DAR F21 Project Status 6 Clustering on Weekly Transactions

IDEA-Blockchain (DeFi)

Duke Kwon

11/2/21

Contents

Weekly Work Summary

- RCS ID: kwond2
- Project Name: IDEA-Blockchain (DeFi)
- General Summary:
  - Redesigning of features based on weekly characteristics of user transactions was done to improve over all clusterings - ideally to characterize groups of users with distinct styles of transactions at the weekly level. Visualizations and analysis of the new clusters are done, with added survival plots to distinguish between users.

Github Items

- Summary of github issues
  - Addressed Issue #86, "Create approach to create clusters using weekly data and put in notebook".
- Github Commits:
- Branch: dar-kwond2
- Directory: https://github.rpi.edu/DataINCITE/IDEA-Blockchain/tree/master/DefiResearch/StudentNotebooks/Assignment06
- kwond2_assignment6.Rmd
- kwond2_assignment6.pdf
- kwond2_assignment6.html

Contributions

- References:
  - N/A
- Shared Code Base:
  - The codebase for some of the visualizations are taken kwond2_assignment5. The survival plots now use a modified/tweaked version of Soumya's survival function.

- Personal Contribution
  - All code and explanation in this notebook was done by me, except the survival plots (though are slightly modified).

Discussion of Primary Findings

- What did you want to know?
  - Although clustering on averages of user transactions was able to group users with some noticeable differences, very little was found for their weekly transaction behavior. Thus, we wanted to see if generating new features, mainly based on weekly averages of transactions, would improve the overall clusterings.
- How did you go about finding it?
  - Rather than using panel data methods or dealing with tensors (i.e, User x Time x TransactionType x Amount), the idea was to fix the type of transaction (i.e. the most common, borrows) and compute their average across time weekly by user (feature for per week). Then, dimensionality reduction and clustering was performed on this dataset.
- What did you find?
  - Clusterings done on the data with weekly features had noticeable differences in their transaction types and amounts across time.

```
if (!require("umap")) {
  install.packages("umap")
  library(umap)
}
if (!require("dbscan")) {
  install.packages("dbscan")
  library(dbscan)
}
if (!require("ggridges")) {
  install.packages("ggridges")
  library(ggridges)
}
if (!require("data.table")) {
  install.packages("data.table")
  library(data.table)
}

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

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

```
if (!require("ggplot2")) {
  install.packages("ggplot2")
  library(ggplot2)
}
if (!require("knitr")) {
  install.packages("knitr")
  library(knitr)
}
if (!require("dplyr")) {
  install.packages("dplyr")
```

```
  library(dp)
}
if (!require("devtools")) {
  install.packages("devtools")
  library(devtools)
}
if (!require("RColorBrewer")) {
  install.packages("RColorBrewer")
  library(RColorBrewer)
}
if (!require("beeswarm")) {
  install.packages("beeswarm")
  library(beeswarm)
}
if (!require("tidyverse")) {
  install.packages("tidyverse")
  library(tidyverse)
}
if (!require("ggbeeswarm")) {
  install.packages("ggbeeswarm")
  library(ggbeeswarm)
}
if (!require("xts")) {
  install.packages("xts")
  library(xts)
}
if (!require("plotly")) {
  install.packages("plotly")
  library(plotly)
}
if(!require("lubridate")) {
  install.packages("lubridate")
  library(lubridate)
}
if(!require("survival")) {
  install.packages("survival")
  library(survival)
}
if(!require("survminer")) {
  install.packages('survminer')
  library(survminer)
}
if(!require("ranger")){
  install.packages("ranger")
  library(ranger)
}
if(!require("ggfortify")){
  install.packages("ggfortify")
  library(ggfortify)
}
if(!require("patchwork")){
  install.packages("patchwork")
  library(patchwork)
```

```
}
```

Data Loading

We start with loading in the data generated from the previous assignments. Mainly, df.users (which contains the averages for unique users across the entire dataset), and df_tr, which is just the original transactions dataset.

```
df.users <-read.csv("df_users.csv") ## read users parsed csv
df.users <- df.users[, -which(names(df.users) %in% c("X"))]
df_tr<-read_csv('transactions_2.csv') ## original transactions data (not unique users from Jan to Aug. 
```

We use lubridate to generate interpretable times, and will also come into play when generating the weekly features.

```
df_tr$ymd <- as_datetime(df_tr$timestamp) # fix times for the transactions
setDT(df_tr)[, ymd_new := format(as.Date(ymd), '%Y-%m-%V') ] ## '%Y-%m' for just month-year
#df_tr$ymd_new_d <- as.Date(df_tr$ymd_new, "%Y%m%d")
```

Generating Weekly Average Features & Clustering

As briefly explained earlier, clustering previously was done on mainly the features from the df.users dataset, which contains averaged transactions for all types across the entire timeline. Our previous results shows that our clusters, although generated relatively distinct groupings of users by their transactions, their differences were not particularly noticeable using a weekly plot of transactions.

Here, we try to address those issues by generating a dataset that directly encodes weekly averages - for each week, we compute the users averages for that week, and store it as a feature. Since there are multiple types, we actually generate a dataset for each type. We will focus on the borrows transactions, as it comprises of the majority of them. However, further analysis can be done by looking at the other types, and or joining all the datasets. One thing to note is that all users may not do a specific transaction, which would give us a reduced sized dataset. Note that there is some correlation structure within the time, which the clustering algorithm will not know. It may be a poor idea to cluster on a dataset joined by all transaction types weekly, in case the clustering learns a nonexistent structure from noise. I.e. for example, it's unlikely that week 5 repay and week 45 liquidation are related (compared to something like week 5 repay and week 6 liquidation), but with enough weeks (samples) the clustering might eventually pick up some noise.

```
# Separate datasets into all the types.
df.bytype <- list()
df.weekly <- list()
type_idx <- 1
for (i_type in df_tr$type %>% unique()) { # for each unique type
  df.bytype[[type_idx]] <- df_tr %>% filter(type == i_type)
  if (i_type == "swap") {
    next
  }
  if (i_type != "liquidation") {
      df.weekly[[type_idx]] <- df.bytype[[type_idx]] %>%
    group_by(user, week = format(as_datetime(timestamp), "%W-%Y")) %>%
    dplyr::summarise( wk_type_means = mean(amount, na.rm = TRUE), wk_transaction_type_N = n()) %>%
    spread(week, wk_type_means)
  }
  else { # Liquidation case. Need to use amountUSDCollateral
    df.weekly[[type_idx]] <- df.bytype[[type_idx]] %>%
    group_by(user, week = format(as_datetime(timestamp), "%W-%Y")) %>%
    dplyr::summarise( wk_type_means = mean(amountUSDCollateral, na.rm = TRUE), wk_transaction_type_N = 
```

```
    spread(week, wk_type_means)
  }

  type_idx <- type_idx + 1
}
```

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

```
# so df.weekly contains a list of transactions weekly done by users uniquely, by type.

