MATP-4910 Final Project

DeFi

Duke Kwon

15 November 2021

Contents

Github Info

- github repository: https://github.rpi.edu/DataINCITE/IDEA-Blockchain/blob/master/DefiResearch/StudentNotebooks/FinalNotebook/dar_final_kwond2_30nov2021.Rmd
- Your github ID: kwond2
- Final notebook: dar_final_kwond2_30nov2021.Rmd
- External Notebook References
  - https://github.rpi.edu/DataINCITE/IDEA-Blockchain/blob/master/DefiResearch/StudentNotebooks/Assignment03/kwond2_assignment3.pdf
- Github Issues
  - https://github.rpi.edu/DataINCITE/IDEA-Blockchain/issues/86
- Contributions
  - The visuals for the density plots (Cole) and the survival plots (Soumya) were used and modified.
  - The externally referenced notebooks include shared code which is made distinct.
- Note to the Reader:

– This notebook is self-sufficient - it contains all the code embedded to reproduce the analysis given that the file structure of the data is preserved. The PDF provides very little code to be read without clutter.

Overview & Problems Tackled

In the decentralized finance (DeFi) space, users have significant freedom of the various transactions they can make on the blockchain at any time, from a large selection of cryptocurrencies, and an abundance of blockchain tech that mimics traditional financial intermediaries. Yet, compared to traditional finance, the DeFi space entails a substantially more risk and volatility. From the general problem of clustering in finance, we mainly explore the subset of characterizing users and identifying groups based on their transaction data, such as high risk groups that make liquidate often or a significant amount, lower risk groups who only deposit to earn a stable return rate, etcetera.

Particularly, we look at more complex methods of dimensionality reduction (UMAP) and clustering (HDBSCAN) on a modified weekly mean transaction encoded dataset to group our users. From our analysis, we obtain distinct clusters (visually) of users, with transaction types, amounts, and frequencies that have reasonable hypotheses (i.e. group of users with high risk behaviors have high liquidation amounts and frequencies). Finally, we elaborate on a general formulation of the UMAP + HDBSCAN procedure that can be used on other clustering tasks, and other analysis that can be done on the clustered users (i.e. look into frequency of coins by transactions, look into statistically significant survival analysis).

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

Data Description

Note: This notebook is meant to be run within the "IDEA-Blockchain/DefiResearch/StudentNotebooks/FinalNotebook" directory. All datasets are access from "IDEA-Blockchain/DefiResearch/Data".

Secondary Datasets

Our experiments first read in an AAVE User transactions dataset "transactions_2.csv" as "df_tr" taken with the time range of November 30th, 2020 to August 19th, 2021. It consists of 481,519 rows of user transactions from 6 different transaction types - borrow, repay, liquidation, deposit, redeem, and swap, along with details (29 columns total) regarding those transactions such as amount. The dataset is misleadingly abundant with NA entries, as a feature for a specific transaction type may not have a value for another transaction type. The difference between the original "transactions.Rds" is the fixing the users to start from 1,2, along with a few other columns as summary statistics.

The columns are brought to 31 by adding two more columns with converted "timestamp" to a datetime column "ymd", and a Year-Month-Week string column, "ymd_new", for convenience.

```
# We assume we're in the FinalNotebook Directory.
data_path <- paste(dirname(dirname(getwd())), "/Data", sep = "")
setwd(data_path)
df_tr <- read_csv("transactions_2.csv")
#For convenience, the Lubridate package is used here to add two more columns (total 26+2=28) from "time
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
print("Dataset Range:");print("Earliest:")
```

```
## [1] "Dataset Range:"
```

```
## [1] "Earliest:"
```

```
df_tr$timestamp %>% min() %>% as_datetime()
```

```
## [1] "2020-11-30 23:11:40 UTC"
```

```
print("Final:")
```

```
## [1] "Final:"
```

```
df_tr$timestamp %>% max() %>% as_datetime()
```

```
## [1] "2021-08-19 04:51:33 UTC"
```

```
df_tr$type %>% unique()
```

```
## [1] "borrow"      "repay"       "liquidation" "deposit"     "redeem"
## [6] "swap"
```

For tidiness, we omit the outputs. The Rmd contains the full view.

```
print("AAVE Transactions Dataset:")
df_tr %>% head(5)
df_tr %>% str()
print("Summary Statistics:")
df_tr %>% summary()
print("Column Names:")
df_tr %>% colnames()
```

The second data we indirectly use is a modified form of the AAVE User transactions dataset, "df.users", which contains summary statistics by unique user. It consists of 40,659 rows of unique users, with 29 columns/attributes signifying mean statistics for each transaction type (i.e. number of borrows, mean liquidation, etc). Generating this dataset can be sourced from "https://github.rpi.edu/DataINCITE/IDEA-Blockchain/blob/master/DefiResearch/StudentNotebooks/users.Rmd".

```
# We assume we're in the FinalNotebook Directory.
data_path <- paste(dirname(dirname(getwd())), "/Data", sep = "")
setwd(data_path)
df.users <-read.csv("df_users.csv") ## read users parsed csv
df.users <- df.users[, -which(names(df.users) %in% c("X"))]
```

Once again, the data can be accessed via the Rmd.

```
print("Unique Users Transactions Averaged:")
df.users %>% head(5)
df.users %>% str()
print("Summary Statistics:")
df.users %>% summary()
print("Column Names:")
df.users %>% colnames()
```

Primary Dataset: Generating Weekly Average Features

The primary dataset we use is once again based on the AAVE User transactions dataset, which is contained in the list "df.weekly" (Accessible via "df.weekly[[i]]"). Instead of having a single average value for each unique transaction & user, here we generate a dataset that directly encodes weekly averages - for each week, we compute the users averages for each type of transaction, and store it as a feature. Since there are multiple types of transactions, the "df.weekly" list is indexed by transaction type. For example, "df.weekly[[1]]" would contain unique users' weekly averaged amounts for borrows. We focus on the borrows transactions only, as a majority of the users have borrow transactions. Though, one thing to note is that all users may not do a specific transaction (i.e. there are definitely a large subset of users who don't have liquidations).

For our analysis on the borrows, "df.weekly[[1]]", the data consists of 16,607 unique users, with 41 columns, where 39 of the columns represent a week within our AAVE dataset time frame, which holds the averaged amount for borrows in that week.

(Source Hidden)

```
print("Unique Users Averaged Weekly Borrows:")
df.weekly[[1]] %>% head(5)
df.weekly[[1]] %>% str()
print("Summary Statistics:")
df.weekly[[1]] %>% summary()
print("Column Names:")
df.weekly[[1]] %>% colnames()
```

Results

Problem 1

Given that we would like to group users by their transactions that may be useful in characterizing behaviors of risk or profit, a standard dimensionality reduction and clustering method was used (PCA + K-Means). The reference notebook kwond2_assignment3.Rmd (linked above via Github) provides an in-depth analysis of the comparison between the PCA + K-Means versus our method here, UMAP + HDBSCAN. Furthermore, our original AAVE data encoded a high level, full timespan view of the transactions. This resulted in our clusters yielding poorer results when looking for patterns at the weekly transaction level.

To circumvent these issues, a new, weekly dataset was generated which encoded weekly features (specified above), along with our alternative dimensionality reduction + clustering method.

Methods

The dataset used was "df.weekly[[1]]", specified in the datasets section of this notebook, which consists of unique user average borrow amounts per week as features.

UMAP (uniform manifold approximation projection), effectively learns a manifold approximation of the distribution our data lies in (in the simplest case, effectively generating a directed weighted graph via a style of knn), and maps the approximation to a lower dimensional space by minimizing a style of cross entropy error, iteratively minimizing the objective so that close points are weighted close together, and far points are pushed apart.

Afterwards, under the assumption that weekly data may have some special, nonlinear structures, we cluster via a density based approach by HDBSCAN (hierarchical density-based spatial clustering and applications). HDBSCAN is effective in mitigating the issues of some data not being close to means centers in a simpler k-means approach, but being locally close to other points that are locally close to the mean centers. We use a moderately high Minimum Points hyperparameter (100), which determines minimum points required to be a cluster. For the other parameters, we use the default which tend to be good enough for our analysis. The clustering is done in a higher dimensional space (5D) to preserve some of the finer details of the data.

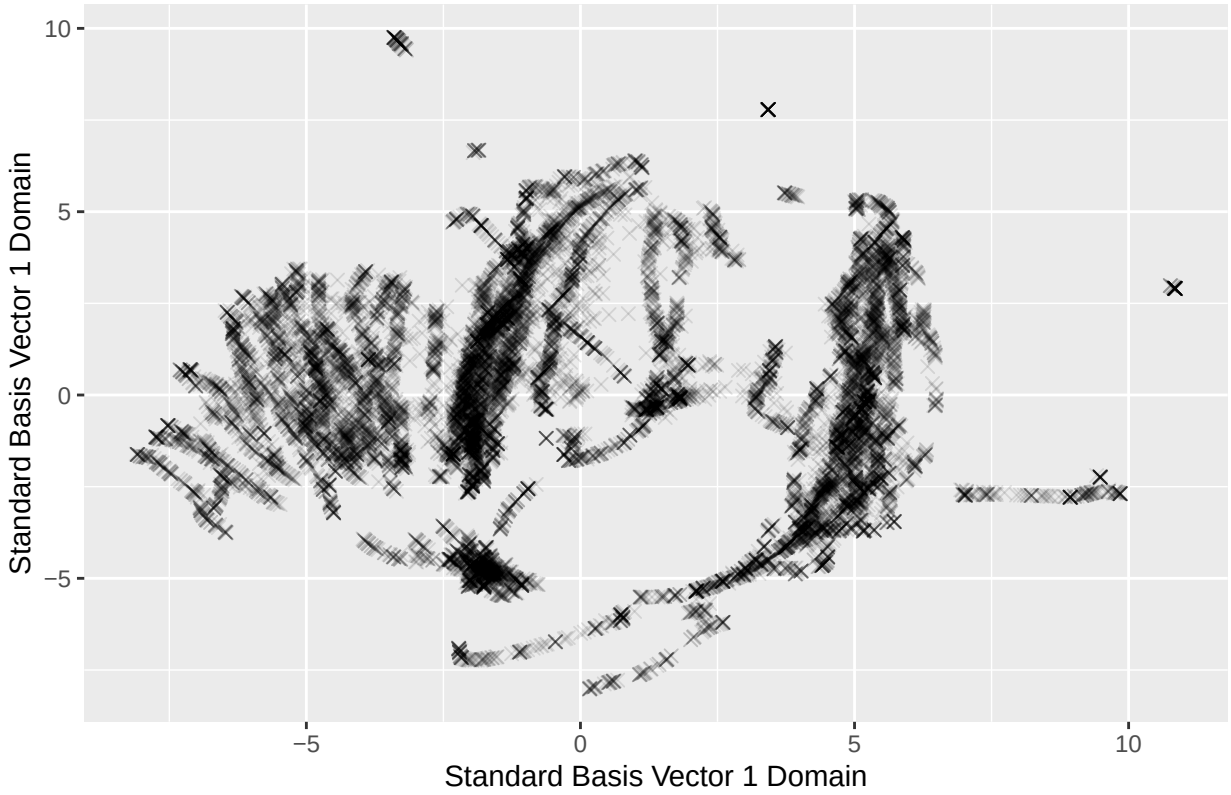
One thing to note is that HDBSCAN does not require the user to specify the number of clusters as a hyperparameter, and in our case is mainly dependent on the MinPts parameter as mentioned above. In our example, we produce 4 clusters, one mainly being an outlier group.

