HW 3 Report

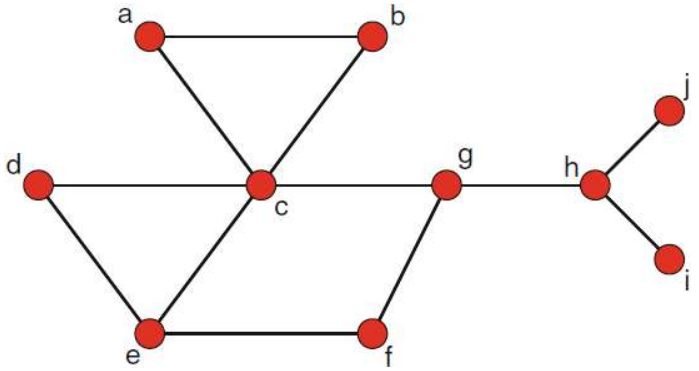
Yongchao Qiao

Introduction

There are many tools for viewing, analyzing and assessing the properties of individual nodes and ties. For prominence, a member (actor) is prominent if the related ties (connections) make the actor visible to the other members in the whole network. Generally, centrality and prestige will be considered for non-directed and directed networks respectively. A node with more direct ties is more prominent than nodes with fewer or no ties which is the general idea of degree centrality. So the degree centrality is simply the degree of each node. As for closeness centrality, it means that nodes are more prominent to the extent they are close to all other nodes in the network. Finally, betweenness centrality measures the extent that a node sits 'between' pairs of other nodes in the network, such that a path between the other nodes has to go through that node.

A network-level conclusion can be got by centralization based on the centrality function. In the graph level, the cutpoint is a node if dropped, would increase the number of components in the network. Similarly, the bridge is an edge, if removed, would split one component into two. Both Cliques and k-Cores can identify network subgroups. Typically, a clique is a maximally complete subgraph which means a subset of nodes that have all possible ties among them. As for k-Cores, it is a maximal subgraph where each vertex is connected to at least k other vertices in the subgraph Both cliques and k-cores are subgroup identification that rely completely on the pattern of internal ties (within group). Community detection algorithms have been developed for complex subgroup identification. For example, modularity is the extent to which nodes show clustering patterns with greater density within each cluster and less density among different clusters. The value of modularity is from -1 to 1 and the closer to 1, the more the network shows clustering with respect to the given node grouping. The interpretation of modularity is more meaningful when compared to another one.

I Verify the properties of the network manually



1. Degree centrality of node g

Firstly, this is a undirected graph and there are 3 edges connecting with node g. Then the degree centrality of node g is 3 since the degree centrality is simply the degree of each node.

2. Closeness centrality of node g

The formula to calculate the closeness is:

$$\text{Closeness centrality(g)} = \frac{The\ number\ of\ total\ nodes\ -\ 1}{Total\ distances\ between\ node\ g\ and\ any\ oth\quad nodes\ in\ the\ graph}$$

Since the number of total nodes is 10; The distances between node g and any other nodes are:

$dist_{(a\text{-}g)} = 2$; $dist_{(b\text{-}g)} = 2$; $dist_{(c\text{-}g)} = 1$; $dist_{(d\text{-}g)} = 2$; $dist_{(e\text{-}g)} = 2$; $dist_{(f\text{-}g)} = 1$; $dist_{(h\text{-}g)} = 1$; $dist_{(i\text{-}g)} = 2$; $dist_{(j\text{-}g)} = 2$.

Then the total distances between node g and any other nodes in the graph are:

Total distances = 2+2+1+2+2+1+1+2+2 = 15

Therefore, Closeness centrality(g) = $\frac{10-1}{15} = \frac{9}{15} = 0.6$

3. Betweenness centrality of node g

The formula to calculate the closeness is:

$$C_B(n_i) = \sum_{j<k} g_{jk}(n_i)/\ g_{jk};$$

where $g_{jk}(n_i)$ is the number of shortest paths between nodes *j* and *k* that contain node *i* and *i* is not equal to *j* and *k*.

Table 1 The ratio of shortest paths contains node g and all shortest paths for each paired node

nodes	a	b	c	d	e	f	h	i	j
a	/	0	0	0	0	1/2	1	1	1
b	/	/	0	0	0	1/2	1	1	1
c	/	/	/		0	1/2	1	1	1
d	/	/	/	/	0	0	1	1	1
e	/	/	/	/	/	0	1	1	1
f	/	/	/	/	/	/	1	1	1
h	/	/	/	/	/	/	/	0	0
i	/	/	/	/	/	/	/	/	0
j	/	/	/	/	/	/	/	/	/

Thus, Betweenness centrality(g) = 1/2+1+1+1+1/2+1+1+1+1/2+1+1+1+1+1+1+1+1+1+1+1+1+1+1=19.5

4. All 2-cores and 3-cores

Initially, we specify node a, b and c can be regarded as a 2-cores subgraph since all nodes are connected to at least 2 other nodes in this subgraph, then we need to identify whether this is the maximal 2-cores subgraph. Typically, we can add node d and e such that the graph containing node a, b, c, d and e can still be regarded as 2-cores subgraph since all nodes are connected to at least 2 other nodes in this subgraph. Finally, we add node f and g to expand this subgraph and found that the maximal subgraph is the graph containing node a, b, c, d, e, f and g since all nodes are connected to at least 2 other nodes in this subgraph and node h does not belong to this 2-cores subgraph because it only connects with one node, named g.

