ECE 232E Project 1
Random Graphs and Random Walks

Part I

1. Create random networks using Erdo ¨s-R´enyi (ER) model

(a) We first create undirected networks with n=1000 nodes with probability for drawing an edge between two arbitrary vertices 0.003, 0.004, 0.01, 0.05 and 0.1. The degree distribution of each p is shown below.

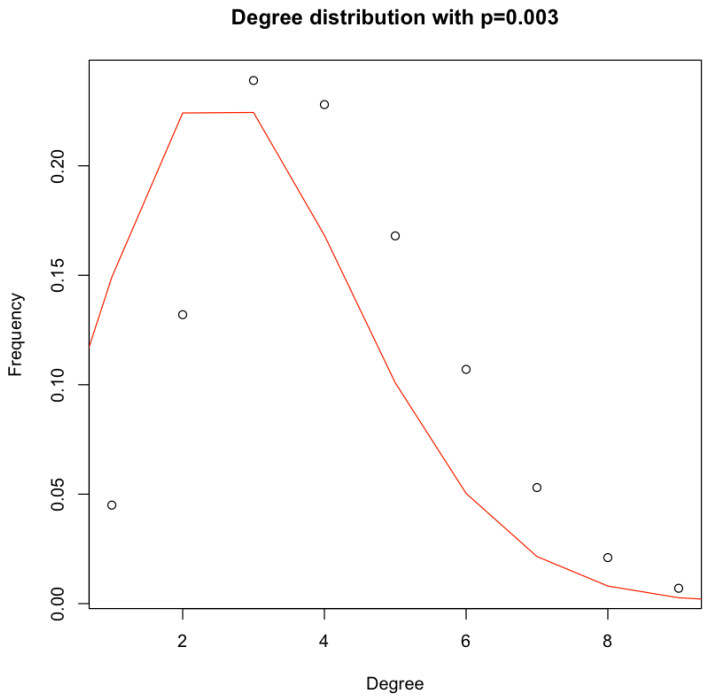


Figure 1 Degree Distribution with p=0.003

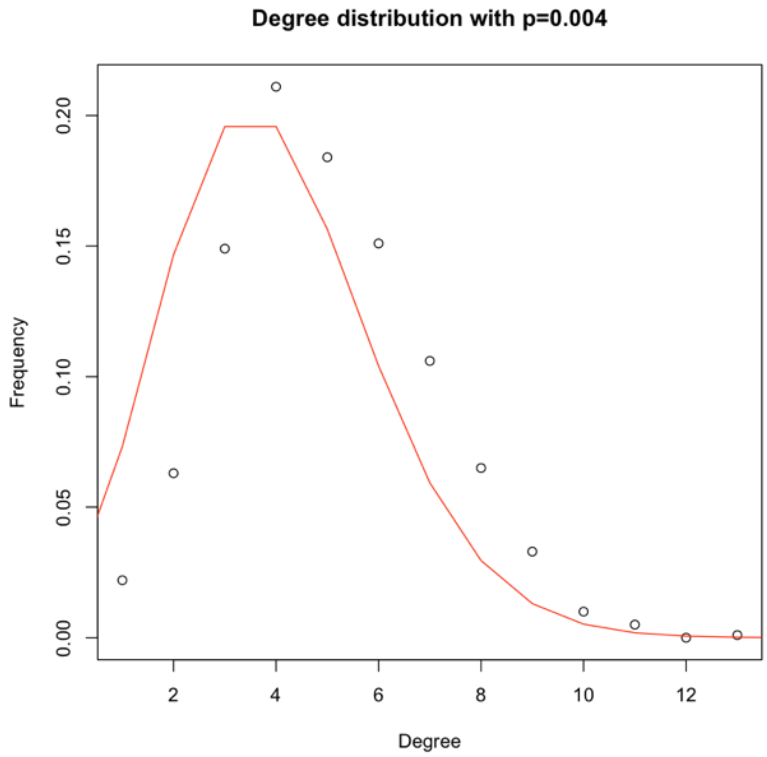


Figure 2 Degree Distribution with p=0.004

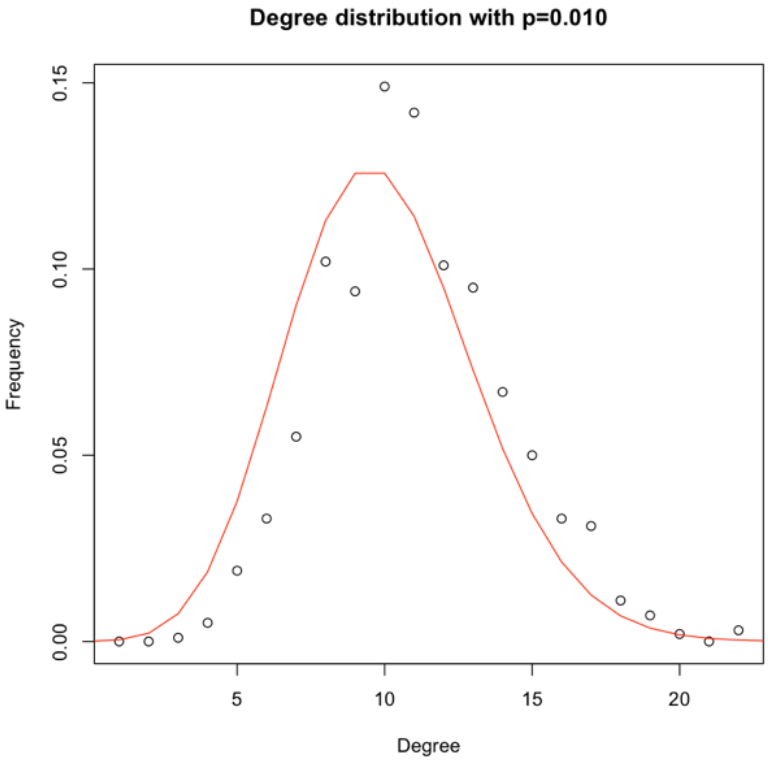


Figure 3 Degree Distribution with p=0.010

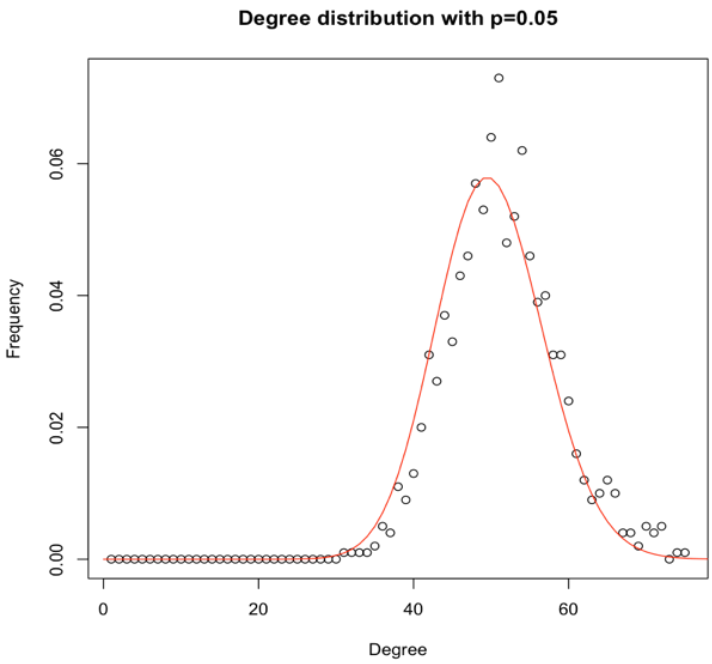


Figure 4 Degree Distribution with p=0.05

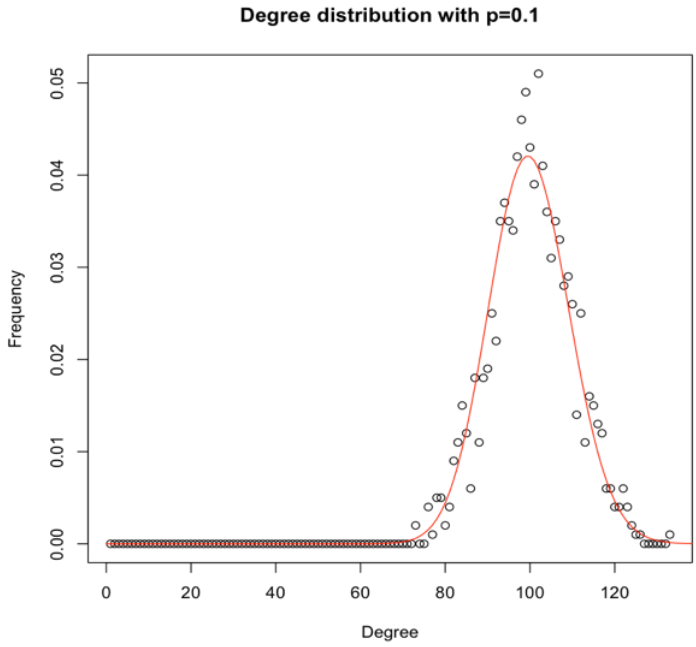


Figure 5 Degree Distribution with p=0.1

From the figures above, we can find that the distribution is binomial distribution. The node connects with another node with probability p, which satisfies binomial distribution, K~B(N-1,p). The theoretical mean and variance can be calculated with the formulas:

$$E(k) = (N-1) * p$$

$$\sigma^2 = (N-1) * p * (1-p)$$

The theoretical results and empirical results are shown in table 1. And they are match.

Table 1. Mean and Variance of the Degree Distributions

p	Theoretical Mean	Empirical Mean	Theoretical Variance	Empirical Variance
0.003	2.997	2.952	3.14	2.936632632632 63
0.004	3.996	3.992	4.16	3.911847847847 85
0.01	9.99	10.042	10.02	9.485721721721 72
0.05	49.95	50.458	45.38	49.798034034034
0.1	99.9	99.392	85.57	90.198534534534 5

(b) Then we analyze the graphs built above from the connection. Table 2 contains information whether the graph is connected, the probability it is connected, and GCC diameter if not connected.