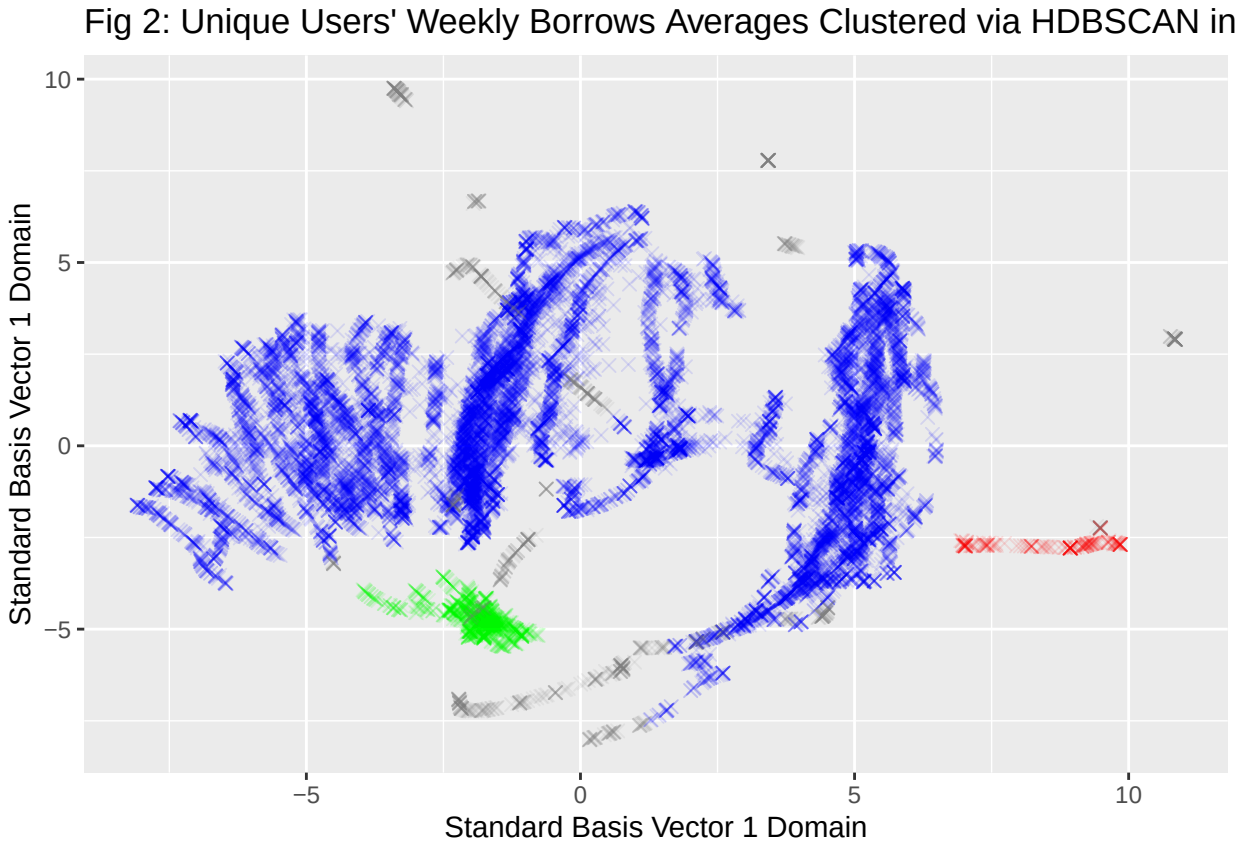
Figure 1 is the resulting 2D projection of the UMAP dimensionality reduction. Default parameters were used regarding tolerance/convergence/number of iterations, as it is an iterative technique. The algorithm is not deterministic unless a random seed is set.

Figure 2 is the resulting users colored by cluster. HDBSCAN was run on the original dataset mapped to 5D. We omit a legend for the sake of large cluster outputs (HDBSCAN learns the number of clusters unlike K-means, and so may produce a large number of clusters dependent on the HDBSCAN hyperparameters). In

grey, we have outlier points. The number of outliers are highly dependent on both the UMAP and HDBSCAN parameters.

Fig 1: Unique Users' Weekly Borrows Averages Projected via UMAP





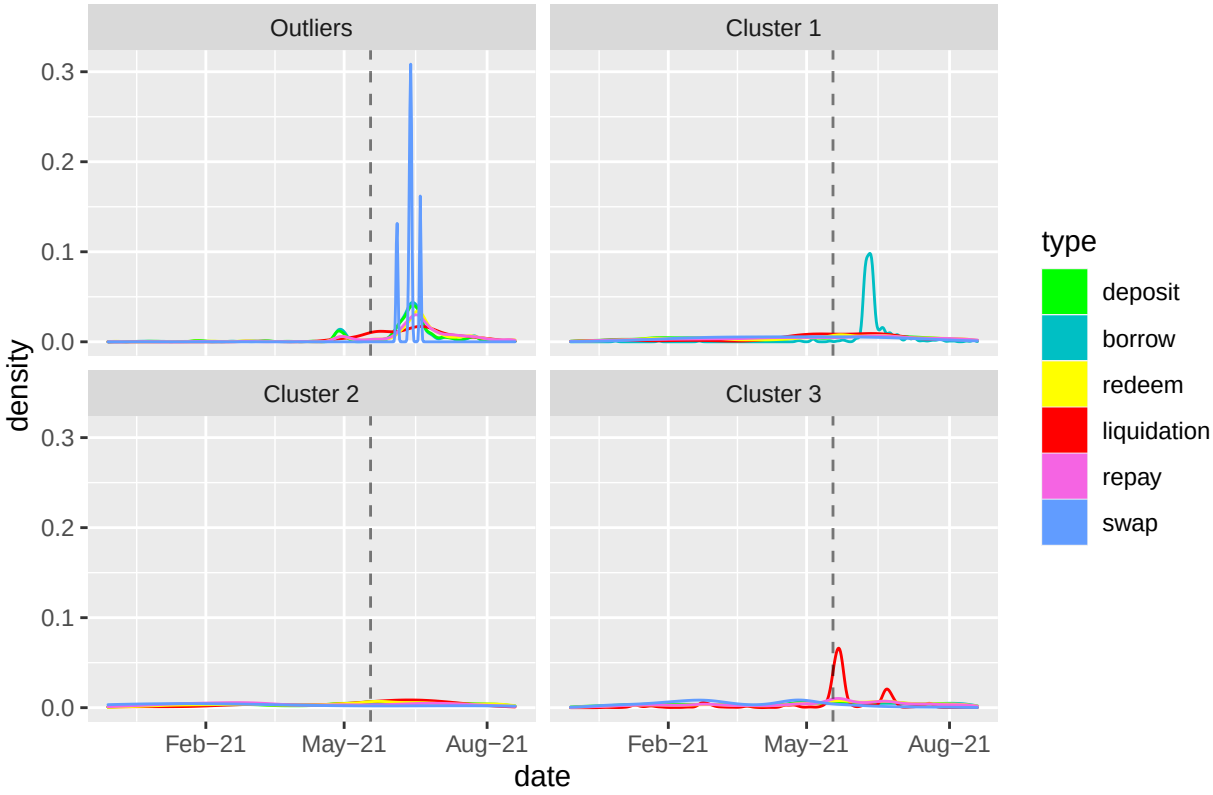
Results

The bulk of our research can be summed into transaction density plots of the various clusters (see appendix and referenced kwond2_assignment4.Rmd for more analysis, and an alternate training run with different clusters).

In figure 3, the clusters were produced using a random seed of 43, 5 dimensions for UMAP, and 50 MinPts for HDBSCAN. The outlier group had users with large density of swaps within May-21, June-21, and July. Cluster 1 on the other hand grouped users who did moderate amount of borrows within that similar timeline. Cluster 2 does not show much activity throughout time, which likely means they are generally inactive users, doing very few transactions, or transactions spaced out through time. Finally, cluster 3 shows a large density of liquidations between May-21 and June-21, which are likely users who were liquidated from the instability caused by the China crypto ban fiasco in Mid-late May.

```
## Loading required package: anytime
```

Fig 3: Transaction Types Over Time by UMAP+HDBSCAN User Clusters



We also output the total number of each type of transaction for each cluster in Table 1. The large spikes in densities for swaps for the outlier group is likely a few users with abnormally large valued swaps, which would make sense as outliers. In cluster 2, despite having a substantial number of transactions, the densities show relatively low probabilities of each type, as the amounts are likely very small. The appendix shows a bit more of this, with stacked bar plots comparing the amounts per type and by cluster.

Table 1: Number of Transactions by Cluster

Cluster	Borrows	Repays	Liquidations	Deposits	Redeems	Swaps
Outliers	1468	901	71	1574	760	5
1	384	1698	103	76210	67291	6
2	8352	1229	66	43862	19514	14
3	84773	56714	6049	70360	39140	975

Discussion

Generally, it seems to be that our new features are not much of an improvement from the non-weekly features, at least without proper tuning of the number of UMAP dimensions and the min points in HDBSCAN (if we compare it to our alternate training run, which is in the appendix). Three clusters do not seem like enough to characterize a few ten thousand users. The issue is likely due to the fact that we limit our dataset to users who have had at least one borrow within our timeframe. For example, it may be the case that a user borrowed earlier than our timeframe, but then liquidated within our timeframe. This user would not be included within our borrow dataset.

However, within the appendix, an alternate training run was done - we are certainly able to cluster users with specific behaviors to a weekly transaction scale.

