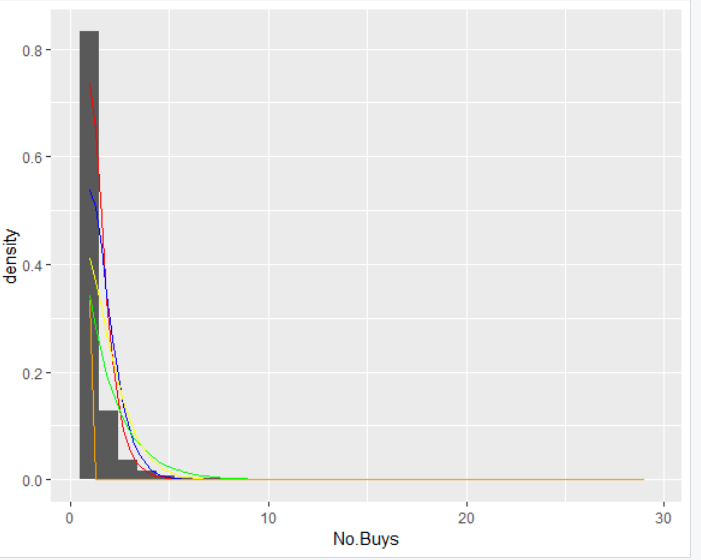
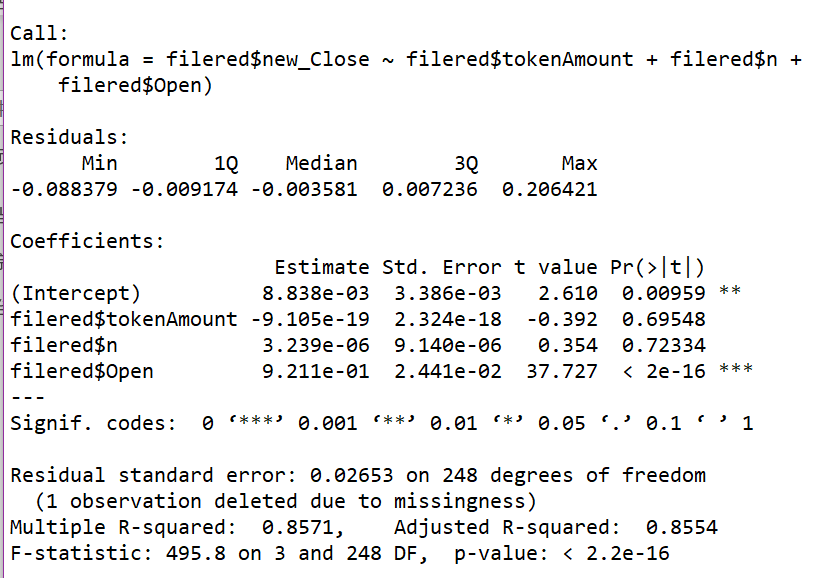
Tierion

