MATP-4910 Final Project Notebook Template

DeFi

Christopher Cammilleri

5 December 2021

Contents

Final Project: Github Info

- github repository:
- Your github ID: cammic
- Final notebook: dar_final_cammic_05122021.Rmd
- Summary of github contributions including github issues addressed.
  - Added python notebook where visualizations on liquidatees/non-liquidatess was conducted (liquidations.ipynb)
- All code done on my own

Overview & Problems Tackled

For my research, I investigated users who have received liquidations in Aave. Aave is a Decentralized Finance protocol in which users can deposit and borrow cryptocurrencies. Loans can be taken out by users for as long as they wish, so long as they posses the collateral required to maintain the loan principal and loan interest. When the ratio of amount borrowed to amount collateral gets too large, the loan may be liquidated. In this case, a liquidator may return some of the principal the user failed to repay, in return for a portion of the loan's collateral. My main research goals were as followed:

- Investigate the patterns of behavior that belong to users who have received liquidations
- Examine the connections between liquidators and liquidatees
- Cluster users according to these patterns, and identify clusters with the highest risk

Data Description

The data I used for this project was version 2 of the Aave user transaction data, titled transactionsv2.rds. The data was sourced from the Aave API hosted on https://thegraph.com/en/. I filtered out rows containing transactions made by the Aave protocol. The resulting data contains 685,824 rows and 34 columns. It spans the time period from November 30th, 2020, to October 17th 2021.

Results

Problem 1

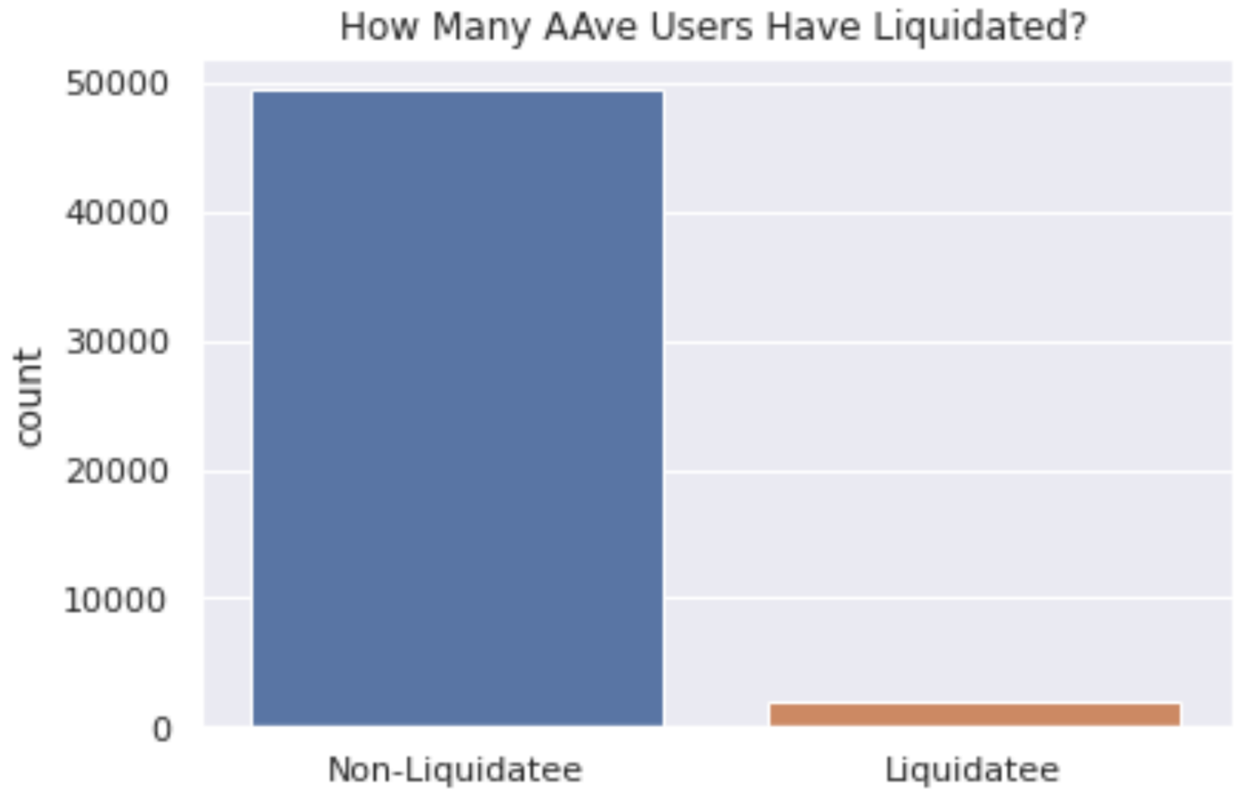
The first problem I set out to investigate was the patterns of behaviors belonging to users who liquidate. By doing this, we can see which factors distinguish the liquidatees from the non-liquidatees. This can help us to classify transactions which are "risky".

