DAR F21 Project Status 4 Observing Trends through Users' Average Transactions

IDEA-Blockchain (DeFi)

Duke Kwon

10/14/21

Contents

Weekly Work Summary

- RCS ID: kwond2
- Project Name: IDEA-Blockchain (DeFi)
- General Summary:
- This week, I worked on trying to observe patterns among the clusters from HDBSCAN to be able to distinguish between types of users. Mainly, lookin
- I was able to locate some differences in the characteristics of the clusters, through mean heatmaps and plotting the density plots of features by cluster.
- Github Commits:
- Branch: dar-kwond2 (However main code will be available in this notebook)
- Directory: https://github.rpi.edu/DataINCITE/IDEA-Blockchain/tree/master/DefiResearch/StudentNotebooks/Assignment04
- kwond2_assignment4.Rmd
- kwond2_assignment4.pdf
- kwond2_assignment4.html
- References:
- N/A
- Shared Code Base:
- Clusters generated from the previous Rmd, kwond2_assignment3.Rmd are used, otherwise no other shared codebase.

Personal Contribution

- All code and explanation in this notebook was done by me.

Discussion of Primary Findings

- Problem: We had clusters of the users' average transactions generated from the nonlinear dimensionality reduction techniques and density based clustering, but no sense of interpreting what the clusters meant.
- Proposed Solution: We use visualization tools such as heatmaps and density estimates of the features to compare differences by cluster.
- Results: We produced various heatmaps of features like proportion and mean liquidations or borrows, and various density plots of those features to visualize the spread of the cluster. No concrete claims were able to be made - more representative features and generally more data needs to be used.

General Table of Contents:

Problem Exploration

We start by reading in the clusterings that we wrote out to from our UMAP and HDBSCAN from the previous assignment.

```
df.scaled <- read.csv("users_scaled.csv")
km_clusters <- read.csv("km_clusters.csv")
# remove indices generated from writing to csv.
df.scaled <- df.scaled[, -which(names(df.scaled) %in% c("X"))]
km_clusters <- km_clusters[, -which(names(km_clusters) %in% c("X"))]
#install.packages("umap")
#install.packages("dbscan")
library(umap)
library(dbscan)
library(ggplot2)
library(devtools)
```

```
## Loading required package: usethis
```

```
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 1.3.1 --
```

```
## v tibble  3.1.4     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     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
```

Doing a preliminary heatmap of the original data, it had turned out "timeactive" feature was controlling the clustering entirely. So, we remove the feature and quickly do UMAP + HDBSCAN, this time with stricter hyperparameters on the clustering (require at least 500 points to be a cluster).

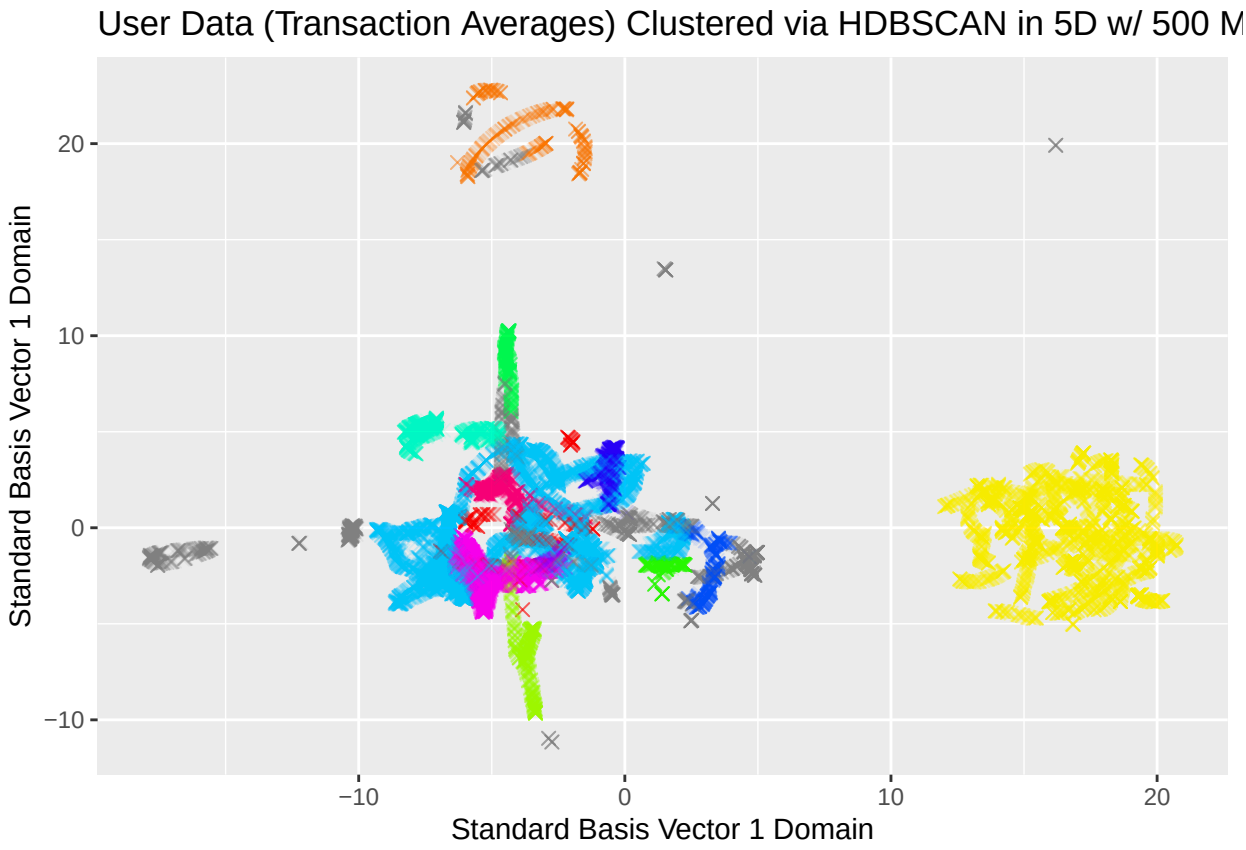
```
df.scaled <- df.scaled[, -which(names(df.scaled) %in% c("timeactive"))] ## remove timeactive case
set.seed(43)
users.umap_5 = umap(df.scaled, n_components = 5)
```

```
## Warning: failed creating initial embedding; using random embedding instead
```

```
minimum_points = 500
set.seed(43)
users.hdb_5 <- hdbscan(users.umap_5$layout, minPts = minimum_points)
```

This turns out to produce drastically different results. However, we do have to note that UMAP is an iterative, random algorithm that minimizes a loss, which means our visualizations will vary quite a bit.