1. library(magrittr)
2. library(dplyr)
3. library(ggplot2)
4. library(readr)
5. library(fitdistrplus)
6. library(DAAG)
7. library("ggplot2")
8. library(anytime)
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
16. #figure out how many users indruced those unnormal transcation
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
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$n),]
25. #top 30 active buyers and number of buys
26. top\_30\_buyers**<-buys\_sorted\_dec**%**>**%head(30)
27. top\_30\_buyers
29. ########################################Question 1############################################
31. #####group by user pairs#####
32. buys\_pairs**<-tierionFiltered**%**>**% group\_by(fromID, toID) %**>**% summarise(nn = n()) %**>**% ungroup
33. for (row in 1:nrow(buys\_pairs)) {
34. a**<-buys\_pairs**[row,"fromID"]
35. b**<-buys\_pairs**[row,"toID"]
36. for (inner\_row in row:nrow(buys\_pairs)) {
37. c**<-buys\_pairs**[inner\_row,"fromID"]
38. d**<-buys\_pairs**[inner\_row,"toID"]
39. if(a==d&&b==c){
40. buys\_pairs[inner\_row,"fromID"]**<-d**
41. buys\_pairs[inner\_row,"toID"]**<-c**
42. }
43. }
44. }
45. buys\_pairs**<-tierionFiltered**%**>**% group\_by(fromID\*toID+fromID+toID) %**>**% summarise(nn = n()) %**>**% ungroup
46. buys\_pairs**<-tierionFiltered**%**>**% group\_by(fromID, toID) %**>**% summarise(nn = n()) %**>**% ungroup
47. buys\_pair\_sorted\_asc**<-buys\_pairs**[order(buys\_pairs$n),]
48. buys\_pair\_less\_30**<-subset**(buys\_pair\_sorted\_asc,n**<30**)
50. #####find out estimates of paramaters of several distribution based on the buys\_pairs data 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$n, 'weibull')
60. weibull\_dis
62. gofstat(list(exp\_dis, gamma\_dis, lnorm\_dis))
63. descdist(buys\_sorted\_asc$n,boot=1000)
65. #lognorm
66. fit\_lnorm **<-** fitdist(buys\_pair\_less\_30$n,"lnorm")
67. summary(fit\_lnorm)
68. plot(fit\_lnorm)
69. cdfcomp(fit\_lnorm)
71. #exp
72. fit\_exp **<-** fitdist(buys\_pair\_less\_30$n,"exp")
73. summary(fit\_exp)
74. plot(fit\_exp)
75. cdfcomp(fit\_exp)
77. #gamma
78. fit\_gamma **<-** fitdist(buys\_pair\_less\_30$n,"gamma")
79. summary(fit\_gamma)
80. plot(fit\_gamma)
81. cdfcomp(fit\_gamma)
83. #weibull
84. fit\_weibull **<-** fitdist(buys\_pair\_less\_30$n,"weibull")
85. summary(fit\_weibull)
87. #normal
88. fit\_normal **<-** fitdist(buys\_pair\_less\_30$n,"norm")
89. summary(fit\_normal)
91. #pois
92. #normal
93. fit\_pois **<-** fitdist(buys\_pair\_less\_30$n,"pois")
94. summary(fit\_pois)
96. #unif
97. fit\_unif **<-** fitdist(buys\_pair\_less\_30$n,"unif")
98. summary(fit\_unif)
99. plot(fit\_unif)
100. cdfcomp(fit\_unif)
102. ######################draw graph#######################
103. all\_density **<-** ggplot(data=buys\_pair\_less\_30) +
104. geom\_histogram(bins=30,aes(x = buys\_pair\_less\_30$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("No.Buys")
112. all\_density
114. ########################################Question 1############################################

117. ########################################Question 2############################################
118. tierion\_prices **<-** read\_delim("C:/Users/ygaoq/OneDrive/MyDocuments/2019 Spring/Statistics/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'))
122. tierion **<-** read\_delim('C:/Users/ygaoq/OneDrive/MyDocuments/2019 Spring/Statistics/Project/Blockchain-Tokens-Data-Analytics/networktierionTX.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')
132. ## merge the prices and edge
133. tierion\_merged**<-merge**(x = tierion\_prices, y = tierion\_filtered, by = "date", all.x = TRUE)
135. ################Determin K##########################
136. top\_30\_buyers**<-buys\_sorted\_dec**%**>**%head(30)
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\_merged$date))
145. filered**<-filter\_K\_tierion\_merged**%**>**% group\_by(num\_Date) %**>**% summarise(nn = n(),Close=mean(Close),tokenAmount=sum(tokenAmount),Open=mean(Open))
146. shift **<-** function(x, n){
147. c(x[-(seq(n))], rep(NA, n))
148. }
149. filered$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)
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/Tierion/Final\_Result.txt")

