Bitcoin Trust Network Analysis

Agnes Jiang, Weihao Zeng, Shreeya Goyal

March 6, 2018

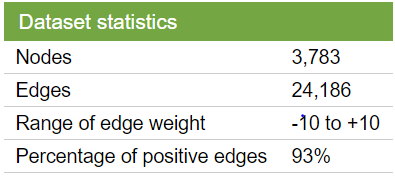
Video link: <https://youtu.be/33kgX-q9gDo>

## Part 1: Dataset Introduction

Bitcoin is a cryptocurrency and worldwide payment system. It is the first decentralized digital currency, as the system works without a central bank or single administrator­­. The network is peer-to-peer and tr­­­­­ansactions take place between users directly, without an intermediary. These transactions are verified by network nodes through the use of cryptography and recorded in a public distributed ledger called a blockchain. [1]

In this project, we are using a data set we obtained from snap.stanford.edu. The network is weighted signed network. Weighted signed networks (WSNs) are networks in which edges are labeled with positive and negative weights. WSNs can capture like/dislike, trust/distrust, and other social relationships between people.

This network consists of who-trusts-whom network of people who trade using Bitcoin on a platform called Bitcoin Alpha. Since Bitcoin users are anonymous, there is a need to maintain a record of users' reputation to prevent transactions with fraudulent and risky users. To maintain anonymity and ensure that a user is not defrauded during a transaction, users needed a measure of trustworthiness and to validate another user’s reputation. Members of Bitcoin Alpha rate other members in a scale of -10 (total distrust) to +10 (total trust).



*Figure 1: Dataset Statistics*

## Part 2: Overview of the dataset

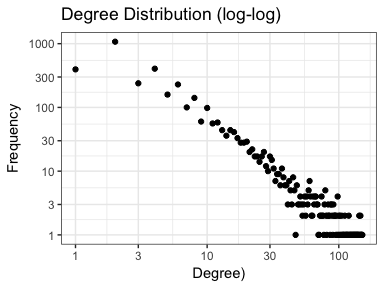
bitcoin\_graph <- **graph.data.frame**(d = bitcoin\_data[,1:3], directed = T)   
**summary**(bitcoin\_graph)

## IGRAPH 41f8926 DN-- 3783 24186 --   
## + attr: name (v/c), strength (e/n)

There are 3,783 nodes and 24,186 edges in the network.

### Degree Distribution

G\_degrees = **degree**(bitcoin\_graph)  
G\_degree\_hist = **as.data.frame**(**table**(G\_degrees))  
G\_degree\_hist[,1] <- **as.numeric**(G\_degree\_hist[,1])  
**ggplot**(G\_degree\_hist, **aes**(x = G\_degrees, y = Freq)) + **geom\_point**() +  
 **scale\_x\_continuous**("Degree", breaks = **c**(1, 3, 10, 30, 100, 300), trans = "log10") +  
 **scale\_y\_continuous**("Frequency", breaks = **c**(1, 3, 10, 30, 100, 300, 1000), trans = "log10") +  
 **ggtitle**("Degree Distribution (log-log)") + **theme\_bw**()



It's a kind of power-law distribution, indicating sparsely connected network in the real world. Older users that entered the transaction market earlier have more edges (transactions) on average, and they prefer to attach to those nodes with many connections.

### Shortest Path & Degree

sp\_in <- **shortest.paths**(bitcoin\_graph, mode='in')  
sp\_out <- **shortest.paths**(bitcoin\_graph, mode='out')  
indegree <- **degree**(bitcoin\_graph, mode='in')  
outdegree <- **degree**(bitcoin\_graph, mode='out')  
sp\_in\_vec <- **vector**()  
sp\_out\_vec <- **vector**()  
for (i in 1:**vcount**(bitcoin\_graph)) {  
 sp\_in\_vec[i] <- **mean**(sp\_in[i,])  
 sp\_out\_vec[i] <- **mean**(sp\_out[i,])  
}  
node\_stats\_df <- **cbind**(indegree, outdegree, sp\_in\_vec, sp\_out\_vec)  
**paste**("Average degree: ",**round**(**mean**(indegree[**which**(indegree != Inf)]), 2))

## [1] "Average degree: 6.39"

**paste**("Average shortest path: ",**round**(**mean**(sp\_in[**which**(sp\_in != Inf)]), 2))

## [1] "Average shortest path: 3.68"

The mean degree of 6.4 shows that on average every user has transactions with 6 people in the transaction market. The mean shortest path of 3.7 indicates that to get first-hand of impression about a user, we need to ask on average 3 users with directed connection.

