Kevin Wang (604036846)

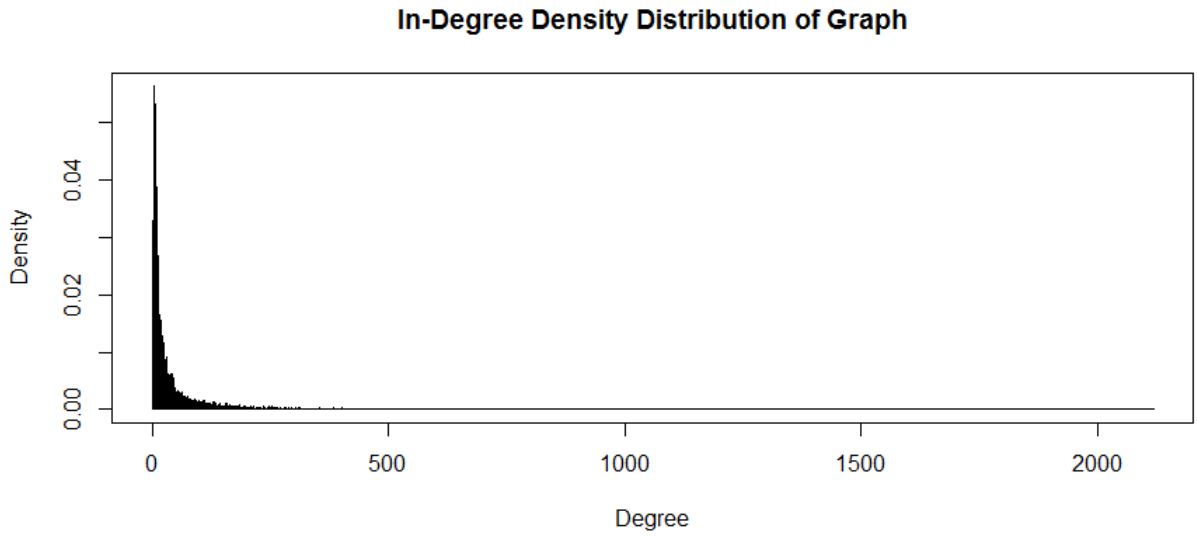
EE 232E Homework 3

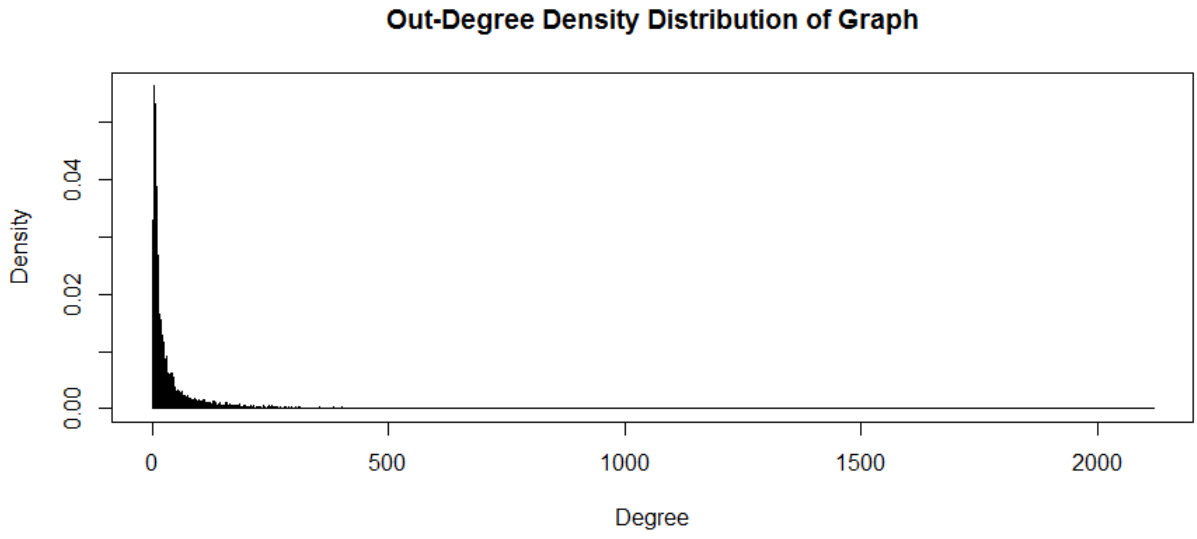
The coding for this assignment was done in R.

Problem 1

The graph/network is not connected according to *is.connected*, so the giant connected component is found using the method from the HW1 sample codes. There are 8 clusters, with 7 of them being only of size 2. The rest of the vertices are in the GCC.

Problem 2





It is noted that the in and out degree graphs are basically the same (most nodes might have the same in/out degrees), so the total degree plot would only include even numbers.

Problem 3

The network is converted to an undirected graph using *as.undirected*, and what to do with the weights is specified to match the instructions. Unless specified, the interesting traits listed in the table have to do with the GCC of the network.

	Fast Greedy	Label Propagation
Modularity of Network	0.3284922	0.000217338
Modularity of GCC	0.328771	0.0001494257
Number of Communities	8	5
Biggest Community Size	2316	10472

The two different algorithms did not provide similar results. The fast greedy method provided much better modularity, while the label propagation method provided results with very low modularity. It is evident that fast greedy method found more evenly distributed communities, while label propagation found one large community and some small ones (which is probably why its modularity is low).

Problem 4

The largest community is found to be community 5. The vertices are isolated and a subgraph is created using *induced_subgraph*. The following table summarizes the subcommunity structure of community 5:

Modularity	Number of Communities	Biggest Community Size
0.3626932	8	438

Problem 5

The following table summarizes the structures of the subcommunities that were larger than 100 nodes, which ended up being all of the communities.

Community ID	Community Size	Modularity	Number of Sub-communities	Biggest Sub-Community Size
1	1836	0.2230858	7	492
2	791	0.4193492	15	262
3	1701	0.3716341	9	502
4	1213	0.3975836	9	281
5	2316	0.3626932	8	438
6	634	0.4785346	15	170
7	963	0.5002231	14	296
8	1033	0.5053173	14	248

Problem 6

To compute the M_i vectors, random walks are started from a node using the defaults of 100 walkers and 100 time steps. Since there were a lot of nodes, the node that is selected to be node i is chosen at random. The visiting probabilities v_j are then computed using *netrw*, with the teleportation setting set to always teleport to the initial node (local.pagerank). The top 30 visiting probabilities and which nodes and communities they correspond to are found to compute M_i, which essentially represents the probability of visiting different communities from the initial node.

The following table shows 3 nodes that might be in more than one community. Each row represents a row in the M_i vector, corresponding to the probability of visiting that community. Only the top 30 node visiting probabilities are considered, so the total probabilities do not add up to 1.

The threshold of 10% is set for a node to be considered within a certain community. Nodes that are thought to be in multiple communities would be those that have at least two values in M that are greater than 0.1.

M_i: P(visit community)	Node 102	Node 8444	Node 3931
1	0.0124	0.1107	0.1164
2	0.0035	0.0099	0.0046
3	0.1024	0.0274	0.0148
4	0.2618	0.0131	0.245
5	0.0077	0.0027	0.0103
6	0	0.1996	0
7	0	0.0095	0
8	0	0.009	0