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. }



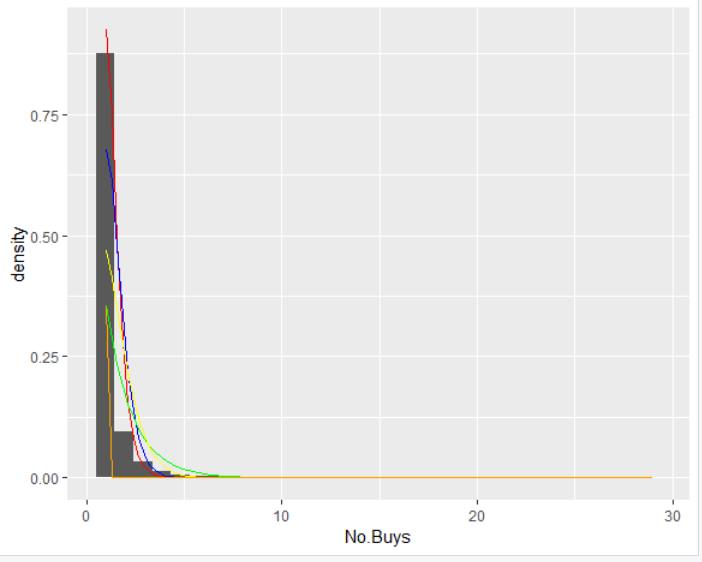


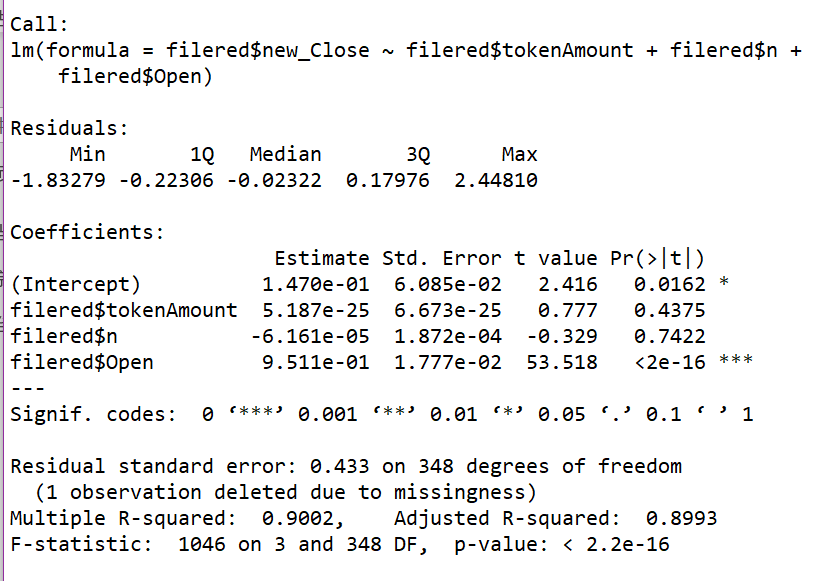
Aragon

1. library(magrittr)
2. library(dplyr)
3. library(ggplot2)
4. library(readr)
5. library(fitdistrplus)
6. library(DAAG)
7. library("ggplot2")
8. library(anytime)
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
16. #figure out how many users indruced those unnormal transcation
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
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$n),]
25. #top 30 active buyers and number of buys
26. top\_30\_buyers**<-buys\_sorted\_dec**%**>**%head(30)
27. top\_30\_buyers
29. ########################################Question 1############################################
31. #####group by user pairs#####
32. buys\_pairs**<-aragonFiltered**%**>**% group\_by(fromID, toID) %**>**% summarise(nn = n()) %**>**% ungroup
33. for (row in 1:nrow(buys\_pairs)) {
34. a**<-buys\_pairs**[row,"fromID"]
35. b**<-buys\_pairs**[row,"toID"]
36. for (inner\_row in row:nrow(buys\_pairs)) {
37. c**<-buys\_pairs**[inner\_row,"fromID"]
38. d**<-buys\_pairs**[inner\_row,"toID"]
39. if(a==d&&b==c){
40. buys\_pairs[inner\_row,"fromID"]**<-d**
41. buys\_pairs[inner\_row,"toID"]**<-c**
42. }
43. }
44. }
45. buys\_pairs**<-aragonFiltered**%**>**% group\_by(fromID\*toID+fromID+toID) %**>**% summarise(nn = n()) %**>**% ungroup
46. buys\_pairs**<-aragonFiltered**%**>**% group\_by(fromID, toID) %**>**% summarise(nn = n()) %**>**% ungroup
47. buys\_pair\_sorted\_asc**<-buys\_pairs**[order(buys\_pairs$n),]
48. buys\_pair\_less\_30**<-subset**(buys\_pair\_sorted\_asc,n**<30**)
49. buys\_pair\_data**<-buys\_pair\_less\_30**
51. #####find out estimates of paramaters of several distribution based on the buys\_pairs data 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$n, 'weibull')
61. weibull\_dis
63. gofstat(list(exp\_dis, gamma\_dis, lnorm\_dis))
64. descdist(buys\_sorted\_asc$n,boot=1000)
66. #lognorm
67. fit\_lnorm **<-** fitdist(buys\_pair\_less\_30$n,"lnorm")
68. summary(fit\_lnorm)
69. plot(fit\_lnorm)
70. cdfcomp(fit\_lnorm)
72. #exp
73. fit\_exp **<-** fitdist(buys\_pair\_less\_30$n,"exp")
74. summary(fit\_exp)
75. plot(fit\_exp)
76. cdfcomp(fit\_exp)
78. #gamma
79. fit\_gamma **<-** fitdist(buys\_pair\_less\_30$n,"gamma")
80. summary(fit\_gamma)
81. plot(fit\_gamma)
