1. Objectives

In this homework, we are supposed to explore some basic features of several network models with the assistance of igraph tool. Since igraph supports R language best, we are also supposed to learn some basic R grammar. Specifically, random graph model, fat-tailed distribution model, evolution simulation model, forest fire model will be explored, as well as their community structure and modularity. Generally speaking, we're expected to achieve the following goals:

- Be familiar with random graph generation
- Understand basic concepts in graph theory
- Practice and get used to igraph tools in R

2. Problems

2.1. Problem 1 - Random Networks

(a)

Figure1.1 shows the figures of degree distribution of randomly generated graph with probability p = 0.1, 0.05, 0.001 respectively. It can be easily seen that when p increases, the degrees of nodes tend to increase also.

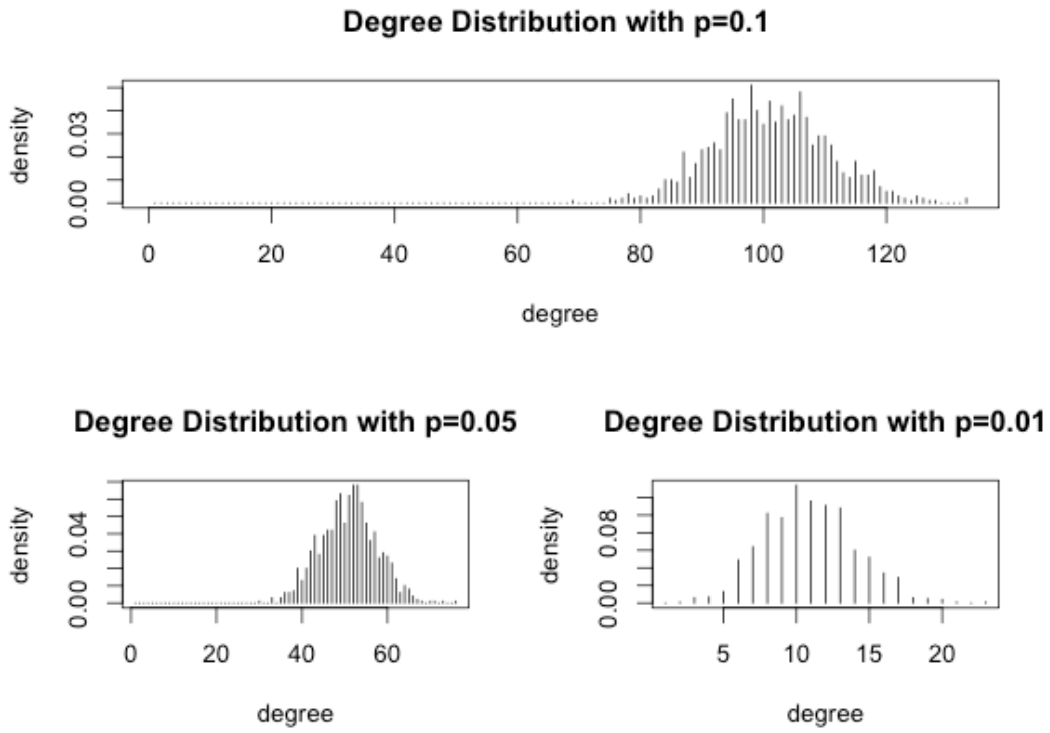

Figure1.1 Degree Distribution for randomly generated graph

(b)

Since the graphs are randomly generated, the fact whether they're connected is also random. The diameter also varies every time. Hence it's more reasonable to repeat generation and compute the average of connection status and diameter. We repeated for 100 times and the average is calculated as following:

	P=0.1	P=0.05	P=0.01
Connectivity	100% connected	100% connected	98% connected
Diameter	3	3	5.51

Table1.1, Average connectivity and Diameter for randomly generated graphs

(c)

We use binary search to find Pc between 0 and 1, such that is.connected (Pc-0.0001)=FALSE and is.connected (Pc+0.0001)=TRUE. And the same story as (b), this process should be repeated and the average will be calculated. We repeated 100 times to generate a reasonable outcome. The final Pc finds is Pc = 0.00751

(d)

The expectation of the number of isolated nodes could be represent as

$$n \approx N(1-p)^N = N(1-\frac{Np}{N})^N \approx Ne^{-Np}$$

N represents the number of nodes (when N is relatively large, consider N-1 as N), p represents the probability to form an edge between two nodes.