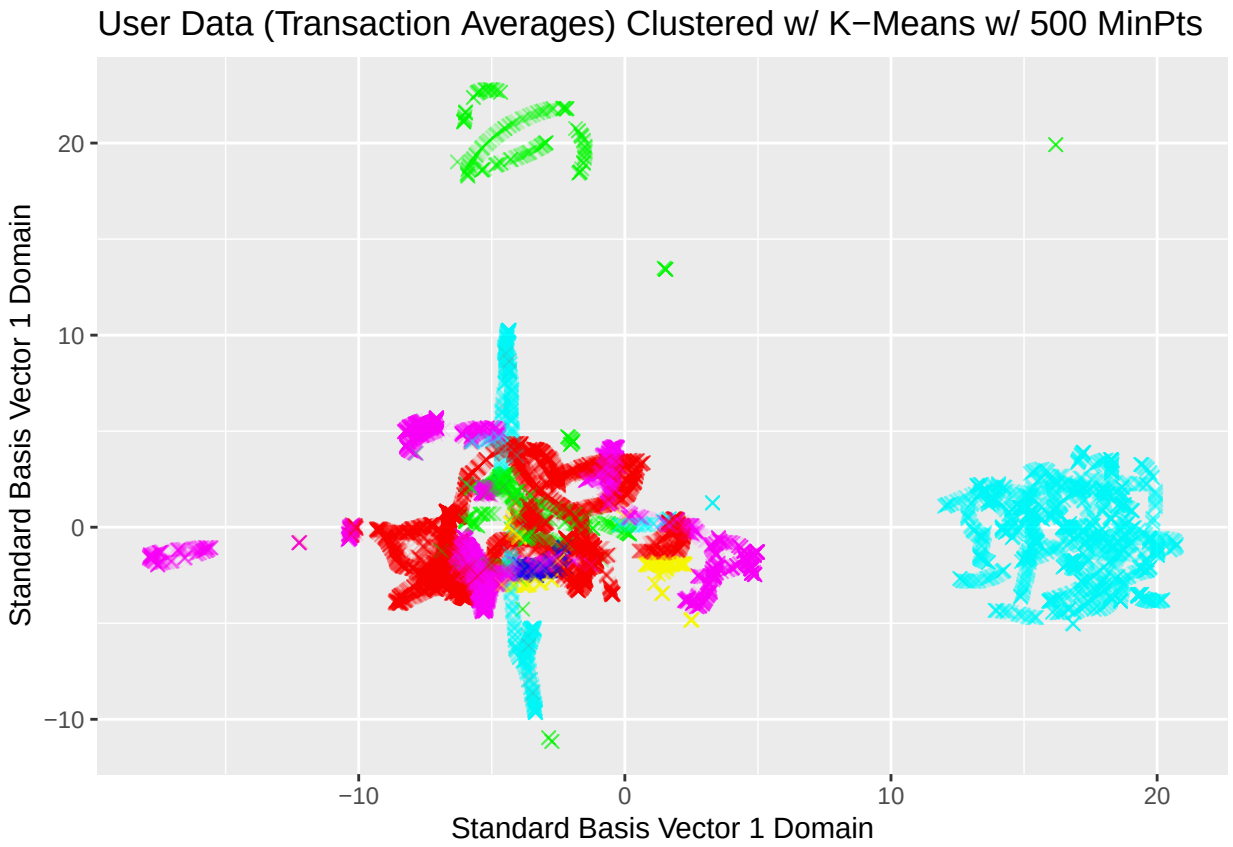
```
df_umap <- data.frame(users.umap_5$layout)
n_colors = max(users.hdb_5$cluster)
pre_colors <- rainbow(n_colors)
palette1 <- c("#808080", pre_colors) ## set 0 to a light grey as outliers
color_labels <- vector("character", length = length(users.hdb_5$cluster))
for (i in 0:n_colors) {
  color_labels[users.hdb_5$cluster == i] = palette1[i+1]
}
cluster_names <- c("Outliers", c(1:n_colors))

ggplot(df_umap, aes(x=X1, y=X2)) + geom_point(alpha = 0.1, size = 2, shape = 4, colour = color_labels)
```



Originally, we thought that k-means clustered the data similarly to UMAP + HDBSCAN. However, once we remove the "timeactive" column (which also likely changes the seed of UMAP internally), we see that the k-means clusters only have partially similar clusterings now.

```
n_colors_km = max(km_clusters)
pre_colors_km <- rainbow(n_colors_km)
palette3 <- c("#808080", pre_colors_km) ## set 0 to a light grey as outliers
color_labels_km <- vector("character", length = length(km_clusters))
for (i in 0:n_colors_km) {
  color_labels_km[km_clusters == i] = palette3[i+1]
}
cluster_names_km <- c("Outliers", c(1:n_colors_km))
```

```
ggplot(df_umap, aes(x=X1, y=X2)) + geom_point(alpha = 0.1, size = 2, shape = 4, colour = color_labels_k
```



The problem we wish to dive to into now obviously appears - what do the clusterings mean? UMAP does not have intuitive directions/bases compared to PCA which tries to capture the most variance in the principal components. It'll be necessary to look at the summary statistics, a heatmap of the features, and compare the densities for the features.

Implementations & Results

Here we just have some preliminary aliases/and data organizing.

```
#df.4 <- read_csv(file='users_clustered.csv')
out_labels <- users.hdb_5$cluster
joined_data <- df.scaled
joined_data$hdb_clusters <- out_labels
df.4 <- joined_data
mx <- max(df.4$hdb_clusters)
dict <- table(df.4$hdb_clusters)
df.dict <- as.data.frame(dict)
out_labels <- df.4$hdb_clusters
```

We compute the means of the columns (and as a sanity check do it with the original data as well). We also display and color code the 3 most prominent (largest) clusters which we'll be taking a look at in the density/histogram plots.

```
df <- read.csv("transactions.csv") ## read original transactions CSV
df.users <-read.csv("df_users.csv") ## read users parsed csv
df.users <- df.users[, -which(names(df.users) %in% c("X"))]
```

