BlockchainL Project Status Notebook Template

Decentralized Finance

Chris Cammilleri

11 November 2021

Contents

Weekly Work Summary 1
Personal Contribution 1
Discussion of Primary Findings 1

Weekly Work Summary

- RCS ID: cammic
- Project Name: Blockchain
- Updated dataset
- Performed clustering to help distinguish liquidator behavior

Personal Contribution

All contributions were completed by me.

Discussion of Primary Findings

The first thing I did this week was update the dataset. Some of the changes included creating an alias for User Id's, adding collateral change transactions, adding a column representing if the transaction was done by an Aave protocol smart contract, and fixing the prices for WETH. The updated dataset is called transactionsv2.Rds and has been uploaded under the Data file.

My second task was to perform clustering on the users, but this time based on the features that distinguish liquidators and non-liquidators. The goal of this is to find user behaviors that are similar to those who liquidate. The code for this process is below

```
#import libraries
library(ggplot2)
library(ggbiplot)
```

```
## Loading required package: plyr
## Loading required package: scales
## Loading required package: grid
```

```
library(gplots)
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
library(RColorBrewer)
library(beeswarm)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v tibble  3.0.6     v dplyr   1.0.7
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
## v purrr   0.3.4

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::arrange()    masks plyr::arrange()
## x readr::col_factor() masks scales::col_factor()
## x purrr::compact()    masks plyr::compact()
## x dplyr::count()      masks plyr::count()
## x purrr::discard()    masks scales::discard()
## x dplyr::failwith()   masks plyr::failwith()
## x dplyr::filter()     masks stats::filter()
## x dplyr::id()         masks plyr::id()
## x dplyr::lag()        masks stats::lag()
## x dplyr::mutate()     masks plyr::mutate()
## x dplyr::rename()     masks plyr::rename()
## x dplyr::summarise()  masks plyr::summarise()
## x dplyr::summarize()  masks plyr::summarize()
```

```
library(ggbeeswarm)
library(foreach)
```

```
##
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

```
library(doParallel)
```

```
## Loading required package: iterators

## Loading required package: parallel
```

Load Data

We start by loading version 2 of the transaction data in to a dataframe, and remove transactions done by the Aave protocol.

```
#load in csv file to data frame
df<-read_csv(file='~/Blockchain/transactions2.csv')
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_logical(),
##   amount = col_double(),
```

```
##   borrowRate = col_double(),
##   borrowRateMode = col_character(),
##   onBehalfOf = col_character(),
##   pool = col_character(),
##   reserve = col_character(),
##   timestamp = col_double(),
##   user = col_character(),
##   type = col_character(),
##   reservePriceETH = col_double(),
##   reservePriceUSD = col_double(),
##   amountUSD = col_double(),
##   user_alias = col_character(),
##   onBehalfOf_alias = col_character(),
##   datetime = col_datetime(format = "")
## )
## i Use `spec()` for the full column specifications.

## Warning: 92268 parsing failures.
##    row               col           expected                                     actual
## 180307 collateralAmount  1/0/T/F/TRUE/FALSE 0.3308551927545562                         '~/Blockchain,
## 180307 collateralReserve 1/0/T/F/TRUE/FALSE WETH                                       '~/Blockchain,
## 180307 liquidator        1/0/T/F/TRUE/FALSE 0x0c9d28f3d6a076484a357ad75a6a3b4df71c3f87 '~/Blockchain,
## 180307 principalAmount   1/0/T/F/TRUE/FALSE 639.17                                     '~/Blockchain,
## 180307 principalReserve  1/0/T/F/TRUE/FALSE GUSD                                       '~/Blockchain,
## ...... ................. .................. .......................................... ............
## See problems(...) for more details.
```

