Project 2: Social Network Mining

Duan Li 005026839
Di Ma 004945175
Yuan Shao 504880181
Weijie Tang 305029285

Part 1: Facebook Network

1.1 Structural properties of the facebook network

Q1

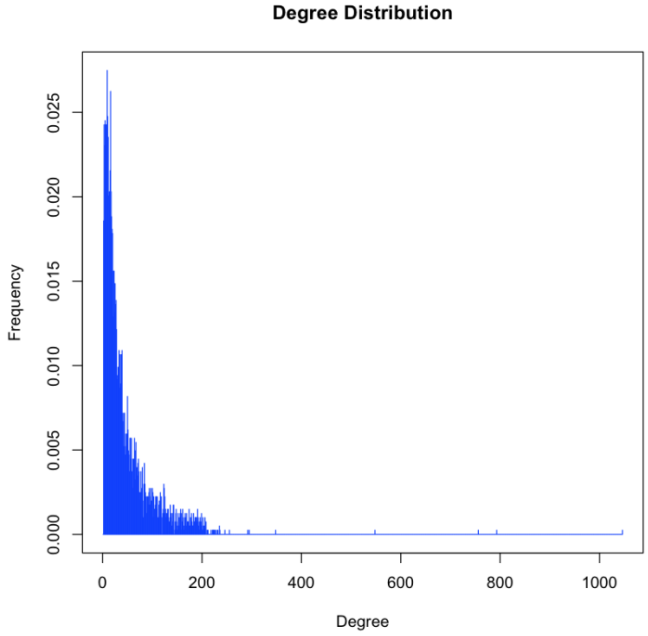
After we construct the facebook network from the edge list file “facebook_combined.txt”, we use *is.connect* function to find that the network is connected. Then we use *vcount*, *ecount*, or *summary* function to find out that the network has 4039 nodes and 88234 edges.

Q2

We use *diameter* function to get that the diameter of the facebook network is 8.

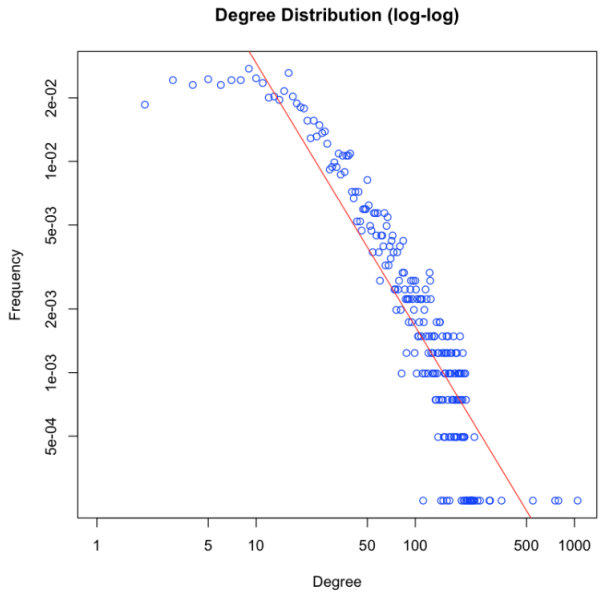
Q3

Then we plot the degree distribution using a histogram as shown below. The average degree is about 43.691.



Q4

We plot the degree distribution in a log-log scale, and fit a linear model to it using *lm* function. The fitted line is the red line in the following figure, and according to the coefficients from the outputs of the *lm* function, its slope is about -1.2475.



1.2 Personalized network

Q5

We use *neighbors* function to get all the neighbors of node 1; then we use *induced.subgraph* function with node 1 and its neighbors to get its personalized network. The personalized network of node 1 has 348 nodes and 2866 edges.

Q6

The diameter of node 1's personalized network is 2. We state that the upper bound for the diameter is 2, and the lower bound for the diameter is 0.

Q7

When the diameter of the personalized network equals 2, which is the upper bound, it means at least two of node 1's friends are not each other's friend. When the diameter of the personalized network equals 0, which is the lower bound, it means node 1 has no friend. Besides, when the diameter of the personalized network equals 1, it means all node 1's friends know each other.

1.3 Core node's personalized network

Q8

After inspecting the degree of all the nodes, we find that there are 40 core nodes, and the average degree of them is 279.375.