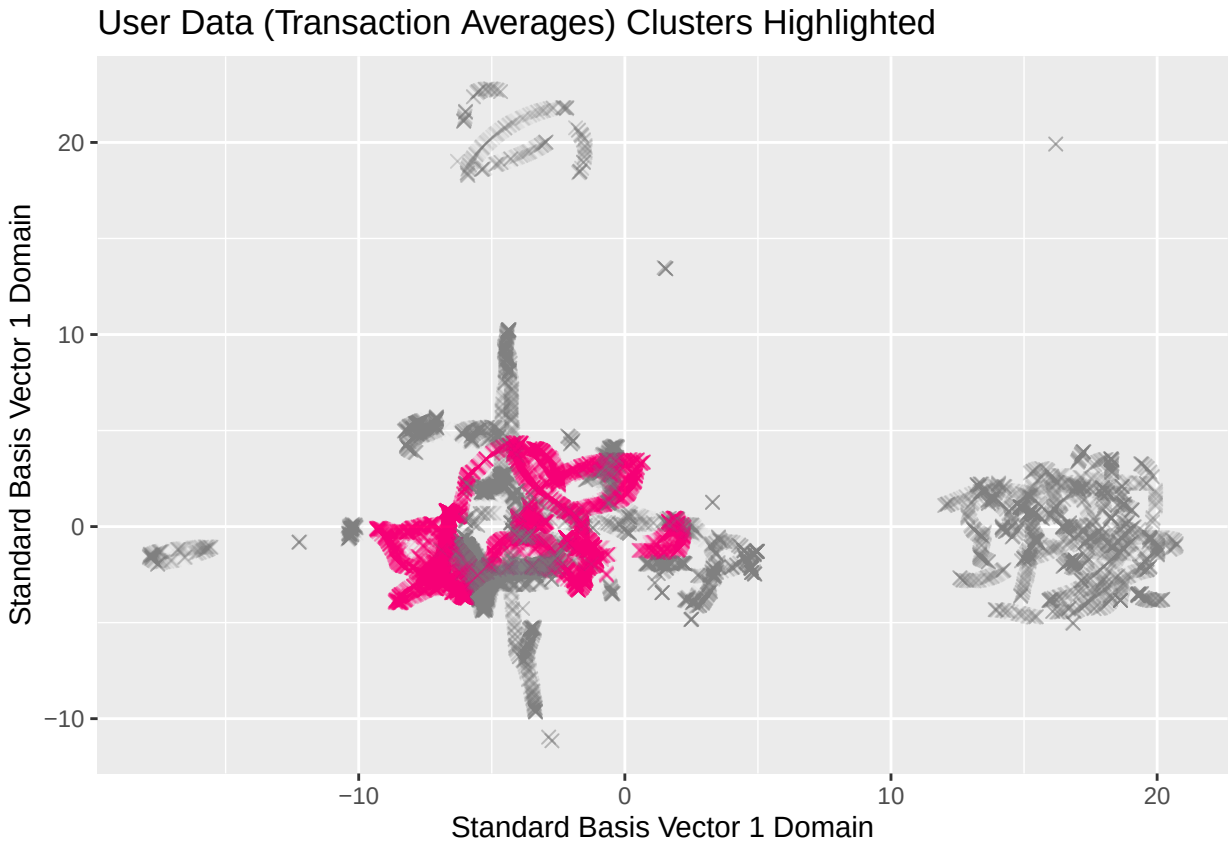
```r
dropped_df_liq <- drop_na(df, amountUSDCollateral) ## dataset with all user transactions with non-NAN l
dropped_df_borrow <- drop_na(df, amount) ## dataset with non-NAN amounts borrowed
N = mx
check_top_n = 3
liquidations_on <- TRUE
number_of_dims_scaled <- (df.scaled %>% dim)[2]

hdb_centers <- matrix(0, N, 29) # 29 attributes of the original dataset.
hdb_centers_2 <- matrix(0, N, number_of_dims_scaled) # ~ 16 attributes of the dataset we ran PCA on.
topclusters <- top_n(df.dict, n=N, Freq) %>% arrange(desc(Freq)) # get the top n sized clusters and the
for (i in 1:N) {
  c_idx <- topclusters[i,1]
  #if (c_idx == 0) {
  #  next
  #  } # skip the outlier case, when the cluster id is 0.
  c_user_idx <- which(df.4$hdb_clusters == c_idx) # pick out all indexes of users with the specific clu
  c_users_id <-df.users[c(c_user_idx),] # now get the actual users into a df from their indices from ab
  transact_users <- which(dropped_df_liq$user == c_users_id) #get all transactions of the cluster of us
  transact_users_amount <- which(dropped_df_borrow$user == c_users_id)

  ## Compute colmeans of all clusters, and insert to a matrix to do a heatmap analysis.
  clm <- colMeans(c_users_id, na.rm = TRUE)
  hdb_centers[i, 1:29] <- clm
  clm2 <- colMeans(df.scaled[c(c_user_idx),])
  hdb_centers_2[i, 1:number_of_dims_scaled] <- clm2
  # The following if-statement is to display which clusters are being selected.
  # Further d
  if ((i <= check_top_n)) {
    sprintf("Cluster Number: %d\nCluster Size %d\n", c_idx, length(c_user_idx)) %>% cat()
    n_colors = unique(users.hdb_5$cluster)
    pre_colors <- rainbow(n_colors)
    palette1 <- c("#808080", pre_colors) ## set 0 to a light grey as outliers
    color_labels <- vector("character", length = length(users.hdb_5$cluster))
    for (i in 0:n_colors) {
      color_labels[users.hdb_5$cluster == i] <- palette1[i+1]
    }
    alpha_vals <- c(rep(0.05,length(color_labels))) ## set custom alpha values
    alpha_vals[c(which(out_labels == c_idx))] <- 0.1
    color_vals <- c(rep("#808080",length(color_labels))) ## set custom colors
    color_vals[c(which(out_labels == c_idx))] <- color_labels[c_idx]
    zeroed_color_labels = color_vals
    plot1 <- ggplot(df_umap, aes(x=X1, y=X2))
    plot1 <- plot1 + geom_point(alpha = alpha_vals, size = 2, shape = 4, colour = zeroed_color_labels)
      print(plot1)
  }


}
```

```
## Cluster Number: 9
## Cluster Size 11945
```

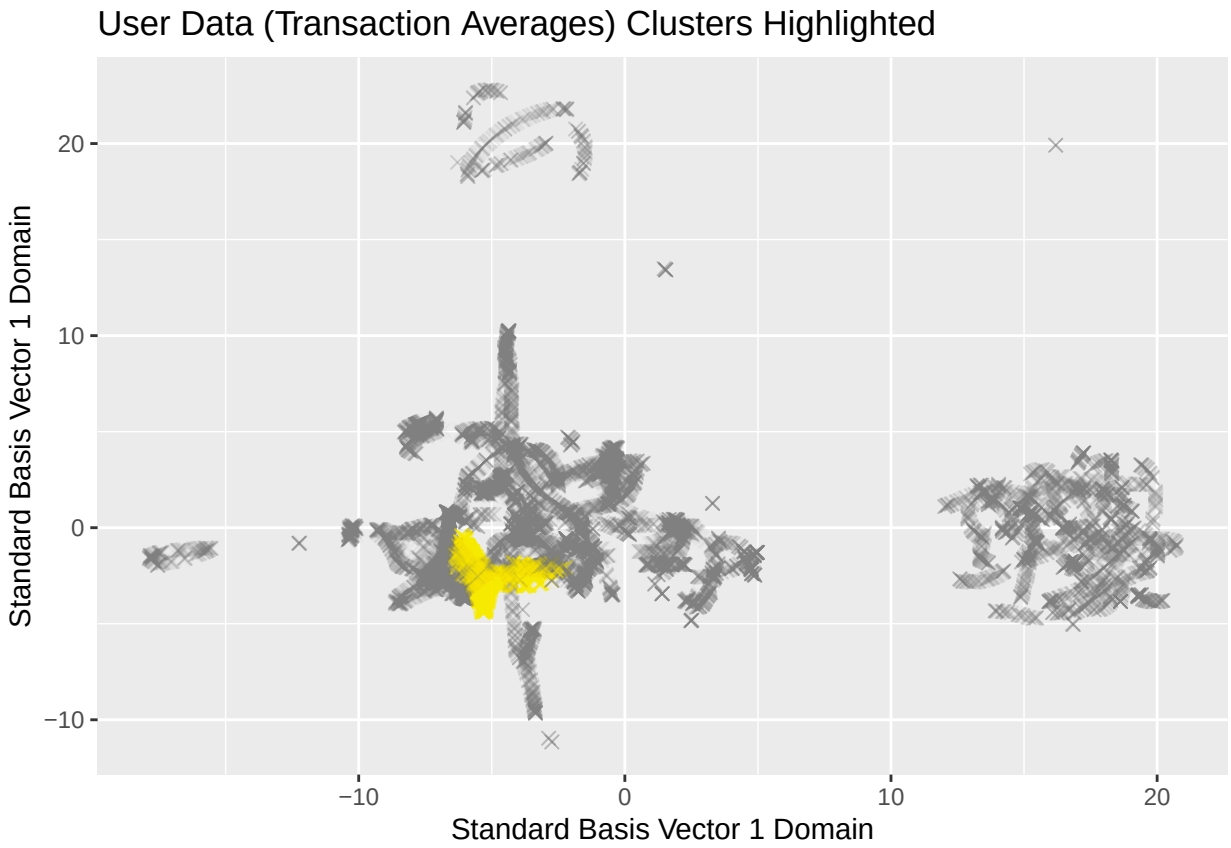
```
## Warning in 0:n_colors: numerical expression has 14 elements: only the first used
```