Typically, there is no 3-cores in this graph since we can not find a maximal subgraph such that all nodes are connected to at least 3 other nodes in this subgraph

Besides, node h, i and j are belonging to the 1-core subgraph since all nodes are connected to at least 1 other nodes in this subgraph.

Conclusion:

The 2-cores subgraph is the subgraph containing node a, b, c, d, e, f and g. However, there is no 3-cores subgraph.

II Apply R function

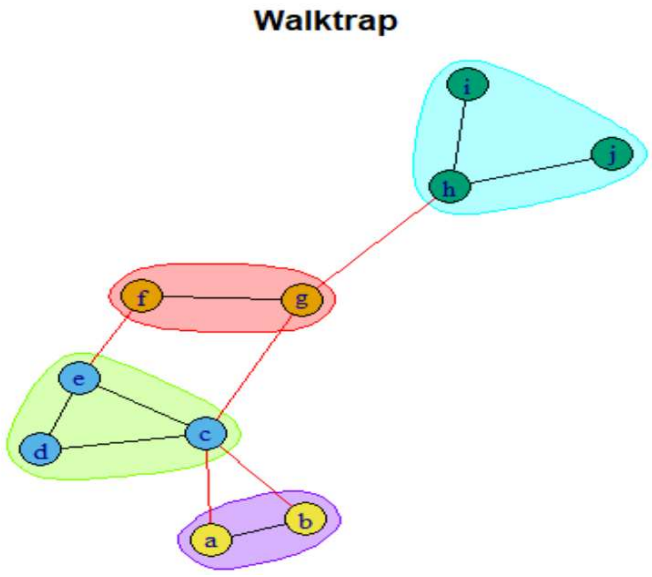
1. modularity function

```
gr = c(rep(1,7),rep(2,3))
modularity(inet, gr)

## [1] 0.2465278
```

Here, we calculate the modularity of this network by the gr which assign the nodes into 2 groups based on the coreness. Specifically, first 7 nodes belong to group 1 and the other 3 nodes belong to group 2. Finally, the result is 0.2465278, greater than 0, which generally shows that this network is clustering with respect to the grouping based on coreness.

2. cluster_walktrap & plot function

```
cw <- cluster_walktrap(inet)
modularity(cw)

## [1] 0.2951389
```



After applying the cluster_walktrap function, we can see the modularity is 0.2951389 which is closer to 1 and bigger than the modularity of the grouping based on coreness. Therefore, this network is more clustering with respect to the cluster_walktrap grouping. Also, the plot of cluster_walktrap grouping is showed above.

Appendix

```
library(UserNetR)
library(statnet)
library(RColorBrewer)

netmat = rbind(
               c(0,1,1,0,0,0,0,0,0,0),
               c(1,0,1,0,0,0,0,0,0,0),
               c(1,1,0,1,1,0,1,0,0,0),
               c(0,0,1,0,1,0,0,0,0,0),
               c(0,0,1,1,0,1,0,0,0,0),
               c(0,0,0,0,1,0,1,0,0,0),
               c(0,0,1,0,0,1,0,1,0,0),
               c(0,0,0,0,0,0,1,0,1,1),
               c(0,0,0,0,0,0,0,1,0,0),
               c(0,0,0,0,0,0,0,1,0,0))


net = network(netmat, matrix.type = "adjacency", directed = FALSE)  # Use adjac
ency matrix to creat the network

network.vertex.names(net)=c("a","b","c","d","e","f","g","h","i","j") # Set the
 vertex.names attribute
```

Centrality

```
cat("Degree Centrality\n")

print(degree(net, gmode="graph")[7]) # Calculate the degree centrality

cat("Closeness Centrality\n")

print(closeness(net,gmode = "graph")[7]) # Calculate the closeness centrality

cat("Betweenness Centrality\n")

print(betweenness(net,gmode = "graph")[7])# Calculate the betweenness centrali
ty
```

K-cores

```
detach(package:statnet,unload=TRUE)
library(igraph)
library(intergraph)

inet = asIgraph(net)  # Transfer the network to igraph
coreness=graph.coreness(inet) # Claculate the coreness to identify k-cores
coreness
```

Modularity

```
gr = c(rep(1,7),rep(2,3)) # Based on coreness we assign first 7 nodes into group 1  and the other 3 nodes into group 2

modularity(inet, gr) # calculate the modularity of nodes grouping by gr
```

```
## [1] 0.2465278
```

Cluster_walktrap

```
cw <- cluster_walktrap(inet) # Use function cluster_walktrap to group the internet
membership(cw) # Identify the membership of the result of cluster_walktrap grouping

modularity(cw) # calculate the modularity of nodes grouping by cluster_walktrap

plot(cw, inet, vertex.label=
 V(inet)$vertex.names, main="Walktrap") #plot the graph
```