1.3.1 Community structure of core node's personalized network

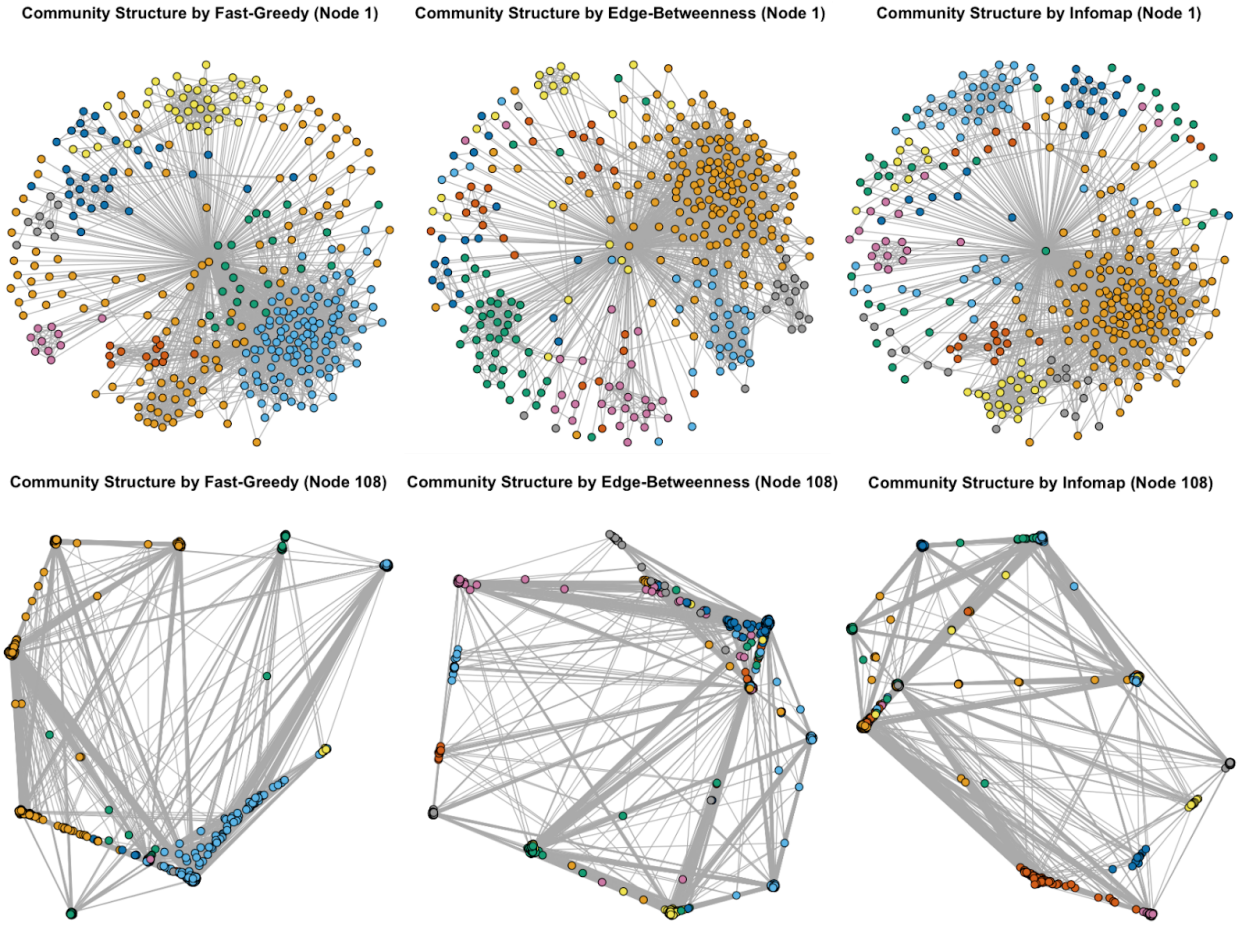
Q9

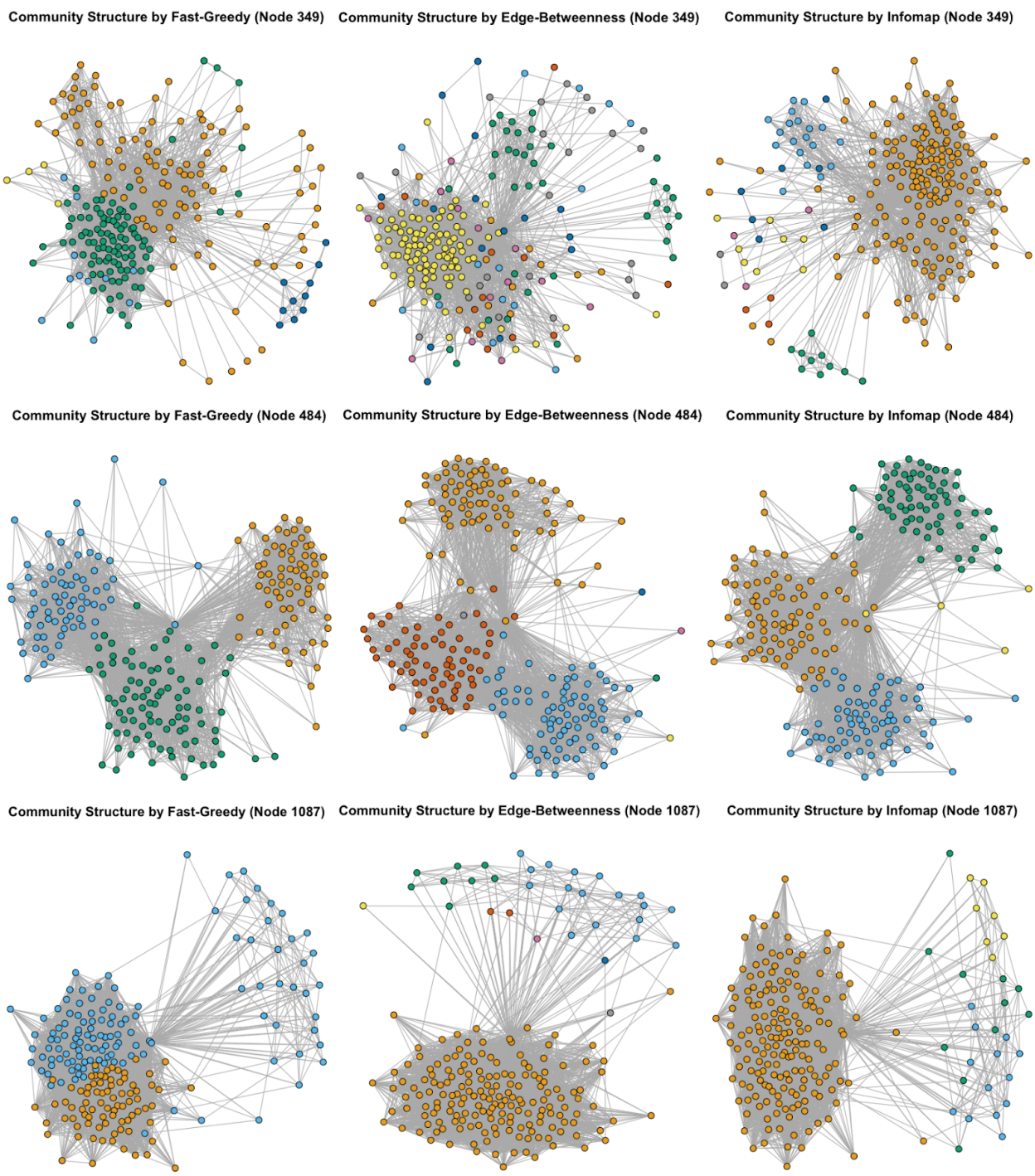
We run Fast-Greedy, Edge-Betweenness, and Infomap community detection algorithms on the personalized network of node 1, node 108, node 349, node 484 and node 1087. From the modularity scores and the community structure visualization results, we find that on average the

Fast-Greedy community detection algorithm gives the best modularity score. Also there are some commonalities among the detected communities by different algorithms.

Modularity scores (personalized network)

Core node	Fast-Greedy	Edge-Betweenness	Infomap
Node 1	0.413	0.353	0.389
Node 108	0.436	0.507	0.508
Node 349	0.252	0.134	0.095
Node 484	0.507	0.489	0.515
Node 1087	0.146	0.028	0.027