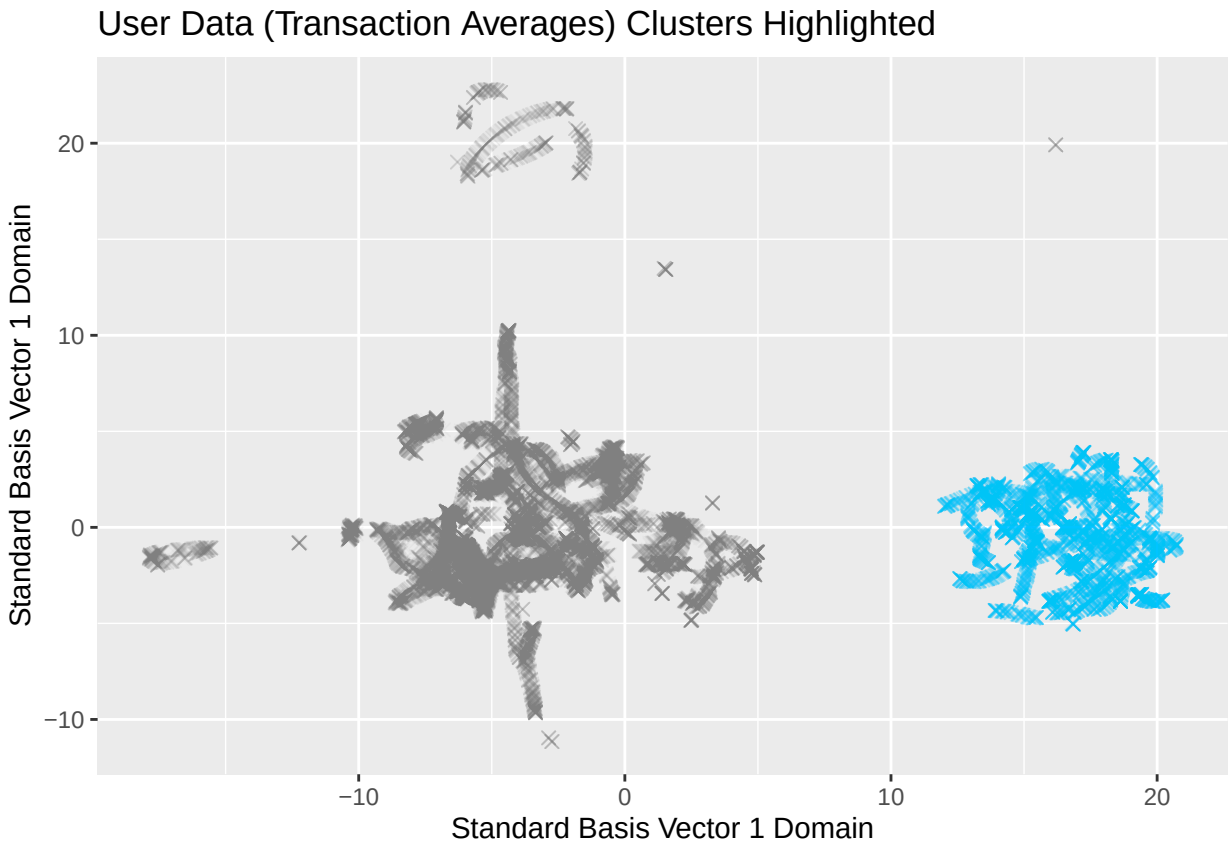
```
## Cluster Number: 13
```

```
## Cluster Size 7966
## Warning in 0:n_colors: numerical expression has 14 elements: only the first used
```




```
## Cluster Number: 4
## Cluster Size 7758
## Warning in 0:n_colors: numerical expression has 14 elements: only the first used
```


User Data (Transaction Averages) Clusters Highlighted
Standard Basis Vector 1 Domain
Standard Basis Vector 1 Domain
20
10
0
−10
−10
0
10
20

