

DeFi Example Notebook: Jessica Coco

Data Analytics Research Assignment 1 (Fall 2021)

Jessica Coco

09/01/2021

Introductory Decentralized Finance (DeFi) Research Notebook

This notebook is broken into two main parts:

- * Part 1 is a basic introduction to github and RStudio Server
- * Part 2 is an introduction to the DeFi transaction dataset

This R Notebook and its related R scripts provide a very basic introduction to an interesting **Decentralized Finance (DeFi)** dataset. All data was obtained by querying an API on The Graph, an indexing protocol for querying networks like Ethereum, for transaction data based on the AAVE protocol. For more information on AAVE see the AAVE developer notes. The AAVE protocol is based on Ethereum, an important cryptocurrency platform.

The RPI github repository for all the code required for this notebook, including a snapshots of AAVE transaction and user data, may be found at:

- <https://github.rpi.edu/DataINCITE/IDEA-Blockchain>

The IDEA-Blockchain github also contains notebooks used to harvest the AAVE dataset, which you are welcome to examine.

BEFORE YOU BEGIN

To contribute or submit to any RPI github repository you must validate your RPI github.com ID and send a confirmation email to John Erickson at erickj4@rpi.edu. Please do the following **now**:

- Browse to <http://github.rpi.edu>
- Log in using your RPI credentials
- **PLEASE DO THIS IMMEDIATELY BEFORE READING ANY FURTHER!!**

DAR ASSIGNMENT 1: CLONING A NOTEBOOK AND UPDATING THE REPOSITORY

In this assignment we're asking you to...

- clone the IDEA-Blockchain github repository...
- create a personal branch using git...
- copy and change an R notebook...
- generate ("knit") a PDF based on that notebook, and...
- add these new files by "committing" and "pushing" to the github repository

The instructions which follow explain how to accomplish this.

NOTE: For DAR Fall 2021 you *must* be using RStudio Server on the IDEA Cluster. Instructions for accessing “The Cluster” appear at the end of this notebook. Also, don’t forget to validate your RPI github ID as above and email erickj4@rpi.edu

Cloning an RPI github repository

The recommended procedure for cloning and using this repository is as follows:

- Access the RPI network via VPN
 - See <https://itssc.rpi.edu/hc/en-us/articles/360008783172-VPN-Connection-and-Installation> for information
- Access RStudio Server on the IDEA Cluster at <http://lp01.idea.rpi.edu/rstudio-ose/>
 - You must be on the RPI VPN!!
- Access the Linux shell on the IDEA Cluster by clicking the **Terminal** tab of RStudio Server (lower left panel).
 - You now see the Linux shell on the IDEA Cluster
 - `cd` (change directory) to enter your home directory using: `cd ~`
 - Type `pwd` to confirm
 - NOTE: Advanced users may use `ssh` to directly access the Linux shell from a macOS or Linux command line
- Type `git clone https://github.rpi.edu/DataINCITE/IDEA-Blockchain.git` from within your home directory
 - This will create a new directory **IDEA-Blockchain**
- In the Linux shell, `cd` to **IDEA-Blockchain/DefiResearch/StudentNotebooks**
 - Type `ls -al` to list the current contents
 - Don’t be surprised if you see many files!
- In the Linux shell, type `git checkout -b dar-yourrcs` where `yourrcs` is your RCS id
 - For example, if your RCS is `erickj4`, your new branch should be `dar-erickj4`
 - It is *critical* that you include your RCS id in your branch id
- Now in the RStudio Server UI, navigate to the **IDEA-Blockchain/DefiResearch/StudentNotebooks** directory via the **Files** panel (lower right panel)
 - Under the **More** menu, set this to be your **R working directory**
 - *Setting the correct working directory is essential for interactive R use!*

REQUIRED FOR ASSIGMENT 1

1. In RStudio...
 - Open `blockchain-notebook-f21.Rmd`
 - **NOTE:** When opening this `.Rmd` RStudio may warning you that some required packages are not installed, and ask you if you wish to install them.
 - Go ahead and say yes. **DO NOT INTERRUPT PACKAGE INSTALLATION!**
 - **Save As...** using a *new, original, descriptive* filename that **includes your RCS ID!**
 - Example filename for user `erickj4`: `erickj4-assignment1-f21.Rmd`
 - You should see the file appear in the listing in your **Files** panel.
2. Edit your new notebook using RStudio and save your results...
 - Change the `title:` and `subtitle:` headers (at the top of the file)
 - Change the `author:`
 - Change the `date:`
 - **Save** your changes (`<ctrl-s>` works!)
3. Use the RStudio **Knit** command (top of the editing window) to create HTML and/or PDF files
 - Use the down arrow next to the word **Knit** and select **Knit to HTML** (Optional but handy for previewing)
 - Use the down arrow next to the word **Knit** and select **Knit to PDF** (Required for assignment)
 - Repeat as necessary
4. In the Linux terminal, use `git add` to add each new file you want to add to the repository

