DAR F21 Project Status Notebook

DeFi

Cole Paquin

11/18/2021

Contents

Weekly Work Summary

NOTE: Follow an outline format; use bullets to express individual points.

- RCS ID: Paquic
- Project Name: DeFi
- Summary of work since last week- I have been working on visualizations to distinguish how coins are used over time.
  - Describe the important aspects of what you worked on and accomplished As we will see below, I have created charts to show the different transactions types and their amounts by coin. I've also attempted to model the traditional utilization rate, which has shown some interesting results.
- Summary of github issues added and worked
  - Issue #86 (Closed) - Added my clusters to the app folder to be used for survival analysis
  - Issue #93/#95 - Many of the nansen charts have been made (will close soon)
- Summary of github commits
  - branch name - dar-paquic
  - Github Commits:
    - paquic-Assignment6-f21.Rmd
    - paquic_Assignment6.Rmd
    - paquic_Assignment6.pdf

Personal Contribution

All code has been written by me.

Discussion of Primary Findings

- Discuss primary findings:

- What did you want to know? I wanted to know if different coins are used in different ways, and if there is a way that we can better visualize the differences.
- How did you go about finding it? I used the concept of utilization rate to be able to show that different types of coins have very different usages. This means that we look at the amount of money being borrowed relative to the total amount of liquidity.
- What did you find?

First we do some basic setup and create our main dataframe.

```
#import libraries
if (!require("ggplot2")) {
   install.packages("ggplot2")
   library(ggplot2)
}
if (!require("knitr")) {
   install.packages("knitr")
   library(knitr)
}
if (!require("reshape2")) {
   install.packages("reshape2")
   library(reshape2)
}
library(ggbiplot)
library(gplots)
library(RColorBrewer)
library(beeswarm)
library(tidyverse)
library(tidyquant)
library(ggbeeswarm)
library(foreach)
library(doParallel)
library(Rtsne)
library(anytime)
```

```
#load in csv file to data frame
df<-readRDS("~/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
## 5               NA                           NA             NA
## 6               NA                           NA             NA
##   principalReserve reservePriceETHPrincipal reservePriceUSDPrincipal
## 1                                       NA                       NA
## 2                                       NA                       NA
## 3                                       NA                       NA
## 4                                       NA                       NA
## 5                                       NA                       NA
## 6                                       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
```

```
df.weekly<- df%>%group_by(user, user_alias)%>%
  dplyr::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  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)
}
```

```
## 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.
```

```
df.weekly <- rename(df.weekly, "name" = "user_alias")

head(df.weekly)
```

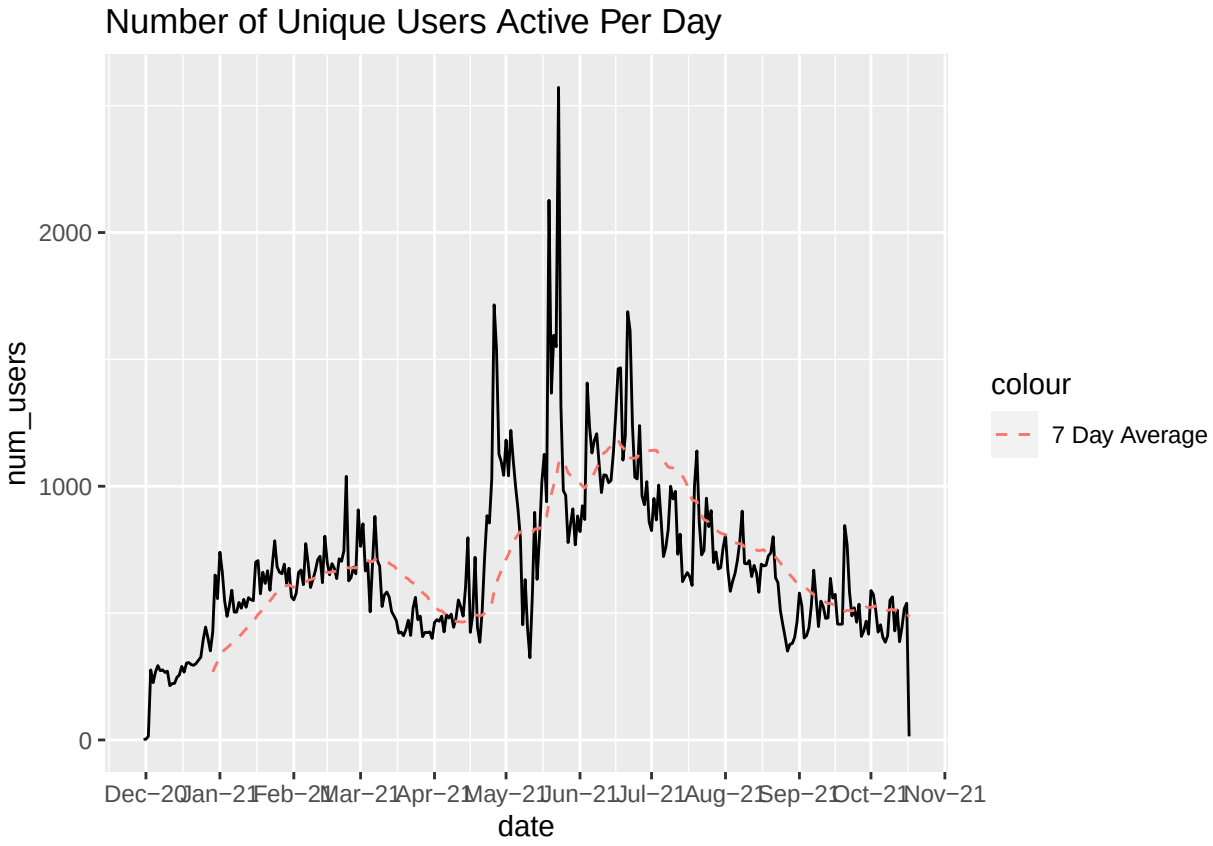
```
##           user             name  timefirst   timelast  N timeactive
## 1 1.000000e+00    Tracy Nisbett 2021-07-22 2021-07-22  1     0 days
## 2 5.700500e+04 Millicent Gandhi 2021-02-22 2021-07-12  2   140 days
## 3 2.577533e+33  Steven Ferguson 2021-02-13 2021-10-09 60   238 days
## 4 6.663597e+34   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")
```