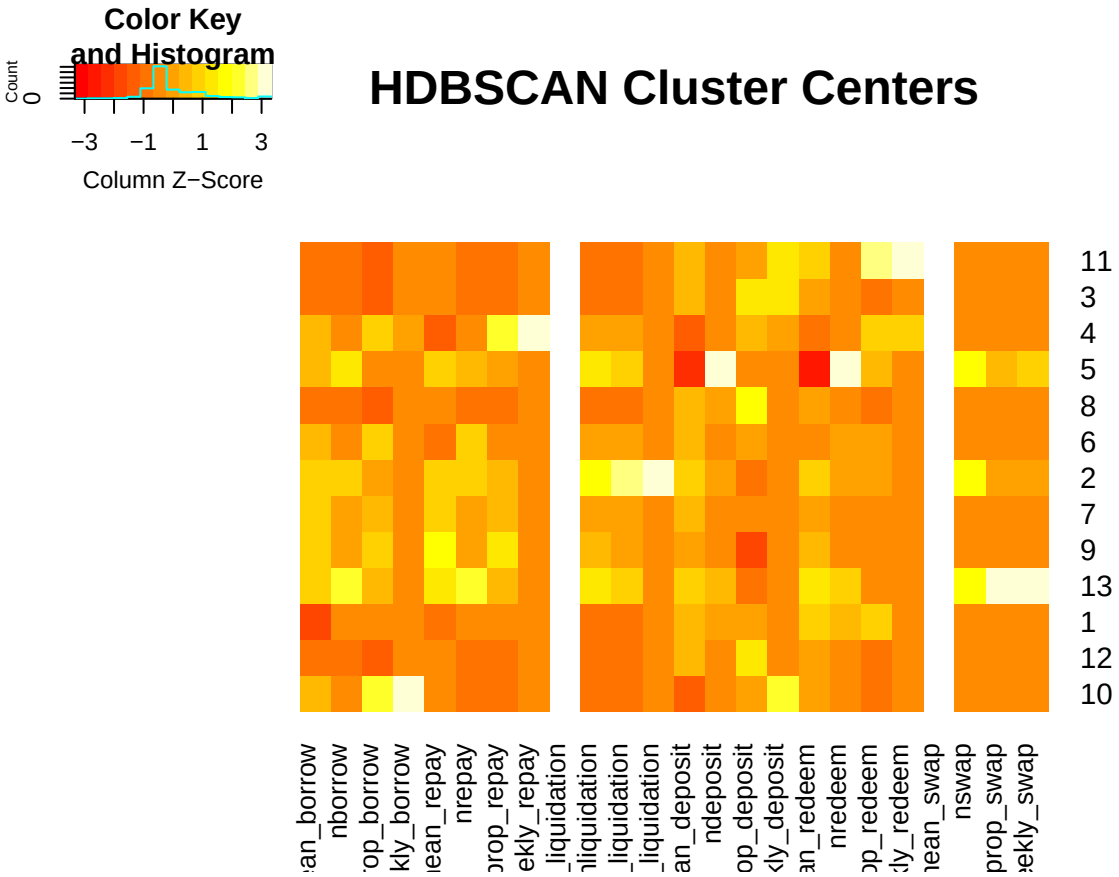

User Data (Transaction Averages) Clusters Highlighted
Standard Basis Vector 1 Domain
Standard Basis Vector 1 Domain
20
10
0
−10
−10
0
10
20

```
hdb_centers.df <- as.data.frame(hdb_centers)
colnames(hdb_centers.df) <- df.users %>% colnames
hdb_centers_zeroed.df <-hdb_centers.df%>%replace(is.na(.),0)

final_hdb_centers.df <- hdb_centers_zeroed.df[, -which(names(hdb_centers_zeroed.df) %in% c("user", "tim
final_hdb_centers.df %>% head
```

```
##   mean_borrow  nborrow prop_borrow weekly_borrow mean_repay   nrepay prop_repay
## 1   -3.019757 1.000000   0.1711168  9.773507e-01  -1.015745 1.000000 0.08664982
## 2    8.953964 6.920252   0.2818398  1.189362e+01   5.521202 5.415247 0.23802359
## 3    0.000000 0.000000   0.0000000  0.000000e+00   0.000000 0.000000 0.00000000
## 4    7.524975 2.224242   0.5017890  6.279298e+04  -3.658588 1.545881 0.52210417
## 5    6.534683 9.782497   0.2232531  1.045209e+02   6.060166 4.508096 0.19128316
## 6    7.653002 2.173748   0.4568145  5.693482e+02  -2.162508 6.000000 0.10995740
##   weekly_repay mean_liquidation nliquidation prop_liquidation
## 1 2.412025e+00                0     0.000000       0.00000000
## 2 1.361535e+01                0     3.473518       0.22481637
## 3 0.000000e+00                0     0.000000       0.00000000
## 4 1.066135e+05                0     1.250000       0.04227121
## 5 6.681452e+01                0     3.234694       0.10074349
## 6 7.850776e-01                0     1.000000       0.04230102
##   weekly_liquidation mean_deposit  ndeposit prop_deposit weekly_deposit
## 1         0.00000000     8.008987  4.586820    0.4899565       92.84204
## 2       763.30404018     9.305933  6.169558    0.2765692       10.07332
## 3         0.00000000     6.130784  1.002191    0.8078690   606125.29002
## 4         0.10046328    -9.691976  3.895265    0.6065296   181742.51030
## 5         0.40016648   -19.051041 32.478488    0.3489875      382.83761
## 6         0.08431872     8.100948  2.341154    0.4844630      606.04285
##   mean_redeem   nredeem prop_redeem weekly_redeem mean_swap    nswap  prop_swap
## 1    8.044384  5.346254   0.4404440      99.97664         0 0.000000 0.00000000
## 2    9.434365  3.852097   0.2017360      11.43265         0 1.557692 0.01090052
## 3    0.000000  0.000000   0.0000000       0.00000         0 0.000000 0.00000000
## 4   -5.470847  1.941292   0.4651326  223794.24701         0 0.000000 0.00000000
## 5  -19.440920 17.612270   0.3062691     377.48539         0 1.714286 0.02220027
## 6   -3.627873  2.166667   0.1529684       1.36244         0 0.000000 0.00000000
##   weekly_swap
## 1  0.00000000
## 2  0.08140369
## 3  0.00000000
## 4  0.00000000
## 5  0.46590425
## 6  0.00000000
```

There exists NA data as some users don't have averages for some of the transactions.

```
#make heatmap of cluster centers
heatmap.2(final_hdb_centers.df %>% as.matrix(),
scale = "column",
dendrogram = "none",
Colv=FALSE,
cexCol=1.0,
main = "HDBSCAN Cluster Centers",
trace ="none")
```



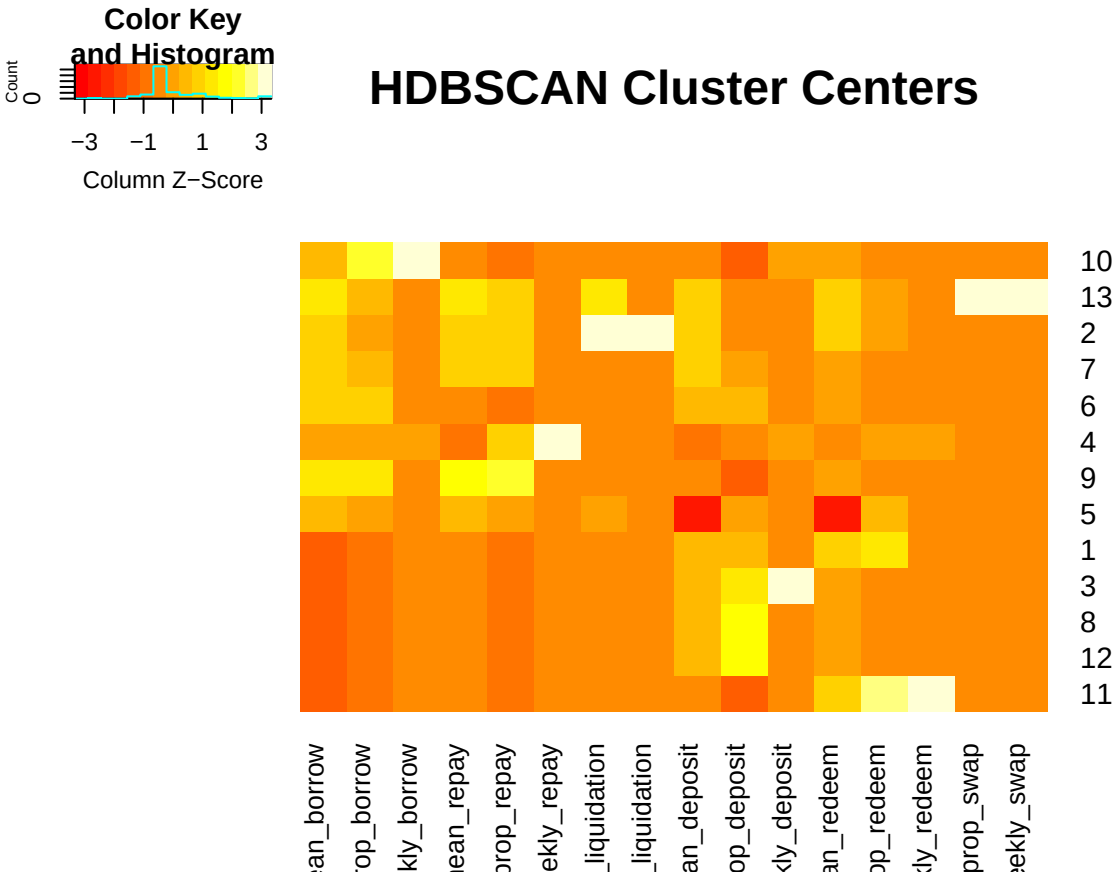

```
#clm2 %>% head
```

We just set their values to 0.

```
hdb_centers_2.df <- as.data.frame(hdb_centers_2)
colnames(hdb_centers_2.df) <- df.scaled %>% colnames
hdb_centers_zeroed_2.df <-hdb_centers_2.df%>%replace(is.na(.),0)
final_hdb_centers_2.df <- hdb_centers_zeroed_2.df
```

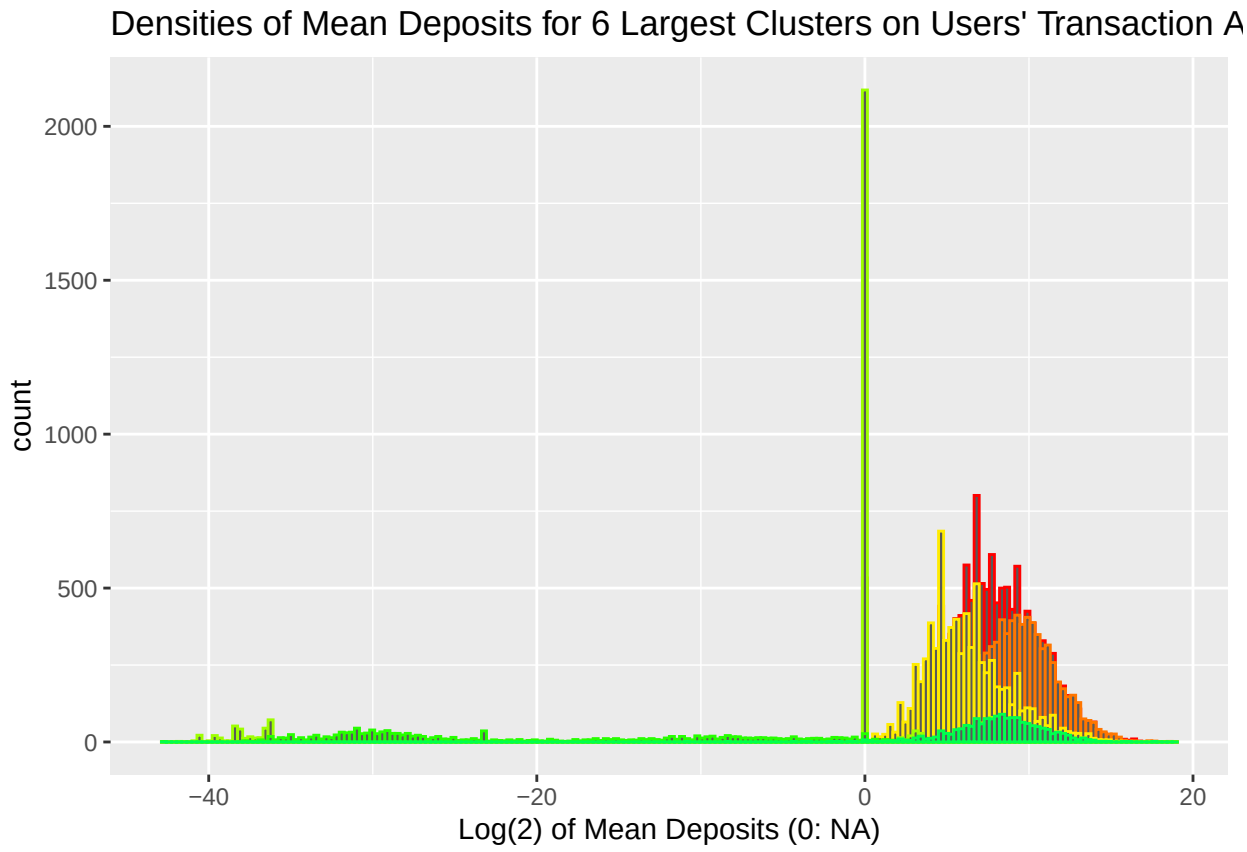
We get the following heatmap of our HDBSCAN mean clusters. Certain features definitely have a noticeable range, such as mean deposit. It might interesting to take a look at the density/histogram plots of the averaged values per user and compare the clusters.

```
#make heatmap of cluster centers
heatmap.2(final_hdb_centers_2.df %>% as.matrix(),
scale = "column",
dendrogram = "none",
Colv=FALSE,
cexCol=1.0,
main = "HDBSCAN Cluster Centers",
trace ="none")
```