## code for the general case, where we originally planned on grouping by type as well.
#full_test <- df_tr %>%
#    group_by(user, week = format(as_datetime(timestamp), "%W-%Y"), type) %>%
#    dplyr::summarise( wk_type_means = mean(amount, na.rm = TRUE), wk_transaction_type_N = n() ) %>%
#    spread(week, wk_type_means)
#full_test
```

For each transaction type, we do not scale since the magnitude of the amounts are important.

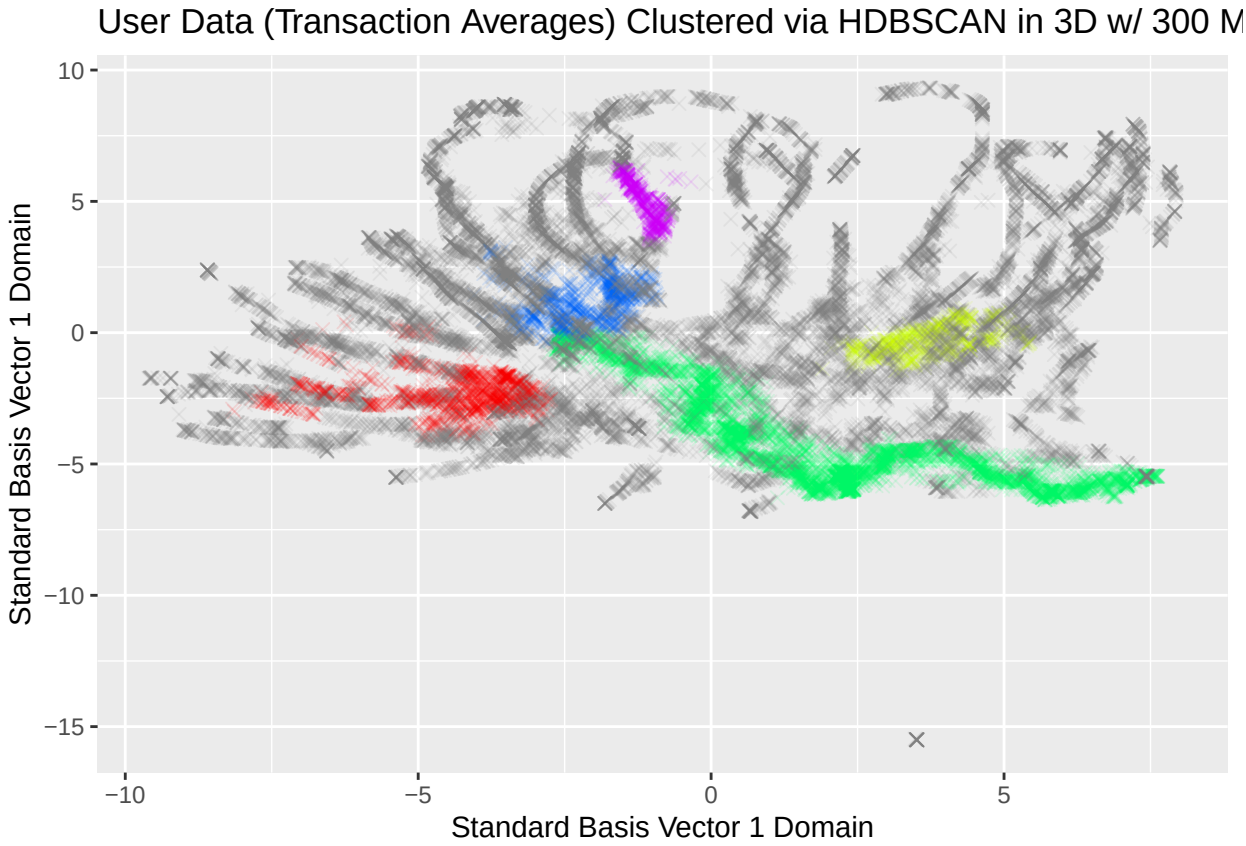
```
umap3_list <- list()
hdb3_list <- list()
for (i in c(1)) { # go through all 5  types
  df.temp.type <- df.weekly[[i]][, -which(names(df.weekly[[i]]) %in% c("wk_transaction_type_N"))] # don
df.temp.type <-df.temp.type %>% replace(is.na(.),0) # sparse dataframe
set.seed(43)
umap3_list[[i]] <- umap(df.temp.type, n_components = 3)
minimum_points = 300
set.seed(43)
hdb3_list[[i]] <- hdbscan(umap3_list[[i]]$layout, minPts = minimum_points)
}
```

Visualizations

UMAP

We start with UMAP (dimensionality reduction similar to t-SNE) once again. Comparing it to the previous clusters, we notice that the clustering from the weekly features have very different structure compared to the original full averaged dataset visualization we did in assignment 3. The main noticeable characteristics are the "lines" of users with similar weekly transaction patterns.

```
df_umap <- data.frame(umap3_list[[1]]$layout)
n_colors = max(hdb3_list[[1]]$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(hdb3_list[[1]]$cluster))
for (i in 0:n_colors) {
  color_labels[hdb3_list[[1]]$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) +
```



To take a look at some of the visualizations used previously and to compare them, we once again do some preprocessing on the data.

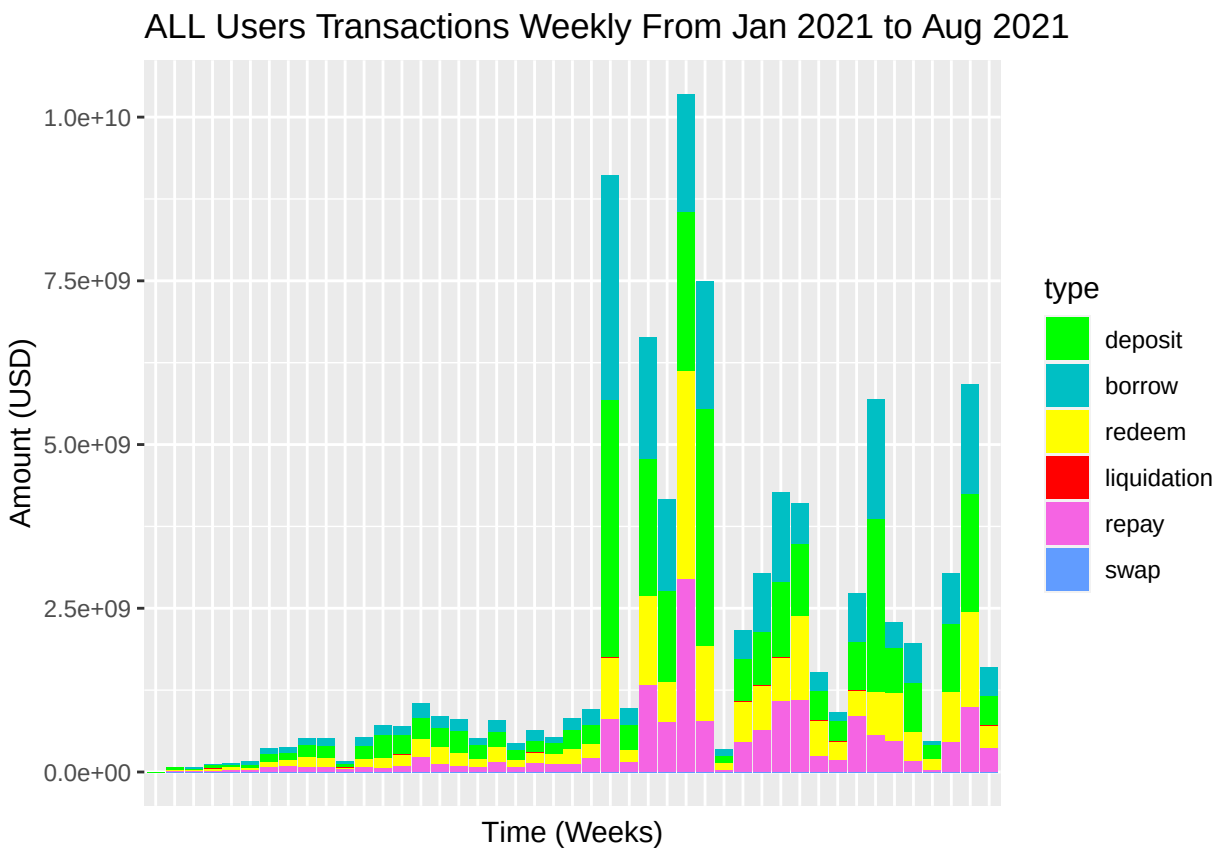
```
df_borrows <- df.weekly[[1]]
df_borrows["hdb_clusters"] <- hdb3_list[[1]]$cluster
df.4 <- df_borrows
N <- max(df.4$hdb_clusters)
dict <- table(df.4$hdb_clusters) #
df.dict <- as.data.frame(dict)    # Unused except to sort by top clusters in the next code block
```

Plotting Weekly Transactions of All Users

Once again we start by taking a look at the entire user transactions data, to have a baseline result to compare the plots by cluster to.

```
# generate N list of custom colors
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)

# Start plotting weekly by time. Start with all Users first
df_tr %>%
  dplyr::count(ymd_new, type, wt = amount, name = "amount") %>%
ggplot() +
```

