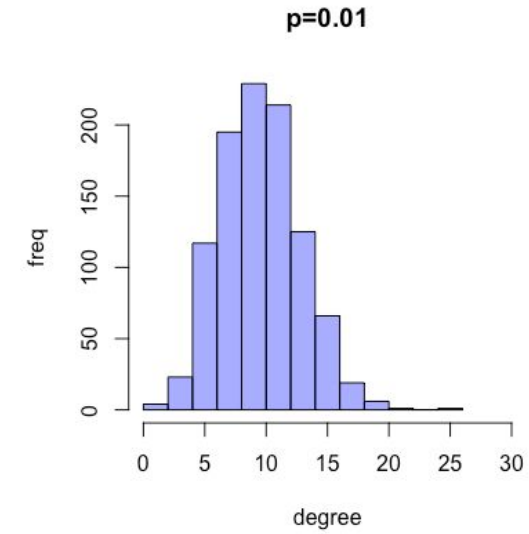
EE232E Homework #1 Report

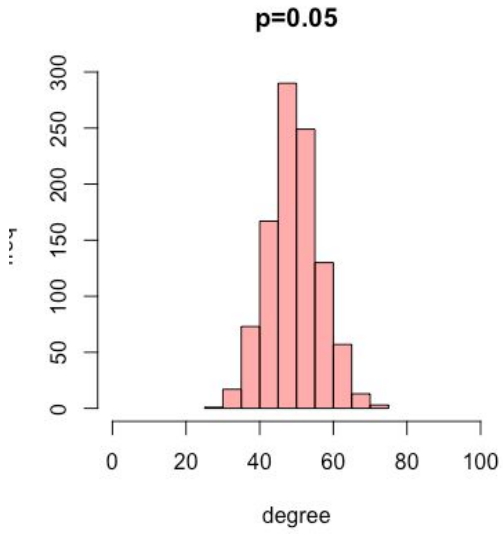
Dong Joon Kim, Yuanzhi Gao, Yanzhe Liu

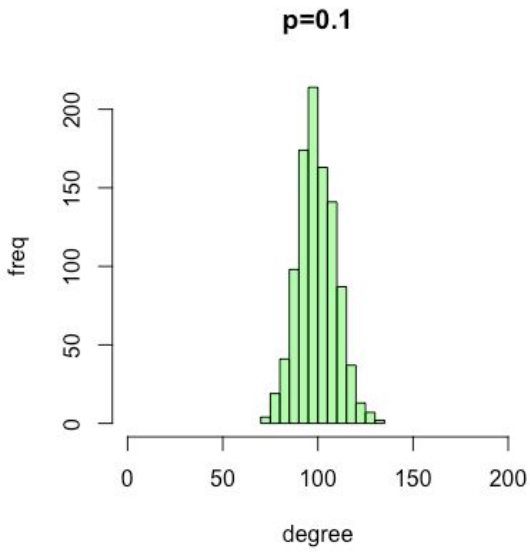
Problem 1:

1(a)

We created these three random network using the Erdos-Renyi model (erdos.renyi.game with type “gnp”) that allows us to draw an edge between two arbitrary vertices with given probability (0.01, 0.05 and 0.1). The degree distributions for these three networks are plotted as follow:







1(b)

We used is_connected and diameter functions to check the connectivity and diameter of our generated random networks. The results are as follows:

Probability	Connected/Disconnected	Diameter
0.01	Connected	5
0.05	Connected	3
0.1	Connected	3

1(c)

In order to find the threshold probability that when p < pc the generated random networks are disconnected, and when p > pc the generated random networks are connected, we used a while loop to keep incrementing the probability parameter by 0.001 (initially 0.001) of creating the random network until the network is connected (is_connected return TRUE). **The threshold probability we found is 0.007.**

