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)
  - Added python notebook where user daily account balance data was calculated (DailyUserData.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. A liquidation is when a user is forcibly required to give up their collateral to cover for the principal they failed to repay. 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
- Predict whether a user will receive a liquidation in a certain time frame (work in progress)

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.

There are seven types of transactions represented in the dataset. Below are their descriptions along with their corresponding columns:

1.) Deposit: User deposits tokens into the liquidity pool to receive interest

- type: type of transaction ("redeem" for redeems)
- user: id of user who initiated the transaction
- onBehalfOf: id of user who will redeem the tokens in most cases, user = onBehalfOf
- timestamp: Unix Timestamp of transaction
- pool: id of lending pool from which currency is redeemed
- reserve: symbol of currency used in transaction
- reservePriceETH: price of currency at time of transaction, in Ether
- reservePriceUSD: price of currency at time of transaction, in USD
- amount: number of currency tokens redeemed
- amountUSD: amount being borrowed, in USD reservePriceUSD X amount

2.) Redeem: User removes tokens in the liquidity pool

- type: type of transaction ("redeem" for redeems)
- user: id of user who initiated the transaction
- onBehalfOf: id of user who will redeem the tokens in most cases, user = onBehalfOf
- timestamp: Unix Timestamp of transaction
- pool: id of lending pool from which currency is redeemed
- reserve: symbol of currency used in transaction
- reservePriceETH: price of currency at time of transaction, in Ether
- reservePriceUSD: price of currency at time of transaction, in USD
- amount: number of currency tokens redeemed
- amountUSD: amount being borrowed, in USD reservePriceUSD X amount

3.) Borrow: User borrows tokens from the liquidity pool, which gathers interest that the user will need to pay back

- type: type of transaction ("borrow" for borrows)
- user: id of user who initiated the transaction
- onBehalfOf: id of user who will incur the debt in most cases, user = onBehalfOf

- timestamp: Unix Timestamp of transaction
- pool: id of lending pool from which currency is borrowed
- reserve: symbol of currency used in transaction
- reservePriceETH: price of currency at time of transaction, in Ether
- reservePriceUSD: price of currency at time of transaction, in USD
- amount: number of currency tokens borrowed
- amountUSD: amount being borrowed, in USD reservePriceUSD X amount
- borrowRate: interest rate of loan (APR)
- borrowRateMode: whether the loan has a variable or stable interest rate

4.) Repay: User repays tokens borrowed from the liquidity pool

- type: type of transaction ("borrow" for borrows)
- user: id of user who initiated the transaction
- onBehalfOf: id of user whose borrow will be repaid in most cases, user = onBehalfOf
- timestamp: Unix Timestamp of transaction
- pool: id of lending pool for which currency is being repaid
- reserve: symbol of currency used in transaction
- reservePriceETH: price of currency at time of transaction, in Ether
- reservePriceUSD: price of currency at time of transaction, in USD
- amount: number of currency tokens repaid
- amountUSD: amount being repaid, in USD reservePriceUSD X amount

5.) Swap: User swaps the interest rate mode (stable, variable) of one of their loans

- type: type of transaction ("swap" for swaps)
- user: id of user who initiated the transaction
- timestamp: Unix Timestamp of transaction
- pool: id of lending pool from which borrowed currency is being swapped
- reserve: symbol of currency used in transaction
- borrowRateModeFrom: interest rade mode swapping from
- borrowRateModeTo: interest rate mode swapping to
- stableBorrowRate: the stabe interest rate for the deposit
- variableBorrowRate: the variable floating rate for the deposit

6.) Liquidation: If user no longer possesses the collateral to maintain a borrow, the loan will be liquidated, forcing the user to pay back their collateral to cover for the principal they failed to repay

- type: type of transaction ("liquidation" for liquidations)
- user: id of user who received liquidation
- timestamp: Unix Timestamp of transaction
- pool: id of lending pool from which loan is being liquidated

- collateralReserve: symbol of currency used as collateral in loan (what is being repaid)
- principalReserve: symbol of currency which has been borrowed
- reservePriceETHCollateral: price of collateral currency at time of transaction, in Ether
- reservePriceUSDCollateral: price of collateral currency at time of transaction, in USD
- reservePriceETHPrincipal: price of principal currencyat time of transaction, in Ether
- reservePriceUSDPrincipal: price of principal currency at time of transaction, in USD
- collateralAmount: number of currency tokens repaid
- principalAmount: number of currency tokens borrowed for loan
- amountUSDPrincipal: amount of principal being borrowed, in USD reservePriceUSDPrincipal X principalAmount
- amountUSDCollateral: amount being of collateral being repaid, in USD reservePriceUSDCollateral X collateralAmount

7.) Collateral Change: User specifies a token they have deposited to be used as collateral for borrows

- type: type of transaction ("collateral" for collateral change)
- user: id of user who received liquidation
- timestamp: Unix Timestamp of transaction
- pool: id of lending pool from which collateral is being changed
- reserve: symbol of currency used in transaction
- fromState: true or false, representing if collateral is changing from specified reserve
- toState: true or false, representing if collateral is changing to specified reserve

Results

Problem 1

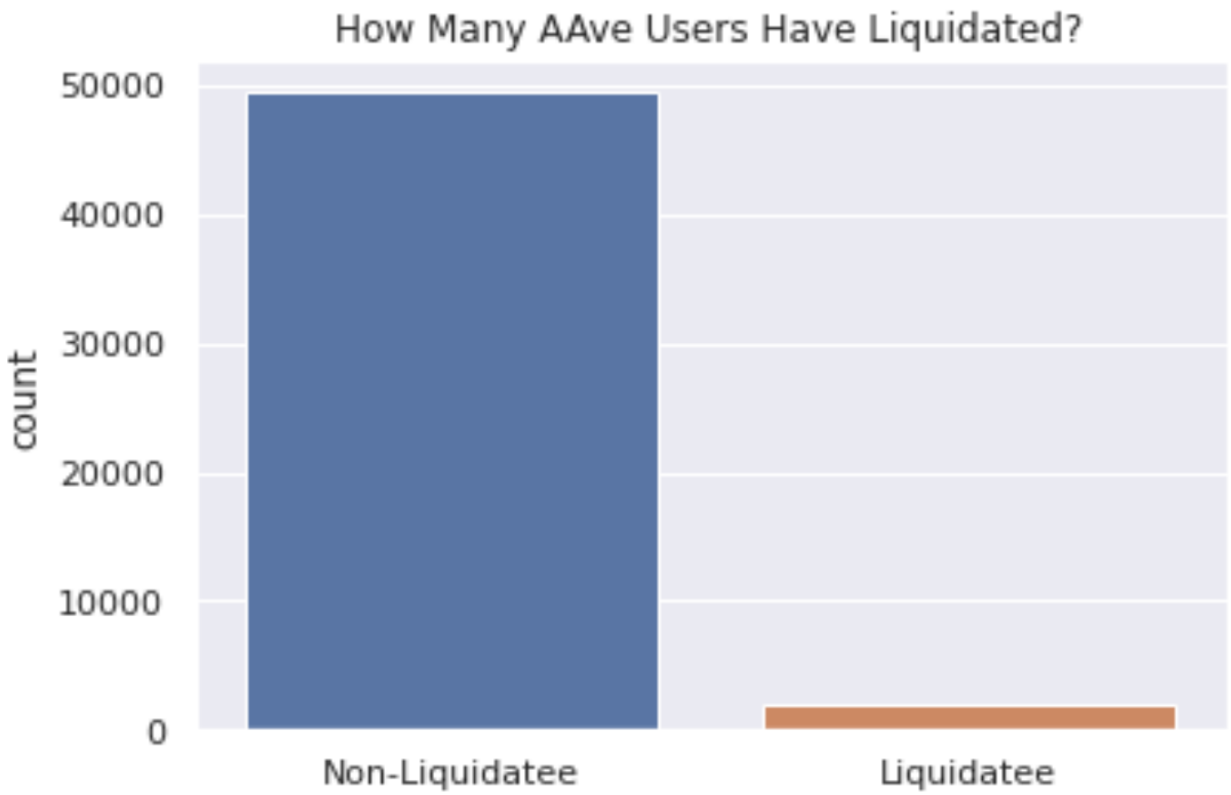
The first problem I set out to investigate was the patterns of behaviors belonging to users who liquidate. By doing this, we can see which factors distinguish the liquidatees from the non-liquidatees. A liquidatee is defined as a user who has received a liquidation at some point during their time active. This can help us to classify transactions which are "risky".