```
  geom_bar(aes(x=ymd_new,y= amount, fill = type),
           stat='identity') + ggtitle("ALL Users Transactions Weekly From Jan 2021 to Aug 2021") + labs
          theme(
           axis.text.x=element_blank(),
           axis.ticks.x=element_blank())+
          scale_fill_manual("type", values = c("deposit"="green","borrow" = pgg[4], "redeem" = "yellow"
```



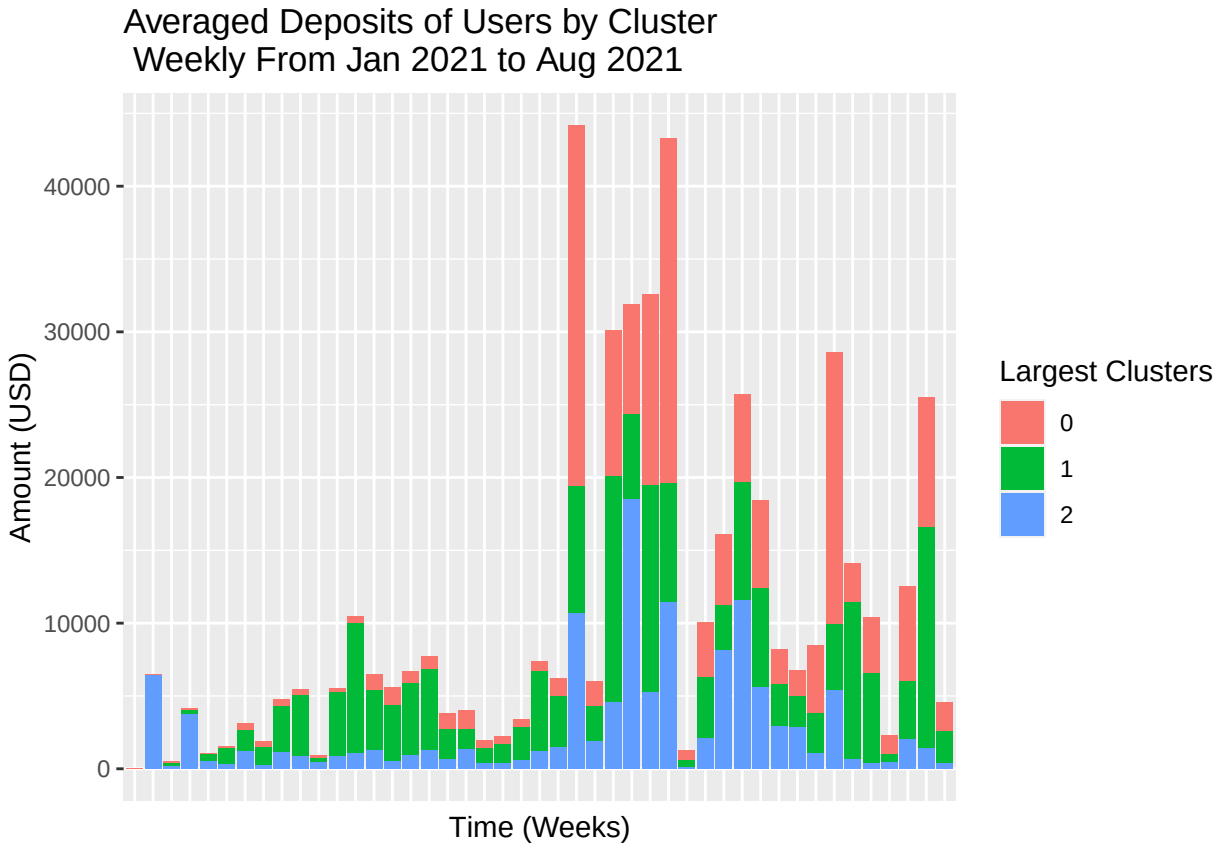

```
# Once again, a bit of data preprocessing.
deposit_only <- list()
liquidation_only <- list()
repay_only_list <- list()
redeem_only_list <- list()
borrow_o <- list()
for (i in 1:3) {
  dep_only_rows <- cluster_transactions[[i]][cluster_transactions[[i]]$type == "deposit"]
  liq_only_rows  <- cluster_transactions[[i]][cluster_transactions[[i]]$type == "liquidation"]
  rep_r <-  cluster_transactions[[i]][cluster_transactions[[i]]$type == "repay"]
  red_r <-  cluster_transactions[[i]][cluster_transactions[[i]]$type == "redeem"]
  bor_r <-  cluster_transactions[[i]][cluster_transactions[[i]]$type == "borrow"]
  dep_only_rows$cluster <- i
  liq_only_rows$cluster <- i
  rep_r$cluster <- i
  red_r$cluster <- i
  bor_r$cluster <- i
  deposit_only[[i]] <- dep_only_rows
  liquidation_only[[i]] <- liq_only_rows
  repay_only_list[[i]] <- rep_r
```

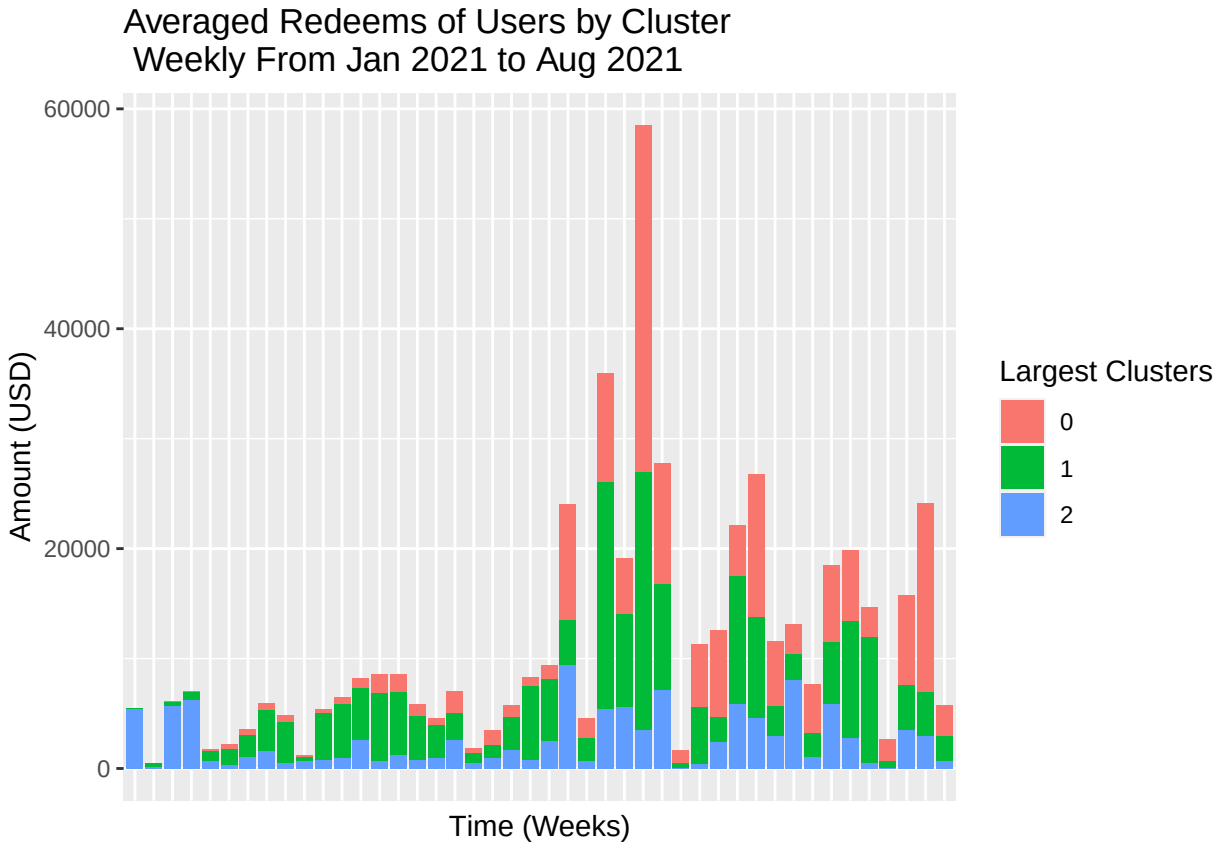
```
  redeem_only_list[[i]] <- red_r
  borrow_o[[i]] <- bor_r
  #print(dim(deposit_only[[i]]))
  deposit_only[[i]]$amount <- deposit_only[[i]]$amount/dim(deposit_only[[i]])[1]
  repay_only_list[[i]]$amount <- repay_only_list[[i]]$amount/dim(repay_only_list[[i]])[1]
  redeem_only_list[[i]]$amount <- redeem_only_list[[i]]$amount/dim(redeem_only_list[[i]])[1]
  borrow_o[[i]]$amount <-borrow_o[[i]]$amount/dim(borrow_o[[i]])[1]
  liquidation_only[[i]]$amountUSDCollateral <- liquidation_only[[i]]$amountUSDCollateral/dim(liquidatio
}
dep.c<- do.call("rbind", list(deposit_only[[1]], deposit_only[[2]], deposit_only[[3]]))
red.c <-do.call("rbind", list(redeem_only_list[[1]], redeem_only_list[[2]], redeem_only_list[[3]]))
liq.c <-  do.call("rbind", list(liquidation_only[[1]], liquidation_only[[2]],liquidation_only[[3]]))
rep.c <- do.call("rbind", list(repay_only_list[[1]], repay_only_list[[2]], repay_only_list[[3]]))
bor.c <- do.call("rbind", list(borrow_o[[1]], borrow_o[[2]], borrow_o[[3]]))

