EE232E - Graphs and Network Flows
Homework 1

Swati Arora: 404758379
Vishank Bhatia: 304758488
Anshita Mehrotra: 904743371

Question 1: Creating random networks

Part (a)

We used *erdos.renyi.game()* of the igraph package in R to create random network graphs. Three random graphs were created, where the probability p for drawing an edge between two arbitrary vertices 0.01, 0.05 and 0.1 respectively. The number of nodes was specified as 1000. The degree distribution curves are depicted in the form of line graphs below:

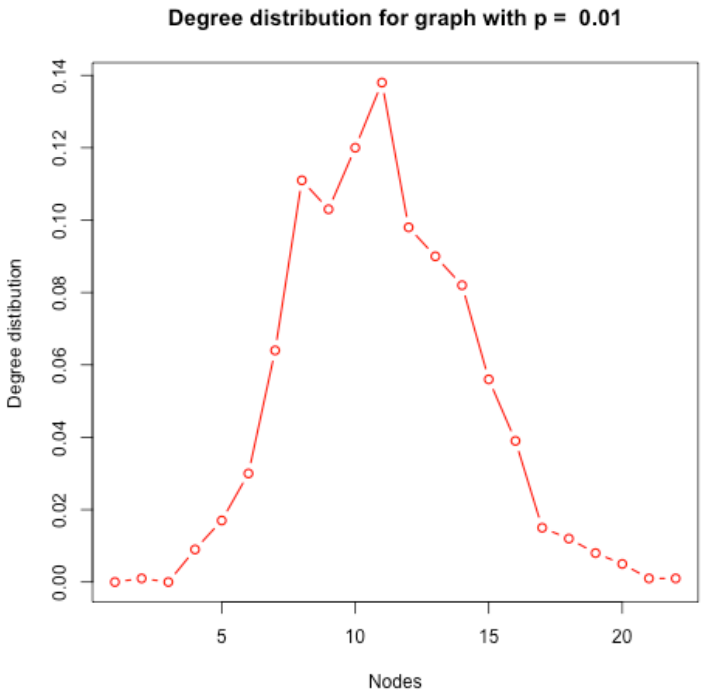

Figure 1: Degree distribution for graph with p = 0.01

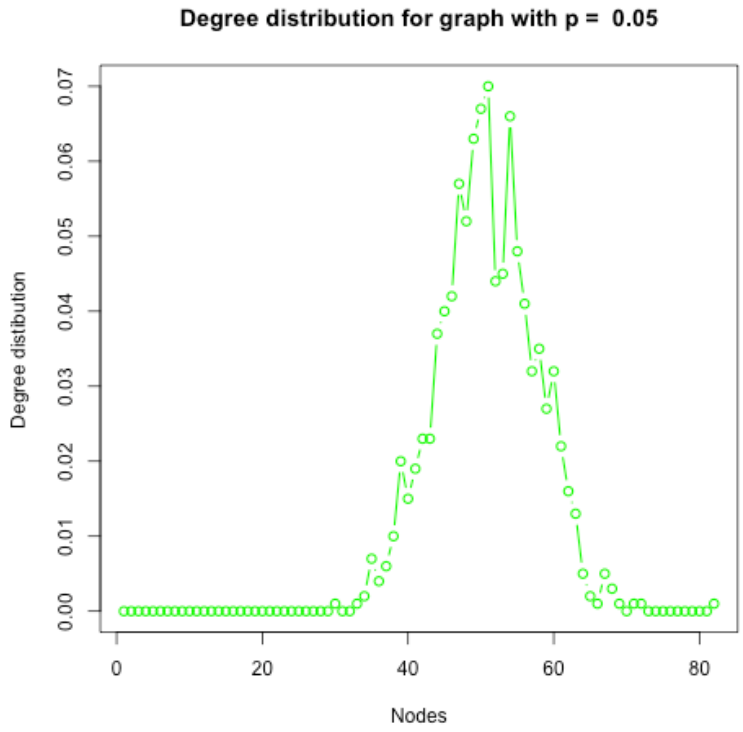

Figure 2: Degree distribution for graph with p = 0.05

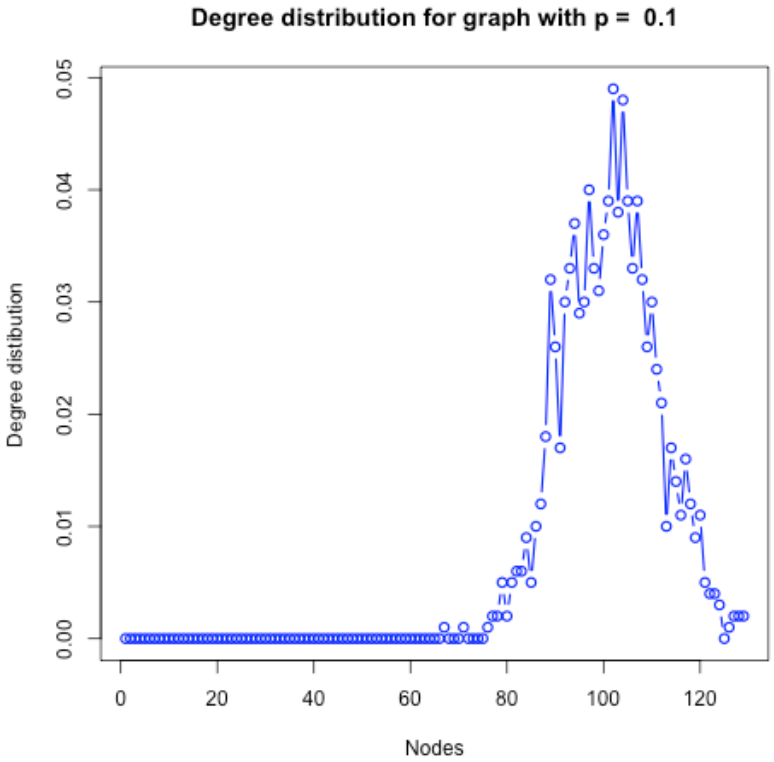

Figure 3: Degree distribution for graph with p = 0.1

The same graphs can also be obtained in terms of histogram as shown below:

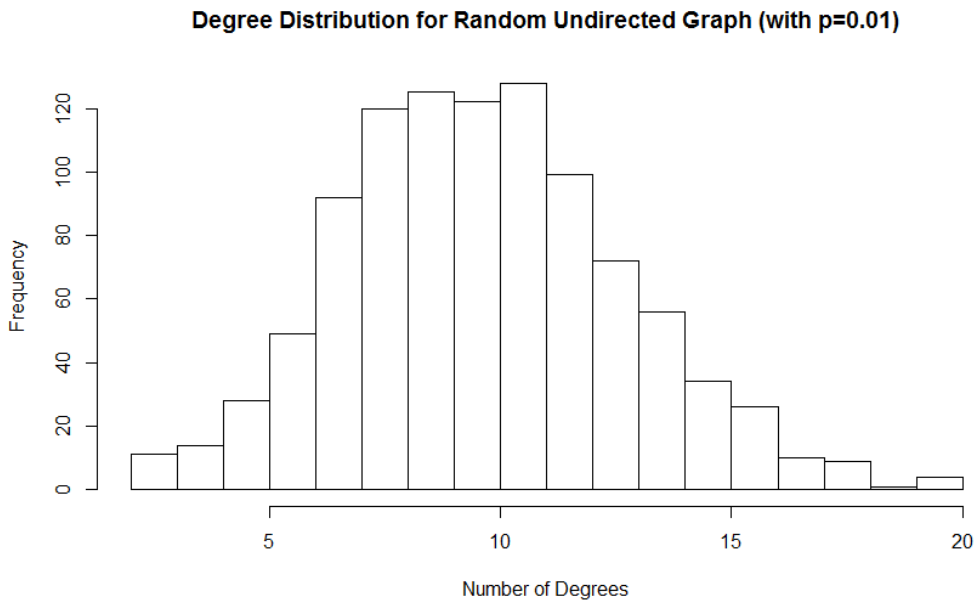

Figure 4: Degree distribution for graph with p = 0.01

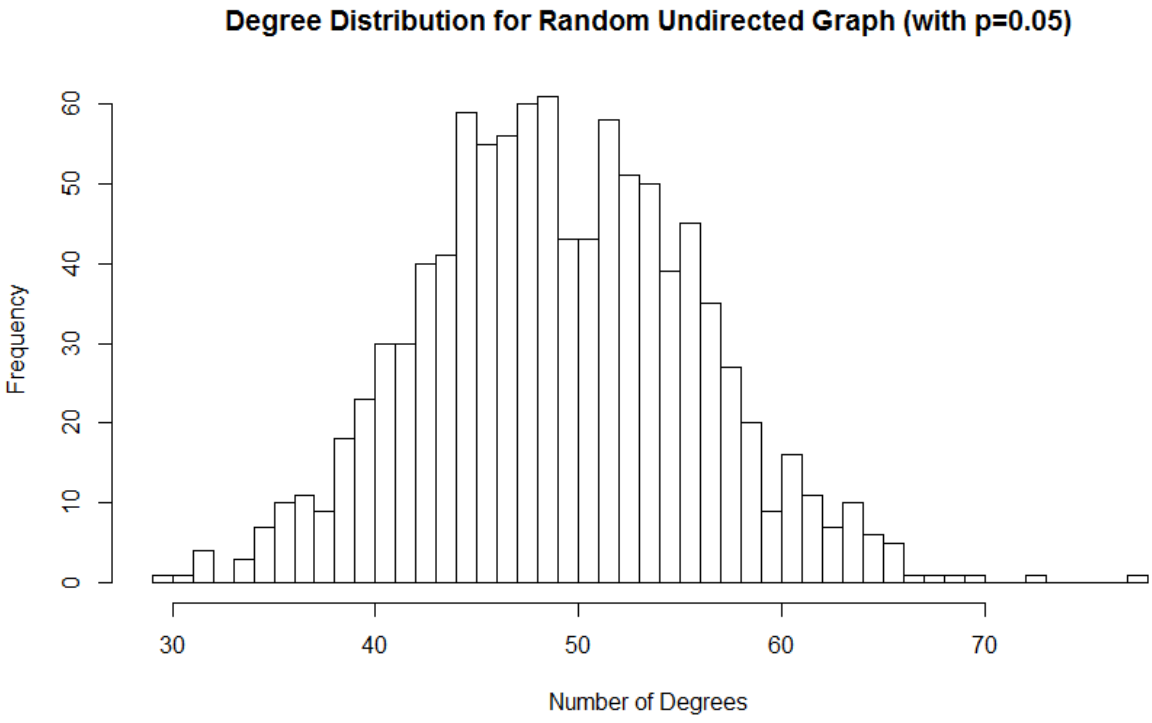

Figure 5: Degree distribution for graph with p = 0.05

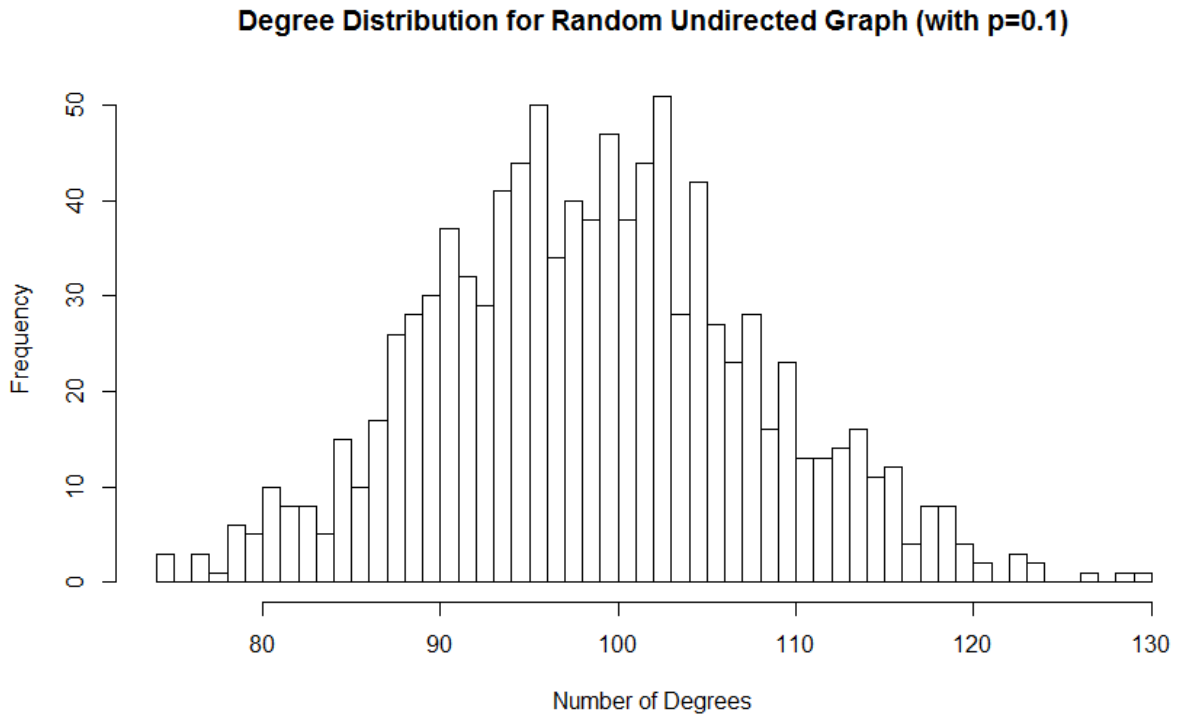

Figure 6: Degree distribution for graph with p = 0.10

Part (b)

Here, we find the diameters for the graphs generated using *diameter()* method and if they are connected or disconnected using *is.connected()* method. The result are as follows:

Table 1: Diameter and Connectivity of the random network graphs generated

Probability for drawing an edge between two random vertices	Diameter	Connectivity
0.01	6	Connected
0.05	3	Connected
0.10	3	Connected

Part (c)

Here, we find a value p_c (to three significant figures), so that when $p < p_c$ the generated random networks are disconnected, and when $p > p_c$ the generated random networks are connected.