Methods

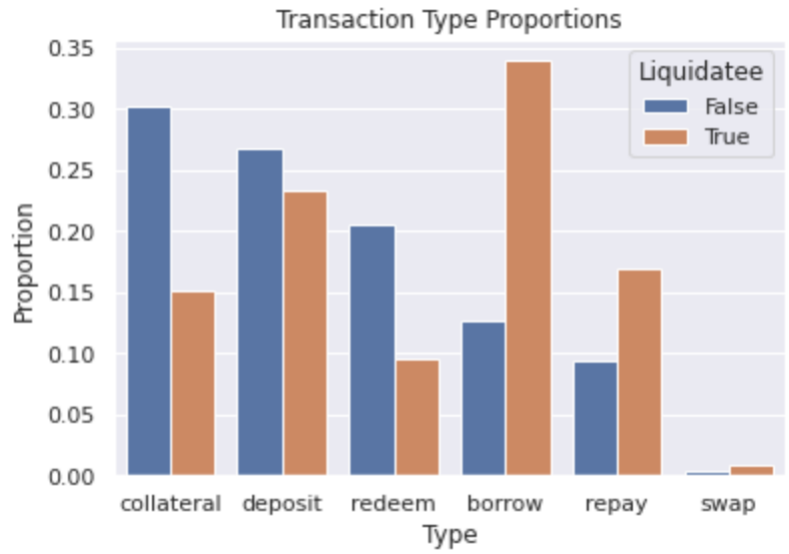
First, I identified whether each transaction belonged to a user who is a liquidatee. Then, I seperated in to the data in to transactions made by liquidatee's, and transactions made by non-liquidatee's. Using this data, I made several visualizations showing differences in transacions between liquidatees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

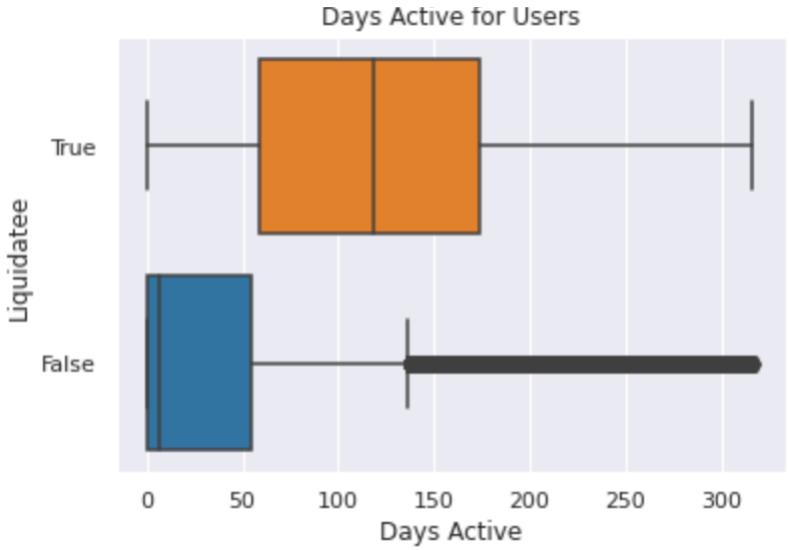


Second, we look at the proportion of transaction types for each type of users. We a bar plot showing the proportion of transaction types for both groups of users. This visualization will give us an idea for the types of transactions that are common to liquidatees. From the chart, we can see that liquidatees 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.

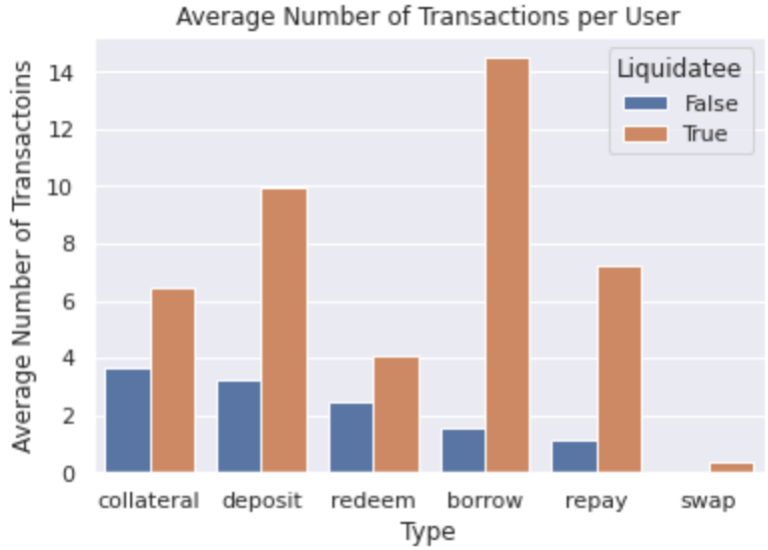


To follow, we investigate the number of days users spend active. To do this, we take the difference in time (in days) between a user's first transaction and a user's last transaction. We plot this data in boxplots, separating liquidatees and non-liquidatees. We see that liquidatees spend a significantly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

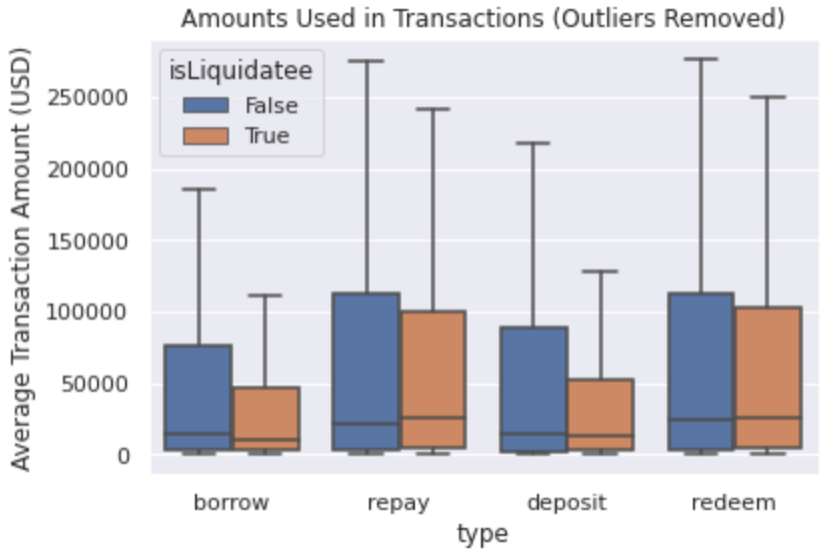
active than other users.