```r
#remove protocol smart contracts
df<-filter(df,df$protocolContract==FALSE)

head(df)
```

```
## # A tibble: 6 x 34
##     amount borrowRate borrowRateMode onBehalfOf   pool  reserve timestamp user
##      <dbl>      <dbl> <chr>          <chr>        <chr> <chr>       <dbl> <chr>
## 1   41502.       6.27 Variable       0x94ee9c600~ Main  DAI        1.62e9 0x94e~
## 2 7000000        2.59 Variable       0x51346d389~ Main  USDT       1.62e9 0x513~
## 3   15000        8.80 Variable       0x416d7f382~ Main  USDC       1.62e9 0x416~
## 4    8193.      48.7  Stable         0x78cbc5e9e~ Main  USDC       1.62e9 0x78c~
## 5   11000        3.23 Variable       0xbed4dbd30~ Main  USDT       1.63e9 0xbed~
## 6   40000        5.74 Variable       0x2627ffc9a~ Main  USDT       1.62e9 0x262~
## # ... with 26 more variables: type <chr>, reservePriceETH <dbl>,
## #   reservePriceUSD <dbl>, amountUSD <dbl>, collateralAmount <lgl>,
## #   collateralReserve <lgl>, liquidator <lgl>, principalAmount <lgl>,
## #   principalReserve <lgl>, reservePriceETHPrincipal <lgl>,
## #   reservePriceUSDPrincipal <lgl>, reservePriceETHCollateral <lgl>,
## #   reservePriceUSDCollateral <lgl>, amountUSDPincipal <lgl>,
## #   amountUSDCollateral <lgl>, borrowRateModeFrom <lgl>, ...
```

Group Data by User

Next, we group users by the factors that distiniguish liquidators and non-liquidators. These are the time a user is active, their proportion of transaction types, and number of each type of transaction.

```r
#group by user and get time of user's first and last transaction, as well as number of transactions
df.users<- df%>%group_by(user)%>%
```

```r
  summarise(timefirst=min(timestamp), timelast=max(timestamp), N=n())

#get the time the user has been active
df.users$timeactive<-df.users$timelast-df.users$timefirst

#get user's transaction information
for(Type in c(unique(df$type))){
  #filter for only transactions of certain type
  df.type <-filter(df%>%group_by(user)%>%
                     count(type),type==Type)

  #add counts of transaction types to df
  ntypes<-paste("logn_",Type,sep='')
  colnames(df.type)[3]<-ntypes
  df.type<-df.type%>%replace(is.na(.),0)
  df.type[ntypes]<-log(df.type[ntypes]+1)

  df.users<-merge(x=df.users,y=select(df.type,user,ntypes),by="user",all.x=TRUE)


  #get proportion of transaction types and weekly number of transaction type
  df.users[paste("p_",Type,sep='')]<-(df.users[ntypes])/((df.users$N))
}
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(ntypes)` instead of `ntypes` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```r
head(df.users)
```

```
##                                          user  timefirst   timelast  N
## 1 0x0000000000000000000000000000000000000001 1626953591 1626953591  1
## 2 0x000000000000000000000000000000000000dead 1613977574 1626148294  2
## 3 0x0000000000007f150bd6f54c40a34d7c3d5e9f56 1613241549 1633818799 60
## 4 0x00000000000cd56832ce5dfbcbff02e7ec639bc9 1622302305 1622568495 10
## 5 0x00000000005dbcb0d0513fcda746382fe8a53468 1619325602 1619328421  4
## 6 0x00000000009a41862f3b2b0c688b7c0d1940511e 1607151100 1608903952  4
##   timeactive logn_borrow  p_borrow logn_repay   p_repay logn_liquidation
## 1          0          NA        NA         NA        NA               NA
## 2   12170720          NA        NA         NA        NA               NA
## 3   20577250          NA        NA         NA        NA               NA
## 4     266190    1.098612 0.1098612   1.098612 0.1098612               NA
## 5       2819          NA        NA         NA        NA               NA
## 6    1752852          NA        NA         NA        NA               NA
##   p_liquidation logn_deposit  p_deposit logn_redeem   p_redeem logn_swap p_swap
## 1            NA           NA         NA          NA         NA        NA     NA
## 2            NA           NA         NA          NA         NA        NA     NA
## 3            NA    3.1354942 0.05225824   3.4965076 0.05827513        NA     NA
## 4            NA    1.0986123 0.10986123   1.0986123 0.10986123        NA     NA
## 5            NA    0.6931472 0.17328680   0.6931472 0.17328680        NA     NA
## 6            NA    0.6931472 0.17328680   0.6931472 0.17328680        NA     NA
##   logn_collateral p_collateral
## 1       0.6931472   0.69314718
## 2       1.0986123   0.54930614
```

