DAR F21 Project Status Notebook

DeFi Assignment 5

Cole Paquin

10/28/2021

Contents

Weekly Work Summary

- RCS ID: paquic
- Project Name: DeFi
- Summary of work since last week
  - Describe the important aspects of what you worked on and accomplished Since the last assignment, I have done three major things. First, I have adjusted all of the code to work for transactionsV2, giving me more data to work on. Then, I adjusted previous graphs both for Aaron's presentation and to make them look clearer. Finally, I created a new dataframe that summarizes users on a weekly basis and began to look into if we can see different clusters from before.
- Summary of github commits

paquic_Assignment5-f21.Rmd – All raw code and charts from this week paquic_Assignment5.Rmd – What you are currently reading with select charts and descriptions paquic_Assignmentt.pdf

Personal Contribution

Other than some of the initial df.users and PCA which was partially written at the start of the semester, all of the code was written by me.

Discussion of Primary Findings

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

I wanted to see how our data looked now that we have almost a full year of data. Would we see a different pattern in transactions? I also wanted to break users down by there weekly activity as opposed to their raw data. This could lead to new methods of clustering and possibly a better understanding of their behavior.

`* How did you go about finding it?`

First, I adjusted a lot of my previous code to adapt to the new dataset. Not only do we now have more data points, we also have a new type of transactions and some other different features (such as the naming). After

that, most of the time was spent organizing a new dataframe and making some intital graphs to show the general trends of our users.

* What did you find?

```
#import libraries
if (!require("ggplot2")) {
   install.packages("ggplot2")
   library(ggplot2)
}
```

```
## Loading required package: ggplot2
```

```
if (!require("knitr")) {
   install.packages("knitr")
   library(knitr)
}
```

```
## Loading required package: knitr
```

```
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)
```

```
## 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()
```

```
library(tidyquant)
```

```
## Loading required package: lubridate
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
## Loading required package: PerformanceAnalytics
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
##
## Attaching package: 'xts'
```

```
## The following objects are masked from 'package:dplyr':
##
##     first, last
```

```
##
## Attaching package: 'PerformanceAnalytics'
```

```
## The following object is masked from 'package:gplots':
##
##     textplot
```

```
## The following object is masked from 'package:graphics':
##
##     legend
```

```
## Loading required package: quantmod
```

```
## Loading required package: TTR
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
## == Need to Learn tidyquant? ==========================================================
## Business Science offers a 1-hour course - Learning Lab #9: Performance Analysis & Portfolio Optimiza
## </> Learn more at: https://university.business-science.io/p/learning-labs-pro </>
```

```
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
```

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

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

```
#load in csv file to data frame
df<-read_rds("~/transactionsv2.rds")
head(df)
```

```
##       amount borrowRate borrowRateMode   onBehalfOf         pool reserve
## 1      15.00  0.2590658       Variable 1.117217e+48 1.034668e+48    WETH
## 2   41501.63  6.2749368       Variable 8.502518e+47 1.034668e+48     DAI
## 3 7000000.00  2.5896280       Variable 4.635974e+47 1.034668e+48    USDT
## 4   15000.00  8.8025409       Variable 3.735263e+47 1.034668e+48    USDC
## 5    8193.19 48.7470516         Stable 6.896232e+47 1.034668e+48    USDC
## 6   11000.00  3.2250550       Variable 1.089455e+48 1.034668e+48    USDT
##    timestamp         user   type reservePriceETH reservePriceUSD  amountUSD
## 1 1633275840 1.168069e+48 borrow    1.0000000000    3421.8708189   51328.06
## 2 1621340435 8.502518e+47 borrow    0.0002852900       0.9948044   41286.00
## 3 1622477822 4.635974e+47 borrow    0.0003812835       1.0000000 7000000.00
## 4 1619775984 3.735263e+47 borrow    0.0003611000       1.0043389   15065.08
## 5 1615481632 6.896232e+47 borrow    0.0005562201       0.9993909    8188.20
## 6 1626914745 1.089455e+48 borrow    0.0004971100       1.0000000   11000.00
##   collateralAmount collateralReserve liquidator principalAmount
## 1               NA                 NA         NA              NA
## 2               NA                 NA         NA              NA
## 3               NA                 NA         NA              NA
## 4               NA                 NA         NA              NA
## 5               NA                 NA         NA              NA
## 6               NA                 NA         NA              NA
##   principalReserve reservePriceETHPrincipal reservePriceUSDPrincipal
## 1               NA                       NA                       NA
## 2               NA                       NA                       NA
## 3               NA                       NA                       NA
## 4               NA                       NA                       NA
## 5               NA                       NA                       NA
## 6               NA                       NA                       NA
##   reservePriceETHCollateral reservePriceUSDCollateral amountUSDPincipal
## 1                        NA                        NA                NA
## 2                        NA                        NA                NA
## 3                        NA                        NA                NA
## 4                        NA                        NA                NA
## 5                        NA                        NA                NA
## 6                        NA                        NA                NA
##   amountUSDCollateral borrowRateModeFrom borrowRateModeTo stableBorrowRate
```

```
## 1                  NA                                                    NA
## 2                  NA                                                    NA
## 3                  NA                                                    NA
## 4                  NA                                                    NA
## 5                  NA                                                    NA
## 6                  NA                                                    NA
##   variableBorrowRate fromState toState protocolContract     user_alias
## 1                 NA                                True Gladys Marquez
## 2                 NA                               False  Angel Prather
## 3                 NA                               False   Jack Crowley
## 4                 NA                               False    Jim Dickens
## 5                 NA                               False  Leonard Reyes
## 6                 NA                               False      Jill Carn
##   onBehalfOf_alias            datetime
## 1  Evelyn Terrazas 2021-10-03 15:44:00
## 2    Angel Prather 2021-05-18 12:20:35
## 3     Jack Crowley 2021-05-31 16:17:02
## 4      Jim Dickens 2021-04-30 09:46:24
## 5    Leonard Reyes 2021-03-11 16:53:52
## 6        Jill Carn 2021-07-22 00:45:45
```

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, user_alias)%>%
  summarise(timefirst=min(anydate(timestamp)), timelast=max(anydate(timestamp)), N=n())
```

```
## `summarise()` has grouped output by 'user'. You can override using the `.groups` argument.
```

```
#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)
}
```

```
## 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       user_alias  timefirst   timelast  N timeactive
## 1 1.000000e+00    Tracy Nisbett 2021-07-22 2021-07-22  1     0 days
## 2 5.700500e+04 Millicent Gandhi 2021-02-22 2021-07-12  2   140 days
## 3 2.577533e+33  Steven Ferguson 2021-02-13 2021-10-09 60   238 days
## 4 6.663597e+34   Ronald Masella 2021-05-29 2021-06-01 10     3 days
## 5 4.867107e+35   Steven Ahumada 2021-04-25 2021-04-25  4     0 days
## 6 8.009427e+35   Melissa Esters 2020-12-05 2020-12-25  4    20 days
##   total_borrow nborrow prop_borrow total_repay nrepay prop_repay
## 1           NA      NA          NA          NA     NA         NA
## 2           NA      NA          NA          NA     NA         NA
## 3           NA      NA          NA          NA     NA         NA
## 4     9.447565       2   0.1990291    9.417525      2  0.1990291
## 5           NA      NA          NA          NA     NA         NA
## 6           NA      NA          NA          NA     NA         NA
##   total_liquidation nliquidation prop_liquidation total_deposit ndeposit
## 1                NA           NA               NA            NA       NA
## 2                NA           NA               NA            NA       NA
## 3                NA           NA               NA     64.160153       22
## 4                NA           NA               NA     10.009139        2
## 5                NA           NA               NA      3.997229        1
## 6                NA           NA               NA      4.910344        1
##   prop_deposit total_redeem nredeem prop_redeem total_swap nswap prop_swap
## 1           NA           NA      NA          NA         NA    NA        NA
## 2           NA           NA      NA          NA         NA    NA        NA
## 3    0.3656716    90.345340      32   0.5315091         NA    NA        NA
## 4    0.1990291    10.008833       2   0.1990291         NA    NA        NA
## 5    0.2441860     3.997237       1   0.2441860         NA    NA        NA
## 6    0.2441860     4.914500       1   0.2441860         NA    NA        NA
##   total_collateral ncollateral prop_collateral
## 1               NA           1       0.8076923
## 2               NA           2       0.8913043
## 3               NA           6       0.1003317
## 4               NA           2       0.1990291
## 5               NA           2       0.4767442
## 6               NA           2       0.4767442
```

```
nrow(df.users)
```

```
## [1] 51421
```

```
length(unique(df.users$user))
```

```
## [1] 51421
```

As long as those two numbers are the same, we are good to proceed. Next, we select only the columns we wish to cluster on.

```
#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 now scale the data to prepare for PCA.