1.3.2 Community structure with the core node removed

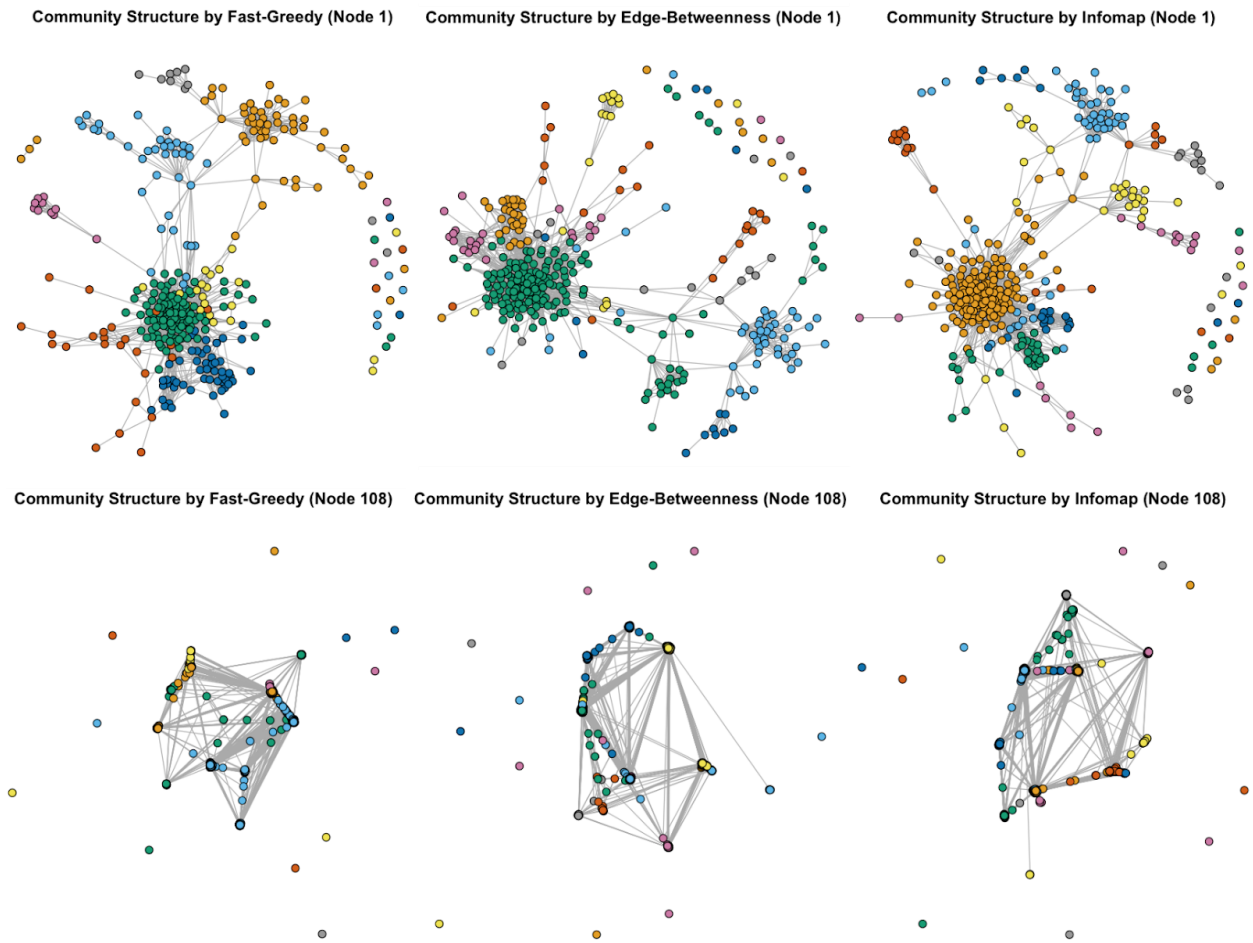
Q10

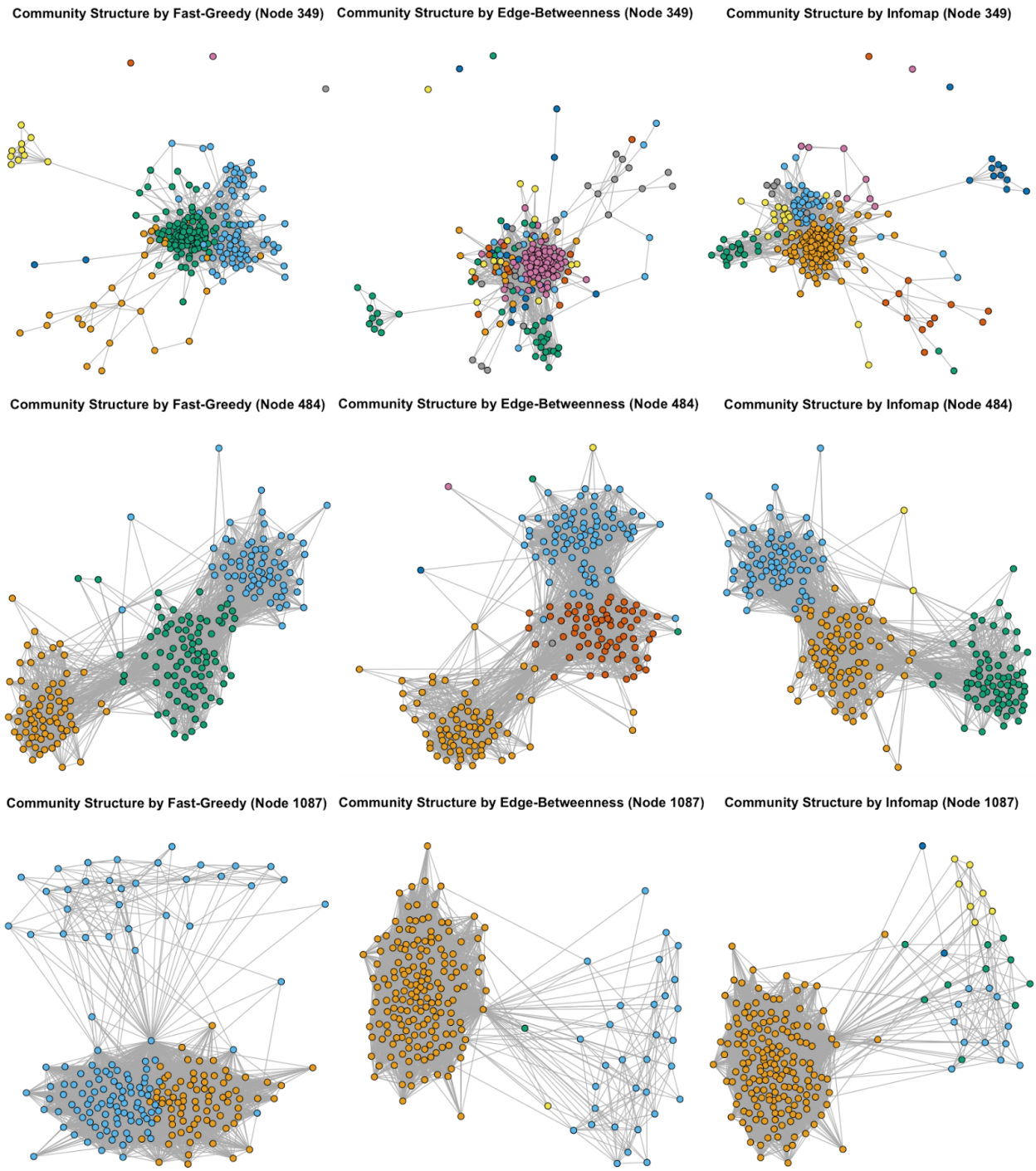
To explore the effect on the community structure of a personalized network when its core node is removed, we run same algorithms on the modified personalized network of the same five nodes. Compare with the previous results, we find that almost every modularity scores have increased when we do community detection on modified personalized networks, meaning the

detected community structures are better. This result is consistent with the intuition that since the core node is connected to all the other nodes in the personalized network, thus it may connect two different communities and then trick the algorithms to get worse community detection result. As a result, removing the core node improves the performance of community detection on its personalized network. It is also worth mentioning that removing the core node might result in some outliers which are not connected to any other nodes.

Modularity scores (modified personalized network)

Core node	Fast-Greedy	Edge-Betweenness	Infomap
Node 1	0.442	0.416	0.418
Node 108	0.458	0.521	0.520
Node 349	0.246	0.151	0.245
Node 484	0.534	0.515	0.543
Node 1087	0.148	0.032	0.027





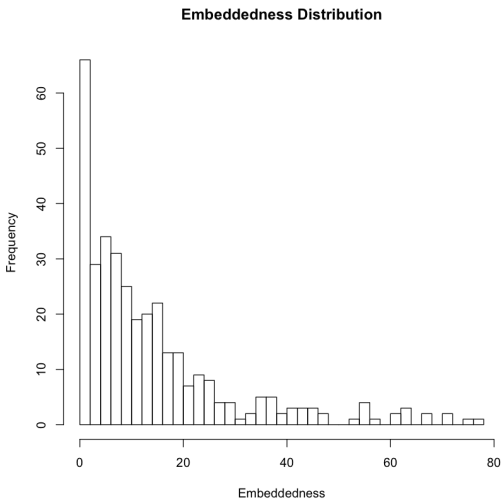
1.3.3 Characteristic of nodes in the personalized network