The main limitation of the results are the certainty of our hypothesis/intuition. Despite the fact that it may look like our clusters are "good", it may be beneficial to run some cluster metrics, or test the "accuracy" of the clusterings with a synthetic, labeled data set (i.e. manually encode high liquidations or high risk behavior).

Problem 2

Regarding our clusterings, we've done very little work connecting it to other student work which may provide more insight into the quality of our clusters. We hope that survival analysis of borrows to repays of users by cluster show noticeable differences that match the transaction behavior of the density and bar plots (appendix).

Methods

We use the survminer library (which can be found in the references) to generate our plots. The primary type of survival plot we look at is the first borrow to the "death" of first repay.

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

  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") + ylim(c(
  return (p1)
}
```

Results

It terms of interpreting the survival plots, it seems that users from cluster 3 generally repay faster/repay in general compared to the overall dataset. We discuss more about figures 4-7 below.

Fig 4 : Cluster 0 : First Borrow to Repay Survival Analysis

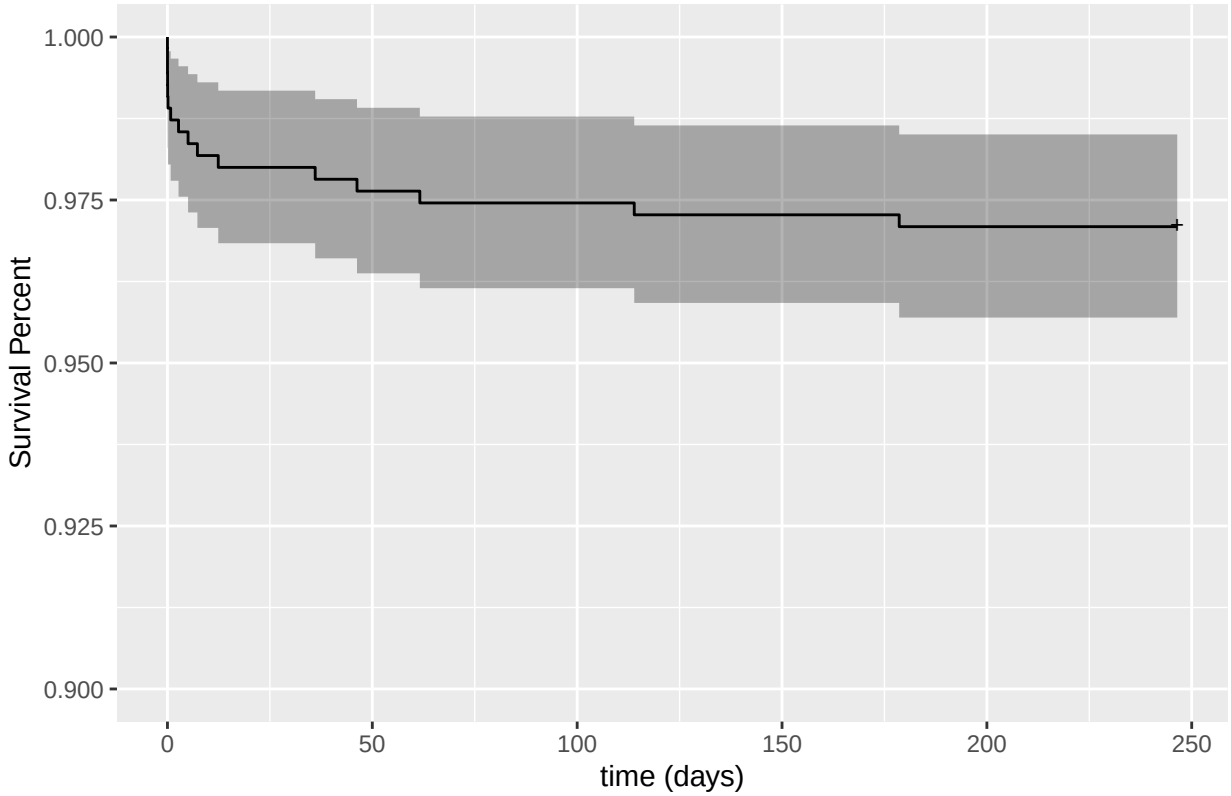