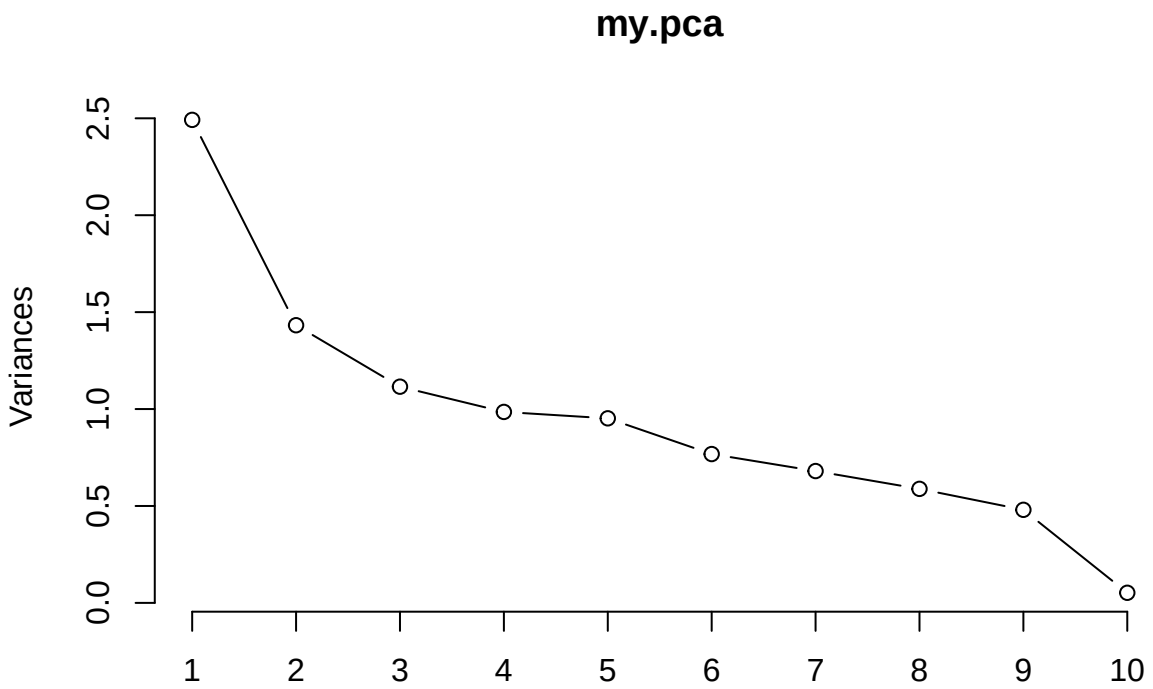
```
#scale data
df.scaled<-df.sub%>%mutate_all(scale)

summary(df.scaled)
```

```
##         N.V1            timeactive.V1      total_borrow.V1
##  Min.   : -0.03731   Min.   :-0.660775   Min.   : -0.29603
##  1st Qu.: -0.03455   1st Qu.:-0.660775   1st Qu.: -0.07604
##  Median : -0.02902   Median :-0.533611   Median : -0.07604
##  Mean   :  0.00000   Mean   : 0.000000   Mean   :  0.00000
##  3rd Qu.: -0.01797   3rd Qu.: 0.308853   3rd Qu.: -0.04018
##  Max.   :165.20520   Max.   : 4.409903   Max.   :192.70991
##    prop_borrow.V1      total_repay.V1       prop_repay.V1
##  Min.   :-0.607299   Min.   : -0.58188   Min.   :-0.548497
##  1st Qu.:-0.607299   1st Qu.: -0.15136   1st Qu.:-0.548497
##  Median :-0.607299   Median : -0.15136   Median :-0.548497
##  Mean   : 0.000000   Mean   :  0.00000   Mean   : 0.000000
##  3rd Qu.: 0.472519   3rd Qu.: -0.06830   3rd Qu.: 0.348512
##  Max.   : 5.604328   Max.   :133.03588   Max.   : 7.265537
##  prop_liquidation.V1  total_deposit.V1     prop_deposit.V1
##  Min.   :-0.148911   Min.   : -0.12047   Min.   :-1.336251
##  1st Qu.:-0.148911   1st Qu.: -0.02140   1st Qu.:-0.918390
##  Median :-0.148911   Median : -0.01801   Median : 0.053378
##  Mean   : 0.000000   Mean   :  0.00000   Mean   : 0.000000
##  3rd Qu.:-0.148911   3rd Qu.: -0.01270   3rd Qu.: 0.864931
##  Max.   :17.564940   Max.   :194.36031   Max.   : 4.354092
##    total_redeem.V1     prop_redeem.V1       prop_swap.V1
##  Min.   : -0.09703   Min.   :-0.875874   Min.   :-0.122610
##  1st Qu.: -0.02269   1st Qu.:-0.875874   1st Qu.:-0.122610
##  Median : -0.01964   Median :-0.524572   Median :-0.122610
##  Mean   :  0.00000   Mean   : 0.000000   Mean   : 0.000000
##  3rd Qu.: -0.01414   3rd Qu.: 1.103412   3rd Qu.:-0.122610
##  Max.   :142.99910   Max.   : 7.034300   Max.   :25.134235
##   prop_collateral.V1
##  Min.   :-2.2705030
##  1st Qu.:-0.6397704
##  Median : 0.0692437
##  Mean   : 0.0000000
##  3rd Qu.: 0.1728870
##  Max.   : 2.8502429
```

Now, we perform PCA on the scaled data, while removing outliers that can throw off our results .

```
df.scaled <- df.scaled[-c(41100, 44263, 44658, 14211, 20033, 6638),]
df.users <- df.users[-c(41100, 44263, 44658, 14211, 20033, 6638),]

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

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



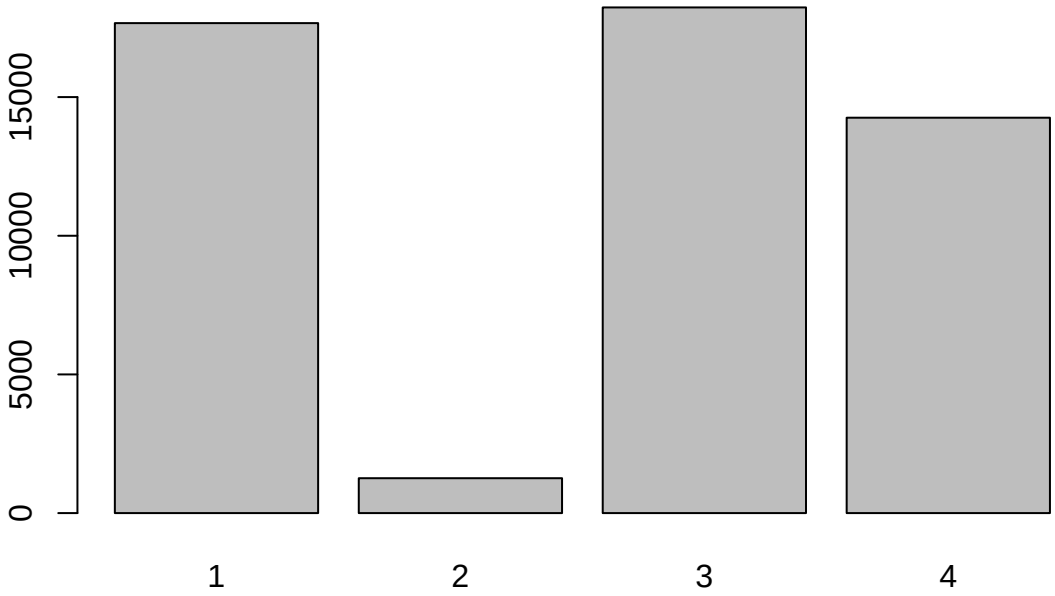

```r
#summary of pca
summary(my.pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5    PC6     PC7
## Standard deviation     1.5785 1.1968 1.0561 0.9926 0.97578 0.8763 0.82471
## Proportion of Variance 0.2603 0.1496 0.1165 0.1029 0.09945 0.0802 0.07104
## Cumulative Proportion  0.2603 0.4099 0.5264 0.6293 0.72877 0.8090 0.88001
##                           PC8     PC9    PC10    PC11    PC12    PC13
## Standard deviation     0.76698 0.69297 0.22854 0.14054 0.08141 0.04039
## Proportion of Variance 0.06144 0.05016 0.00546 0.00206 0.00069 0.00017
## Cumulative Proportion  0.94146 0.99162 0.99707 0.99914 0.99983 1.00000
```

```r
ncomps=5
```

Ne run the kmeans clustering algorithm on the data. We select the number of clusters to be equal to 4, although we could mess around with other choices to see if they better define groups.

```r
#run kmeans algorithm
set.seed(1)
km <-kmeans(df.scaled,4)
#plot frequencies of each cluster
barplot(table(km$cluster),main="Kmeans Cluster Size")
```

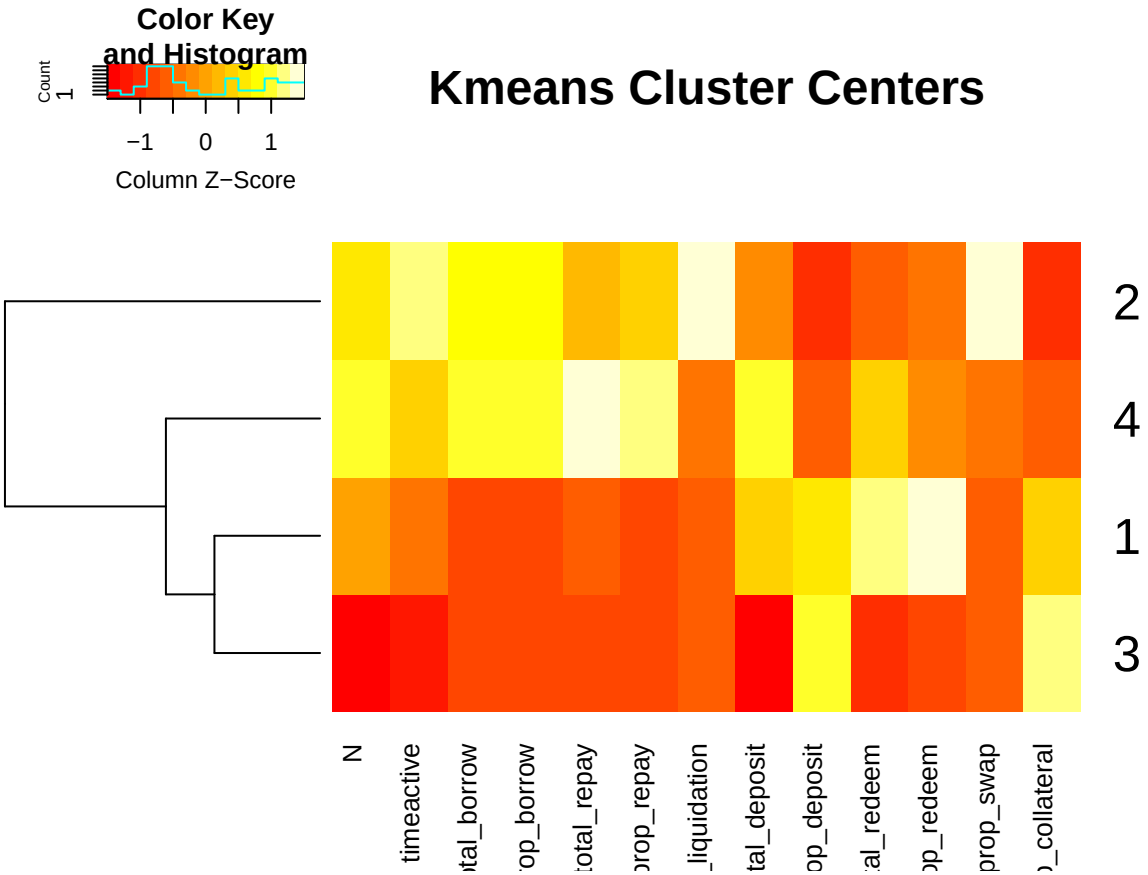