- Type: `git add yourfilename.Rmd`
 - Type: `git add yourfilename.html` (created when you knit to HTML)
 - Type: `git add yourfilename.pdf` (created when you knit to PDF)
5. When you're ready, in Linux commit your changes:
 - Type: `git commit -m "some comment"` where "some comment" is a useful comment describing your changes
 - This commits your changes to your local repo, and sets the stage for your next operation.
 6. Finally, "push" the commits in your working branch to the RPI github repo
 - Type: `git push origin dar-yourrrcs` (where `dar-yourrrcs` is the branch you've been working in)
 - Your changes are now safely on the RPI github, under your branch name.
 7. **REQUIRED:** On the RPI github, submit a pull request for your branch:
 - In a web browser, navigate to <https://github.rpi.edu/DataINCITE/IDEA-Blockchain>
 - In the drop-down that says **Master** (top left above the list of files) click and select *your* branch
 - It should give you the option to create a pull request
 - **Submit a pull request for your branch**
 - Eventually one of the DAR instructors will merge your branch into the master branch of the repo.
 8. Confirm what you just did; make the following additional edits:
 - What is the location of the github: *Type repository URL here*
 - What is your github ID: *Type your RCS ID here*
 - What is the name of your new branch: *Type your new branch name here*
 - What is the name of your new (copied) notebook: *Type your new notebook name here*
 - Save your changes and **knit** an updated PDF.
 9. Re-commit these fresh changes to the github
 - Confirm that you are still in your branch; type: `git branch`
 - `git add` your Rmd and PDF
 - `git commit -m` with a fresh message
 - `git push origin` your branch.
 - Go to github and select your branch again; if your previous push has already been merged, submit another pull request.
 - More than likely your previous pull request hasn't been merged and your newest commit was automatically added to your existing request.
 10. Download your PDF and upload to LMS:
 - In the RStudio **Files** panel, select your newly created PDF file (check the checkbox to its left) and select **Export** under the **More** menu.
 - This downloads your PDF file to your personal machine.
 - Now proceed to LMS and upload the PDF you created!

Please also see this handy github "cheatsheet": <https://education.github.com/git-cheat-sheet-education.pdf>

Exploring a DeFi Transaction Dataset using AAVE

This section is provided as background and is not required for Assignment 1.

What is AAVE?

From the developer site: *Aave is a decentralised non-custodial liquidity protocol where users can participate as depositors or borrowers. Depositors provide liquidity to the market to earn a passive income, while borrowers are able to borrow in an over-collateralised (perpetually) or under-collateralised (one-block liquidity) fashion... The (Aave) protocol is implemented as a set of smart contracts on top of the Ethereum blockchain. Smart contracts guarantee safety and do not require a middleman.*

For (much) more detail refer to the AAVE Protocol V2.0 Whitepaper

Prepare Transaction Data

We begin by loading our prepared AAVE transaction data into a dataframe. The dataset has over 400,000 rows, and 27 columns.