Fig 5 : Cluster 1 : First Borrow to Repay Survival Analysis

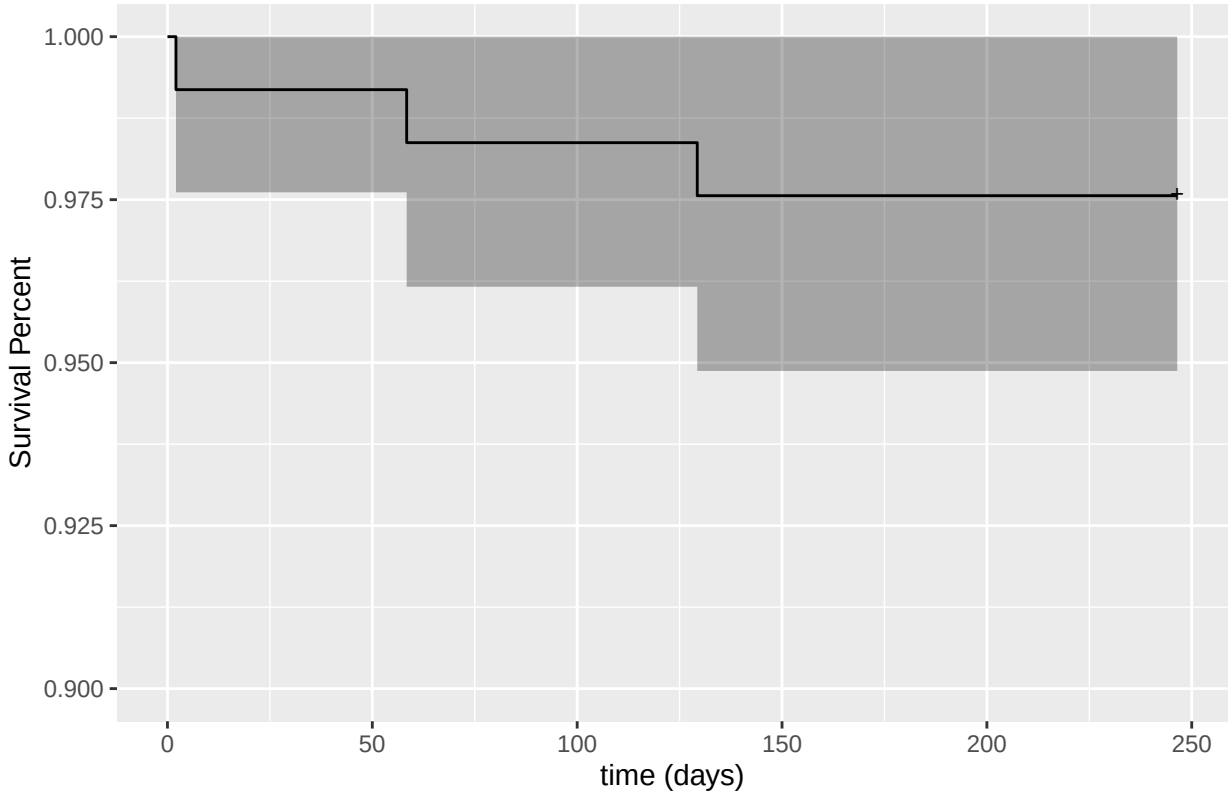


Fig 6 : Cluster 2 : First Borrow to Repay Survival Analysis

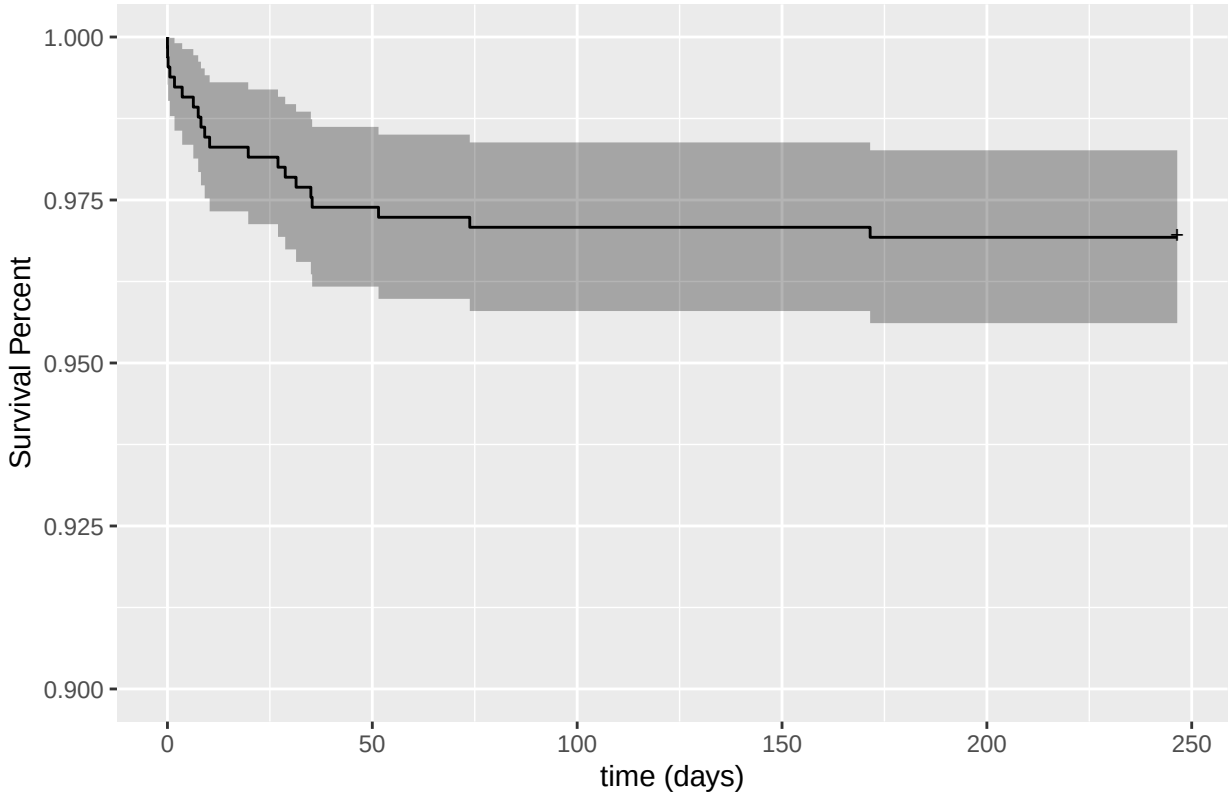


Fig 7 : Cluster 3 : First Borrow to Repay Survival Analysis

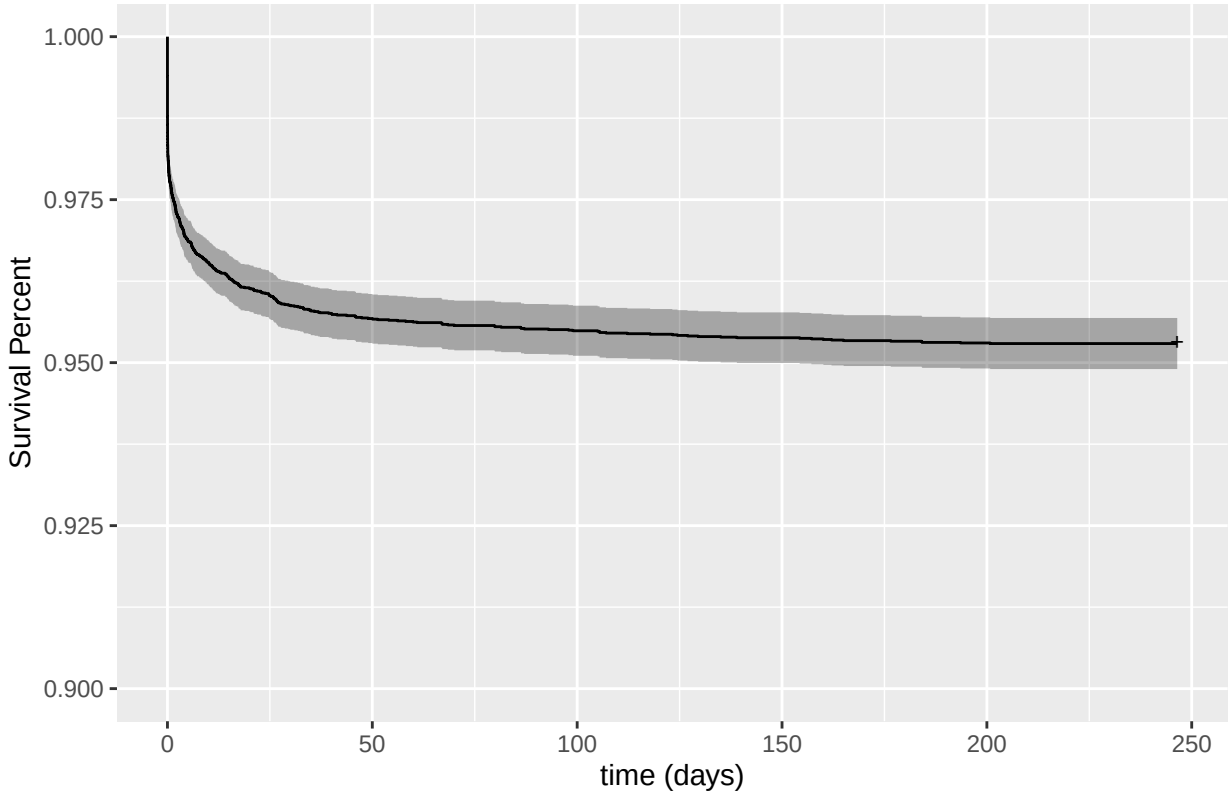
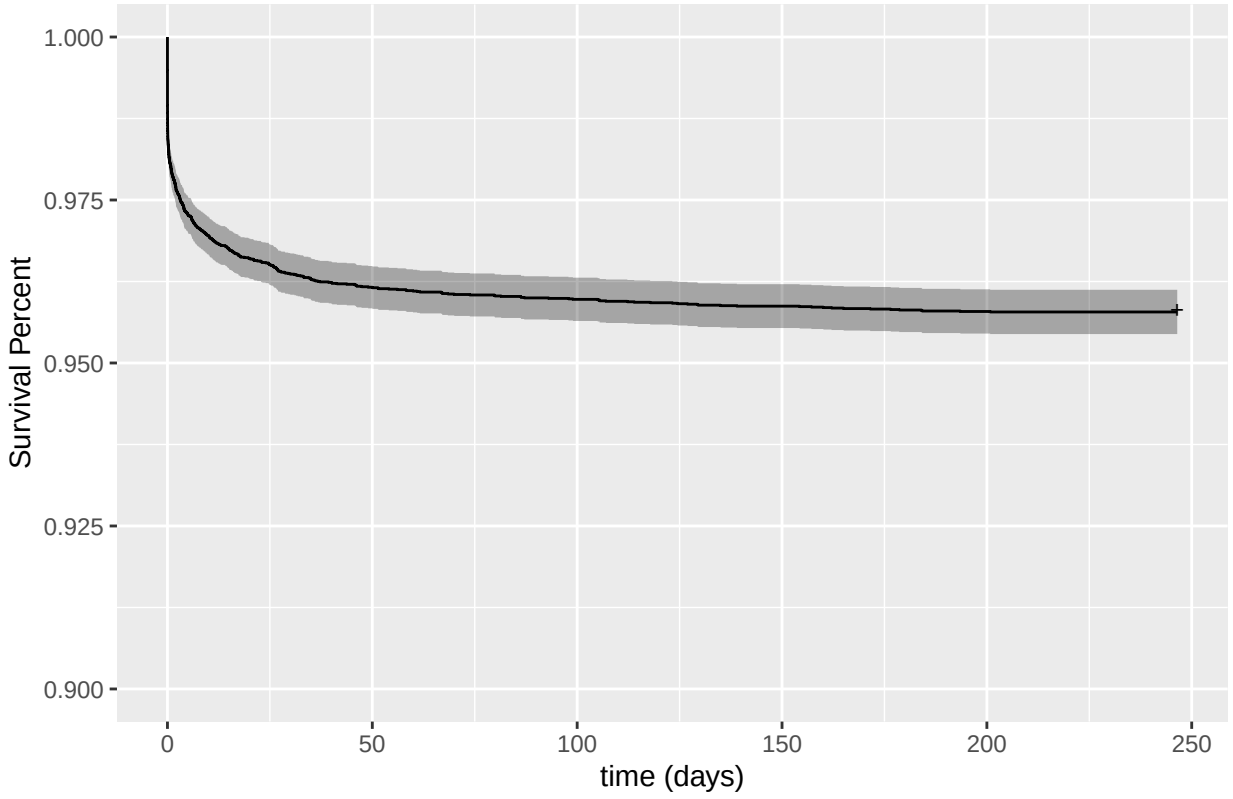


Fig 8 :Entire Dataset: First Borrow to Repay Survival Analysis



Discussion

The results of the survival plots fall flat, effectively tellng us that our clusterings are not very ideal in this case - it is is likely there are size differences among our clusters (the appendix also includes some of the comparisons of mean amount rather than densities which tells us about how the cluster sizes are skewed). Raising the the minimum number of points along with clustering on a larger dimension space may improve results.

One of the main limitations of the survival plots here is once again similar to the clustering problem - visual inspection and comparison may lead to dangerous or misleading conclusions. There does exist a statistically significant variant that could be used to make more definitive claims. Of course, this would be contingent on improving/enforcing a certain cluster size to guarantee enough samples per cluster/survival plot.