Methods

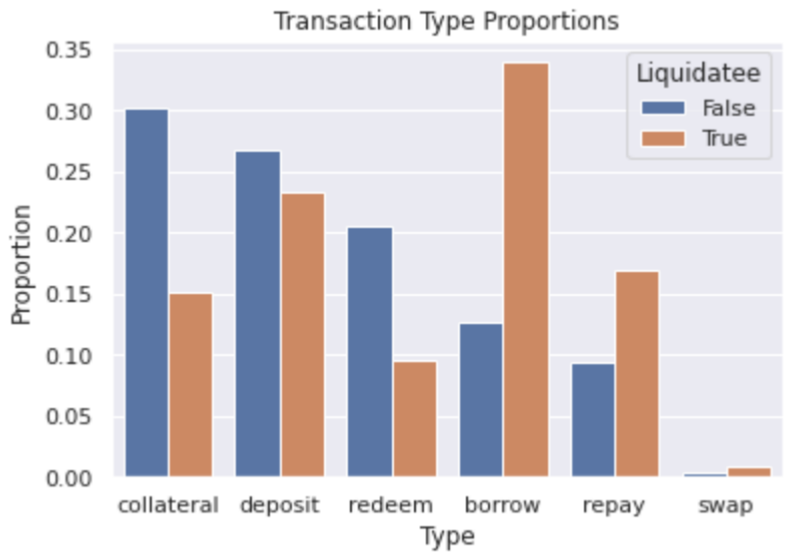
First, I identified whether each transaction belonged to a user who is a liquidatee. Then, I seperated in to the data in to transactions made by liquidatee's, and transactions made by non-liquidatee's. Using this data, I made several visualizations showing differences in transacions between liquidatees and non-liquidatees.

Results

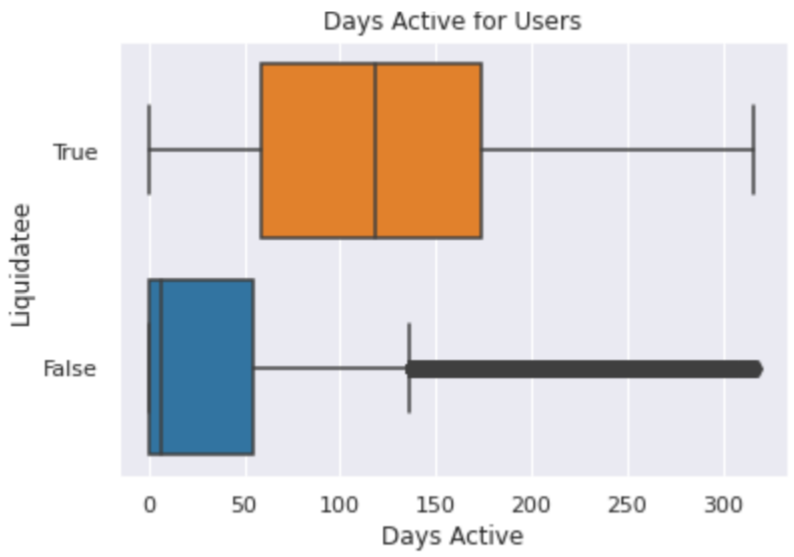
To get a general idea of the number of users who get liquidated at some point, we start by making a bar chart showing the number of liquidatees and non-liquidatees. We can see that only a small number of users will end up receving a liquidation. Of the 51,419 users, only 1,993 are liquidatees.



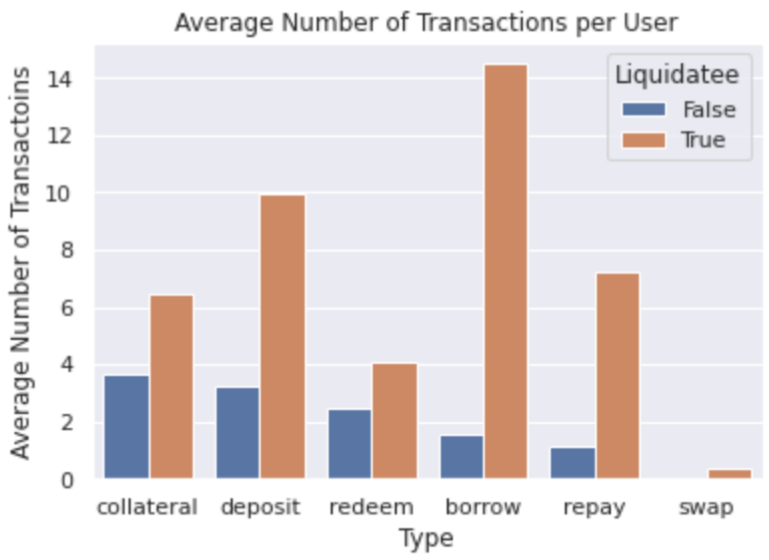
Second, we look at the proportion of transaction types for each type of users. We a bar plot showing the proportion of transaction types for both groups of users. This visualization will give us an idea for the types of transactions that are common to liquidatees. From the chart, we can see that liquidatees have a large proportion of borrows in comparison to non-liquidatees, and a small proportion of redeems. This makes sense, as a liquidation happens when the amount a user borrows gets too large without being repaid.



