

Tierion

```
1. library(magrittr)
2. library(dplyr)
3. library(ggplot2)
4. library(readr)
5. library(fitdistrplus)
6. library(DAAG)
7. library("ggplot2")
8. library(anytime)
9.
10. tierion <- read_delim('C:/Users/ygaoq/OneDrive/MyDocuments/2019 Spring/Statistics/Project/Blockchain-Tokens-Data-Analytics/networktierionTX.txt', delim = " ", col_names = F)
11. names(tierion) <- c('fromID', 'toID', 'unixTime', 'tokenAmount')
12. decimals <- 10^8
13. supply <- 1 * 10^9
14. tierionFiltered <-
  filter(tierion, tokenAmount < decimals * supply) #filter out all outliers
15.
16. #figure out how many users induced those unnormal transaction
17. tierion_outliers <- filter(tierion, tokenAmount >= decimals * supply)
18. user_outliers <- tierion_outliers %>% group_by(toID) %>% summarise(nn = n()) %>% ungroup
19. number_users_outliers <- nrow(user_outliers)
20. number_users_outliers
21.
22. #get top X buyers data
23. buys <-
  tierionFiltered %>% group_by(toID) %>% summarise(nn = n()) %>% ungroup #change the supply and decimals amount
24. buys_sorted_dec <- buys[order(-buys$nn),]
25. #top 30 active buyers and number of buys
26. top_30_buyers <- buys_sorted_dec %>% head(30)
27. top_30_buyers
28.
29. #####Question 1#####
30.
31. #####group by user pairs#####
32. buys_pairs <-
  tierionFiltered %>% group_by(fromID, toID) %>% summarise(nn = n()) %>% ungroup
33. for (row in 1:nrow(buys_pairs)) {
34.   a <- buys_pairs[row, "fromID"]
35.   b <- buys_pairs[row, "toID"]
36.   for (inner_row in row:nrow(buys_pairs)) {
37.     c <- buys_pairs[inner_row, "fromID"]
38.     d <- buys_pairs[inner_row, "toID"]
39.     if (a == d && b == c) {
40.       buys_pairs[inner_row, "fromID"] <- d
41.       buys_pairs[inner_row, "toID"] <- c
42.     }
43.   }
44. }
45. buys_pairs <-
  tierionFiltered %>% group_by(fromID*toID+fromID+toID) %>% summarise(nn = n()) %>% ungroup
p
```

```

46. buys_pairs<-
  tierionFiltered%>% group_by(fromID, toID) %>% summarise(nn = n()) %>% ungroup
47. buys_pair_sorted_asc<-buys_pairs[order(buys_pairs$n),]
48. buys_pair_less_30<-subset(buys_pair_sorted_asc,n<30)
49.
50. #####find out estimates of paramaters of several distribution based on the buys_pairs d
    ata set#####
51. exp_dis <- fitdist(buys_pair_less_30$n, 'exp')
52. exp_dis
53. gamma_dis <- fitdist(buys_pair_less_30$n, 'gamma')
54. gamma_dis
55. lnorm_dis <- fitdist(buys_pair_less_30$n, 'lnorm')
56. lnorm_dis
57. pois_dis <- fitdist(buys_pair_less_30$n, 'pois')
58. pois_dis
59. weibull_dis <- fitdist(buys_pair_less_30$n, 'weibull')
60. weibull_dis
61.
62. gofstat(list(exp_dis, gamma_dis, lnorm_dis))
63. descdist(buys_sorted_asc$n,boot=1000)
64.
65. #lognorm
66. fit_lnorm <- fitdist(buys_pair_less_30$n,"lnorm")
67. summary(fit_lnorm)
68. plot(fit_lnorm)
69. cdfcomp(fit_lnorm)
70.
71. #exp
72. fit_exp <- fitdist(buys_pair_less_30$n,"exp")
73. summary(fit_exp)
74. plot(fit_exp)
75. cdfcomp(fit_exp)
76.
77. #gamma
78. fit_gamma <- fitdist(buys_pair_less_30$n,"gamma")
79. summary(fit_gamma)
80. plot(fit_gamma)
81. cdfcomp(fit_gamma)
82.
83. #weibull
84. fit_weibull <- fitdist(buys_pair_less_30$n,"weibull")
85. summary(fit_weibull)
86.
87. #normal
88. fit_normal <- fitdist(buys_pair_less_30$n,"norm")
89. summary(fit_normal)
90.
91. #pois
92. #normal
93. fit_pois <- fitdist(buys_pair_less_30$n,"pois")
94. summary(fit_pois)
95.
96. #unif
97. fit_unif <- fitdist(buys_pair_less_30$n,"unif")
98. summary(fit_unif)
99. plot(fit_unif)
100.      cdfcomp(fit_unif)
101.
102. #####draw graph#####
103. all_density <- ggplot(data=buys_pair_less_30) +
104.      geom_histogram(bins=30,aes(x = buys_pair_less_30$n, ..density..)) +