Furthermore, looking into other type of survival/death (i.e. rather than doing borrows to repays, consider deposits), may provide some insight on a set of good clusters. The hierarching theme here is our clusters actually did not perform optimally, as we have some clusters that are too small to have meaningful weight behind them.

Summary and Recommendations

DeFi data is very difficult to work with regarding clustering, and doing hyperparameter search to find optimal, intuitive clusterings can easily be done, and would definitely improve upon the results. Also, developing better features is always a difficult task that could be improved on. Formulating a way to incorporate all transaction types weekly averages rather than just using borrows could result in different clusters as well, since for our analysis we only looked at the borrowers data. As a start, one could look into the other transaction datasets (i.e. df.weekly[[5]], for example, tracks unique users with at least one liquidation).

Regarding these specific results, I'd advocate against using the hyperparameters selected here, as our

clusterings were very poor this iteration. There is a great deal of expandability that can be done not only in the user data space, but clustering coins as well, or looking at the types of coins a cluster of users may transact with.

For survival analysis, only a shallow depth was explored. Looking into the statistical significance of the survival plots would be useful to make mathematical claims rather than visual ones.

References

The following are references to official libraries/docs (which links the original papers) of the main algorithms used.

- https://umap-learn.readthedocs.io/en/latest/
- https://cran.r-project.org/web/packages/umap/vignettes/umap.html
- https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html
- https://cran.r-project.org/web/packages/dbscan/vignettes/hdbscan.html
- https://cran.r-project.org/web/packages/survminer/index.html

Appendix

Here, we provide the results to the density plots for an alternate UMAP + HDBSCAN clustering run. UMAP is an iterative problem and may change depending on the random seed. Recall that HDBSCAN then learns the best number of clusters. It is not a hyperparameter that you need to tune. However, depending on the MinPts hyperparameter (for HDBSCAN), you may get different cluster sizes, and cluster numbers.

The clustering was able to group two large liquidation groups of users, Cluster 4 and 5, and more stable borrows such as clusters 1 and 2 (though it looks like our clustering located time as a pattern, as cluster 2 transactions happen in late May).

Here, we plot a baseline stacked barplot to understand weekly transactions within the general dataset.