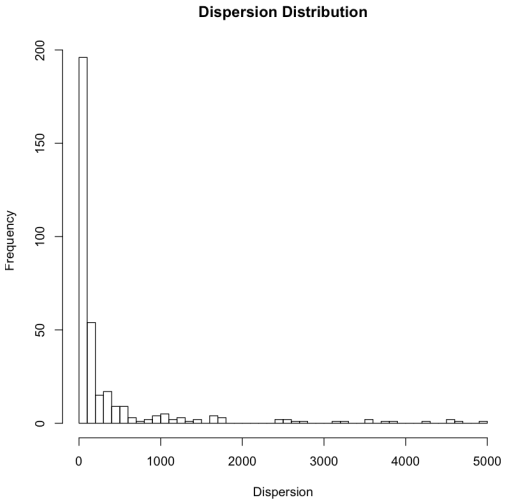
Q11

embeddedness = node's degree - 1 (mutual friends excluding the node itself)

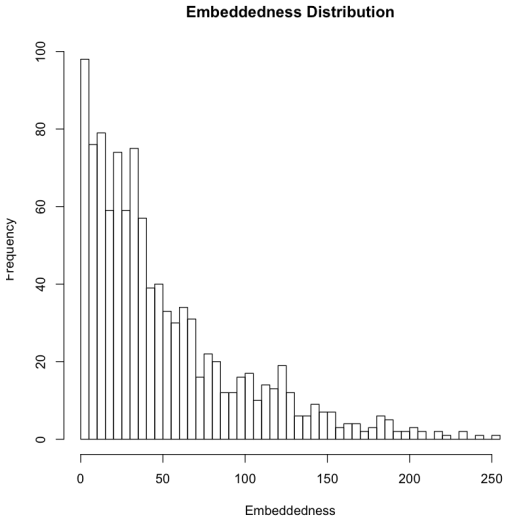
Q12

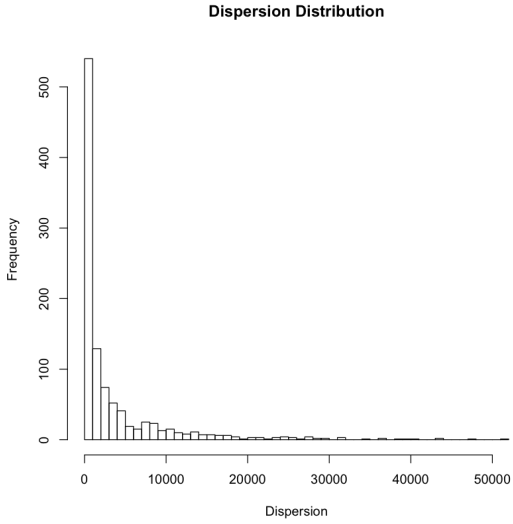
When corenode id = 1, embeddedness and dispersion distribution plots are shown as below.



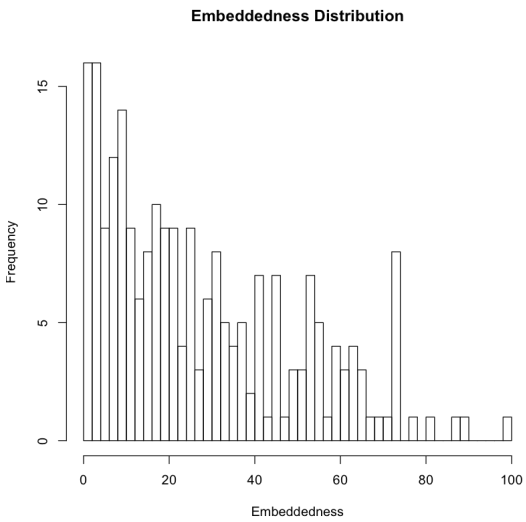


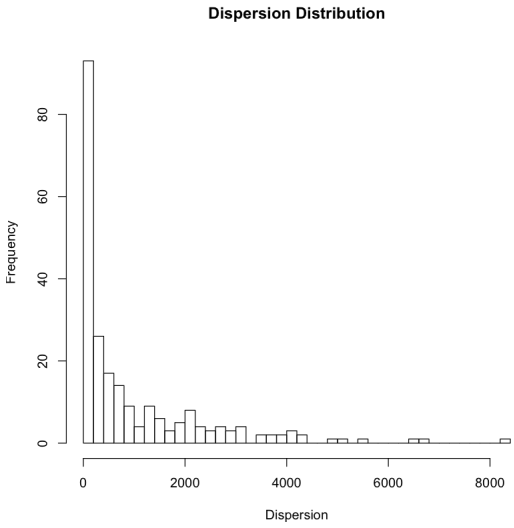
When corenode id = 108, embeddedness and dispersion distribution plots are shown as below.



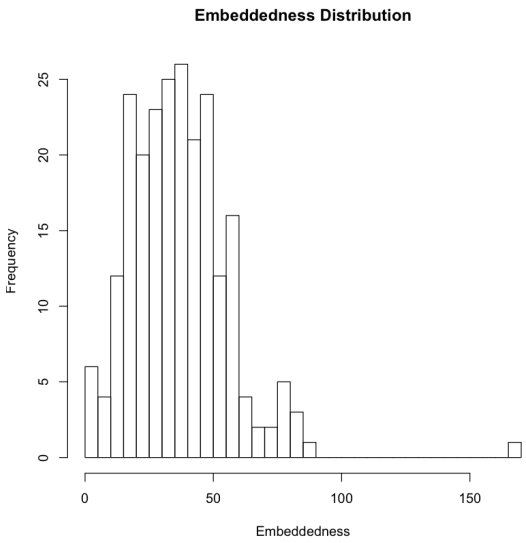


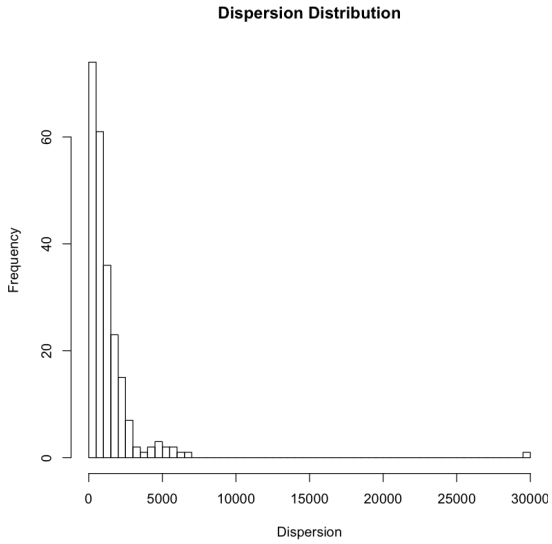
When corenode id = 349, embeddedness and dispersion distribution plots are shown as below.



