assignment2

Giridhar Manoharan

3/6/2017

Question 1

The fourth dataset that I am using here is the Facebook social network downloaded from SNAP - Standford Network Analysis Project (http://snap.stanford.edu/data/egonets-Facebook.html). It consists of 10 anonymized egonets. The nodes represent users and the edges (or links) represent friendship between the two users. The edges are undirected.

To identify the important nodes in a network, the following centrality measures are used:

*__Degree centrality__ - It is the number of links incident upon a node. A higher value indicates that the node is more central or important.

*__Eigenvector centrality__ - It is a measure of the influence of a node in a network. It assigns relative scores to all nodes in the network based on the concept that connections to high-scoring nodes contribute more to the score of the node in question than equal connections to low-scoring nodes.

*__PageRank__ - It is a variant of eigenvector centrality. It uses links between nodes as a measure of importance. Each node is assigned a score based upon its number of in-coming links (in-degree). These links are also weighted depending on the relative score of its originating node. The result is that nodes with many in-coming links are influential and nods to which they are connected share some of that influence. The main difference to eignen centrality is that PageRank takes link direction and weight into account.

*__Klienberg's centrality__ - It is a variant of eigenvector centrality that works well for acyclic networks. Here, each vertex has two centralities, known as the authority score and the hub score. The authority score is derived from the incoming links and the hub score is derived from the outgoing links. Hub is a vertex that points to many important authorities while authority is vertex pointed to by many important hubs.

*__Betweeness centrality__ - It is a measure of centrality in a graph based on shortest paths. It quantifies the number of times a node acts as a bridge along the shortest path between two other nodes.

```
internetGraph = read_graph("../assignment1/as-22july06/as-22july06.gml", format = "gml")
politicalGraph = read_graph("../assignment1/polblogs/polblogs.gml", format = "gml")
neuralNetGraph = read_graph("../assignment1/celegansneural/celegansneural.gml", format = "gml")
#facebookGraph = read_graph("./assignment2/facebook/0.edges", directed = FALSE)
facebookGraph = read_graph("./facebook_combined.txt", directed = FALSE)

#problem 1
networks = data.frame(Network = character(4), Degree = integer(4), Eigenvector_centrality = double(4), PageRank
= double(4), Klienberg_authority_score = double(4), Klienber_hub_score = double(4), Betweeness_centrality = dou
ble(4), stringsAsFactors = FALSE)

#internet
networks$Network[1] = "Internet"
networks$Degree[1] = which.max(degree(internetGraph))
networks$Eigenvector_centrality[1] = which.max(eigen_centrality(internetGraph)$vector)
networks$PageRank[1] = which.max(page_rank(internetGraph)$vector)
networks$Klienberg_authority_score[1] = which.max(authority_score(internetGraph)$vector)
networks$Klienber_hub_score[1] = which.max(hub_score(internetGraph)$vector)
networks$Betweeness_centrality[1] = which.max(betweenness(internetGraph))

#political
networks$Network[2] = "Political blogs"
networks$Degree[2] = which.max(degree(politicalGraph))
networks$Eigenvector_centrality[2] = which.max(eigen_centrality(politicalGraph)$vector)
networks$PageRank[2] = which.max(page_rank(politicalGraph)$vector)
networks$Klienberg_authority_score[2] = which.max(authority_score(politicalGraph)$vector)
networks$Klienber_hub_score[2] = which.max(hub_score(politicalGraph)$vector)
networks$Betweeness_centrality[2] = which.max(betweenness(politicalGraph))

#neural net
networks$Network[3] = "Neural network"
networks$Degree[3] = which.max(degree(neuralNetGraph))
networks$Eigenvector_centrality[3] = which.max(eigen_centrality(neuralNetGraph)$vector)
networks$PageRank[3] = which.max(page_rank(neuralNetGraph)$vector)
networks$Klienberg_authority_score[3] = which.max(authority_score(neuralNetGraph)$vector)
networks$Klienber_hub_score[3] = which.max(hub_score(neuralNetGraph)$vector)
networks$Betweeness_centrality[3] = which.max(betweenness(neuralNetGraph))

#facebook
networks$Network[4] = "Facebook"
networks$Degree[4] = which.max(degree(facebookGraph))
networks$Eigenvector_centrality[4] = which.max(eigen_centrality(facebookGraph)$vector)
networks$PageRank[4] = which.max(page_rank(facebookGraph)$vector)
networks$Klienberg_authority_score[4] = which.max(authority_score(facebookGraph)$vector)
networks$Klienber_hub_score[4] = which.max(hub_score(facebookGraph)$vector)
networks$Betweeness_centrality[4] = which.max(betweenness(facebookGraph))

kable(networks, caption = "Important nodes based on centrality measures", format = "markdown")
```

