DAR F21 Project Status Notebook

DeFi Assignment 4

Cole Paquin

10/14/2021

Contents

Weekly Work Summary 1
Discussion of Primary Findings 1

Building Dataframe - SKIP **1**

Creating Clusters **4**

Graphing Transactions over Time **8**

Predicting Liquidators **11**

Weekly Work Summary

- RCS ID: paquic
- Project Name: DeFi
- New this week: Visualization of cluster transactions over time and regression models.

Discussion of Primary Findings

- Discuss primary findings:
  - What did you want to know?

  Among the clusters that I used, I wanted to be able to better visualize their activity over time. Also, I wanted to look into seeing if we could use regression models to predict how many liquidations a user would have.

  - How did you go about finding it?

  I used ggplot and facet_grid to be able to show how the users behavied. These charts allow us to see differences between clusters other than just looking at the cluster centers. For the regression, I used a ridge regression model on different features to try to predict how many liquidations a user has already had.

  - What did you find?

Building Dataframe - SKIP

```
#import libraries
```

```r
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)
```

```
## Warning in system("timedatectl", intern = TRUE): running command 'timedatectl'
## had status 1

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

## v tibble  3.0.6     v dplyr   1.0.4
## 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
```

```r
library(doParallel)
```

```
## Loading required package: iterators
## Loading required package: parallel
```

```
library(Rtsne)
library(anytime)
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

We begin by loading the csv file in to a dataframe.

```
#load in csv file to data frame
df<-readRDS("~/transactions.Rds")
```

We reformat the dataframe so that each row represents a user, and the columns represent a summarization of the user's transaction history.

```
#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)%>%
  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$timefirst
#get amounts for columns
df$logUSD<-log10(df$amountUSD)
df$logCollateralUSD<-log10(df$amountUSDCollateral)
#get user's transaction information
for(Type in unique(df$type)){
  #filter for only transactions of certain type
  df.type <-filter(df%>%group_by(user)%>%
                   count(type),type==Type)

  #add means of each transaction type
  if(Type!="liquidation" || Type!="swap"){
    df.sum<-filter(df,type==Type)%>%
      group_by(user)%>%
      summarise(Sum=sum(logUSD))
    colnames(df.sum)[2]<-paste('total_',Type,sep='')
    df.users<-merge(x=df.users,y=df.sum,by="user",all.x=TRUE)
  }

  #add counts of transaction types to df
  ntypes<-paste("n",Type,sep='')
  colnames(df.type)[3]<-ntypes
  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("prop_",Type,sep='')]<-(df.users[ntypes]+.05)/((df.users$N)+.3)
  #df.users[paste("weekly_",Type,sep='')]<-df.users[ntypes]/(((df.users$timeactive)+1)/(3600*24*7))
}
```

```
## 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.
```

```
#subset only columns we wish to scale by removing columns that we will not cluster on
df.sub<-select(df.users,-c(user,timefirst,timelast,nborrow,nrepay,nswap,nliquidation,nredeem,ndeposit, 
#repalce missing values as 0's
df.sub<-df.sub%>%replace(is.na(.),0)
```

We scale the data to prepare for PCA.

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

head(df.scaled)
```

```
##              N timeactive total_borrow prop_borrow total_repay prop_repay
## 1  0.12340615  2.5963829 -0.018098571  -0.6823453 -0.09659086 -0.5838345
## 2 -0.01276291 -0.6237177  0.008811225   0.4408348  0.16048805  0.9639993
## 3 -0.03269008 -0.6803428 -0.018098571  -0.6823453 -0.09659086 -0.5838345
## 4 -0.03269008 -0.3040833 -0.018098571  -0.6823453 -0.09659086 -0.5838345
## 5 -0.02936889 -0.6303294 -0.001613675   2.1426229 -0.01290475  1.4101599
## 6 -0.03269008 -0.6808614 -0.018098571  -0.6823453 -0.09659086 -0.5838345
##   prop_liquidation total_deposit prop_deposit total_redeem prop_redeem
## 1        -0.165215  -0.130588078 -0.031764349 -0.313479996   1.5425794
## 2        -0.165215   0.007397830 -0.762439872  0.012528750   0.1799580
## 3        -0.165215   0.004685514  0.002019969  0.006565713   1.1263329
## 4        -0.165215   0.005097473  0.002019969  0.007475566   1.1263329
## 5        -0.165215   0.002882137 -1.663546910  0.002600763  -0.9355814
## 6        -0.165215   0.006767538  1.588274139  0.002600763  -0.9355814
##     prop_swap
## 1 -0.09209154
## 2 -0.09209154
## 3 -0.09209154
## 4 -0.09209154
## 5 -0.09209154
## 6 -0.09209154
```