```
km$size
```

```
## [1] 17667  1258 18233 14257
```

We see that we have three fairly large clusters with one smaller one. Finally, we view the centers of the kmeans clusters.

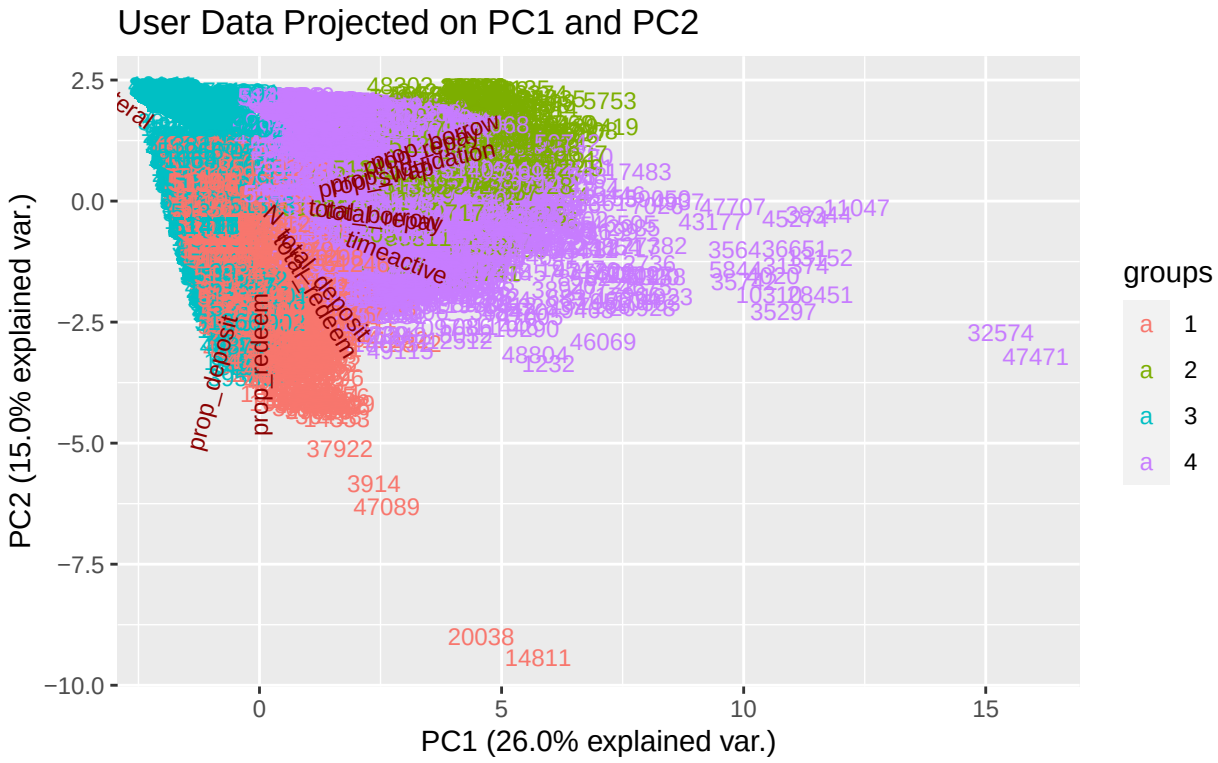
```
#make heatmap of cluster centers
heatmap.2(km$centers,
scale = "col",
dendrogram = "row",
Colv=FALSE,
cexCol=1.0,
main = "Kmeans Cluster Centers",
trace ="none")
```



We see the our smaller cluster (cluster 2), makes up the majority of liquidations. Also, the distinctions between our other larger clusters have become apparent. Cluster 1 has a lot of redeems, 4 has a lot of borrows and repays, and cluster 3 are our newer users with less transactions.

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
```



Now, for conviencence, I want to convert our timestamp into the actual date for future use.

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

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

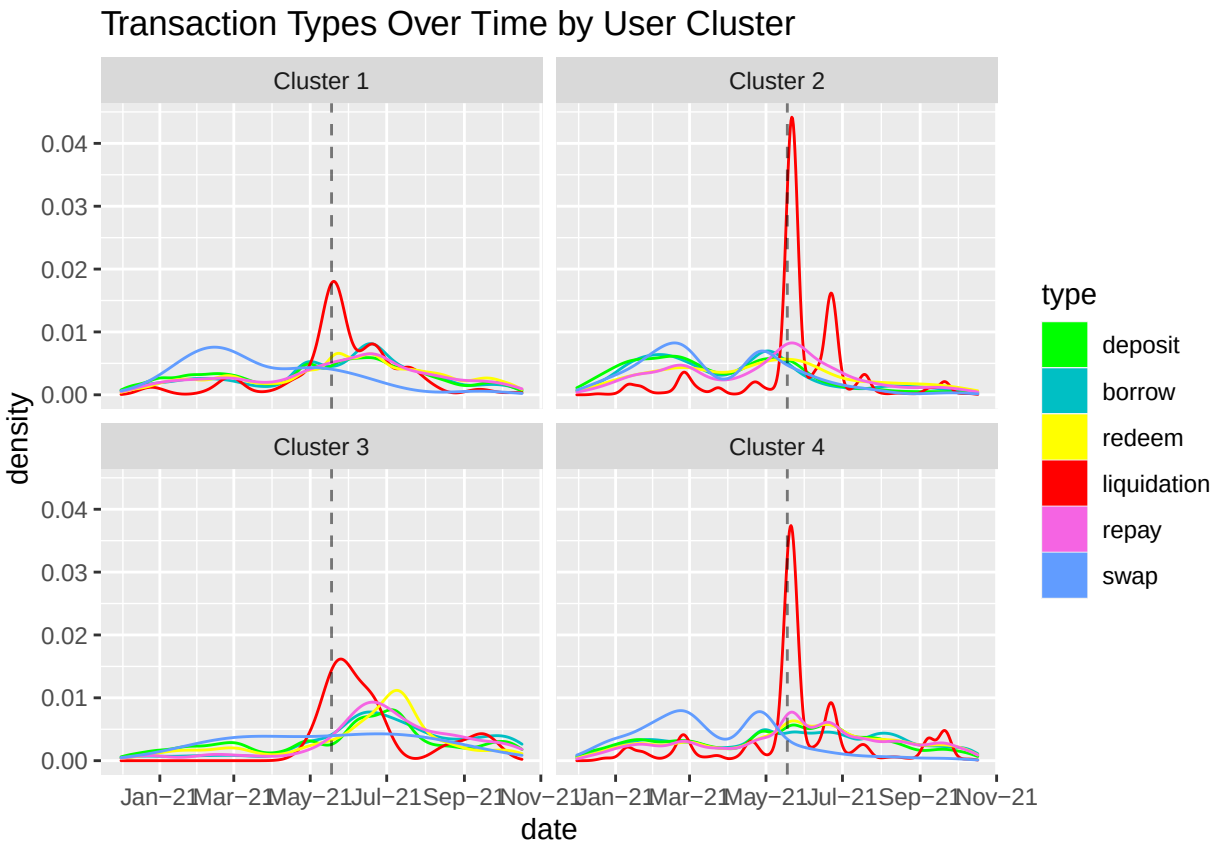
```
max(df$date)
```

```
## [1] "2021-10-17"
```

The next set of code adds the cluster as a column in our origininal dataframe will allow us to do charts by cluster. Also, I use custom colors that we will see in the next graph.

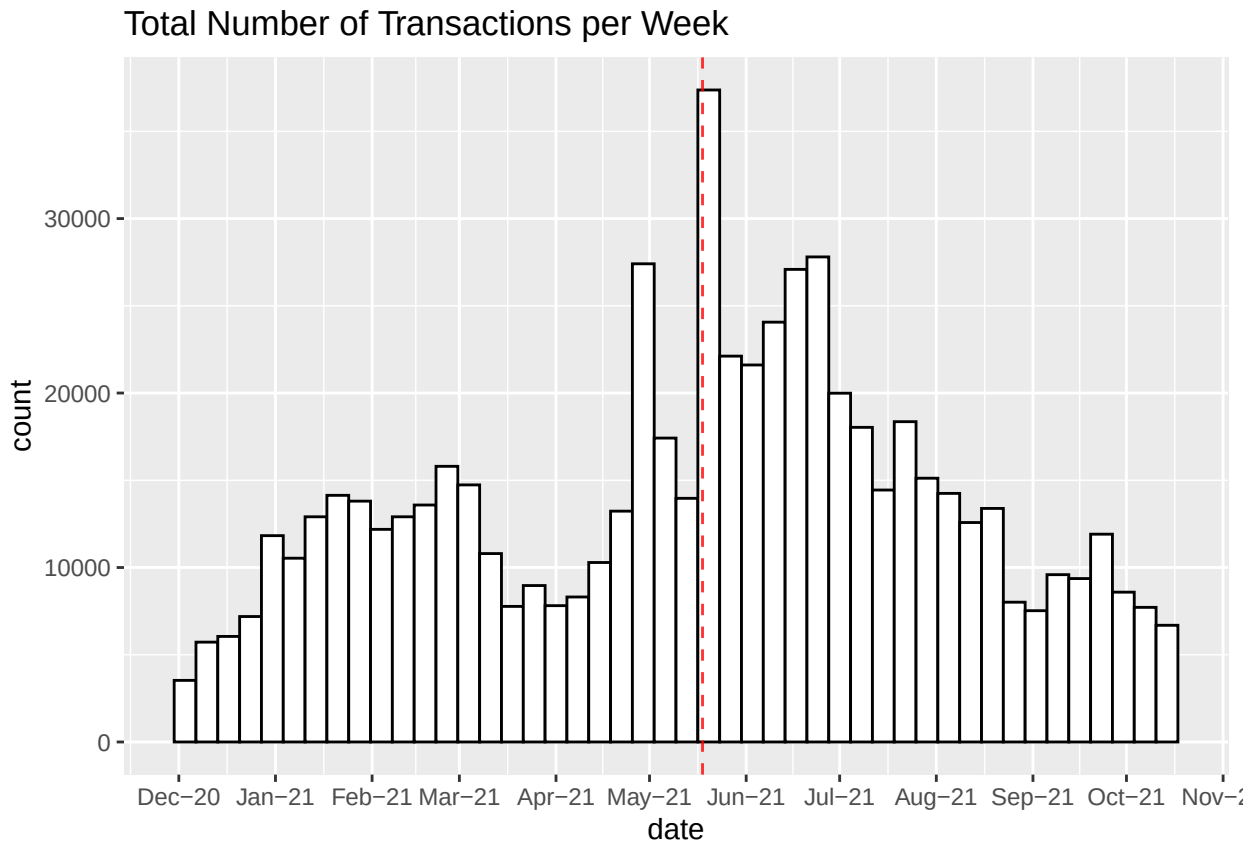
```
df.clusters <- data.frame(user = df.users$user, cluster = km$cluster)
df <- left_join(df, df.clusters, by = "user")
gg_color_hue <- function(n) {
  hues = seq(15, 375, length = n + 1)
  hcl(h = hues, l = 65, c = 100)[1:n]
}
# 6-list of ggplot colors explicitly specified
pgg <- gg_color_hue(6)
```

```
cluster_names <- c('1' = "Cluster 1", '2' = "Cluster 2", '3' = "Cluster 3", '4' = "Cluster 4")
ggplot(data = df[!(is.na(df$cluster)) & (df$type != "collateral"), ], aes(x = date,  group = type, colo
  geom_density()+
  ggtitle("Transaction Types Over Time by User Cluster")+
  geom_vline(xintercept = as.numeric(as.Date("2021-05-18")), linetype=2, alpha = 0.5, color = "black")+
  scale_x_date(date_breaks = "2 months", date_labels = "%b-%y")+
  scale_color_manual("type", values = c("deposit"="green","borrow" = pgg[4], "redeem" = "yellow", "liqu
  scale_fill_manual("type", values = c("deposit"="green","borrow" = pgg[4], "redeem" = "yellow", "liqui
  facet_wrap(~ cluster, labeller = as_labeller(cluster_names))
```