Network	Degree	Eigenvector_centrality	PageRank	Klienberg_authority_score	Klienber_hub_score	Betweeness_centrality
Internet	4	4	4	4	4	4
Political	855	55	155	155	512	855

blogs

Network	Degree	Eigenvector_centrality	PageRank	Klienberg_authority_score	Klienber_hub_score	Betweeness_centrality
Neural network	45	13	45	45	126	178
Facebook	108	1913	3438	1913	1913	108

Observations

Internet - Node 4 has the highest value in all of the centrality measures. So, Node 4 is the important node in this network.

Political blogs - Node 155 has the highest PageRank score and Klienberg authority score while Node 855 has the highest degree centrality and betweeness centrality scores. So, both Node 155 and 855 are important nodes in this network.

Neural network - Node 45 has the highest degree centrality, pagerank and klienberg authority scores. So, we can say Node 45 is the important node in this network.

Facebook - Node 1913 has the highest eigenvector centrality, klienberg authority and klienberg hub scores. So, Node 1913 is the important node in this network. Node 108 has highest degree centrality and betweeness centrality scores. So, we can consider Node 108 to be the second important node in this network.

Question 2 a,b

```
erdosRenyi20 = erdos.renyi.game(20, 80, type = "gnm")
barabasi20 = barabasi.game(n=20, m=5, directed=FALSE)
erdosRenyi40 = erdos.renyi.game(40, 120, type = "gnm")
barabasi40 = barabasi.game(n=40, m=3, directed = FALSE)

graphs = data.frame(Graph = character(4), Nodes = integer(4), Edges = integer(4), Min_degree = integer(4), Max_
degree = integer(4), Avg_path_length = numeric(4), Diameter = integer(4), Global_cluster_coeff = numeric(4), Se
cond_small_eigen_val = numeric(4), Largest_eigen_val = numeric(4), stringsAsFactors = FALSE)

graphs$Graph[1] = "Erdos-Renyi-20"
graphs$Nodes[1] = vcount(erdosRenyi20)
graphs$Edges[1] = ecount(erdosRenyi20)
graphs$Min_degree[1] = min(degree(erdosRenyi20))
graphs$Max_degree[1] = max(degree(erdosRenyi20))
graphs$Avg_path_length[1] = mean_distance(erdosRenyi20)
graphs$Diameter[1] = diameter(erdosRenyi20)
graphs$Global_cluster_coeff[1] = transitivity(erdosRenyi20, type = "globalundirected")
lap = graph.laplacian(erdosRenyi20)
eig = eigen(lap)
eig_val = eig$values
graphs$Second_small_eigen_val[1] = sort(eig_val, decreasing=F)[2]
graphs$Largest_eigen_val[1] = sort(eig_val, decreasing=T)[1]

graphs$Graph[2] = "Barabasi-20"
graphs$Nodes[2] = vcount(barabasi20)
graphs$Edges[2] = ecount(barabasi20)
graphs$Min_degree[2] = min(degree(barabasi20))
graphs$Max_degree[2] = max(degree(barabasi20))
graphs$Avg_path_length[2] = mean_distance(barabasi20)
graphs$Diameter[2] = diameter(barabasi20)
graphs$Global_cluster_coeff[2] = transitivity(barabasi20, type = "globalundirected")
lap = graph.laplacian(barabasi20)
eig = eigen(lap)
eig_val = eig$values
graphs$Second_small_eigen_val[2] = sort(eig_val, decreasing=F)[2]
graphs$Largest_eigen_val[2] = sort(eig_val, decreasing=T)[1]

graphs$Graph[3] = "Erdos-Renyi-40"
graphs$Nodes[3] = vcount(erdosRenyi40)
graphs$Edges[3] = ecount(erdosRenyi40)
graphs$Min_degree[3] = min(degree(erdosRenyi40))
graphs$Max_degree[3] = max(degree(erdosRenyi40))
graphs$Avg_path_length[3] = mean_distance(erdosRenyi40)
graphs$Diameter[3] = diameter(erdosRenyi40)
graphs$Global_cluster_coeff[3] = transitivity(erdosRenyi20, type = "globalundirected")
lap = graph.laplacian(erdosRenyi40)
eig = eigen(lap)
eig_val = eig$values
graphs$Second_small_eigen_val[3] = sort(eig_val, decreasing=F)[2]
graphs$Largest_eigen_val[3] = sort(eig_val, decreasing=T)[1]

graphs$Graph[4] = "Barabasi-40"
graphs$Nodes[4] = vcount(barabasi40)
graphs$Edges[4] = ecount(barabasi40)
graphs$Min_degree[4] = min(degree(barabasi40))
graphs$Max_degree[4] = max(degree(barabasi40))
graphs$Avg_path_length[4] = mean_distance(barabasi40)
graphs$Diameter[4] = diameter(barabasi40)
graphs$Global_cluster_coeff[4] = transitivity(barabasi40, type = "globalundirected")
lap = graph.laplacian(barabasi40)
eig = eigen(lap)
eig_val = eig$values
graphs$Second_small_eigen_val[4] = sort(eig_val, decreasing=F)[2]
graphs$Largest_eigen_val[4] = sort(eig_val, decreasing=T)[1]

kable(graphs, format = "markdown")
```