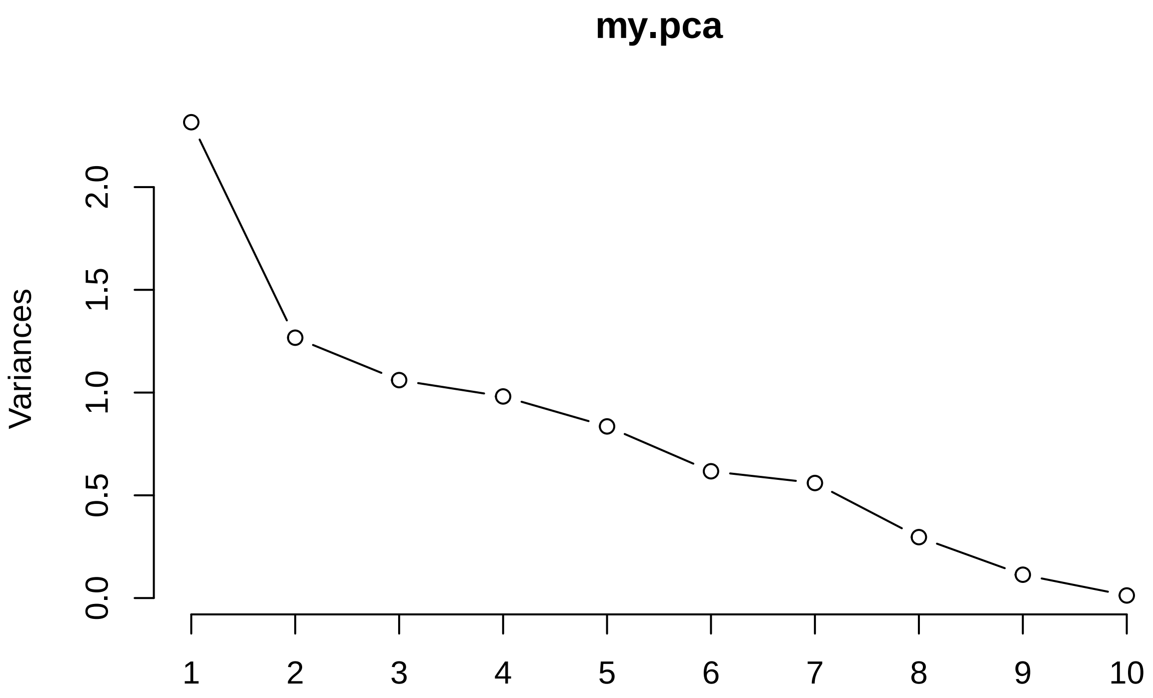
Creating Clusters

```
#Get rid of outliers
df.scaled <- df.scaled[-c(35019, 32511, 5258),]
df.users <- df.users[-c(35019, 32511, 5258),]

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

```
## Importance of components:
##                          PC1   PC2    PC3    PC4    PC5     PC6     PC7     PC8
## Standard deviation     1.522 1.126 1.0299 0.9906 0.9140 0.78554 0.74827 0.54455
## Proportion of Variance 0.287 0.157 0.1315 0.1216 0.1035 0.07647 0.06939 0.03675
## Cumulative Proportion  0.287 0.444 0.5754 0.6970 0.8006 0.87704 0.94643 0.98318
##                            PC9    PC10    PC11    PC12
## Standard deviation     0.33709 0.11265 0.07286 0.06406
## Proportion of Variance 0.01408 0.00157 0.00066 0.00051
## Cumulative Proportion  0.99726 0.99883 0.99949 1.00000
```

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

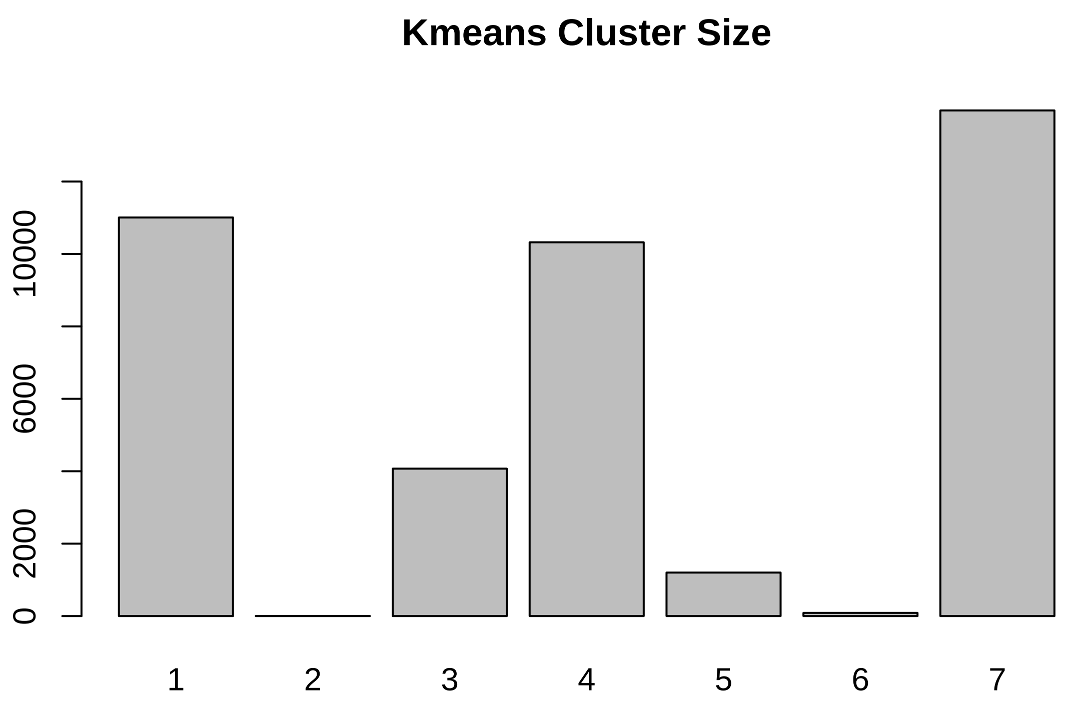