```

```

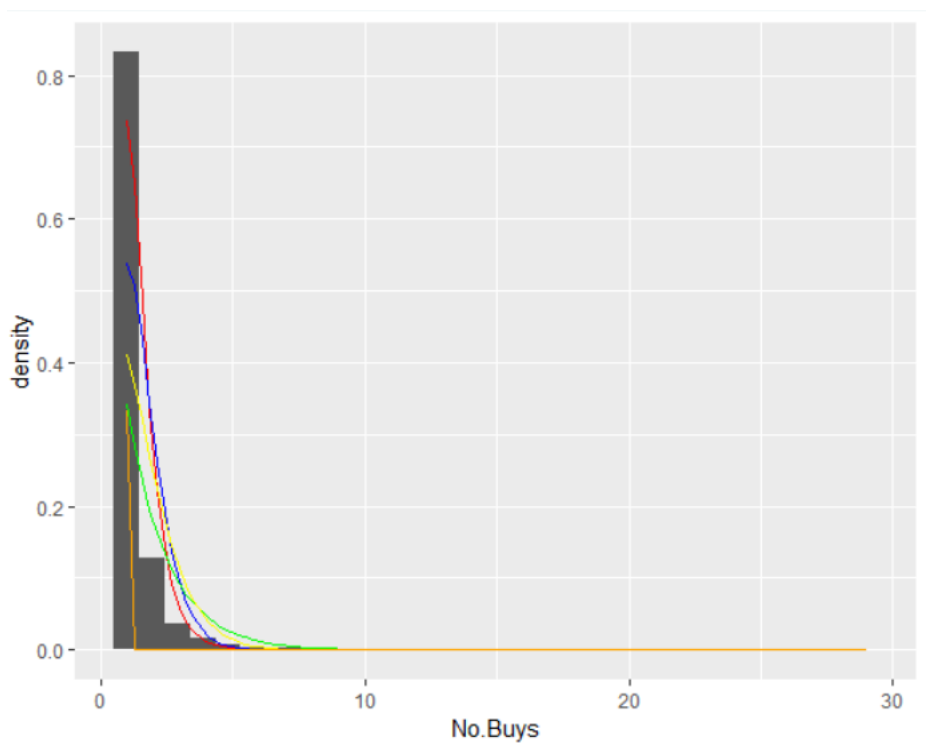
105.      stat_function(fun = dlnorm, args = list(meanlog = 0.2373987, sdlog = 0.4791316
),
106.          colour = "red")+
107.      stat_function(fun = dgamma, args = list(shape = 3.020395, rate=2.000602),
108.          colour = "blue")+
109.      stat_function(fun=dexp, args=list(rate=0.6623558),colour="green")+
110.      stat_function(fun=dweibull, args=list(shape=1.360851, scale=1.678697),colour="
yellow")+
111.      stat_function(fun=dpois, args=list(lambda=1.509763),colour="orange")+ xlab("N
o.Buys")
112.  all_density
113.
114.  #####Question 1#####
115.
116.  #####Question 2#####
117.
118.  tierion_prices <- read_delim("C:/Users/ygaoq/OneDrive/MyDocuments/2019 Spring/St
atistics/Project/Blockchain-Tokens-Data-
Analytics/tierion", delim = "\t", col_names = T) #load token price data
119.  names(tierion_prices) <- make.names(names(tierion_prices))
120.  tierion_prices <- tierion_prices %>% mutate(date = as.Date(Date, format = '%m/%d
/%Y'))
121.
122.  tierion <- read_delim('C:/Users/ygaoq/OneDrive/MyDocuments/2019 Spring/Statistic
s/Project/Blockchain-Tokens-Data-
Analytics/networkktierionTX.txt', delim = " ", col_names = F)
123.  names(tierion) <- c('fromID', 'toID', 'unixTime', 'tokenAmount')
124.  decimals <- 10^8
125.  supply <- 1 * 10^9
126.  tierion_filtered <-filter(tierion,tokenAmount < decimals * supply)
127.  ## convert data type of unixTime
128.  tierion_filtered <- tierion_filtered %>%
129.      mutate(date = anydate(unixTime))
130.  names(tierion_filtered) <- c('fromID', 'toID', 'unixTime', 'tokenAmount', 'date'
)
131.
132.  ## merge the prices and edge
133.  tierion_merged<-
merge(x = tierion_prices, y = tierion_filtered, by = "date", all.x = TRUE)
134.
135.  #####Determin K#####
136.  top_30_buyers<-buys_sorted_dec%>%head(30)
137.
138.  top_K<-c(1:30)
139.  count <- 1
140.  for (val in top_K) {
141.      top_K_buyers<-buys_sorted_dec%>%head(val)
142.      filter_K_tierion_merged<-filter(tierion_merged,toID %in% top_K_buyers$toID)
143.      filter_K_tierion_merged=transform(filter_K_tierion_merged,average_price= (Open
+Close)/2)
144.      filter_K_tierion_merged$num_Date <- as.numeric(as.POSIXct(filter_K_tierion_me
rged$date))
145.      filered<-
filter_K_tierion_merged%>% group_by(num_Date) %>% summarise(nn = n(),Close=mean(Close),
tokenAmount=sum(tokenAmount),Open=mean(Open))
146.      shift <- function(x, n){
147.          c(x[-(seq(n))], rep(NA, n))
148.      }
149.      filered$new_Close<-shift(filered$Close,1)