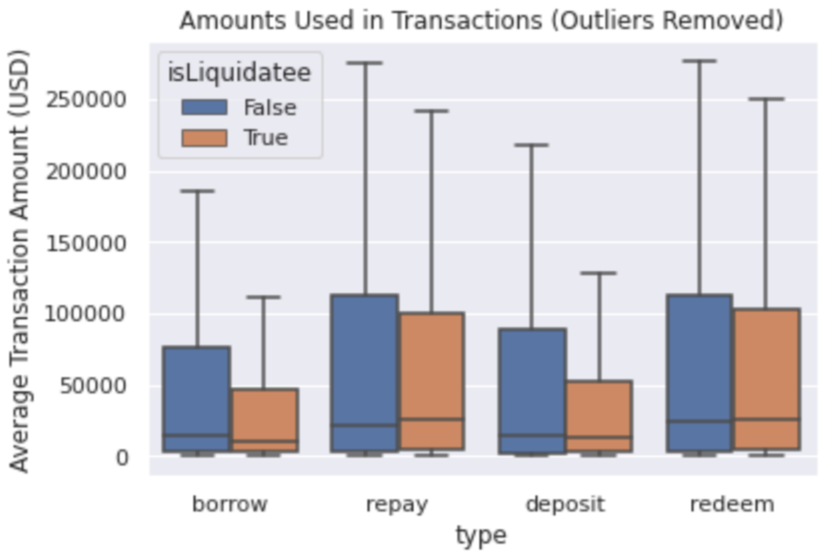
To follow, we investigate the number of days users spend active. To do this, we take the difference in time (in days) between a user's first transaction and a user's last transaction. We plot this data in boxplots, separating liquidatees and non-liquidatees. We see that liquidatees spend a significantly more amount of time active than other users.



Next, we examine the average number of transactions per type made by users. We create a bar plot showing the average number of transactions made by both groups of users, for each transaction type. Clearly, liquidatees make significantly more transactions across their activity period, for all types.



Finally, we look at the amount of money used in the different transaction types for both groups of users. We make boxplots for each transaction type for both groups showing the amounts of money used in the transactions, in USD. We see no significant difference in the median amount of money used.



Discussion

From these findings, we can see that there are three main ways to distinguish liquidatees form non-liquidatees - time active, transaction type proportions, and average number of transactions. A lot of the distinguishing factors make sense. For instance, lquidatees tend to be active linger. This intuitively makes sense, as the longer someone is active, the more likely they are to receive a liquidation. This same principal applies to the average number of transaction. The other factor, transaction amount, did not seem to have much of a significant difference between groups of users.

Problem 2

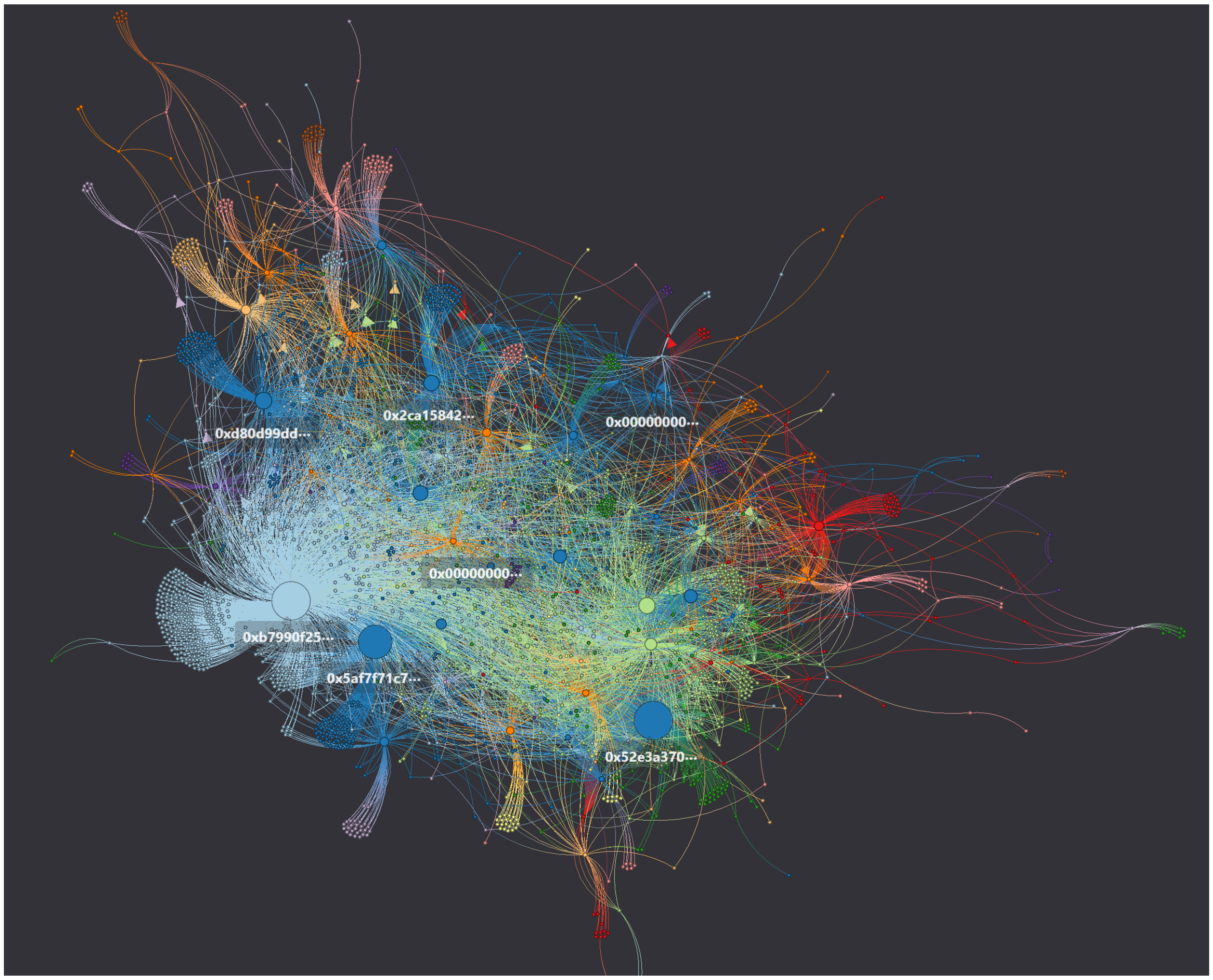
The next task that I wanted to investigate was the connections between liquidatees and liquidators. In Aave, a liquidation on a user is initiated by a liquidator, who hopes to receieve a profit from completing the transaction. In this section I examine a graph that connects liquidators with liquidatees. There are 193 liquidators represented in the data.