```
ncomps=5
```

Although we could pick several different numbers of clusters, I chose 7 because it gives us a good amount of distinction between them.

Finally, we run the kmeans clustering algorithm on the data. We select the number of clusters equal to the number of selected components.

```
#run kmeans algorithm
set.seed(1)
km <-kmeans(df.scaled,7)
#plot frequencies of each cluster
barplot(table(km$cluster),main="Kmeans Cluster Size")
```

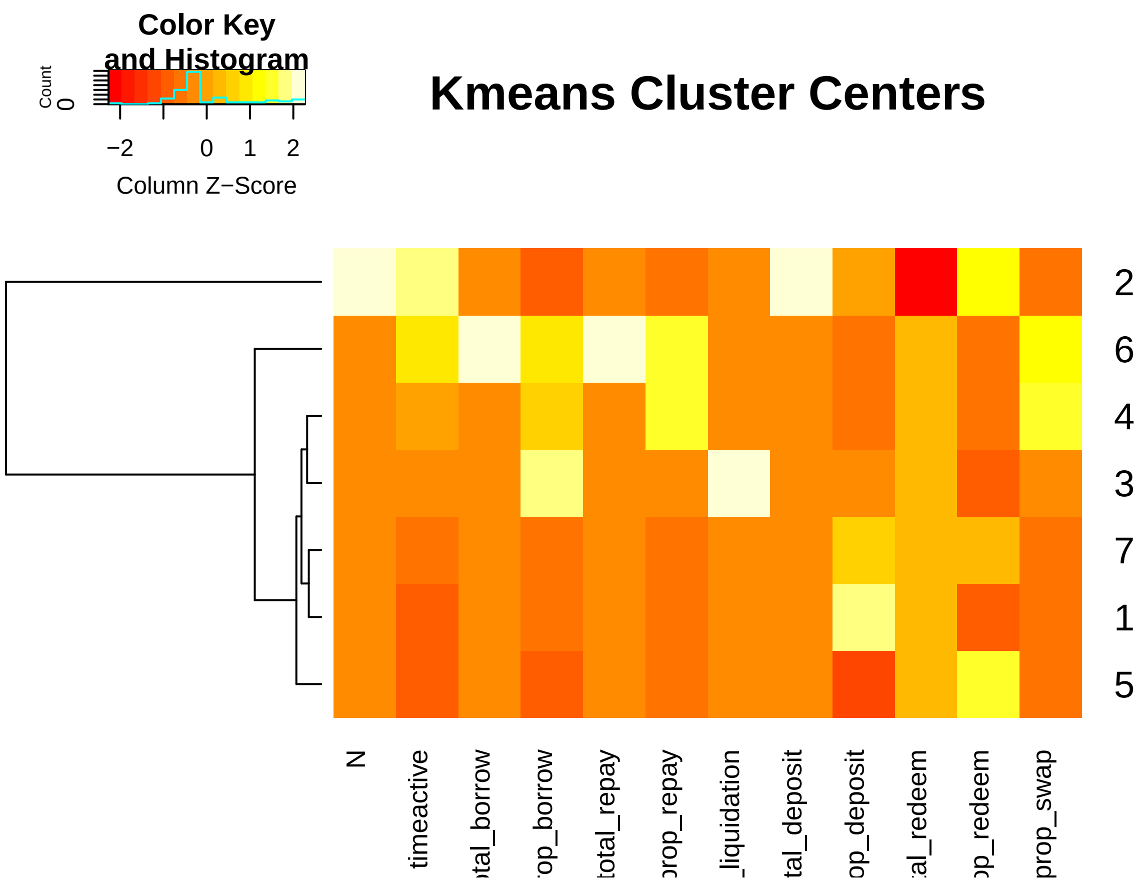