# Plot by transaction type weekly
loop_labels <- c("deposit", "redeem", "repay", "borrow", "liquidation")
cap_loop_labels <- c("Deposits", "Redeems", "Repays", "Borrows", "Liquidations")
facet_data_list <- list(dep.c, red.c, rep.c, bor.c, liq.c)
```

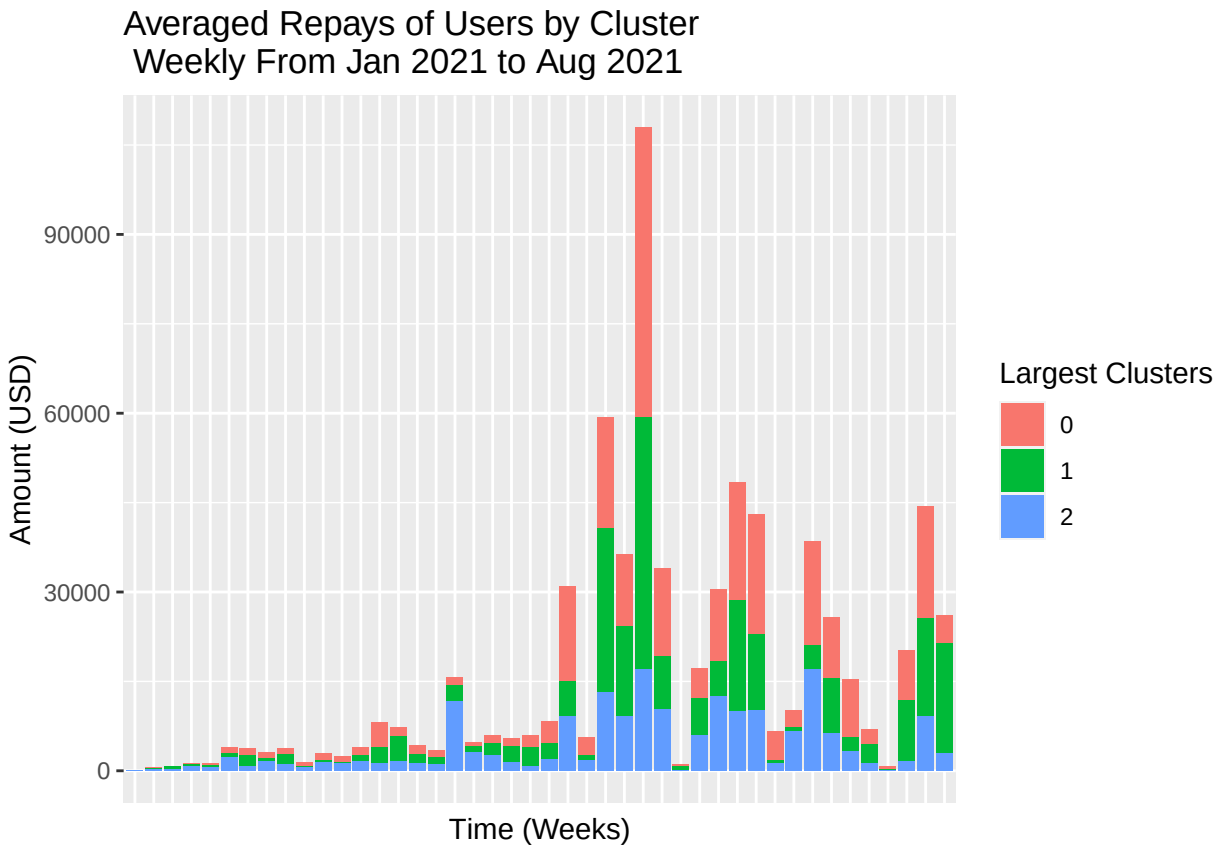
Plotting Weekly Transactions by Type, Stacked by Cluster

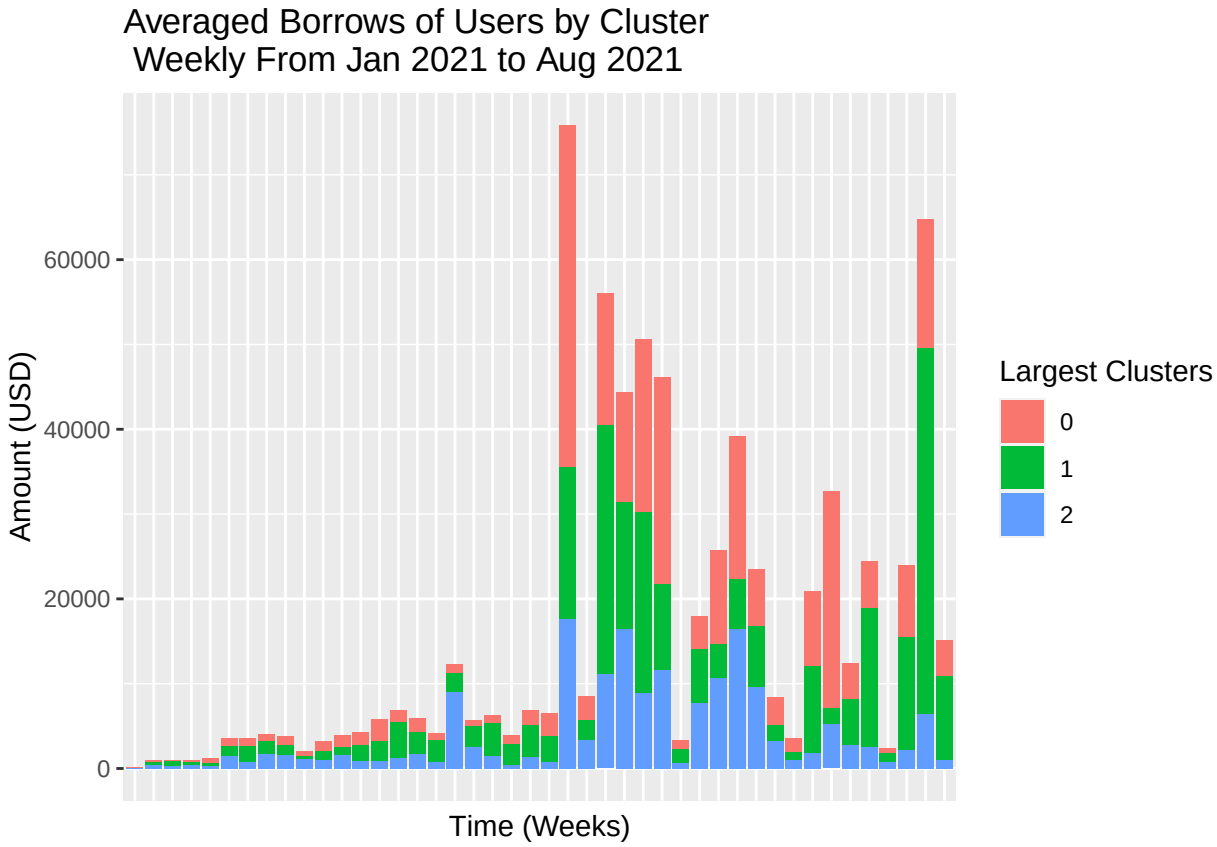
Here, we plot the averaged (by number of unique users per cluster) of amount USD deposits, redeems, repays, and borrows weekly. The stacked values indicate the amount for that specific cluster. Remark: Recall the weekly features were generated based on borrows, so the clustering will likely find distinct differences mainly in the borrows. However, we know that all the transaction types are generally highly correlated weekly.

```
fct_plot_list <- list()
for (m in 1:4) {
  pltobj <- ggplot(dplyr::count(facet_data_list[[m]] ,ymd_new, cluster, wt =amount, name = "amount")) +
  geom_bar(aes(x=ymd_new,y= amount, fill = as.factor(cluster)),
           stat='identity') + ggtitle(paste0("Averaged ", cap_loop_labels[m], " of Users by Cluster\n W
          theme(
           axis.text.x=element_blank(),
           axis.ticks.x=element_blank()) + labs(fill="Largest Clusters")
  print(pltobj)
  fct.plt <- pltobj +  facet_grid(cols = vars(cluster))
  fct_plot_list[[m]] <-fct.plt