We could see from the equation that if we increase p, then n will go down. So for the boundary case, the number of isolated nodes should be equals to 1. $Ne^{-Np_c} = 1$,which means $p_c = \frac{\ln N}{N}$. So when N =1000 $p_c \approx 0.0069$. Compare with the result in (c), we can find out that a small error occurred.

2.2 Problem 2 - Network of Fat-tailed Degree Distribution

(a)

Degree Distribution Plot:

Figure2.1 shows the degree distribution of the fat-tailed graph and the distribution in log space, respectively.

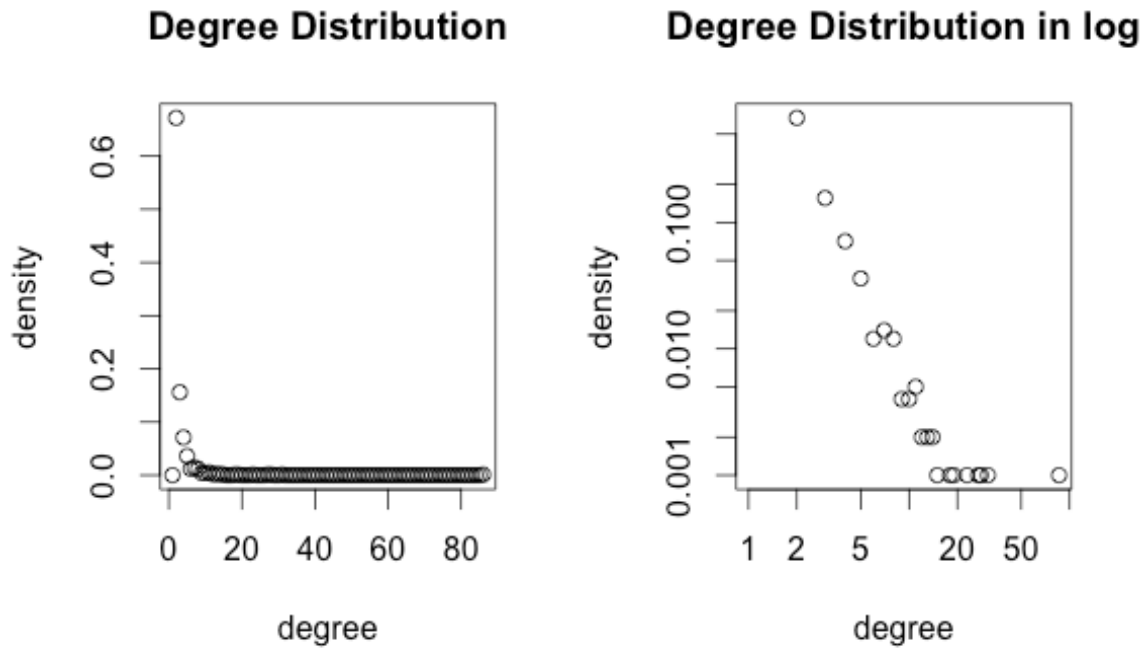

Figure2.1 Fat-tailed Degree Distribution (1000 nodes)

Diameter:

The diameter of the graph is 17.25(mean value for 100 times).

(b)

Connectivity:

The graph is always connected when we repeated for 100 times.

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

Number of communities (best split): 37

Modularity (best split): 0.9275642

Modularity:

The modularity for the graph is 0.9275642. 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.

(c)

For a larger graph, which has 10000 nodes:

Degree Distribution:

Figure2.2 shows the degree distribution of the fat-tailed graph and the distribution in log space, respectively.