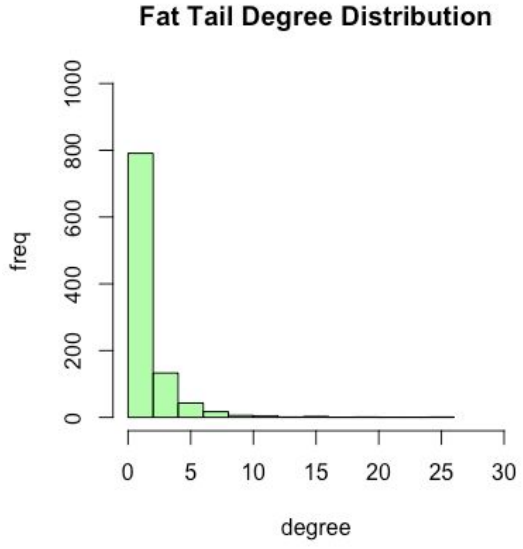
1(d)

The threshold probability can be analytically be derived by $\frac{ln(N)}{N} = \frac{ln(1000)}{1000} \approx 0.0069$

Problem 2:

2(a)

We generated our network using the Barabasi-Albert Model (barabasi.game function), since the degree distribution resulting from the BA model is scale free and it is proportional to X^{-3} . The degree distribution plot is as follow:



By using diameter function, we found that **the diameter for this specific random network is 20.**

2(b)

By using is_connected function, we found that **this fat tail network is connected.**

To find the giant connected component (GCC), we first find the clusters of the network by using function cluster(fat_tail_network), and use induced_subgraph(fat_tail_network, which(cluster$membership == which.max(cluster$csize))) to find the GCC. Finally, we use fastgreedy.community(giant_component) to get the community structure.

The **modularity** we measured is 0.934219.

The reason why modularity is so large is that network that has smaller edge density usually have a larger modularity and since this network is has degree distribution in X^{-3} and has a relatively large number of nodes, thus the modularity is high.

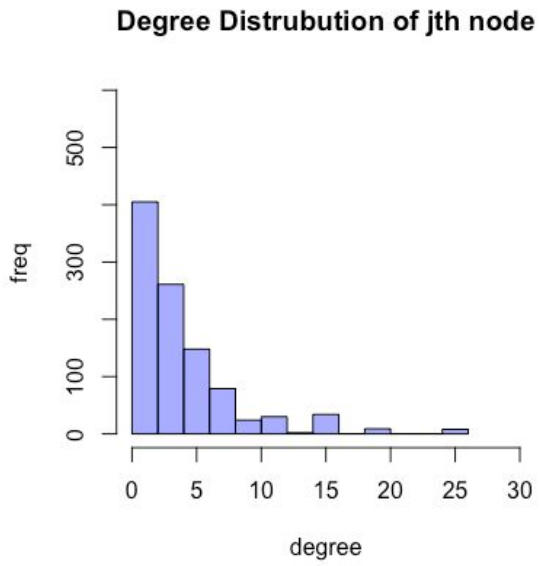
2(c)

We followed the same procedure to generate a fat tail network with 10000 nodes and the **modularity** we computed is 0.977786, which is larger than the smaller network's.

As mentioned in the last question, network that has smaller edge density usually has a larger modularity and since the network distribution is X^{-3} , thus larger number of nodes lead to smaller edge density and thus larger modularity.

2(d)

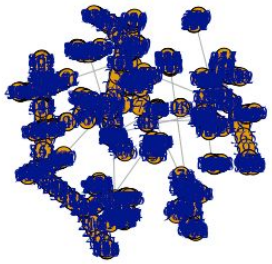
We randomly picked a node i, and then randomly pick a neighbor j of that node (if only one neighbor, then pick choose that neighbor as node j). We repeat this process for 1000 times and get the degree distribution of nodes j as follow:



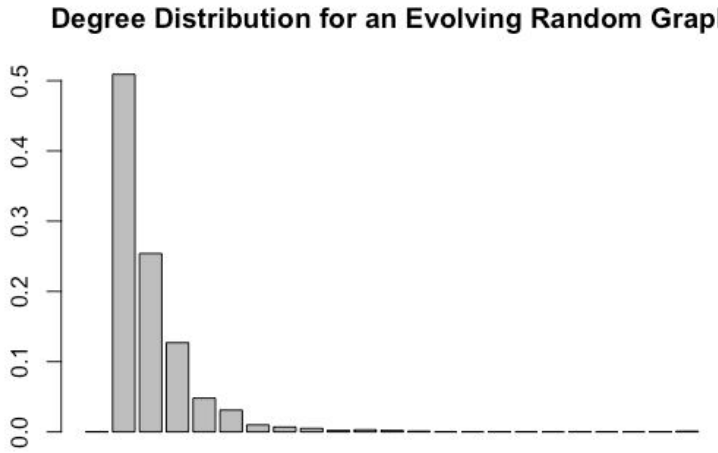
Problem 3:

3(a)

We generated an evolving random graph with preferential attachments and aging using aging.ba.game function. The following graph is the plot of the generated evolving random graph with PA exponent of 1 and age exponent of -1.