}
```

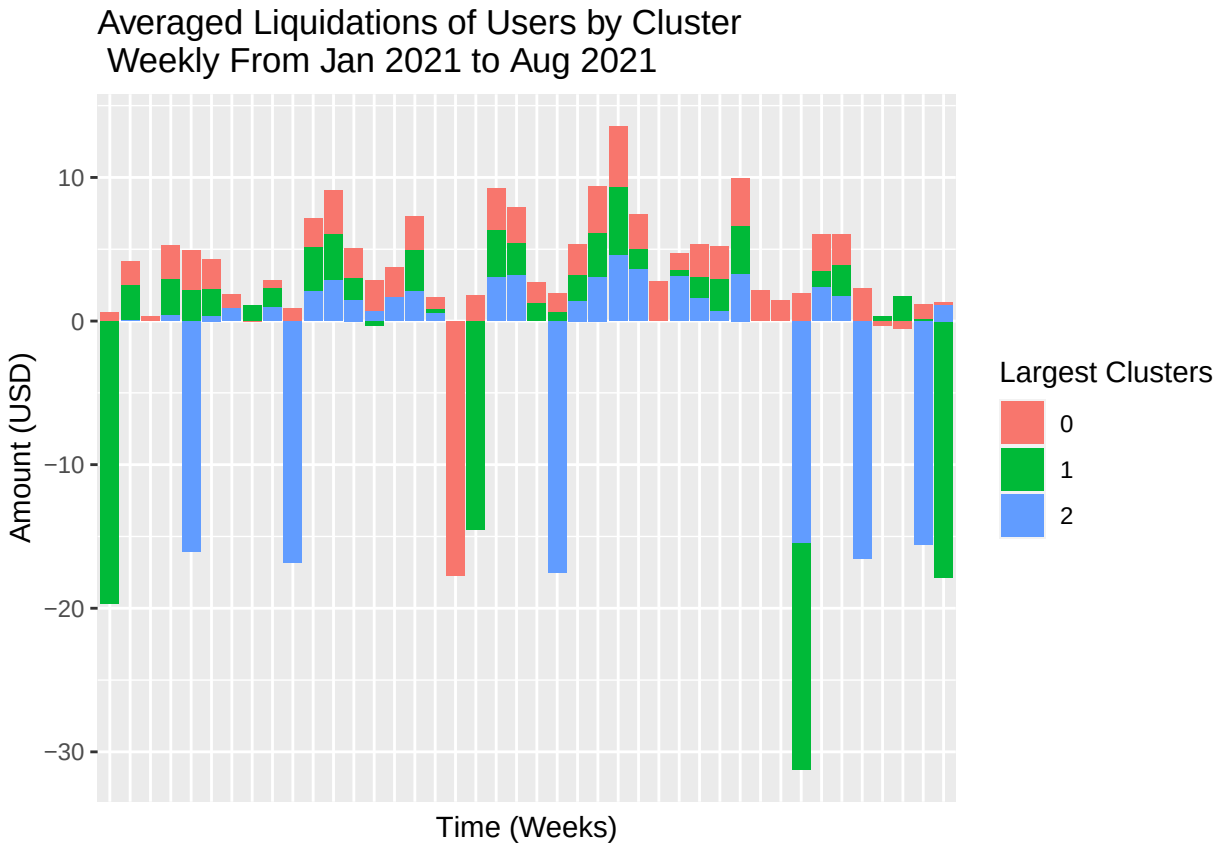

Averaged Deposits of Users by Cluster
Weekly From Jan 2021 to Aug 2021
Amount (USD)
40000
30000
20000
10000
0
Time (Weeks)
Largest Clusters
0
1
2


Averaged Redeems of Users by Cluster
Weekly From Jan 2021 to Aug 2021
Amount (USD)
60000
40000
20000
0
Time (Weeks)
Largest Clusters
0
1
2


Averaged Repays of Users by Cluster
Weekly From Jan 2021 to Aug 2021
Amount (USD)
90000
60000
30000
0
Time (Weeks)
Largest Clusters
0
1
2


Averaged Borrows of Users by Cluster
Weekly From Jan 2021 to Aug 2021
Amount (USD)
60000
40000
20000
0
Time (Weeks)
Largest Clusters
0
1
2

```
  pltobj <- ggplot(dplyr::count(facet_data_list[[5]] ,ymd_new, cluster, wt = amountUSDCollateral, name =
  geom_bar(aes(x=ymd_new,y= log10(amountUSDCollateral), fill = as.factor(cluster)),
           stat='identity') + ggtitle(paste0("Averaged ", cap_loop_labels[5], " of Users by Cluster\n W
          theme(
           axis.text.x=element_blank(),
           axis.ticks.x=element_blank()) + labs(fill="Largest Clusters")
  print(pltobj)
```



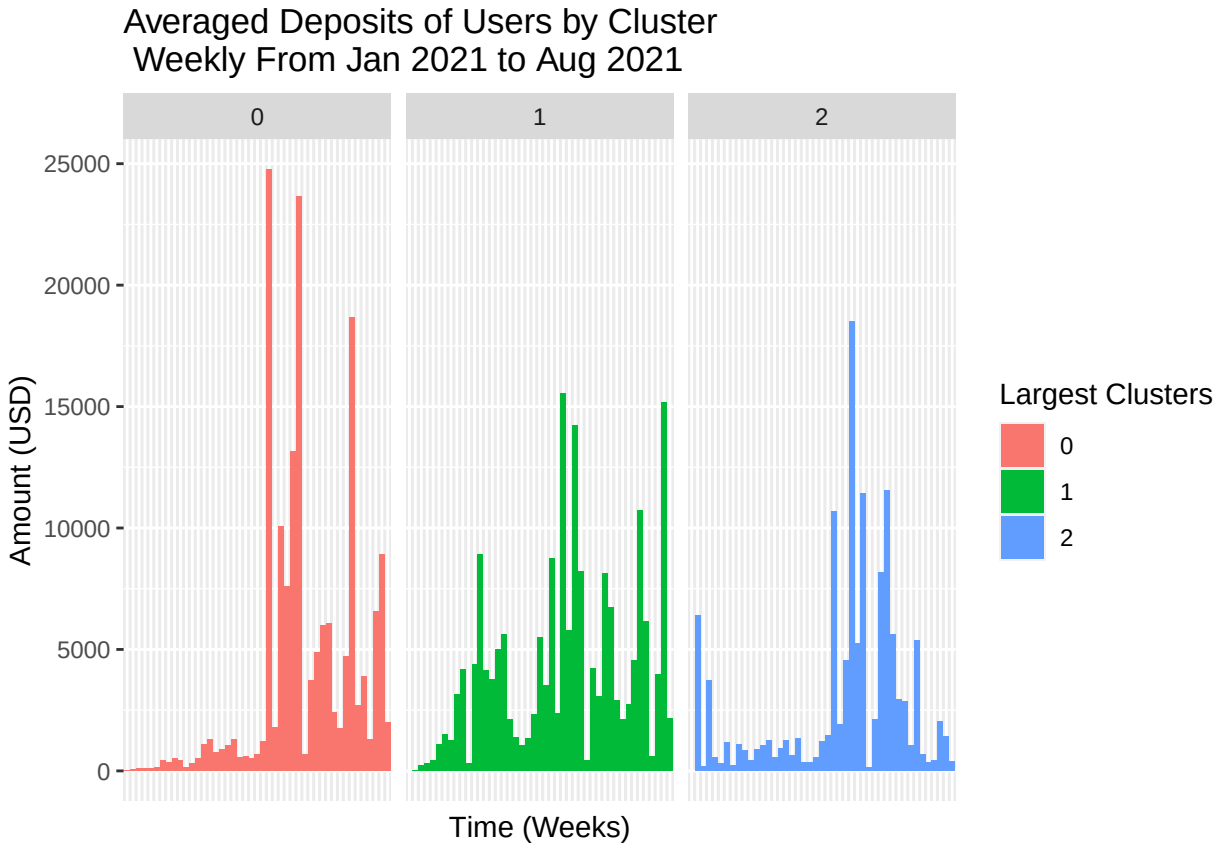

```
  fct.plt <- pltobj +  facet_grid(cols = vars(cluster))
  fct_plot_list[[5]] <- fct.plt