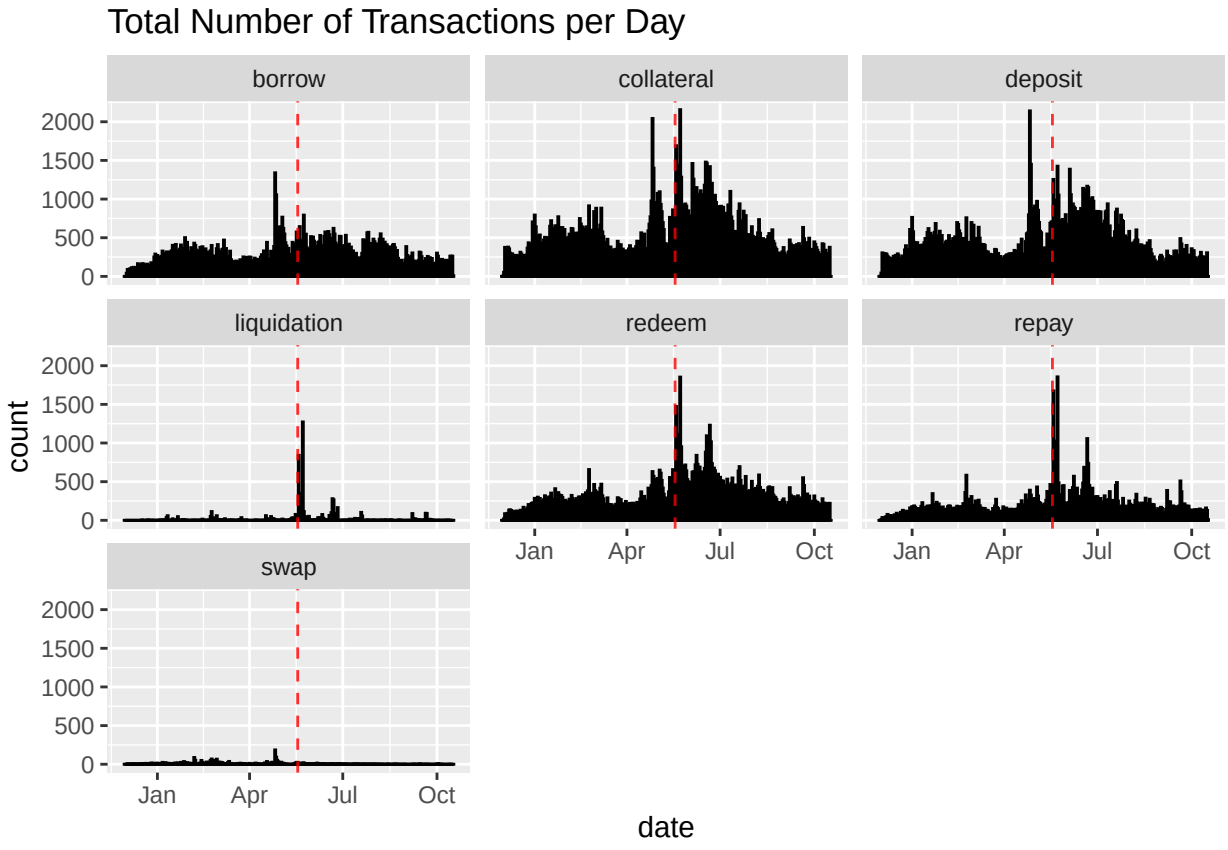


Although this is the final graph I showed in Assignment 4 it is now much more refined. Not only has the data and clustering changed, but I changed a lot of the aesthetics of the chart to make it more readable.

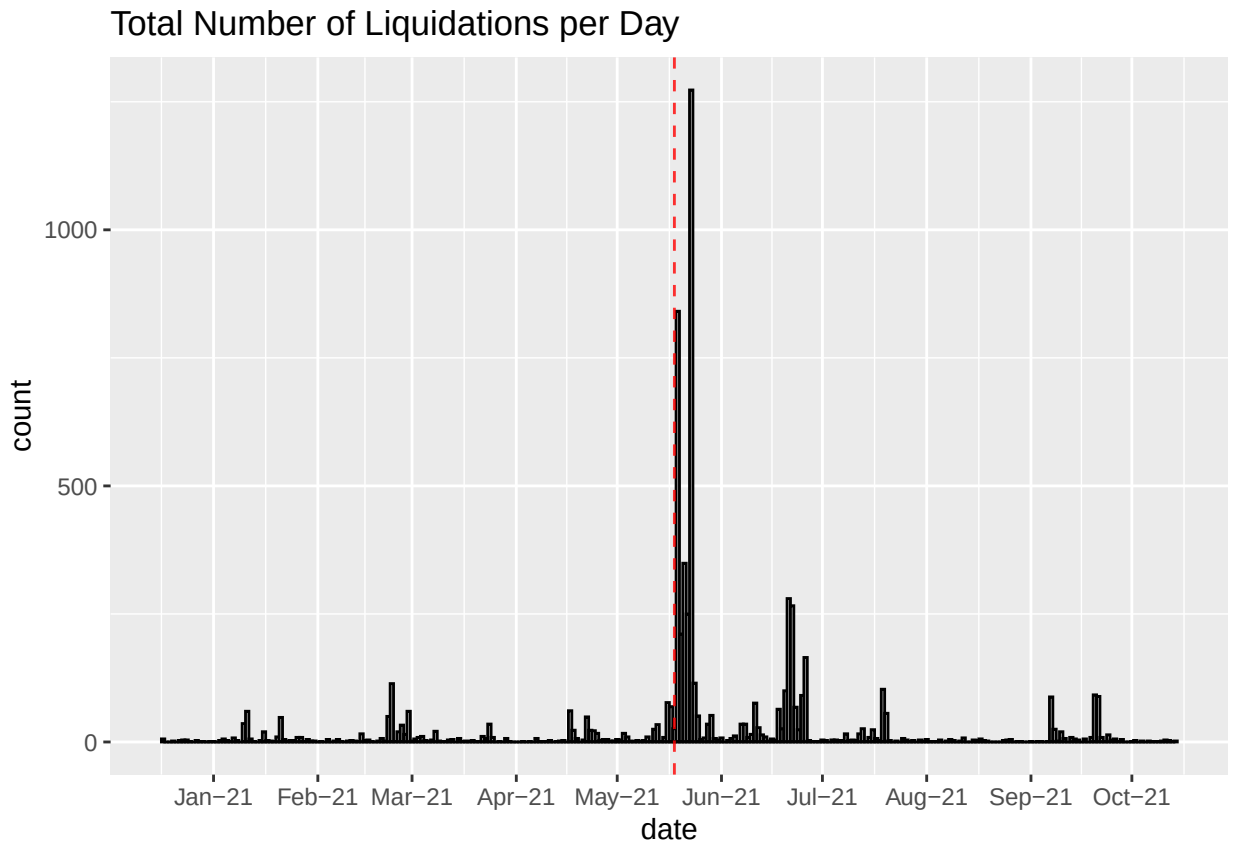
```
df <- filter(df, cluster != "NA")
ggplot(df, aes(x = date)) +
  geom_histogram(binwidth = 7, color = "black", fill = "white")+
  geom_vline(xintercept = as.numeric(as.Date("2021-05-18")), linetype=2, alpha = 0.8, color = "red")+
  scale_x_date(date_breaks = "1 month", date_labels = "%b-%y")+
  ggtitle("Total Number of Transactions per Week")
```



