DAR F21 Project Status Notebook

DeFi

Cole Paquin

9/30/2021

Contents


```
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)
```

```
## 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(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)
```

Weekly Work Summary

- RCS ID: Paquic
- Project Name: DeFi
- Summary of work since last week
- I changed and created new features in the users dataframe with the hope of running PCA and explaining more of the variance of the users. -Then, I used tSNE to better visualize different clusters of users and see if our KMeans clusters translate well into the higher dimensions.
- Summary of github commits
  - Branch name dar-paquic

Personal Contribution

Other than some of the original PCA creation from the Users.Rmd, all of the following code and illustrations were created by me.

Discussion of Primary Findings

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

I wanted to look into the behaviors of different users and see if we could find patterns. Going in, I assumed we would be able to see groups such as yield farmers and heavy borrowers. I believe that this could be one of the better ways to be able to identify users who may be more likely to liquidate. On top of this, clustering users is a good way to see how Aave is used and what people are looking to achieve in teh DeFi world.

```
* How did you go about finding it?
```

First, I created new features in the users dataframe. Not only did I include the proportion of each type of transaction, I also used the total number of transactions and the total amount of money featured in each type of their transaction.

```r
library(dplyr)
#load in csv file to data frame
df<- readRDS("~/transactions.Rds")
#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<-df$amountUSD
df$logCollateralUSD<-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 sum 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
  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.
```

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

```
##           user  timefirst   timelast  N timeactive total_borrow nborrow
## 1 2.577533e+33 1613241549 1628484854 49   15243305           NA      NA
## 2 6.663597e+34 1622302305 1622568495  8     266190   107352.589       2
## 3 4.867107e+35 1619325602 1619328421  2       2819           NA      NA
## 4 8.009427e+35 1607151100 1608903952  2    1752852           NA      NA
## 5 1.358443e+37 1627228884 1627464322  3     235438     2051.175       2
## 6 8.092096e+37 1616874620 1616875027  2        407           NA      NA
##   prop_borrow total_repay nrepay prop_repay total_liquidation nliquidation
## 1          NA          NA     NA         NA                NA           NA
## 2   0.2469880   104038.10      2  0.2469880                NA           NA
## 3          NA          NA     NA         NA                NA           NA
## 4          NA          NA     NA         NA                NA           NA
## 5   0.6212121     1163.21      1  0.3181818                NA           NA
```

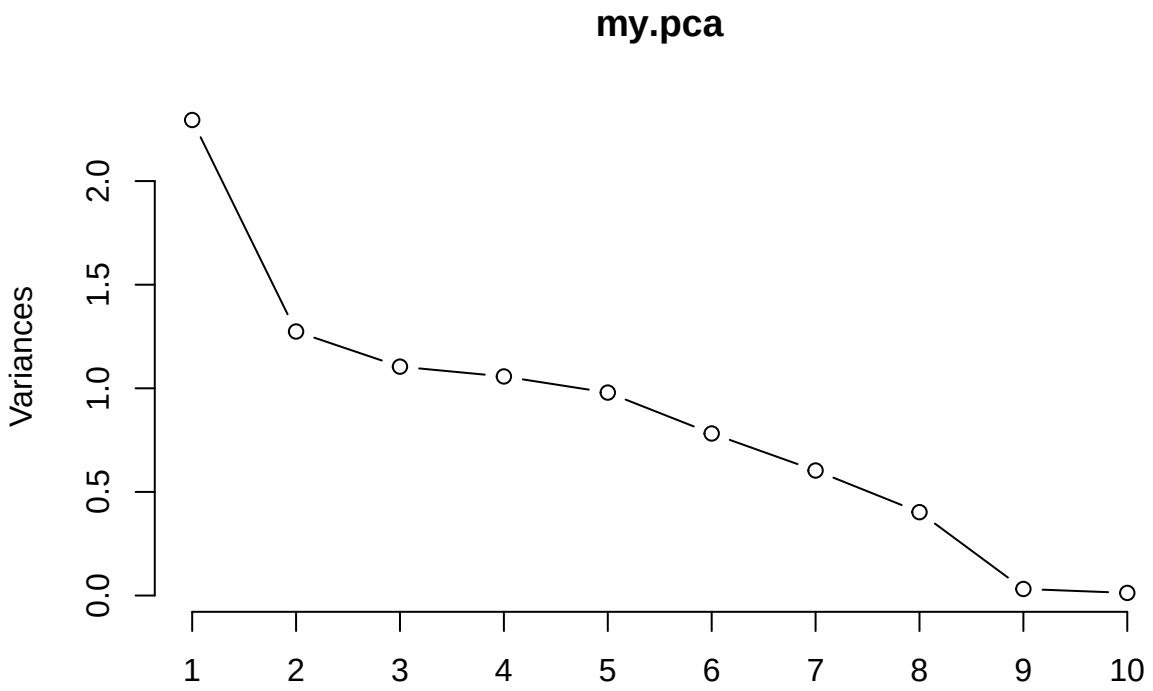
```
## 6          NA          NA     NA         NA                NA          NA
##   prop_liquidation total_deposit ndeposit prop_deposit total_redeem nredeem
## 1               NA    253300.000       22    0.4472617     28066.44      27
## 2               NA    202133.229        2    0.2469880    202071.41       2
## 3               NA      9936.395        1    0.4565217      9936.58       1
## 4               NA     81347.508        1    0.4565217     82129.56       1
## 5               NA            NA       NA           NA           NA      NA
## 6               NA     40464.301        2    0.8913043           NA      NA
##   prop_redeem total_swap nswap prop_swap
## 1   0.5486815         NA    NA        NA
## 2   0.2469880         NA    NA        NA
## 3   0.4565217         NA    NA        NA
## 4   0.4565217         NA    NA        NA
## 5          NA         NA    NA        NA
## 6          NA         NA    NA        NA
```