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

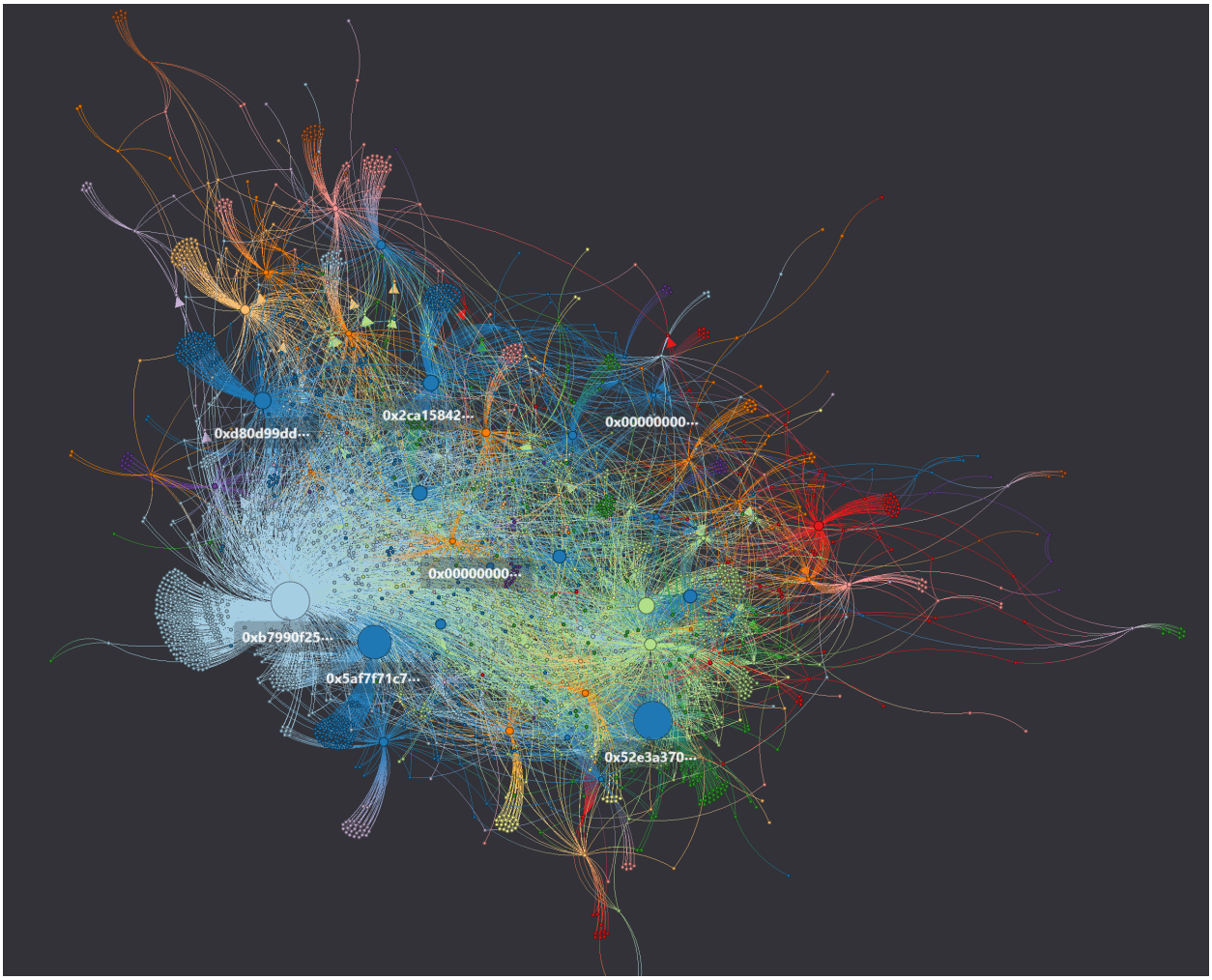
The next task that I wanted to investigate was the connections between liquidatees and liquidators. In Aave, a liquidation on a user is initiated by a liquidator, who hopes to receieve a profit from completing the transaction. In this section I examine a graph that connects liquidators with liquidatees. There are 193 liquidators represented in the data, and 2,258 liquidatees.

Methods

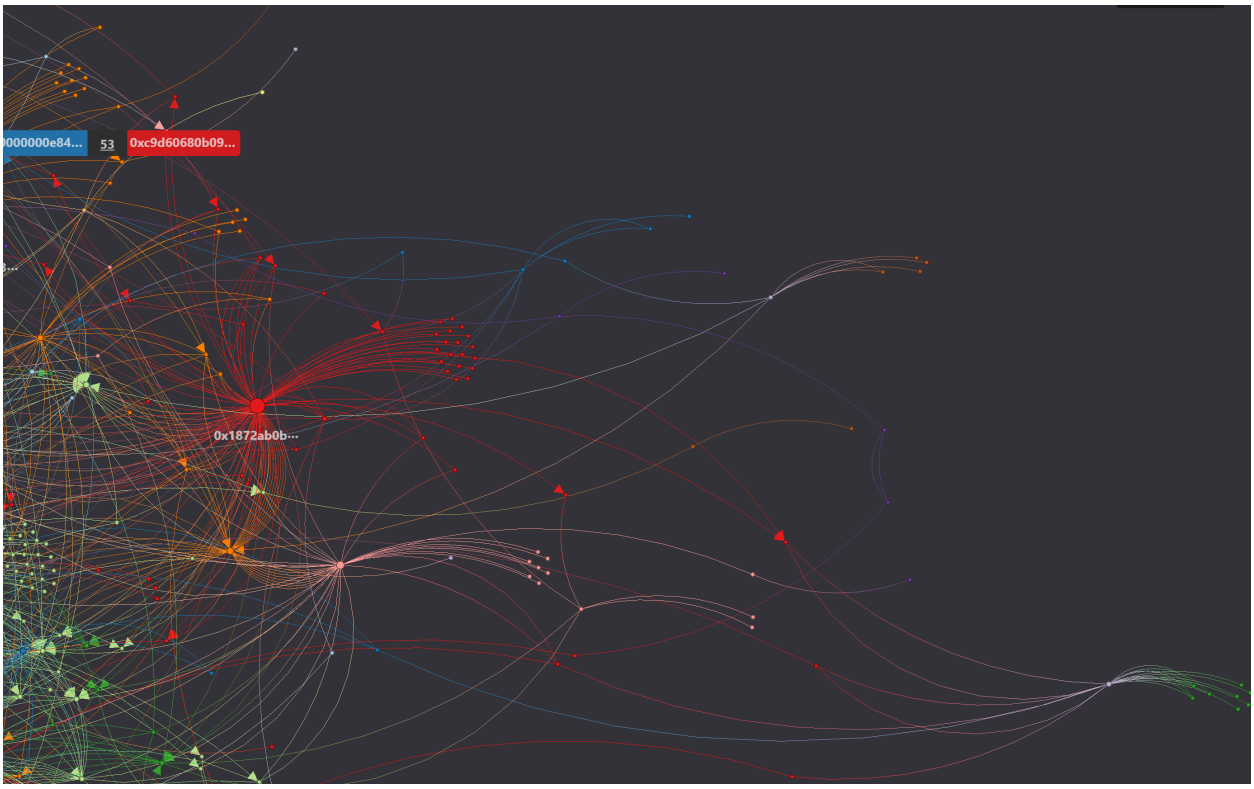
We create a graph using the graphistry package in python. To do this, we simply create a graph object and pass in the the columns for nodes and edges as parameters. The nodes in this graph are liquidators and liquidatees. The nodes are connected by liquidation transactions from liquidators to liquidatees.