```
## 3        1.9459101   0.03243184
## 4        1.0986123   0.10986123
## 5        1.0986123   0.27465307
## 6        1.0986123   0.27465307
```

We clean the user data by replacing NaNs with 0s, removing unnecessary columns, and scaling the data.

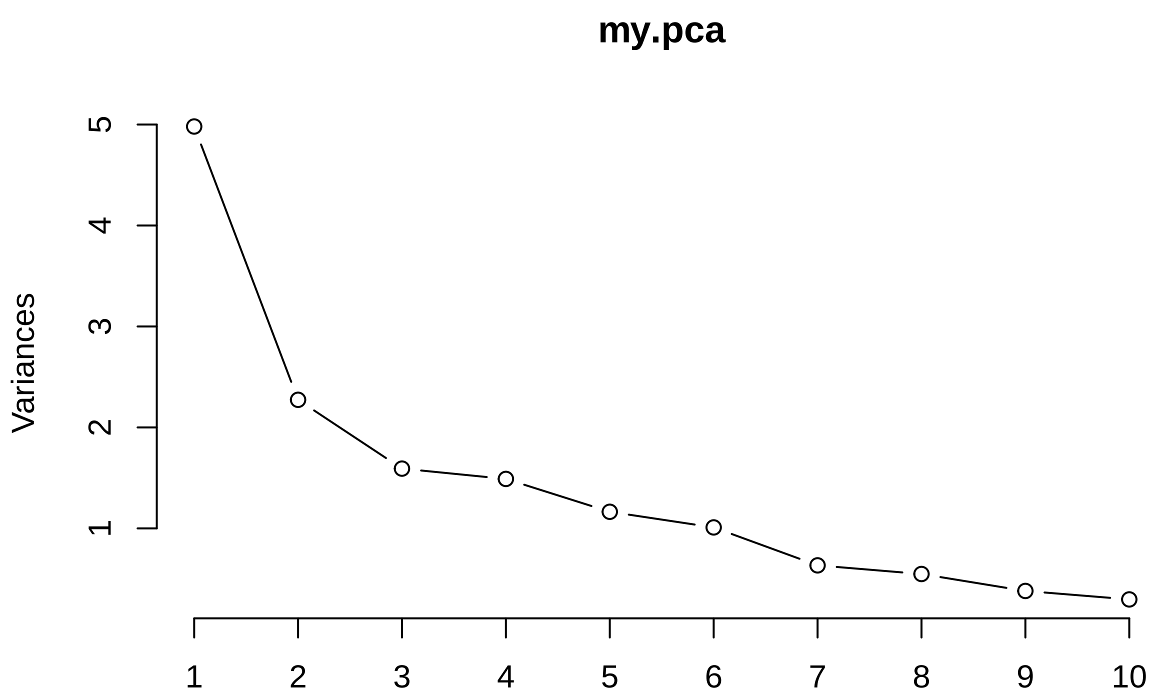
```
#replace missing values as 0's
df.noNans<-df.users%>%replace(is.na(.),0)

#drop columns
df.sub<-select(df.noNans,-c(user,timefirst,timelast,N))