Graph	Nodes	Edges	Min_degree	Max_degree	Avg_path_length	Diameter	Global_cluster_coeff	Second_small_eigen_val	Largest_e
Erdos-Renyi-20	20	80	4	13	1.589474	3	0.3662207	3.485873	
Barabasi-20	20	85	5	16	1.552632	2	0.5280313	3.816537	
Erdos-Renyi-40	40	120	2	10	2.208974	4	0.3662207	1.221607	
Barabasi-40	40	114	3	17	2.211539	4	0.2118100	1.269410	

Observations

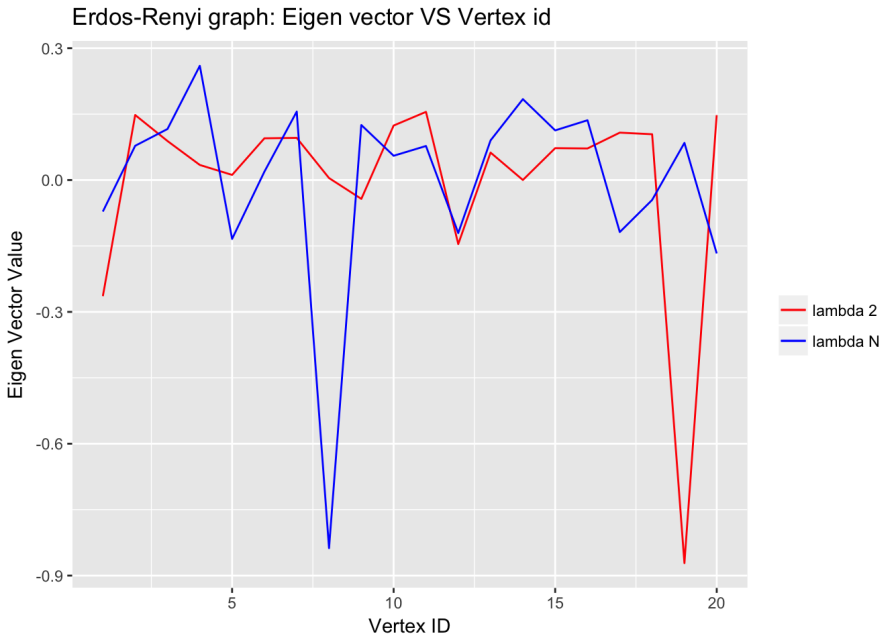
The number of links is almost same for both Erdos-Renyi and Barabasi. Barabasi graphs has higher minimum degree and maximum degree values compared to Erdos-Renyi graphs. Average path length and diameter is almost same for Erdos-Renyi and Barabasi graphs. Global

clustering coefficient of Barabasi 40 node graph is very less compared to other graphs indicating that nodes are less clustered in this network compared to other networks. Second smallest eigen value has a lower value for 40 node graphs compared to 20 node graphs. Barabasi graphs have a higher largest eigen value than Erdos-Renyi graphs.