Results

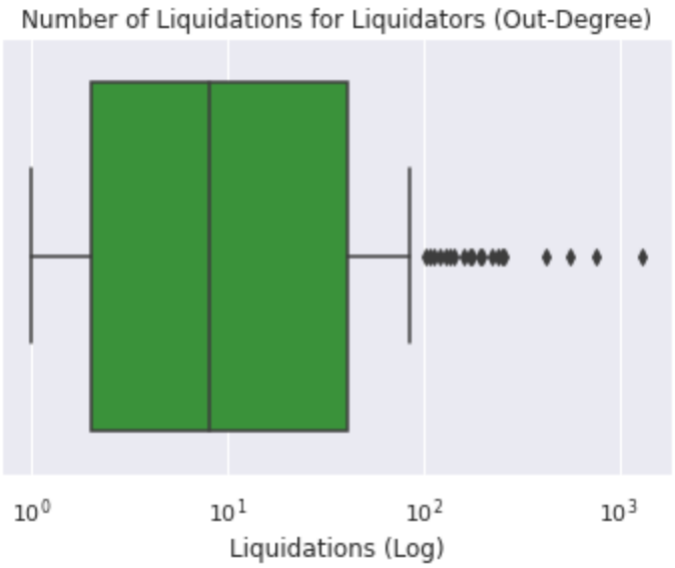
Below, is a zoomed out image of the graph. From this view, we can't observe much other than that there are many nodes and many connections.



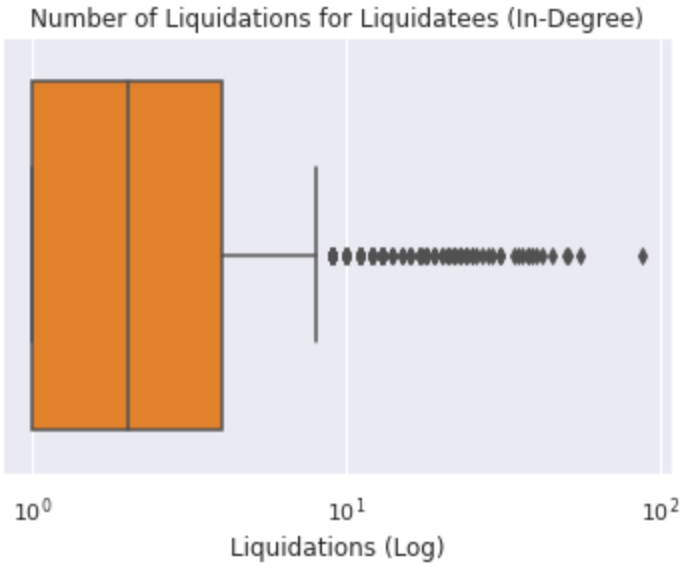
Next, is a zoomed in image of the graph. Here, we can more closely see the connections between liquidators and liquidatees. We can see that some liquidators liquidate more than others, and some users receieve more than one liquidations from the same liquidator.



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 third task I set out to accomplish was to cluster users in a way that could classify them according to their liquidation risk. To do so, I created a new dataset where the rows are users, and columns are statistics summarizing a user's behavior. For each user, I had features for the number of days they were active, their proportion for each transaction type to total number of transactions, and the log number of each transaction type.

Methods

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

```
#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
```

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/transactions.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(),
##   amountETH = 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: 116164 parsing failures.
##    row               col           expected                                    actual
## 198687 collateralAmount  1/0/T/F/TRUE/FALSE 0.3308551927545562                         '~/Blockchain/
## 198687 collateralReserve 1/0/T/F/TRUE/FALSE WETH                                        '~/Blockchain/
## 198687 liquidator        1/0/T/F/TRUE/FALSE 0x0c9d28f3d6a076484a357ad75a6a3b4df71c3f87 '~/Blockchain/
## 198687 principalAmount   1/0/T/F/TRUE/FALSE 639.17                                      '~/Blockchain/
## 198687 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 37
##     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 29 more variables: type <chr>, reservePriceETH <dbl>,
## #   reservePriceUSD <dbl>, amountETH <dbl>, amountUSD <dbl>,
## #   collateralAmount <lgl>, collateralReserve <lgl>, liquidator <lgl>,
## #   principalAmount <lgl>, principalReserve <lgl>,
## #   reservePriceETHPrincipal <lgl>, reservePriceUSDPrincipal <lgl>,
## #   reservePriceETHCollateral <lgl>, reservePriceUSDCollateral <lgl>,
## #   amountETHPincipal <lgl>, amountETHCollateral <lgl>, ...
```

Next, we create a new dataset where the rows are users, and the columns are features describing their behavior. For the columns, we choose the features that we found were able to distinguish liquidatees and non-liquidatees. These are the number of days a user is active, their proportions of each transaction type to all their transactions, and their log number of each transaction type.

```
#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
```

```r
#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.
```

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

```
##                                            user  timefirst   timelast  N
## 1 0x0000000000000000000000000000000000000001 1626953591 1626953591  1
## 2 0x000000000000000000000000000000000000dead 1613977574 1626148294  2
## 3 0x0000000000005117dd3a72e64a705198753fdd54 1638566584 1638566584  1
## 4 0x000000000007f150bd6f54c40a34d7c3d5e9f56 1613241549 1638613846 73
## 5 0x00000000000cd56832ce5dfbcbff02e7ec639bc9 1622302305 1622568495 10
## 6 0x00000000005dbcb0d0513fcda746382fe8a53468 1619325602 1619328421  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          0          NA        NA         NA        NA               NA
## 4   25372297          NA        NA         NA        NA               NA
## 5     266190    1.098612 0.1098612   1.098612 0.1098612               NA
## 6       2819          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           NA         NA          NA         NA        NA     NA
## 4            NA    3.2188758 0.04409419   3.7376696 0.05120095        NA     NA
## 5            NA    1.0986123 0.10986123   1.0986123 0.10986123        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       0.6931472   0.69314718
## 4       2.1972246   0.03009897
## 5       1.0986123   0.10986123
## 6       1.0986123   0.27465307
```

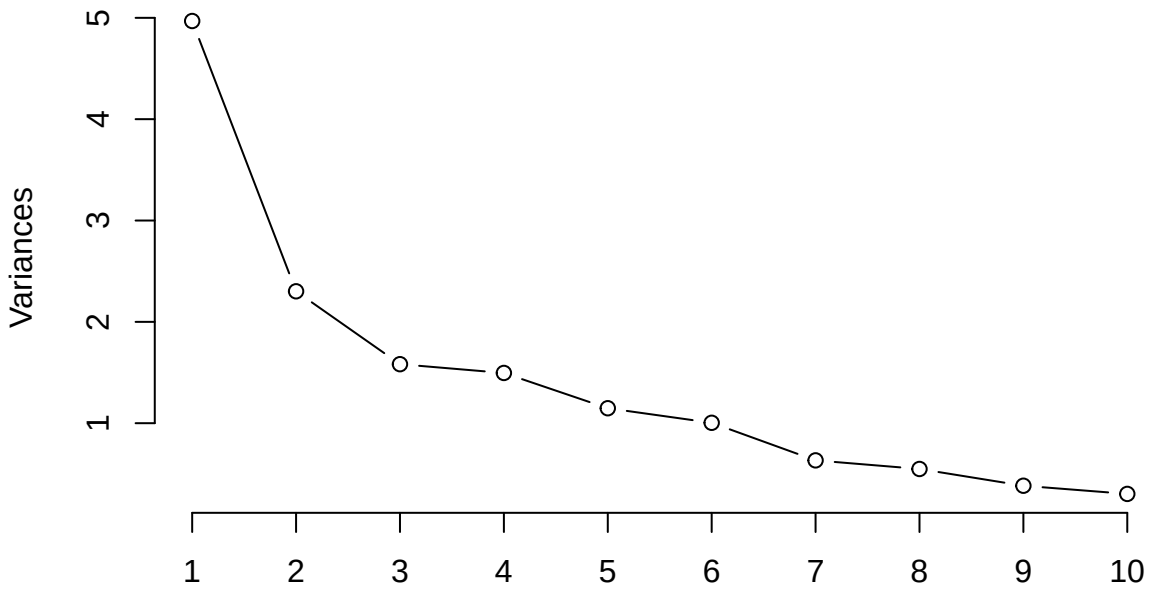
We clean the user data by replacing NaNs with 0s, removing unnecessary columns, and scaling the data.