```

```

150.     num_rows<-nrow(filered)
151.     filered[-num_rows,]
152.     regression<-lm(filered$new_Close~filered$tokenAmount+filered$n+filered$Open)
153.
154.     setwd("C:/Users/ygaoq/Desktop/Tierion")
155.     yourfilename=paste("W",val,".txt",sep="")
156.     capture.output(summary(regression),append = TRUE,file = "C:/Users/ygaoq/Desktop/
p/Tierion/Final_Result.txt")
157.
158.
159.     summary(regression)
160.     par(mfcol=c(2,2))
161.     setwd("C:/Users/ygaoq/Desktop/Tierion")
162.     yourfilename=paste("A",val,".png",sep="")
163.     png(file=yourfilename)
164.     opar <- par(mfrow=c(2,2))
165.     plot(regression)
166.     dev.off()
167. }

```



```

Call:
lm(formula = filtered$new_Close ~ filtered$tokenAmount + filtered$n +
    filtered$Open)

Residuals:
      Min       1Q   Median       3Q      Max
-0.088379 -0.009174 -0.003581  0.007236  0.206421

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    8.838e-03  3.386e-03   2.610  0.00959 **
filtered$tokenAmount -9.105e-19  2.324e-18  -0.392  0.69548
filtered$n       3.239e-06  9.140e-06   0.354  0.72334
filtered$Open    9.211e-01  2.441e-02  37.727 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.02653 on 248 degrees of freedom
(1 observation deleted due to missingness)
Multiple R-squared:  0.8571,    Adjusted R-squared:  0.8554
F-statistic: 495.8 on 3 and 248 DF,  p-value: < 2.2e-16

```

Aragon

```

1. library(magrittr)
2. library(dplyr)
3. library(ggplot2)
4. library(readr)
5. library(fitdistrplus)
6. library(DAAG)
7. library("ggplot2")
8. library(anytime)
9.
10. aragon <- read_delim('C:/Users/ygaoq/OneDrive/MyDocuments/2019 Spring/Statistics/Project/Blockchain-Tokens-Data-Analytics/networkaragonTX.txt', delim = " ", col_names = F)
11. names(aragon) <- c('fromID', 'toID', 'unixTime', 'tokenAmount')
12. decimals <- 10^18
13. supply <- 39609523
14. aragonFiltered <-
    filter(aragon, tokenAmount < decimals * supply) #filter out all outliers
15.

