EE232E Graphs and Network Flows

Spring 2017_HW1

Team Member:
Xiongfeng Hu (304753117)
Younan Liang (504759929)
Dui Lin (504759948)

Table of Content

1. Introduction

This assignment is intended to review some the generative models for networks, degree distributions and get familiar with using igraph package in R. Since igraph supports R language best, which is also the recommended language in this course, we decided to resolve all the problems and implement our solutions with R. Specifically, random graph model, fat-tailed distribution model, evolution simulation model and forest fire model will be explored, as well as their community structure and modularity. Generally speaking, we're expected to achieve the following goals: (1) Be familiar with random graph generation; (2) Understand basic concepts in graph theory; (3) Practice and get used to igraph tools in R.

2. Problem Solutions

2.1 Create random networks

2.1.a)

The degree distributions are as follows:

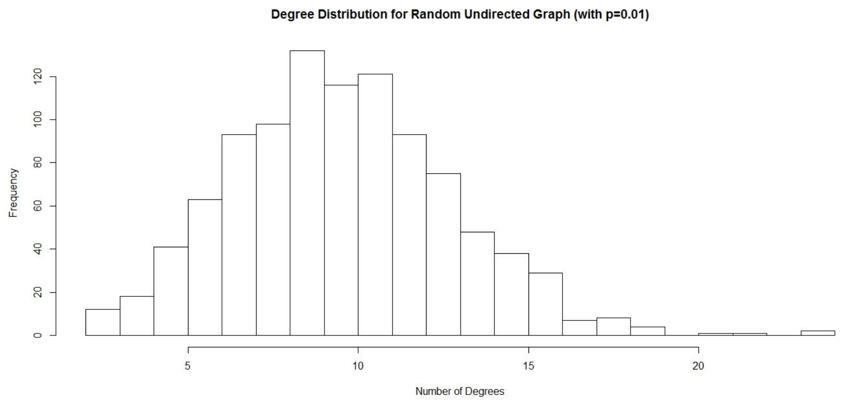

Figure 1. p=0.01

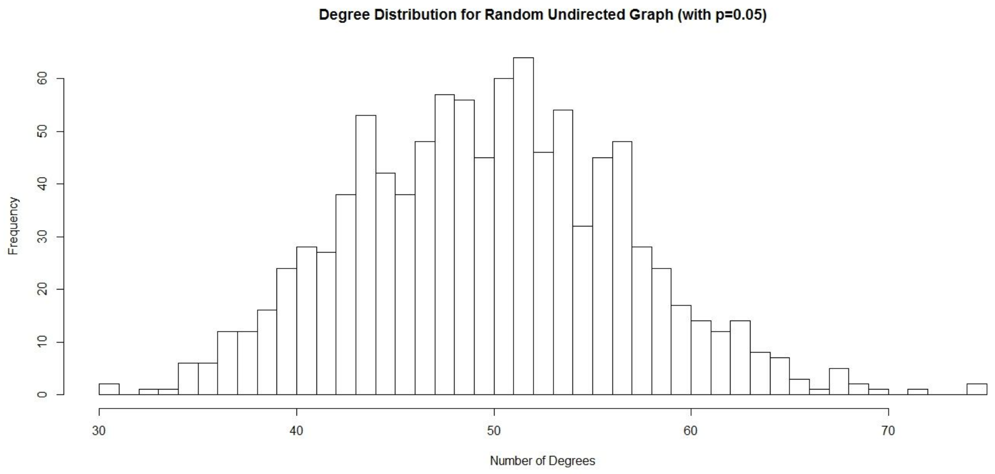

Figure 2. p=0.05

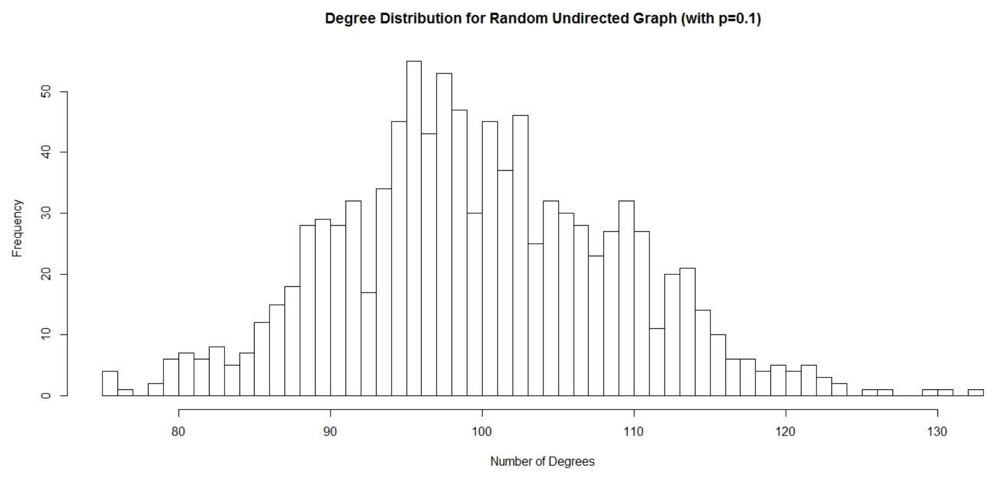

Figure 3. p=0.1

Let us check again: In figure 1, the medium value is at p*N=0.01*1000=10;
In figure 2, the medium value is at p*N=0.05*1000=50;
In figure 3, the medium value is at p*N=0.1*1000=100.

2.1.b)

By programming we find that all these 3 networks are connected. This is intuitively right. Even if p=0.01, a node is averagely connected to 10 other nodes, so it is connected.

To calculate the diameter of these networks, we must consider the uncertainty of the randomly generated networks. Therefore we regenerated each of these networks 50 times, and the calculate the average diameter of these networks. The average diameter is 5.30(p=0.01), and 2.94(p=0.05 or 0.1).