#scale data
df.scaled<-df.sub%>%mutate_all(scale)

head(df.scaled)
```

```
##   timeactive logn_borrow   p_borrow logn_repay    p_repay logn_liquidation
## 1 -0.6607034  -0.5948092 -0.5767928 -0.5326785 -0.5233319       -0.1781988
## 2  1.5792965  -0.5948092 -0.5767928 -0.5326785 -0.5233319       -0.1781988
## 3  3.1265038  -0.5948092 -0.5767928 -0.5326785 -0.5233319       -0.1781988
## 4 -0.6117116   0.7550176  0.8194631  1.0118444  1.4103249       -0.1781988
## 5 -0.6601846  -0.5948092 -0.5767928 -0.5326785 -0.5233319       -0.1781988
## 6 -0.3380940  -0.5948092 -0.5767928 -0.5326785 -0.5233319       -0.1781988
##   p_liquidation logn_deposit  p_deposit logn_redeem    p_redeem  logn_swap
## 1     -0.152345   -1.1361043 -1.1876000  -0.7831018 -0.84366455 -0.1475553
## 2     -0.152345   -1.1361043 -1.1876000  -0.7831018 -0.84366455 -0.1475553
## 3     -0.152345    3.1486264 -0.7510665   4.3350964 -0.07661018 -0.1475553
## 4     -0.152345    0.3651766 -0.2698862   0.8250502  0.60239900 -0.1475553
## 5     -0.152345   -0.1889015  0.2599323   0.2315291  1.43724676 -0.1475553
## 6     -0.152345   -0.1889015  0.2599323   0.2315291  1.43724676 -0.1475553
##      p_swap logn_collateral p_collateral
## 1 -0.119591      -0.8222386  2.355551183
## 2 -0.119591      -0.1007502  1.543169443
## 3 -0.119591       1.4069395 -1.376020111
## 4 -0.119591      -0.1007502 -0.938716358
## 5 -0.119591      -0.1007502 -0.008009183
## 6 -0.119591      -0.1007502 -0.008009183
```

Principal Component Analysis

Next, we perform principal component analysis, and create an eblow chart to show the explained variance. We can see a clear elbow at 3 components.

```
#perform pca on data
my.pca<-prcomp(df.scaled,retx=TRUE,center=FALSE,scale=FALSE) # Run PCA and save to my.pca

#make scree plot
plot(my.pca, type="line")
```

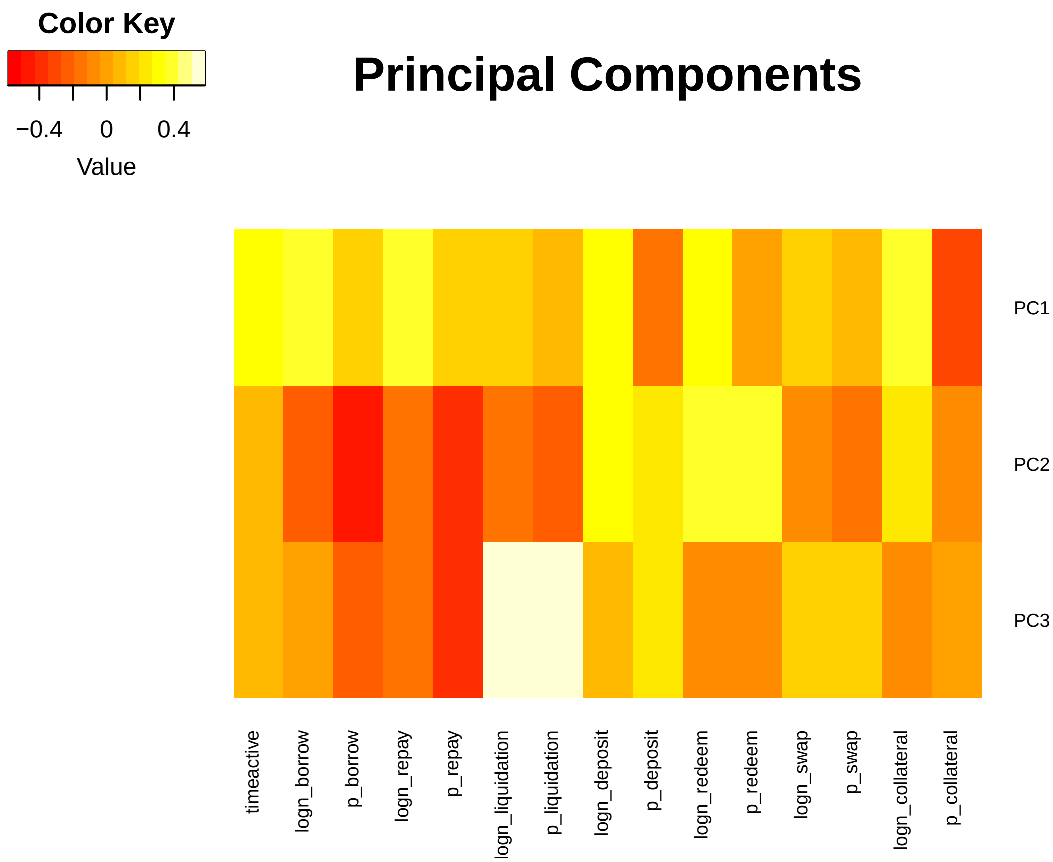



