Homework 1

Aozhu Chen 004773895
Ruoxi Zhang 404753334
Ning Xin 704775693

1. Create random networks

(a) Create three undirected random networks with 1000 nodes, and the probability p for drawing an edge between two arbitrary vertices 0.01, 0.05 and 0.1 respectively. Plot the degree distributions.

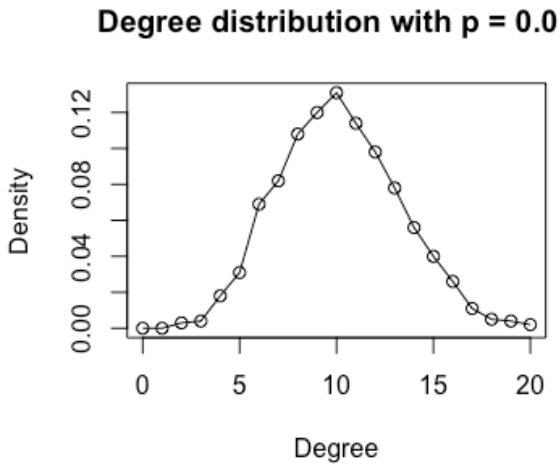


Figure 1.1 Degree distribution with p = 0.01

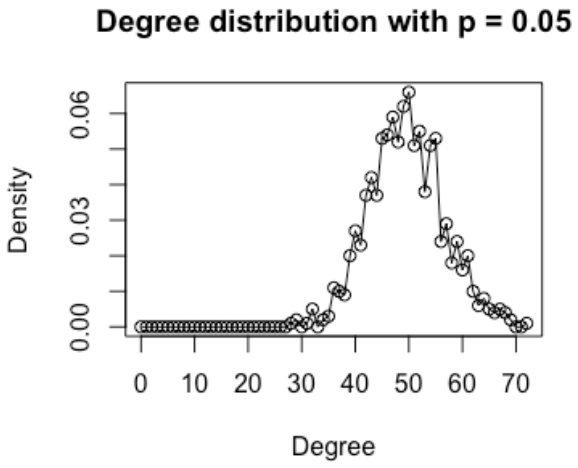


Figure 1.2 Degree distribution with p = 0.05

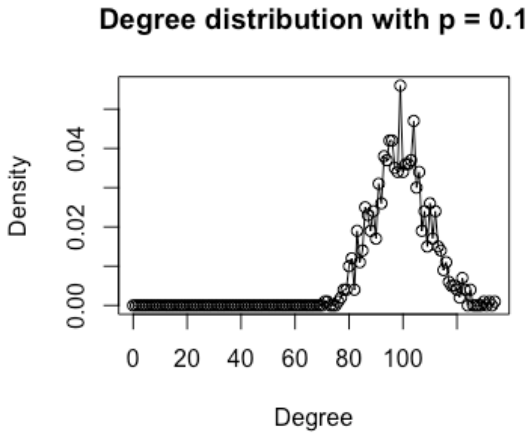


Figure 1.3 Degree distribution with p = 0.1

(b) Are these networks connected or disconnected? What are the diameters of these networks?

Initial results:

Firstly, by using our graphs from part (a), we can observe the initial connectivity and diameter.
The initial connectivity of p = 0.01 is connected and the diameter is 5.
The initial connectivity of p = 0.05 is connected and the diameter is 3.
The initial connectivity of p = 0.1 is connected and the diameter is 3.
As we all know, all the graphs were generated randomly, so we decided to calculate the average of 100 times of the connectivity and diameters.

Average results:

The average connectivity probability of p = 0.01 is 0.95 and the average diameter is 5.39.
The average connectivity probability of p = 0.05 is 1 and the average diameter is 3.
The average connectivity probability of p = 0.1 is 1 and the average diameter is 3.

(c) Try to numerically find a value pc (to three significant figures), so that when p < pc the generated random networks are disconnected, and when p > pc the generated random networks are connected.