Methods

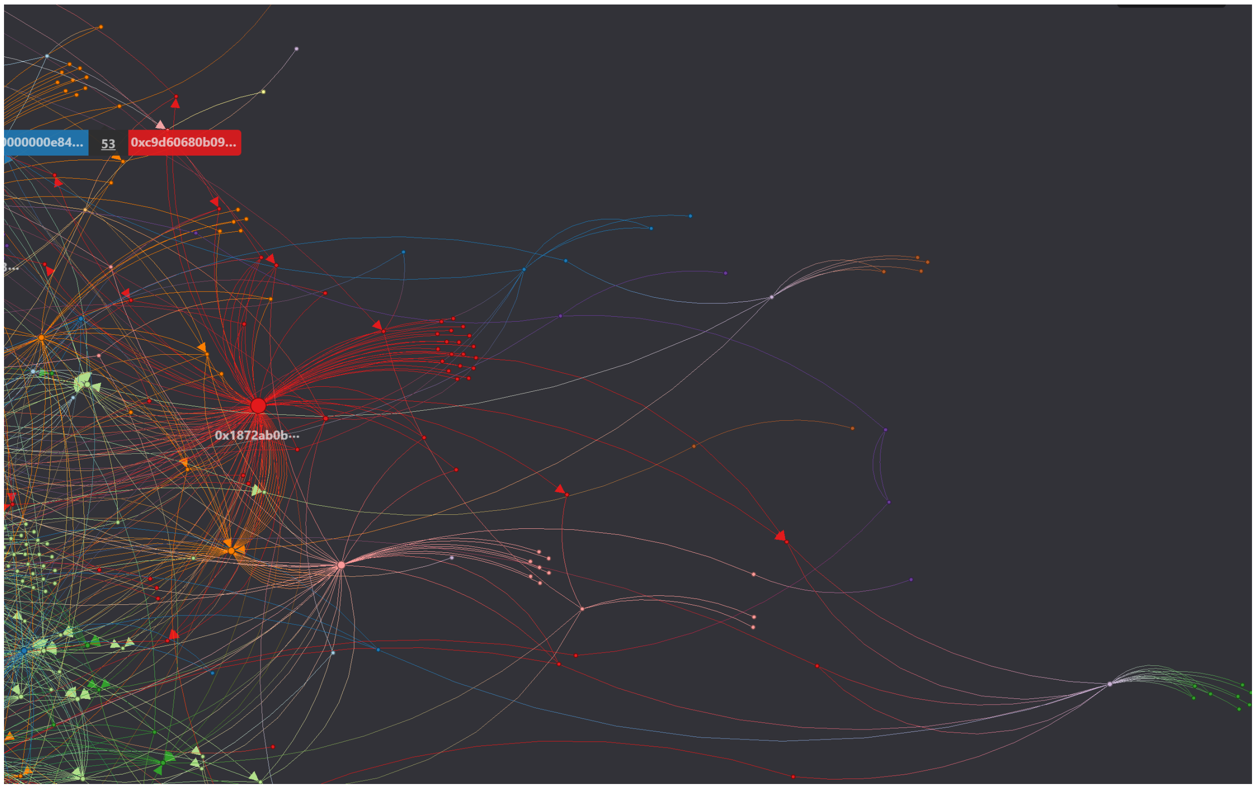
We create a graph using the graphistry package in python. The nodes in this graph are liquidators and liquidatees. The nodes are connected by liquidation transactions from liquidators to liquidatees.

Results

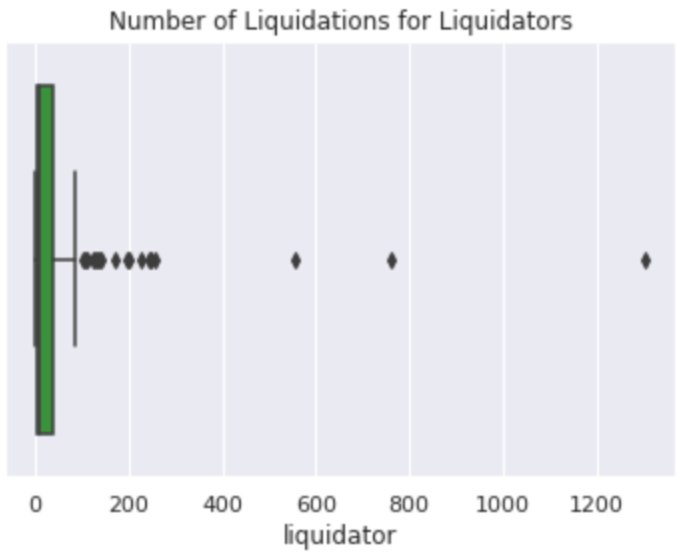
Below, is a zoomed out image of the graph. From this view, we can't observe much other than that there are many nodes and many connections.