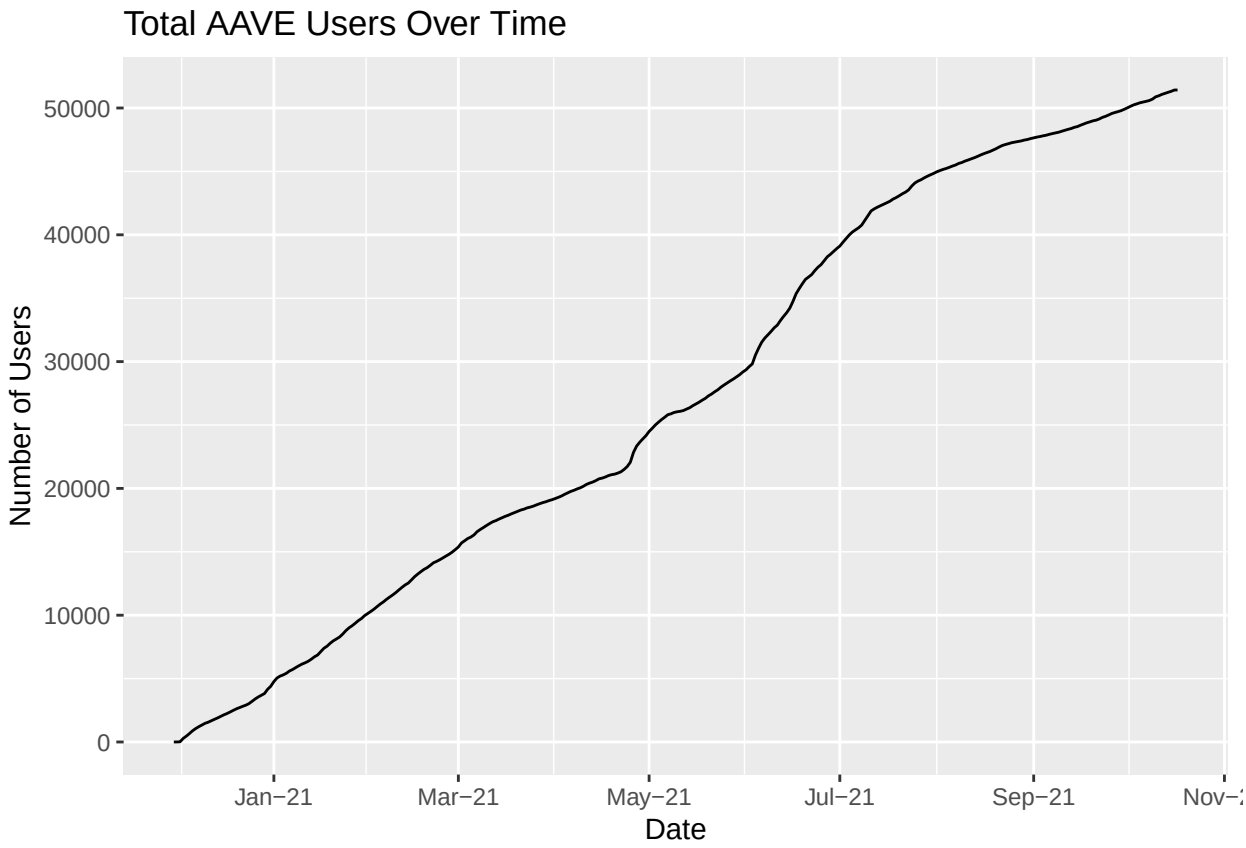
So I knew that we wanted to produce some of nansen charts, both as a sanity check and to potentially add to

our app. I first decided to work on the overall user summaries. This gave me two plots, one for users per day and one for the cumulative number of users over time.

```
df["date"] = anydate(df$timestamp)
unique_users <- df %>%
  group_by(date)%>%
  summarise(num_users = n_distinct(user))
ggplot(unique_users) + geom_line(aes(x = date, y = num_users))+
  ggtitle("Number of Unique Users Active Per Day")+
  scale_x_date(date_breaks = "1 month", date_labels = "%b-%y")+
  geom_ma(n = 30, aes(x=date, y = num_users, colour = "7 Day Average"))
```




```
cum_users <- df.weekly %>%
  select(timefirst, user)
cum_users <- cum_users %>%
  count(timefirst)
cum_users <- cum_users %>%
  mutate(cum = cumsum(n))
ggplot(cum_users) + geom_line(aes(x = timefirst, y = cum))+
  ggtitle("Total AAVE Users Over Time")+
  scale_x_date(date_breaks = "2 months", date_labels = "%b-%y")+
  labs(x = "Date", y = "Number of Users")
```