In this question, we used method "is.connected()". More specifically, if the method return true, we will catch the threshold probability of p; if the method return false, we are going to increase the probability threshold (initial is 0.0000) of 0.0005 at each time.
After calculating the average of 50 random graphs, we get the following result.

Average results:

Pc = 0.00729 which means that when p < 0.00729 the generated random networks are disconnected, and when p > 0.00729 the generated random networks are connected.

(d) Can you analytically derive the value of pc?

By learning Erdos-Reyni Random Network, we know that each possible edge is picked with probability = p. When P=1, it is fully-connected.

E(Isolated node)

$$\approx \lim_{n\to\infty} n(1-p)^{n-1}$$

$$= \lim_{n\to\infty} ne^{-(n-1)p}$$

By setting the number of E(Isolated node) equal to 1, we can get the

$$p_c = \lim_{n\to\infty} \frac{\ln n}{n-1}$$

$\approx \ln(1000) / 1000$

$= 0.0069$

By comparing the result of part (d) and part (c), we can figure out the value of Pc that is almost same to each other, which is 0.0069 and 0.00729 respectively.

2. Create a network with a fat-tailed degree distribution

(a) Createanundirectednetworkwith1000 nodes, whose degree distribution is proportional to x−3. Plot the degree distribution. What is the diameter?

Degree distribution of graph with 1000 nodes:

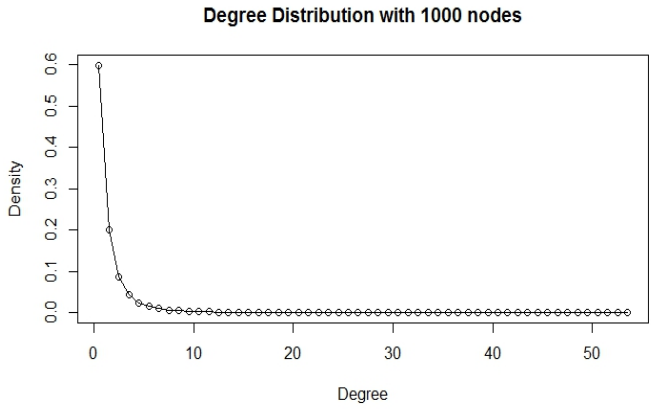


Figure 2.1 Degree distribution of 1000-node graph

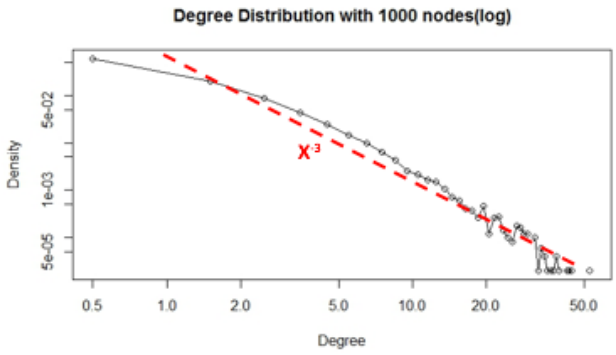


Figure 2.2 Logarithmic degree distribution of 1000-node graph

The average diameter of each cluster is 20.4.

(b) Is the network connected? Find the giant connected component (GCC) and use fast greedy method to find the community structure. Measure the modularity. Why is the modularity so large?

Whether the graph is fully connected can be determined by function *is.connected(graph)*. After we call this method on the generated graph, we found that it is fully connected.

The community structure of the network is generated through fast-greedy algorithm, and the modularity of this structure is measured by function *modularity(community)*, the value is approximately 0.930. (These 2 result can be checked by running the script).

The reason why the modularity is large is due to the phenomenon of "preferential attachment", which indicates that newly added nodes tends to connect to the nodes with higher degree. Besides, the mutual connection between low-degree nodes can also make great contribution to the large community as well.

(c) Try to generate a larger network with 10000 nodes whose degree distribution is proportional to x−3. Compute the modularity. Is it the same as the smaller network's?

Create a network with 10000 nodes.