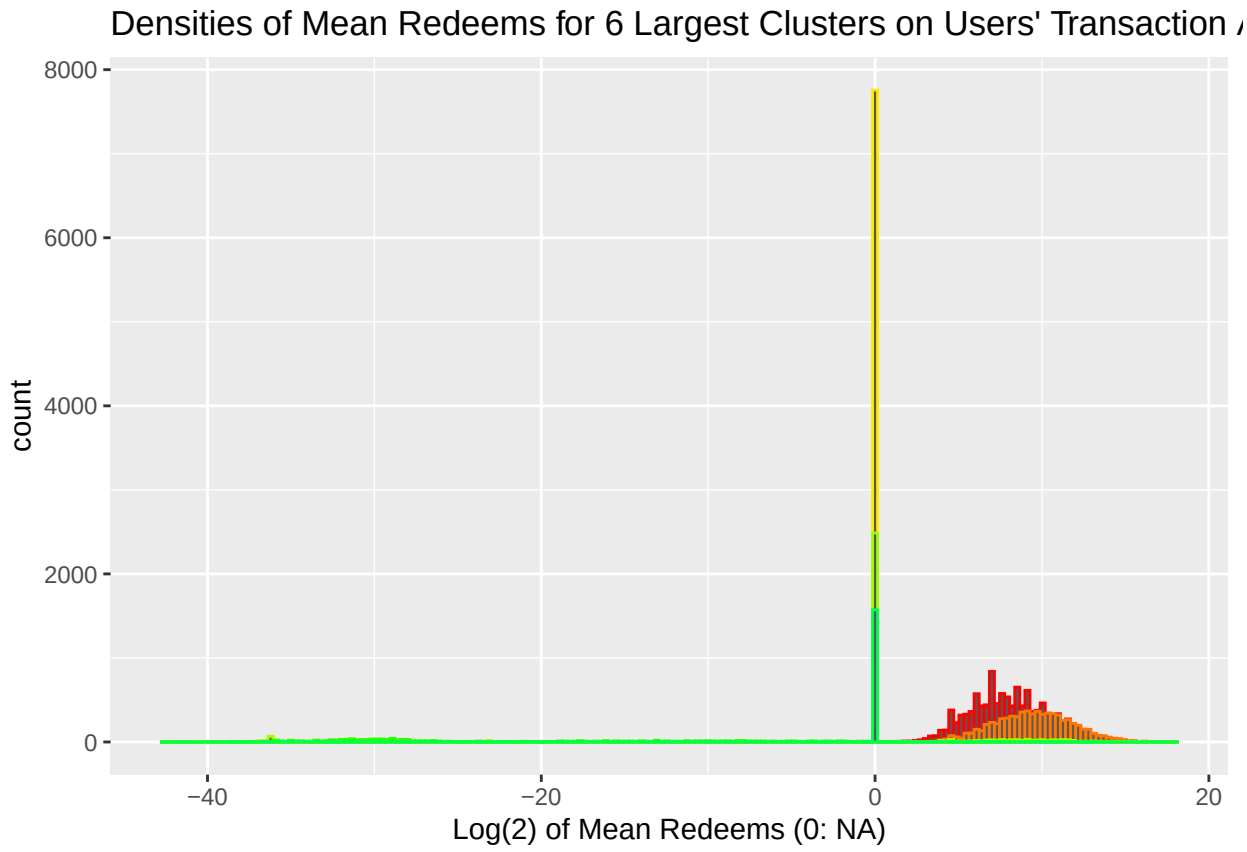


We output the histogram style densities of mean deposits, mean redeems, and mean repays. From the plot of mean deposits, we see that the clusters do have different range of values, which may represent different the different type of users. For example, we see that the orange density-histogram mainly has the larger values overall, which we can also see if we display the summary statistics. We also notice a significant chunk of users (cluster) that are centered at 0, likely for the NA values. Similarly, this is observable in mean redeems, and mean repays. So our clustering was able to group users who don't do specific transactions as well. Furthermore, we can also take a look at the proportion of transactions, which we show next.