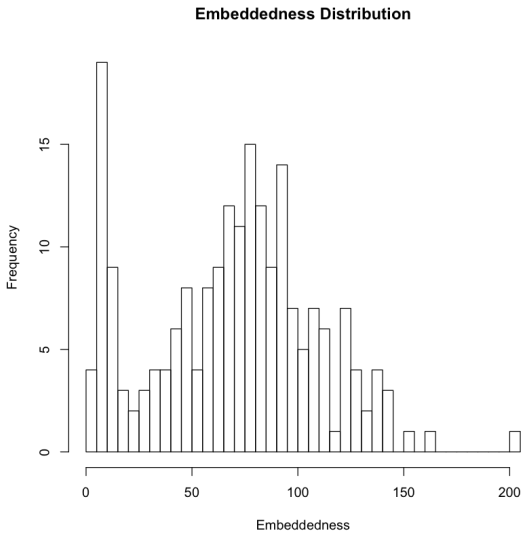


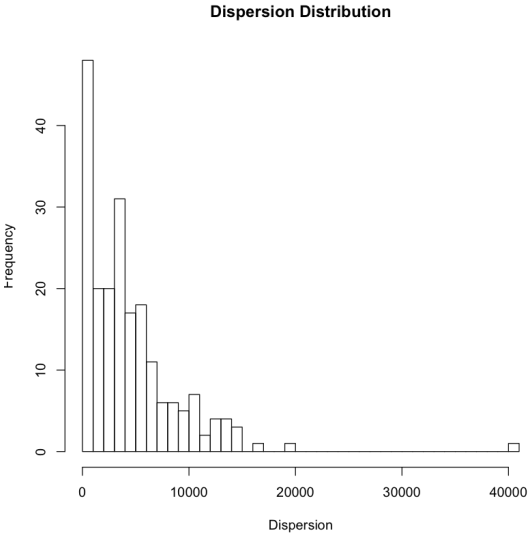
When corenode id = 484, embeddedness and dispersion distribution plots are shown as below.





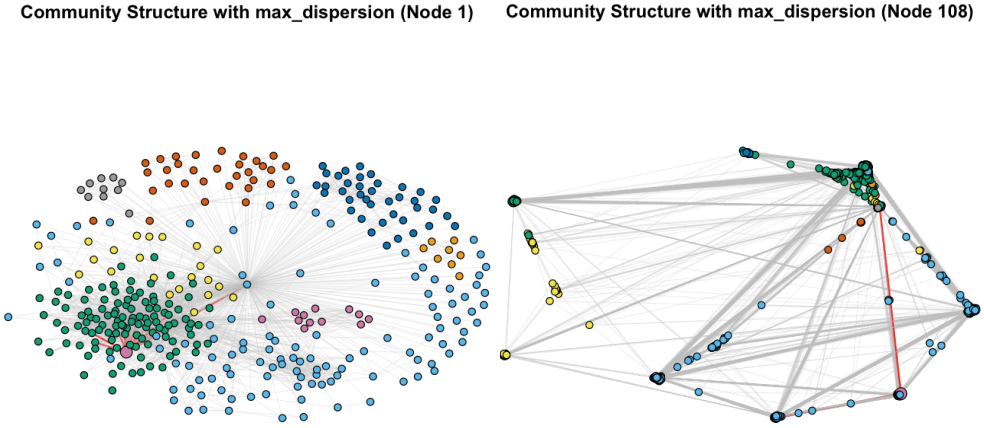
When corenode id = 1087, embeddedness and dispersion distribution plots are shown as below.



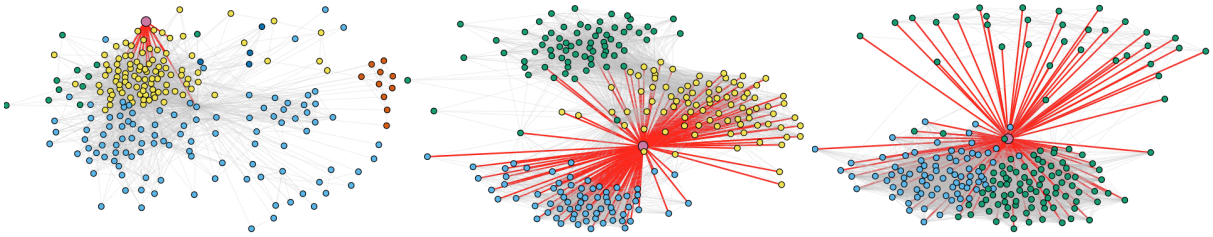


Q13

When corenode id=1,108,349,484,1087, below are community network (max dispersion) with highlight node id=56,1888,376,107,107. (inf = diameter+constant, constant = 2)



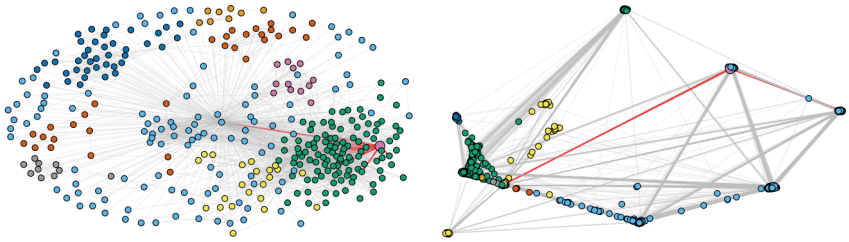
Community Structure with max_dispersion (Node 349) Community Structure with max_dispersion (Node 484) Community Structure with max_dispersion (Node 1087)



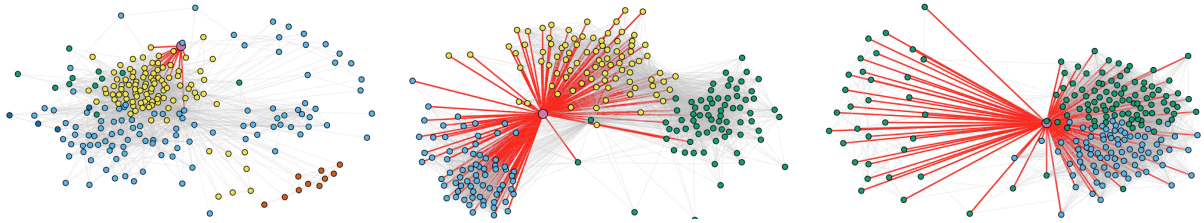
Q14

When corenode id=1,108,349,484,1087, below are community network (max embeddedness) with highlight node id=56,1888,376,107,107.

Community Structure with max_embeddedness (Node 1) Community Structure with max_embeddedness (Node 108)

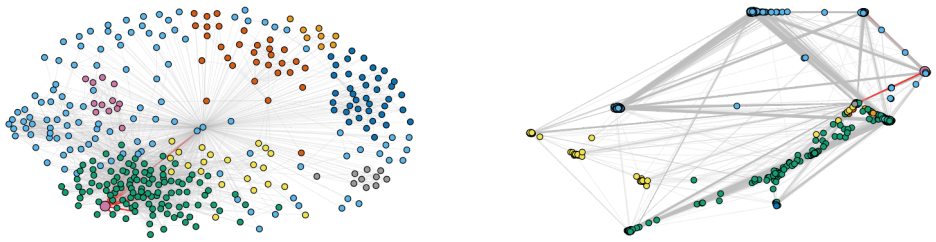


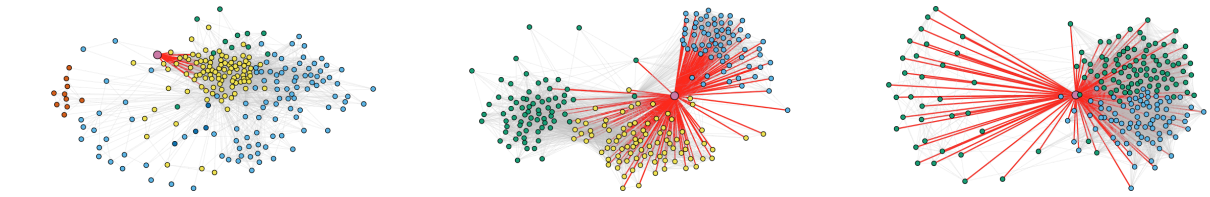
Community Structure with max_embeddedness (Node 349) Community Structure with max_embeddedness (Node 484) Community Structure with max_embeddedness (Node 1087)



When corenode id=1,108,349,484,1087, below are community network (max ratio of dispersion/embeddedness) with highlight node id=56,1888,376,107,107.