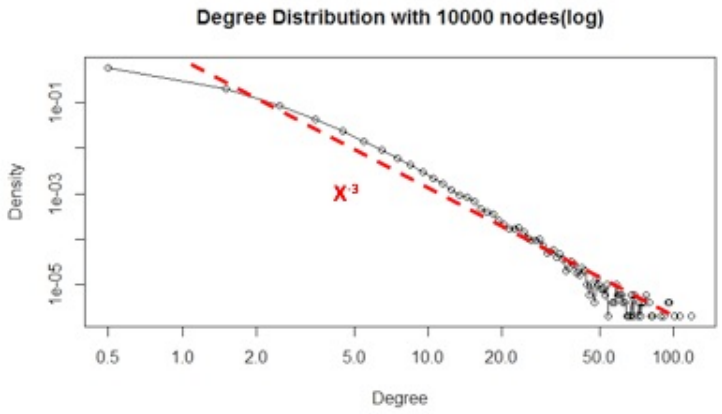


Figure 2.3 Logarithmic degree distribution of 10000-node graph

The modularity is calculated to be 0.978.

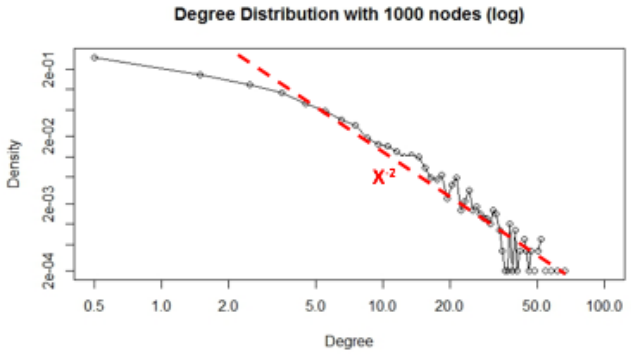
This is generally because for large scale networks, networks with smaller edge density always have higher modularity. As is the case in power law networks; and the modularity increases with the size of the network.

This characteristic originates from the phenomenon of "preferential attachment", which means new nodes that added in existed random network tend to connect to the high degree nodes. Besides, as the number of nodes increases, the low-degree node will connect between themselves and in turn make a significant contribution to the modularity.

(d) You can randomly pick a node i, and then randomly pick a neighbor j of that node. Measure and plot the degree distribution of nodes j that are picked with this process.

Given a network, we can get a group of sample nodes by picking nodes randomly from this network. Then the sample should be uniform. Then from the samples we have now, we pick a random node's neighbor, then we can get a second sample of the nodes. The last thing to determine is to find whether these 2 samples are equivalent.

The answer is no. In the first sample process, nodes are randomly picked, but in the second sample process, the nodes with larger degrees have larger probability to be picked. And from the result we can see that the degree distribution of this fat-tailed network has a power law index of -2.



3. Creates a random graph by simulating its evolution

We use aging.prefatt.game() method to generate the random graph by simulating its evolution. The parameters we chose is pa.exp=1, aging.exp = -1.

(a) Each time a new vertex is added it creates a number of links to old vertices and the probability that an old vertex is cited depends on its in-degree (preferential attachment) and age. Produce such an undirected network with 1000 nodes. Plot the degree distribution.