```

```

16. #figure out how many users induced those unnormal transaction
17. aragon_outliers<- filter(aragon,tokenAmount >= decimals * supply)
18. user_outliers <- aragon_outliers %>% group_by(toID) %>% summarise(nn = n()) %>% ungroup

19. number_users_outliers<-nrow(user_outliers)
20. number_users_outliers
21.
22. #get top X buyers data
23. buys<-
  aragonFiltered%>% group_by(toID) %>% summarise(nn = n()) %>% ungroup #change the supply
  and decimals amount
24. buys_sorted_dec<-buys[order(-buys$n),]
25. #top 30 active buyers and number of buys
26. top_30_buyers<-buys_sorted_dec%>%head(30)
27. top_30_buyers
28.
29. #####Question 1#####
30.
31. #####group by user pairs####
32. buys_pairs<-
  aragonFiltered%>% group_by(fromID, toID) %>% summarise(nn = n()) %>% ungroup
33. for (row in 1:nrow(buys_pairs)) {
34.   a<-buys_pairs[row,"fromID"]
35.   b<-buys_pairs[row,"toID"]
36.   for (inner_row in row:nrow(buys_pairs)) {
37.     c<-buys_pairs[inner_row,"fromID"]
38.     d<-buys_pairs[inner_row,"toID"]
39.     if(a==d&&b==c){
40.       buys_pairs[inner_row,"fromID"]<-d
41.       buys_pairs[inner_row,"toID"]<-c
42.     }
43.   }
44. }
45. buys_pairs<-
  aragonFiltered%>% group_by(fromID*toID+fromID+toID) %>% summarise(nn = n()) %>% ungroup

46. buys_pairs<-
  aragonFiltered%>% group_by(fromID, toID) %>% summarise(nn = n()) %>% ungroup
47. buys_pair_sorted_asc<-buys_pairs[order(buys_pairs$n),]
48. buys_pair_less_30<-subset(buys_pair_sorted_asc,n<30)
49. buys_pair_data<-buys_pair_less_30
50.
51. #####find out estimates of parameters of several distribution based on the buys_pairs d
  ata set####
52. exp_dis <- fitdist(buys_pair_data$n, 'exp')
53. exp_dis
54. gamma_dis <- fitdist(buys_pair_data$n, 'gamma')
55. gamma_dis
56. lnorm_dis <- fitdist(buys_pair_data$n, 'lnorm')
57. lnorm_dis
58. pois_dis <- fitdist(buys_pair_data$n, 'pois')
59. pois_dis
60. weibull_dis <- fitdist(buys_pair_data$n, 'weibull')
61. weibull_dis
62.
63. gofstat(list(exp_dis, gamma_dis, lnorm_dis))
64. descdist(buys_sorted_asc$n,boot=1000)
65.
66. #lognorm
67. fit_lnorm <- fitdist(buys_pair_less_30$n,"lnorm")