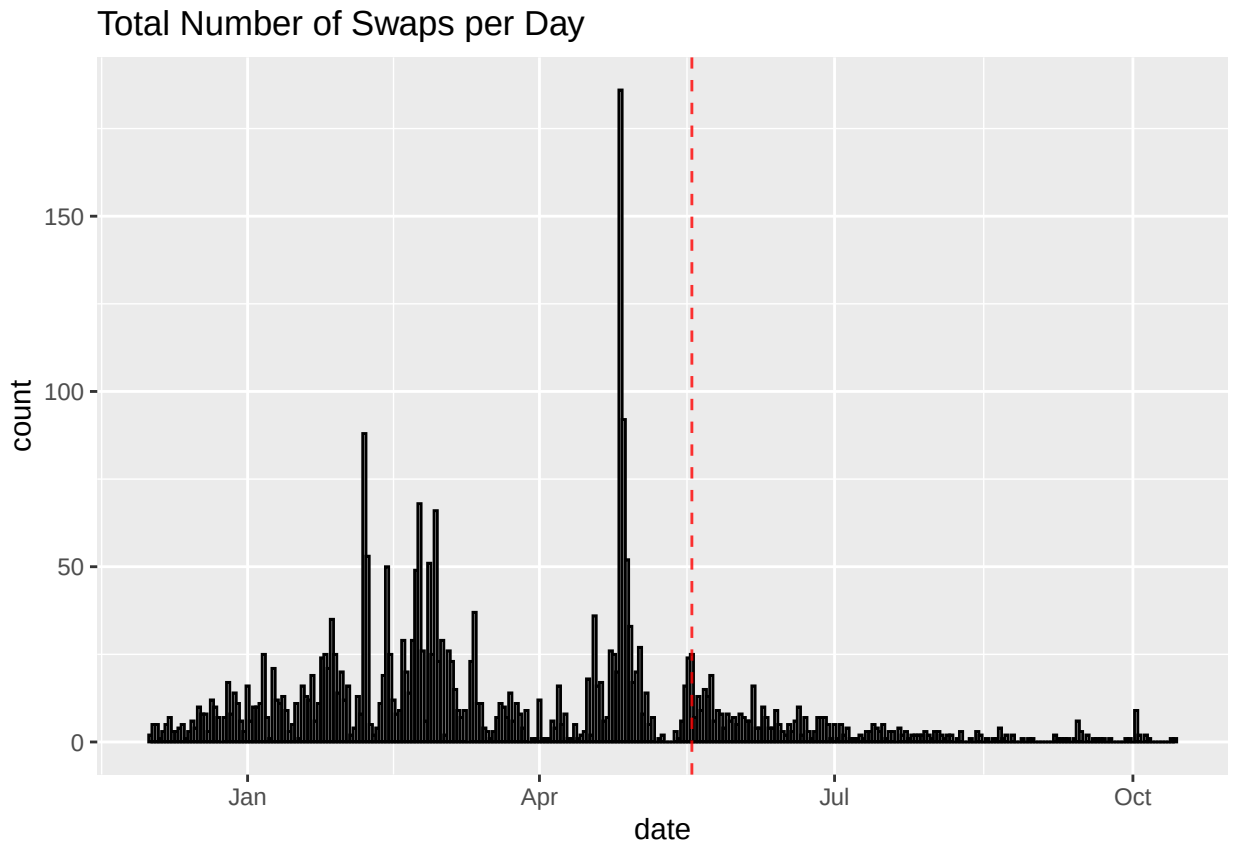

```
ggplot(df, aes(x = date)) +
  geom_histogram(binwidth = 1, color = "black", fill = "white")+
  geom_vline(xintercept = as.numeric(as.Date("2021-05-18")), linetype=2, alpha = 0.8, color = "red")+
  ggtitle("Total Number of Transactions per Day")+
  facet_wrap(~ type)
```




```
ggplot(filter(df, type == "liquidation"), aes(x = date)) +
  geom_histogram(binwidth = 1, color = "black", fill = "white")+
  geom_vline(xintercept = as.numeric(as.Date("2021-05-18")), linetype=2, alpha = 0.8, color = "red")+
    scale_x_date(date_breaks = "1 month", date_labels = "%b-%y")+
  ggtitle("Total Number of Liquidations per Day")
```

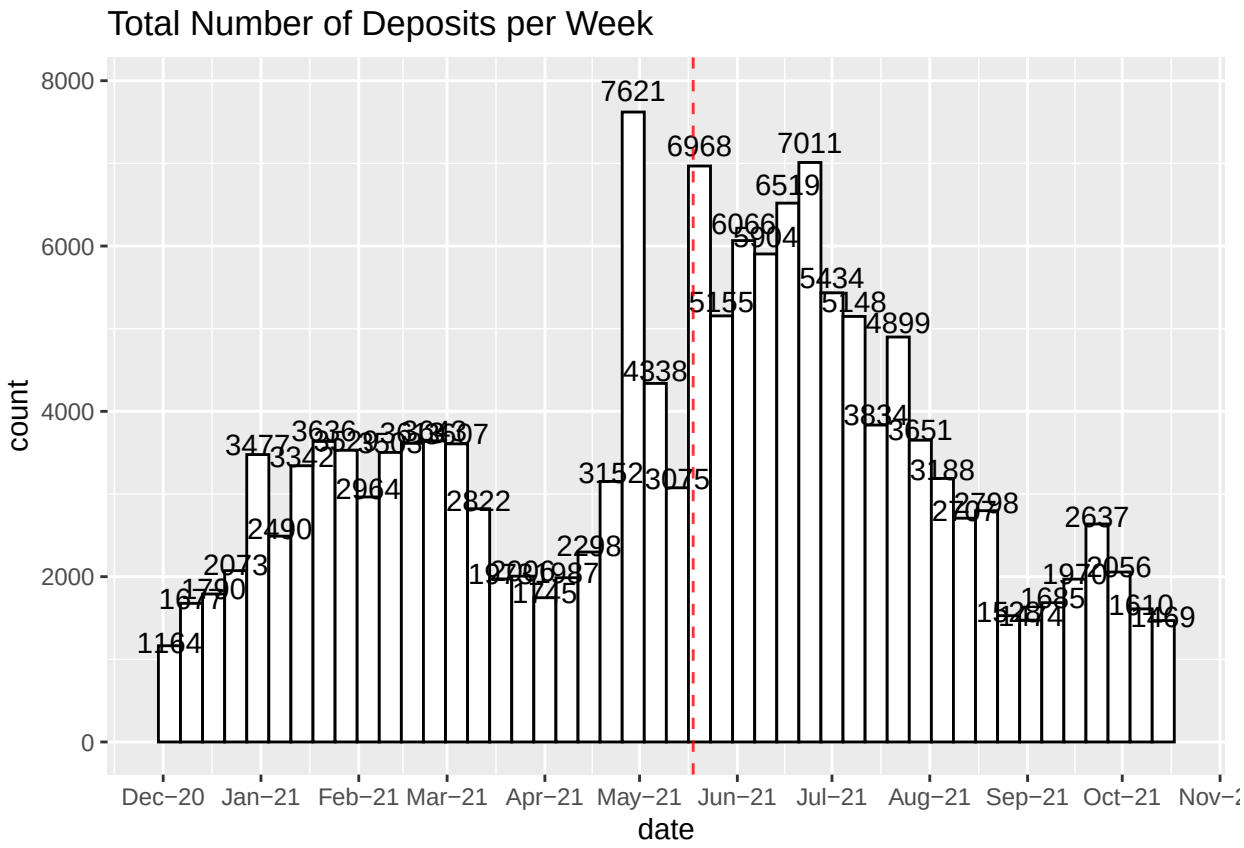



```
ggplot(filter(df, type == "swap"), aes(x = date)) +
  geom_histogram(binwidth = 1, color = "black", fill = "white")+
  geom_vline(xintercept = as.numeric(as.Date("2021-05-18")), linetype=2, alpha = 0.8, color = "red")+
  ggtitle("Total Number of Swaps per Day")
```



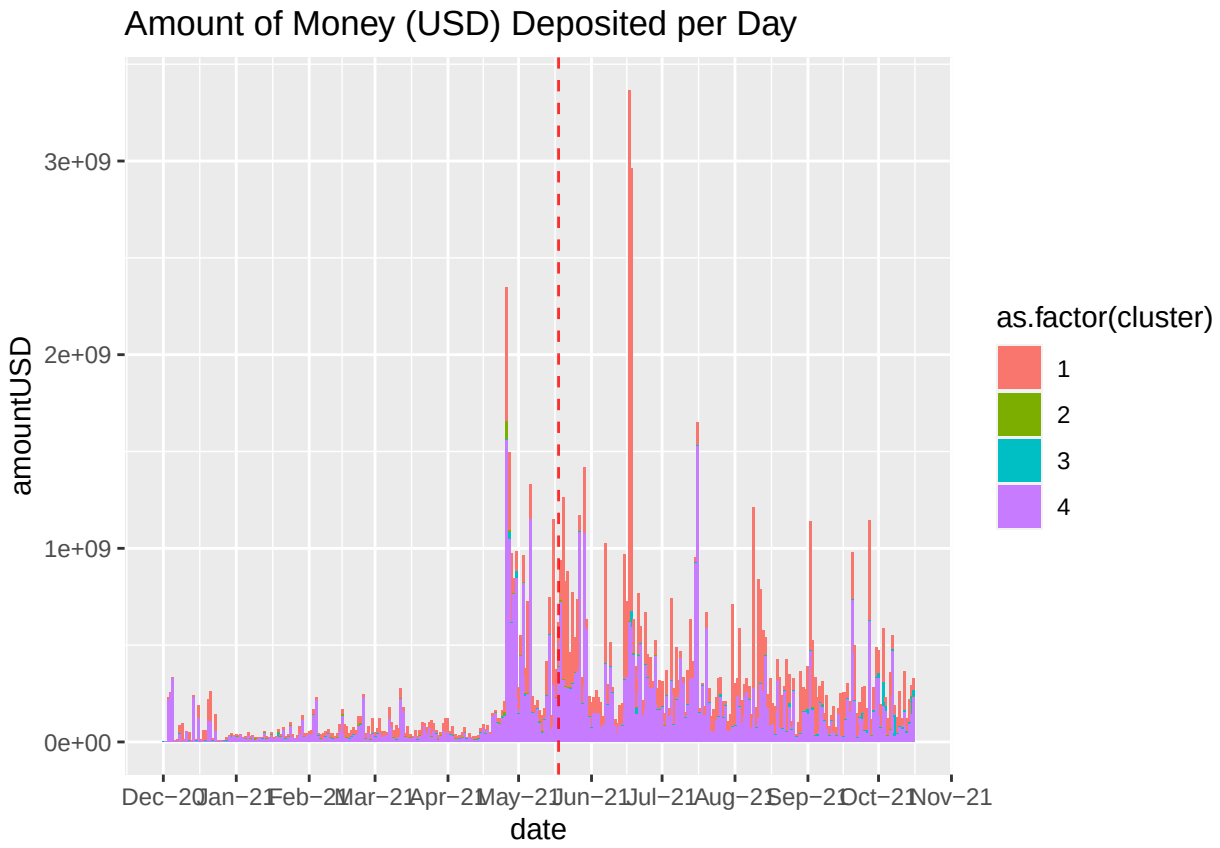
The previous charts help to show what the density plots cannot. They show the true scale of the transactions over time. They also help to show us the general trends in the data and how the shock events have impacted the different transaction types. However, one drawback of this is that it is tough to visualize an individual line of the chart.

```
ggplot(filter(df, type == "deposit"), aes(x = date)) +
  geom_histogram(binwidth = 7, color = "black", fill = "white")+
  stat_bin(binwidth=7, geom='text', color='black', aes(label=..count..),
           position=position_stack(vjust = 1.035))+
  geom_vline(xintercept = as.numeric(as.Date("2021-05-18")), linetype=2, alpha = 0.8, color = "red")+
  ggtitle("Total Number of Deposits per Week")+
  scale_x_date(date_breaks = "1 month", date_labels = "%b-%y")
```



We can also do this by week, which gives us a much less cluttered chart. While I still need to find a way to make these labels look better, we can see how this shows us that deposits spiked this summer but have fallen since. This seems to be a big more of a drop than it was in other transaction types.