For this we start with a probability of 0.010 (from part (b) we know that the graph is connected at this value) and keep decreasing the value by 0.001 till we get the value at which the graph is disconnected. The result obtained is:

Threshold Probability (Pc) = 0.0070

Part (d)

We can use the **Erdos-Renyi asymptotic expression** to determine the value of threshold probability. The random network graph g will be disconnected, if the value of link density p is less than the threshold P_c.

$$P_c \sim \log(N) / N \qquad [p > P_c]$$

$$P_c \sim \log(1000)/ 1000 = \mathbf{0.0069}$$

The value obtained is very close to the value obtained in part (c).

Question 2: Creating a fat-tailed degree distribution network

An undirected network having 1000 nodes with fat-tailed degree distribution with degree distribution proportional to x^{-3} is created using *barabasi.game()* method and *sample_degseq()* method of igraph package in R. The degree distribution has been shown below:

Part (a)

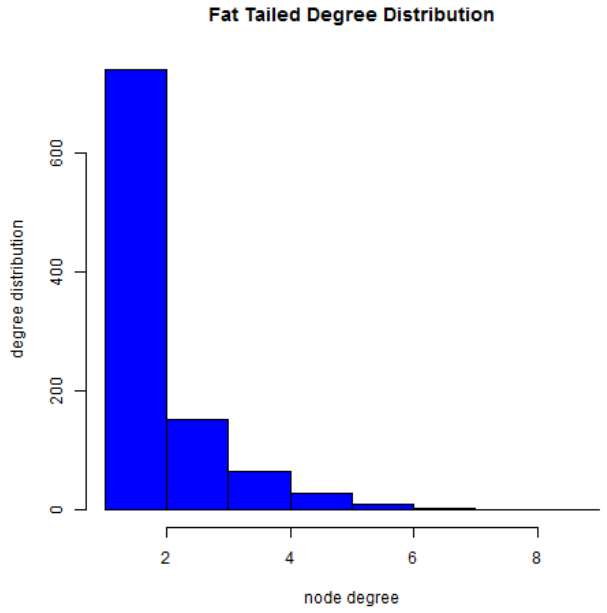

Figure 7: Fat Tailed Degree Distribution using BA model

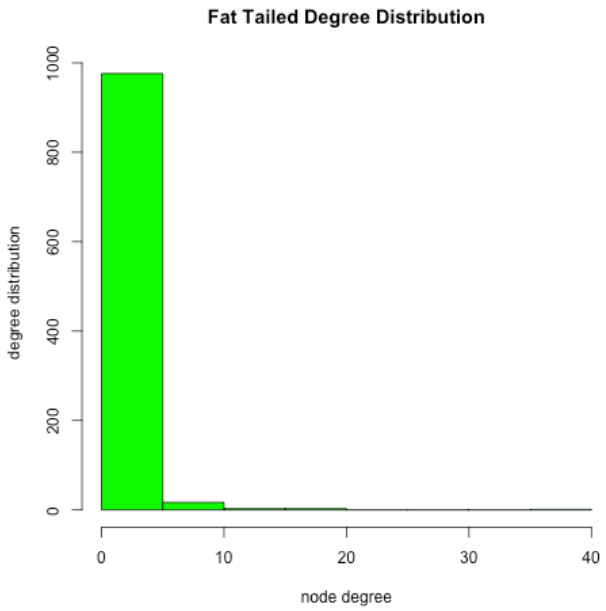

Figure 8: Fat Tailed Degree Distribution using Degree Sequence

Now, to calculate the diameter of the graph, we use the *diameter()* method of igraph package.

Table 2: Diameter of the graph

Method	Diameter
barabasi.game()	31
sample_degseq()	15

Part (b)

(i) The connectivity of the network can be determined by using *is.connected()* method. **The connectivity of the fat tailed degree distribution network is found to be TRUE i.e. the network is connected.**

(ii) Since there exists a path between any two randomly selected nodes in the network, **the Giant Connected Component (GCC) of the network is the whole network**.

(iii) The community structure of the graph is determined using *fastgreedy.community()* method and the corresponding modularity is measured using *modularity()* method. The modularity of the graph is a measure of the concentration of the connection between nodes. The networks having high modularity consist of nodes that have dense connections with the nodes of the same module while sparse connections with the nodes of other modules. **The modularity of the graph is 0.936.**

(iv) The high modularity of the network stems from the fact that barabasi model uses preferential attachment of nodes i.e. new nodes inserted into the network are linked to the already existing nodes having higher degree. Hence, the nodes in the graph have dense connections with other nodes in the same module but sparse connections with nodes in the other modules.