```

```

68. summary(fit_lnorm)
69. plot(fit_lnorm)
70. cdfcomp(fit_lnorm)
71.
72. #exp
73. fit_exp <- fitdist(buys_pair_less_30$n,"exp")
74. summary(fit_exp)
75. plot(fit_exp)
76. cdfcomp(fit_exp)
77.
78. #gamma
79. fit_gamma <- fitdist(buys_pair_less_30$n,"gamma")
80. summary(fit_gamma)
81. plot(fit_gamma)
82. cdfcomp(fit_gamma)
83.
84. #weibull
85. fit_weibull <- fitdist(buys_pair_less_30$n,"weibull")
86. summary(fit_weibull)
87.
88. #normal
89. fit_normal <- fitdist(buys_pair_less_30$n,"norm")
90. summary(fit_normal)
91.
92. #pois
93. #normal
94. fit_pois <- fitdist(buys_pair_less_30$n,"pois")
95. summary(fit_pois)
96.
97. #unif
98. fit_unif <- fitdist(buys_pair_less_30$n,"unif")
99. summary(fit_unif)
100.   plot(fit_unif)
101.   cdfcomp(fit_unif)
102.
103.   #####draw graph#####
104.   all_density <- ggplot(data=buys_pair_less_30) +
105.     geom_histogram(bins=30,aes(x = buys_pair_less_30$n, ..density..)) +
106.     stat_function(fun = dlnorm, args = list(meanlog = 0.1537565, sdlog = 0.4012176
107.   ),
108.     colour = "red")+
109.     stat_function(fun = dgamma, args = list(shape = 4.034757, rate=3.040920),
110.     colour = "blue")+
111.     stat_function(fun=dexp, args=list(rate=0.7536976),colour="green")+
112.     stat_function(fun=dweibull, args=list(shape=1.464309, scale=1.488009),colour="
113.   yellow")+
114.     stat_function(fun=dpois, args=list(lambda=1.326792),colour="orange")+
115.     xlab("No.Buys")
116.   all_density
117.
118.   #####Question 1#####
119.   #####Question 2#####
120.   aragon_prices <- read_delim("C:/Users/ygaoq/OneDrive/MyDocuments/2019 Spring/Statistics/Project/Blockchain-Tokens-Data-Analytics/aragon", delim = "\t", col_names = T) #load token price data
121.   names(aragon_prices) <- make.names(names(aragon_prices))

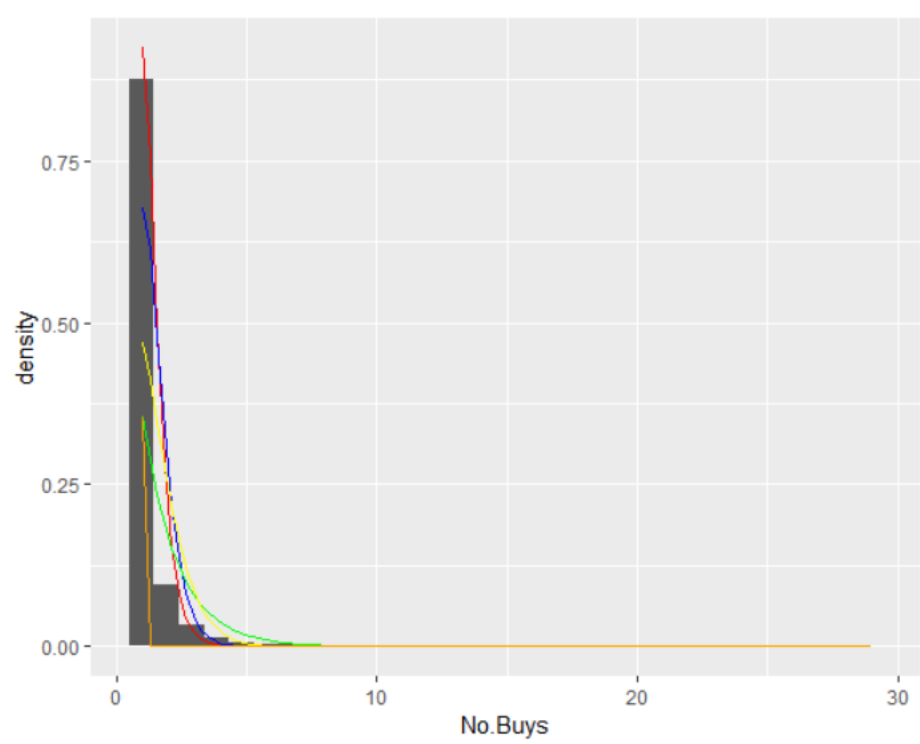
```

```

122.     aragon_prices <- aragon_prices %>% mutate(date = as.Date(Date, format = '%m/%d/%
Y'))
123.
124.     aragon <- read_delim('C:/Users/ygaoq/OneDrive/MyDocuments/2019 Spring/Statistics
/Project/Blockchain-Tokens-Data-
Analytics/networkkaragonTX.txt', delim = " ", col_names = F)
125.     names(aragon) <- c('fromID', 'toID', 'unixTime', 'tokenAmount')
126.     decimals <- 10^18
127.     supply <- 39609523
128.     aragon_filtered <- filter(aragon, tokenAmount < decimals * supply)
129.     ## convert data type of unixTime
130.     aragon_filtered <- aragon_filtered %>%
131.       mutate(date = anydate(unixTime))
132.     names(aragon_filtered) <- c('fromID', 'toID', 'unixTime', 'tokenAmount', 'date')

133.
134.     ## merge the prices and edge
135.     aragon_merged <-
merge(x = aragon_prices, y = aragon_filtered, by = "date", all.x = TRUE)
136.
137.     #####Determin K#####
138.     top_30_buyers <- buys_sorted_dec %>% head(30)
139.
140.     top_K <- c(1:30)
141.     count <- 1
142.     for (val in top_K) {
143.       top_K_buyers <- buys_sorted_dec %>% head(val)
144.       filter_K_aragon_merged <- filter(aragon_merged, toID %in% top_K_buyers$toID)
145.       filter_K_aragon_merged = transform(filter_K_aragon_merged, average_price = (Open + C
lose)/2)
146.       filter_K_aragon_merged$num_Date <- as.numeric(as.POSIXct(filter_K_aragon_merg
ed$date))
147.       filered <-
filter_K_aragon_merged %>% group_by(num_Date) %>% summarise(nn = n(), Close = mean(Close), t
okenAmount = sum(tokenAmount), Open = mean(Open))
148.       shift <- function(x, n){
149.         c(x[-(seq(n))], rep(NA, n))
150.       }
151.       filered$new_Close <- shift(filered$Close, 1)
152.       num_rows <- nrow(filered)
153.       filered[-num_rows,]
154.       regression <- lm(filered$new_Close ~ filered$tokenAmount + filered$n + filered$Open)
155.
156.       setwd("C:/Users/ygaoq/Desktop/aragon")
157.       yourfilename = paste("W", val, ".txt", sep = "")
158.       capture.output(summary(regression), append = TRUE, file = "C:/Users/ygaoq/Deskto
p/aragon/Final_Result.txt")
159.
160.
161.       summary(regression)
162.       par(mfcol = c(2, 2))
163.       setwd("C:/Users/ygaoq/Desktop/aragon")
164.       yourfilename = paste("A", val, ".png", sep = "")
165.       png(file = yourfilename)
166.       opar <- par(mfrow = c(2, 2))
167.       plot(regression)
168.       dev.off()
169.     }