```
#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 -0.6547065  -0.5885918 -0.5721290 -0.5330984 -0.5258178       -0.1807222
## 2  1.2806371  -0.5885918 -0.5721290 -0.5330984 -0.5258178       -0.1807222
## 3 -0.6547065  -0.5885918 -0.5721290 -0.5330984 -0.5258178       -0.1807222
## 4  3.3799038  -0.5885918 -0.5721290 -0.5330984 -0.5258178       -0.1807222
## 5 -0.6123780   0.7597721  0.8272694  1.0110078  1.4072437       -0.1807222
## 6 -0.6542583  -0.5885918 -0.5721290 -0.5330984 -0.5258178       -0.1807222
##   p_liquidation logn_deposit  p_deposit logn_redeem   p_redeem logn_swap
## 1     -0.154171   -1.1388494 -1.1903722  -0.7948098 -0.8602138 -0.141967
## 2     -0.154171   -1.1388494 -1.1903722  -0.7948098 -0.8602138 -0.141967
## 3     -0.154171   -1.1388494 -1.1903722  -0.7948098 -0.8602138 -0.141967
## 4     -0.154171    3.2896196 -0.8215746   4.6505219 -0.1905790 -0.141967
## 5     -0.154171    0.3726008 -0.2715082   0.8057352  0.5766131 -0.141967
## 6     -0.154171   -0.1852305  0.2589744   0.2150216  1.4061282 -0.141967
##       p_swap logn_collateral p_collateral
## 1 -0.1151227      -0.8263869  2.392849732
## 2 -0.1151227      -0.1045062  1.570295871
## 3 -0.1151227      -0.8263869  2.392849732
## 4 -0.1151227       1.8514377 -1.398786388
## 5 -0.1151227      -0.1045062 -0.942666505
## 6 -0.1151227      -0.1045062 -0.000305614
```

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")
```

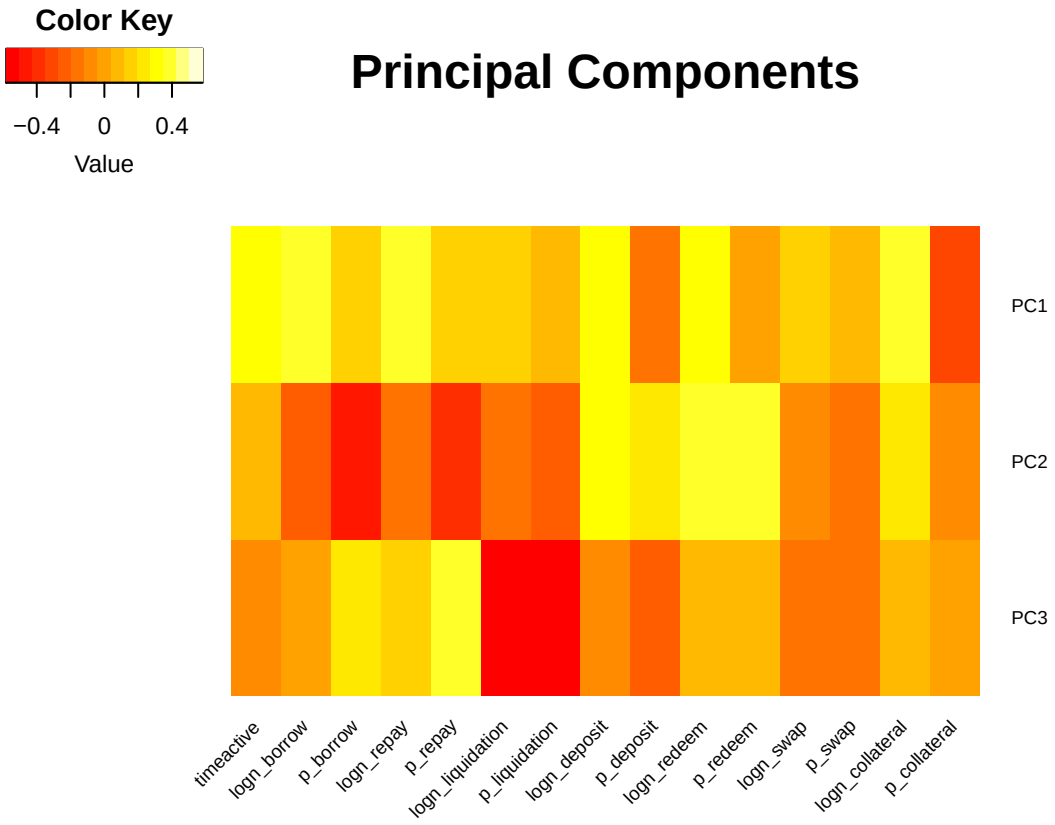



```
#summary of PCA
summary(my.pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.2290 1.5171 1.2579 1.22297 1.07129 1.00195 0.79566
## Proportion of Variance 0.3312 0.1534 0.1055 0.09971 0.07651 0.06693 0.04221
## Cumulative Proportion  0.3312 0.4847 0.5902 0.68988 0.76639 0.83332 0.87552
##                           PC8     PC9    PC10    PC11   PC12    PC13    PC14
## Standard deviation     0.74048 0.61880 0.54978 0.50463 0.4189 0.33241 0.22658
## Proportion of Variance 0.03655 0.02553 0.02015 0.01698 0.0117 0.00737 0.00342
## Cumulative Proportion  0.91208 0.93761 0.95776 0.97473 0.9864 0.99380 0.99722
##                           PC15
## Standard deviation     0.20425
## Proportion of Variance 0.00278
## Cumulative Proportion  1.00000
```