Next, is a zoomed in image of the graph. Here, we can more closely see the connections between liquidators and liquidatees. We can see that some liquidators liquidate more than others, and some users receieve more than one liquidations from the same liquidator.



In addition, we examine the in-degree and out-degree of the graph. The out degree represents the number of liquidations liquidators execute. We can see that most liquidators only make about one liquidation - however, some of the largest liquidators have hundreds.



Here, we have the in-degree graph. This represents the number of liquidations liquidatees receive. The median is above one, which indicate that users who receive liquidations are likely to repeat this behavior.

Discussion

From the above analysis, we can observe that liquidatees are likely to have multiple liquidations over their lifetime. This could indicate that past liquidations are a good predictor of liquidations in the future. In addition, we can see that a small group liquidators represent the majority of liquidations.

Problem 3

The final task I set out to accomplish was to cluster users in a way that could classify them according to their liquidation risk.

Methods

The code for this task is provided below. We begin by importing the necessary libraries.

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

```
## Loading required package: plyr

## Loading required package: scales

## Loading required package: grid
```

```r
library(gplots)
```

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

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

```r
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()
```

```r
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
```

We load 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.
```

```
#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>, ...
```

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

```
#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)%>%
  dplyr::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)%>%
                     dplyr::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.
```

```
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
```

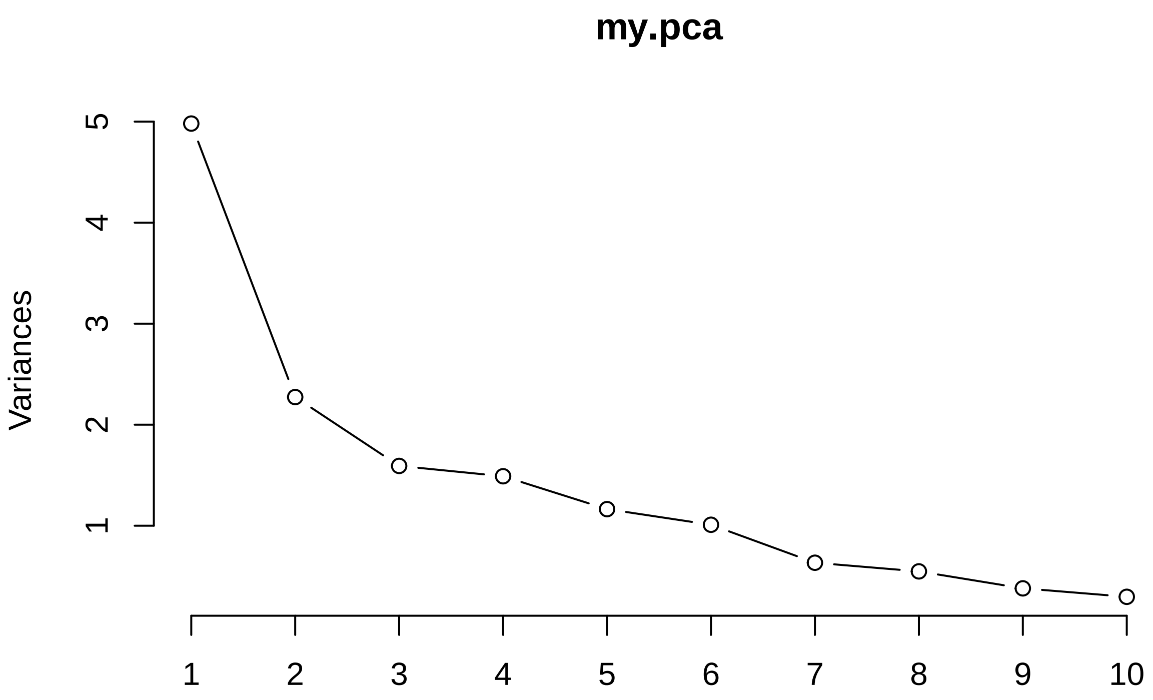
We clean the user data by replacing NaNs with 0s, removing unnecessary columns, and scaling the data.

```
#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
```

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")
```