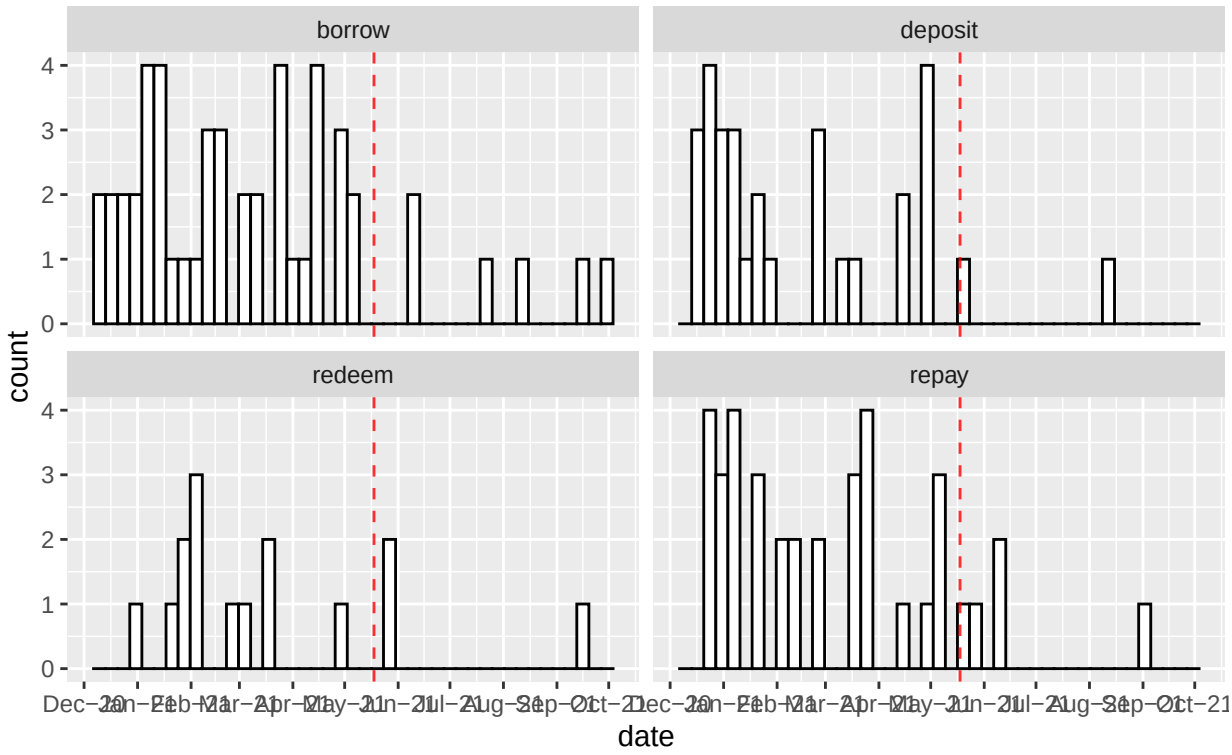
```
ggplot(data = filter(df, type == "deposit"), aes(x = date, y = amountUSD, fill = as.factor(cluster)))+
  geom_col()+
  geom_vline(xintercept = as.numeric(as.Date("2021-05-18")), linetype=2, alpha = 0.8, color = "red")+
  ggtitle("Amount of Money (USD) Deposited per Day")+
  scale_x_date(date_breaks = "1 month", date_labels = "%b-%y")
```



What is interesting about this graph is that it shows us that even though the number of deposits is lower than in the beginning of 2021, the amount of money being deposited is still higher. This also does a good job of showing that there are specific days where a lot of money is being deposited. Now, I will get into showing how our larger users have acted over time.

```
threshold <- 100
user_freq <- df %>%
  count(onBehalfOf_alias)
user_freq <- filter(user_freq, n >= threshold)
top_users <- user_freq$onBehalfOf_alias
```

```
#Get a random user with more than 100 transactions and graph their use
example_user <- sample(top_users, 1)
user_transactions <- df %>%
  filter(onBehalfOf_alias == example_user)
ggplot(user_transactions, aes(x = date)) +
  geom_histogram(binwidth = 7, color = "black", fill = "white")+
  geom_vline(xintercept = as.numeric(as.Date("2021-05-18")), linetype=2, alpha = 0.8, color = "red")+
  ggtitle(example_user, "Total Number of Transactions per Day")+
  scale_x_date(date_breaks = "1 month", date_labels = "%b-%y")+
  facet_wrap(~ type)
```




```
first <- min(as.Date(user_transactions$date))
recent <- max(user_transactions$date)
num <- nrow(user_transactions)
most_active <- names(sort(-table(user_transactions$date)))[1]
cat(paste(example_user))
```

```
## Dale Dickens
```

```
cat(paste("\nFirst Transaction: ", first))
```

```
##
## First Transaction:  2020-12-08
```

```
cat(paste("\nMost Recent Transaction: ", recent))
```

```
##
## Most Recent Transaction:  2021-09-29
```

```
cat(paste("\nNumber of Transactions: ", num))
```

```
##
## Number of Transactions:  132
```

```
cat(paste("\nMost Active Day: ", most_active))
```

```
##
## Most Active Day:  2020-12-26
```

```
types_counts <- user_transactions %>%
  count(type) %>%
  mutate(n / (as.numeric(recent - first)/7))
```

```
types_counts <- rename(types_counts, "Total" = "n")
types_counts <- rename(types_counts, "Weekly Average" = "n/(as.numeric(recent - first)/7)")
kable(types_counts)
```

type	Total	Weekly Average
borrow	50	1.1864407
deposit	30	0.7118644
redeem	15	0.3559322
repay	37	0.8779661

We can also find different statistics from their use that will be applied into creating my weekly dataframe. I can't comment too much on the results of these graphs because it's random and setting the seed would defeat the purpose, however the threshold helps to make sure there is enough to visualize. Now I will start making my weekly dataframe.

#Weekly User Analysis

```
df.weekly<- df%>%group_by(user, user_alias)%>%
  summarise(timefirst=min(anydate(timestamp)), timelast=max(anydate(timestamp)), N=n())
```

```
## `summarise()` has grouped output by 'user'. You can override using the `.groups` argument.
```

```
#get the time the user has been active
df.weekly$timeactive<-df.weekly$timelast-df.weekly$timefirst
#get amounts for columns
df$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.weekly<-merge(x=df.weekly,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.weekly<-merge(x=df.weekly,y=select(df.type,user,ntypes),by="user",all.x=TRUE)
}
df.weekly <- rename(df.weekly, "name" = "user_alias")

head(df.weekly)
```

```
##           user             name  timefirst   timelast  N timeactive
## 1 1.000000e+00    Tracy Nisbett 2021-07-22 2021-07-22  1     0 days
## 2 5.700500e+04 Millicent Gandhi 2021-02-22 2021-07-12  2   140 days
```

```
## 3 2.577533e+33  Steven Ferguson 2021-02-13 2021-10-09 60   238 days
## 4 6.663597e+34   Ronald Masella 2021-05-29 2021-06-01 10     3 days
## 5 4.867107e+35   Steven Ahumada 2021-04-25 2021-04-25  4     0 days
## 6 8.009427e+35   Melissa Esters 2020-12-05 2020-12-25  4    20 days
##   total_borrow nborrow total_repay nrepay total_liquidation nliquidation
## 1           NA      NA          NA     NA                NA           NA
## 2           NA      NA          NA     NA                NA           NA
## 3           NA      NA          NA     NA                NA           NA
## 4     9.447565       2    9.417525      2                NA           NA
## 5           NA      NA          NA     NA                NA           NA
## 6           NA      NA          NA     NA                NA           NA
##   total_deposit ndeposit total_redeem nredeem total_swap nswap total_collateral
## 1            NA       NA           NA      NA         NA    NA               NA
## 2            NA       NA           NA      NA         NA    NA               NA
## 3     64.160153       22    90.345340      32         NA    NA               NA
## 4     10.009139        2    10.008833       2         NA    NA               NA
## 5      3.997229        1     3.997237       1         NA    NA               NA
## 6      4.910344        1     4.914500       1         NA    NA               NA
##   ncollateral
## 1           1
## 2           2
## 3           6
## 4           2
## 5           2
## 6           2
```

```
df.weekly$timeactive =as.numeric( df.weekly$timeactive / 7)
df.weekly <- rename(df.weekly, "weeks" = "timeactive")
```