Community Structure with max_dispersion_embeddedness (Node 1) Community Structure with max_dispersion_embeddedness (Node 108





Q15

In each plot, the largest pink node is the highlight node, the edges between highlight node and its neighbors are red, and all the other edges are grey. For corenode id=1,108,349,484,1087, the highlight node id=56,1888,376,107,107 are the same for all three plots (max_dispersion, max_embeddedness, max_dispersion_embeddedness). (for dispersion, inf = diameter+constant where constant = 2)

Embeddedness is the number of mutual friends and captures how much the two partners' social circles overlap. Embeddedness is a natural predictor for identifying a user's partner. However, embeddedness has limitations because many individuals have large clusters of friends from well-defined communities (company, college) and people within these clusters know each other with high embeddedness, even though they do not necessarily correspond to particularly strong ties. For example, students in the same class have many mutual friends and thus have high embeddedness value but they are strongly connected, say couple.

Notice that the highlighted node in Q13 has the maximum embeddedness and thus has the most mutual friends with the core node. Additionally, the highlighted node is in the community whose size is the largest among all the communities generated. It is supported by Q13 plots because there are many red edges coming out of the highlighted node. More importantly, for corenode id=1,108,349,484,1087, the highlight node with maximum embeddedness is the same as the node with maximum dispersion. This makes sense because the dispersion is the sum of all distance between each pair of mutual friends. In other words, the dispersion is positively proportional to mutual friends, which defines the embeddedness.

Dispersion is the measurement of differing members of a population. Dispersion is a better measurement than embeddedness because it looks not just at the number of mutual friends of two people, but also at the network structure on these mutual friends. The larger the dispersion is, the more likely mutual friends do not know each other. For example, if A and B both graduate from UCLA and work at the same company, then they have many mutual friends from the same college and company but mutual classmates do not know mutual colleagues. In this way, mutual friends are dispersed and in A's network, B is the node with max dispersion.

In Q14 first 5 plots, the highlighted node has the maximum dispersion and thus has many mutual friends with the corenode or the mutual friends spread out and not know each other. It is

supported by Q14 plots especially those with corenode id=484,1087 because the common friends are in different color and separated in different communities.

The ratio of dispersion to embeddedness is normalization of dispersion on embeddedness. It is positively correlated to dispersion and negatively correlated to embeddedness. Dispersion is relatively large when the embeddedness is large, although most mutual friends know each other. High ratio of dispersion to embeddedness indicates that the node have mutual friends from different contexts regardless of number of mutual friends. The ratio is a normalization that combines dispersion and embeddedness, which is a more concise indicator than single embeddedness or dispersion.

In Q 14 last 5 plots, the highlighted node has the maximum ratio of dispersion to embeddedness and thus has large dispersion value but small embeddedness. It is supported by Q14 plots because the highlighted node with corenode id=1,108,349 have very few mutual friends with the corenode, which leads to small embeddedness, and the highlighted node with corenode id=484,1087 have mutual friends separated in different communities, which results in large dispersion. Furthermore, for corenode id=1,108,349,484,1087, the highlight node with maximum ratio is the same as the node with maximum embeddedness. This shows the normalization of dispersion by embeddedness and the separation of mutual friends.

In conclusion, embeddedness is a natural predictor for identifying a user's partner but has limitation. Dispersion is a better measurement than embeddedness because it looks not just at the number of mutual friends of two people, but also at the network structure on these mutual friends. The ratio of dispersion to embeddedness is better because normalized dispersion and applying dispersion recursively lead to increased performance.

1.4 Friend recommendation in personalized networks

1.4.3 Creating the list of users

Q16
Nr is 31 53 75 90 93 102 118 133 134 136 137 and |Nr| is 11.

1.4.4 Average accuracy of friend recommendation algorithm

Q17
Based on the average accuracy values, Adamic Admar measure algorithm is the best.

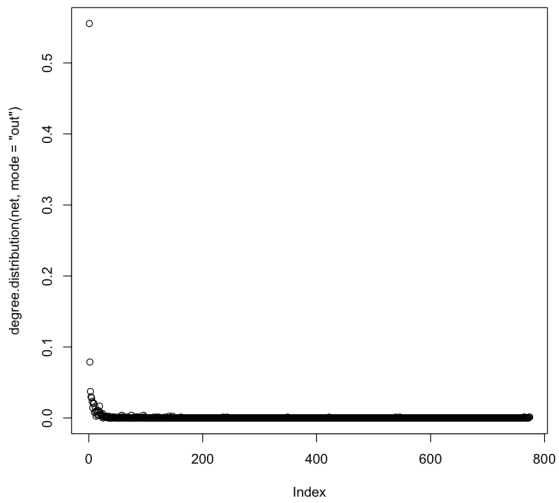
Common Neighbors measure	0.828857733175915
Jaccard measure	0.820151843106389
Adamic Adar measure	0.829369670733307

The Common Neighbors measure between nodes i and j is equal to the number of common neighbors between nodes i and j. The major weakness of the common-neighbor measure is that it does not account for the relative number of common neighbors between them as compared to the number of other connections. For example, if node i is a very famous person and knows lots of people from different communities, then the mutual friends of node i and j may not know each other. But it has good performance on this network because there is no need to eliminate the effect of celebrities.
The Jaccard measure adjusts much better to the variations in the degrees of the nodes between which the link prediction is measured. Also, the Jaccard measure works well in the network with popular public figures. Since there is no famous person in the network, the Jaccard measure has the worst performance.
The Adamic-Adar measure is designed to account for the varying importance of the different common neighbors. It can be viewed as a weighted version of the common-neighbor measure, where the weight of a common neighbor is a decreasing function of its node degree. Although there is no celebrity in the network, the Adamic-Adar measure fixes the problem of the Common Neighbors measure and thus has the best performance.

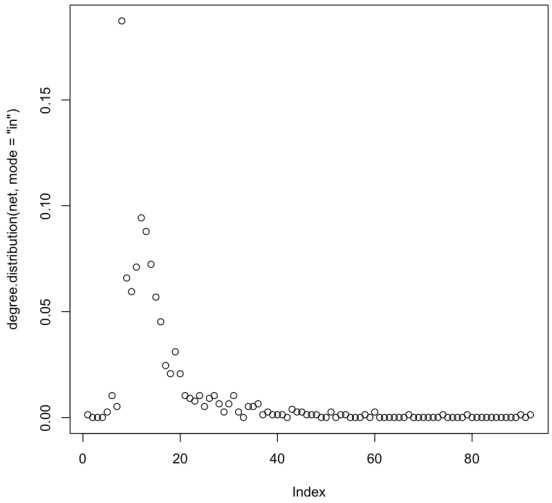
Part 2: Google+ Network

Q18: By calculating the total amount of egoNodes in the dataset we know it has 132 personal networks.