```r
#summary of PCA
summary(my.pca)
```

```
## Importance of components:
##                          PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.232 1.5077 1.2616 1.22050 1.07895 1.00464 0.79592
## Proportion of Variance 0.332 0.1515 0.1061 0.09931 0.07761 0.06729 0.04223
## Cumulative Proportion  0.332 0.4836 0.5897 0.68899 0.76660 0.83389 0.87612
##                            PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     0.74016 0.61644 0.54435 0.50409 0.41703 0.33886 0.22398
## Proportion of Variance 0.03652 0.02533 0.01975 0.01694 0.01159 0.00766 0.00334
## Cumulative Proportion  0.91264 0.93798 0.95773 0.97467 0.98626 0.99392 0.99726
##                           PC15
## Standard deviation     0.20260
## Proportion of Variance 0.00274
## Cumulative Proportion  1.00000
```

We select 3 principal components. Then, we create a heatmap showing the makeups of these components. PC3 seems to be the component that separates liquidations.

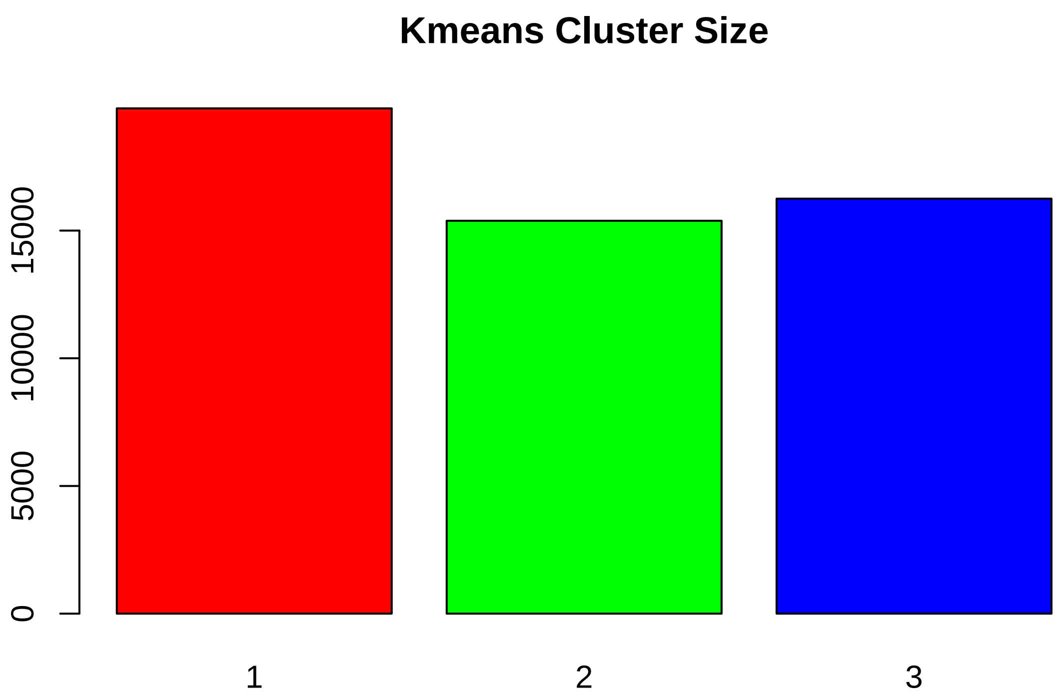
```r
#select 4 components
ncomps=3