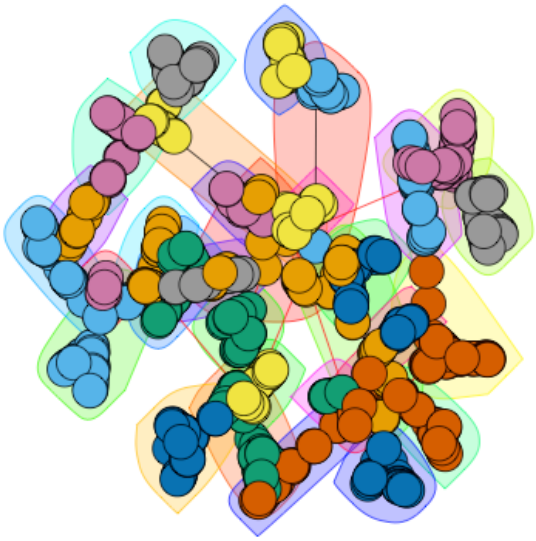

Figure 9: Community Structure using BA model

Figure 10: Community Structure using Degree Sequence

Table 3: Connectivity, Modularity and Number of Communities

Method	Modularity	Communities	Connectivity
barabasi.game()	0.9793	32	True
sample_degseq()	0.9079	237	False

Part (c)

For a larger network with 10000 nodes, there has been a slight increase in the modularity in this larger network as compared to the smaller network generated above with 1000 nodes which can be attributed to the higher density due to more nodes.

Table 4: Modularity and Number of Communities

Method	Modularity	Communities
barabasi.game()	0.9790	104
sample_degseq()	0.9307	2245

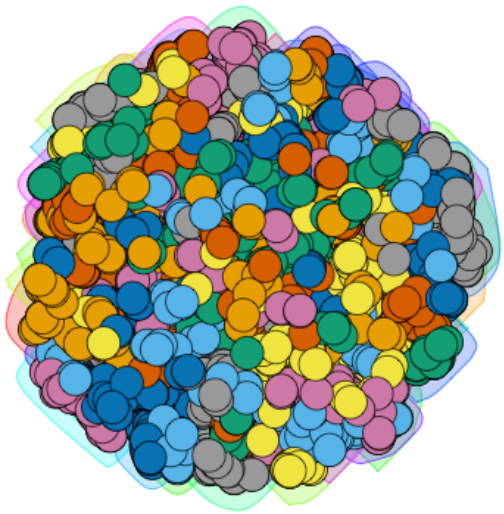

Figure 11: Community Structure using BA model

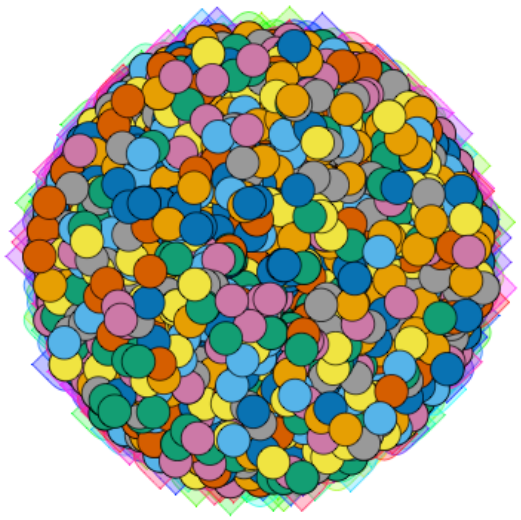

Figure 12: Community Structure using Degree Sequence

Part (d)

We randomly pick a node i and then randomly select its neighbor j. This is done by running a loop for all the nodes in the network and then plotting the degree distribution of the nodes j that are picked in the process. This is done by two methods – *barabasi.game()* and *sample_degseq()*.