This is similar to making the user dataframe, but now I will get all of the data into weekly terms as opposed to consolidated transaction data.

```
one_day <- df.weekly %>%
  filter(weeks == 0)
df.weekly <- df.weekly %>%
  filter(weeks > 0)
df.weekly$N =df.weekly$N / df.weekly$weeks
df.weekly[,7:20] <- df.weekly[,7:20] / df.weekly$weeks
one_day$N <- one_day$N * 7
one_day[,7:20] <- one_day[,7:20] * 7
df.weekly <- rbind(df.weekly, one_day)
df.weekly <- rename(df.weekly, "weekly_n" = "N")
head(df.weekly)
```

```
##           user             name  timefirst   timelast  weekly_n      weeks
## 1 5.700500e+04 Millicent Gandhi 2021-02-22 2021-07-12  0.100000 20.0000000
## 2 2.577533e+33  Steven Ferguson 2021-02-13 2021-10-09  1.764706 34.0000000
## 3 6.663597e+34   Ronald Masella 2021-05-29 2021-06-01 23.333333  0.4285714
## 4 8.009427e+35   Melissa Esters 2020-12-05 2020-12-25  1.400000  2.8571429
## 5 1.358443e+37     Howard Brian 2021-07-25 2021-07-28 11.666667  0.4285714
## 6 3.732290e+40   James Carlisle 2021-05-14 2021-08-18  5.687500 13.7142857
##   total_borrow  nborrow total_repay    nrepay total_liquidation nliquidation
## 1           NA       NA          NA        NA                NA           NA
## 2           NA       NA          NA        NA                NA           NA
## 3    22.044318 4.666667   21.974226 4.6666667                NA           NA
```

```
## 4           NA       NA          NA        NA                NA           NA
## 5    13.504313 4.666667    7.153202 2.3333333                NA           NA
## 6     8.735614 1.750000    2.356094 0.5104167                NA           NA
##   total_deposit  ndeposit total_redeem   nredeem total_swap nswap
## 1            NA        NA           NA        NA         NA    NA
## 2      1.887063 0.6470588     2.657216 0.9411765         NA    NA
## 3     23.354658 4.6666667    23.353944 4.6666667         NA    NA
## 4      1.718620 0.3500000     1.720075 0.3500000         NA    NA
## 5            NA        NA           NA        NA         NA    NA
## 6      9.802558 1.9687500     3.508889 0.7291667         NA    NA
##   total_collateral ncollateral
## 1               NA   0.1000000
## 2               NA   0.1764706
## 3               NA   4.6666667
## 4               NA   0.7000000
## 5               NA   4.6666667
## 6               NA   0.7291667
```

```
liquids <- df.weekly %>%
  filter(nliquidation > 0)
```

We can see that many of our rows have been changed. I'm still looking for more to put in this dataframe as it is certainly not a finished product. Similarly to the users, I now perform PCA and clustering to try to see how good these variables are.

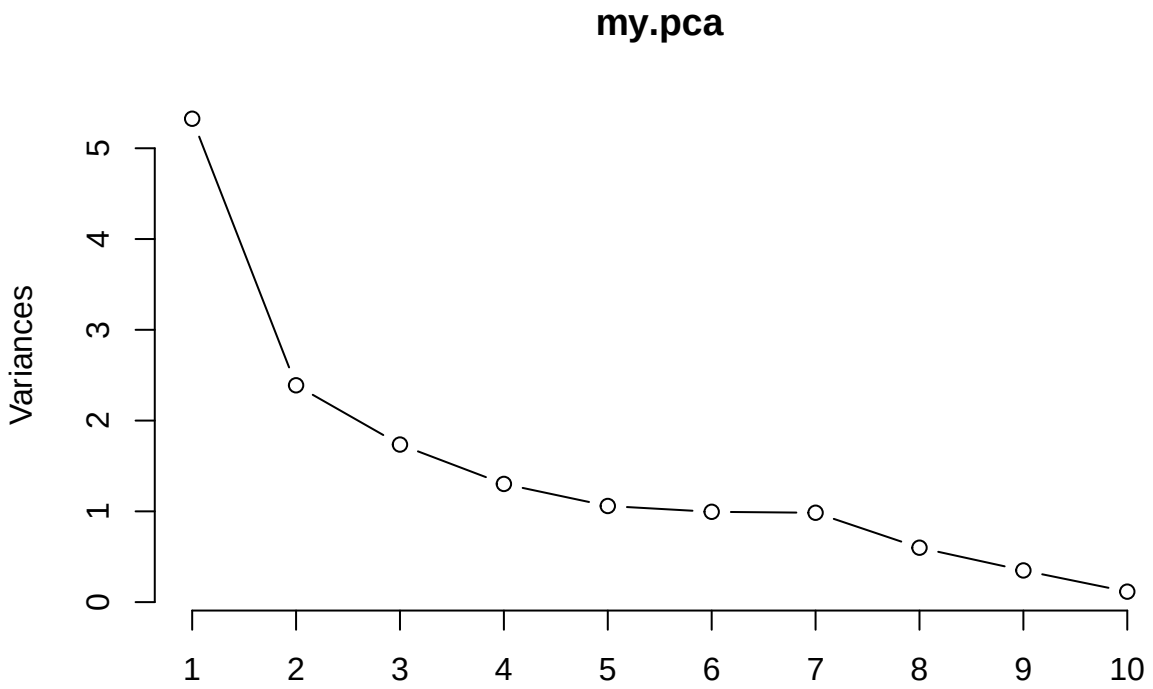
```
df.sub<-select(df.weekly,-c(user,name, total_liquidation, total_swap, total_collateral))
#repalce missing values as 0's
df.sub[is.na(df.sub)] <- 0
#scale data
df.scaled<-df.sub%>%mutate_all(scale)

summary(df.scaled)
```

```
##      timefirst.V1         timelast.V1         weekly_n.V1
##  Min.   :-1.8272760   Min.   :-2.3258258   Min.   :-0.57167
##  1st Qu.:-0.8985930   1st Qu.:-0.5850006   1st Qu.:-0.50560
##  Median : 0.0783333   Median : 0.0845475   Median :-0.22118
##  Mean   : 0.0000000   Mean   : 0.0000000   Mean   : 0.00000
##  3rd Qu.: 0.7175566   3rd Qu.: 0.7784429   3rd Qu.: 0.27367
##  Max.   : 2.0442467   Max.   : 1.5575535   Max.   :74.25928
##       weeks.V1        total_borrow.V1       nborrow.V1       total_repay.V1
##  Min.   :-0.660794   Min.   :-5.04899   Min.   :-0.28730   Min.   :-4.99129
##  1st Qu.:-0.660794   1st Qu.:-0.27658   1st Qu.:-0.28730   1st Qu.:-0.21186
##  Median :-0.533564   Median :-0.27658   Median :-0.28730   Median :-0.21186
##  Mean   : 0.000000   Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.00000
##  3rd Qu.: 0.309332   3rd Qu.:-0.18277   3rd Qu.:-0.19334   3rd Qu.:-0.16927
##  Max.   : 4.380677   Max.   :70.29485   Max.   :74.69806   Max.   :44.91449
##      nrepay.V1        nliquidation.V1    total_deposit.V1       ndeposit.V1
##  Min.   :-0.22272   Min.   : -0.02660   Min.   :-14.60598   Min.   : -0.37810
##  1st Qu.:-0.22272   1st Qu.: -0.02660   1st Qu.: -0.33794   1st Qu.: -0.37207
##  Median :-0.22272   Median : -0.02660   Median : -0.28208   Median : -0.31992
##  Mean   : 0.00000   Mean   :  0.00000   Mean   :  0.00000   Mean   :  0.00000
##  3rd Qu.:-0.17669   3rd Qu.: -0.02660   3rd Qu.:  0.15452   3rd Qu.:  0.61091
##  Max.   :45.80048   Max.   :214.11367   Max.   :134.31993   Max.   :121.55280
##   total_redeem.V1       nredeem.V1          nswap.V1         ncollateral.V1
```

```
##  Min.   :-6.09382   Min.   :-0.24537   Min.   : -0.04726   Min.   :-0.72507
##  1st Qu.:-0.22357   1st Qu.:-0.24537   1st Qu.: -0.04726   1st Qu.:-0.66594
##  Median :-0.21753   Median :-0.23685   Median : -0.04726   Median :-0.40407
##  Mean   : 0.00000   Mean   : 0.00000   Mean   :  0.00000   Mean   : 0.00000
##  3rd Qu.:-0.13241   3rd Qu.:-0.14212   3rd Qu.: -0.04726   3rd Qu.: 0.39842
##  Max.   :90.86916   Max.   :94.31349   Max.   :118.31172   Max.   :63.58153
```

Now, we perform PCA on the scaled data.

```
#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     2.308 1.5456 1.3175 1.1410 1.0291 0.99740 0.99229
## Proportion of Variance 0.355 0.1593 0.1157 0.0868 0.0706 0.06632 0.06564
## Cumulative Proportion  0.355 0.5143 0.6300 0.7168 0.7874 0.85370 0.91934
##                           PC8     PC9   PC10    PC11    PC12    PC13      PC14
## Standard deviation     0.77418 0.59066 0.3397 0.28624 0.19818 0.15824 8.615e-15
## Proportion of Variance 0.03996 0.02326 0.0077 0.00546 0.00262 0.00167 0.000e+00
## Cumulative Proportion  0.95930 0.98255 0.9902 0.99571 0.99833 1.00000 1.000e+00
##                           PC15
## Standard deviation     1.626e-16
## Proportion of Variance 0.000e+00
## Cumulative Proportion  1.000e+00
```

```
ncomps=5
```

For consistency, I still select 4 clusters, although it appears that 7 may be the best choice (have to try them all out).