```

Call:

```
lm(formula = filered$new_Close ~ filered$tokenAmount + filered$n +
    filered$Open)
```

Residuals:

Min	1Q	Median	3Q	Max
-1.83279	-0.22306	-0.02322	0.17976	2.44810

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	1.470e-01	6.085e-02	2.416	0.0162 *
filered\$tokenAmount	5.187e-25	6.673e-25	0.777	0.4375
filered\$n	-6.161e-05	1.872e-04	-0.329	0.7422
filered\$Open	9.511e-01	1.777e-02	53.518	<2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.433 on 348 degrees of freedom
(1 observation deleted due to missingness)

Multiple R-squared: 0.9002, Adjusted R-squared: 0.8993

F-statistic: 1046 on 3 and 348 DF, p-value: < 2.2e-16

Bitqy

```
1. library(magrittr)
2. library(dplyr)
3. library(ggplot2)
4. library(readr)
5. library(fitdistrplus)
6. library(DAAG)
7. library("ggplot2")
8. library(anytime)
9.
10. bitqy <- read_delim('C:/Users/ygaoq/OneDrive/MyDocuments/2019 Spring/Statistics/Project
    /Blockchain-Tokens-Data-Analytics/networkbitqyTX.txt', delim = " ", col_names = F)
11. names(bitqy) <- c('fromID', 'toID', 'unixTime', 'tokenAmount')
12. decimals <- 2
13. supply <- 1 * 10^10
14. bitqyFiltered <-
    filter(bitqy, tokenAmount < decimals * supply) #filter out all outliers
15.
16. #figure out how many users induced those unnormal transaction
17. bitqy_outliers <- filter(bitqy, tokenAmount >= decimals * supply)
18. user_outliers <- bitqy_outliers %>% group_by(toID) %>% summarise(nn = n()) %>% ungroup
19. number_users_outliers <- nrow(user_outliers)
20. number_users_outliers
21.
22. #get top X buyers data
23. buys <-
    bitqyFiltered %>% group_by(toID) %>% summarise(nn = n()) %>% ungroup #change the supply
    and decimals amount
24. buys_sorted_dec <- buys[order(-buys$nn),]
25. #top 30 active buyers and number of buys
26. top_30_buyers <- buys_sorted_dec %>% head(30)
27. top_30_buyers
28.
29. #####Question 1#####
    #####
30.
31. #####group by user pairs#####
32. buys_pairs <-
    bitqyFiltered %>% group_by(fromID, toID) %>% summarise(nn = n()) %>% ungroup
33. for (row in 1:nrow(buys_pairs)) {
34.   a <- buys_pairs[row, "fromID"]
35.   b <- buys_pairs[row, "toID"]
36.   for (inner_row in row:nrow(buys_pairs)) {
37.     c <- buys_pairs[inner_row, "fromID"]
38.     d <- buys_pairs[inner_row, "toID"]
39.     if (a == d && b == c) {
40.       buys_pairs[inner_row, "fromID"] <- d
41.       buys_pairs[inner_row, "toID"] <- c
42.     }
43.   }
44. }
45. buys_pairs <-
    bitqyFiltered %>% group_by(fromID*toID+fromID+toID) %>% summarise(nn = n()) %>% ungroup
46. buys_pairs <-
    bitqyFiltered %>% group_by(fromID, toID) %>% summarise(nn = n()) %>% ungroup
```

```

47. buys_pair_sorted_asc<-buys_pairs[order(buys_pairs$n),]
48. buys_pair_less_30<-subset(buys_pair_sorted_asc,n<30)
49. buys_pair_data<-buys_pair_less_30
50.
51. #####find out estimates of paramaters of several distribution based on the buys_pairs d
    ata set#####
52. exp_dis <- fitdist(buys_pair_data$n, 'exp')
53. exp_dis
54. gamma_dis <- fitdist(buys_pair_data$n, 'gamma')
55. gamma_dis
56. lnorm_dis <- fitdist(buys_pair_data$n, 'lnorm')
57. lnorm_dis
58. pois_dis <- fitdist(buys_pair_data$n, 'pois')
59. pois_dis
60. weibull_dis <- fitdist(buys_pair_data$n, 'weibull')
61. weibull_dis
62.
63. gofstat(list(exp_dis, gamma_dis, lnorm_dis))
64. descdist(buys_sorted_asc$n,boot=1000)
65.
66. #lognorm
67. fit_lnorm <- fitdist(buys_pair_less_30$n,"lnorm")
68. summary(fit_lnorm)
69. plot(fit_lnorm)
70. cdfcomp(fit_lnorm)
71.
72. #exp
73. fit_exp <- fitdist(buys_pair_less_30$n,"exp")
74. summary(fit_exp)
75. plot(fit_exp)
76. cdfcomp(fit_exp)
77.
78. #gamma
79. fit_gamma <- fitdist(buys_pair_less_30$n,"gamma")
80. summary(fit_gamma)
81. plot(fit_gamma)
82. cdfcomp(fit_gamma)
83.
84. #weibull
85. fit_weibull <- fitdist(buys_pair_less_30$n,"weibull")
86. summary(fit_weibull)
87.
88. #normal
89. fit_normal <- fitdist(buys_pair_less_30$n,"norm")
90. summary(fit_normal)
91.
92. #pois
93. #normal
94. fit_pois <- fitdist(buys_pair_less_30$n,"pois")
95. summary(fit_pois)
96.
97. #unif
98. fit_unif <- fitdist(buys_pair_less_30$n,"unif")
99. summary(fit_unif)
100. plot(fit_unif)
101. cdfcomp(fit_unif)
102.
103. #####draw graph#####
104. all_density <- ggplot(data=buys_pair_less_30) +
105.   geom_histogram(bins=30,aes(x = buys_pair_less_30$n, ..density..)) +