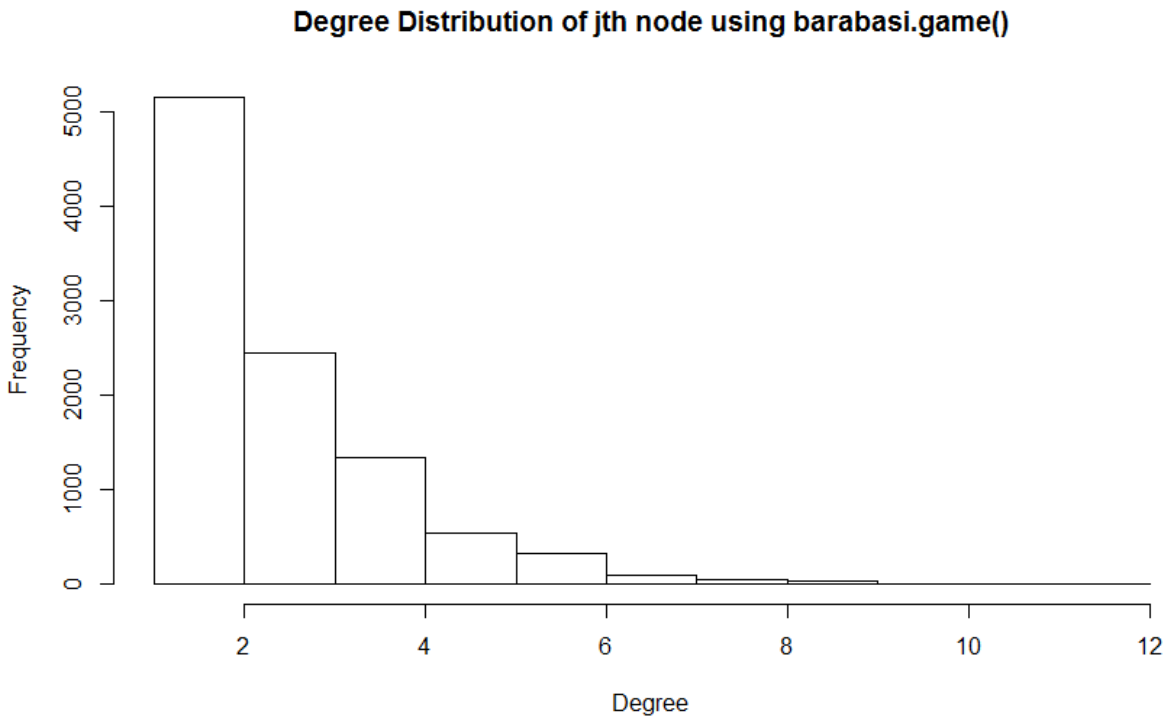

Figure 13: Degree Distribution of j^{th} node using BA model

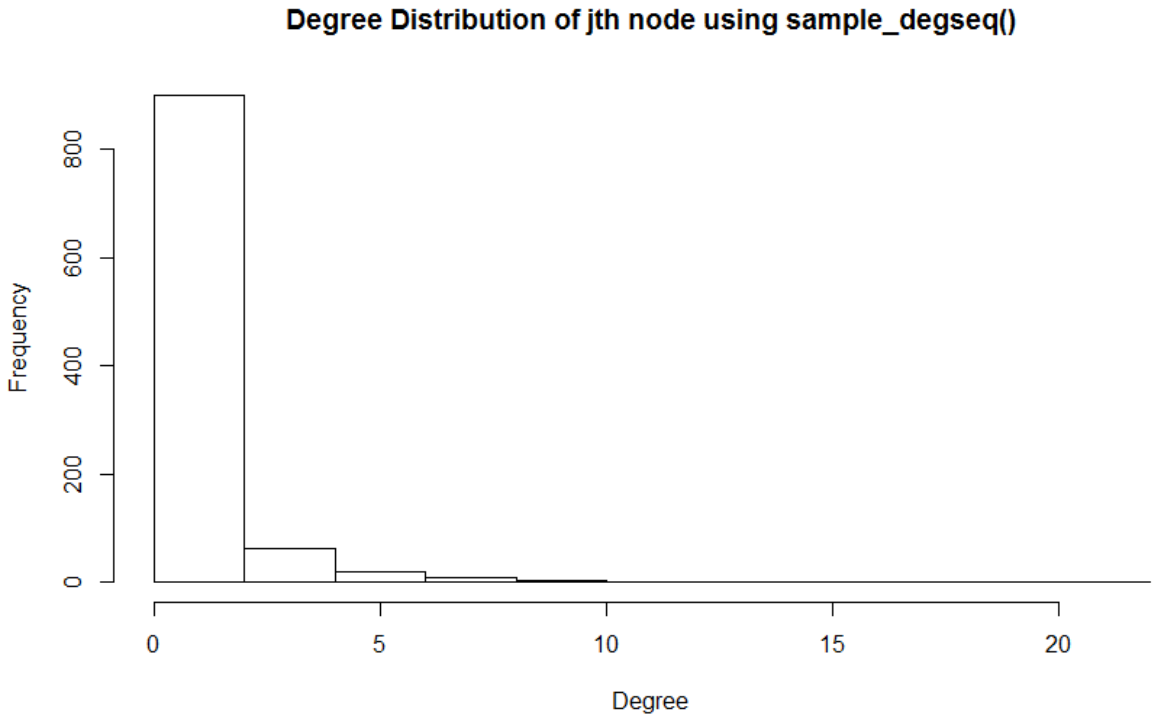

Figure 14: Degree Distribution of j^{th} node given degree sequence

Question 3: Random graph by simulating its evolution

In order to generate graphs that are simulated by its evolution, we use *aging.barabasi.game()* method. It uses a preferential attachment approach wherein each new node is added to the old nodes whose probability of being spotted depending on its degree and age.