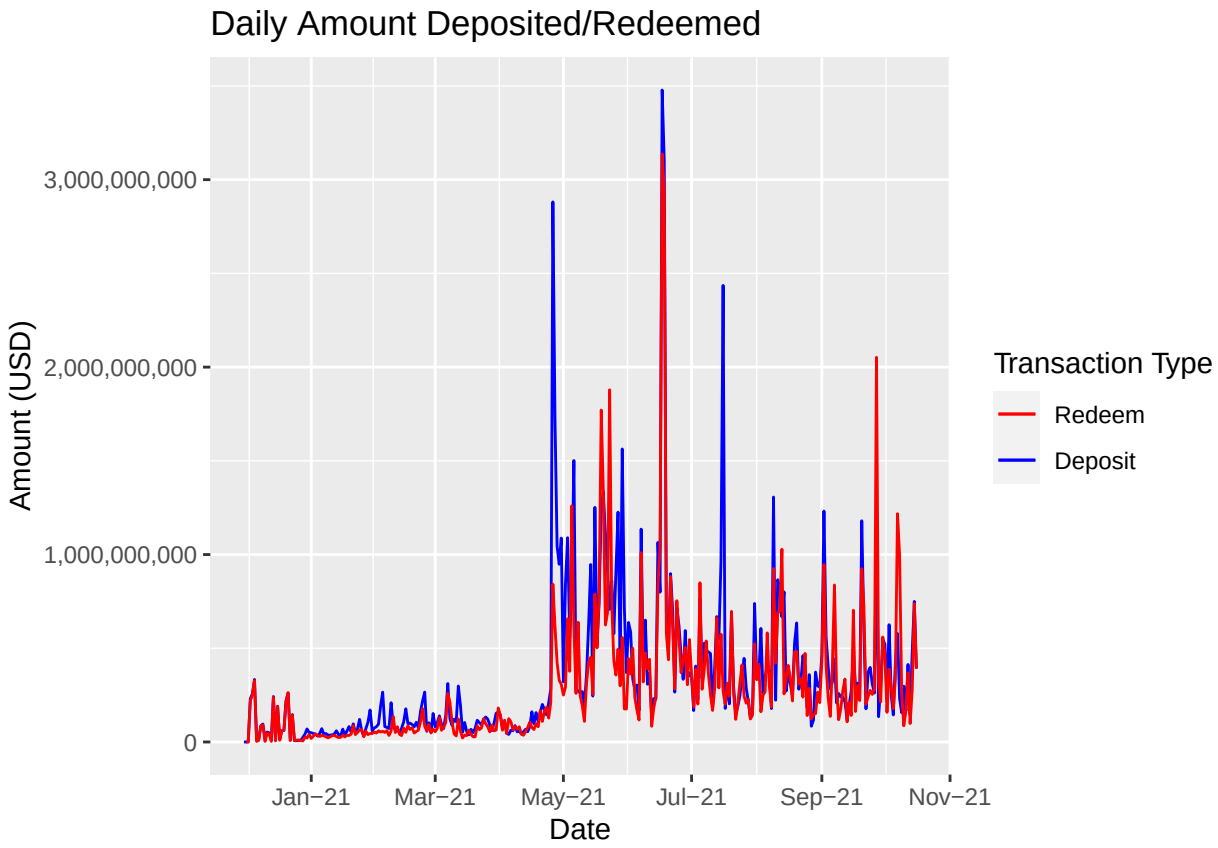


These graphs both do a good job of showing how AAVE has progressed over time. Although the number of active users per day does not appear to be increasing, we see a constant increase in new users. This means that AAVE is still doing a good job of attracting new customers, but does not see as much activity per user as they used to.

Transactions By Coin

Now, I moved into the more exciting work. Jason and I were looking to build a coin analysis page in our app, but we needed visualizations to do it. First, I wanted to look at the relationship between deposits and redeems over time. Before I look at individual coins, let's look at the overall picture.

```
all_deposits <- df %>%
  filter(type == "deposit") %>%
  group_by(date) %>%
  summarise(value = sum(amountUSD))
all_redeems <- df %>%
  filter(type == "redeem") %>%
  group_by(date) %>%
  summarise(value = sum(amountUSD))
ggplot() +
  geom_line(data = all_deposits, aes(x = date, y = value, color = "Deposit"))+
  geom_line(data = all_redeems, aes(x = date, y = value, color = "Redeem"))+
  scale_color_manual(name = "Transaction Type", values = c("Redeem" = "red", "Deposit" = "blue"))+
  scale_y_continuous(labels = comma)+
  scale_x_date(date_breaks = "2 months", date_labels = "%b-%y")+
  ggtitle("Daily Amount Deposited/Redeemed")+
  xlab("Date")+
  ylab("Amount (USD)")
```



We can see that they usually move together, although there are a few occasions where one of them jumps without the other. However, without seeing the number of transactions as well, we cannot tell if these are simply a few large transactions or many small ones. Next, I will show this same chart, but specified by individual coin.

```
top_10_graphs <- df %>%
  add_count(reserve) %>%
  filter(dense_rank(-n) <= 10 & (type == "deposit" | type == "redeem"))
deposits <- top_10_graphs %>%
  filter(type == "deposit") %>%
  group_by(date, reserve) %>%
  summarise(value = sum(amountUSD), reserve = reserve)
```

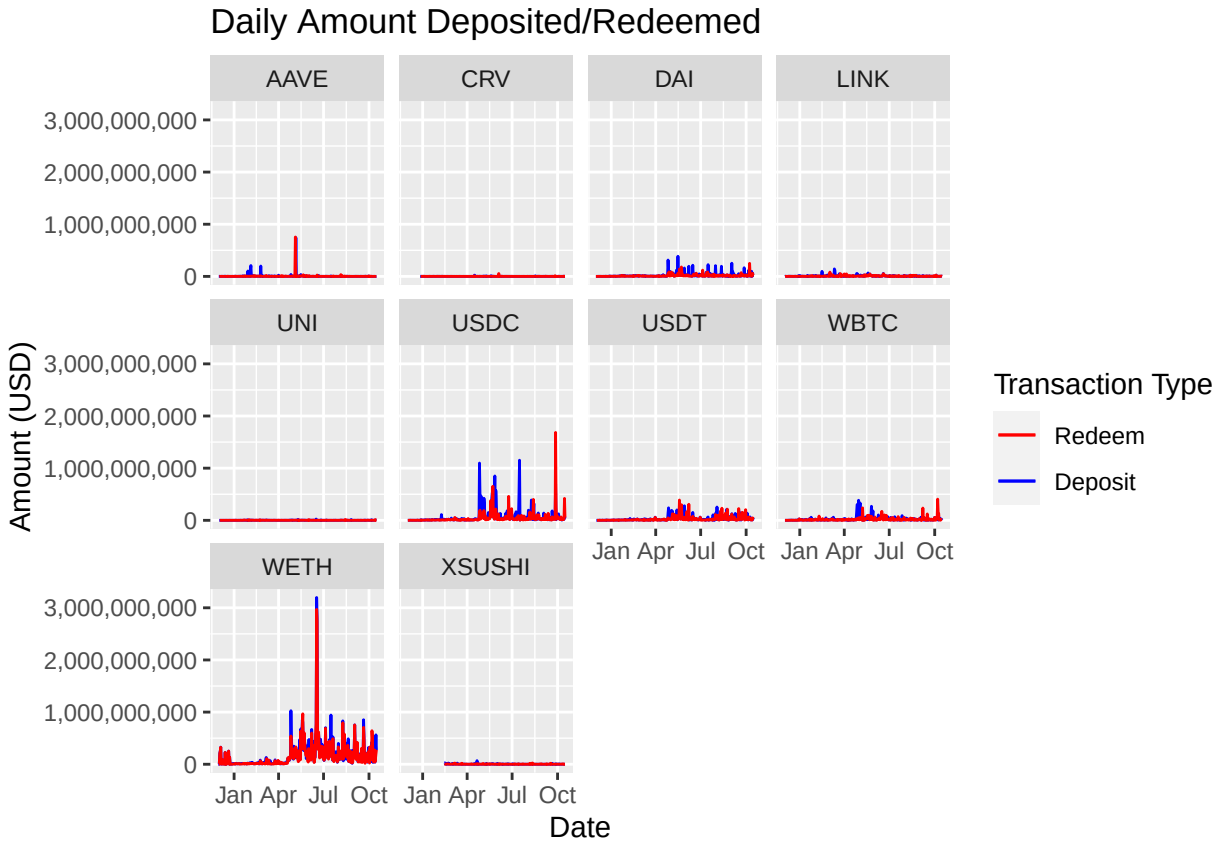
```
## `summarise()` has grouped output by 'date', 'reserve'. You can override using the `.groups` argument
```

```
redeems <- top_10_graphs %>%
  filter(type == "redeem") %>%
  group_by(date, reserve) %>%
  summarise(value = sum(amountUSD), reserve = reserve)
```