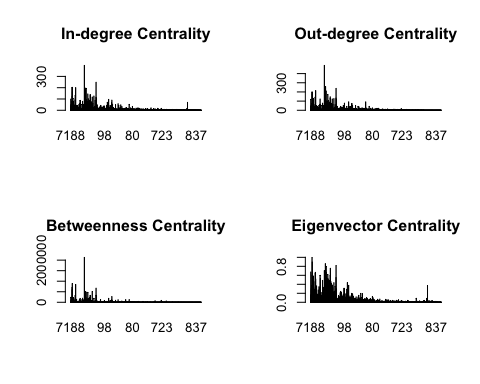
### Centrality analysis

indegree <- **degree**(bitcoin\_graph, mode='in')  
outdegree <- **degree**(bitcoin\_graph, mode='out')  
betweenness <- **betweenness**(bitcoin\_graph)  
bitcoin\_undirected <- **as.undirected**(bitcoin\_graph, mode='collapse')  
ev\_obj <- **evcent**(bitcoin\_undirected)  
eigen <- ev\_obj$vector  
central <- **data.frame**(**V**(bitcoin\_graph)$name, indegree, outdegree, betweenness, eigen)  
  
top\_indegree = **as.vector**(central[**order**(-central$indegree),1][1:5])  
top\_outdegree = **as.vector**(central[**order**(-central$outdegree),1][1:5])  
top\_betweenness = **as.vector**(central[**order**(-central$betweenness),1][1:5])  
top\_eigen = **as.vector**(central[**order**(-central$eigen),1][1:5])  
**data.frame**(**cbind**(top\_indegree, top\_outdegree, top\_betweenness, top\_eigen))

## top\_indegree top\_outdegree top\_betweenness top\_eigen  
## 1 1 1 1 11  
## 2 3 8 2 2  
## 3 2 3 4 177  
## 4 11 4 11 3  
## 5 4 7 3 7

We found that user with id 1 stands on the center of this network in most of the ways to method centrality.

**par**(mfrow=**c**(2,2))  
**barplot**(central$indegree, names.arg=central$V.bitcoin\_graph..name, main = "In-degree Centrality")  
**barplot**(central$outdegree, names.arg=central$V.bitcoin\_graph..name, main = "Out-degree Centrality")  
**barplot**(central$betweenness, names.arg=central$V.bitcoin\_graph..name, main = "Betweenness Centrality")  
**barplot**(central$eigen, names.arg=central$V.bitcoin\_graph..name, main = "Eigenvector Centrality")



* From these 4 plots of centralities, we see that indegree has almost the same pattern with outdegree. It indicates that the rating in this anonymous transaction network is highly interactional. No matter whether the transaction is successful or not, users are very likely to get feedback rating if they give ratings to the one they made transaction.
* The other graphs of centrality also show similar patterns when looking at peaks and low values. But the eigenvector draws our attention. Users with high eigenvectors are not those with the highest degrees, and instead, they are tightly connected to high-scoring users. In our case, if we are not able to connect with users with the highest degrees first, we can try to find those with high eigenvector and ask them to help introduce.
* The high variance of centrality within the network also indicates a very typical network in the social topic, when different users have different patterns of connecting to others. Some are new in the transaction system when they do not have many connections, but some have joined very early and have active interaction inside.

### Centrality correlation

**paste**("Centrality correlation: ", **round**(**cor**(central[,2:5]),3))

## [1] "Centrality correlation:

## indegree outdegree betweenness eigen  
## indegree 1.000 0.970 0.889 0.898  
## outdegree 0.970 1.000 0.908 0.880  
## betweenness 0.889 0.908 1.000 0.682  
## eigen 0.898 0.880 0.682 1.000

**paste**("Reciprocity: ", **round**(**reciprocity**(bitcoin\_graph),3))

## [1] "Reciprocity: 0.832"

* When looking at the correlation of different centrality methods, we see extremely high correlation of 0.97 between indegree and outdegree centrality. This indicates a very integrated network when the ratings are two-ways in most of the cases. The high reciprocity also supports this finding.
* The other centralities are also highly correlations, showing that we may not able to tell much difference from those methods of centralities, and either method can easily capture our important nodes. This is consistent with the result I showed previously about the most important users.
* The relatively low correlation between eigenvector and betweenness centrality also catches our attention. It indicates that while there will be overlap in leadership of this transaction network, users can have many connections but still be on the periphery of a larger community, and users can have few connections but be in the center of the information flows.