#make heatmap for PCs
V <- t(my.pca$rotation[,1:ncomps]) # We transpose to make the principal components be rows
heatmap.2(V, main='Principal Components', cexRow=0.75, cexCol=0.75, scale="none", dendrogram="none",
Colv= FALSE, Rowv=FALSE, tracecol=NA,density.info='none')
```



Clustering

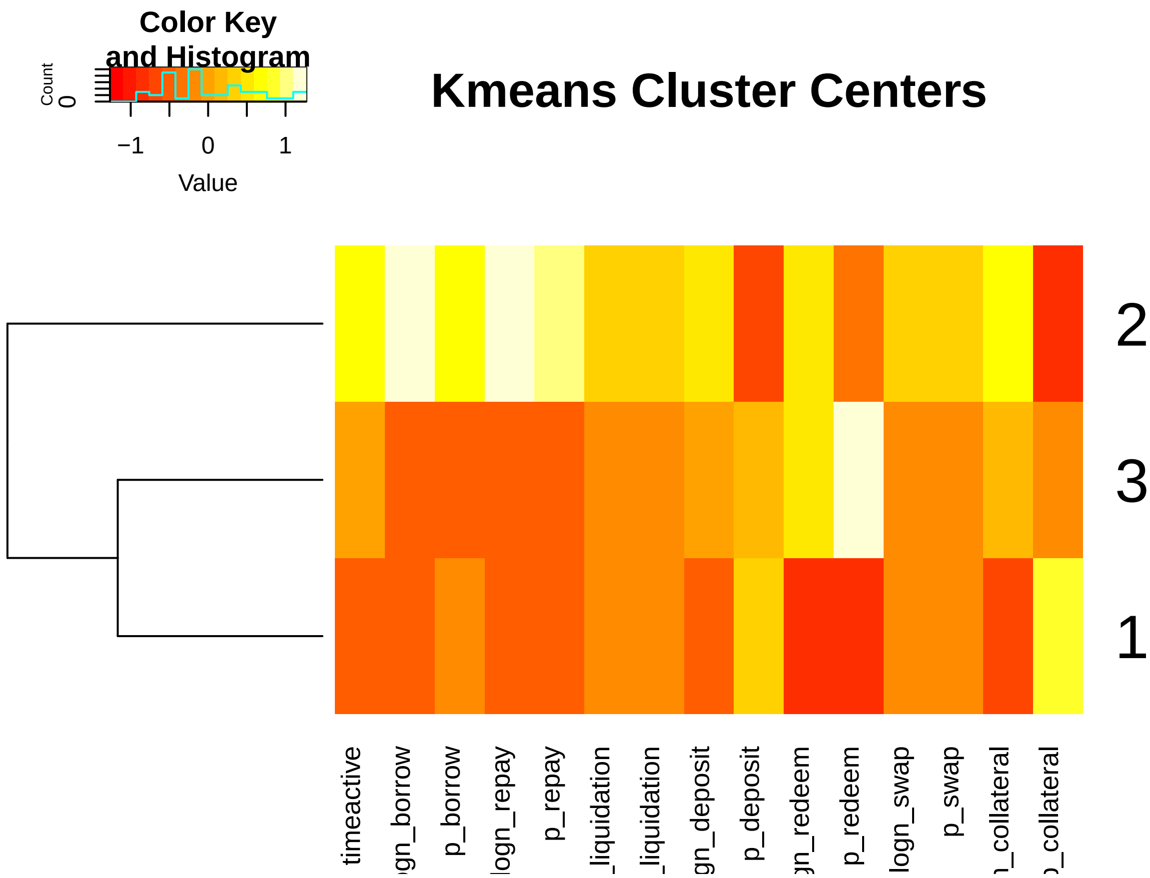
Finally, we perform kmeans clustering on the data. We select 3 clusters for the 3 principal components. The cluster sizes are roughly even.