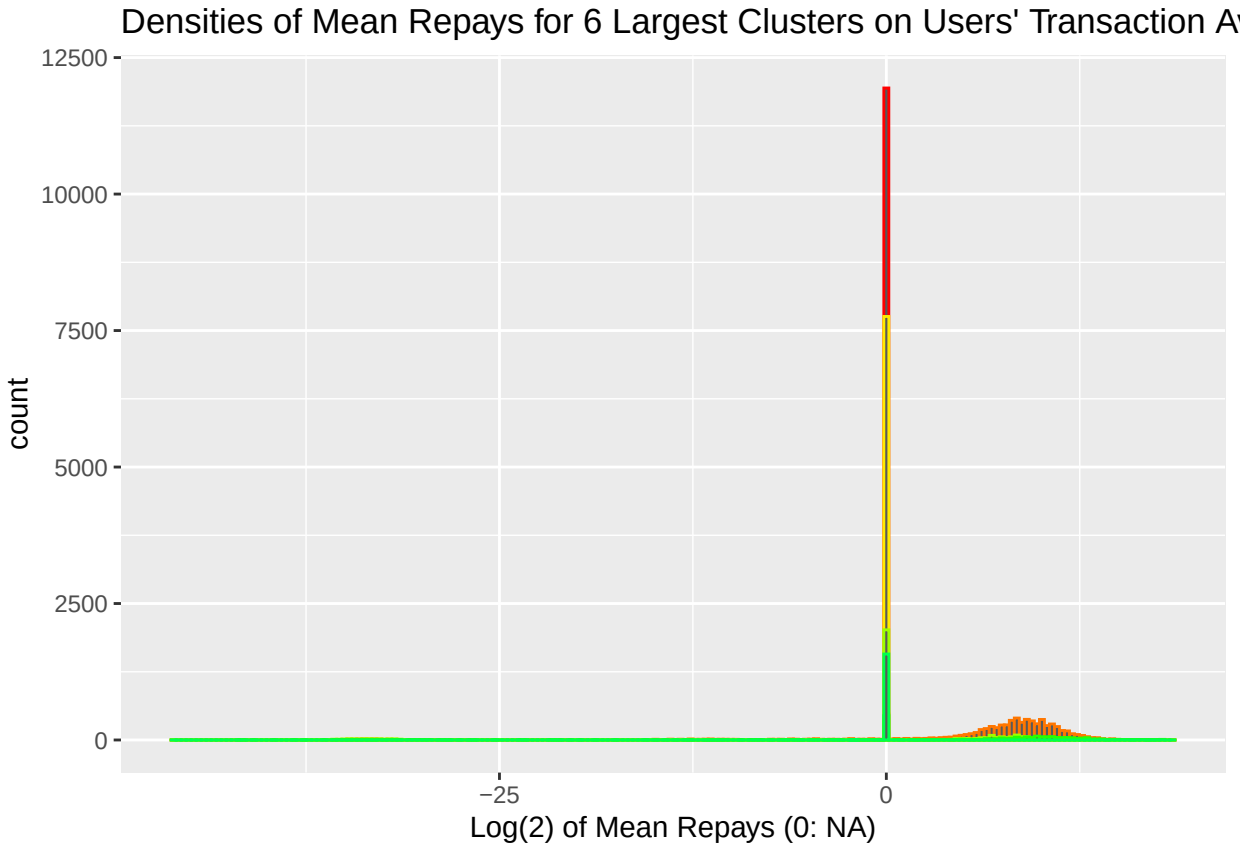
```
N_den = 6
plot_den <- ggplot() + labs(title = "Densities of Mean Deposits for 6 Largest Clusters on Users' Transa
plot_red <- ggplot() + labs(title = "Densities of Mean Redeems for 6 Largest Clusters on Users' Transac
plot_repay <- ggplot() + labs(title = "Densities of Mean Repays for 6 Largest Clusters on Users' Transa

for (i in 1:N_den) { ## outliers not included, skip 0
  c_idx <- topclusters[i,1]
  c_user_idx <- which(df.4$hdb_clusters == c_idx)
  c_users_id <- df.users[c(c_user_idx),]
  c_users_id_replaced <-c_users_id%>%replace(is.na(.),0)

   plot_den <- plot_den + geom_histogram(aes(x=mean_deposit), color = palette1[i+1], data=c_users_id_re
   plot_red <- plot_red + geom_histogram(aes(x=mean_redeem), color =  palette1[i+1], data=c_users_id_re
   plot_repay <- plot_repay + geom_histogram(aes(x=mean_repay), color =  palette1[i+1], data=c_users_id
}

print(plot_den + xlab("Log(2) of Mean Deposits (0: NA)"))
```




```
print(plot_red + xlab("Log(2) of Mean Redeems (0: NA)"))
```



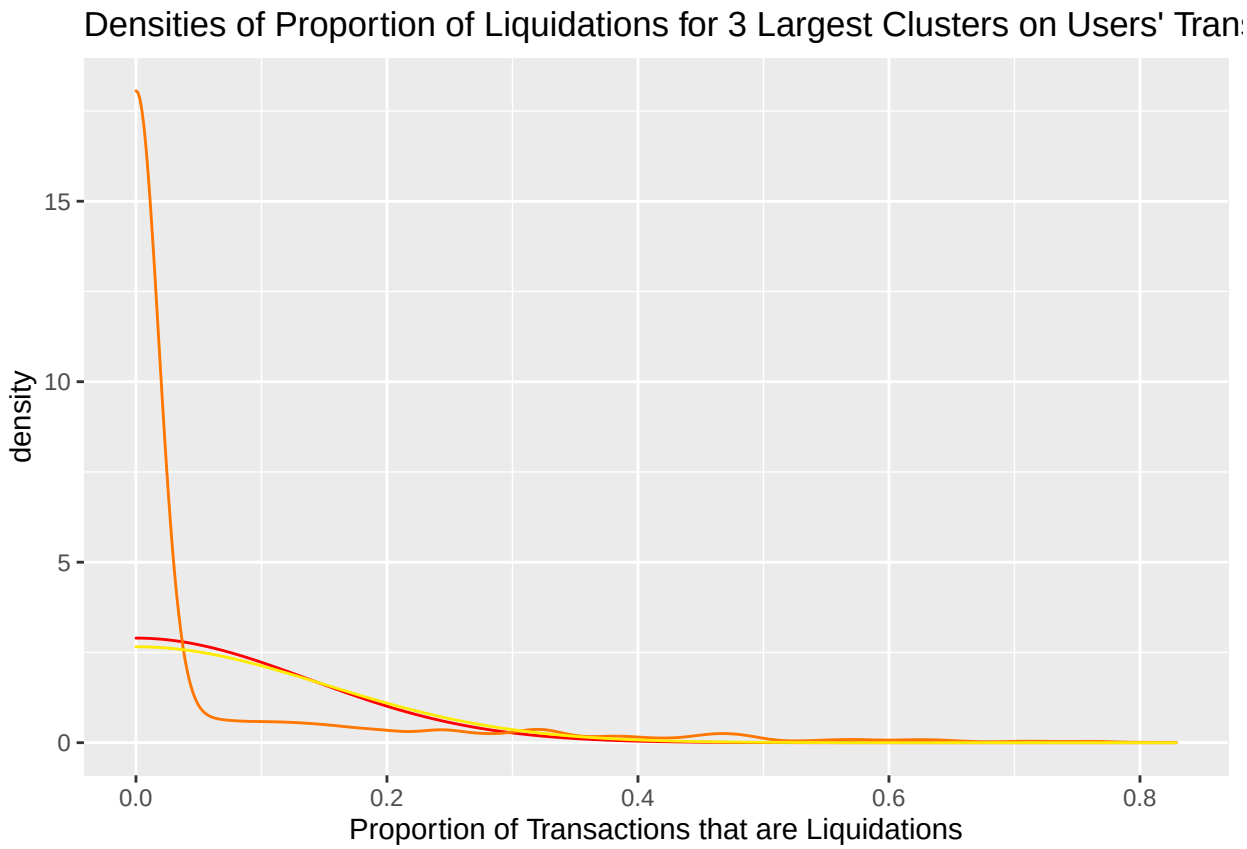

```
print(plot_repay + xlab("Log(2) of Mean Repays (0: NA)"))
```