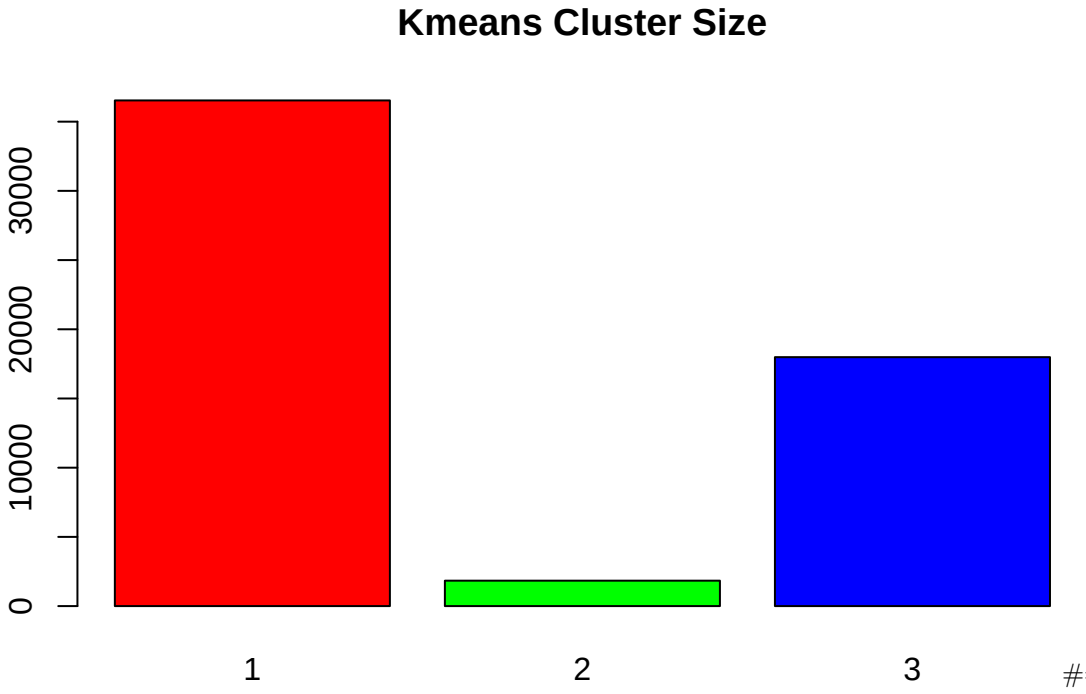
We select 3 principal components. Then, we create a heatmap showing the makeups of these components.

```
#select 3 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',srtCol=45)
```



Finally, we perform kmeans clustering on the data. We select 3 clusters for the 3 principal components. The cluster sizes are imbalanced. The majority of users fall in to cluster 1

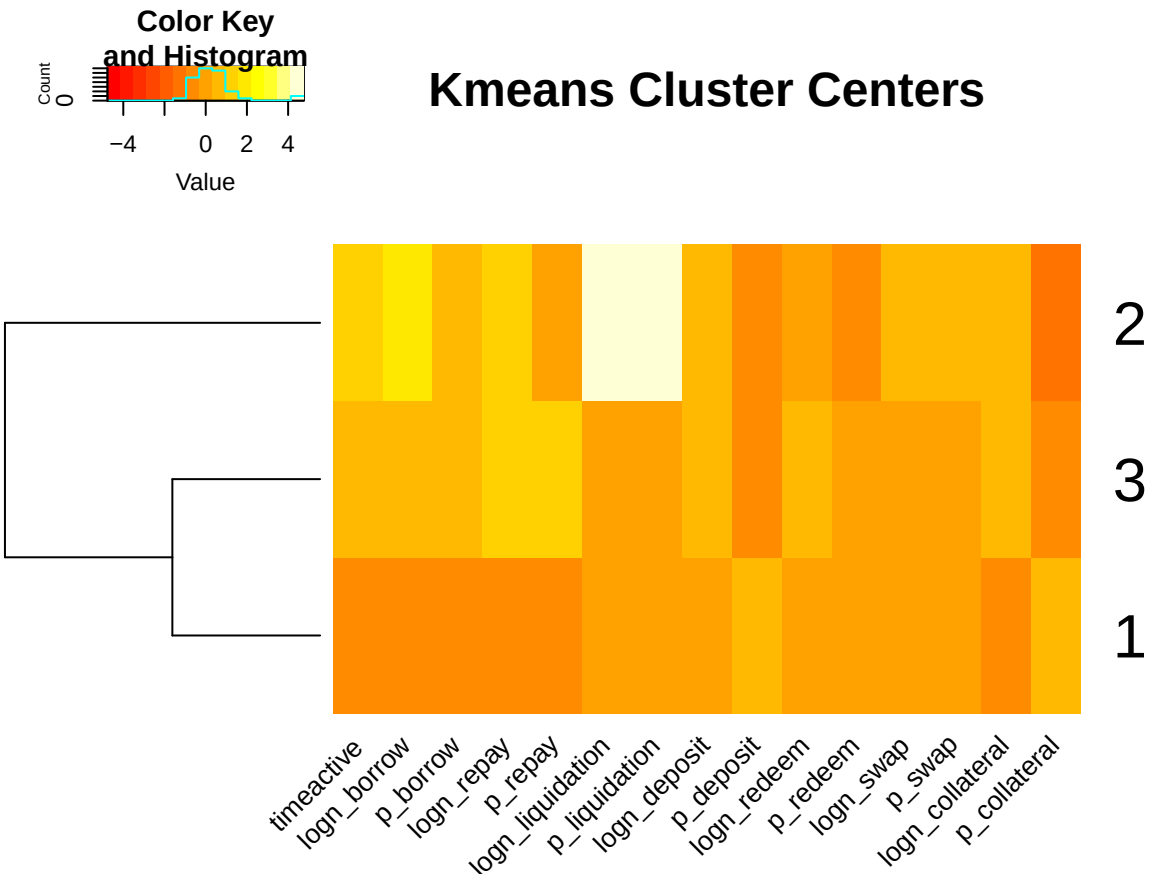
```
#run k-means algorithm
pca.matrix<-my.pca$x[,1:ncomps]
set.seed(1)
km <-kmeans(df.scaled,ncomps)

#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

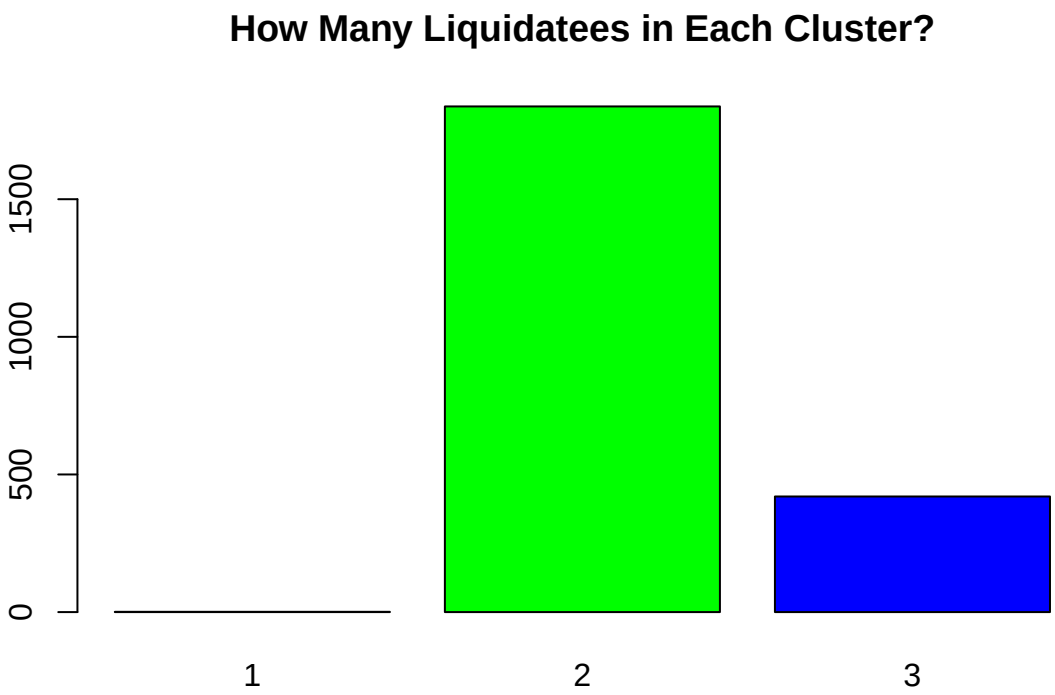
We analyze the clusters by making a heatmap of their means. Cluster 2 is the high risk cluster, which we can see from the cluster means for the columns corresponding to liquidations. When trying to find patterns of risk, we can look to the means in cluster 2. The most important factors are a low proportion of collateral changes, and a large number of borrows.