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

```
deposits <- distinct(deposits, date, reserve, .keep_all= TRUE)
redeems <- distinct(redeems, date, reserve, .keep_all= TRUE)
ggplot() +
  geom_line(data = deposits, aes(x = date, y = value, color = "Deposit"))+
  geom_line(data = redeems, aes(x = date, y = value, color = "Redeem"))+
  scale_color_manual(name = "Transaction Type", values = c("Redeem" = "red", "Deposit" = "blue"))+
  scale_y_continuous(labels = comma)+
```

```
  facet_wrap(~ reserve)+
  ggtitle("Daily Amount Deposited/Redeemed")+
  xlab("Date")+
  ylab("Amount (USD)")
```



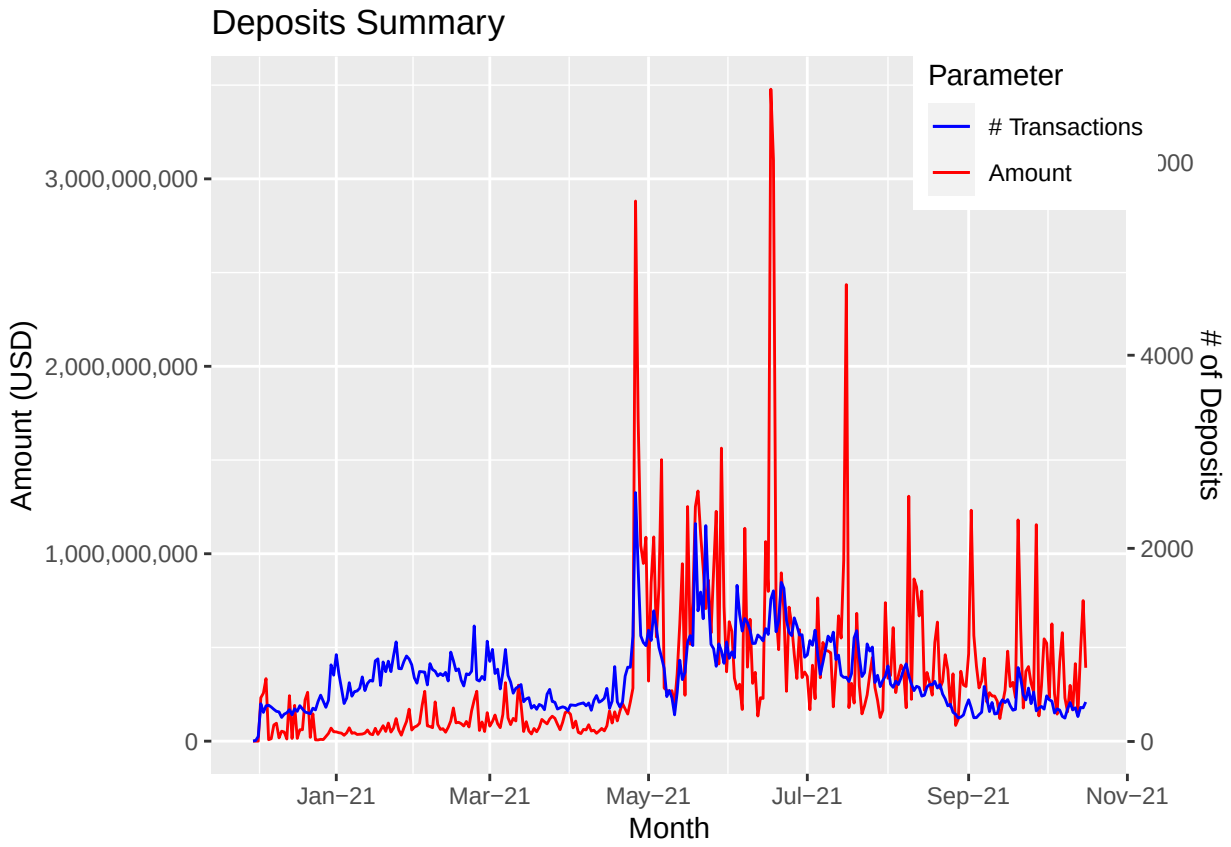
This shows us the same chart, but for the top 10 most used coins in the dataset. This gives us a better scale of what makes up the total transactions and shows how certain coins (WETH & USDC) make up most of our last chart. We can do the same thing with other transactions, but that is not needed to be shown here. Next, I'll move on to what I mentioned above about showing both transaction amounts and counts on the same chart to get a better understanding of user activity.

