   
## 10 74302 183229 2017-07-08 1988000   
## # ... with 15 more rows

tenxpayTokenData

## # A tibble: 384 x 7  
## TimeStamp Open High Low Close Volume MarketCap  
## <date> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 2018-07-15 0.746 0.748 0.722 0.730 4856290 81509100  
## 2 2018-07-14 0.698 0.801 0.696 0.746 7753930 76261300  
## 3 2018-07-13 0.600 0.845 0.600 0.702 12405200 65521200  
## 4 2018-07-12 0.599 0.617 0.569 0.588 4906510 65393900  
## 5 2018-07-11 0.593 0.656 0.568 0.600 5622960 64739100  
## 6 2018-07-10 0.629 0.666 0.592 0.592 4754660 68694000  
## 7 2018-07-09 0.644 0.646 0.616 0.629 3774090 70287200  
## 8 2018-07-08 0.655 0.667 0.642 0.643 3202230 71505900  
## 9 2018-07-07 0.631 0.655 0.626 0.654 3336940 68905100  
## 10 2018-07-06 0.685 0.685 0.614 0.634 4293480 74808700  
## # ... with 374 more rows

TopBuyerData<-join(top\_buyers, tenxpayTokenData)

## Joining by: TimeStamp

TopBuyerData

## fromID toID TimeStamp TokenAmount Open High Low  
## 1 74302 250834 2017-07-07 7108079 73.060000 73.060000 1.050000  
## 2 1722175 1839267 2017-09-03 5000000 3.620000 3.640000 3.270000  
## 3 74302 2476250 2017-07-08 3402000 1.110000 1.220000 1.080000  
## 4 2476250 297243 2017-07-27 3359999 0.974673 0.983180 0.927986  
## 5 326195 249800 2017-07-09 3027229 1.110000 1.110000 0.871622  
## 6 250834 1940497 2017-07-08 2500000 1.110000 1.220000 1.080000  
## 7 6431626 2335406 2017-07-18 2130877 0.709172 0.764586 0.630566  
## 8 298944 296381 2017-08-10 2000000 2.140000 3.110000 2.090000  
## 9 354421 6437070 2017-07-07 2000000 73.060000 73.060000 1.050000  
## 10 74302 183229 2017-07-08 1988000 1.110000 1.220000 1.080000  
## 11 1839267 1722175 2017-07-16 1700000 0.598683 0.611108 0.478733  
## 12 6437092 321994 2017-07-07 1669000 73.060000 73.060000 1.050000  
## 13 1722175 1839267 2017-09-04 1500000 3.530000 3.530000 2.330000  
## 14 250834 1940497 2017-07-07 1500000 73.060000 73.060000 1.050000  
## 15 6431626 6438343 2017-07-09 1500000 1.110000 1.110000 0.871622  
## 16 354421 6437070 2017-07-07 1243342 73.060000 73.060000 1.050000  
## 17 1722175 1839267 2017-08-11 1200000 2.870000 4.950000 2.660000  
## 18 249800 183003 2017-07-18 1154255 0.709172 0.764586 0.630566  
## 19 962843 1940570 2018-03-26 1099999 1.330000 1.350000 1.160000  
## 20 962843 2335409 2018-03-26 1099999 1.330000 1.350000 1.160000  
## 21 6438343 6433429 2017-07-09 1075190 1.110000 1.110000 0.871622  
## 22 298944 296381 2017-09-04 1000000 3.530000 3.530000 2.330000  
## 23 298944 296381 2017-08-11 1000000 2.870000 4.950000 2.660000  
## 24 298944 296381 2017-08-12 1000000 4.510000 6.210000 4.230000  
## 25 298944 296381 2017-08-14 1000000 5.100000 5.370000 4.310000  
## Close Volume MarketCap  
## 1 1.100000 17115200 NA  
## 2 3.530000 13885000 378656000  
## 3 1.100000 9172650 NA  
## 4 0.978920 1375970 102011000  
## 5 0.901177 5116090 116174000  
## 6 1.100000 9172650 NA  
## 7 0.699843 1299010 74222900  
## 8 2.810000 36245500 223846000  
## 9 1.100000 17115200 NA  
## 10 1.100000 9172650 NA  
## 11 0.569964 762642 62658900  
## 12 1.100000 17115200 NA  
## 13 2.750000 25177900 369749000  
## 14 1.100000 17115200 NA  
## 15 0.901177 5116090 116174000  
## 16 1.100000 17115200 NA  
## 17 4.520000 72243200 300417000  
## 18 0.699843 1299010 74222900  
## 19 1.230000 5138750 139482000  
## 20 1.230000 5138750 139482000  
## 21 0.901177 5116090 116174000  
## 22 2.750000 25177900 369749000  
## 23 4.520000 72243200 300417000  
## 24 4.670000 144769000 471783000  
## 25 4.580000 40308400 533626000