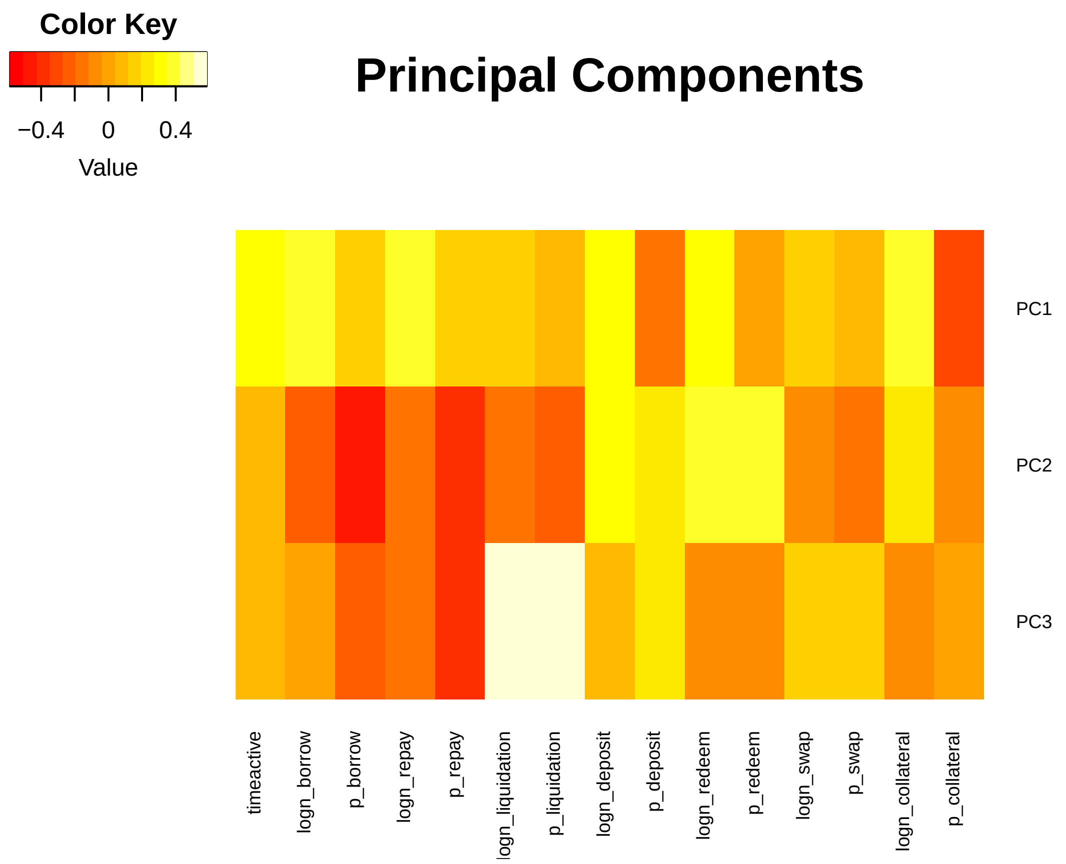
```
#summary of PCA
summary(my.pca)
```

```
## 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.

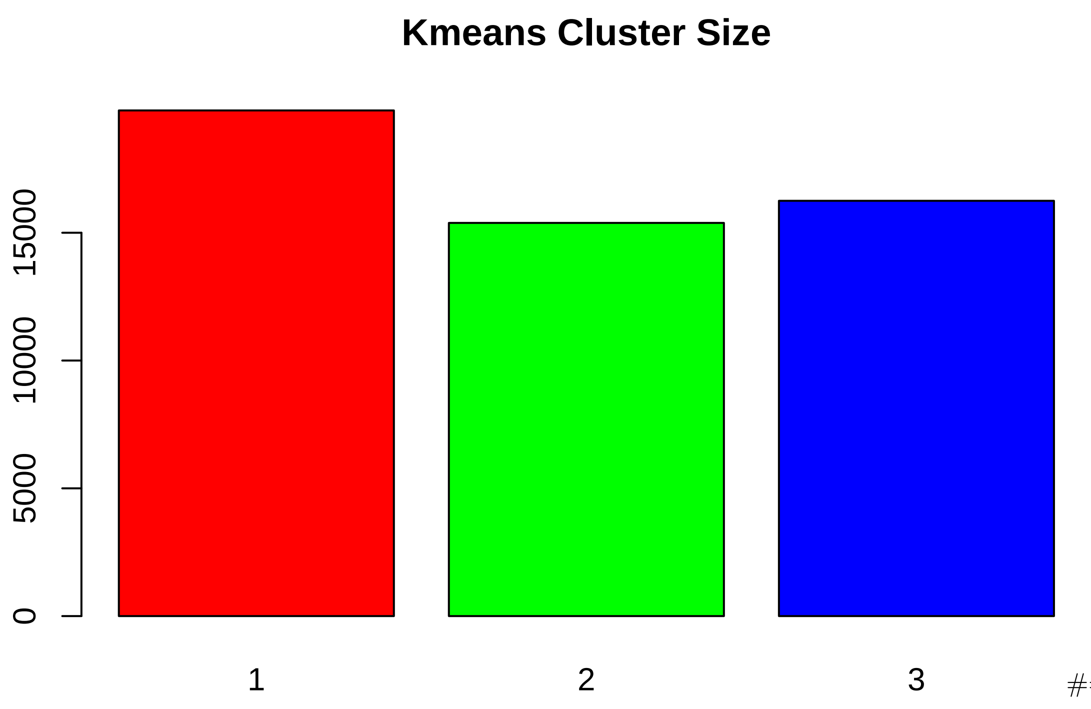
```
#select 4 components
ncomps=3
```