82. cdfcomp(fit\_gamma)
84. #weibull
85. fit\_weibull **<-** fitdist(buys\_pair\_less\_30$n,"weibull")
86. summary(fit\_weibull)
88. #normal
89. fit\_normal **<-** fitdist(buys\_pair\_less\_30$n,"norm")
90. summary(fit\_normal)
92. #pois
93. #normal
94. fit\_pois **<-** fitdist(buys\_pair\_less\_30$n,"pois")
95. summary(fit\_pois)
97. #unif
98. fit\_unif **<-** fitdist(buys\_pair\_less\_30$n,"unif")
99. summary(fit\_unif)
100. plot(fit\_unif)
101. cdfcomp(fit\_unif)
103. ######################draw graph#######################
104. all\_density **<-** ggplot(data=buys\_pair\_less\_30) +
105. geom\_histogram(bins=30,aes(x = buys\_pair\_less\_30$n, ..density..)) +
106. stat\_function(fun = dlnorm, args = list(meanlog = 0.1537565, sdlog = 0.4012176),
107. colour = "red")+
108. stat\_function(fun = dgamma, args = list(shape = 4.034757, rate=3.040920),
109. colour = "blue")+
110. stat\_function(fun=dexp, args=list(rate=0.7536976),colour="green")+
111. stat\_function(fun=dweibull, args=list(shape=1.464309, scale=1.488009),colour="yellow")+
112. stat\_function(fun=dpois, args=list(lambda=1.326792),colour="orange")+
113. xlab("No.Buys")
114. all\_density
116. ########################################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'))
124. aragon **<-** read\_delim('C:/Users/ygaoq/OneDrive/MyDocuments/2019 Spring/Statistics/Project/Blockchain-Tokens-Data-Analytics/networkaragonTX.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')
134. ## merge the prices and edge
135. aragon\_merged**<-merge**(x = aragon\_prices, y = aragon\_filtered, by = "date", all.x = TRUE)
137. ################Determin K##########################
138. top\_30\_buyers**<-buys\_sorted\_dec**%**>**%head(30)
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$toID)
145. filter\_K\_aragon\_merged=transform(filter\_K\_aragon\_merged,average\_price= (Open+Close)/2)
146. filter\_K\_aragon\_merged$num\_Date **<-**  as.numeric(as.POSIXct(filter\_K\_aragon\_merged$date))
147. filered**<-filter\_K\_aragon\_merged**%**>**% group\_by(num\_Date) %**>**% summarise(nn = n(),Close=mean(Close),tokenAmount=sum(tokenAmount),Open=mean(Open))
148. shift **<-** function(x, n){
149. c(x[-(seq(n))], rep(NA, n))
150. }
151. filered$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)
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/Desktop/aragon/Final\_Result.txt")