Dual-Axis Charts

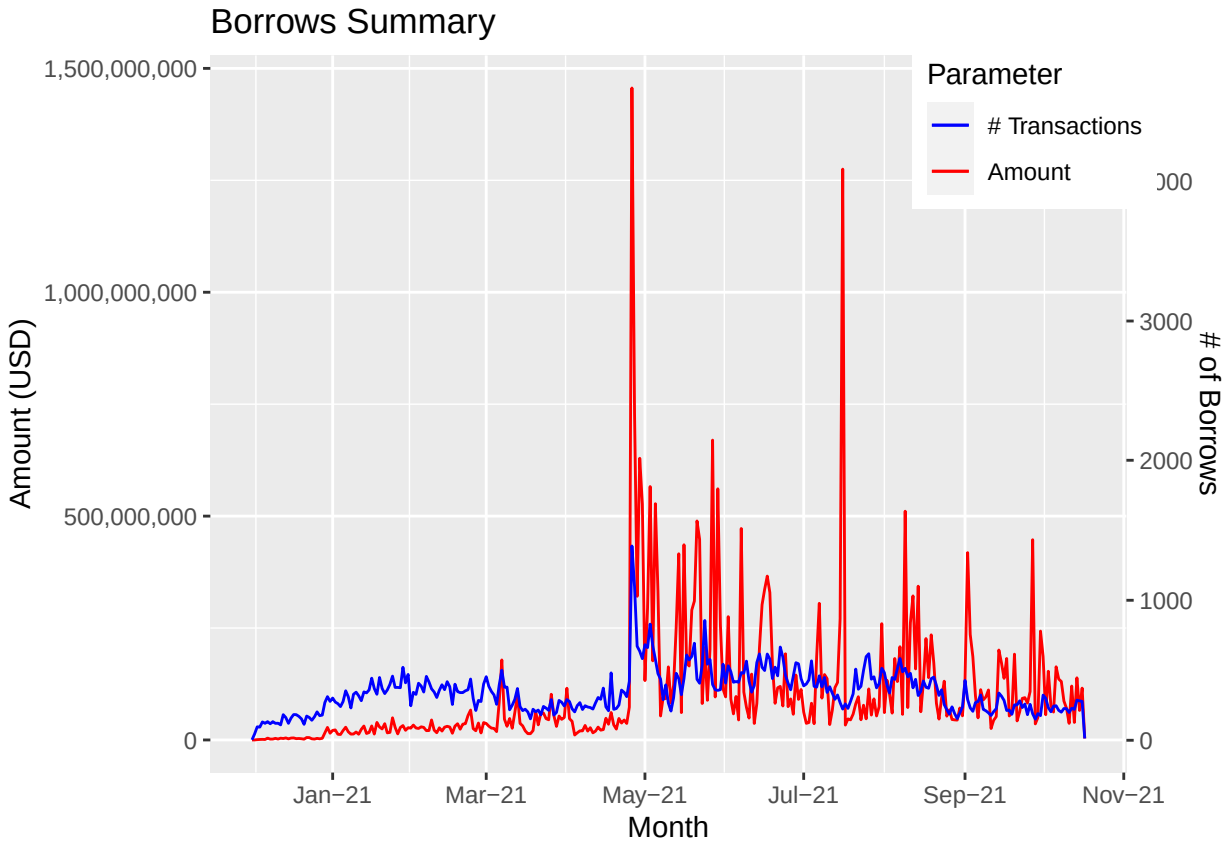
I'll only show it for two transaction types, but this code can be easily applied to all of them. Let's first look at deposits, and then borrows.

```
all_deposits <- df %>%
  filter(type == "deposit") %>%
  group_by(date) %>%
  summarise(value = sum(amountUSD), count = n())
all_borrows <- df %>%
  filter(type == "borrow") %>%
  group_by(date) %>%
  summarise(value = sum(amountUSD), count = n())
all_redeems <- df %>%
  filter(type == "redeem") %>%
  group_by(date) %>%
  summarise(value = sum(amountUSD), count = n())
```

```r
x <- mean(all_deposits$value) / mean(all_deposits$count)

ggplot(all_deposits, aes(x = date))+
  geom_line(aes(y = value, colour = "Amount"))+
  geom_line(aes(y = count*x, colour = "# Transactions"))+
# Sec.Axis adds the second axis, the rest is all formatting
  scale_y_continuous(labels = comma, sec.axis = sec_axis(~./x, name = "# of Deposits"))+
  scale_colour_manual(values = c("blue", "red"))+
  labs(y = "Amount (USD)",x = "Month",colour = "Parameter")+
  scale_x_date(date_breaks = "2 months", date_labels = "%b-%y")+
  theme(legend.position = c(0.9, 0.9))+
  ggtitle("Deposits Summary")
```




```r
x <- mean(all_borrows$value) / mean(all_borrows$count)

ggplot(all_borrows, aes(x = date))+
  geom_line(aes(y = value, colour = "Amount"))+
  geom_line(aes(y = count*x, colour = "# Transactions"))+
  scale_y_continuous(labels = comma, sec.axis = sec_axis(~./x, name = "# of Borrows"))+
  scale_colour_manual(values = c("blue", "red"))+
  labs(y = "Amount (USD)",x = "Month",colour = "Parameter")+
  scale_x_date(date_breaks = "2 months", date_labels = "%b-%y")+
  theme(legend.position = c(0.9, 0.9))+
  ggtitle("Borrows Summary")
```



These both seem to show the same trend. Transactions counts have stayed fairly steady over time, whereas the amounts have swung much wider. It appears the the average transaction amount has increased over time (we could also plot that). However, the biggest part of this is that is shows the benefit of having more than one axis, so that we can better compare data of different scales.

Coin Utilization

That concept will be used when showing our estimated utilization rate. Utilization Rate is very important because it determines the borrow rate for each coin. Every coin has an optimal utilization that can be very different. These optimal rates can be found on https://docs.aave.com/risk/liquidity-risk/borrow-interest-rate. First let's look at how we can try to model and graph utilization for one coin, in this case USDT.

```
coins <- "USDT"
dates <- as.data.frame(unique(df$date))
colnames(dates) <- c("date")
# We want all different transaction types by coin, liquidation is tricky
coin <- df %>%
  filter(reserve == coins)
dep<- coin %>%
  filter(type == "deposit") %>%
  group_by(date) %>%
  summarise(date = date, deposit_amount = cumsum(sum(amount)))%>%
  distinct()
```

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

```
red<-coin %>%
  filter(type == "redeem") %>%
  group_by(date) %>%
```

```
  summarise(date= date, redeem_amount = cumsum(sum(amount)))%>%
  distinct()
```