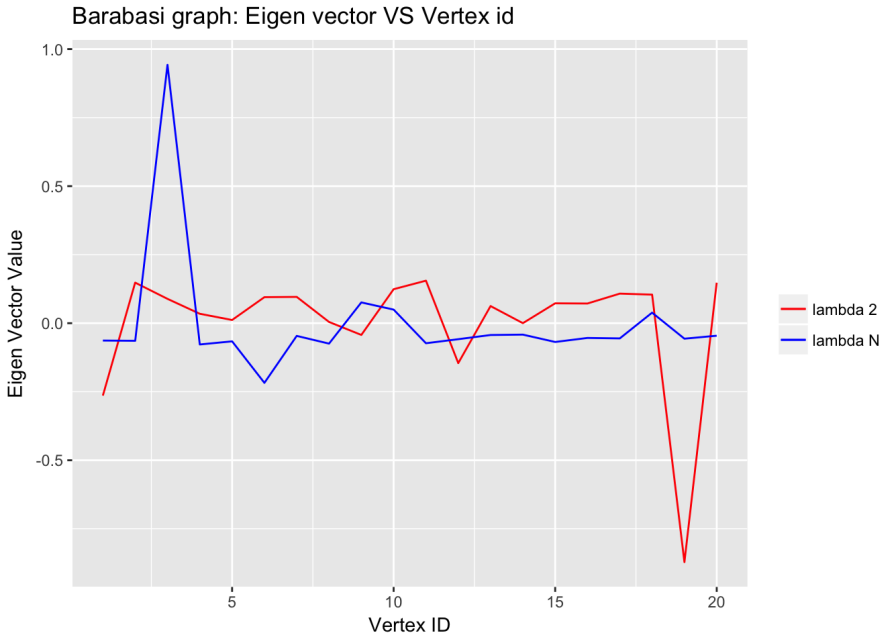
Question 2 c

```
erdos_g = data.frame(x = c(1:20), lambda2 = eigen(graph.laplacian(erdosRenyi20))$vectors[,19], lambdaN = eigen(
graph.laplacian(erdosRenyi20))$vectors[,which.max(eigen(graph.laplacian(erdosRenyi20))$values)])
barabasi_g = data.frame(x = c(1:20), lambda2 = eigen(graph.laplacian(erdosRenyi20))$vectors[,19], lambdaN = eig
en(graph.laplacian(barabasi20))$vectors[,which.max(eigen(graph.laplacian(barabasi20))$values)])

ggplot(erdos_g, aes(x = x)) +
  geom_line(aes(y = lambda2, colour = "lambda 2")) +
  geom_line(aes(y = lambdaN, colour="lambda N")) +
  scale_colour_manual("",
                      breaks = c("lambda 2", "lambda N"),
                      values = c("red", "blue")) +
  xlab("Vertex ID") + ylab("Eigen Vector Value") + labs(title = "Erdos-Renyi graph: Eigen vector VS Vertex id")
```




```
ggplot(barabasi_g, aes(x = x)) +
  geom_line(aes(y = lambda2, colour = "lambda 2")) +
  geom_line(aes(y = lambdaN, colour="lambda N")) +
  scale_colour_manual("",
                      breaks = c("lambda 2", "lambda N"),
                      values = c("red", "blue")) +
  xlab("Vertex ID") + ylab("Eigen Vector Value") + labs(title = "Barabasi graph: Eigen vector VS Vertex id")
```



Observations

From the above “Eigen vector value VS Vertex-id” plots we can say that, the eigen vector values of both the graphs are erratic around the zero

value line. Most of the eigen vector values (lambda_2 and lambda_N) of Barabasi graph are more close to zero value line than the eigen vector values in Erdos-Renyi graph. Both the graphs have one or two eigen vector values deviating very much away from the zero value line.