```
#make heatmap of cluster centers
heatmap.2(km$centers,
scale = "none",
dendrogram = "row",
Colv=FALSE,
cexCol=1.0,
main = "Kmeans Cluster Centers",
trace ="none",
srtCol=45)
```



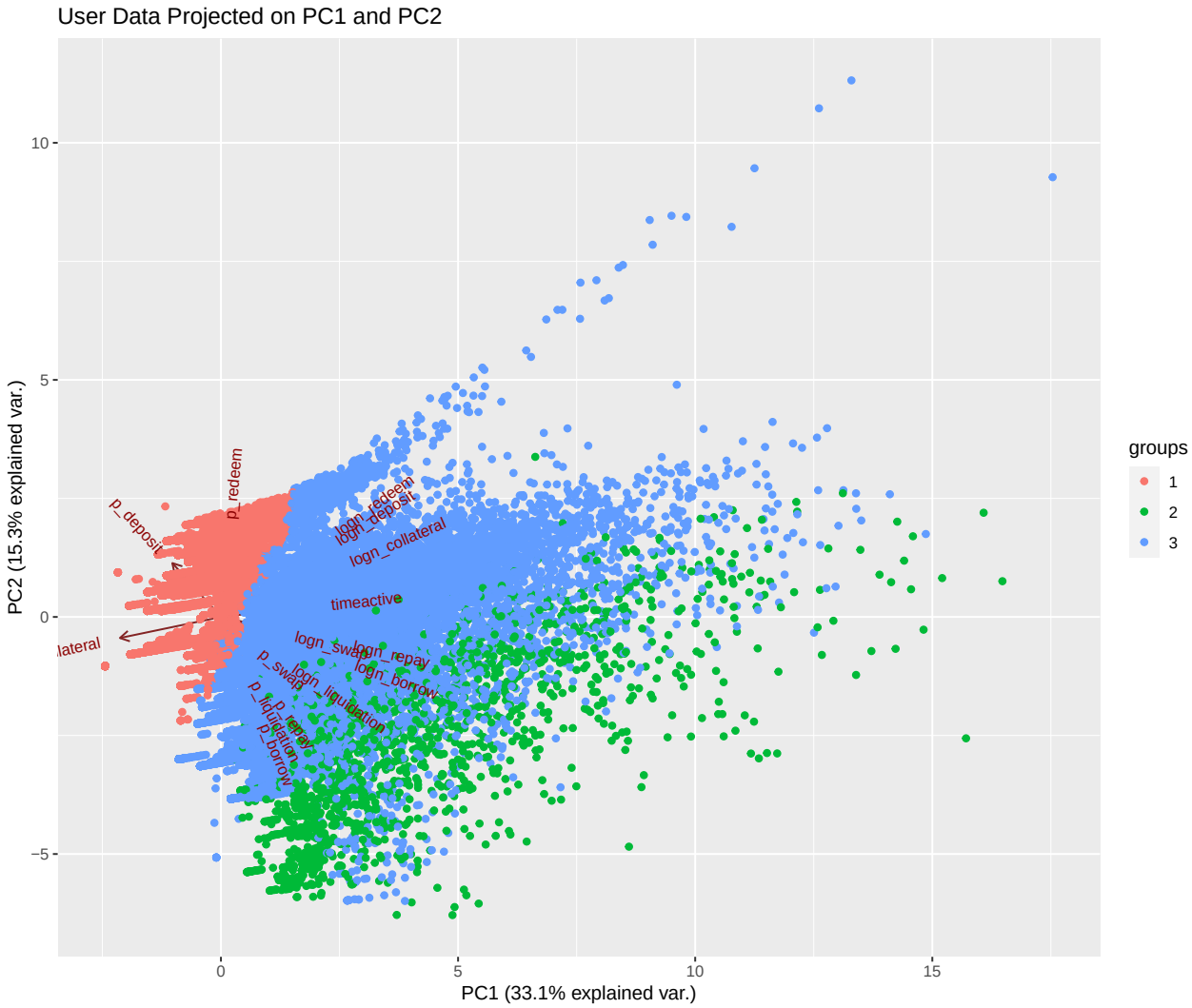
Next, we observe that liquidators are almost exclusive to this cluster. A significant amount of liquidatees also belong to cluster 3. It seems that cluster 1 is a low risk group, cluster 2 a high risk group, and cluster 3 a medium risk group.

```
#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 Liquidatees in Each Cluster?',col=c('red','green',
```



To conclude, we create a biplot of the different clusters. We can see that cluster 1 users are more condensed in their behavior patterns, while there is more of a spread for clusters 2 and 3.

```
#make biplot for clusters
plot1<-ggbiplot(my.pca,choices=c(1,2),
  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 1 was the low risk group. This group performed few borrows and repays, and experienced few liquidations. These users seem to use Aave as a savings account, where they deposit their cryptocurrencies they are not using to generate interest. Cluster 3 is the medium risk group. They perform borrows and repays in addition to their deposits and redeems, and have a significant number of liquidations, but very low proportionally to the size of their cluster. Cluster 2 is the high risk group. Almost all users in this group are liquidatees. The key difference between cluster 3 and cluster 2 is that cluster 3 redeems less and repays more. Both of these attributes are what is required to maintain a good health factor.

Problem 4

The final task that I worked on was predicting whether a user would liquidate within a certain time frame. This task is still a work in progress.

Methods

To begin this task, I created an algorithm in python to transform the data into a different format. The algorithm keeps track of a user's total deposit amounts and borrow amounts for each day. Every time a user makes a transaction, it updates the user's accounts balances. In addition, every day the user's balances are updated with an estimation of what they would gather in interest. The notebook where I gather this data is

titled DailyUserData.ipynb.