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

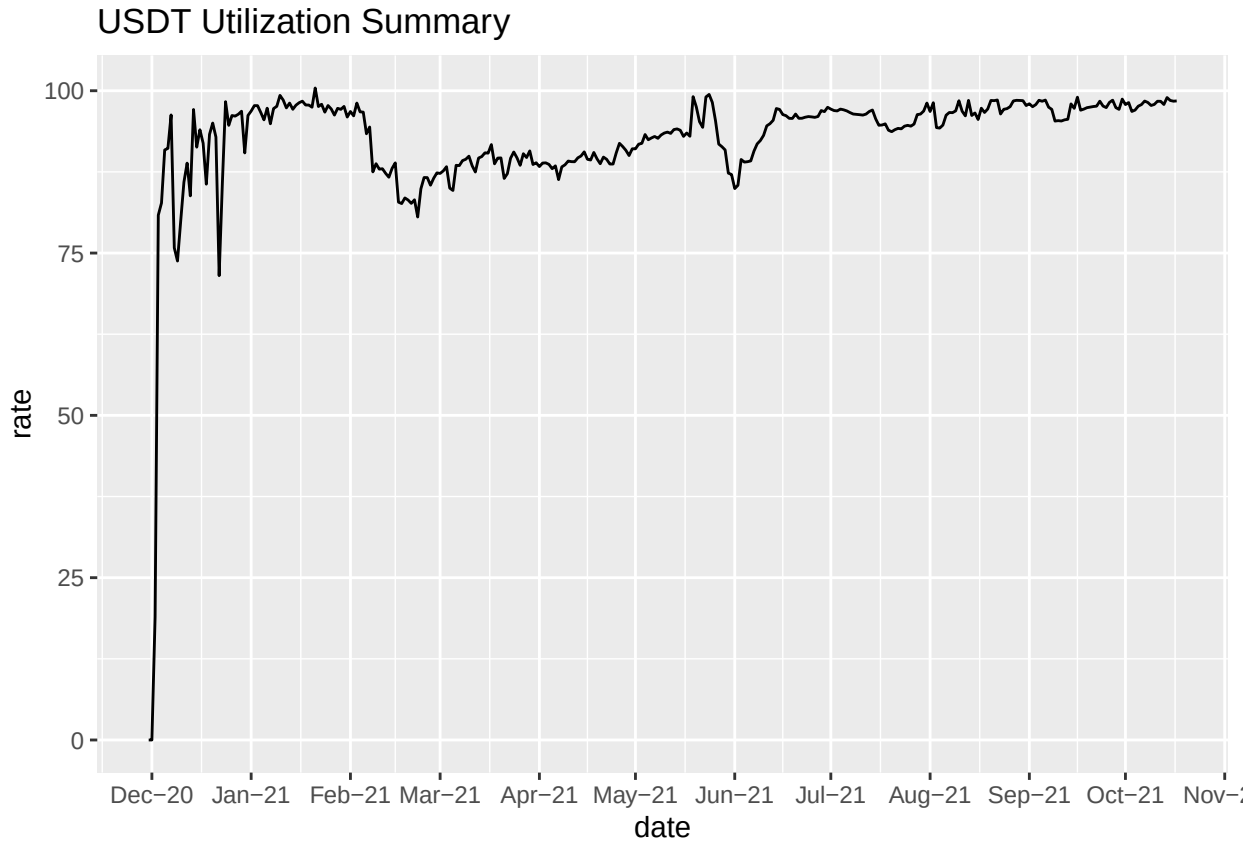
```
rep <- coin %>%
  filter(type == "repay") %>%
  group_by(date) %>%
  summarise(date = date, repay_amount = cumsum(sum(amount)))%>%
  distinct()
```

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

```
bor <- coin %>%
  filter(type == "borrow") %>%
  group_by(date) %>%
  summarise(date = date, borrow_amount = cumsum(sum(amount)))%>%
  distinct()
```

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

```
util_summary <- dates %>% left_join(dep, by = "date")
util_summary <- util_summary %>% left_join(red, by = "date")
util_summary <- util_summary %>% left_join(rep, by = "date")
util_summary <- util_summary %>% left_join(bor, by = "date")
util_summary[is.na(util_summary)] <- 0
util_summary <- util_summary %>%
  arrange(date)
util_summary$cum_deposit <- cumsum(util_summary$deposit_amount)
util_summary$cum_redeem <- cumsum(util_summary$redeem_amount)
util_summary$cum_borrow <- cumsum(util_summary$borrow_amount)
util_summary$cum_repay <- cumsum(util_summary$repay_amount)
util_summary$all_money <- util_summary$cum_deposit - util_summary$cum_redeem

# Best attempt at approximating utilization ratio
util_summary$rate <- round((util_summary$cum_borrow) / (util_summary$all_money +util_summary$cum_repay)
util_summary[is.na(util_summary)] <- 0
# https://docs.aave.com/risk/liquidity-risk/historical-utilization
ggplot(util_summary)+
  geom_line(aes(x = date, y = rate))+
    scale_x_date(date_breaks = "1 month", date_labels = "%b-%y")+
    ggtitle(paste(coins,  "Utilization Summary"))
```



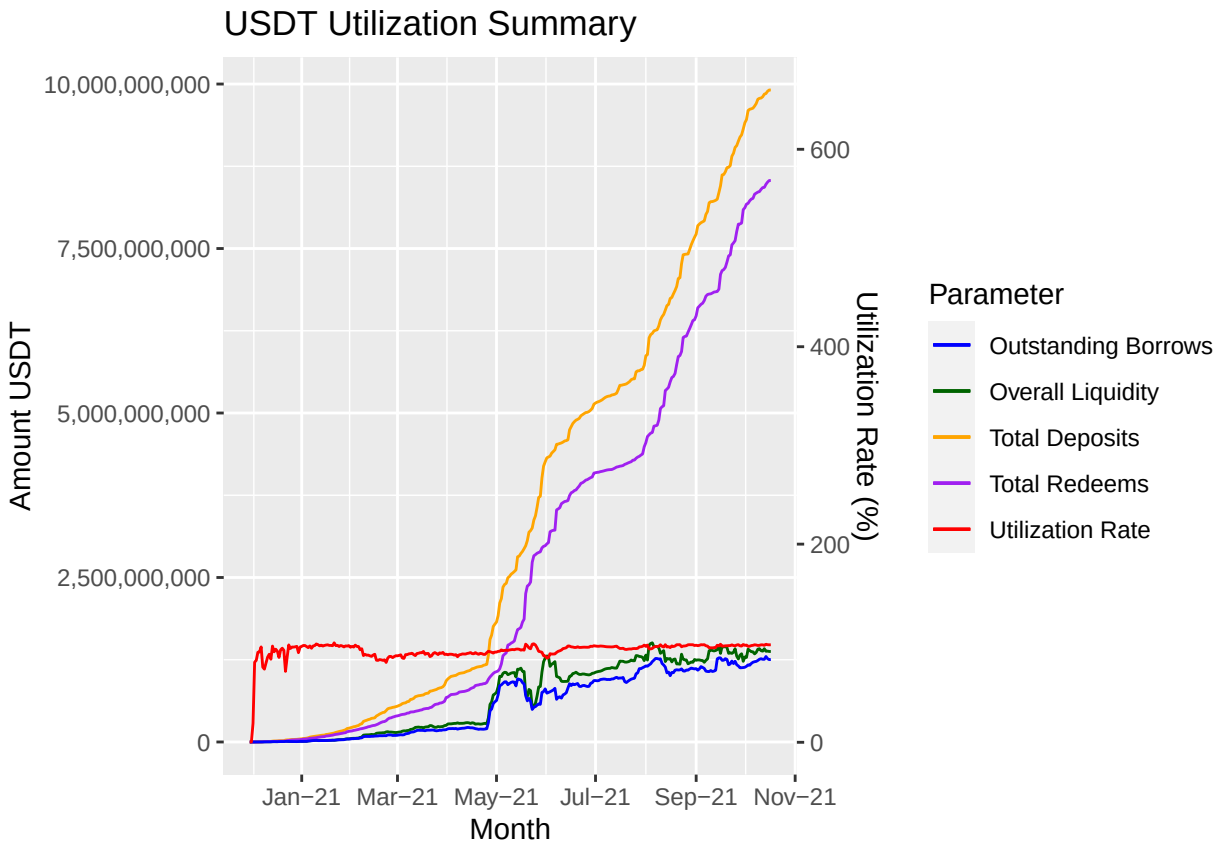
We can see how this is heavily utilized. This makes sense because the optimal utilization on AAVE is 90%. Let's see how this compares.

```
mean(util_summary$rate)
```

```
## [1] 92.63391
```