Part (a)

The degree distribution of evolving graph is as follows:

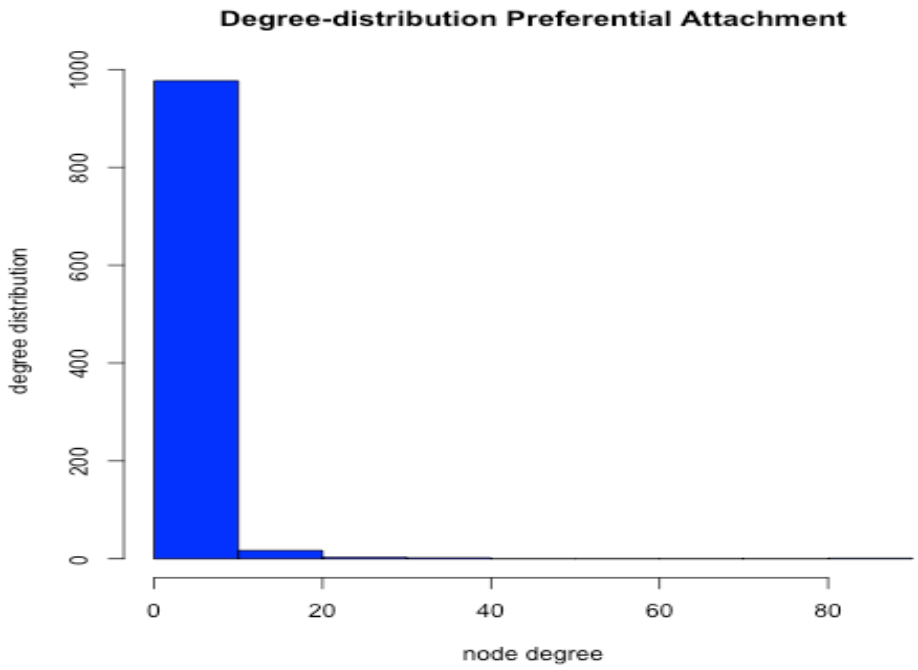

Figure 15 : Degree distribution of evolving graph

Part (b)

The community structure was computed using *fastgreedy.community()* method and modularity for it is computed using *modularity()* method.

Modularity of graph: 0.915

Community cluster = No of communities: 33

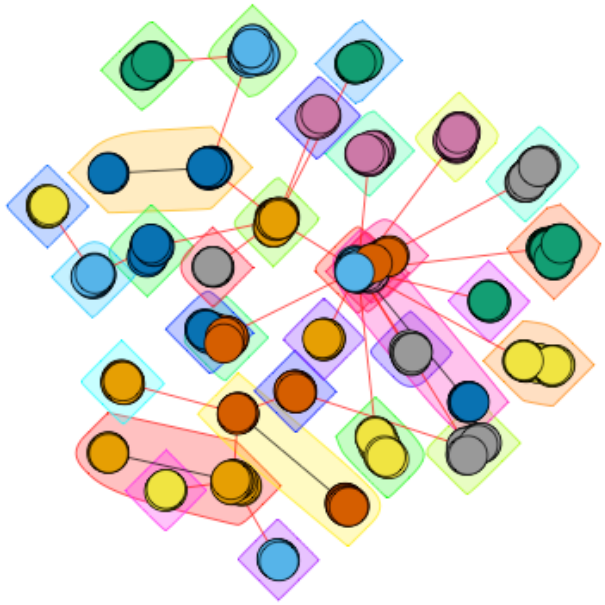

Figure 16 : Community Structure of evolving graph

Question 4: Forest Fire Model To Create Directed Network

In order to generate growing network model resembling forest fire, *forest.fire.game()* method was used.

Part (a)

In and Out degree distribution of growing network using Forest Fire Model is as follows:

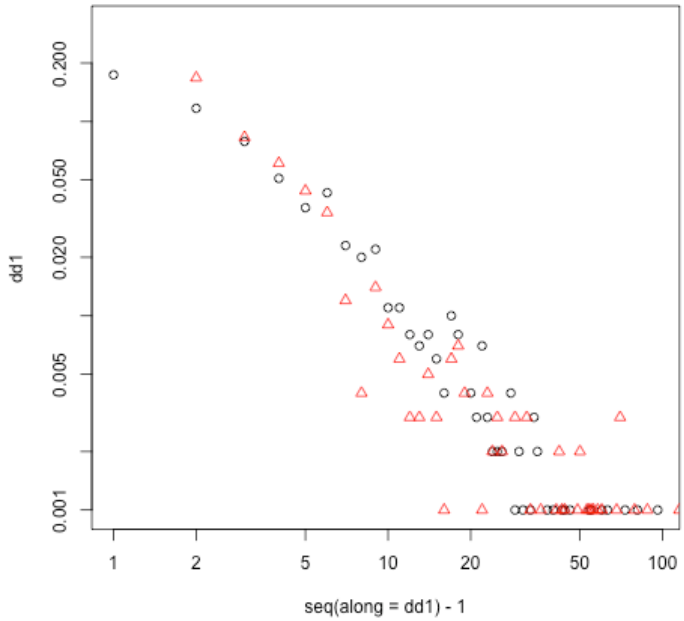

Figure 17 : In and Out Degree distribution for Forest Fire Network

In degree distribution for graph with 1000 nodes:

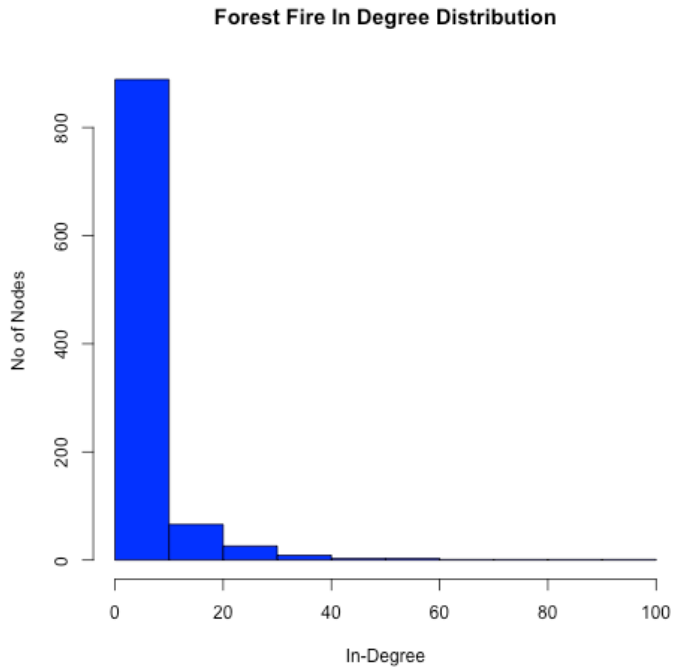

Figure 18 : Histogram showing In-Degree distribution

Out degree distribution for graph with 1000 nodes:

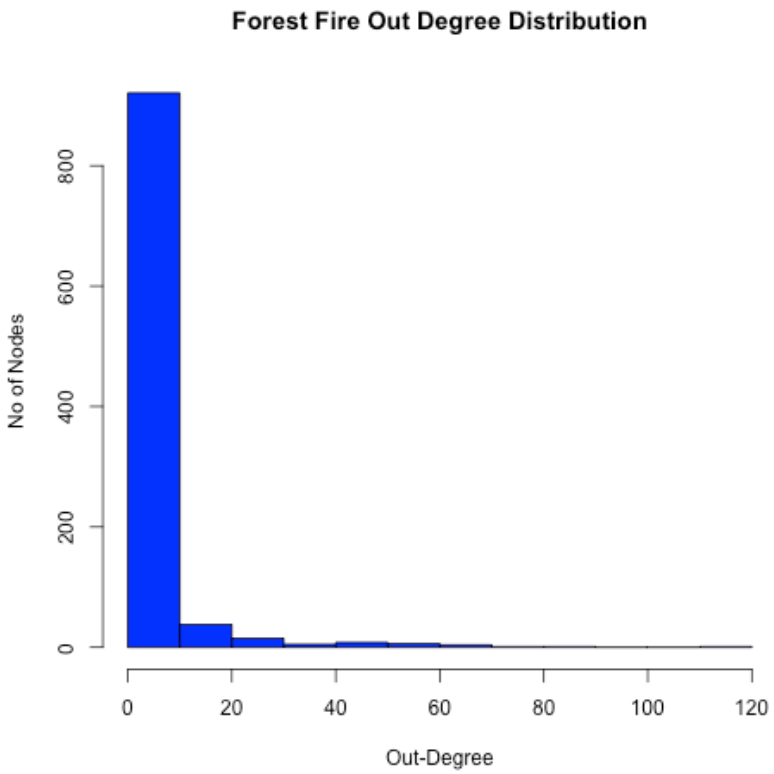

Figure 19 : Histogram showing Out-Degree distribution

Part (b)

To obtain the diameter of the graph, *diameter()* method was used

Diameter of Forest Fire Graph: 11

Part (c)

The community structure was computed using *fastgreedy.community()* method and modularity for it is computed using *modularity()* method.

Modularity of the directed network is: 0.445

No. of communities: 24

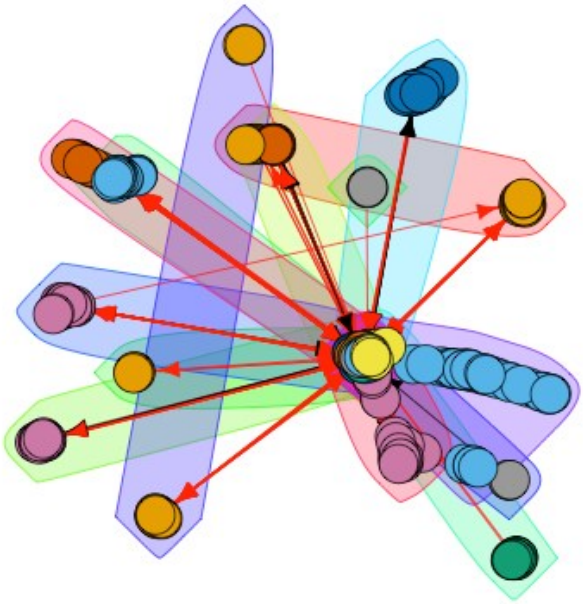

Figure 20 : Community Structure of Forest Fire graph