facet_data_list[[1]]$cluster %>% head
```

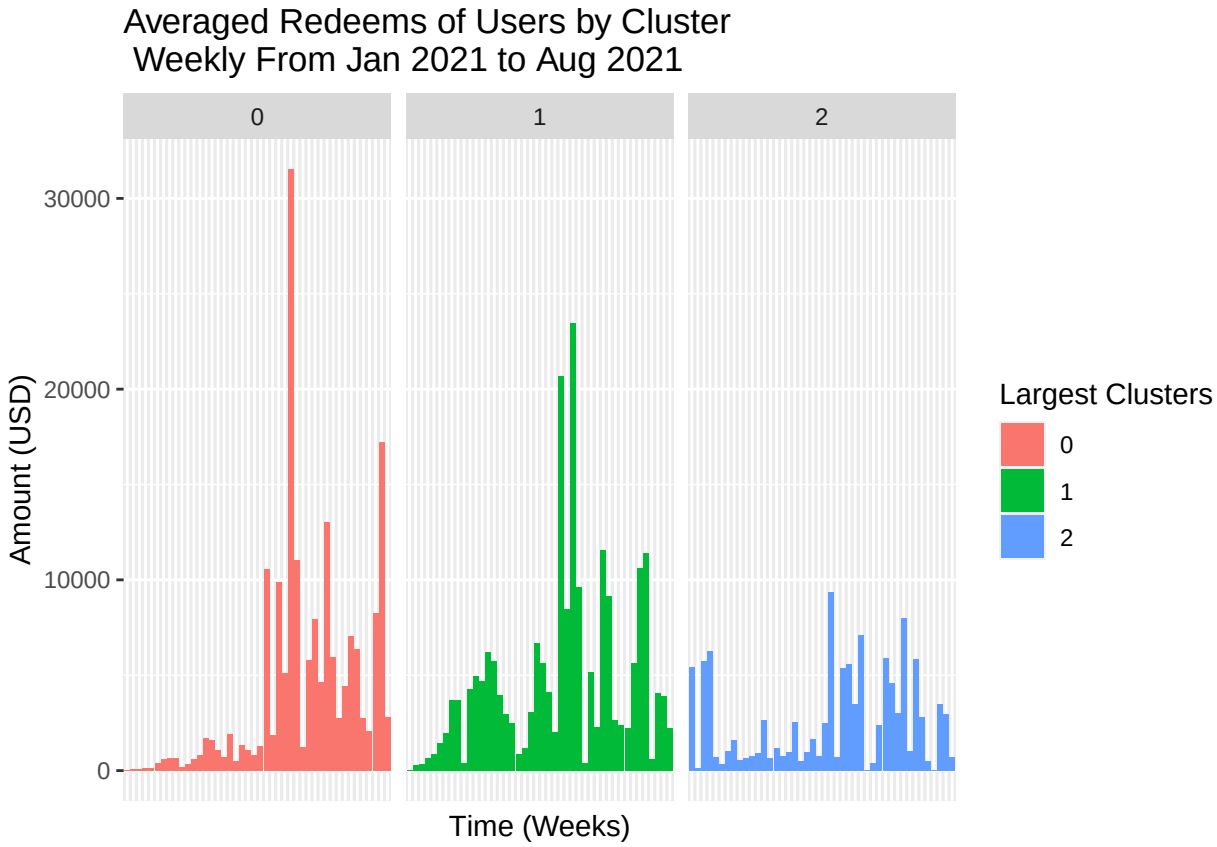
```
## [1] 0 0 0 0 0 0
## Levels: 0 1 2 3 4 5
```

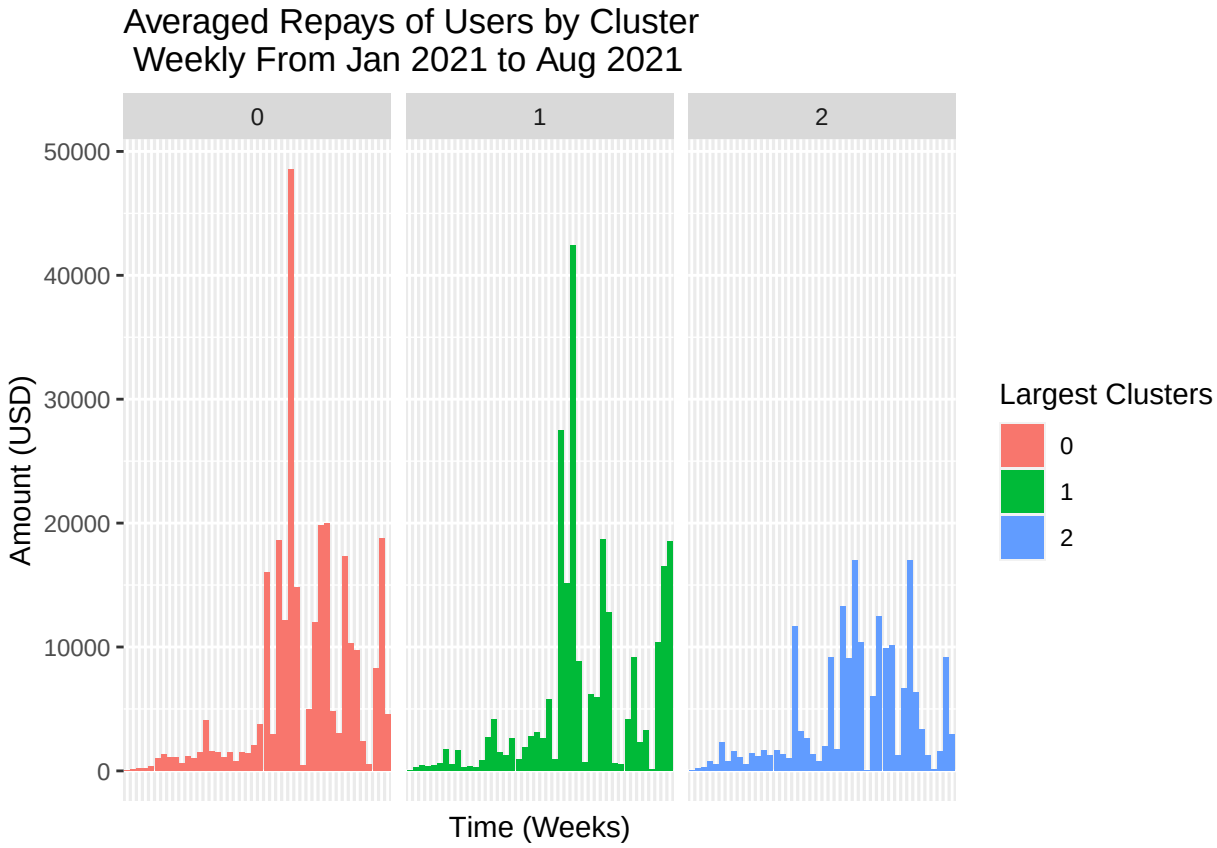
Side by Side Comparisons of Clusters & Averaged Transaction Types Weekly

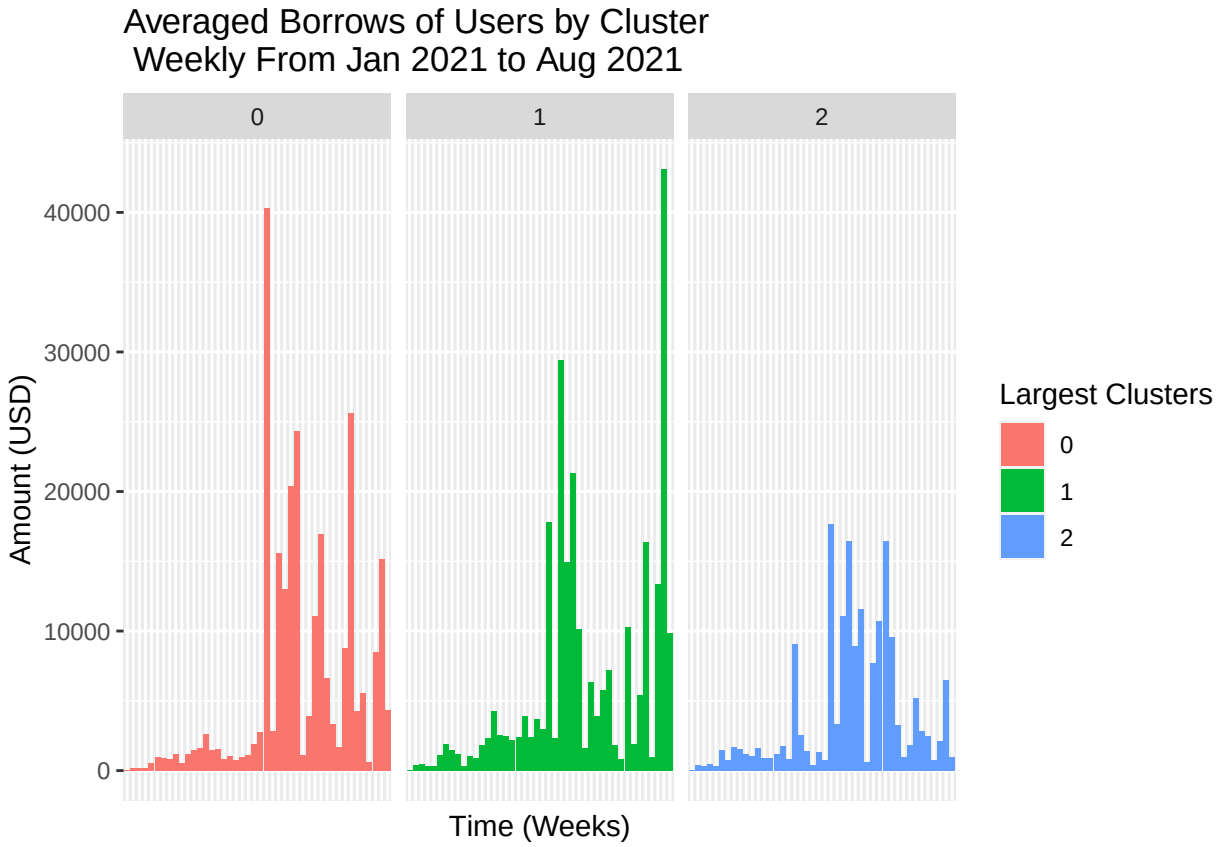
WE follow it up with facet_grid.

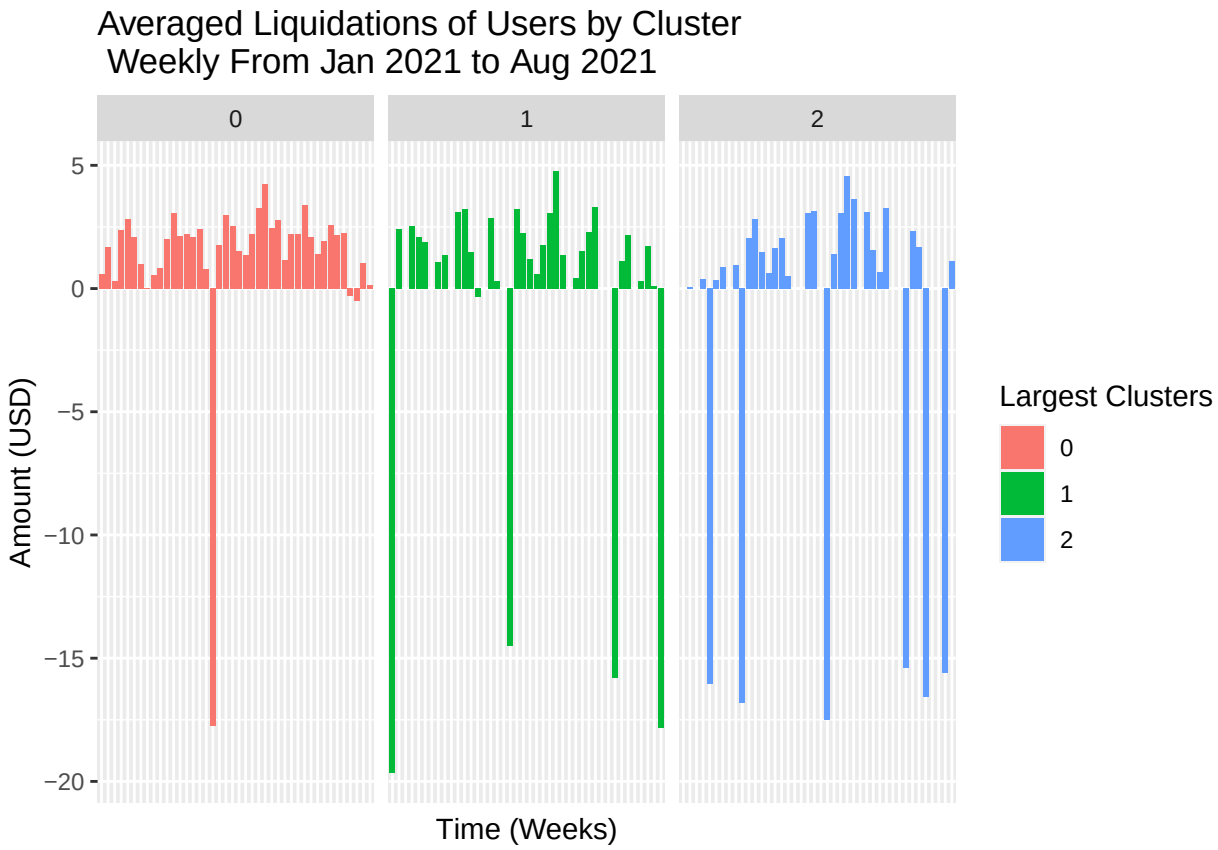
```
## now, facet plot
for (m in 1:5) {
  print(fct_plot_list[[m]])
}
```


Averaged Deposits of Users by Cluster
Weekly From Jan 2021 to Aug 2021
0
1
2
25000
20000
15000
10000
5000
0
Amount (USD)
Time (Weeks)
Largest Clusters
0
1
2


Averaged Redeems of Users by Cluster
Weekly From Jan 2021 to Aug 2021
0
1
2
30000
20000
10000
0
Amount (USD)
Time (Weeks)
Largest Clusters
0
1
2


Averaged Repays of Users by Cluster
Weekly From Jan 2021 to Aug 2021
0
1
2
50000
40000
30000
20000
10000
0
Amount (USD)
Time (Weeks)
Largest Clusters
0
1
2


Averaged Borrows of Users by Cluster
Weekly From Jan 2021 to Aug 2021
0
1
2
40000
30000
20000
10000
0
Amount (USD)
Time (Weeks)
Largest Clusters
0
1
2