We are directly loading the dataframe from an Rds archive instead of a CSV file to conserve space.

```
#load Rds (binary version of csv file) into dataframe
df<-read_rds('../Data/transactions.Rds')
```

```
# Let's take a quick look
head(df)
```

```
##      amount borrowRate borrowRateMode  onBehalfOf      pool reserve
## 1  41501.63   6.274937      Variable 8.502518e+47 1.034668e+48    DAI
## 2 7000000.00   2.589628      Variable 4.635974e+47 1.034668e+48    USDT
## 3   15000.00   8.802541      Variable 3.735263e+47 1.034668e+48    USDC
## 4    8193.19  48.747052      Stable 6.896232e+47 1.034668e+48    USDC
## 5   11000.00   3.225055      Variable 1.089455e+48 1.034668e+48    USDT
## 6   40000.00   5.739208      Variable 2.178337e+47 1.034668e+48    USDT
##      timestamp      user      type reservePriceETH reservePriceUSD amountUSD
## 1 1621340435 8.502518e+47 borrow   2.852900e+14      0.9948044   41286.00
## 2 1622477822 4.635974e+47 borrow   3.812835e+14      1.0000000  7000000.00
## 3 1619775984 3.735263e+47 borrow   3.611000e+14      1.0043389   15065.08
## 4 1615481632 6.896232e+47 borrow   5.562201e+14      0.9993909    8188.20
## 5 1626914745 1.089455e+48 borrow   4.971100e+14      1.0000000   11000.00
## 6 1620936688 2.178337e+47 borrow   2.725248e+14      1.0000000   40000.00
##      collateralAmount collateralReserve principalAmount principalReserve
## 1                  NA                  NA
## 2                  NA                  NA
## 3                  NA                  NA
## 4                  NA                  NA
## 5                  NA                  NA
## 6                  NA                  NA
##      reservePriceETHPrincipal reservePriceUSDPrincipal reservePriceETHCollateral
## 1                  NA                  NA                  NA
## 2                  NA                  NA                  NA
## 3                  NA                  NA                  NA
## 4                  NA                  NA                  NA
## 5                  NA                  NA                  NA
## 6                  NA                  NA                  NA
##      reservePriceUSDCollateral amountUSDPincipal amountUSDCollateral
## 1                  NA                  NA                  NA
## 2                  NA                  NA                  NA
## 3                  NA                  NA                  NA
## 4                  NA                  NA                  NA
## 5                  NA                  NA                  NA
## 6                  NA                  NA                  NA
##      borrowRateModeFrom borrowRateModeTo stableBorrowRate variableBorrowRate
## 1                  NA                  NA
## 2                  NA                  NA
## 3                  NA                  NA
## 4                  NA                  NA
## 5                  NA                  NA
## 6                  NA                  NA
```

```
str(df)
```

```
## 'data.frame': 481519 obs. of 26 variables:
## $ amount : num 41502 7000000 15000 8193 11000 ...
## $ borrowRate : num 6.27 2.59 8.8 48.75 3.23 ...
## $ borrowRateMode : Factor w/ 3 levels "", "Stable", "Variable": 3 3 3 2 3 3 3 3 3 2 ...
## $ onBehalfOf : num 8.50e+47 4.64e+47 3.74e+47 6.90e+47 1.09e+48 ...
## $ pool : num 1.03e+48 1.03e+48 1.03e+48 1.03e+48 1.03e+48 ...
## $ reserve : Factor w/ 50 levels "", "AAVE", "AmmBptBALWETH",...: 29 45 44 44 45 45 44
## $ timestamp : int 1621340435 1622477822 1619775984 1615481632 1626914745 1620936688
## $ user : num 8.50e+47 4.64e+47 3.74e+47 6.90e+47 1.09e+48 ...
## $ type : Factor w/ 6 levels "borrow", "deposit",...: 1 1 1 1 1 1 1 1 1 1 ...
## $ reservePriceETH : num 2.85e+14 3.81e+14 3.61e+14 5.56e+14 4.97e+14 ...
## $ reservePriceUSD : num 0.995 1 1.004 0.999 1 ...
## $ amountUSD : num 41286 7000000 15065 8188 11000 ...
## $ collateralAmount : num NA NA NA NA NA NA NA NA NA NA ...
## $ collateralReserve : Factor w/ 25 levels "", "AAVE", "AmmBptBALWETH",...: 1 1 1 1 1 1 1 1 1 1 ...
## $ principalAmount : num NA NA NA NA NA NA NA NA NA NA ...
## $ principalReserve : Factor w/ 27 levels "", "AmmDAI", "AmmUSDC",...: 1 1 1 1 1 1 1 1 1 1 ...