p	E(diameter)
0.01	5.30
0.05	2.94
0.1	2.94

Figure 4. The relation between p and diameter

2.1.c)

It is obvious that this p_c should less than 0.01. In order to preserve accuracy, we test the p_c from 0.0001 to 0.0099, with the increment of 0.0001. Repeat the calculation for 50 times, and we find the mean p_c value is 0.0069. It means that when a node is connected to less than 7 other nodes averagely, it is very likely that the graph is not connected.

2.1.d)

According to Erdos-Renyi model:

$$p_c = \frac{ln(N)}{N} = \frac{ln(1000)}{1000} = 0.0069$$

It corresponds well with the value got in the experiment. However, this p_c does not necessarily guarantee that the graph will be connected at all times. It means when p is greater than p_c ,the probability that the graph is not connected is very small, or negligible.

2.2 Create a network with a fat-tailed degree distribution

2.2.a)

The degree distribution resulting from the BA model is scale free, in particular, it is a power law of the form x^-3, Figure 2.1 shows the degree distribution of an undirected fat-tailed network and the scatter-plot degree distribution in log space, respectively.

Diameter: The diameter of the graph is 19.84 (mean value for looping 100 times).

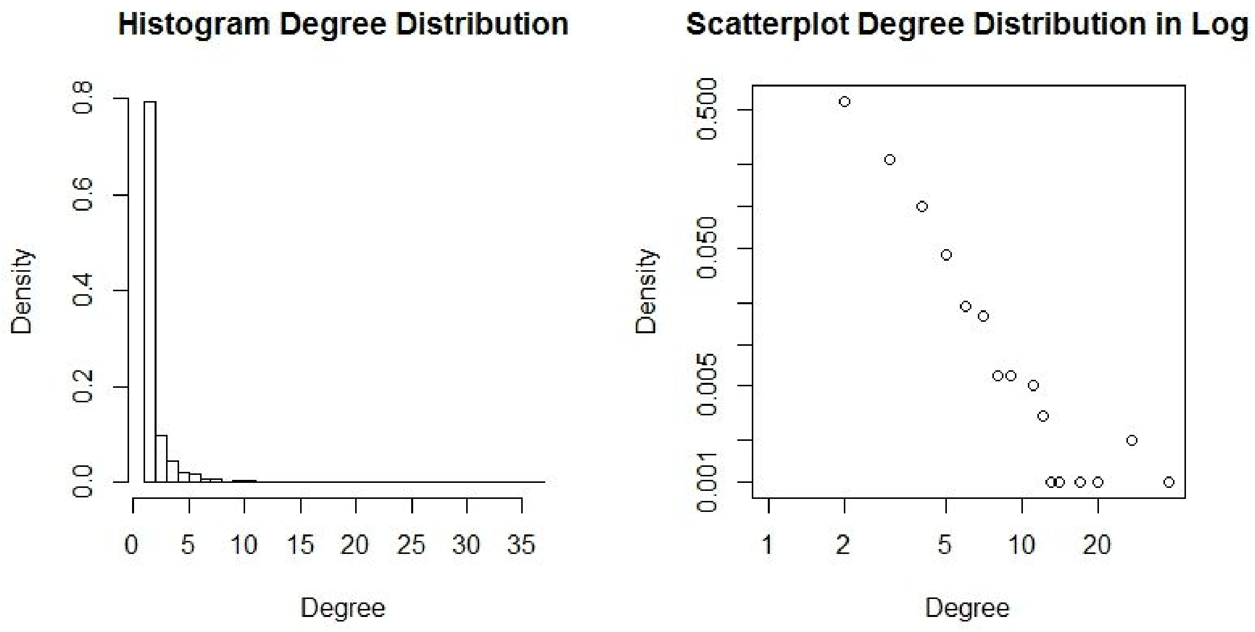

Figure 5. Histogram and Scatterplot fat-tailed degree distribution (1000 nodes)

2.2.b)

Figure 2.2 shows that the distribution of a random generated network with 1000 nodes and its giant connected component distribution (after Non-GCC nodes deleted), respectively.