```

```

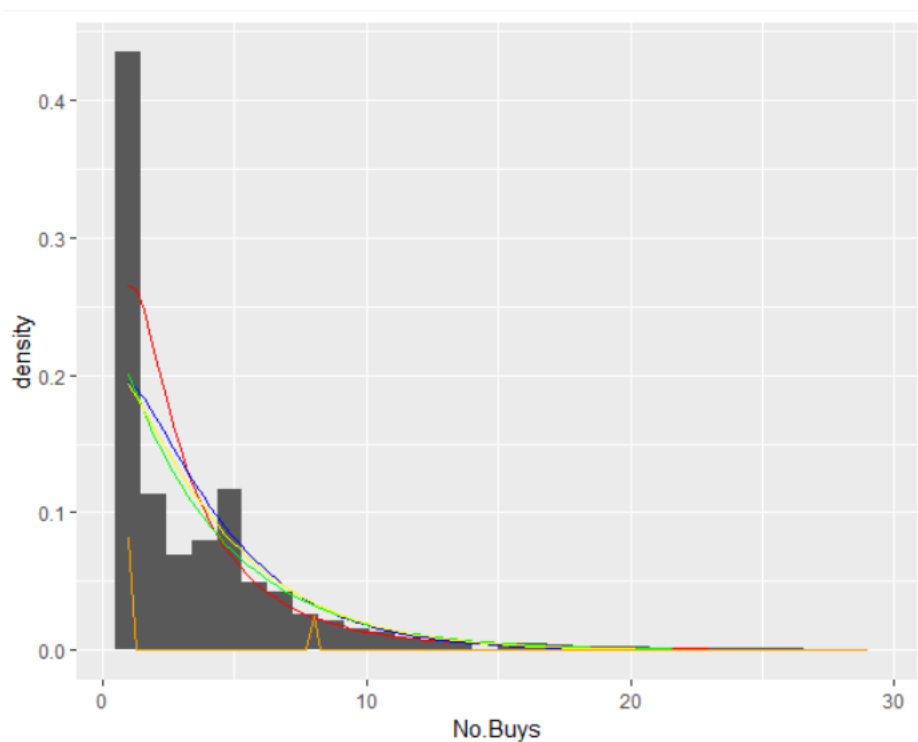
106.      stat_function(fun = dlnorm, args = list(meanlog = 0.9122759, sdlog = 0.9075324
),
107.          colour = "red")+
108.      stat_function(fun = dgamma, args = list(shape = 1.2923088, rate=0.3361498),
109.          colour = "blue")+
110.      stat_function(fun=dexp, args=list(rate=0.2600901),colour="green")+
111.      stat_function(fun=dweibull, args=list(shape=1.083871, scale=3.982783),colour="
yellow")+
112.      stat_function(fun=dpois, args=list(lambda=3.844821),colour="orange")+
113.      xlab("No.Buys")
114.      all_density
115.
116.      #####Question 1#####
#####
117.
118.
119.      #####Question 2#####
#####
120.      bitqy_prices <- read_delim("C:/Users/ygaoq/OneDrive/MyDocuments/2019 Spring/Stat
istics/Project/Blockchain-Tokens-Data-
Analytics/bitqy", delim = "\t", col_names = T) #load token price data
121.      names(bitqy_prices) <- make.names(names(bitqy_prices))
122.      bitqy_prices <- bitqy_prices %>% mutate(date = as.Date(Date, format = '%m/%d/%Y'
))
123.
124.      bitqy <- read_delim('C:/Users/ygaoq/OneDrive/MyDocuments/2019 Spring/Statistics/
Project/Blockchain-Tokens-Data-
Analytics/networkbitqyTX.txt', delim = " ", col_names = F)
125.      names(bitqy) <- c('fromID', 'toID', 'unixTime', 'tokenAmount')
126.      decimals <- 2
127.      supply <- 1 * 10^10
128.      bitqy_filtered <-filter(bitqy,tokenAmount < decimals * supply)
129.      ## convert data type of unixTime
130.      bitqy_filtered <- bitqy_filtered %>%
131.          mutate(date = anydate(unixTime))
132.      names(bitqy_filtered) <- c('fromID', 'toID', 'unixTime', 'tokenAmount', 'date')
133.
134.      ## merge the prices and edge
135.      bitqy_merged<-
merge(x = bitqy_prices, y = bitqy_filtered, by = "date", all.x = TRUE)
136.
137.      #####Determin K#####
138.      top_30_buyers<-buys_sorted_dec%>%head(30)
139.
140.      top_K<-c(1:30)
141.      count <- 1
142.      for (val in top_K) {
143.          top_K_buyers<-buys_sorted_dec%>%head(val)
144.          filter_K_bitqy_merged<-filter(bitqy_merged,toID %in% top_K_buyers$toID)
145.          filter_K_bitqy_merged=transform(filter_K_bitqy_merged,average_price= (Open+Clo
se)/2)
146.          filter_K_bitqy_merged$num_Date <- as.numeric(as.POSIXct(filter_K_bitqy_merged
$date))
147.          filered<-
filter_K_bitqy_merged%>% group_by(num_Date) %>% summarise(nn = n(),Close=mean(Close),to
kenAmount=sum(tokenAmount),Open=mean(Open))
148.          shift <- function(x, n){
149.              c(x[-(seq(n))], rep(NA, n))
150.          }
151.          filered$new_Close<-shift(filered$Close,1)