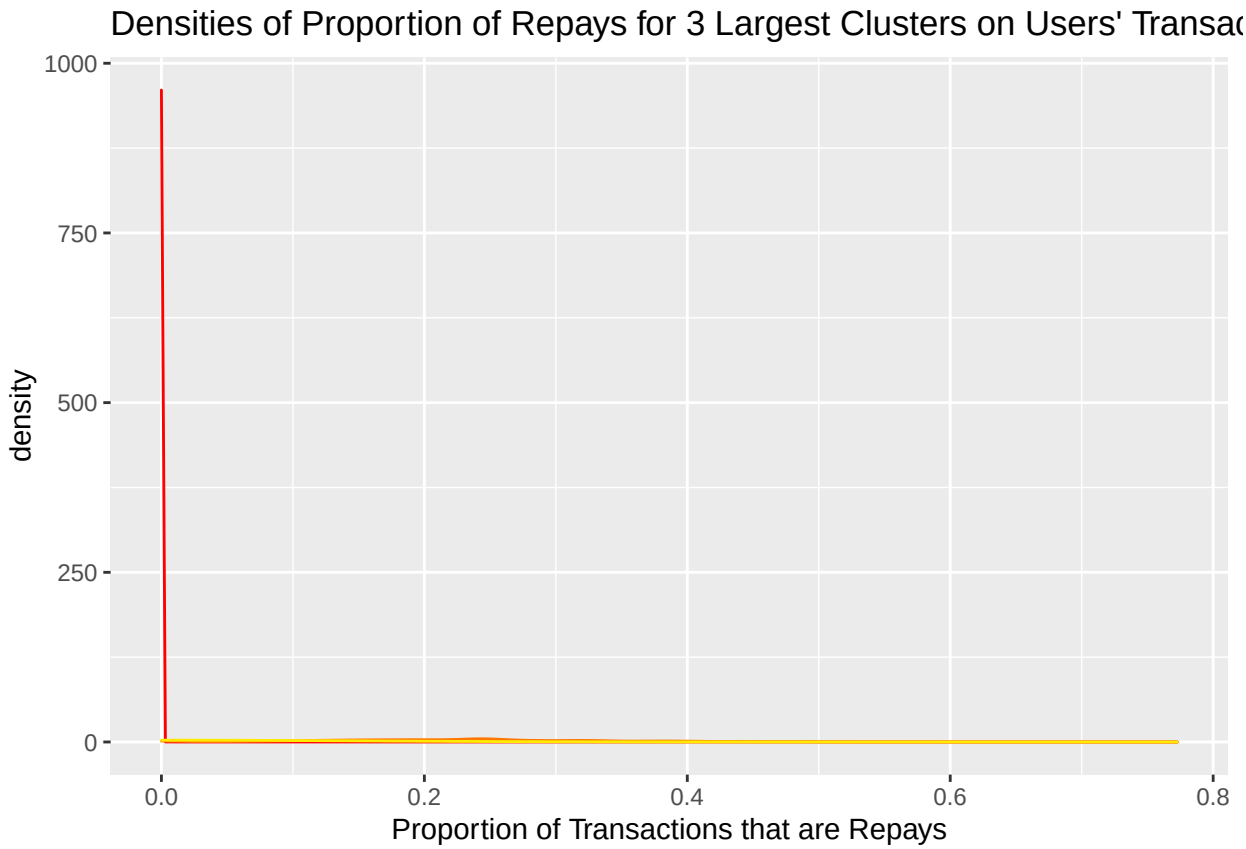
The first plot we show is the density of the proportion of liquidations for the 3 largest clusters. We see that 2 of the clusters have similar density plots in terms of liquidations, and the third clustering has users who rarely get liquidated (as we increase the number of clusters we look at, most clusters tend to have mostly no liquidators, and are captured in the plots shown below). In the third plot, we take a look at the proportion of deposits within those top 3 clusters, and notice that one of the clusters has a high proportion of mean deposits (yellow), which we might expect from a clustering of power users who may liquidate a lot. Or, it could be the cluster where we had a minimal number of liquidations, and those users mainly deposit and rarely get liquidated. In red, we see its proportion of transactions have multiple small peaks, which may mean that specific clustering did not particularly look into separating the mean deposits - so other features would likely have to be looked at.

```
N_den = 3
plot_liq <- ggplot() + labs(title = "Densities of Proportion of Liquidations for 3 Largest Clusters on
plot_prop_repay <- ggplot() + labs(title = "Densities of Proportion of Repays for 3 Largest Clusters on
plot_prop_dep <- ggplot() + labs(title = "Densities of Proportion of Deposits for 3 Largest Clusters on

for (i in 1:N_den) {
  c_idx <- topclusters[i,1]
  if (c_idx == 0) {
    next
  }
  c_user_idx <- which(df.4$hdb_clusters == c_idx)
  c_users_id <- df.users[c(c_user_idx),]
  c_users_id_replaced <-c_users_id%>%replace(is.na(.),0)
  c_users_dropped <- c_users_id %>% drop_na()
plot_liq <- plot_liq + geom_density(aes(x=prop_liquidation), color =  palette1[i+1], data=c_users_id_re
plot_prop_repay <- plot_prop_repay + geom_density(aes(x=prop_repay), color =  palette1[i+1], data=c_use
plot_prop_dep <- plot_prop_dep + geom_density(aes(x=prop_deposit), color =  palette1[i+1], data=c_users
```

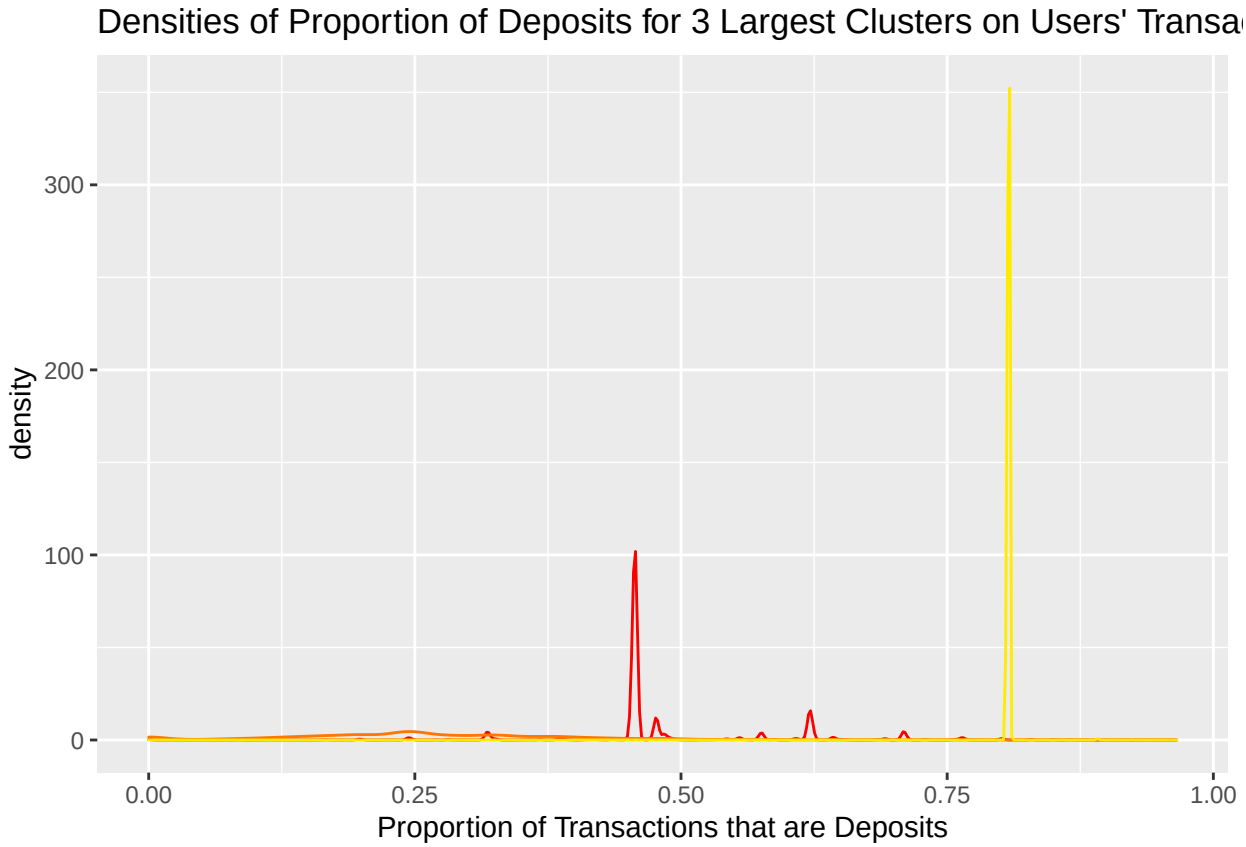
```
}

print(plot_liq)
```




```
print(plot_prop_repay)
```




```
print(plot_prop_dep)
```



Further Work & Conclusions:

Despite our attempt to visualize the clusters through how spread out their means are, it's very dependent on our features. A future step we should likely take is think hard about picking good features that provide a good spread between types of users. Also, the type of analysis we can do at most speculative - even with the visualizations, it's not always clear how their shown transaction behavior can characterize what kind of user they are, nor do we know the number of "true" clusters, and the properties of them.