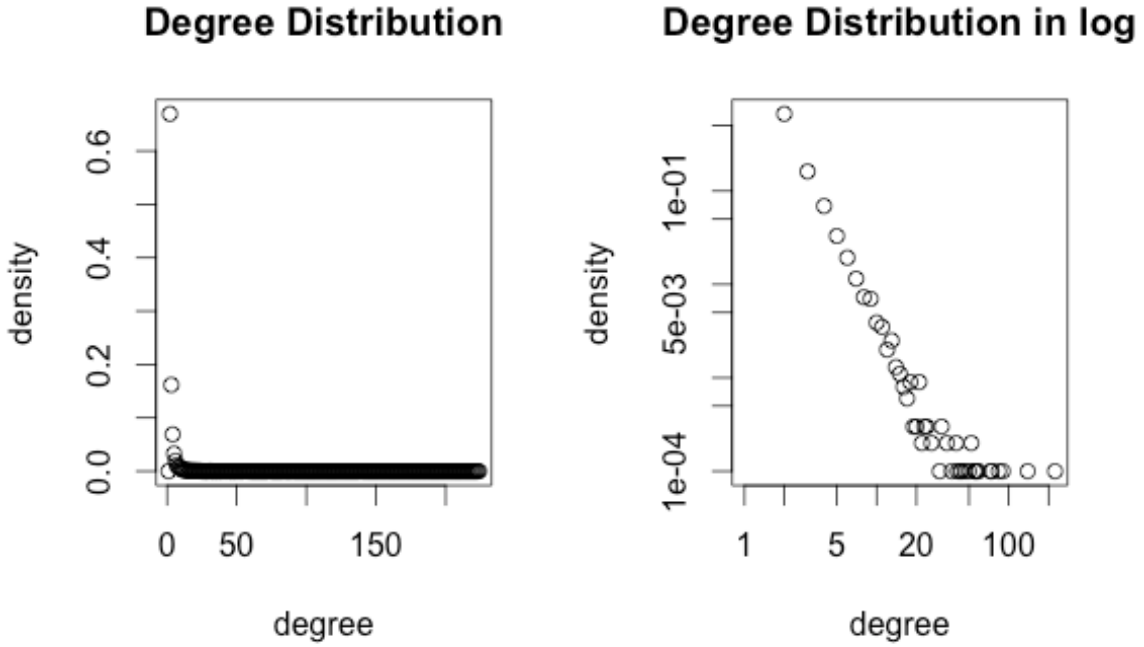

Figure2.2 Fat-tailed Degree Distribution (10000 nodes)

Modularity:

Graph community structure calculated with the fast greedy algorithm

Number of communities (best split): 119

Modularity (best split): 0.9748914

We could see that the modularity is slightly larger than the 1000 nodes graph.

(d)

Degree Distribution

Figure 2.3 shows the degree distribution of the process.

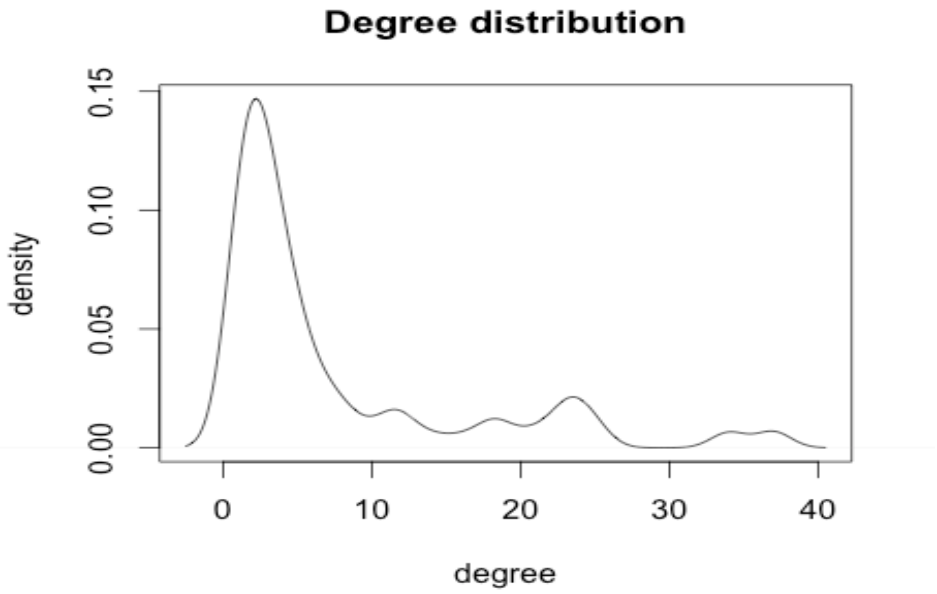

Figure 2.3 Degree Distribution for node j

2.3 Problem 3 - Process in the evolution

For this problem, we use a library function in the igrap called aging.prefatt.game(), which simulates exactly the process in the evolution.

(a)

Degree Distribution:

Figure 3.1 shows the degree distribution of the graph.