### Triad census

census\_labels = **c**('003','012', '102', '021D', '021U', '021C', '111D', '111U',   
 '030T', '030C', '201', '120D', '120U', '120C', '210', '300')  
tc <- **triad.census**(bitcoin\_graph)  
**data.frame**(census\_labels, tc)

## census\_labels tc  
## 1 003 8963414292  
## 2 012 14879710  
## 3 102 36885677  
## 4 021D 43677  
## 5 021U 17083  
## 6 021C 27095  
## 7 111D 93989  
## 8 111U 190593  
## 9 030T 974  
## 10 030C 64  
## 11 201 413062  
## 12 120D 1142  
## 13 120U 1780  
## 14 120C 1163  
## 15 210 7136  
## 16 300 9894

* We also conducted the triad analysis to find patterns within small groups of users in this network. The most frequent patterns are the first three, which makes sense when considering such a large social network. Most users connect to just a few of other users in the system because of limitations such as time and power to transact, while only a few users have strong connections to the others.
* We want to highlight the result on the second half of triad identification. The frequency of the eleventh one, which only exists reciprocal connections between two over three of users, is very high when comparing to the other types.
* It brought a question to us. Since this kind of relation is pretty common, **are we able to predict the relationship between these two people in this triad who have not connected yet based on the information of existing edges?** The good news is that the sufficient data of relations in complete triads (types 9 and 12 to 16) can help us to figure out this question. Based on what information we have, we try to build regression to predict the weighted relationship.

## Part 3: Regression to Predict Weighted Trust

To address this question, we build a linear regression. Suppose we have 3 nodes here, A, B and C. Then the dependent variable would be the strength from the A to C, and the independent variable would be the strength from the A to B, the strength from the B to C, the fairness of A (the average strength that A trust others), the goodness of C (the average strength that C is trusted by others) and the interaction terms.

##   
## Call:  
## lm(formula = onethree ~ onetwo + twothree + goodness + fairness +   
## goodness \* onetwo + goodness \* twothree + goodness \* fairness +   
## fairness \* onetwo + fairness \* twothree + onetwo \* twothree,   
## data = datafor3)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -17.8848 -0.7046 0.0607 1.2175 15.6351   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -0.596636 0.100435 -5.941 3.67e-09 \*\*\*  
## onetwo 0.066357 0.027925 2.376 0.0176 \*   
## twothree 0.015180 0.025092 0.605 0.5453   
## goodness 0.710268 0.044086 16.111 < 2e-16 \*\*\*  
## fairness 0.734457 0.044127 16.644 < 2e-16 \*\*\*  
## onetwo:goodness -0.003984 0.007953 -0.501 0.6165   
## twothree:goodness -0.036598 0.004667 -7.842 9.36e-15 \*\*\*  
## goodness:fairness -0.014242 0.013405 -1.062 0.2882   
## onetwo:fairness -0.003046 0.004749 -0.641 0.5214   
## twothree:fairness -0.030009 0.006647 -4.514 6.94e-06 \*\*\*  
## onetwo:twothree 0.053892 0.003812 14.138 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 2.928 on 1266 degrees of freedom  
## Multiple R-squared: 0.6868, Adjusted R-squared: 0.6844   
## F-statistic: 277.7 on 10 and 1266 DF, p-value: < 2.2e-16

Based on the regression, the adjusted R-square is 0.6844, which means we can predict 68.44% of the variation in the strength from A to C based on this model. Since we have really limited resource in this data (we start from only three columns, which is the id of the ego node, the id of the alternative node and the strength), we want to check whether using a network of 4 which is in similar pattern would enhance the adjusted R-square.