```

```

152.     num_rows<-nrow(filered)
153.     filered[-num_rows,]
154.     regression<-lm(filered$new_Close~filered$tokenAmount+filered$n+filered$Open)
155.
156.     setwd("C:/Users/ygaoq/Desktop/bitqy")
157.     yourfilename=paste("W",val,".txt",sep="")
158.     capture.output(summary(regression),append = TRUE,file = "C:/Users/ygaoq/Desktop
p/bitqy/Final_Result.txt")
159.
160.
161.     summary(regression)
162.     par(mfcol=c(2,2))
163.     setwd("C:/Users/ygaoq/Desktop/bitqy")
164.     yourfilename=paste("A",val,".png",sep="")
165.     png(file=yourfilename)
166.     opar <- par(mfrow=c(2,2))
167.     plot(regression)
168.     dev.off()
169. }

```



Call:

```
lm(formula = filered$new_Close ~ filered$tokenAmount + filered$n +  
    filered$Open)
```

Residuals:

	Min	1Q	Median	3Q	Max
	-0.027199	-0.001834	-0.001114	0.000887	0.032898

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	2.095e-03	6.423e-04	3.262	0.00127	**
filered\$tokenAmount	-4.231e-14	4.916e-14	-0.861	0.39030	
filered\$n	2.345e-06	9.470e-06	0.248	0.80461	
filered\$Open	8.171e-01	3.222e-02	25.356	< 2e-16	***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0064 on 240 degrees of freedom
(1 observation deleted due to missingness)

Multiple R-squared: 0.7473, Adjusted R-squared: 0.7442

F-statistic: 236.6 on 3 and 240 DF, p-value: < 2.2e-16