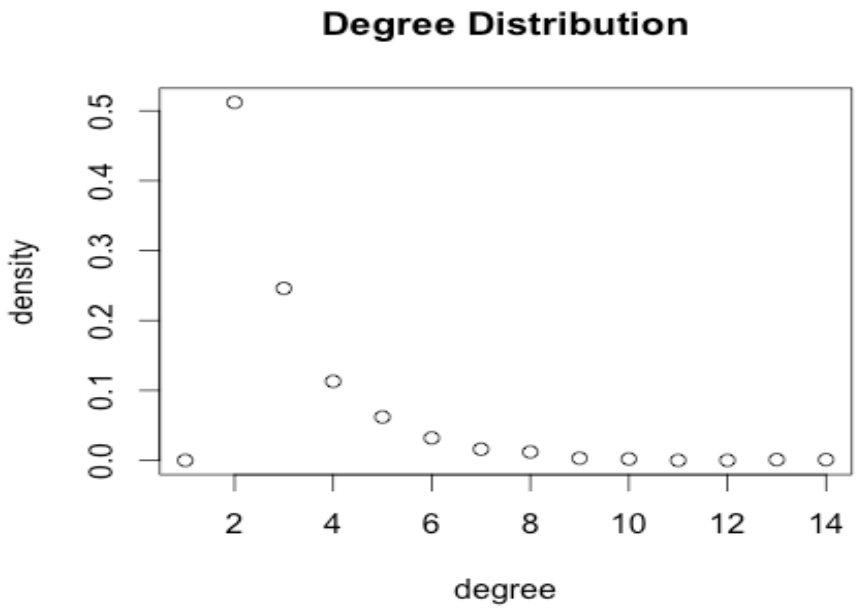

Figure 3.1Degree Distribution

(b)

Modularity:

Graph community structure calculated with the fast greedy algorithm

Number of communities (best split): 32

Modularity (best split): 0.9353798

Figure 3.2 shows the community structure in the graph by using fast greedy algorithm.

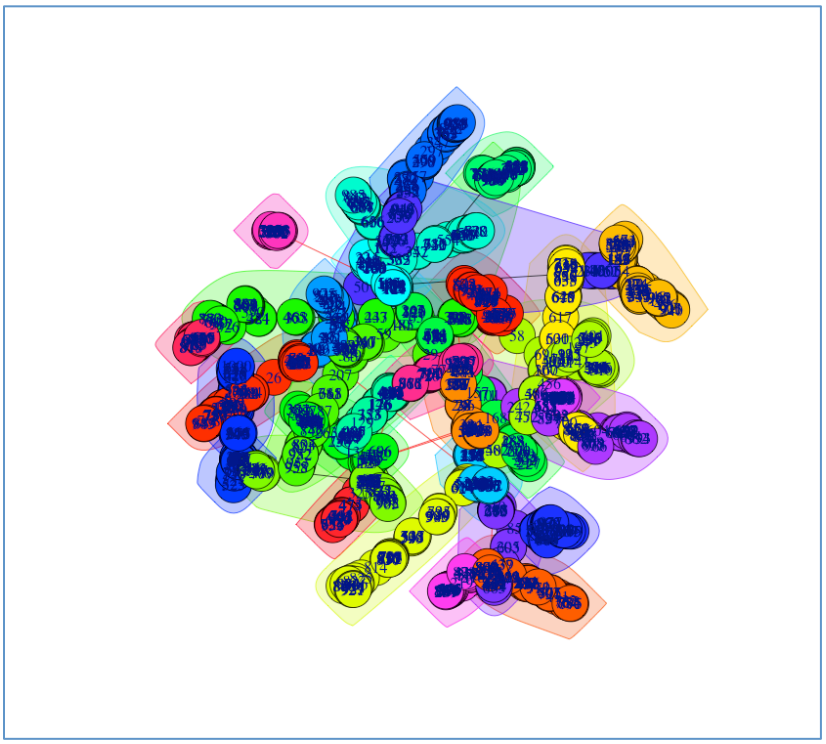

Figure 3.2 Community Structure

2.4. Problem 4 – Network of Forest fire model

(a)

We generate a graph with 1000 nodes using function forest.fire.game(), and the forward burning probability is set to 0.5, while the backward burning ratio is set to 0.5. And we generated 1 ambassador. The in-degree and out-degree distributions are plotted as below:

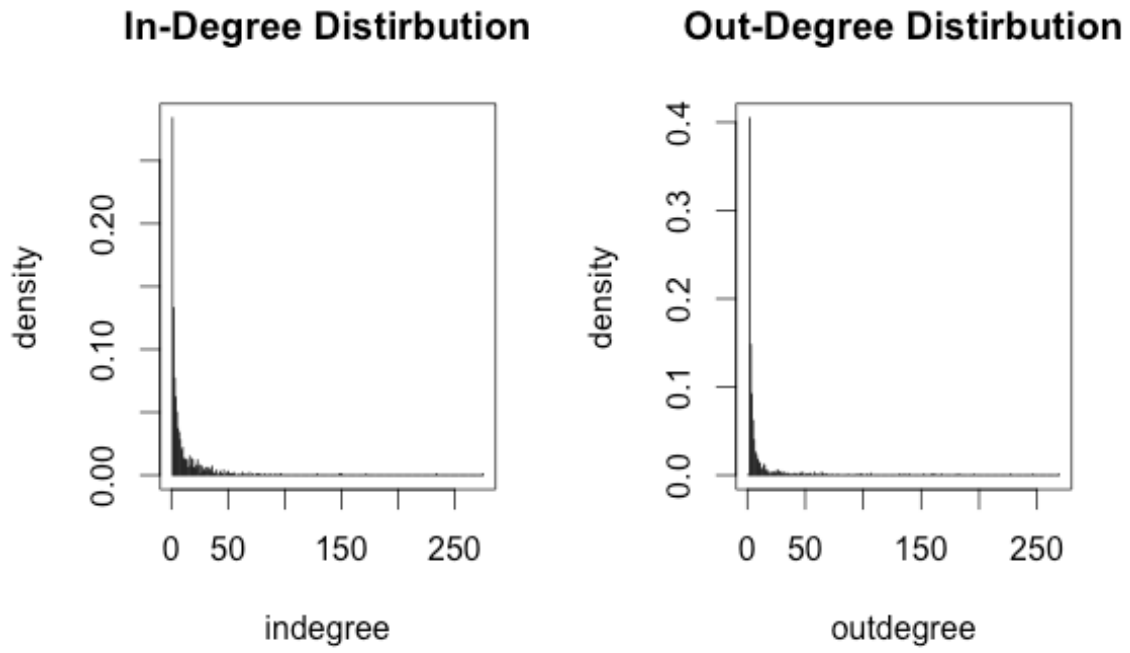

Figure 4.1 In-degree and Out-degree for graph of forest fire model

(b)

As practiced before, we repeated 100 times to compute the average diameter, which is equal to 9.52

(c)

Since fast greedy method is only applicable for undirected graph, we use walk trap method to generate the community structure, which is shown as below. And its modularity is 0.0682.

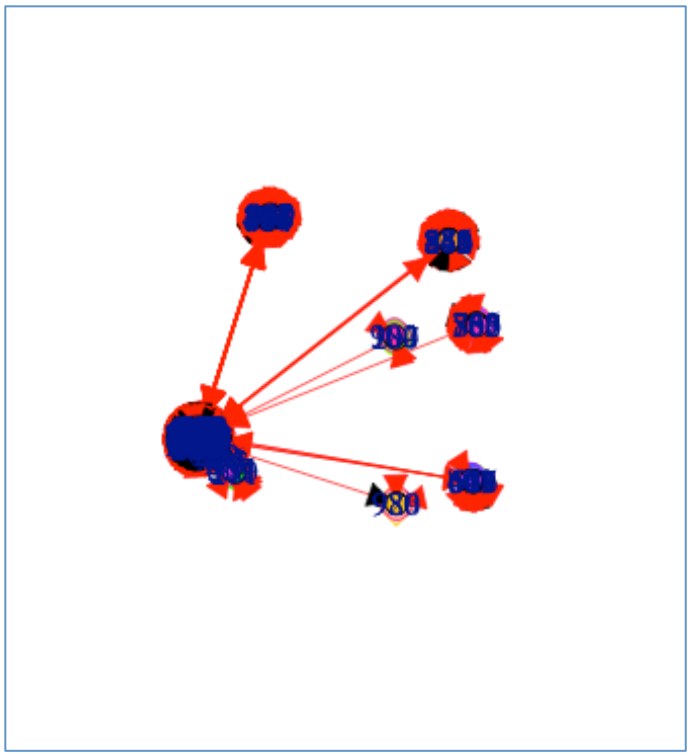

Figure 4.2 Community structure for graph of forest fire model

3. Difficulty encountered

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.

In general, the problems are not hard, and it's a warm-up for us to get familiar with igraph and R programming.