Then, I ran PCA and Kmeans on our scaled dataset, which is a good linear way to get and visualize clusters of the users. After this, I used tSNE (a nonlinear visualization method) to see if our original clusters translate better into the higher dimensions.

`* What did you find?`

First, I remove one outlier that throws off our PCA results. Although this case could be analyzed, it does not represent the actions of a normal user. We are interested in seeing how much variance is explained by each PCA component.

```
#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,N
#repalce missing values as 0's
df.sub<-df.sub%>%replace(is.na(.),0)
#scale data for PCA
df.scaled<-df.sub%>%mutate_all(scale)
df.scaled <- df.scaled[-c(9886),]
#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")
```

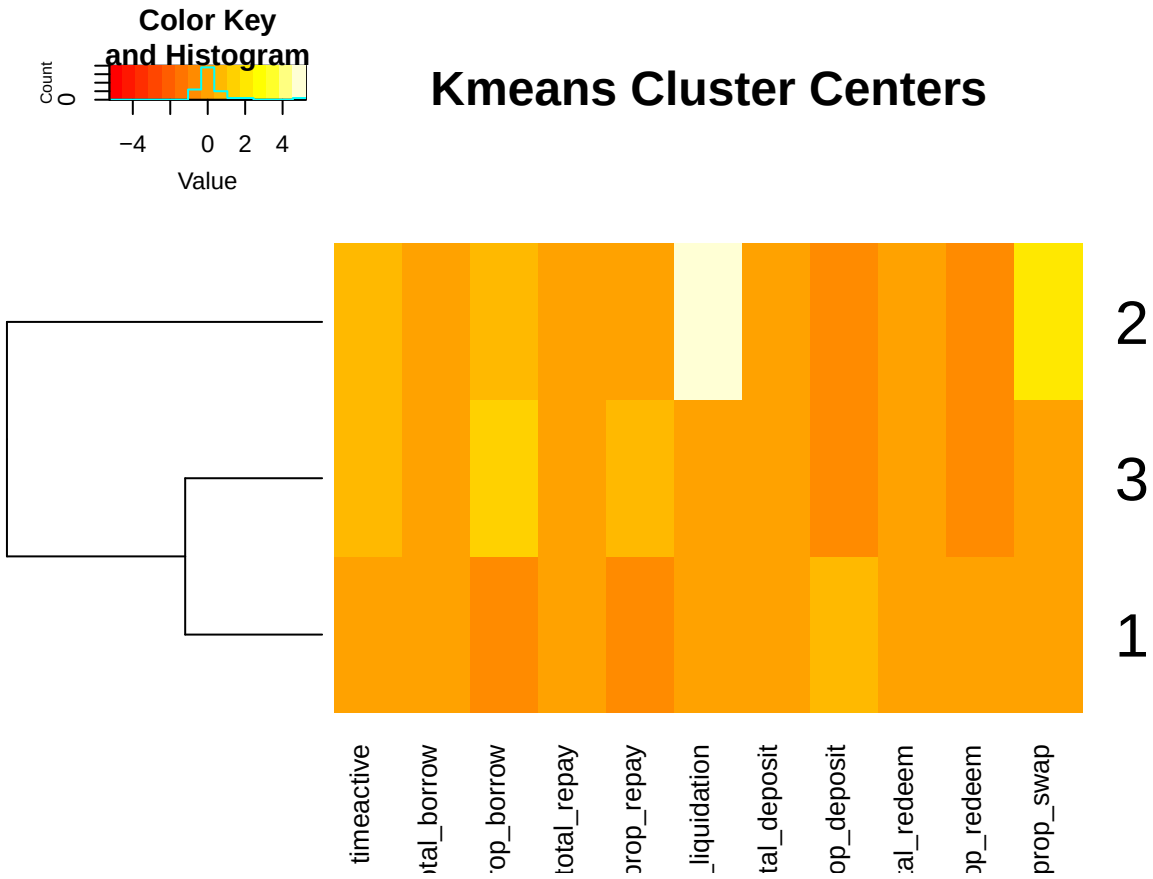


This is one big takeaway that we get from this graph. When we do our clustering, we should look for three cluster since that is where the "elbow" occurs. There is another smaller elbow around nine, however upoin further inspection this proves to be too many clusters that do not add a lot of value. Thus, we will run KMeans looking for 3 clusters.