##   
## Call:  
## lm(formula = onefour ~ onetwo + twothree + threefour + onetwo \*   
## twothree + onetwo \* threefour + twothree \* threefour + goodness +   
## fairness + goodness \* onetwo + goodness \* twothree + goodness \*   
## threefour + goodness \* fairness + fairness \* onetwo + fairness \*   
## twothree + fairness \* threefour, data = datafor4)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -14.3585 -0.7654 0.0134 0.8985 11.2272   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -0.861201 0.071845 -11.987 < 2e-16 \*\*\*  
## onetwo 0.013252 0.018549 0.714 0.47503   
## twothree 0.002475 0.016803 0.147 0.88292   
## threefour 0.036609 0.018103 2.022 0.04326 \*   
## goodness 0.837407 0.037685 22.221 < 2e-16 \*\*\*  
## fairness 0.888600 0.033261 26.716 < 2e-16 \*\*\*  
## onetwo:twothree 0.011059 0.003765 2.938 0.00334 \*\*   
## onetwo:threefour 0.011082 0.004972 2.229 0.02591 \*   
## twothree:threefour 0.010574 0.003536 2.991 0.00281 \*\*   
## onetwo:goodness -0.026207 0.008682 -3.019 0.00257 \*\*   
## twothree:goodness -0.009638 0.007790 -1.237 0.21611   
## threefour:goodness -0.009722 0.004506 -2.157 0.03107 \*   
## goodness:fairness -0.014230 0.015533 -0.916 0.35972   
## onetwo:fairness 0.024299 0.004827 5.034 5.16e-07 \*\*\*  
## twothree:fairness -0.016537 0.007243 -2.283 0.02250 \*   
## threefour:fairness -0.022359 0.008481 -2.636 0.00844 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 2.39 on 2415 degrees of freedom  
## Multiple R-squared: 0.6079, Adjusted R-squared: 0.6054   
## F-statistic: 249.6 on 15 and 2415 DF, p-value: < 2.2e-16

Now the adjusted R-square 0.6054, which is lower, and we would find the adjusted R-square would keep decreasing if we build a similar model for group of 5. It seems that a larger network in similar pattern could not offer more predictive power.

Run some basic statistic to compare the group of 3 and 4 and we decide to use coefficient of variance rather than variance because it is more helpful to reflect the fluctuation around their own means. The result matches what we found in the regression models.

**mean**(datafor3$onethree)

## [1] 0.8888019

**mean**(datafor4$onefour)

## [1] 0.4043603

**sd**(datafor3$onethree)/**mean**(datafor3$onethree)

## [1] 5.864221

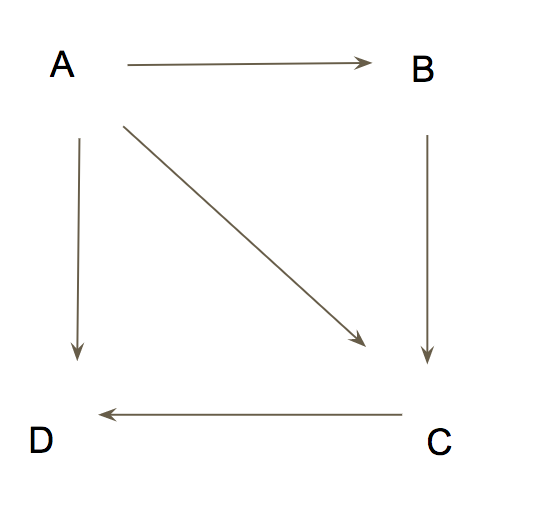
**sd**(datafor4$onefour)/**mean**(datafor4$onefour)

## [1] 9.409377

To make a better prediction, we can use binning for the independent variable or for the dependent variable to increase the adjusted R-square or the AUC (if we have a logistic model after binning), and we can keep adding some attributes of the nodes or some statistics about the strength into the regression formula, but by now we have already reach some conclusions.

## Part 4: Findings

* Suppose there are 4 people, A, B, C and D, then, A knows B, B knows C, C knows D, but A doesn't know C and B doesn't know D, which means there is no sub network in this group of 4. Then the trust from A to D would be weak and not easy to predict. The relation from A to D would improve if A knows C first or B knows D first, which means they form a sub network of 3 people, the following graph is an example that A knows C previously would increase the trust from A to D.



* We find that the network follows the power-law distribution and the correlation between eigenvector and betweenness centrality is relatively low, when we first enter the network and are not able to connect with users with the highest degrees, we can try to find those with high eigenvector and ask them to help introduce. Based on what we found in the prediction part, the strength of trust from the high-degree user to us would be more predictable, stable and powerful than when we are introduced by someone that the high-degree user has no connection to.
* If we fail to build connection with these users, we can connect to those users with high goodness (they receive high-value trust from the others on average), they might not able to introduce us to lots of users, but at least we can have a satisfying transaction with them and we can enlarge our network slowly.

References:

[1] https://en.wikipedia.org/wiki/Bitcoin