## $ reservePriceETHPrincipal : num NA NA NA NA NA NA NA NA NA NA ...
## $ reservePriceUSDPrincipal : num NA NA NA NA NA NA NA NA NA NA ...
## $ reservePriceETHCollateral : num NA NA NA NA NA NA NA NA NA NA ...
## $ reservePriceUSDCollateral : num NA NA NA NA NA NA NA NA NA NA ...
## $ amountUSDPrincipal : num NA NA NA NA NA NA NA NA NA NA ...
## $ amountUSDCollateral : num NA NA NA NA NA NA NA NA NA NA ...
## $ borrowRateModeFrom : Factor w/ 3 levels "", "Stable", "Variable": 1 1 1 1 1 1 1 1 1 1 ...
## $ borrowRateModeTo : Factor w/ 3 levels "", "Stable", "Variable": 1 1 1 1 1 1 1 1 1 1 ...
## $ stableBorrowRate : num NA NA NA NA NA NA NA NA NA NA ...
## $ variableBorrowRate : num NA NA NA NA NA NA NA NA NA NA ...
```

```
summary(df)
```

```
## amount borrowRate borrowRateMode onBehalfOf
## Min. : 0 Min. : 0.0 :386542 Min. :2.578e+33
## 1st Qu.: 24 1st Qu.: 3.3 Stable : 18408 1st Qu.:4.174e+47
## Median : 1427 Median : 3.9 Variable: 76569 Median :7.522e+47
## Mean : 191103 Mean : 9.5 Mean :7.592e+47
## 3rd Qu.: 24382 3rd Qu.: 10.8 3rd Qu.:1.168e+48
## Max. :600000000 Max. :10002.0 Max. :1.461e+48
## NA's :7289 NA's :386542 NA's :7289
## pool reserve timestamp user
## Min. :9.862e+47 USDC :105937 Min. :1.607e+09 Min. :2.578e+33
## 1st Qu.:1.035e+48 WETH :105279 1st Qu.:1.615e+09 1st Qu.:4.199e+47
## Median :1.035e+48 USDT : 58266 Median :1.621e+09 Median :8.697e+47
## Mean :1.034e+48 DAI : 55211 Mean :1.620e+09 Mean :8.082e+47
## 3rd Qu.:1.035e+48 LINK : 26404 3rd Qu.:1.624e+09 3rd Qu.:1.173e+48
## Max. :1.035e+48 WBTC : 26344 Max. :1.629e+09 Max. :1.461e+48
## (Other):104078
## type reservePriceETH reservePriceUSD
## borrow : 94977 Min. :1.000e+00 Min. :0.000e+00
## deposit :192006 1st Qu.:2.865e+14 1st Qu.:1.000e+00
## liquidation: 6289 Median :4.652e+14 Median :1.000e+00
## redeem :126705 Mean :3.458e+23 Mean :6.774e+08
## repay : 60542 3rd Qu.:9.411e+14 3rd Qu.:1.000e+00
```

```

## swap      : 1000    Max.    :1.647e+28    Max.    :4.252e+13
##           NA's    :7289           NA's    :7289
## amountUSD collateralAmount collateralReserve principalAmount
## Min.      : 0      Min.      : 0           :475230    Min.      : 0
## 1st Qu.: 70      1st Qu.: 1    WETH    : 2665    1st Qu.: 962
## Median : 5836    Median : 14    LINK    : 1312    Median : 4362
## Mean    : 245851  Mean    : 5451  WBTC    : 686    Mean    : 66005
## 3rd Qu.: 49871   3rd Qu.: 250   AAVE    : 333    3rd Qu.: 21533
## Max.    :754379487 Max.    :4638724 UNI     : 230    Max.    :4475668
## NA's    :7289    NA's    :475230 (Other): 1063    NA's    :475230
## principalReserve reservePriceETHPrincipal reservePriceUSDPrincipal
##           :475230    Min.    :1.000e+00    Min.    : 0.0
## USDC      : 2142    1st Qu.:4.062e+14    1st Qu.: 1.0
## USDT      : 1549    Median :4.682e+14    Median : 1.0
## DAI       : 1459    Mean    :1.556e+17    Mean    : 295.6
## GUSD      : 242     3rd Qu.:5.363e+14    3rd Qu.: 1.0
## TUSD      : 175     Max.    :4.203e+19    Max.    :83819.1
## (Other): 722     NA's    :475230    NA's    :475230
## reservePriceETHCollateral reservePriceUSDCollateral amountUSDPincipal
## Min.      :1.000e+00    Min.      :0.000e+00    Min.      : 0
## 1st Qu.:1.000e+00    1st Qu.:0.000e+00    1st Qu.: 1022
## Median :5.110e+14    Median :1.000e+00    Median : 4481
## Mean    :2.177e+21    Mean    :4.543e+06    Mean    : 67361
## 3rd Qu.:1.110e+16    3rd Qu.:2.600e+01    3rd Qu.: 22066
## Max.    :9.116e+23    Max.    :2.509e+09    Max.    :4571839
## NA's    :475230    NA's    :475230    NA's    :475230
## amountUSDCollateral borrowRateModeFrom borrowRateModeTo stableBorrowRate
## Min.      : 0           :480519           :480519    Min.      : 0.0
## 1st Qu.: 0      Stable : 471    Stable : 529    1st Qu.: 9.0
## Median : 476    Variable: 529    Variable: 471    Median : 10.9
## Mean    : 37060                                Mean    : 11.7
## 3rd Qu.: 7457                                3rd Qu.: 12.0
## Max.    :5029023                                Max.    :154.7
## NA's    :475230                                NA's    :480519
## variableBorrowRate
## Min.      : 0.0
## 1st Qu.: 3.8
## Median : 3.9
## Mean    : 5.7
## 3rd Qu.: 5.1
## Max.    :148.7
## NA's    :480519