```
#run kmeans algorithm
set.seed(1)
km <-kmeans(df.scaled,3)
#plot frequencies of each cluster
barplot(table(km$cluster),main="Kmeans Cluster Size")
```

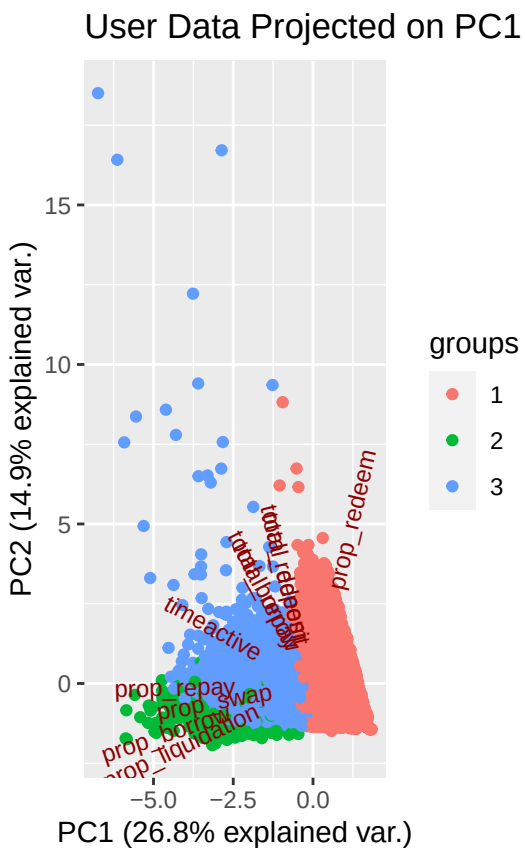
This shows the number of users that appear in each cluster. We can see that cluster two does not have too many users, and cluster three makes up more than half the data. However, until we look at the cluster centers this can not tell us too much about the user behavior.

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



This heatmap of cluster centers allows us to see what each cluster represents. Cluster 1 appears to be focused on users who borrow and repay money at a higher rate, with lower deposit and redeem ratings. Cluster 3, the largest, is made of people who deposit money but are not very active other and that. This could be the yield farmers. Finally, and perhaps most interesting is cluster 2. Cluster 2 consists of the vast majorities of swaps and liquidations. Could this be a correlation/causation? However, these people do not deposit or redeem very often. Now we want to make a biplot to better show how these clusters compare.