The degree distribution for parameters we chose is as following

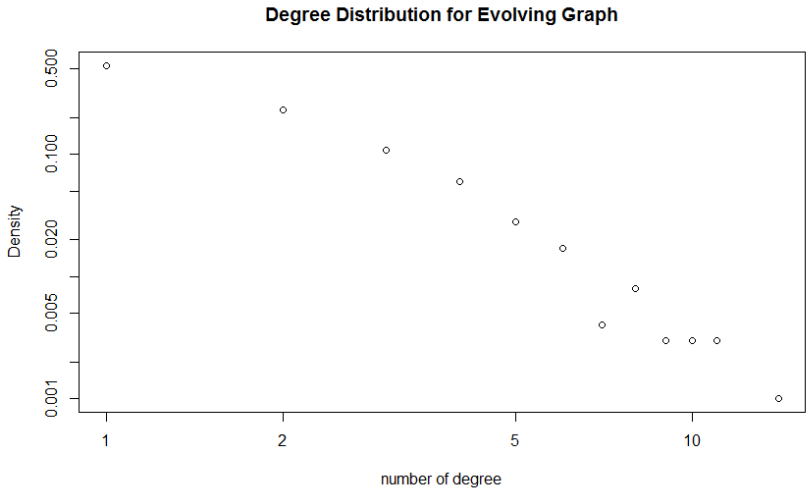


Figure 3.a.1 LogP_k versus K

*P_k = probability that a randomly picked node has degree = k
*K is the node degree

From the figure 3.a.1 we can see that the distribution of the degree distribution is more likely to be a power law distribution.

(b) Use fast greedy method to find the community structure. What is the modularity?

We use fast greedy method to detect the community. The fast greedy method tries to find dense subgraph, also called communities in graphs via directly optimizing a modularity score. The modularity and community structure is as following

Modularity = 0.9357926

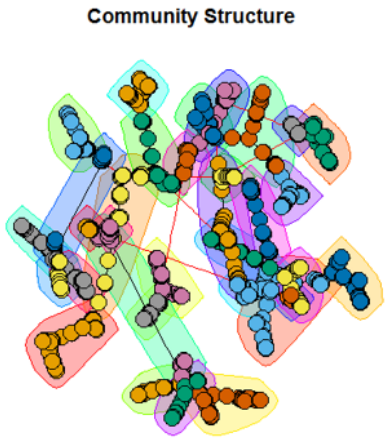


Figure 3.b.1Community Structure of the random graph detected by fast greedy algorithm

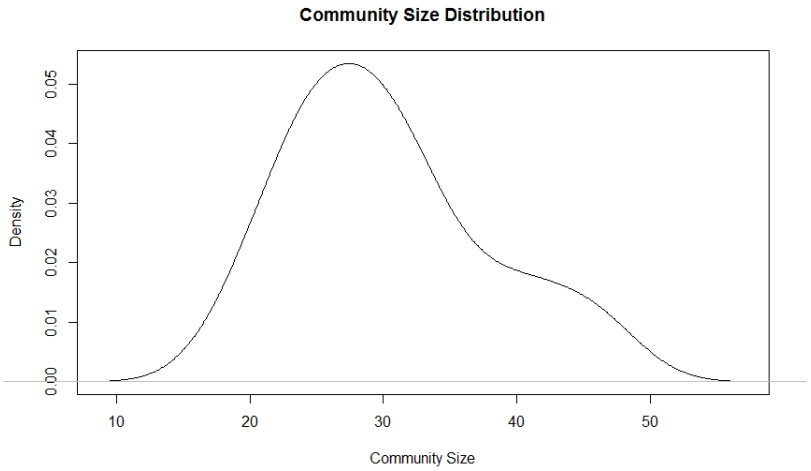


Figure 3.b.2 The summary of the community size distribution

4. Use the forest fire model to create a directed network

In this part we use aging.prefatt.game() to generate a growing network model. The parameter we chose for this method is as following: forward burning probability is 0.37 and backward probability is 0.32/0.37. We run the graph generating for 500 times and calculating the average result.

(a) This is a growing network model, which resembles how the forest fire spreads by igniting trees close by. Plot the in and out degree distributions.

The in degree and our degree distribution is as follow:

I. In degree distribution

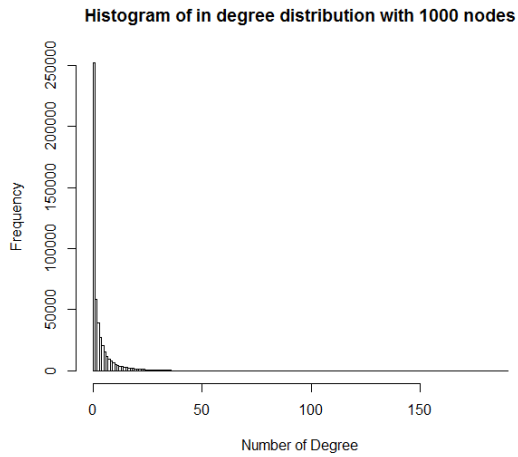


Figure 4.a.1 the frequency of the in degree

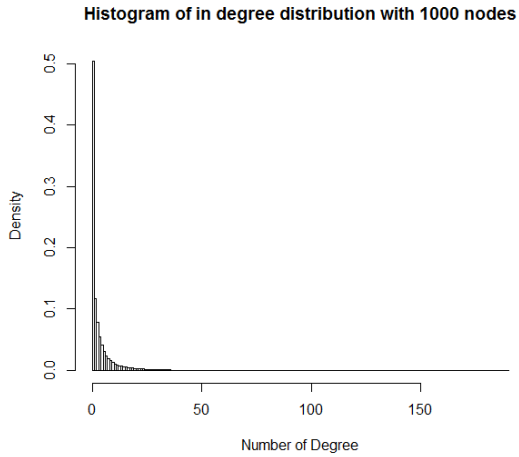


Figure 4.a.2 the density of the in degree

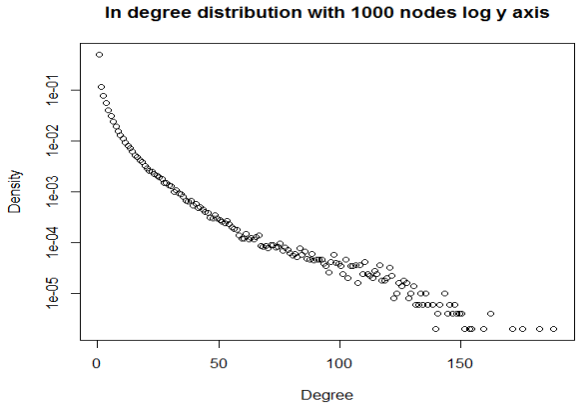


Figure 4.a.3 LogP_k versus K

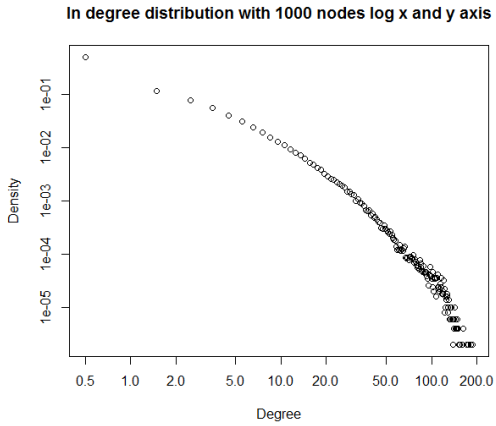


Figure 4.a.4 LogP_k versus LogK

*P_k = probability that a randomly picked node has degree = k
*K is the node degree
From the figure 4.a.3 we can see that the distribution of the in-edge degree distribution is more likely to be a exponetial law distribution.