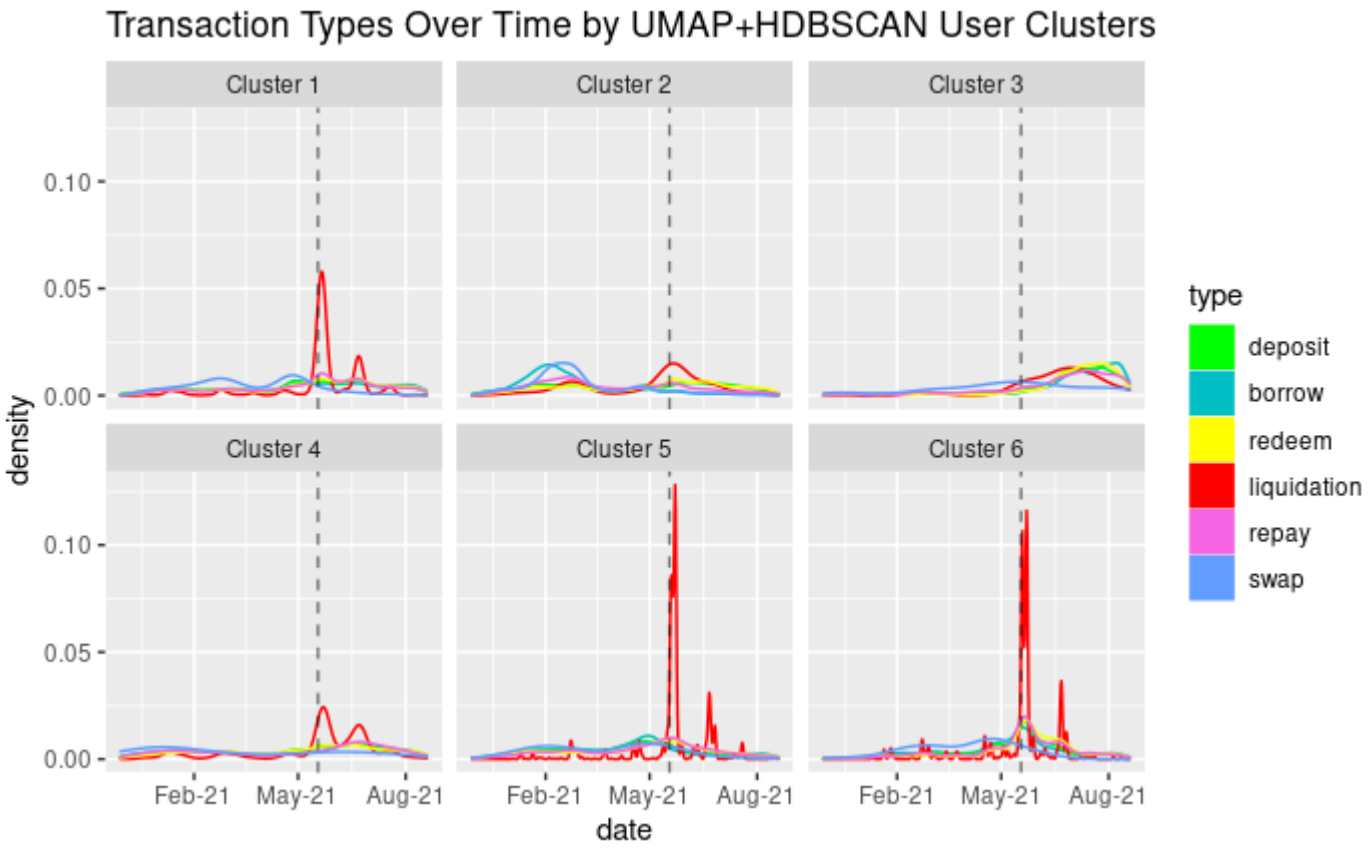
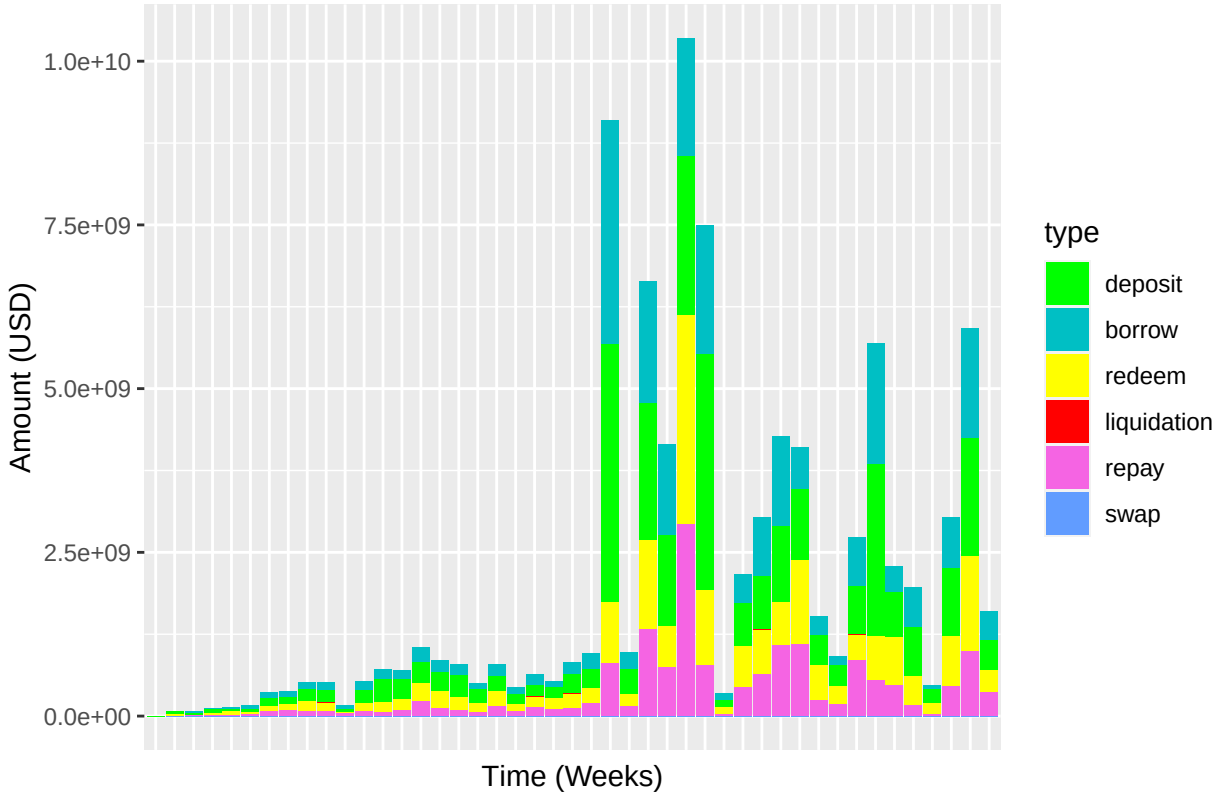


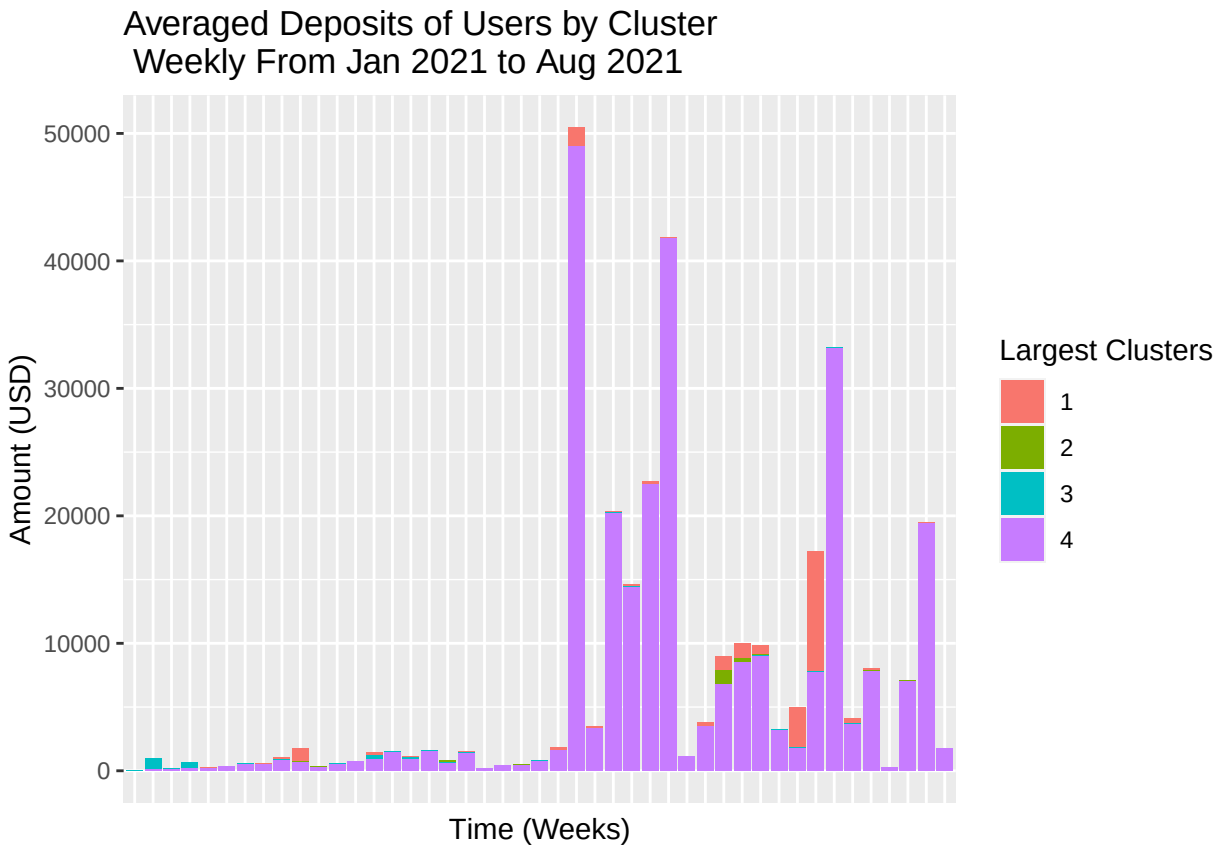
Figure 1: Alternative Transaction Types Over Time by UMAP+HDBSCAN User Clusters

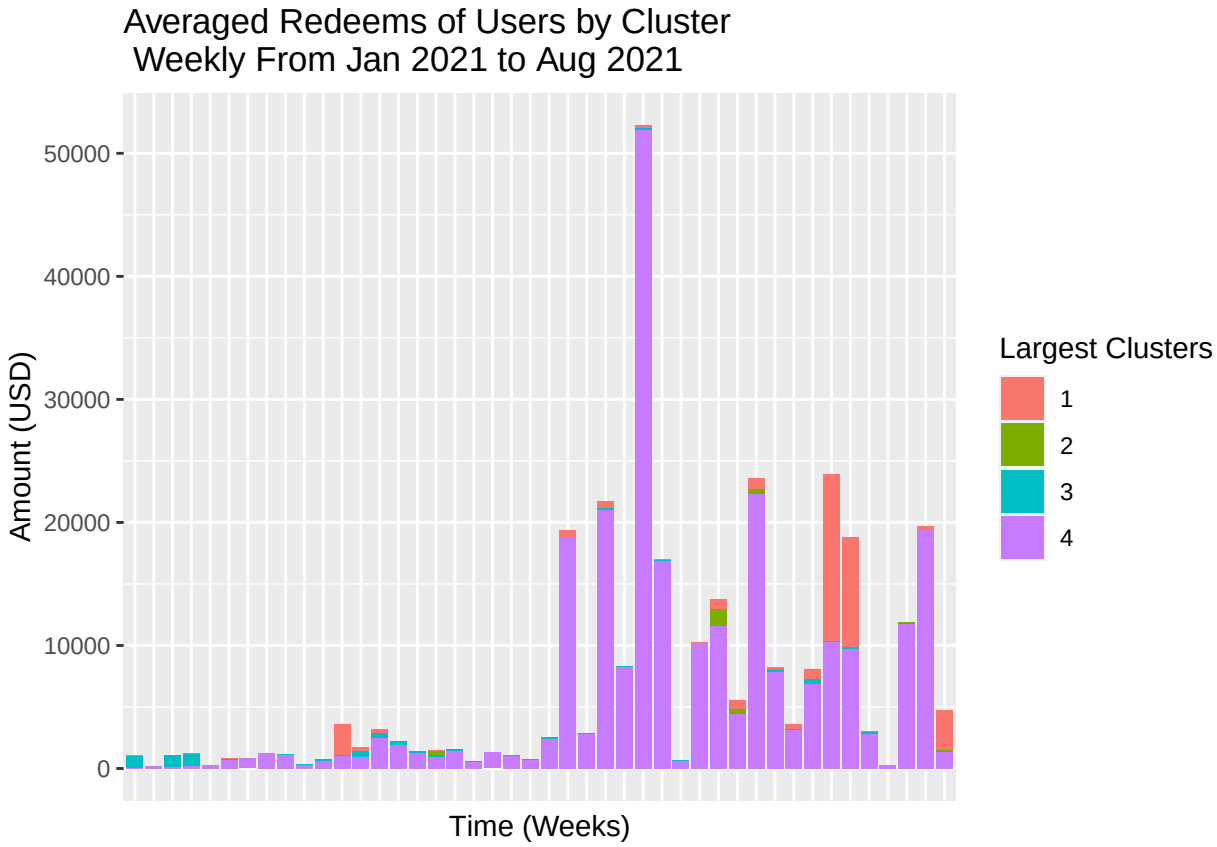
Fig 3: ALL Users Transactions Weekly From Jan 2021 to Aug 2021

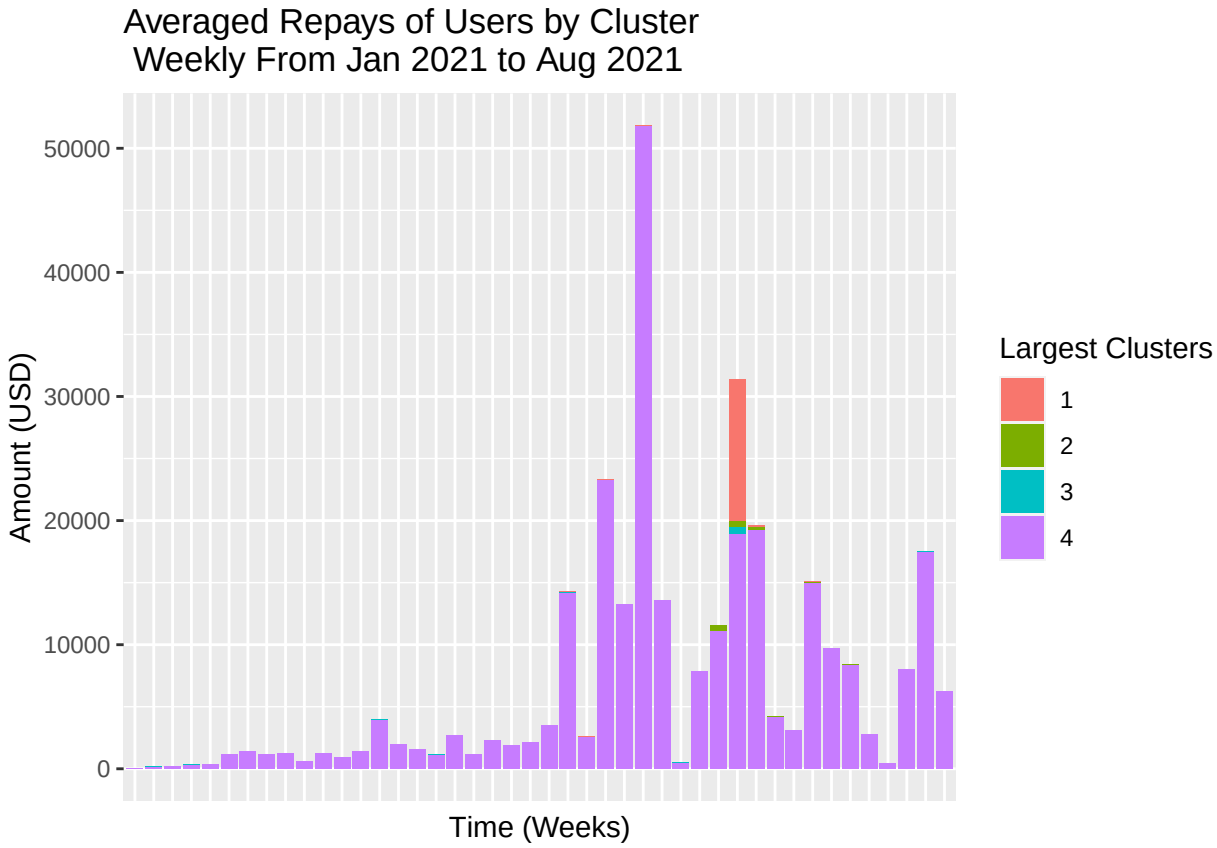


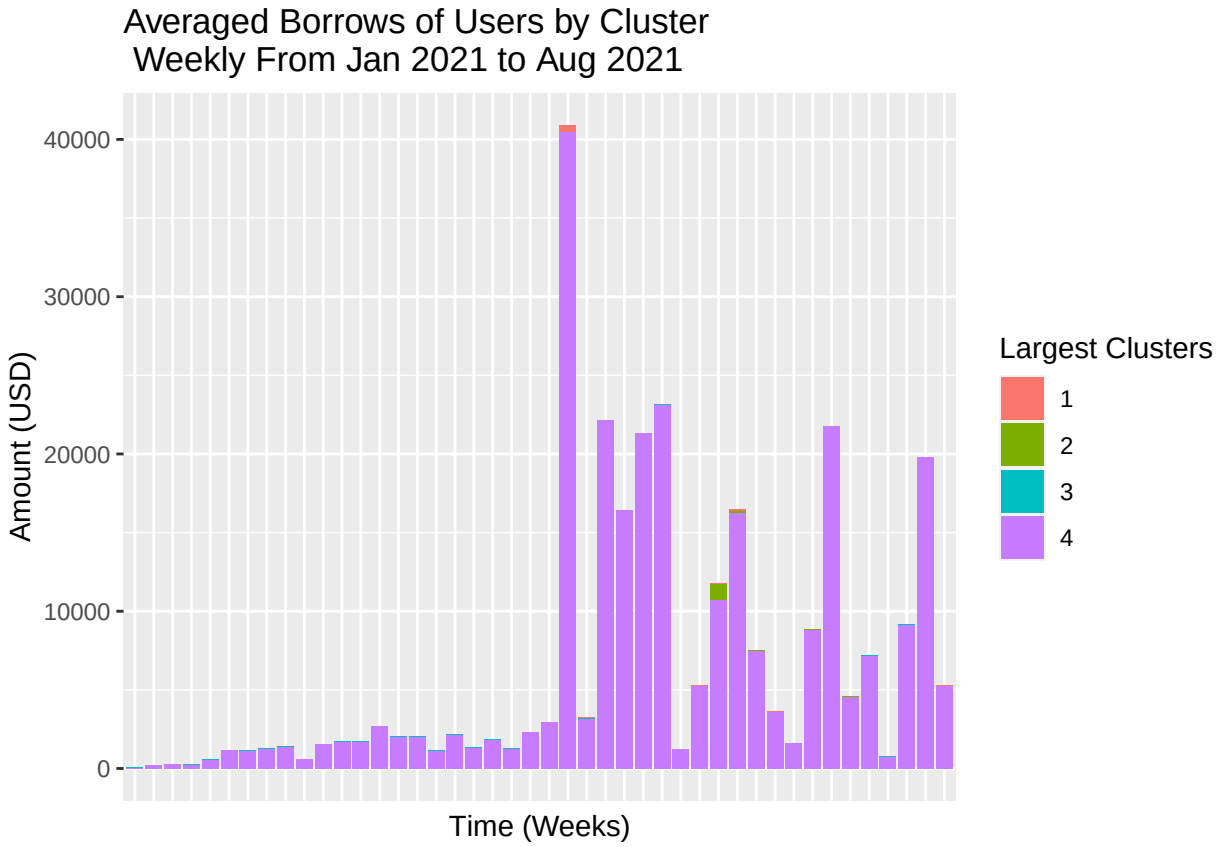
Here, we plot the averaged (by number of unique users per cluster) of amount USD deposits, redeems, repays, and borrows weekly. The stacked values indicate the amount for that specific cluster. Remark: Recall the weekly features were generated based on borrows, so the clustering will likely find distinct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

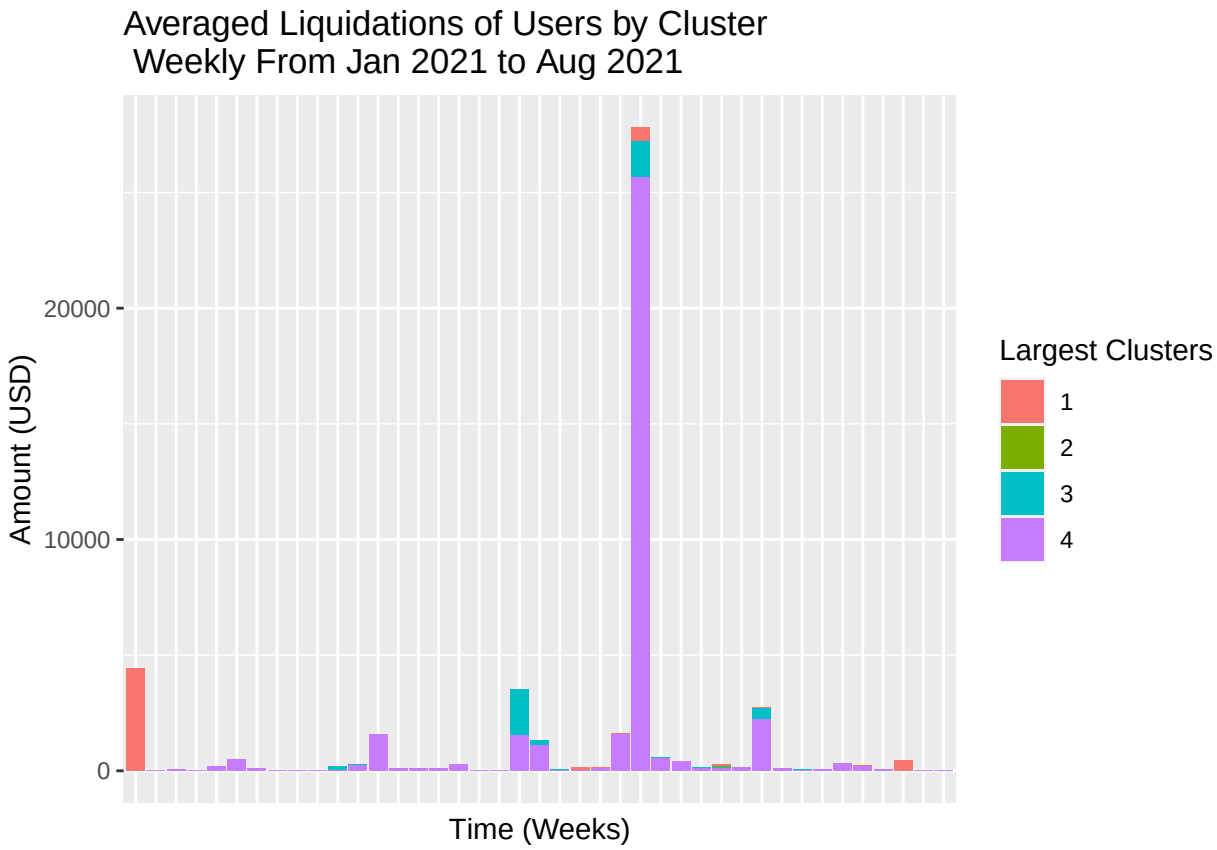
As we mentioned earlier, we see that there is a great imbalance in terms of value of transactions per cluster.

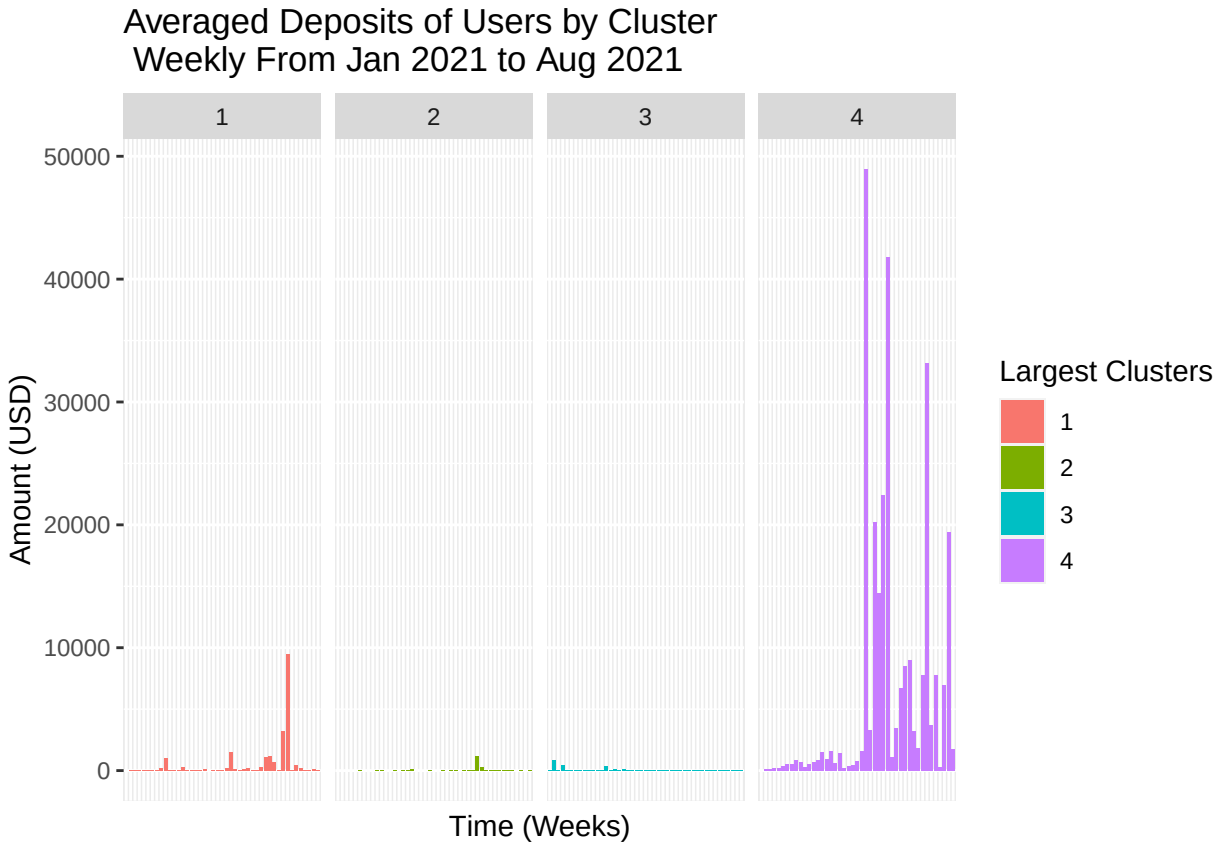

Averaged Deposits of Users by Cluster
Weekly From Jan 2021 to Aug 2021
Amount (USD)
50000
40000
30000
20000
10000
0
Time (Weeks)
Largest Clusters
1
2
3
4

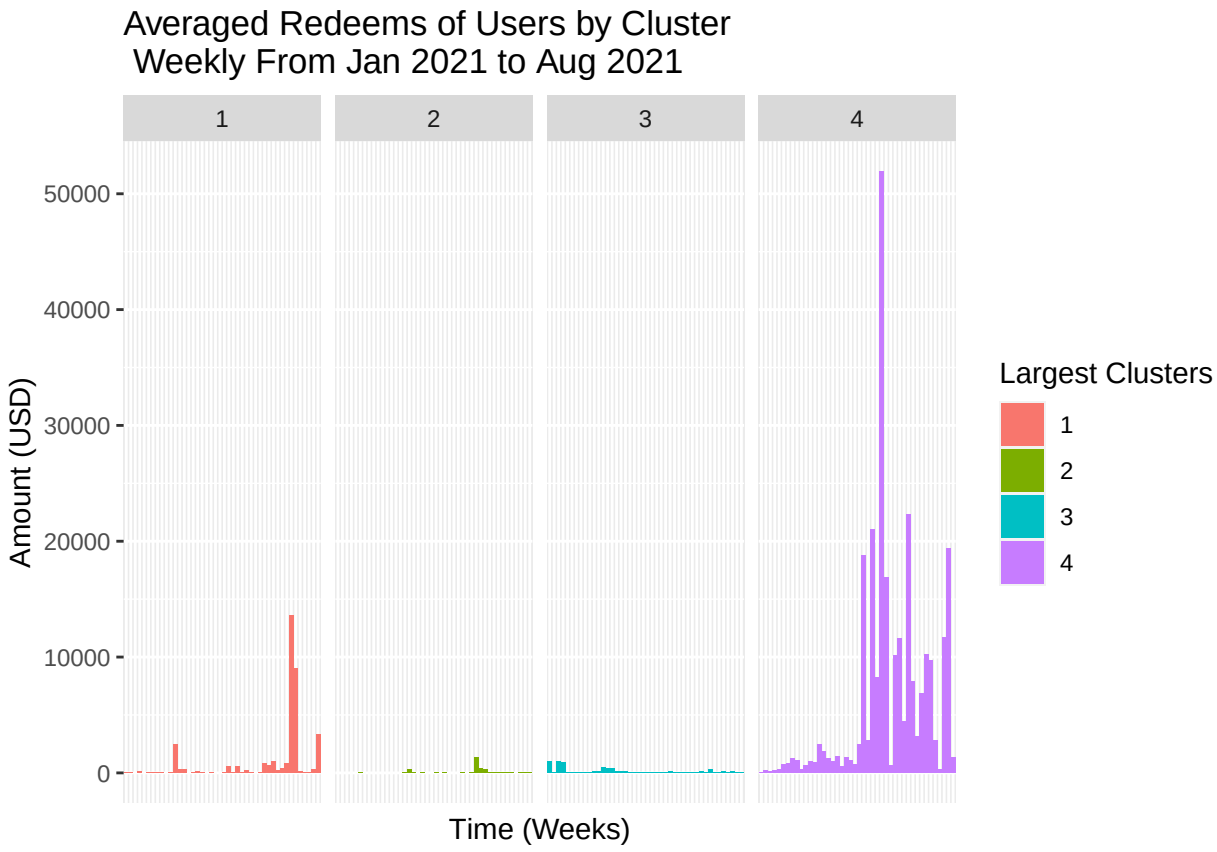

Averaged Redeems of Users by Cluster
Weekly From Jan 2021 to Aug 2021
Amount (USD)
50000
40000
30000
20000
10000
0
Time (Weeks)
Largest Clusters
1
2
3
4