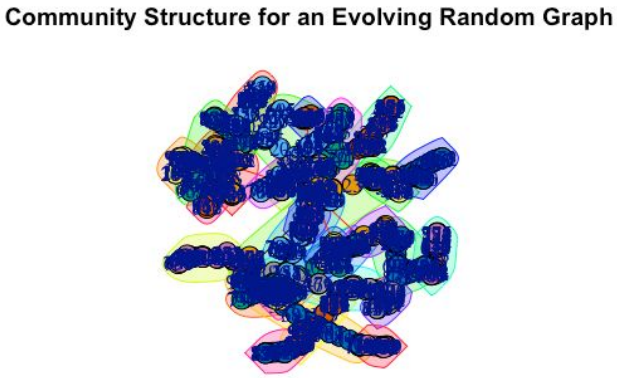


The degree distribution was found using degree_distribution function in igraph library. The degree distribution of the evolving random graph that we obtained earlier is as follows.



3(b)

The community structure was found using fastgreedy.community function in igraph library. The following graph shows the community structure of the evolving graph that we obtained from 3(a).

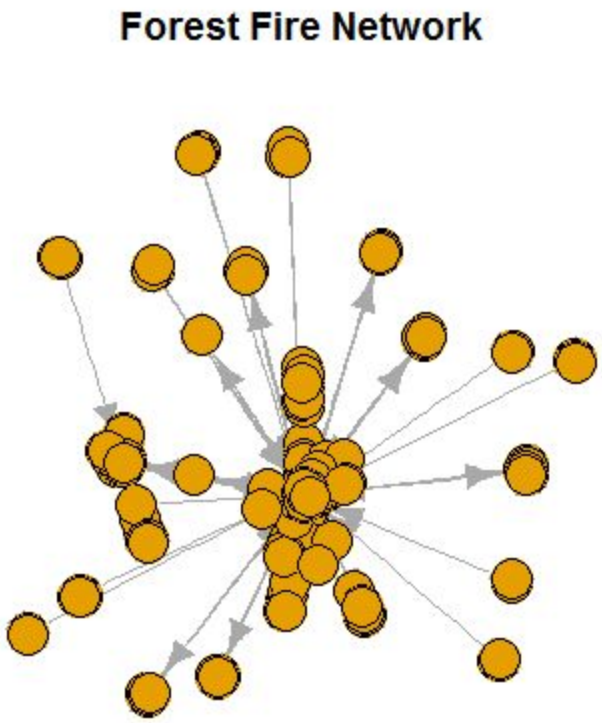


The modularity was calculated by using $modularity attribute of the return value of fastgreedy.community. The modularity is 0.9346699.

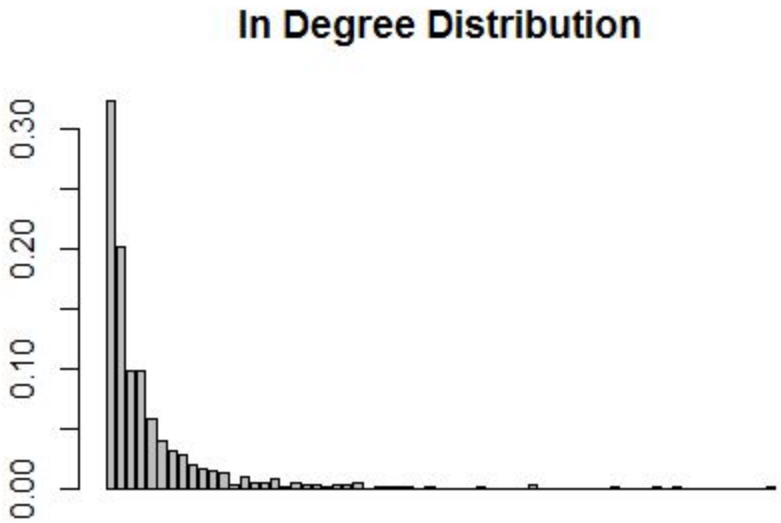
Problem 4:

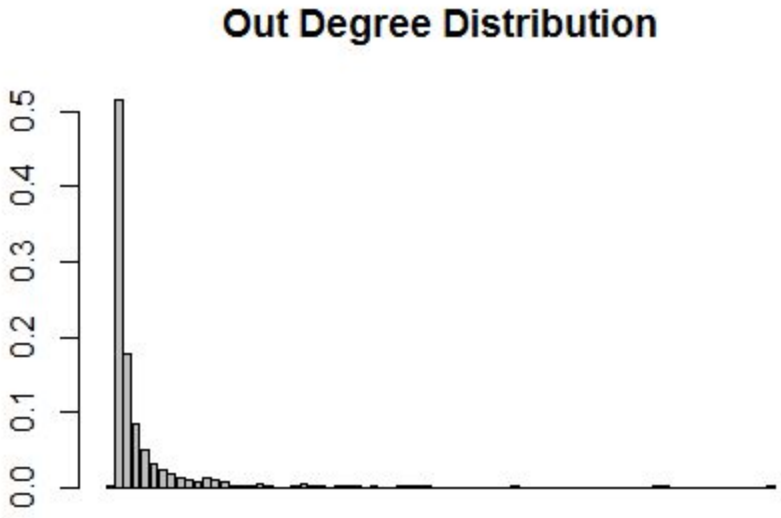
4(a)

We used igraph library's **forest.fire.game** model to create the growing network model, which resembles how the forest fire spreads by igniting trees close by. For parameters, we chose the values from the igraph examples, with number of nodes **n = 1000** and forward burning probability **fw.prob = 0.37** and backward burning ratio **bw.factor = 0.32/0.37**. The network generated is plotted below:



We then plotted the in-degree and out-degree distribution of the network.





From the graph we can clearly see that both the in-degree and out-degree distribution suggest that forest fire network model has a heavy-tailed distribution.

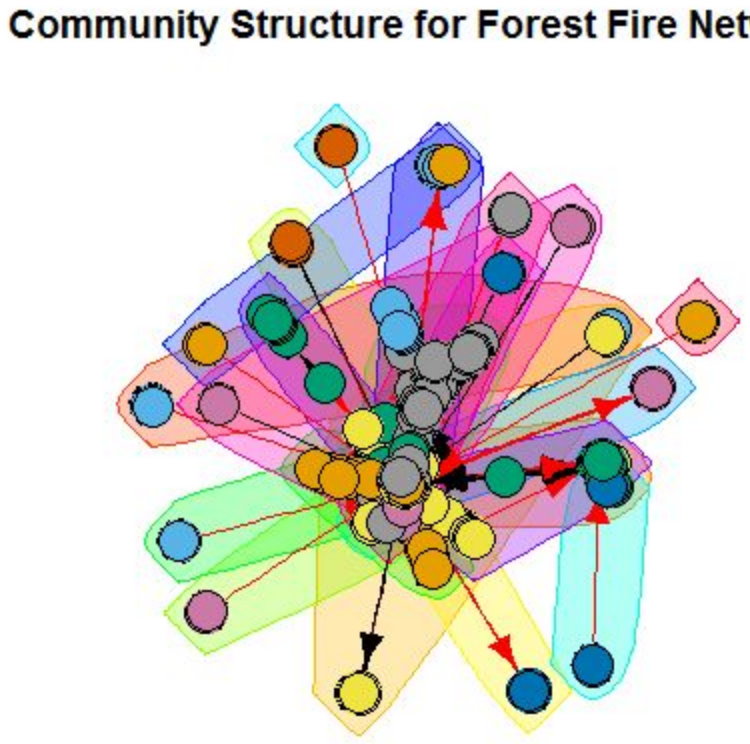
4(b)

We measured the network diameter to be:

diameter(forest_fire_graph) = 11

4(c)

We measured the network community structure using the **spinglass.community** function from igraph library. It is a community detection algorithm that uses the spin-glass model and simulate annealing. We found the community structure of the forest fire graph as the following:



We measured the modularity of the network to be:

modularity(community_structure)=0.5852896