This is fairly close to the desired rate. However, it does not actually visualize how the coin is being used. For that I used the dual-axis technique from the last section to show different transaction types as well.

```
x <- max(util_summary$all_money) / max(util_summary$rate)

ggplot(util_summary, aes(x = date))+
  geom_line(aes(y = all_money, colour = "Overall Liquidity"))+
  geom_line(aes(y = cum_deposit, colour = "Total Deposits"))+
  geom_line(aes(y = cum_redeem, colour = "Total Redeems"))+
  geom_line(aes(y = cum_borrow - cum_repay, colour = "Outstanding Borrows")) +
  geom_line(aes(y = rate*x, colour = "Utilization Rate"))+
  scale_y_continuous(labels = comma, sec.axis = sec_axis(~./x, name = "Utilization Rate (%)"))+
  scale_colour_manual(values = c("blue", "dark green", "orange", "purple", "red"))+
  labs(y = paste("Amount", coins),x = "Month",colour = "Parameter")+
  scale_x_date(date_breaks = "2 months", date_labels = "%b-%y")+
  ggtitle(paste(coins,  "Utilization Summary"))
```



This is able to give us a lot more information about the coin while also showing what my utilization rate is representing. Also not included, but will be better seen on Monday, is how this chart differs between stable and not stable coins. Aaron's done a good job in separating them and that will soon be included in this. Next, I make a dataframe for every coin in our dataframe that gathers all of this data and makes it easier to plot.

```
coins <- unique(df$reserve)
dates <- as.data.frame(unique(df$date))
colnames(dates) <- c("date")
util_summary <- as.Date(dates$date)
util_summary <- as.data.frame(util_summary)
colnames(util_summary) <- c("date")
util_summary <- util_summary %>%
    arrange(date)
#Similar code for above but it works for more than one coin by changing column names
for (coin in coins) {
  coin.df <- df %>%
    filter(reserve == coin)
  dep <- coin.df %>%
    filter(type == "deposit") %>%
    group_by(date) %>%
    summarise(date = date, deposit_amount = cumsum(sum(amount)))%>%
    distinct()
  red<-coin.df %>%
    filter(type == "redeem") %>%
    group_by(date) %>%
    summarise(date= date, redeem_amount = cumsum(sum(amount)))%>%
    distinct()
```