```
km$size
```

```
## [1] 11007     3  4071 10322  1202    88 13963
```

This is interesting. We can see that cluster 2 only contains three users, and 6 has 88. Thus, we want to look at why they are distinctive, but also be sure to not overanalyze their charts since it is a small sample size. Finally, we view the centers of the kmeans clusters.

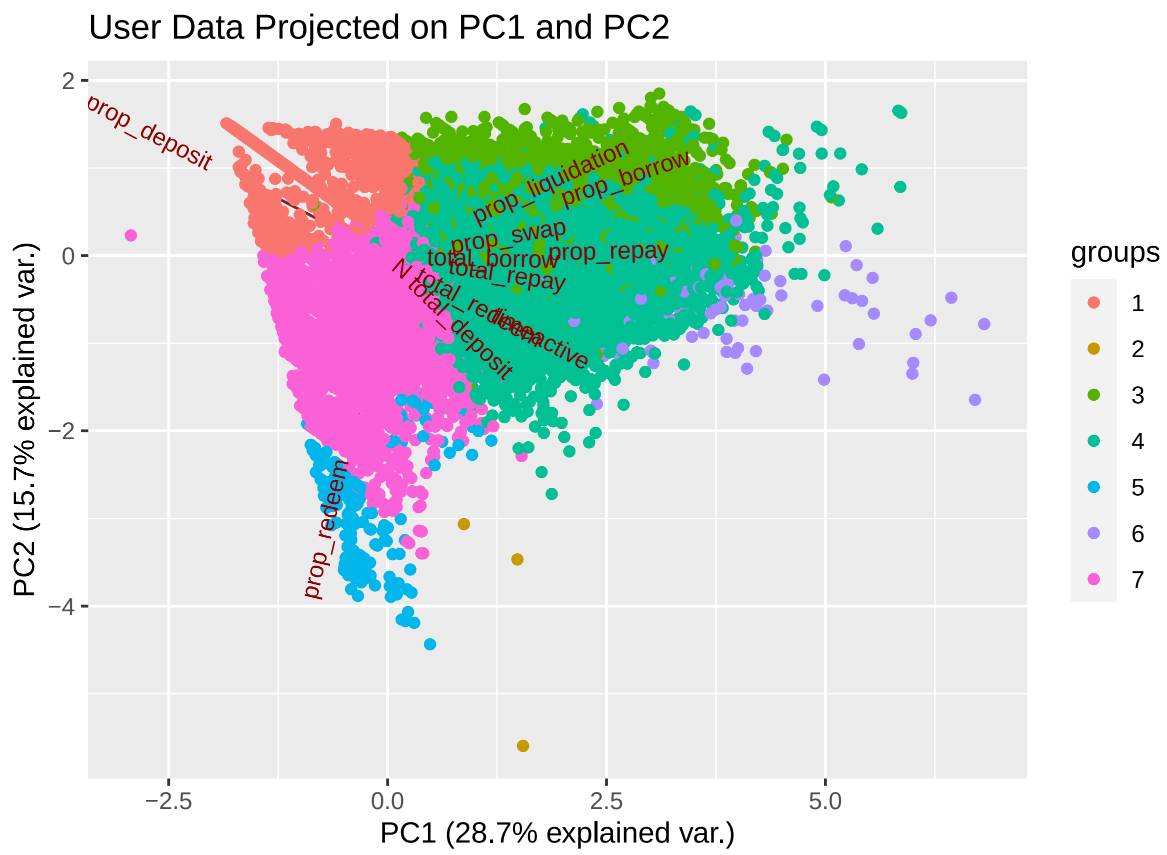
```
#make heatmap of cluster centers
heatmap.2(km$centers,
scale = "col",
dendrogram = "row",
Colv=FALSE,
cexCol=1.0,
main = "Kmeans Cluster Centers",
trace ="none")
```



We see that our tiny cluster, 2, has users that have been active for a long time. Not only that, but they have the most total transactions and have significantly deposited a lot of money. Cluster 6 (the other small one) has users that have borrowed and repayed large amounts of money. Cluster 7, which is our largest cluster, seems to represent a very average user. It should be noted that cluster 3 us made up of the users who were liquidated.

- Exploring the Influential Factors with a Biplot

```
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
```



This biplot gives us a good visualization of the clusters, and we can see that our smaller clusters are seen as outliers in this plot.

Graphing Transactions over Time

Now, I started to visualize how these people acted over time. First, I wanted to check on the date range we were looking at.

