NOTE: Both my projects are in .rmd(R Markdown) format. You will require a R Studio to run/Knit this file. I have already knit this file into a .html file which shows the output as well as the code.

Statistics Projects:

**Project 1 Description**

***September 26, 2018***

**Data Description.**

Our data files (shared on elearning/Ethereum datasets) contain two primary groups: token network edge files, and token price files. The **Ethereum project** is a blockchain platform, and our data comes from there. Although Ethereum started in 2015, most tokens have been created since 2016. As such, tokens have different starting dates, and their data starts from that initial date.Token edge files have this row structure: fromNodeID\ttoNodeID\tunixTime\ttokenAmount\r\n

This row implies that fromNodeID sold tokenAmount of the token to toNodeID at time unixTime. fromNodeID and toNodeID are people who invest in the token in real life; each investor can also use multiple addresses. Two addresses can sell/buy tokens multiple times with multiple amounts. For this reason, the network is considered a weighted, directed multi(edge) graph. Each token has a maximum token count *maxt*

; you can think of *maxt*

as the total circulating token amount.

**Clarification for supply (10/3/2018).** There are two things; first each token has a limited supply (i.e., token count, which can be found on coinmarketcap.com as circulating amount). Then each token may have sub-units. This is similar to dollar in the US. There are around 18 trillion dollars in the economy, and each dollar is divided into 100 cents (subunits). Similarly, there is a token supply, and then there is a subunit for each token. This idea comes from Bitcoin where subunits are called Satoshis, 1 Bitcoin =10^8 satoshis. Coin market cap gives the total supply, but not sub-units, which differ from token to token. Some tokens have 1018

sub-units. That means there can be numbers as big as *totalAmount*∗1018

Etherscan.io gives these sub-units as decimals, please see the Vechain here: It has 18 decimals, which means each Vechain token has 1018

subunits. <https://etherscan.io/token/0xd850942ef8811f2a866692a623011bde52a462c1>

Price files have no extensions, but they are text based. If you open them with a text editor (use notepad++ or similar), you will see this row structure: Date\tOpen\tHigh\tLow\tClose\tVolume\tMarketCap\r

The price data is taken from <https://coinmarketcap.com/>. Open and close are the prices of the specific token at the given date. Volume and MarketCap give total bought/sold tokens and market valuation at the date.

**Preprocessing step**

Find your primary token and load its data. In each token, there may be outlier amounts which are bigger than the total amount of the token. Locate these extreme outliers, if exist, and filter them out. If there are many of these (>30), investigate how many users are included in these transactions.

**Question 1 [Due 10/10/2018]**

Find the distribution of how many times a user 1 - buys, 2 - sells a token. Which discrete distribution type fits these distributions best? Estimate distribution parameters.

**Question 2 [Due 10/24/2018, full project report due 10/31/2018]**

How can we create layers of transactions with increasing amounts? This descriptive statistic is similar to bin selection in histograms. For example, we could choose *layer*1

as those transactions that involve 0.01×*maxt*

in amount. Find a good value for the number of layers and justify your choice.

Once you create layers, you can compute a feature in each layer. An example feature is the number of transactions, another one is the number of unique buyers. As each edge has a unix timestamp, it is easy to compute the edge time to a date. For example, 1294226315 is equivalent to 01/05/2011 @ 11:18am (UTC). See the website <https://www.unixtimestamp.com/index.php> for unix time conversion. R has functions to compute dates from unix time stamps as well. This way, for a given day you can find all layer transactions in that day.

For example, you can say on 10/12/2018 there were 25 transactions in layer 1. The price of token on that date was 3.2$. For each day in a token’s history, you can then correlate price vs feature in time.

Find an algorithm to compute the correlation of price data with each of the layers (hint: start by looking at Pearson correlation).

**Project 2 Description**

***Cuneyt Akcora***

***November, 2018***

**Data Description.**

To refresh your memory on our data, please see project 1 description.

Our data files (shared on elearning/Ethereum datasets) contain two primary groups: token network edge files, and token price files.

Token edge files have this row structure: fromNodeID\ttoNodeID\tunixTime\ttokenAmount\r\n

This row implies that fromNodeID sold tokenAmount of the token to toNodeID at time unixTime. fromNodeID and toNodeID are people who invest in the token in real life; each investor can also use multiple addresses.

Price files have no extensions, but they are text based. If you open them with a text editor (use notepad++ or similar), you will see this row structure: Date\tOpen\tHigh\tLow\tClose\tVolume\tMarketCap\r

The price data is taken from <https://coinmarketcap.com/>. Open and close are the prices of the specific token at the given date. Volume and MarketCap give total bought/sold tokens and market valuation at the date.

**Primary Token.**

In project 2, each group will work with the token that you have studied in the first project. This will be your primary token.

**Preprocessing step**

Find your primary token and load its data. In each token, there may be outlier amounts which are bigger than the total amount of the token. Locate these extreme outliers, if exist, and filter them out. If there are many of these (>30), investigate how many users are included in these transactions.

**Quality**

Perhaps the most important aspect of this project is the presentation. Your report should explain each step in your solution, and provide good visuals. You may use the ggplot2library to draw plots. Data science is the art of finding and presenting actionable insights from data. Your output will be a doc/pdf. RMD and html files will not be accepted.

Create a single project report that covers both project 1 and project 2. The report should have less than 15 figures, and not exceed 20 pages. For each extra page and figure, 2 points will be deducted from your total mark.

Project 2 consists of the following question only.

**Question 1 [Due 11/31/2018]**

We denote the token price in dolar as *Pt* for the *t*th day. Simple price return is given as *Pt*−*Pt*−1*Pt*−1

Extract at least three features from the token network at day *t*−1

and create a multiple linear regression model to explain price return on day *t*

. In this task, you can choose to extract features from a single layer computed in project 1, or you can use all network data. If you use a layer approach, you can build the model on data from a single layer only.

You are free to choose any feature (regressor). Your features can be in terms of numbers (*x*1=

number of transactions), percentages (*x*1=percentage of investors who bought more than 10 tokens), etc. Similarly, you could transform your regressors (*x*1=square root of number of transactions).

In some tokens you may not find three useful features to use in the model. In that case, please explain the candidate features that you tried. If needed, you can create a single regressor model as well. Similarly, you can go beyond three regressors.Finding which features to extract from a dataset is called feature engineering.

Present your regression model, explain residuals and discuss your findings. Explain the adequacy of your regression model.