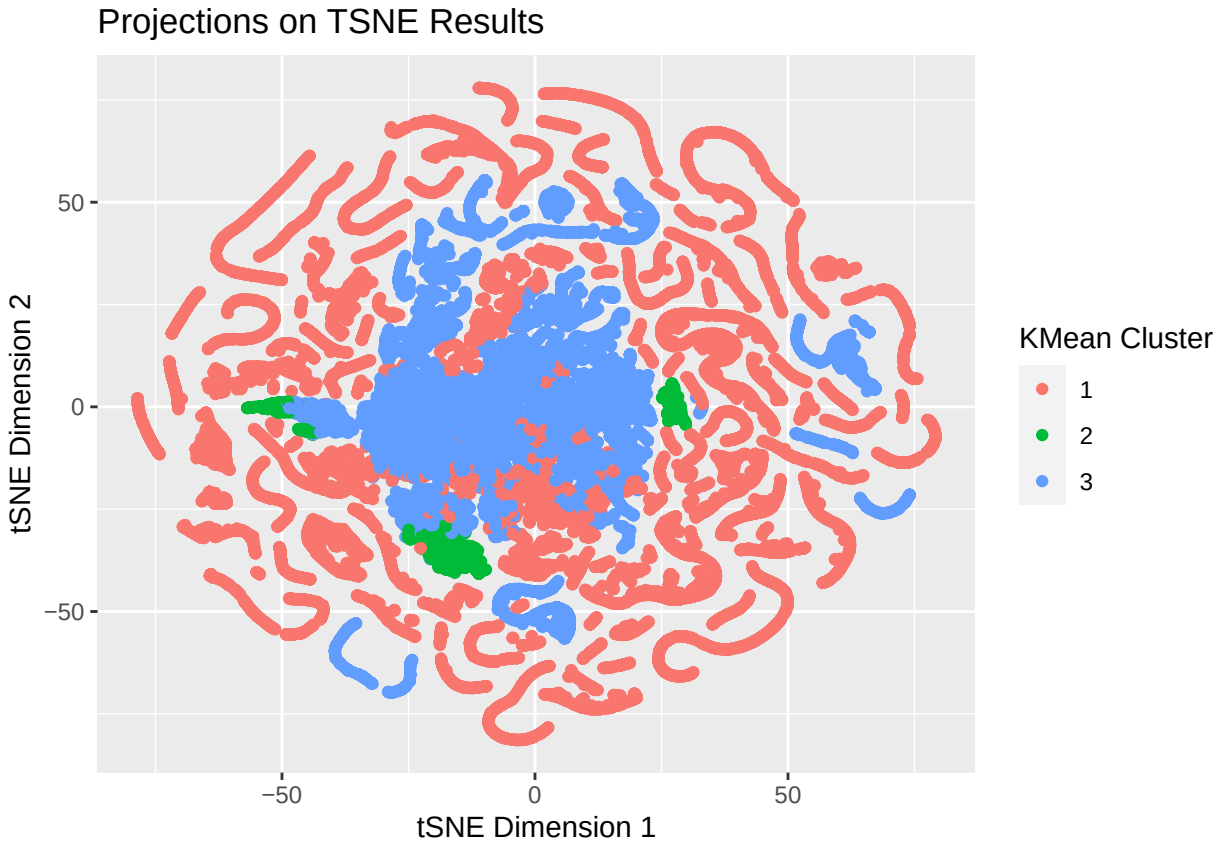
```
# Make biplot
plot1<-ggbiplot(my.pca,choices=c(1,2),
  var.axes=TRUE, # Display axes
  ellipse = FALSE,
  obs.scale=1,
  groups=as.factor(km$cluster)) +
  ggtitle("User Data Projected on PC1 and PC2 ")
plot1
```



This biplot gives us several important takeaways. First, we can see that there are fairly clear divides between our clusters. Also, the first 2 PCA projections are able to explain more than 41% of the variance in our data, which is an improvement from the initial feature set. Ideally, with a bit more fine tuning this number will be able to get even higher. We can see that there are several outliers that fall outside our main group of the users. It should be noted that these all fall into groups 1 or 3, but none are in two. If we are trying to determine liquidation prospects, it is good to know that these outlier users are less likely to be forced to liquidate. However, these outliers also make this plot a little harder to see.

Next, I looked into tSNE, a dimensionality reduction method that helps to visualize our data in lower dimensions. This can help to show us patterns in the data that a linear PCA may have missed.

```
# Run tSNE
set.seed(1)
t <- Rtsne(df.scaled, dims=2, max_iter = 1500, check_duplicates = FALSE)
# Make dataframe
tsne_plot <- data.frame(x = t$Y[,1], y = t$Y[,2])
# Plot
ggplot(tsne_plot, aes(xlab = "tSNE Dimension 1", ylab = "tSNE Dimension 2")) +
  geom_point(aes(x=x, y=y, col = as.factor(km$cluster))) +
  labs(title = "Projections on TSNE Results", x = "tSNE Dimension 1", y = "tSNE Dimension 2", color = "
```



From this, we see that our clusters seem to be representing the data well, but not perfectly. Cluster 1 appears to overlap with parts of cluster 3, and cluster 2 is split into to smaller groups. Despite this, I believe that this shows that these clusters are close, and that we can expect PCA to perform reasonably well on this set of data.