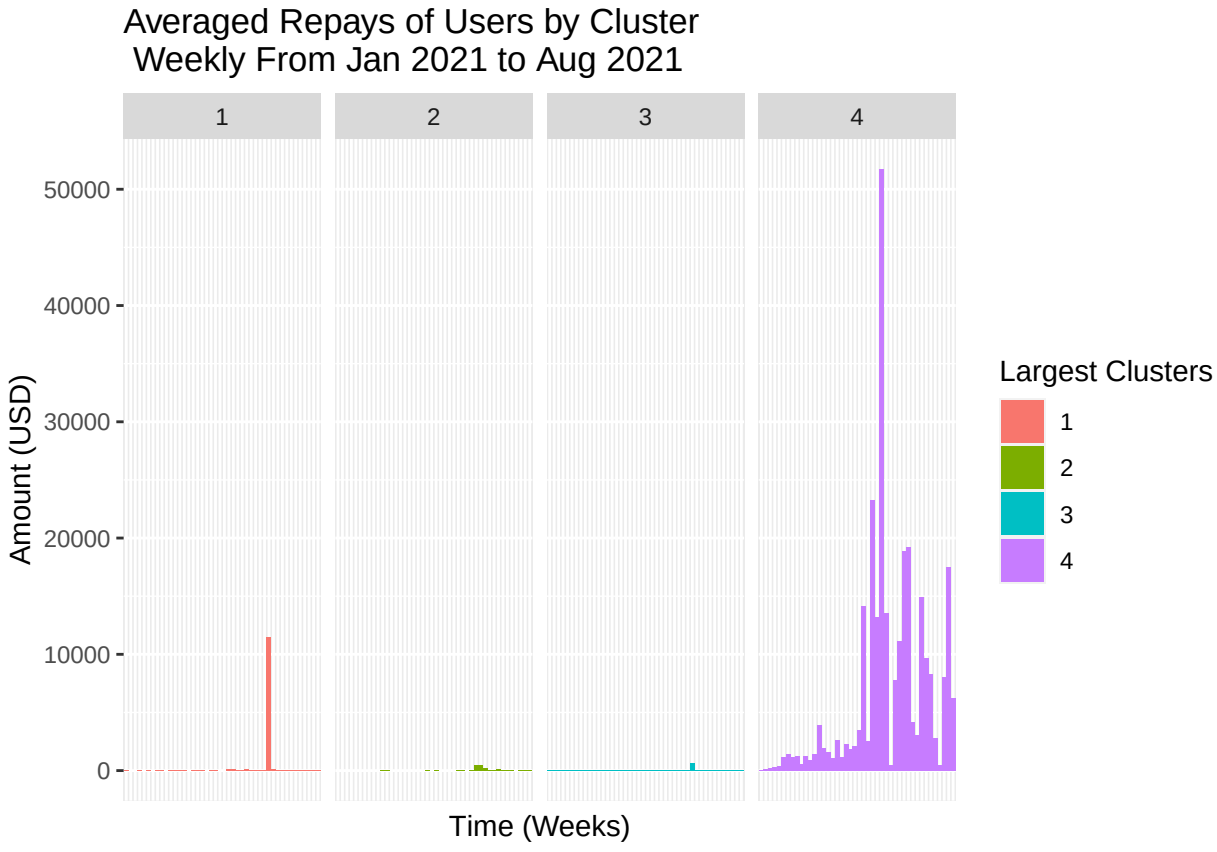

Averaged Repays of Users by Cluster
Weekly From Jan 2021 to Aug 2021
Amount (USD)
50000
40000
30000
20000
10000
0
Time (Weeks)
Largest Clusters
1
2
3
4

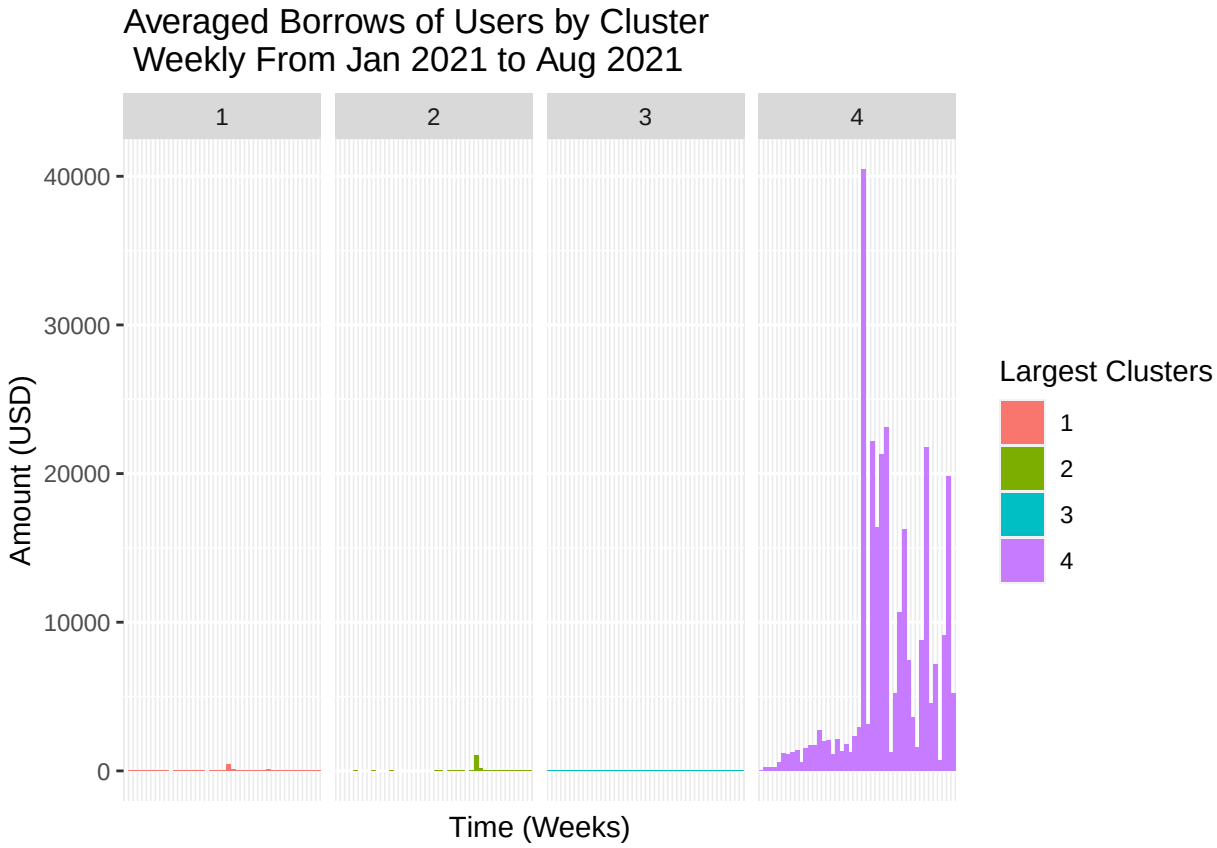

Averaged Borrows of Users by Cluster
Weekly From Jan 2021 to Aug 2021
Amount (USD)
40000
30000
20000
10000
0
Time (Weeks)
Largest Clusters
1
2
3
4

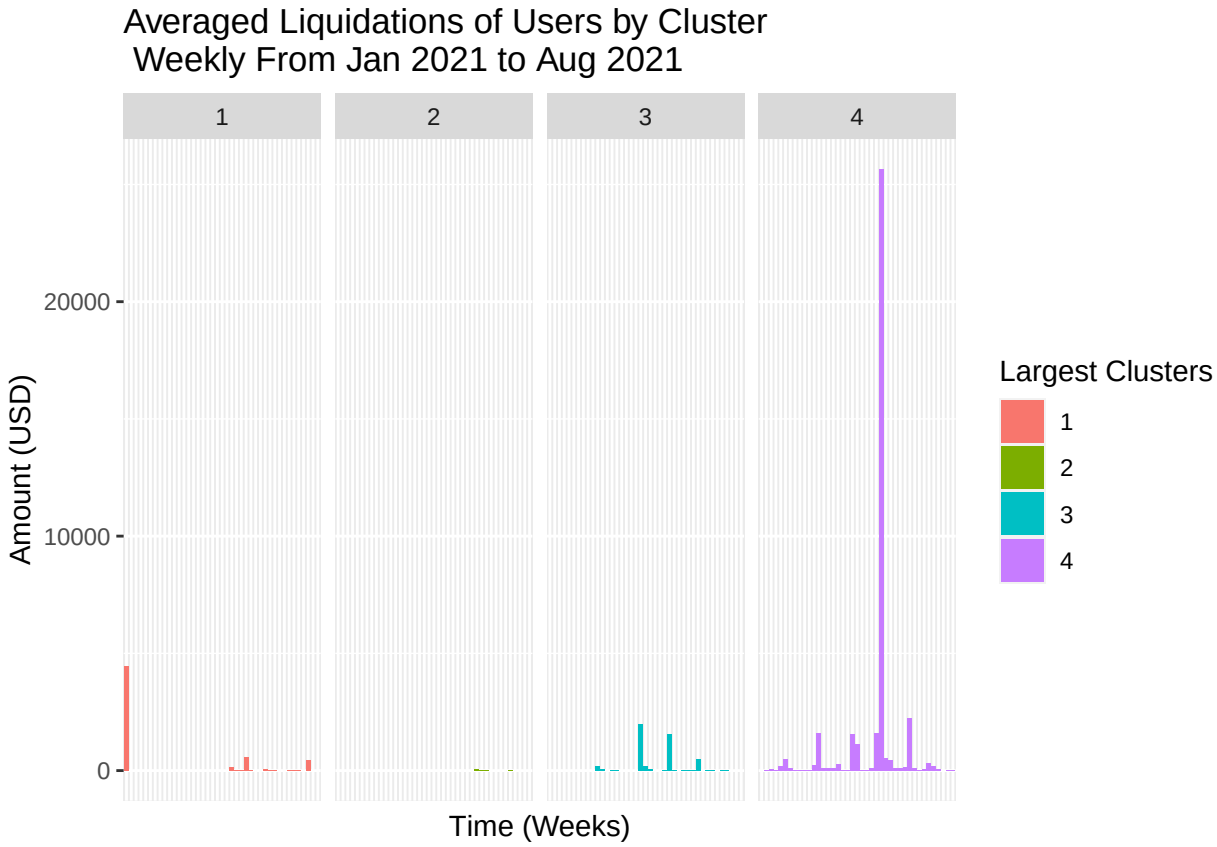

Averaged Liquidations of Users by Cluster
Weekly From Jan 2021 to Aug 2021
Amount (USD)
20000
10000
0
Time (Weeks)
Largest Clusters
1
2
3
4


Averaged Deposits of Users by Cluster
Weekly From Jan 2021 to Aug 2021
1
2
3
4
Amount (USD)
50000
40000
30000
20000
10000
0
Time (Weeks)
Largest Clusters
1
2
3
4


Averaged Redeems of Users by Cluster
Weekly From Jan 2021 to Aug 2021
1
2
3
4
50000
40000
30000
20000
10000
0
Amount (USD)
Time (Weeks)
Largest Clusters
1
2
3
4


Averaged Repays of Users by Cluster
Weekly From Jan 2021 to Aug 2021
1
2
3
4
50000
40000
30000
20000
10000
0
Amount (USD)
Time (Weeks)
Largest Clusters
1
2
3
4


Averaged Borrows of Users by Cluster
Weekly From Jan 2021 to Aug 2021
1
2
3
4
40000
30000
20000
10000
0
Amount (USD)
Time (Weeks)
Largest Clusters
1
2
3
4


Averaged Liquidations of Users by Cluster
Weekly From Jan 2021 to Aug 2021
1
2
3
4
20000
10000
0
Amount (USD)
Time (Weeks)
Largest Clusters
1
2
3
4