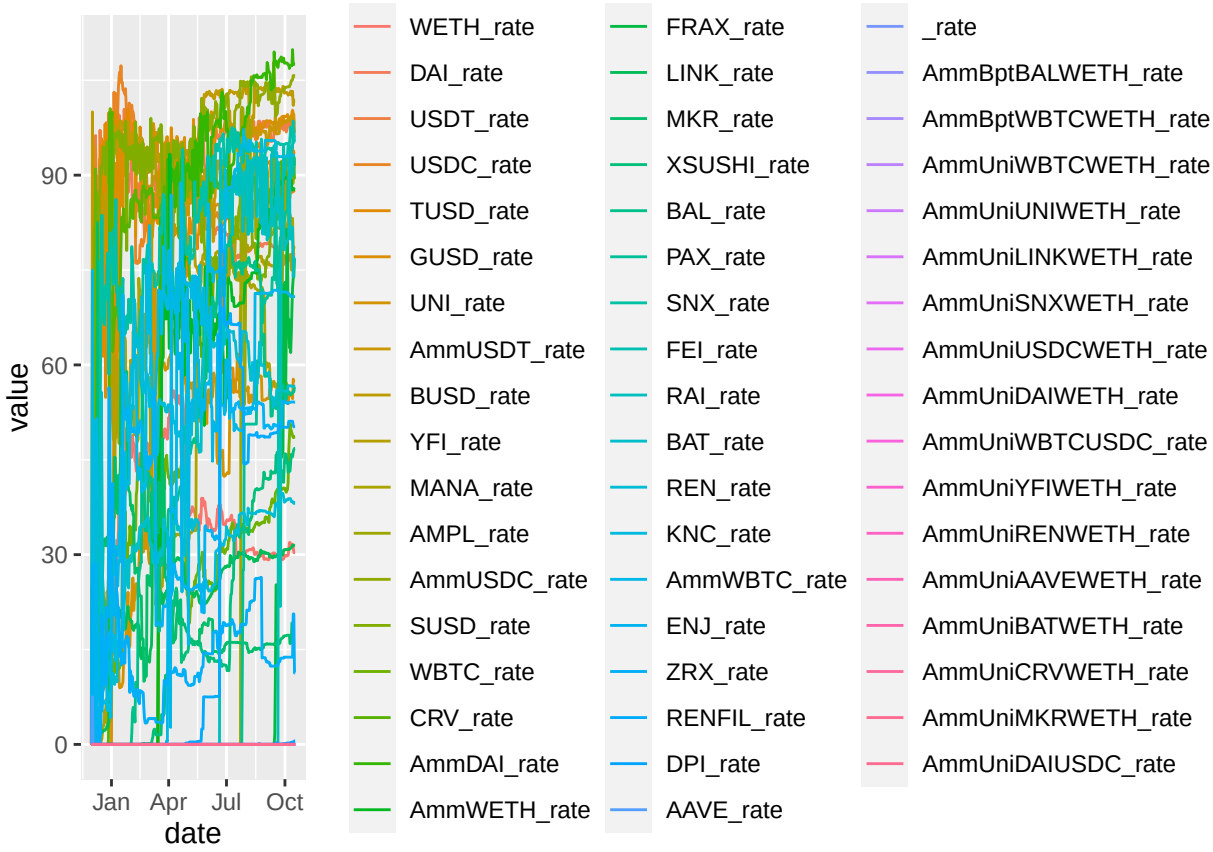
```
  rep <- coin.df %>%
    filter(type == "repay") %>%
    group_by(date) %>%
    summarise(date = date, repay_amount = cumsum(sum(amount)))%>%
    distinct()
  bor <- coin.df %>%
    filter(type == "borrow") %>%
    group_by(date) %>%
    summarise(date = date, borrow_amount = cumsum(sum(amount)))%>%
    distinct()
  util_summary <- util_summary %>% left_join(dep, by = "date")
  util_summary <- util_summary %>% left_join(red, by = "date")
  util_summary <- util_summary %>% left_join(rep, by = "date")
  util_summary <- util_summary %>% left_join(bor, by = "date")
  util_summary[is.na(util_summary)] <- 0

  util_summary$cum_deposit <- cumsum(util_summary$deposit_amount)
  util_summary$cum_redeem <- cumsum(util_summary$redeem_amount)
  util_summary$cum_borrow <- cumsum(util_summary$borrow_amount)
  util_summary$cum_repay <- cumsum(util_summary$repay_amount)
  util_summary$all_money <- util_summary$cum_deposit - util_summary$cum_redeem
  util_summary$rate <- round((util_summary$cum_borrow) / (util_summary$all_money +util_summary$cum_repay
  # Change all the column names to include coin
  colnames(util_summary)[which(names(util_summary)== "deposit_amount")] <- paste(coin, "_dep", sep = '')
  colnames(util_summary)[which(names(util_summary)== "redeem_amount")] <- paste(coin, "_red", sep = '')
  colnames(util_summary)[which(names(util_summary)== "repay_amount")] <- paste(coin, "_rep", sep = '')
  colnames(util_summary)[which(names(util_summary)== "borrow_amount")] <- paste(coin, "_bor", sep = '')
  colnames(util_summary)[which(names(util_summary)== "cum_deposit")] <- paste(coin, "_c_dep", sep = '')
  colnames(util_summary)[which(names(util_summary)== "cum_redeem")] <- paste(coin, "_c_red", sep = '')
  colnames(util_summary)[which(names(util_summary)== "cum_repay")] <- paste(coin, "_c_rep", sep = '')
  colnames(util_summary)[which(names(util_summary)== "cum_borrow")] <- paste(coin, "_c_bor", sep = '')
  colnames(util_summary)[which(names(util_summary)== "all_money")] <- paste(coin, "_all", sep = '')
  colnames(util_summary)[which(names(util_summary)== "rate")] <- paste(coin, "_rate", sep = '')
  head(util_summary)
}
util_summary[is.na(util_summary)] <- 0
#Just shows example for the first two coins
head(util_summary[,1:21])
```

```
##         date    WETH_dep     WETH_red     WETH_rep   WETH_bor   WETH_c_dep
## 1 2020-11-30      0.0002      0.00000  0.000000000   0.000000       0.0002
## 2 2020-12-01      0.0020      0.00200  0.000000000   0.000000       0.0022
## 3 2020-12-02      0.8200      0.26552  0.105000290   0.115000       0.8222
## 4 2020-12-03 374795.1693 372570.72124  0.005000081   5.657405  374795.9915
## 5 2020-12-04 435008.2530 432493.32856  3.135414418  58.425536  809804.2445
## 6 2020-12-05 555710.3020 552695.65909 10.627589546 198.566212 1365514.5465
##     WETH_c_red WETH_c_bor WETH_c_rep   WETH_all WETH_rate     DAI_dep   DAI_red
## 1 0.000000e+00   0.000000  0.0000000    0.00020      0.00      0.0000     0.000
## 2 2.000000e-03   0.000000  0.0000000    0.00020      0.00      5.0000     1.000
## 3 2.675200e-01   0.115000  0.1050003    0.55468     17.43    251.0337   103.000
## 4 3.725710e+05   5.772405  0.1100004 2225.00273      0.26 169902.6744  3113.903
## 5 8.050643e+05  64.197941  3.2454148 4739.92714      1.35 258560.6242 11528.400
## 6 1.357760e+06 262.764153 13.8730043 7754.57007      3.38 439326.8812 33009.094
##        DAI_rep  DAI_bor   DAI_c_dep DAI_c_red DAI_c_bor    DAI_c_rep
```

```
## 1     0.000000      0.0      0.0000     0.000       0.0     0.000000
## 2     0.000000      1.0      5.0000     1.000       1.0     0.000000
## 3     2.000002     25.0    256.0337   104.000      26.0     2.000002
## 4  5006.000010 144101.0 170158.7080  3217.903  144127.0  5008.000012
## 5 75826.182417 232771.2 428719.3322 14746.303  376898.2 80834.182429
## 6   200.000000 264914.5 868046.2134 47755.397  641812.8 81034.182429
##       DAI_all DAI_rate
## 1      0.0000     0.00
## 2      4.0000    25.00
## 3    152.0337    16.88
## 4 166940.8052    83.82
## 5 413973.0297    76.17
## 6 820290.8169    71.21
```

This data frame will be very useful because it allows us to make plots that are less reactive. Here's an example of that.

```
utils <- util_summary[grepl('_rate', colnames(util_summary))]
utils["date"] = util_summary$date
# Reshape package makes it easier to plot
utils <- melt(utils, id.vars = "date")
ggplot(utils, aes(date, y = value, color = variable))+
  geom_line()
```



This shows the utilization rate of every coin in our dataset over time. Although it is not overly useful by itself, it does show the wide range that represents the different usages for every coin. This can be easily adapted into a function that when provided with a list of coins quickly plots their utilization. I believe this can be a very useful tool moving forward. For next week, I am looking to have summary plots for each type

of coin that we have found, while also continuing to improve upon my utilization model.

Appendix

Some more visualizations and work on clusters and potential smart contract activity can be found in paquic-Assignment6-f21.Rmd on github.