Connectivity: We test 100 random generated graphs and find out that the network is always connected; **Giant Connected Component (GCC):** Because the entire graph is connected, so the GCC is the whole graph; **Community Structure:** Graph community structure calculated with the fast greedy algorithm; **Modularity (best split):** 0.9321013; **Why is the modularity so large:** The modularity for the graph is 0.9321013. The reason that it has such a large modularity is because the graph is generated from barabasi model, which has preferential attachment mechanism. In other words, the new added nodes tend to be linked to the nodes with high degree. We could also see this from the degree distribution. So it has dense connections between the nodes within modules but sparse connections between nodes in different modules. Thus the modularity is large for this graph.

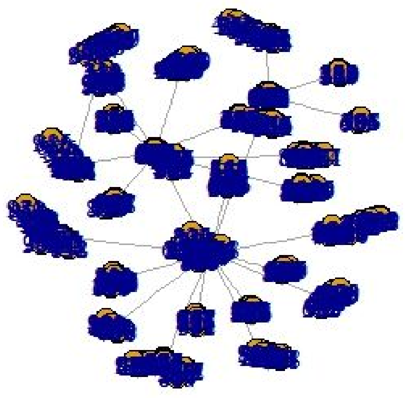
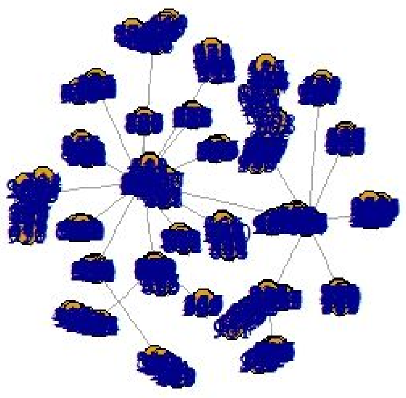

Figure 6. A fat-tailed network (left) and its GCC distribution (right)

2.2.c)

This time we generate a larger network with 10000 nodes whose degree distribution is proportional to x^-3. **Modularity (best split):** 0.9779122; **Comparison with smaller network:** Graph community structure is calculated with the fast greedy algorithm and We could see that the modularity is slightly larger than the 1000 nodes graph.

2.2.d)

Here we randomly pick a node i, and then randomly pick a neighbor j of that node. Figure 2.3 measure and plot the degree distribution of nodes j that are picked with this process.

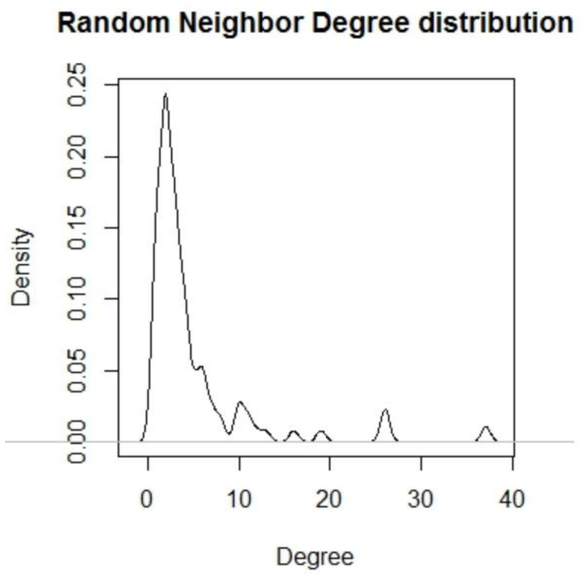

Figure 7. Degree Distribution for node j in the process

2.3 Creates a random graph by simulating its evolution

2.3.a)

We use aging.prefatt.game() method of the igraph package to generate a random graph with 1000 nodes by simulating its evolution. The degree distribution is as below:

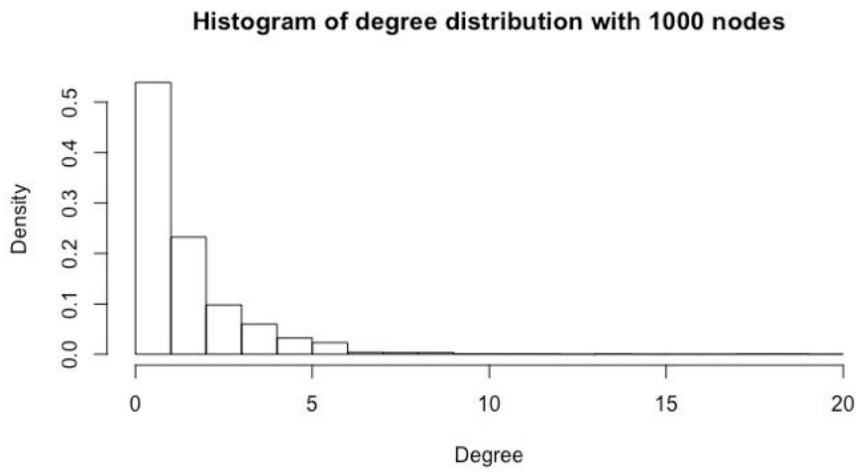

Figure 8. Degree Distribution for a random graph with 1000 nodes

2.3.b)

The community structure and modularity are computed using fast greedy method as below:
Community Structure = Number of Communities = 36
Modularity of graph = 0.9347781

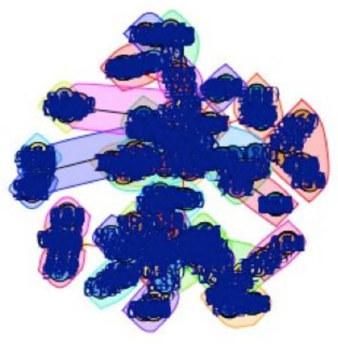

Figure 9. Community Structure of a random graph with 1000 nodes

2.4 Use the forest fire model to create a directed network

2.4.a)

We use the forest.fire.game() method to create a directed network. We chose the forward burning probability as 0.37 and backward probability as 0.32/0.37. The In and Out degree distributions for this network are as below:

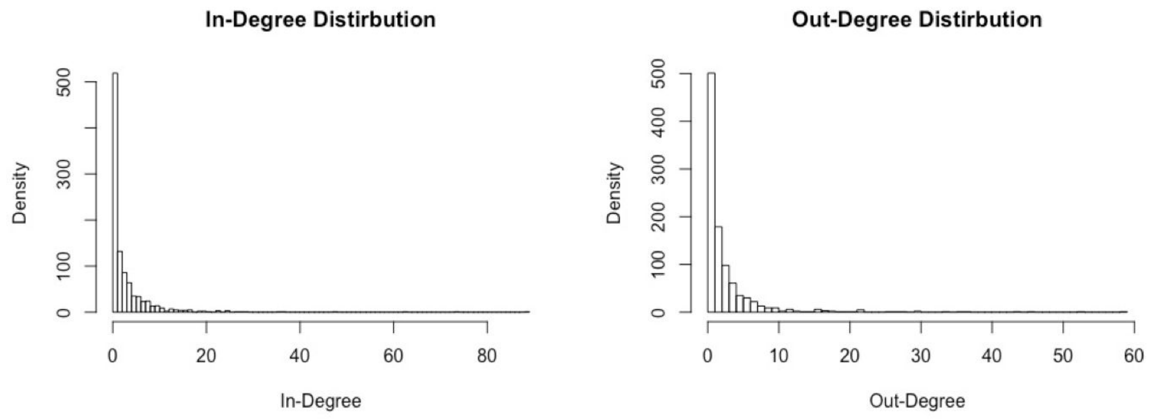

Figure 10. In and Out Degree Distribution for a forest fire network

2.4.b)

The diameter is computed using diameter() method as below:
Diameter of Forest Fire Graph = 11

2.4.c)

The community structure and modularity are computed as below:
Community Structure = Number of Communities = 24
Modularity of graph = 0.6656197

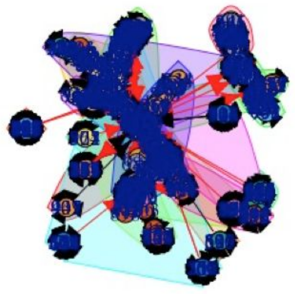

Figure 11. Community Structure of a forest fire network

3. Summary

In general, the problems are not hard, and it’s a warm-up for us to get familiar with igraph and R programming. But we still encountered some problems and finally resolved them. For problem 2, at first, we didn’t know what exactly means degree distribution is proportional to x^-3. We tried different methods by ourselves to create the graph, but it didn’t seem to be correct. Then we do more research on the details of igraph functions and find out the barabasi function does what exactly what we need.