```r
#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')
```



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

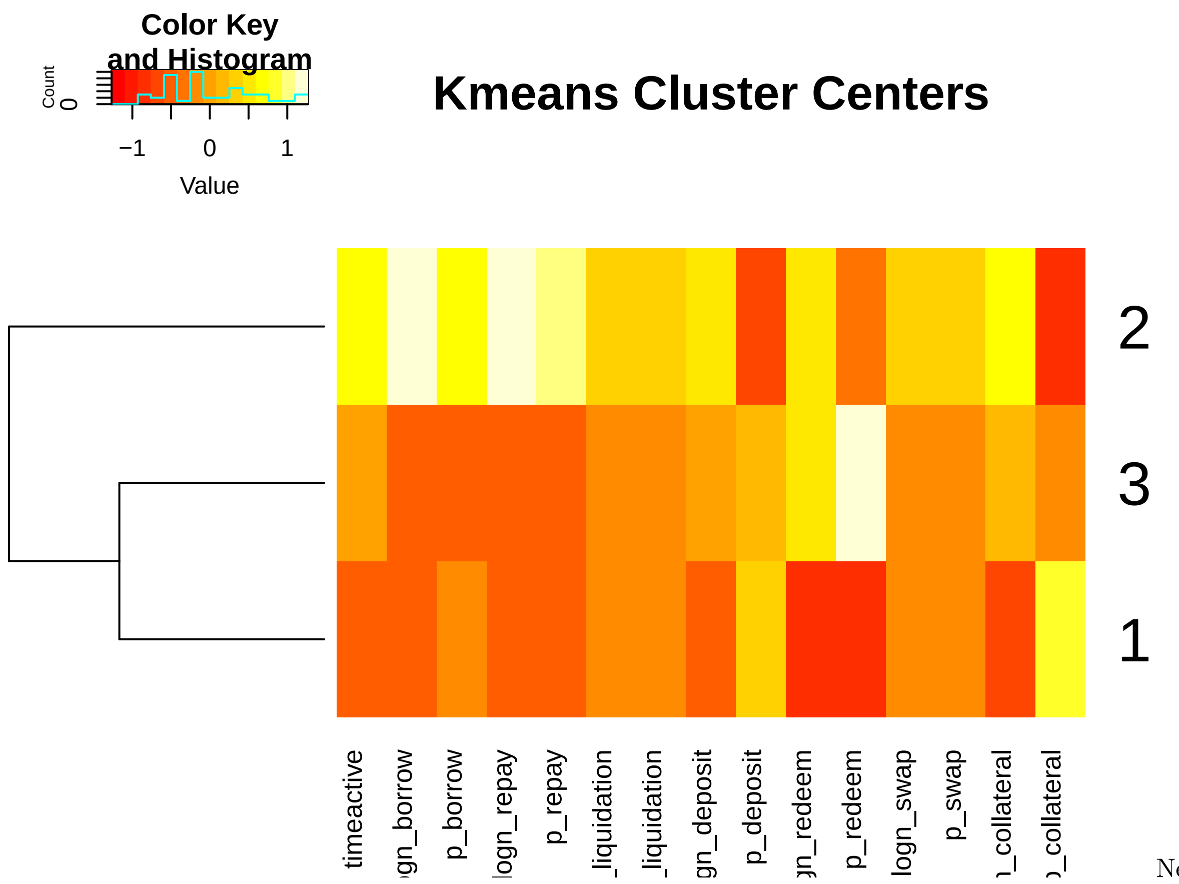
```r
#run k-means algorithm
pca.matrix<-my.pca$x[,1:ncomps]
set.seed(1)
km <-kmeans(df.scaled,ncomps)

#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'))
```



### Results

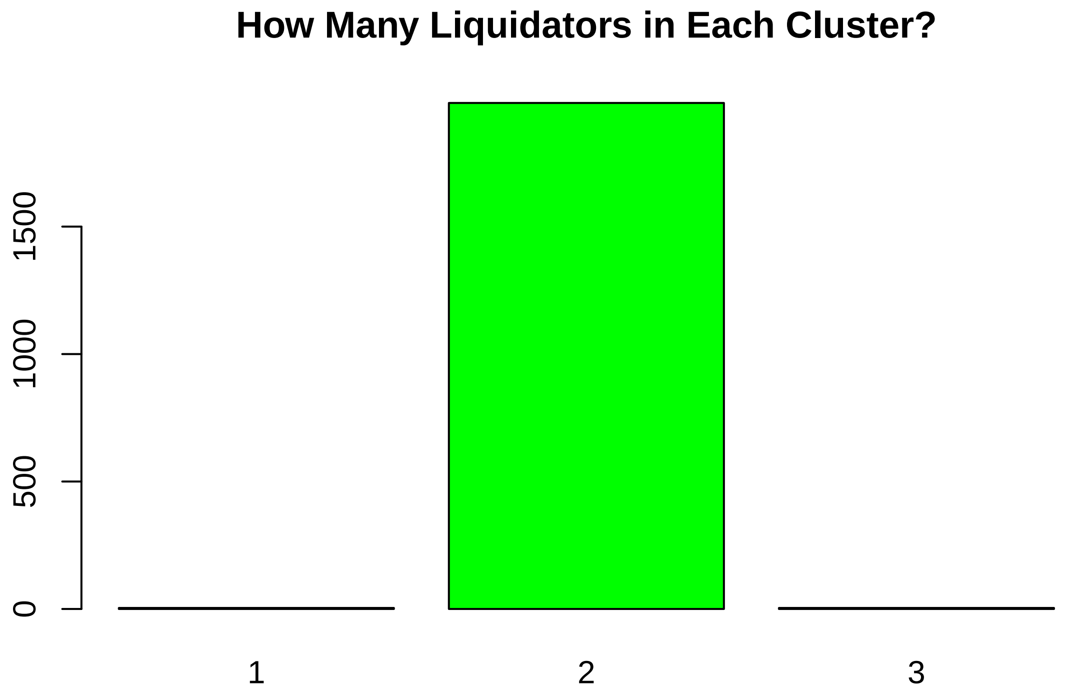
We analyze the clusters by making a heatmap of their means. Cluster 2 is the cluster that has liquidators. When trying to find patterns of behavior that is similar to liquidators, we can look to the means in cluster 2. The most important factors are a low proportion of collateral changes and deposits,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

```r
#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.