Question 4

PageRank in Social Networks:

It alwayes amazed me how facebook finds “people you may know” profiles who were actually my old friends or friends with common interests. The notion that PageRank helps to solve this problem intrigued me. I was fascinated to know that PageRank also helps to find the social status and power of a person.

In essence, PageRank servers three purposes in a social network, where the nodes are people and the edges are some type of social relationship. First, it helps to solve link prediction problems to find individuals that will become friends soon. Second, it helps to evaluate the centrality of the people involved to estimate thier social status and power. Thrid, it helps to evaluate the potential influence of a node on the opinions of the network.

PageRank in Recommender systems:

I used to wonder how amazon recommends me products that I have searched on the web wanting to buy, and netflix recommends me movies that I am willing to watch. PageRank helps in building such good recommender systems too. A recommender system attempts to predict what its users will do based on their past behavior. Localized PageRank helps to score potential predictions by recommender systems and thus aiding them to recommend better products or movies for users.

PageRank in Linux Kernel:

The use of PageRank in operating systems like Linux kernel demonstrates the wide application scope of this algorithm. It is used to analyze the functions used in Linux. The kernel call graph is a network that represents dependencies between functions. PageRank and reverse PageRank, as centrality scores produce an ordering of the most important functions in linux. Further, the distribution of PageRank and reverse PageRank scores is used to characterize the properties of a software system.

Problem 3: Show that the laplacian of the star graph on n vertices (the graph with the edge set $\{(1,u) : 2 \leq u \leq n\}$) has eigenvalue 0 with multiplicity 1, eiger value 1 with multiplicity $n-2$, and eigenvalue n with multiplicity 1.

Lemma 1: Let $G=(V,E)$ be a graph and let $0=\lambda_1 \leq \lambda_2 < \ldots < \lambda_n$ be eiger values of its laplacian matrix. Then $\lambda_2 > 0$ if and only if G is connected.

Proof:

If G is disconnected, then it can be described as the union of two graphs, G_1 and G_2. After suitally re-numbering the vertices, we can write

$$L_G = \begin{bmatrix} L_{G_1} & 0 \\ 0 & L_{G_2} \end{bmatrix}$$

So, L_G has at least two orthogonal eigervectors of eigervalue zero: $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ and $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$

where we have partitioned the vectors. On the other hand, assume that G is connected and that x is an eigervector of L_G of eigervalue 0.

As,

$$L_G x = 0$$

we have,

$$x^T L_G x = \sum_{(u,v) \in E} (x(u) - x(v))^2 = 0$$

Thus for each pair of vertices (u,v) connected by an edge, we have $x(u) = x(v)$. As every pair of vertices u and v are connected by a path, we may inductively apply this fact to show that $x(u) = x(v)$ for all vertices u and v. Thus x must be a constant vector. We conclude that the eigenspace of eigen value 0 has dimension 1.

Let S_n be star graph on n vertices, having edge set $\{(1,u) : 2 \leq u \leq n\}$

~~Lemma 2 : The Laplacian of comple graph has eigenvalue 0 with multiplicity 1 and n with multiplicity n-1.~~

Lemma 2 : Let $G = (V, E)$ be a graph and let i and j be vertices of degree one that are both connected to another vertex k. Then, the vector v is given by $v(u) = \begin{cases} 1 & u = i \\ -1 & u = j \\ 0 & \text{otherwise,} \end{cases}$

is an eigenvector of the Laplacian of G of eigenvalue 1.

Proof:

we can verify that $L_G \cdot v = v$

The existence of this eigen vector implies that $v(i) = v(j)$ for every eigenvector v of a different eigen value.

Applying Lemma 2 to vertices i and $i+1$ for $2 \leq i < n$, we find $n-2$ linearly independent eigen vectors of eigen value 1. To determine the last eigen value, the trace[2] of a matrix equals the sum of its eigen values. We know that the trace of L_{S_n} is $2n-2$, and we have identified $n-1$ eigenvalues that sum to $n-2$. So, the remaining eigen value must be n.

Thus the star graph on n vertices has eigen value 0 with multiplicity 1, eigen value 1 with multiplicity $n-2$ and eigenvalue n with multiplicity 1.