```
df["date"] = anydate(df$timestamp)
min(df$date)
```

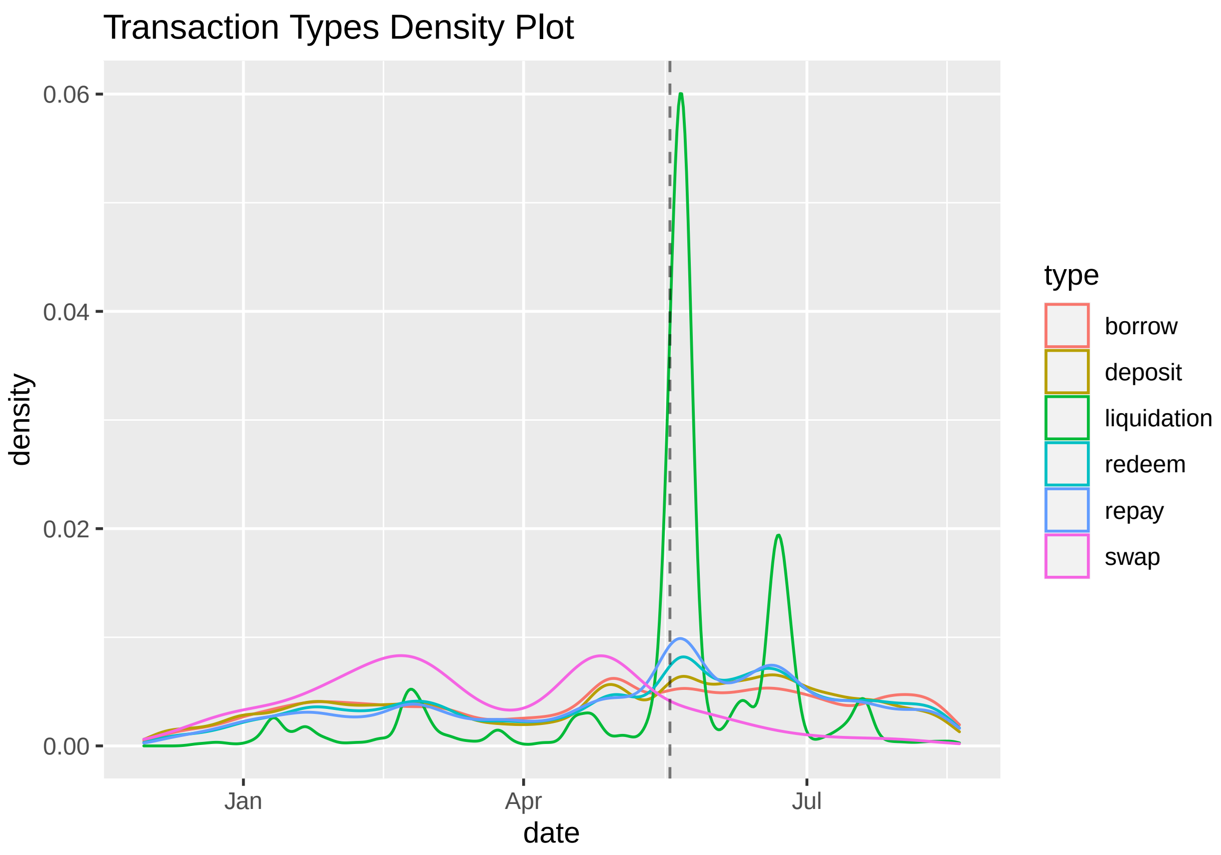
```
## [1] "2020-11-30"
```

```
max(df$date)
```

```
## [1] "2021-08-19"
```

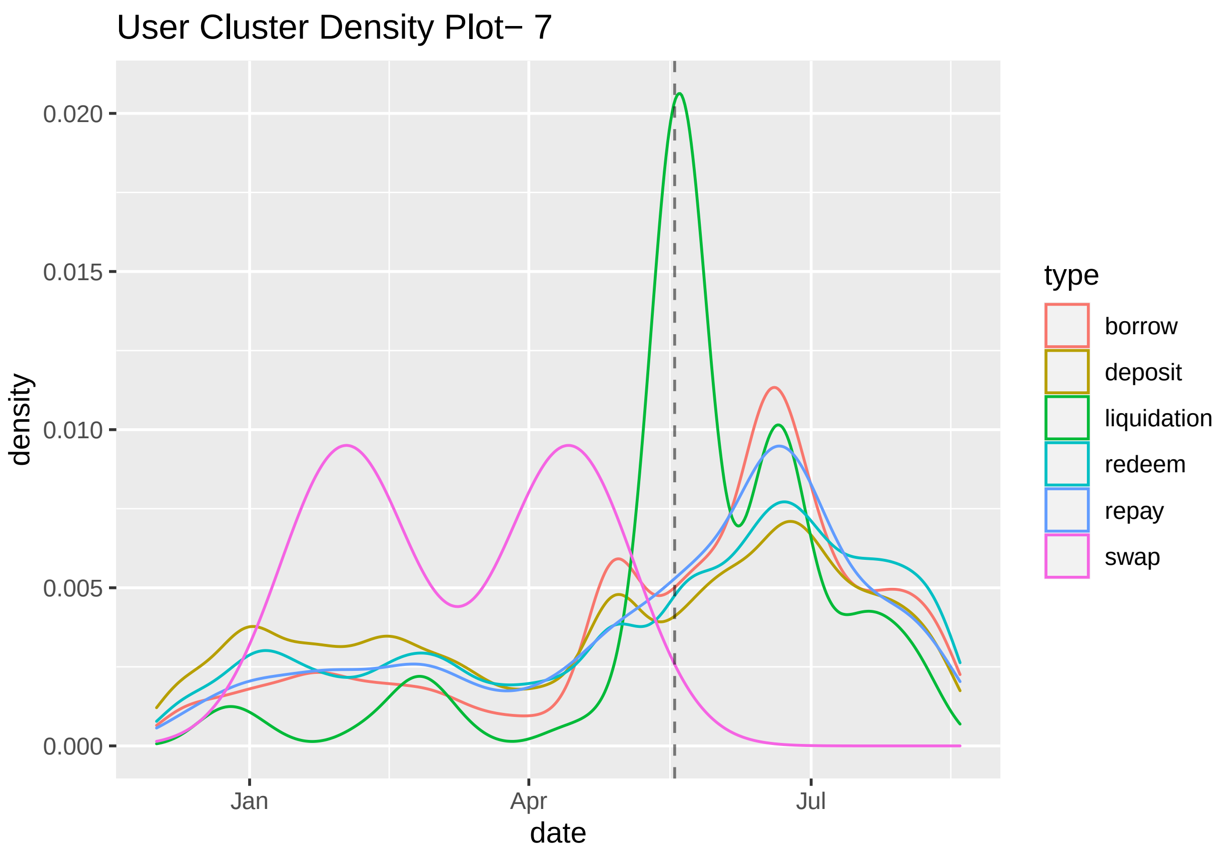
Knowing this, I could graph the densities of all transactions over time. It should be noted that the dotted vertical line represents the day which China announced their crackdown on cryptocurrencies, which we hypothesize could affect our behavior.