```
#run k-means algorithm
pca.matrix<-my.pca$x[,1:ncomps]
set.seed(1)
km <-kmeans(df.scaled,ncomps)

#assign cluster column to Data Frame
df.scaled$cluster<-km$cluster

#plot frequencies of each cluster
barplot(table(km$cluster),main="Kmeans Cluster Size",col=c('red','green','blue'))
```



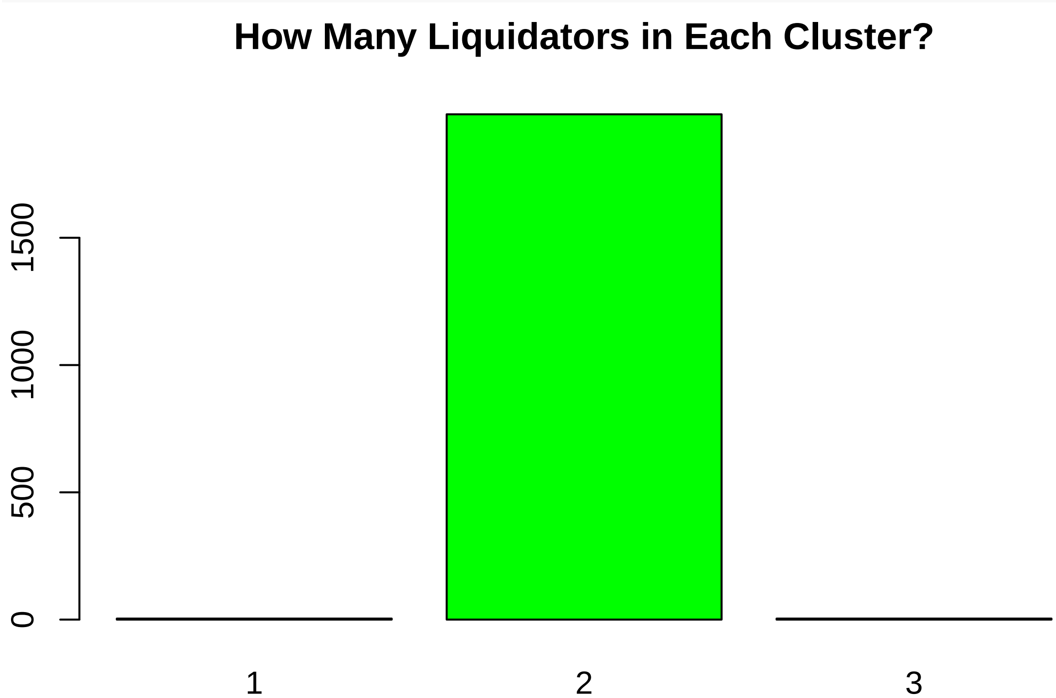
We analyze the clusters by making a heatmap of their means. Cluster 2 is the cluster that has liquidators. When trying to find patterns of behavior that is similar to liquidators, we can look to the means in cluster 2. The most important factors are a low proportion of collateral changes and deposits, and a large number of borrows and repays.

```
#make heatmap of cluster centers
heatmap.2(km$centers,
scale = "none",
dendrogram = "row",
Colv=FALSE,
cexCol=1.0,
main = "Kmeans Cluster Centers",
trace ="none")
```



Next, we observe that liquidators are almost exclusive to this cluster. Almost all liquidations occur within cluster 2.

```
#make barplot of liquidators for each cluster
df.liquidators<-filter(df.scaled, df.scaled$logn_liquidation>0)
barplot(table(df.liquidators$cluster),main='How Many Liquidators in Each Cluster?',col=c('red','green',
```



To conclude, we

create a biplot of the different clusters. We can see that cluster 2 users have a much larger spread than the other clusters.

```
#make biplot for clusters
plot1<-ggbiplot(my.pca,choices=c(1,2),
  labels=rownames(df.scaled), #show point labels
  var.axes=TRUE, # Display axes
  ellipse = FALSE, # Don't display ellipse
  obs.scale=1,
  groups=as.factor(km$cluster)) +
  ggtitle("User Data Projected on PC1 and PC2 ")

plot1
```