Once again, cluster 1 seems to have overall the largest peaks in the middle months, across all types except borrows. There are definitely weekly differences compared to the original clustering. For example, we can see in cluster 2, users deposited a noticeable amount at the 5-15 week range, whereas cluster 1 has minimal amounts, and cluster 3 has mainly high averaged deposits within the first 3 weeks. Overall, it seems like cluster 3 makes has the smallest amount of transactions. Taking a look at the liquidations, We see that cluster 0 has only 1 week of the micro-liquidations, compared to cluster 3 which has quite a significant number of them. These micro-liquidations were intentionally left in the data, as only a few distinct type of users have these occurrences which may be interesting to learn.

Note that these are only the largest 3 clusters, out of 5. A smaller granularity can be used on the clustering to pick up more fine/local similarities between features, that may give us users that are easier to define.

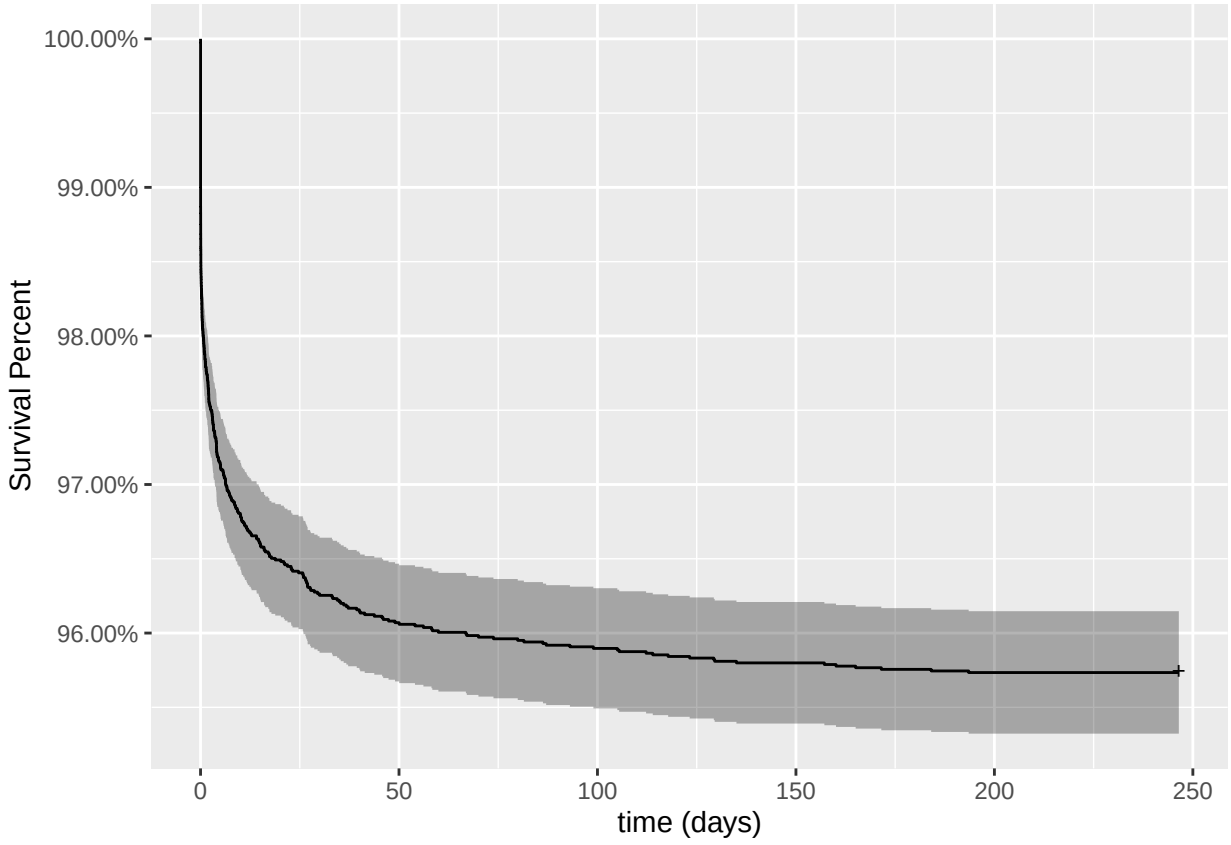
Survival Plots

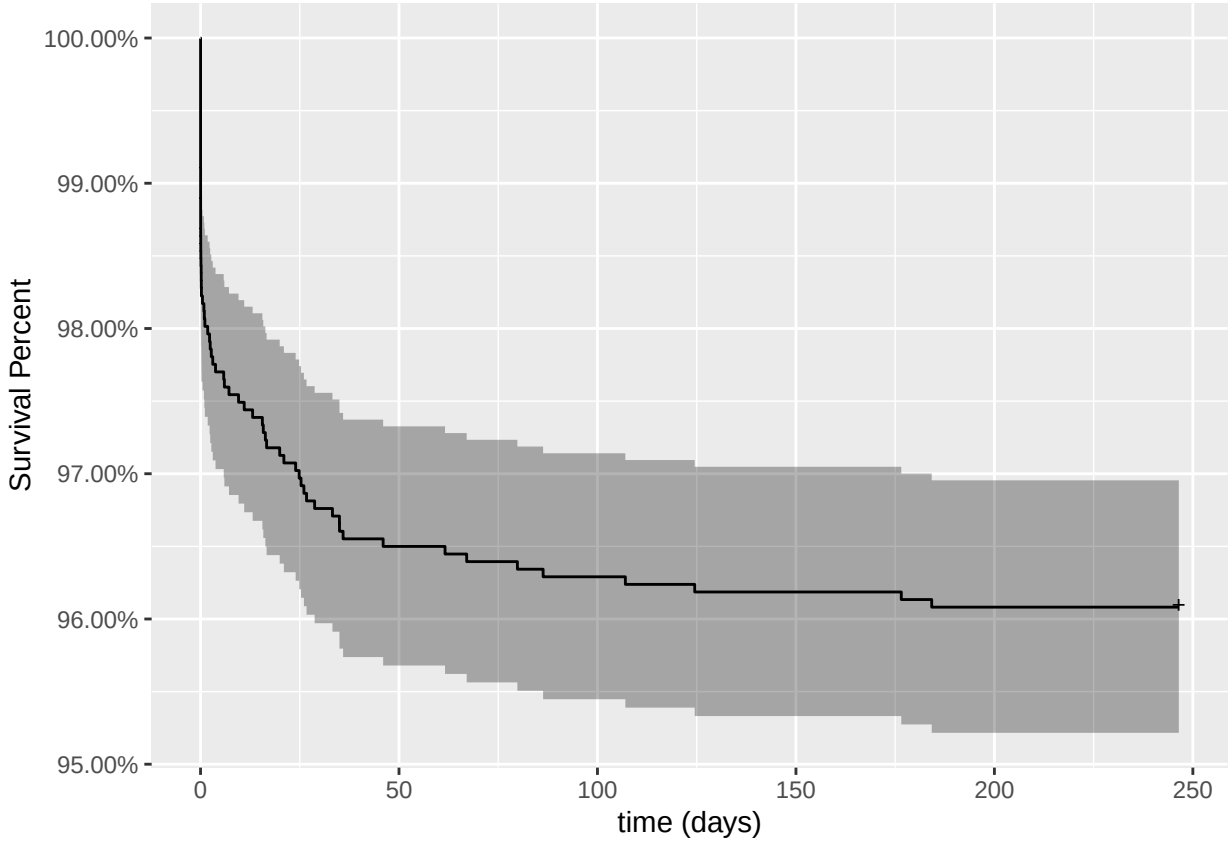
We finally take a look at the survival plots of first borrow to first repay of the users by cluster.

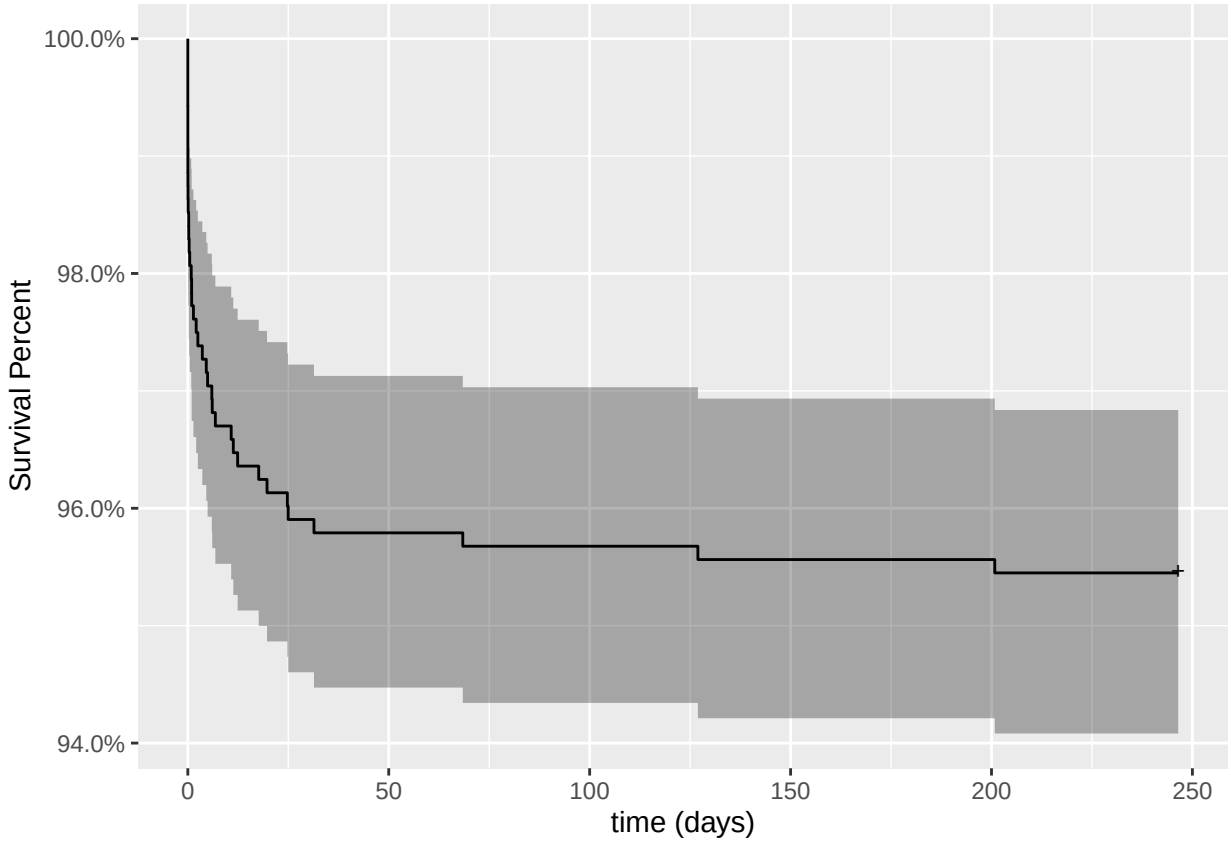
```
generateSurvival <- function(start, end)
{