```
ggplot(data = df, aes(x = date,  group = type, color = type)) +
  geom_density()+
  geom_vline(xintercept = as.numeric(as.Date("2021-05-18")), linetype=2, alpha = 0.5)+
  ggtitle("Transaction Types Density Plot")
```



We can see how liquidations and swaps tend to have peaks at certain times, and there was a large jump in liquidations immediately after China announced. I believe this is correlated with the drop in price of the crypto market. From this, we can start to look at the actions of individual clusters. I will show cluster 7 since it is our largest cluster.

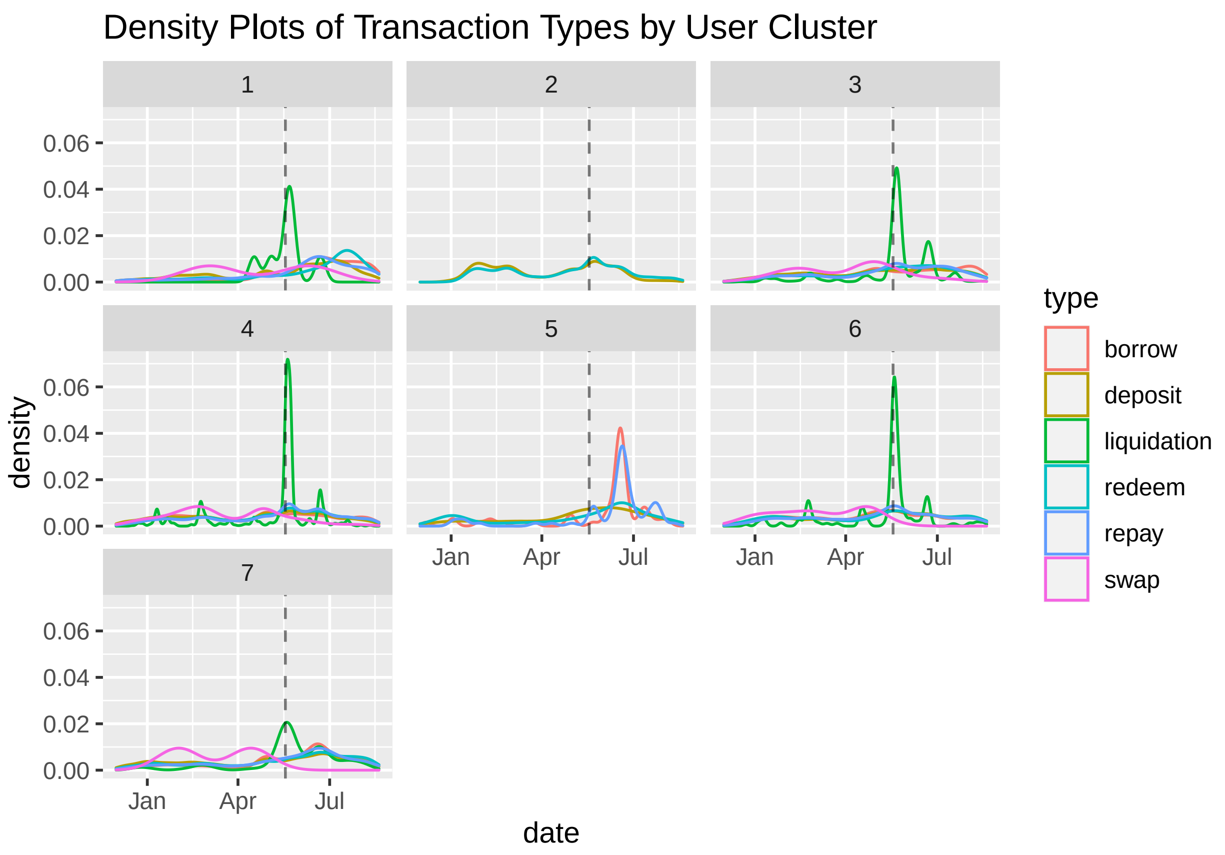
```
user_ids <- filter(df.users, km$cluster == 7)
ggplot(data = filter(df, df$user %in% user_ids$user), aes(x = date,  group = type, color = type)) +
  geom_density()+
  ggtitle("User Cluster Density Plot- 7")+
  geom_vline(xintercept = as.numeric(as.Date("2021-05-18")), linetype=2, alpha = 0.5)
```



Now, I will get a bit fancier and use facet_grid to show all of these clusters at the same time. This allows us to make better comparisons.