```

Analyze Transaction Types

Next, we will examine the different types of transactions present in the data. We make a bar plot to visualize the number of each transaction types. Deposit is the most common type of transaction, whereas swaps are the most rare.

```

#set color palette
colors = brewer.pal(6,"Set2")

#create barplot
barplot(table(df$type), main='Transaction Type Counts', xlab='Type',ylab='Count',col=colors)

```



There are more deposits than borrows, because users often need to overcollateralize for loans.

Now, we will examine the amount of US dollars being used in the different types of transactions. We create box plots for the 4 types of transactions that have the amount feature associated with them, and visualize the distribution of that column for the different transactions. We can see that most transactions are completed with very little money.

```
#create boxplot
boxplot(amountUSD~type,data=df,outline=FALSE,col=colors,main="Transaction Amounts",xlab="Type",ylab="Amount")
```

Transaction Amounts



```
boxplot(log(amountUSD)~type,data=df,outline=FALSE,col=colors,main="Log Transaction Amounts",xlab="Type")
```

Log Transaction Amounts



There are many borrows and repays with high transactions amounts, but deposits and redeems have much lower transactions amounts.

Look at Sample User Transaction Histories

Finally, we will examine the transaction history of different users. To do this, we will select 3 random users from the data who have completed between 100 and 300 transactions. Then, we create swarmplots displaying the different types of transactions those users made over time.

```
#set seed
set.seed(1)

#get 3 random users that have between 100 and 300 transactions
users<-vector(length=3)
count<-0
while(count<=3){
  success<-FALSE
  while(!success){
    #get random user
    ruser<-sample(df$user,1)

    #check for valid number of transactions
    length<-nrow(filter(df,user==ruser))
    if (length>100 && length<300){
      users[count]=ruser
      success<-TRUE
      count<-count+1
    }
  }
}
df.rusers<-filter(df, user %in%users)

#create swarmplot

#liquidations
#borrow
#deposit
#redeem
#repay
#swap

ggplot(df.rusers,aes(user, timestamp,color=type)) +
  geom_beeswarm(cex=1)+
  coord_flip()+
  ggtitle("User Transaction History")
```



Users have very different transactions patterns, which we will try to better understand.

Analyze Individual Currencies (USDT)

USDT is interesting because it has more borrows than deposits. This may be because it is a stable coin.

```
df.usd<-filter(df,reserve=="USDT")
barplot(table(df.usd$type), main='Transaction Type Counts', xlab='Type',ylab='Count',col=colors)
```



APPENDIX: Accessing RStudio Server on the IDEA Cluster

The IDEA Cluster provides five compute nodes (4x 48 cores, 1x 80 cores, 1x storage server)

- The Cluster requires RCS credentials, enabled via registration in class
 - email John Erickson for problems `erickj4@rpi.edu`
- RStudio, Jupyter, MATLAB, GPUs (on two nodes); lots of storage and computes
- Access via RPI physical network or VPN only

RStudio GUI Access for DAR:

- Access the RPI VPN
- Browse to: `http://lp01.idea.rpi.edu/rstudio-ose/` (RStudio Server)
- Log in using your RCS username and password
 - If you cannot log in, contact John Erickson at `erickj4@rpi.edu`

Shared Data on Cluster:

- Users enrolled in DAR have access to `/academics/MATP-4910-F21`
 - Usually DAR users will see a symbolic (“soft”) link in their home directories
 - If you do not, type the following in the **Terminal** via RStudio: `ln -s /academics/MATP-4910-F21/MATP-4910-F21`

For advanced users:

- All `idea_users` have access to shared storage via `/data` (“data” in your home directories)
 - You might wish to use this for data sharing in team projects...
 - ...but we recommend using github for shared code development
- Shell access to nodes: You must access “landing pad” first, then compute node:
- `ssh your_rcs@lp01.idea.rpi.edu` For example: `ssh erickj4@lp01.idea.rpi.edu`
- Then, `ssh` to the desired compute node, e.g.: `ssh idea-node-02`