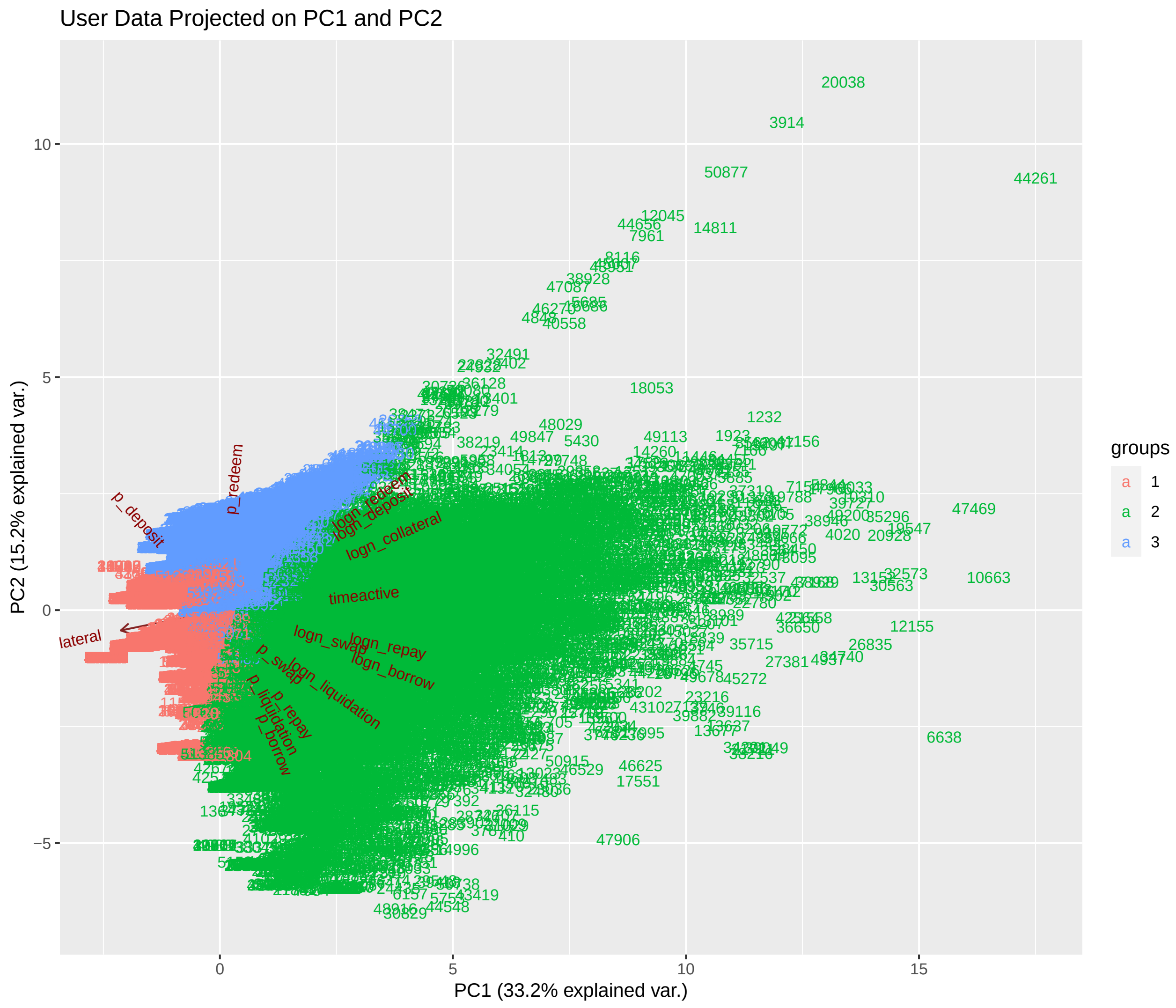
```r
#make barplot of liquidators for each cluster
df.liquidators<-filter(df.scaled, df.scaled$logn_liquidation>0)
barplot(table(df.liquidators$cluster),main='How Many Liquidators in Each Cluster?',col=c('red','green',
```



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
```



### Discussion

The clustering we conducted yields three main clusters. Cluster 2 is our high risk group as it includes almost all of the liquidatees. To see what patterns of behavior are common to liquidatees, we can read across cluster 2's rows in the heatmap. We can see that the most distinguishing behaviors are a high number of borrows and redeems, and low proportions of deposits and collateral changes. Clusters 3 and 1 are our low risk groups. These clusters have much fewer transactions than cluster 2, and have higher proportions of deposits and lower proportions of borrows. The main factor that separates these two clusters is that cluster 3 has a large number and proporiton of redeems compared to cluster 1. This indicates that cluster 3 are users who have taken their money out of the protocol during their lifetime, whereas cluster 1 are users who let their money sit.

Summary and Recommendations

Throughout this research, we were able to discover the pattern of behaviors common to users who receive liquidations. We have found that a high number of borrows and repays, along with a low proportion of deposits and collateral changes places a user in a high risk group for liquidations. It is important to note that these behaviors are correlative rather than predictive. The next step would be to use these findings to predict whether or not a user would liquidate within a certain time frame. In addition to the work shown above, I have been working on an algorithm to convert the user transaction data, in to a data set that would provide the account balances for all users each day. Using this data we can calculate feature that would be relevant to predicting if a user will receive a liquidation, such as health factor. In the future, I plan on training a machine learning model to do just this. In addition in being able to make predictions, the model would also be able to show which features have high predictive capability in predicting liquidations. If the model were to be successful, these findings could be further applied to build a smart contract to perform liquidations on Aave. The liquidator who can fastest liquidate an unhealthy user is the one to receive the liquidation bonus, so being able to predict potential users to liquidate before they occur would be of high value.