```
df.clusters <- data.frame(user = df.users$user, cluster = km$cluster)
df <- left_join(df, df.clusters, by = "user")
```

```
ggplot(data = df[!(is.na(df$cluster)), ], aes(x = date,  group = type, color = type)) +
  geom_density()+
  ggtitle("Density Plots of Transaction Types by User Cluster")+
  geom_vline(xintercept = as.numeric(as.Date("2021-05-18")), linetype=2, alpha = 0.5)+
  facet_wrap(~ cluster)
```



This gives us a much better comparison, at the cost of the detail for each individual graph. That is why it is still helpful to have the individual charts. This does show however, how groups 1, 3, 4, and 6 were the ones who saw the greatest spikes in liquidations in early May.

Predicting Liquidators

A natural question to ask is how well we can predict how many times a user will be liquidated based off of their previous history. I briefly ran a regression model to see how this could work.

Next, we select only the columns we care about.

```
#subset only columns we wish to scale by removing columns that we will not cluster on
df.users$num_liquidations = round(df.users$N * df.users$prop_liquidation)
df.final<-select(df.users,-c(user,timefirst,timelast,nborrow,nrepay,nswap,nliquidation,nredeem,ndeposit
#repalce missing values as 0's
df.final<-df.final%>%replace(is.na(.),0)
```