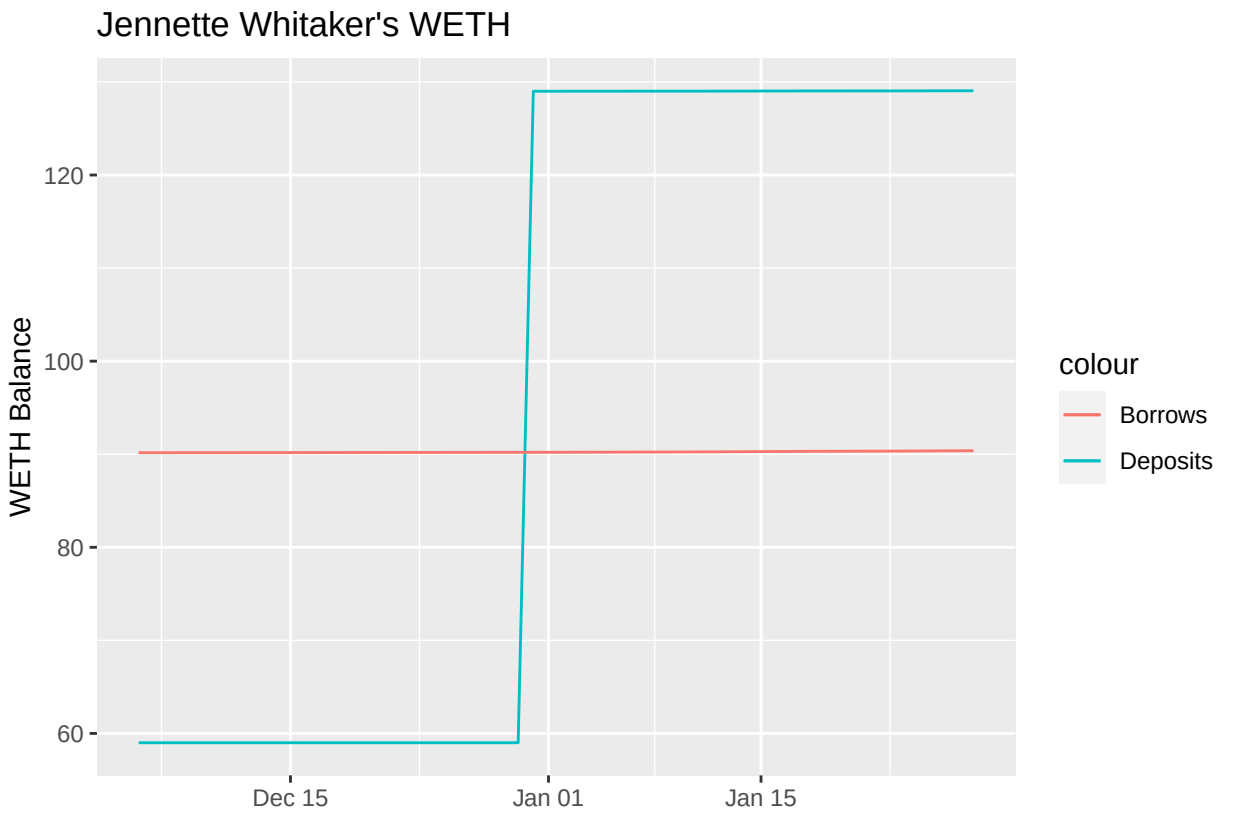
Results

I took a random user from the dataset to plot their account balances. The alias for this user is Jennette Whitaker. Below, we load in this data to a dataframe. The data has a column for each token the user borrowed, and each token the user deposited. T

```
df<-read.csv('JennetteWhitaker.csv')
df$datetime <- as.Date(df$datetime)
head(df)
```

```
##     datetime WBTC_deposited YFI_deposited AAVE_deposited WETH_deposited
## 1 2020-12-05       2.776108      2.012104       189.4519       59.00001
## 2 2020-12-06       2.776163      2.012114       189.4519       59.00003
## 3 2020-12-07       2.777164      2.012118       189.4519       59.00006
## 4 2020-12-08       2.777169      2.012123       189.4519       59.00009
## 5 2020-12-09       2.777169      2.012127       189.4519       59.00012
## 6 2020-12-10       2.777184      2.012132       189.4519       59.00013
##   WETH_borrowed DAI_borrowed USDT_borrowed WBTC_borrowed   WBTC_price
## 1      90.16262            0             0             0 2.886995e-04
## 2      90.16362            0             0             0 2.920228e-04
## 3      90.16463            0             0             0 2.917900e-04
## 4      90.16557            0             0             0 3.324063e+01
## 5      90.16638            0             0             0 3.224000e+01
## 6      90.16698            0             0             0 3.246016e+01
##   WBTC_ETH_deposited YFI_price YFI_ETH_deposited AAVE_price AAVE_ETH_deposited
## 1       8.014609e-04  49.60368          99.80778  0.1535300           29.08655
## 2       8.107029e-04  48.24907          97.08264  0.1477550           27.99246
## 3       8.103488e-04  48.30783          97.20103  0.1539325           29.16280
## 4       9.231485e+01  45.90871          92.37397  0.1446992           27.41354
## 5       8.953594e+01  45.00963          90.56509  0.1442550           27.32938
## 6       9.014785e+01  43.29753          87.12034  0.1391037           26.35347
##   WETH_price WETH_ETH_deposited WETH_ETH_borrowed   DAI_price DAI_ETH_borrowed
## 1          1           59.00001          90.16262 0.001695082                0
## 2          1           59.00003          90.16362 0.001670000                0
## 3          1           59.00006          90.16463 0.001706000                0
## 4          1           59.00009          90.16557 0.001812571                0
## 5          1           59.00012          90.16638 0.001742000                0
## 6          1           59.00013          90.16698 0.001783960                0
##    USDT_price USDT_ETH_borrowed WBTC_ETH_borrowed sum_ETH_deposits
## 1 0.001679000                0                 0         187.8951
## 2 0.001664590                0                 0         184.0759
## 3 0.001694320                0                 0         185.3647
## 4 0.001809820                0                 0         271.1024
## 5 0.001734000                0                 0         266.4305
## 6 0.001780183                0                 0         262.6218
##   sum_ETH_liabilities sum_USD_deposits sum_USD_liabilities health_factor
## 1            90.16262         111909.0            53700.19      2.083958
## 2            90.16362         110583.4            54165.67      2.041577
## 3            90.16463         109403.6            53215.82      2.055847
## 4            90.16557         149795.3            49820.19      3.006718
## 5            90.16638         153650.8            51999.06      2.954877
## 6            90.16698         147525.1            50650.39      2.912616
```

```r
ggplot(df, aes(datetime)) +
  geom_line(aes(y = WETH_deposited, colour = "Deposits")) +
  geom_line(aes(y = WETH_borrowed, colour = "Borrows")) +
  xlab("") +
  ylab("WETH Balance") +
  ggtitle("Jennette Whitaker's WETH")
```



Discussion

Using this data, we can build a machine learning model to predict whether or not a user while liquidate whithin a time frame. An additional column can be added on to the data to represent whether a user received a liquidation in the future from the row's datetime value. In addition, we can sum the user's total deposits and borrows in ETH and USD, and use these values to calculate health factor.

Summary and Recommendations

Throughout this research, we were able to discover the pattern of behaviors common to users who receive liquidations. It seems that the best way to reduce risk on Aave is to keep your number of borrows to a minimum. The users with the least amount of risk used Aave as a savings account, where they only deposited their tokens to generate interest. If a user wishes to borrow money, the most important factor to reduce risk is to repay these loans, and to limit the number of redeems. 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 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.