II. Out degree distribution

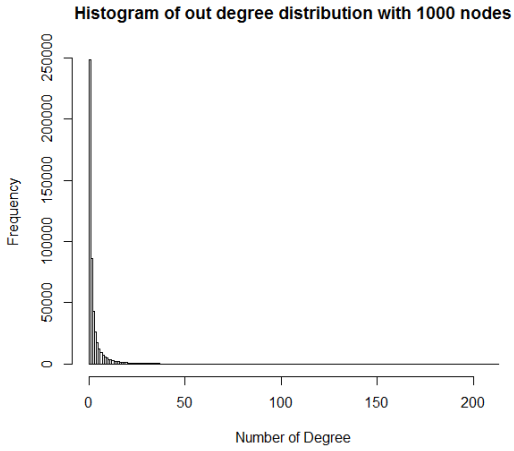


Figure 4.b.1 the frequency of the out degree

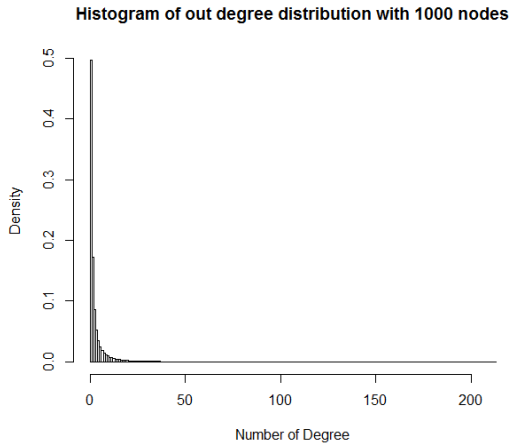


Figure 4.b.2 the density of the out degree

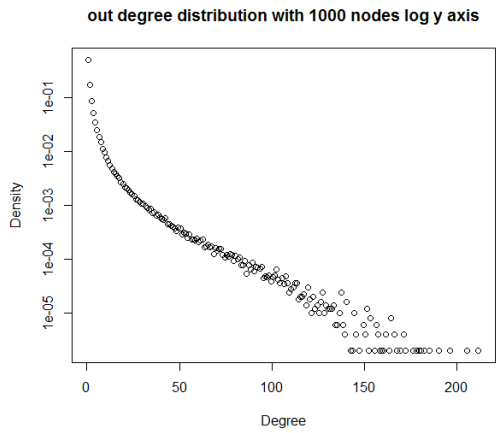


Figure 4.a.3 $LogP_k$ versus K

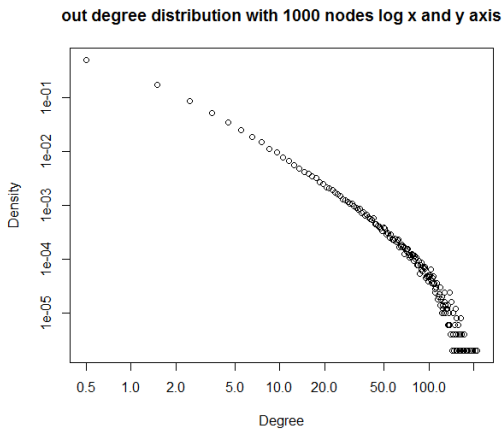


Figure 4.a.4 $LogP_k$ versus LogK

*P_k = probability that a randomly picked node has degree = k
*K is the node degree
From the figures above we can see that the distribution of the out-edge degree distribution is more likely to be a power law distribution.

(b) Measure the diameter.

We measure the diameter of the directed network and repeat measurement for 50 times. The average diameter of the network we got is:

```
Avg Diameter of Forest Fire Graphes of 50 iterations :  10.18
```

(c) Measure the community structure and modularity.

We use walktarp method to detect the community. The walktrap method tries to find densely connected subgraphs, also called communities in a graph via random walks. The idea is that short random walks tend to stay in the same community. Also, we repeat the detection for 50 times. The average modularity and the community structure detected from one of the graph as following:

```
Avg modularity of Forest Fire Graphes of 50 iterations :  0.3921079
```

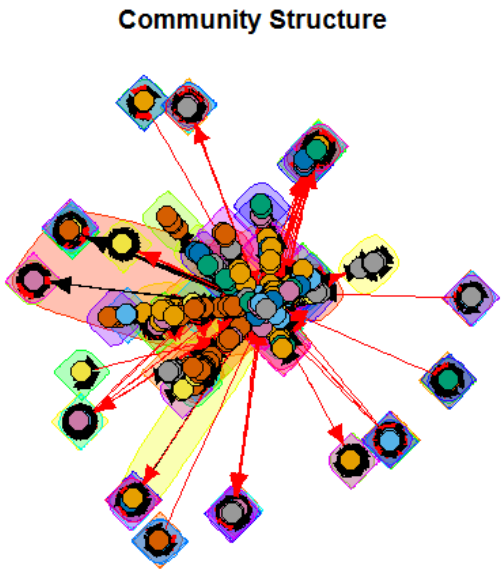


Figure 4.c.1 Community Structure of Forest Fire detected by walktrap algorithm