```
summary(df.final)
```

```
##        N              timeactive         total_borrow        prop_borrow
##  Min.   :    1.00   Min.   :       0   Min.   :-206.659   Min.   :0.0000
##  1st Qu.:    2.00   1st Qu.:      95   1st Qu.:   0.000   1st Qu.:0.0000
##  Median :    2.00   Median :  780584   Median :   0.000   Median :0.0000
##  Mean   :   10.15   Mean   : 3166168   Mean   :   8.626   Mean   :0.1500
##  3rd Qu.:    6.00   3rd Qu.: 4578475   3rd Qu.:   4.999   3rd Qu.:0.2760
##  Max.   :20779.00   Max.   :22282652   Max.   :1698.226   Max.   :0.9855
##   total_repay          prop_repay       total_deposit       prop_deposit
##  Min.   :-1196.772   Min.   :0.00000   Min.   :-7335.57   Min.   :0.0000
##  1st Qu.:    0.000   1st Qu.:0.00000   1st Qu.:    1.99   1st Qu.:0.2470  
##  Median :    0.000   Median :0.00000   Median :    3.67   Median :0.4565  
##  Mean   :    3.429   Mean   :0.09316   Mean   :    8.04   Mean   :0.4560  
##  3rd Qu.:    2.699   3rd Qu.:0.16667   3rd Qu.:    7.75   3rd Qu.:0.7642  
##  Max.   : 1285.102   Max.   :0.95283   Max.   :51917.19   Max.   :0.9999  
##   total_redeem        prop_redeem       prop_swap         num_liquidations
##  Min.   :-45962.20   Min.   :0.0000   Min.   :0.000000   Min.   : 0.0000  
##  1st Qu.:     0.00   1st Qu.:0.0000   1st Qu.:0.000000   1st Qu.: 0.0000  
##  Median :     2.03   Median :0.1541   Median :0.000000   Median : 0.0000  
##  Mean   :     3.53   Mean   :0.2071   Mean   :0.001171   Mean   : 0.1538  
##  3rd Qu.:     4.54   3rd Qu.:0.4565   3rd Qu.:0.000000   3rd Qu.: 0.0000  
##  Max.   : 47217.46   Max.   :1.0000   Max.   :0.456522   Max.   :53.0000
```

Now, we have the features that we will run regression on, as well as what we are trying to predict (num_liquidations). Now we need to split our data into a training and testing set.

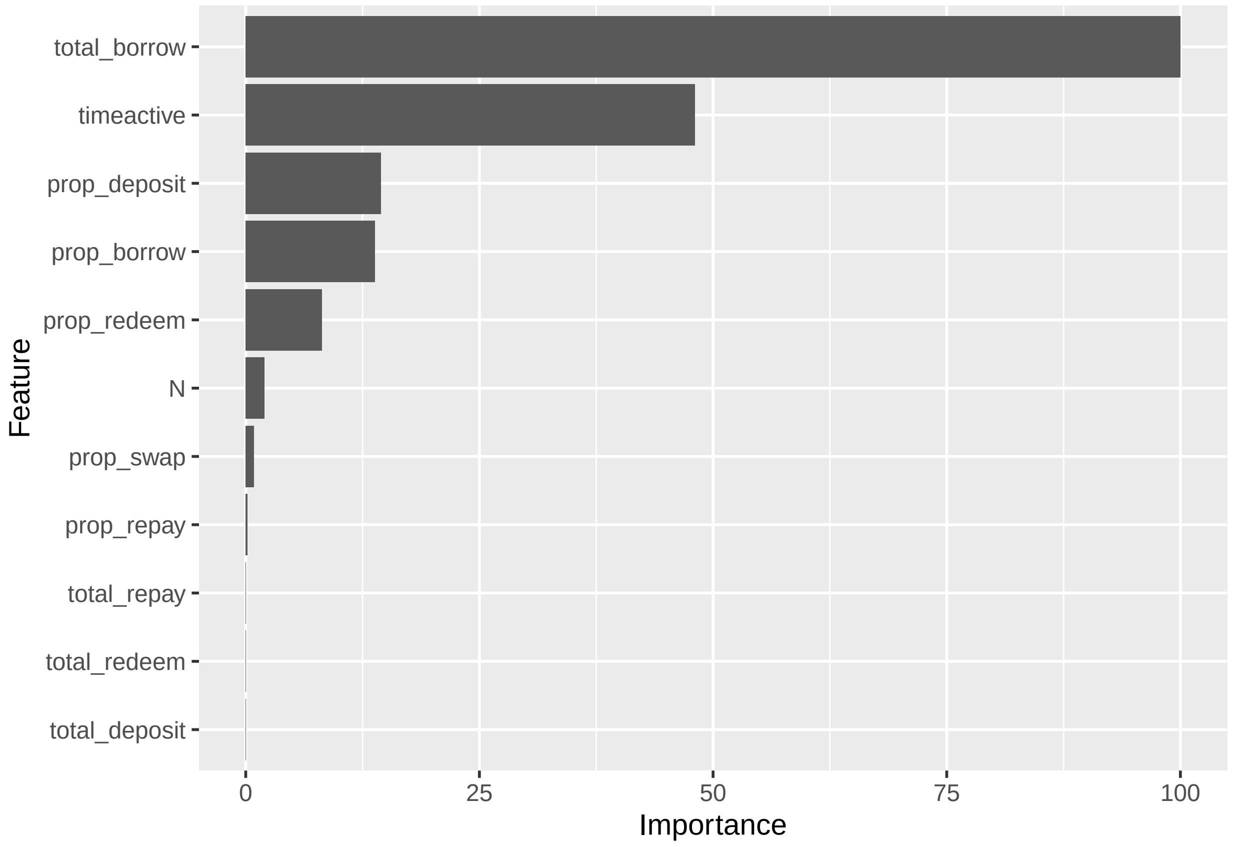
```r
# Split into training and testing data
set.seed(1)
training.samples <- df.final$num_liquidations %>%
  createDataPartition(p = 0.8, list = FALSE)
train.df <- df.final[training.samples, ]
test.df <- df.final[-training.samples, ]
```

Now that we have separated our data, we are able to build the model.

```r
#Build model and get predicted values
ridge_model <- train(num_liquidations ~., data = df.final, method = "ridge")
test_predict <- predict(ridge_model, test.df)
```

This chart shows us the relative importance of the different variables in our dataset.

```r
ggplot(varImp(ridge_model))
```



This chart seems to make sense. Users that borrow a lot of money and have been active for a while will be more likely. Also, it is a bit interesting to me to see how the proportion of deposits is so high, I'm guessing they have an inverse relationship. However, since we have our model, we can now look at its results.

```r
ridge_mse <- data.frame(pred = test_predict, actual = test.df$num_liquidations)
cat("Ridge Regression Model MSE: ", mean((ridge_mse$actual - ridge_mse$pred)^2))
```

```
## Ridge Regression Model MSE:  0.6200892
```

```r
true <- filter(ridge_mse, ridge_mse$actual > 0)
false <- filter(ridge_mse, ridge_mse$actual == 0)

cat("\nMSE where Users Actually Liquidated: ", mean((true$actual - true$pred)^2))
```

```
##
## MSE where Users Actually Liquidated:  11.04948
```

```r
cat("\nMSE of Users with 0 Liquidations: ", mean((false$actual - false$pred)^2))
```

```
##
## MSE of Users with 0 Liquidations:  0.1789023
```

Overall, this fairly simply model does a good job in predicting liquidations. One issue I think we may run into in more advanced models is avoiding models that predict low numbers for every user since most users do not get liquidated. We can see in this model, the MSE is significantly higher for users who had been liquidated. This is to be expected. The goal of future models will be to lower the error for liquidated users. Also, user a binary (or -1 and 1) classification model may help, but also will fail to distinguish between different amounts of liquidation.