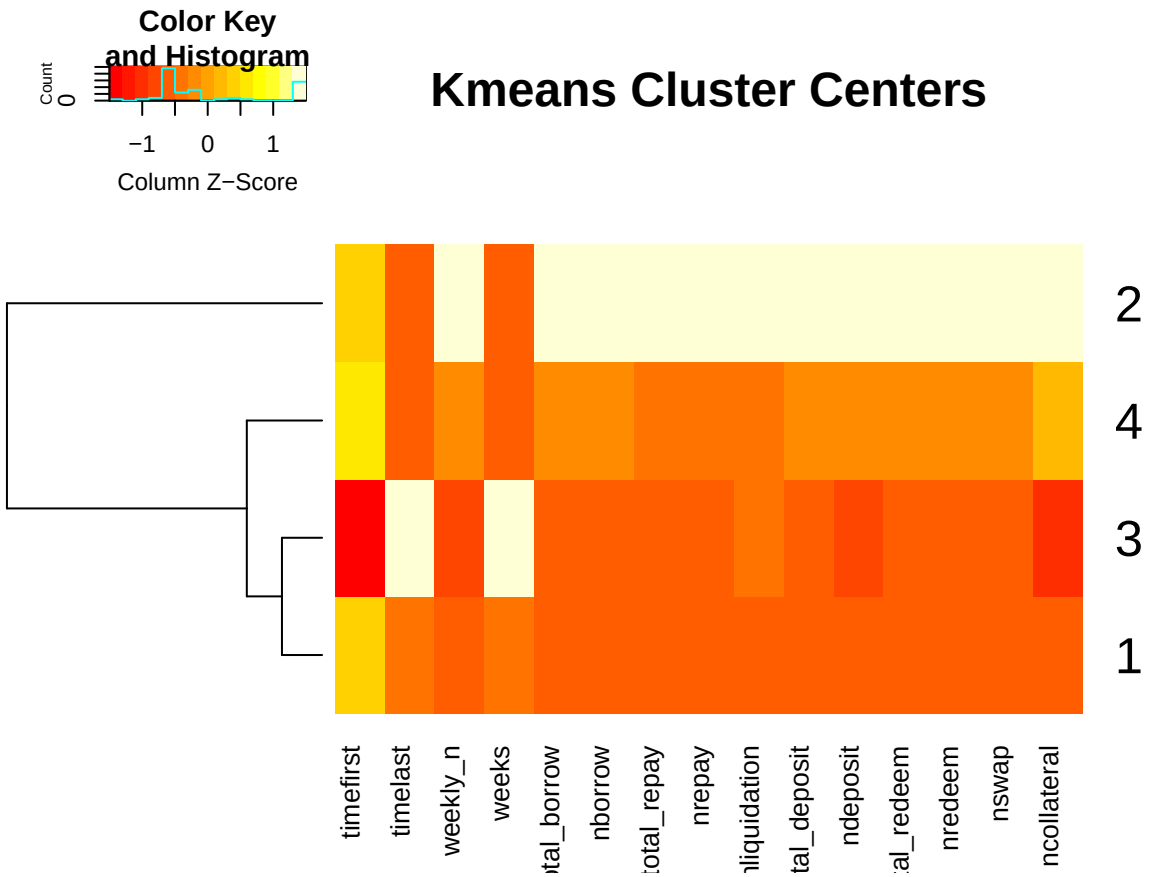
```r
#run kmeans algorithm
set.seed(1)
km <-kmeans(df.scaled,4)
#plot frequencies of each cluster
barplot(table(km$cluster),main="Kmeans Cluster Size")
```


```r
km$size
```

```
## [1] 32770   331 12031  6283
```

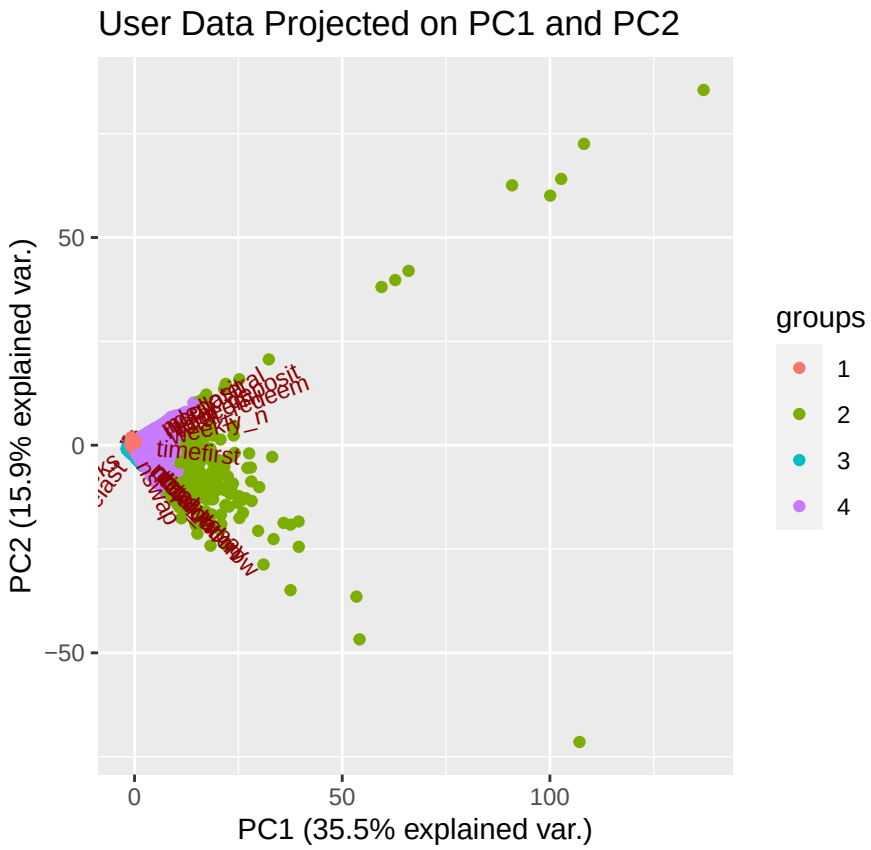
Now, we view the centers of the kmeans clusters. This shows us that Cluster 2 are very active users, while cluster three are our old users. However, cluster 2 dominates the heatmap so we can't see much about transaction types.

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



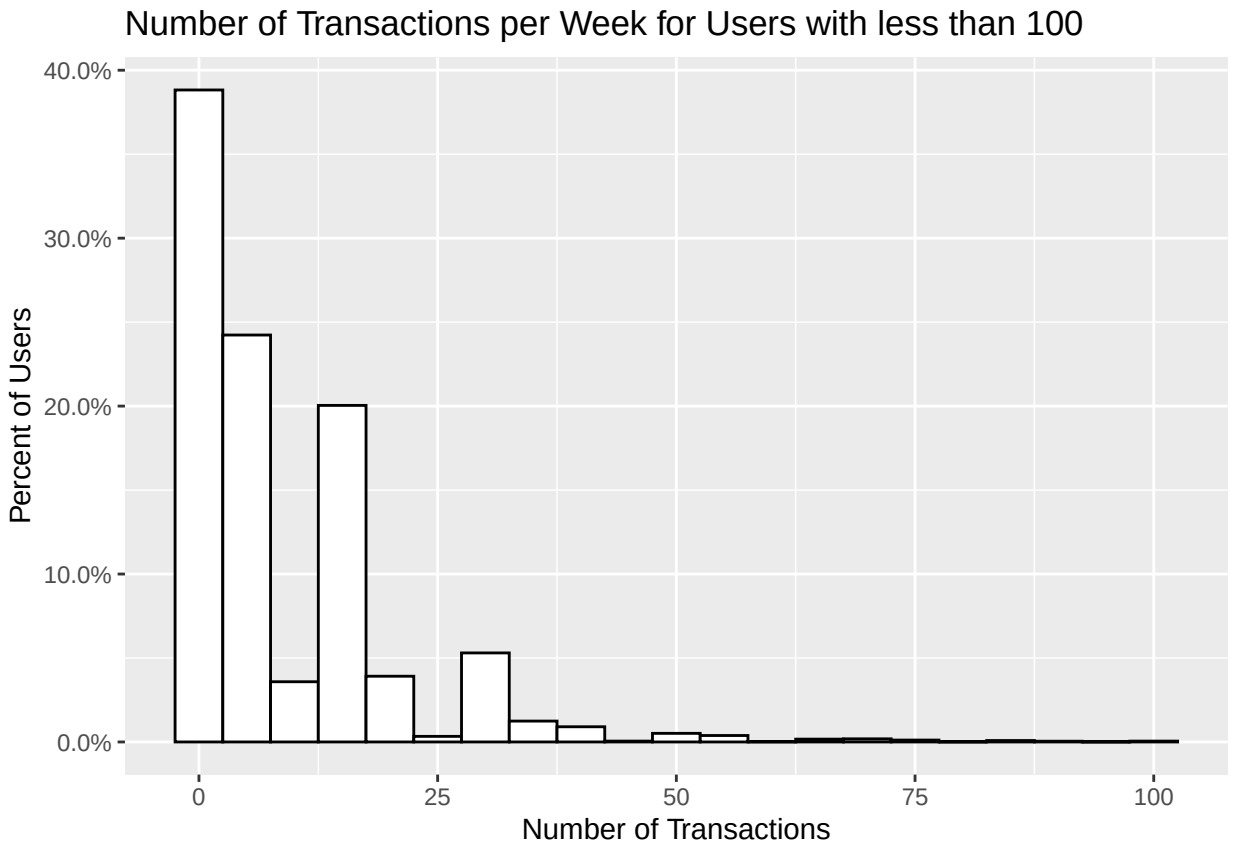
Exploring the Influential Factors with a Biplot

```
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 makes a lot of sense as we can see that most of the outliers are in cluster 2. Now I can start to look at some visualizations for our data.

```
ggplot(df.weekly %>% filter(weekly_n <= 100), aes(x = weekly_n))+
  geom_histogram(binwidth = 5, color = "black", fill = "white", aes(y = ..count../sum(..count..)))+
  ggtitle("Number of Transactions per Week for Users with less than 100")+
  scale_y_continuous(labels=percent)+
  labs(y = "Percent of Users", x = "Number of Transactions")
```



Leaving out our outliers, we can see that almost 40% of users average 5 or less transactions per week, and the majority have 20 or less. This seems to make sense for the average person. Now, I wanted to look at this by cluster.

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

```
mean(df.weekly$weekly_n)
```

```
## [1] 9.483499
```

```
for (x in 1:4){
  cluster_x <- df.clusters %>%
    filter(cluster == x)
  print(paste("Cluster",x, ":",  round(mean(cluster_x$weekly_n), 2)))
}
```

```
## [1] "Cluster 1 : 7.72"
## [1] "Cluster 2 : 111.2"
## [1] "Cluster 3 : 1.16"
## [1] "Cluster 4 : 29.28"
```

We can see that Cluster 2 is clearly our outlier. Users in cluster 3 do not have much activity, and cluster 1 appears to represent our more average user. It will be interesting to see if Cluster 1 is the representative cluster going forward.