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. }





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)
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
16. #figure out how many users indruced those unnormal transcation
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
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$n),]
25. #top 30 active buyers and number of buys
26. top\_30\_buyers**<-buys\_sorted\_dec**%**>**%head(30)
27. top\_30\_buyers
29. ########################################Question 1############################################
31. #####group by user pairs#####
32. buys\_pairs**<-bitqyFiltered**%**>**% group\_by(fromID, toID) %**>**% summarise(nn = n()) %**>**% ungroup
33. for (row in 1:nrow(buys\_pairs)) {
34. a**<-buys\_pairs**[row,"fromID"]
35. b**<-buys\_pairs**[row,"toID"]
36. for (inner\_row in row:nrow(buys\_pairs)) {
37. c**<-buys\_pairs**[inner\_row,"fromID"]
38. d**<-buys\_pairs**[inner\_row,"toID"]
39. if(a==d&&b==c){
40. buys\_pairs[inner\_row,"fromID"]**<-d**
41. buys\_pairs[inner\_row,"toID"]**<-c**
42. }
43. }
44. }
45. buys\_pairs**<-bitqyFiltered**%**>**% group\_by(fromID\*toID+fromID+toID) %**>**% summarise(nn = n()) %**>**% ungroup
46. buys\_pairs**<-bitqyFiltered**%**>**% group\_by(fromID, toID) %**>**% summarise(nn = n()) %**>**% ungroup
47. buys\_pair\_sorted\_asc**<-buys\_pairs**[order(buys\_pairs$n),]
48. buys\_pair\_less\_30**<-subset**(buys\_pair\_sorted\_asc,n**<30**)
49. buys\_pair\_data**<-buys\_pair\_less\_30**
51. #####find out estimates of paramaters of several distribution based on the buys\_pairs data 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$n, 'weibull')
61. weibull\_dis
63. gofstat(list(exp\_dis, gamma\_dis, lnorm\_dis))
64. descdist(buys\_sorted\_asc$n,boot=1000)
66. #lognorm
67. fit\_lnorm **<-** fitdist(buys\_pair\_less\_30$n,"lnorm")
68. summary(fit\_lnorm)
69. plot(fit\_lnorm)
70. cdfcomp(fit\_lnorm)
72. #exp
73. fit\_exp **<-** fitdist(buys\_pair\_less\_30$n,"exp")
74. summary(fit\_exp)
75. plot(fit\_exp)
76. cdfcomp(fit\_exp)
78. #gamma
79. fit\_gamma **<-** fitdist(buys\_pair\_less\_30$n,"gamma")
80. summary(fit\_gamma)
81. plot(fit\_gamma)
82. cdfcomp(fit\_gamma)
84. #weibull
85. fit\_weibull **<-** fitdist(buys\_pair\_less\_30$n,"weibull")
86. summary(fit\_weibull)
88. #normal
89. fit\_normal **<-** fitdist(buys\_pair\_less\_30$n,"norm")
90. summary(fit\_normal)
92. #pois
93. #normal
94. fit\_pois **<-** fitdist(buys\_pair\_less\_30$n,"pois")
95. summary(fit\_pois)
97. #unif
98. fit\_unif **<-** fitdist(buys\_pair\_less\_30$n,"unif")
99. summary(fit\_unif)
100. plot(fit\_unif)
101. cdfcomp(fit\_unif)
103. ######################draw graph#######################
104. all\_density **<-** ggplot(data=buys\_pair\_less\_30) +
105. geom\_histogram(bins=30,aes(x = buys\_pair\_less\_30$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
116. ########################################Question 1############################################

119. ########################################Question 2############################################
120. bitqy\_prices **<-** read\_delim("C:/Users/ygaoq/OneDrive/MyDocuments/2019 Spring/Statistics/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'))
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')
134. ## merge the prices and edge
135. bitqy\_merged**<-merge**(x = bitqy\_prices, y = bitqy\_filtered, by = "date", all.x = TRUE)
137. ################Determin K##########################
138. top\_30\_buyers**<-buys\_sorted\_dec**%**>**%head(30)
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$toID)
145. filter\_K\_bitqy\_merged=transform(filter\_K\_bitqy\_merged,average\_price= (Open+Close)/2)
146. filter\_K\_bitqy\_merged$num\_Date **<-**  as.numeric(as.POSIXct(filter\_K\_bitqy\_merged$date))
147. filered**<-filter\_K\_bitqy\_merged**%**>**% group\_by(num\_Date) %**>**% summarise(nn = n(),Close=mean(Close),tokenAmount=sum(tokenAmount),Open=mean(Open))
148. shift **<-** function(x, n){
149. c(x[-(seq(n))], rep(NA, n))
150. }
151. filered$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)
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/bitqy/Final\_Result.txt")

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. }