  dataSet <- left_join(end,start,by="user") %>%
    dplyr::rename(endTime=timestamp.x) %>%
    dplyr::rename(startTime=timestamp.y) %>%
    group_by(user) %>%
    dplyr::summarise(timeDiff=case_when( min(startTime)-min(endTime) >0 ~   min(startTime)-min(endTime)
    dplyr::mutate(status=case_when(timeDiff==21294796 ~ 0, timeDiff<=0 ~ 0, timeDiff>0 ~ 1)) %>%
    dplyr::select(user,timeDiff,status)

  km <- with(dataSet, Surv(timeDiff/86400, status))
```

```
  km_fit <- survfit(Surv(timeDiff/86400, status) ~ 1, data=dataSet)
  summary(km_fit, times = c(1,30,60,90*(1:10)))
  p1 <- autoplot(km_fit,xlab="time (days)",ylab="Survival Percent",title="Survival Analysis")
  return (p1)
}
```

```
for (i in 1:3) {
borrows <- cluster_transactions[[i]] %>%
  filter(type=="borrow")

repays <- cluster_transactions[[i]] %>%
  filter(type=="repay")


print(generateSurvival(borrows, repays))
}
```




100.00%
99.00%
98.00%
97.00%
96.00%
95.00%
Survival Percent
0
50
100
150
200
250
time (days)


100.0%
98.0%
96.0%
94.0%
Survival Percent
0
50
100
150
200
250
time (days)

By observation, the survival plots look quite similar - Cluster 2 seems to have the "fastest" percentage of death (i.e. for cluster 3 consider at 25th day, survival is 96%, compared to 96% for cluster 2. Cluster 1 is in the middle, at roughly 96.5%). These percentage differences may not necessarily be significant, as the variance is quite high. Definitely some further analysis should be done.

Further Analysis:

Coming up with improved ways to generate weekly features might further improve the clusters. As we mentioned, one could also compute the clusters on the joined dataset (users' weekly transaction averages for all types, so N weeks * D types total time features). Also, looking at thresholding the micro-liquidations might be something try. Although I think they're definitely good to keep since they should be a cluster in itself, we can also discard them as noise, and so it might be interesting to see the new clusters. Finally, looking at significance of the survival plot curves may also be useful to quantitatively declare differences between the data, rather than just visually inspecting.