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

## fromID toID TimeStamp TokenAmount Open High Low  
## 1 74302 250834 2017-07-07 7108079 73.060000 73.060000 1.050000  
## 2 1722175 1839267 2017-09-03 5000000 3.620000 3.640000 3.270000  
## 3 74302 2476250 2017-07-08 3402000 1.110000 1.220000 1.080000  
## 4 2476250 297243 2017-07-27 3359999 0.974673 0.983180 0.927986  
## 5 326195 249800 2017-07-09 3027229 1.110000 1.110000 0.871622  
## 6 250834 1940497 2017-07-08 2500000 1.110000 1.220000 1.080000  
## 7 6431626 2335406 2017-07-18 2130877 0.709172 0.764586 0.630566  
## 8 298944 296381 2017-08-10 2000000 2.140000 3.110000 2.090000  
## 9 354421 6437070 2017-07-07 2000000 73.060000 73.060000 1.050000  
## 10 74302 183229 2017-07-08 1988000 1.110000 1.220000 1.080000  
## 11 1839267 1722175 2017-07-16 1700000 0.598683 0.611108 0.478733  
## 12 6437092 321994 2017-07-07 1669000 73.060000 73.060000 1.050000  
## 13 1722175 1839267 2017-09-04 1500000 3.530000 3.530000 2.330000  
## 14 250834 1940497 2017-07-07 1500000 73.060000 73.060000 1.050000  
## 15 6431626 6438343 2017-07-09 1500000 1.110000 1.110000 0.871622  
## 16 354421 6437070 2017-07-07 1243342 73.060000 73.060000 1.050000  
## 17 1722175 1839267 2017-08-11 1200000 2.870000 4.950000 2.660000  
## 18 249800 183003 2017-07-18 1154255 0.709172 0.764586 0.630566  
## 19 962843 1940570 2018-03-26 1099999 1.330000 1.350000 1.160000  
## 20 962843 2335409 2018-03-26 1099999 1.330000 1.350000 1.160000  
## 21 6438343 6433429 2017-07-09 1075190 1.110000 1.110000 0.871622  
## 22 298944 296381 2017-09-04 1000000 3.530000 3.530000 2.330000  
## 23 298944 296381 2017-08-11 1000000 2.870000 4.950000 2.660000  
## 24 298944 296381 2017-08-12 1000000 4.510000 6.210000 4.230000  
## 25 298944 296381 2017-08-14 1000000 5.100000 5.370000 4.310000  
## Close Volume MarketCap  
## 1 1.100000 17115200 NA  
## 2 3.530000 13885000 378656000  
## 3 1.100000 9172650 NA  
## 4 0.978920 1375970 102011000  
## 5 0.901177 5116090 116174000  
## 6 1.100000 9172650 NA  
## 7 0.699843 1299010 74222900  
## 8 2.810000 36245500 223846000  
## 9 1.100000 17115200 NA  
## 10 1.100000 9172650 NA  
## 11 0.569964 762642 62658900  
## 12 1.100000 17115200 NA  
## 13 2.750000 25177900 369749000  
## 14 1.100000 17115200 NA  
## 15 0.901177 5116090 116174000  
## 16 1.100000 17115200 NA  
## 17 4.520000 72243200 300417000  
## 18 0.699843 1299010 74222900  
## 19 1.230000 5138750 139482000  
## 20 1.230000 5138750 139482000  
## 21 0.901177 5116090 116174000  
## 22 2.750000 25177900 369749000  
## 23 4.520000 72243200 300417000  
## 24 4.670000 144769000 471783000  
## 25 4.580000 40308400 533626000

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

## K=25

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

##   
## Pearson's product-moment correlation  
##   
## data: TopUniqueBuyers$Open and TopUniqueBuyers$Volume  
## t = -0.30509, df = 23, p-value = 0.763  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## -0.4473956 0.3401765  
## sample estimates:  
## cor   
## -0.06348807

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

##   
## Pearson's product-moment correlation  
##   
## data: TopUniqueBuyers$Open and TopUniqueBuyers$MarketCap  
## t = 3555, df = 15, p-value < 2.2e-16  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## 0.9999983 0.9999998  
## sample estimates:  
## cor   
## 0.9999994

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

##   
## Pearson's product-moment correlation  
##   
## data: TopUniqueBuyers$Open and TopUniqueBuyers$TokenAmount  
## t = 1.1195, df = 23, p-value = 0.2745  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## -0.1843696 0.5711508  
## sample estimates:  
## cor   
## 0.2273218

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

##   
## Call:  
## lm(formula = Open ~ Volume + MarketCap + TokenAmount, data = TopUniqueBuyers)  
##   
## Coefficients:  
## (Intercept) Volume MarketCap TokenAmount   
## -2.147e-03 2.800e-11 9.553e-09 8.575e-10

summary(linearModel)

##   
## Call:  
## lm(formula = Open ~ Volume + MarketCap + TokenAmount, data = TopUniqueBuyers)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.0020164 -0.0007561 0.0002111 0.0009199 0.0024893   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -2.147e-03 8.539e-04 -2.514 0.0259 \*   
## Volume 2.800e-11 1.294e-11 2.164 0.0497 \*   
## MarketCap 9.553e-09 3.111e-12 3070.917 <2e-16 \*\*\*  
## TokenAmount 8.575e-10 3.314e-10 2.587 0.0225 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.001364 on 13 degrees of freedom  
## (8 observations deleted due to missingness)  
## Multiple R-squared: 1, Adjusted R-squared: 1   
## F-statistic: 6.054e+06 on 3 and 13 DF, p-value: < 2.2e-16

plot(linearModel)