Q19:
For node id: 109327480479767108490

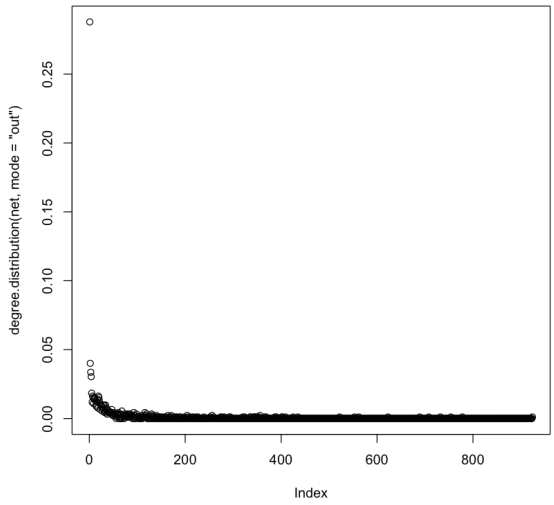


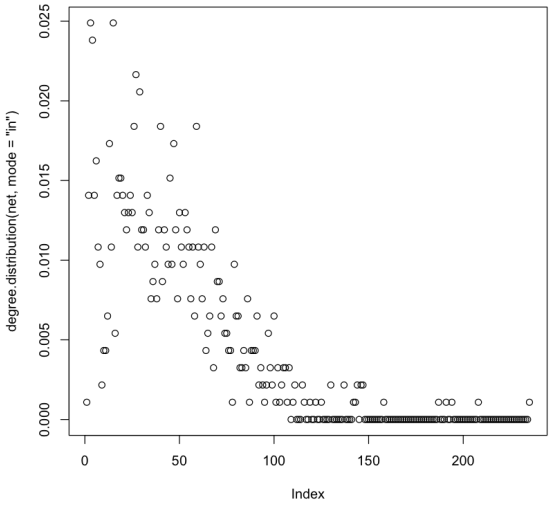
Out degree distribution



In degree distribution

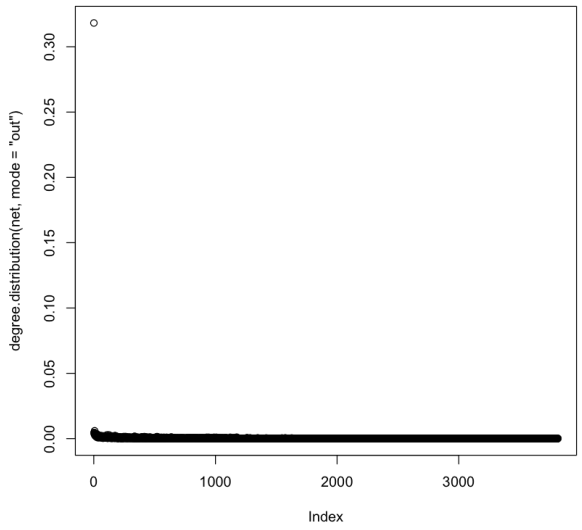
For node id: 115625564993990145546

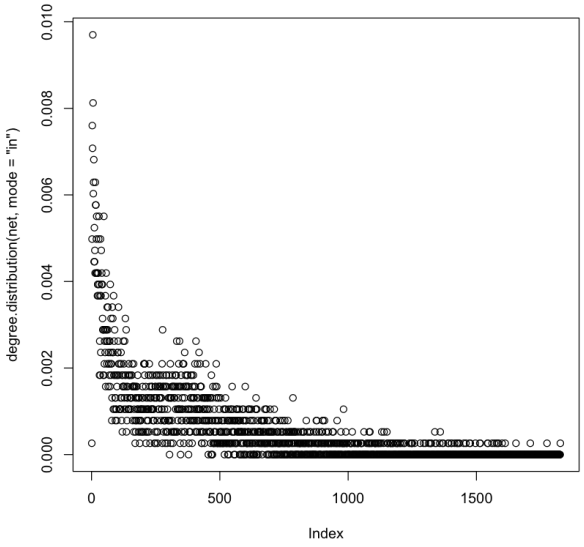




Out degree distribution In degree distribution

For node id: 10137396127944380674 4



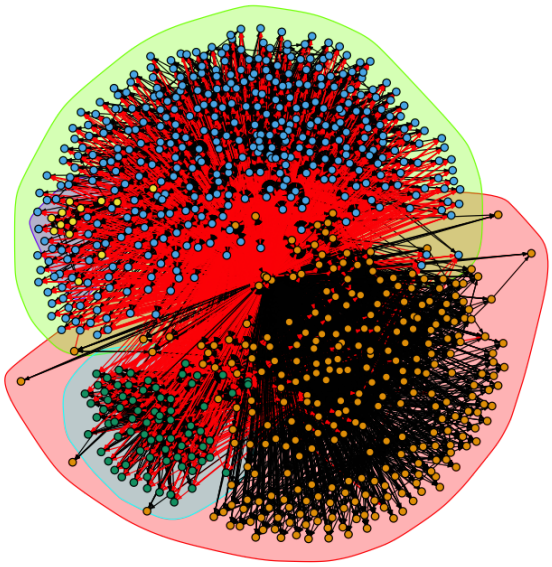


Out degree distribution In degree distribution

Q 20:

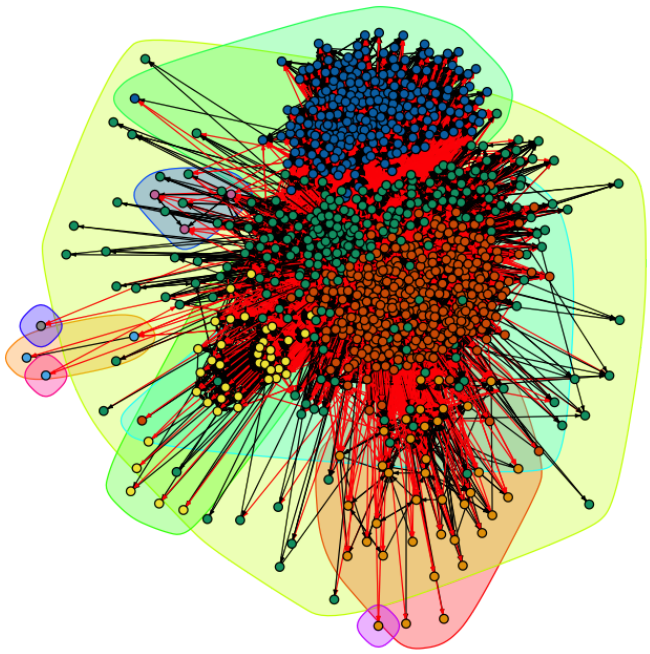
For node id: 109327480479767108490

Modularity score: 0.252765387296677

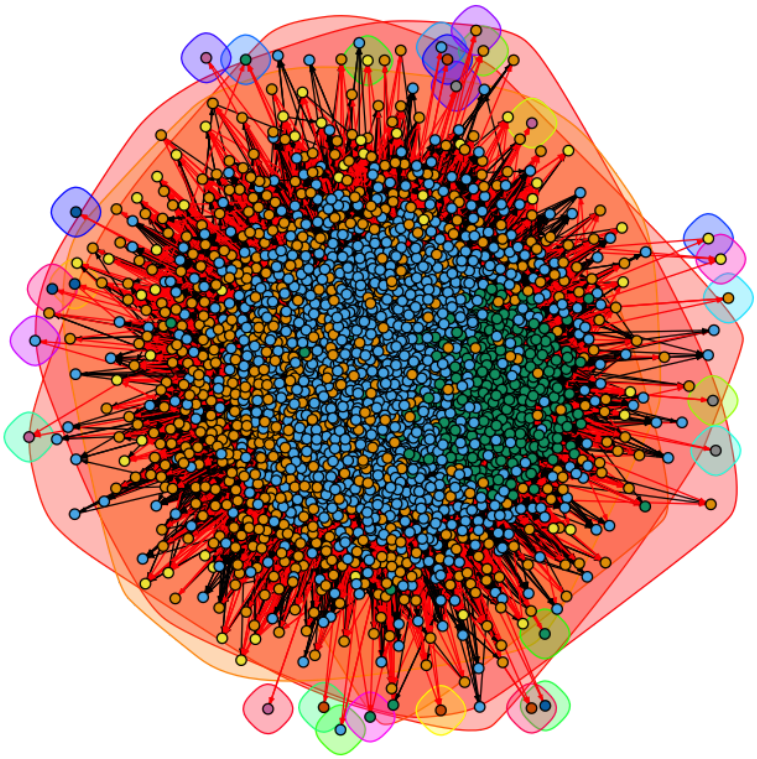


[1] "The 1 th community has size of 288"
[1] "The 2 th community has size of 397"
[1] "The 3 th community has size of 76"
[1] "The 4 th community has size of 13"

For node id: 11562556499399014554 6
Modularity score: 0.319472551345825
[1] "The 1 th community has size of 37"
[1] "The 2 th community has size of 2"
[1] "The 3 th community has size of 256"
[1] "The 4 th community has size of 40"
[1] "The 5 th community has size of 233"
[1] "The 6 th community has size of 350"
[1] "The 7 th community has size of 3"
[1] "The 8 th community has size of 1"
[1] "The 9 th community has size of 1"
[1] "The 10 th community has size of 1"



For node id: 101373961279443806744
Modularity score: 0.191090270876884
[1] "The 1 th community has size of 980"
[1] "The 2 th community has size of 2026"
[1] "The 3 th community has size of 733"
[1] "The 4 th community has size of 49"
[1] "The 5th - 31th community has size of 1"
[1] "The 6 th community has size of 1"
...
[1] "The 31 th community has size of 1"



Q 21: **A clustering result satisfies homogeneity if all of its communities contain only data points which are members of a single circle.**

A clustering result satisfies completeness if all the data points that are members of a given circle are elements of the same communities.

Q22:

Method 1: when calculating the H(C|K) and H(K|C), simply iterate the circles and communities and manually calculate the log base of Cji/ai and Cji/bj and multiplies it with Cji/N¶

Result:

[1] "For node 109327480479767108490 we calculate the result as below: "
[1] "N: 764"
[1] "H_C: 1.51595415381635"
[1] "H_K: 1.45020885763114"

[1] "H_CK: 0.224535374489447"
[1] "H_KC: 0.971822786318783"
[1] "h: 0.851885115440867"
[1] "c: 0.329873913536689"

[1] "For node 115625564993990145546 we calculate the result as below: "
[1] "N: 727"
[1] "H_C: 12.2126251379311"
[1] "H_K: 1.55982884141566"
[1] "H_CK: 6.69385826353235"
[1] "H_KC: 6.90062406548138"
[1] "h: 0.451890303032235"
[1] "c: -3.4239623491117"

[1] "For node 101373961279443806744 we calculate the result as below: "
[1] "N: 521"
[1] "H_C: 0.554456498907601"
[1] "H_K: 0.711725628565447"
[1] "H_CK: 0.552312578092445"
[1] "H_KC: 1.78233064073994"
[1] "h: 0.00386670698130509"
[1] "c: -1.5042383879479"

Method2: when calculating the H(C|K) and H(K|C), transfer it to:

$$\left(\sum_{j=1}^{K}\frac{bj}{N}*\sum_{i=1}^{C}\frac{Cji}{bj}*\log(\frac{Cji}{bj})\right)$$

So we can use entropy function to calculate the

$$\sum_{i=1}^{C}\frac{Cji}{bj}*\log(\frac{Cji}{bj})$$

similarly for H(K|C) we will use the entropy function to calculate

$$\sum_{j=1}^{K}\frac{Cji}{ai}*\log(\frac{Cji}{ai})$$

So that we can get:

$$\left(\sum_{i=1}^{C}\frac{ai}{N}*\sum_{j=1}^{K}\frac{Cji}{ai}*\log(\frac{Cji}{ai})\right)$$

The result is slightly different as the R entropy function might work differently :

[1] "For node 10932748047976710849 0 we calculate the result as below: "
[1] "N: 764"
[1] "H_C: 1.0930883567809"
[1] "H_K: 1.00520818089008"
[1] "H_CK: 0.414829997041996"
[1] "H_KC: 0.673616224340775"
[1] "h: 0.620497286913152"
[1] "c: 0.329873913536689"

[1] "For node 11562556499399014554 6 we calculate the result as below: "
[1] "N: 727"
[1] "H_C: 3.13716655108452"
[1] "H_K: 1.08119096358335"
[1] "H_CK: 2.60014097490328"
[1] "H_KC: 4.78314811509253"
[1] "h: 0.171181723200379"
[1] "c: -3.4239623491117"

[1] "For node 10137396127944380674 4 we calculate the result as below: "
[1] "N: 521"
[1] "H_C: 1.09790227426011"
[1] "H_K: 0.493330612772394"
[1] "H_CK: 1.09748131153569"
[1] "H_KC: 1.23541745845449"
[1] "h: 0.000383424585495007"
[1] "c: -1.5042383879479"

We can tell from the homogeneity and completeness result that all 3 personal networks we chose have a higher homogeneity score than completeness. For the first personal network, it has the highest homogeneity and completeness and the difference is not so huge since it has only 3 circles and 4 communities so nodes in the communities will have a higher chance to be from a single circle. And the gap between the unique nodes amount in all circles and the sum of all circles' nodes amount is not huge (764 vs 1095) which indicates that the overlapping between circles is not very serious.

However, for the second network, the homogeneity and completeness score dropped dramatically. First of all, the second network has 31 circles and 10 communities. And the

gap between the unique nodes amount in all circles and the sum of all circles' nodes amount is 6467 vs 727, which means that the 31 circles for this egoNode have a lot of overlaps. In this case, we can tell that the probability that a community only contains data points which are members of a single circle is low. And by printing out the ai and bi, we can find out that the size of each community is smaller than the size of most circles, so it is relatively unlikely that a circle only contains nodes of a single community, which leads to the negative value of completeness.

ai represents the length of each circle (31)
[1] "ai: "
[1] 6 9 169 276 325 93 73 338 46 62 338 255 485 7 260 363 188 327 314
[20] 48 489 276 79 373 67 10 354 73 300 362 102

bi represents the length of each community (10)
[1] "bi: "
[1] 1 0 146 3 231 345 1 0 0 0

For the last network, it has 3 circles but 31 communities (although most of the communities only have 1 node). The homogeneity and completeness score are still low due to similar reasons as network 2. And since there are 27 communities that only having 1 node and the intersection of such communities and circles are empty, it is very unlikely these communities contain only data points which are members of a single circle, so the homogeneity score is even smaller than the network 2.