Table 2. Connection and GCC diameter

p	Probability	Connected	GCC diameter
0.003	0	False	14
0.004	0	False	11
0.01	0.96	False	5
0.05	1	True	Connected
0.1	1	True	Connected

(c) It turns out that the normalized GCC size is a highly nonlinear function of p, with interesting properties occurring for values where p = O(1/n) and p = O(ln(n)/n). First, we calculate 1/n and ln(n)/n separately, which is 0.001 and 1n1000/1000=0.069. The scatter plot and line plot for p from 0.003 to 0.01 with step 0.0001 is shown in Figure 6.

First, we draw the plot from 0 to 0.003 with step 0.0001 shown in Figure 7. Then we further take a look for each range, [0,0.001], [0.001,0.002] and [0.002,0.003].

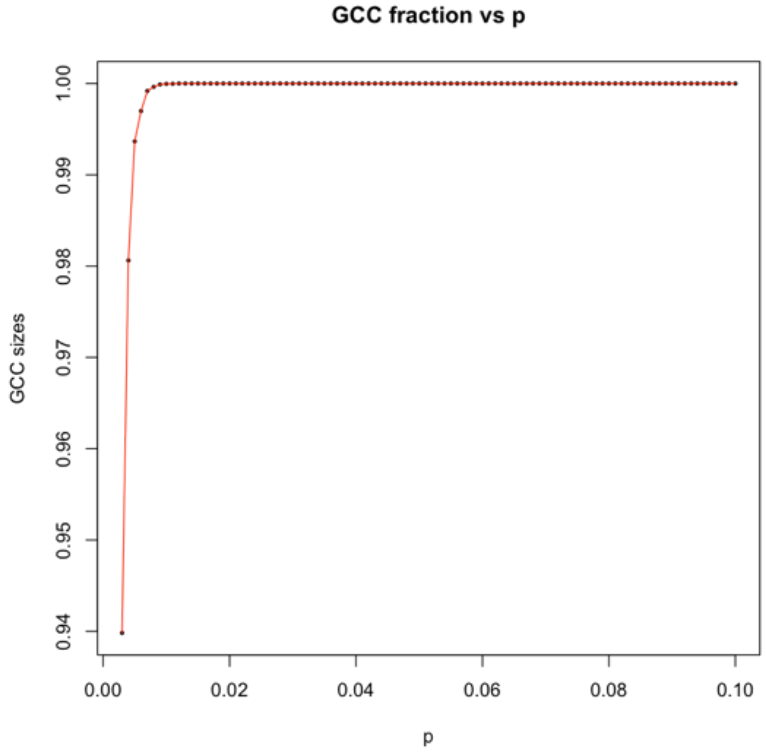


Figure 6 GCC fraction vs p with n = 1000

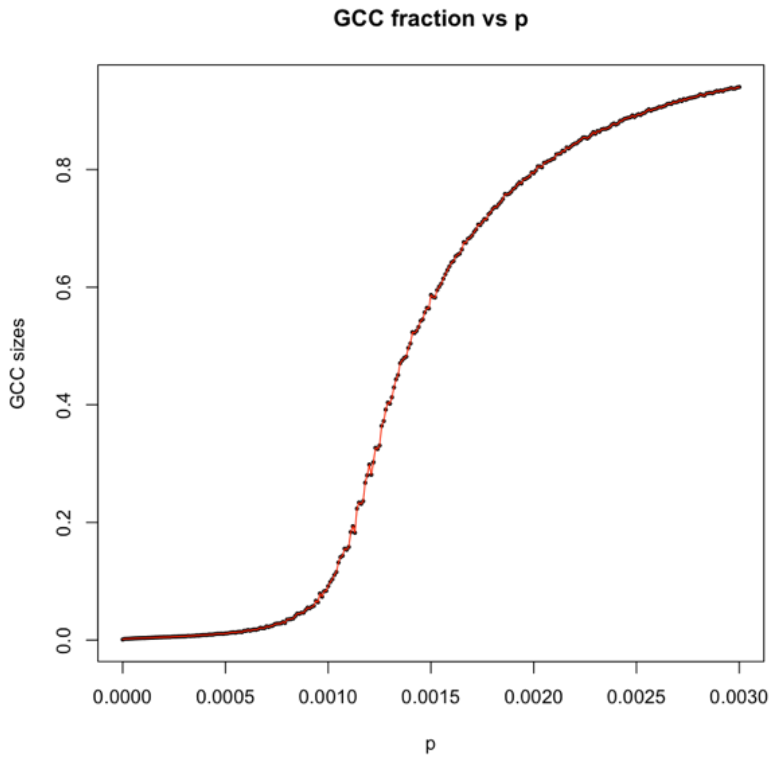


Figure 7(a) GCC fraction vs p with n = 1000 (O(1/n))

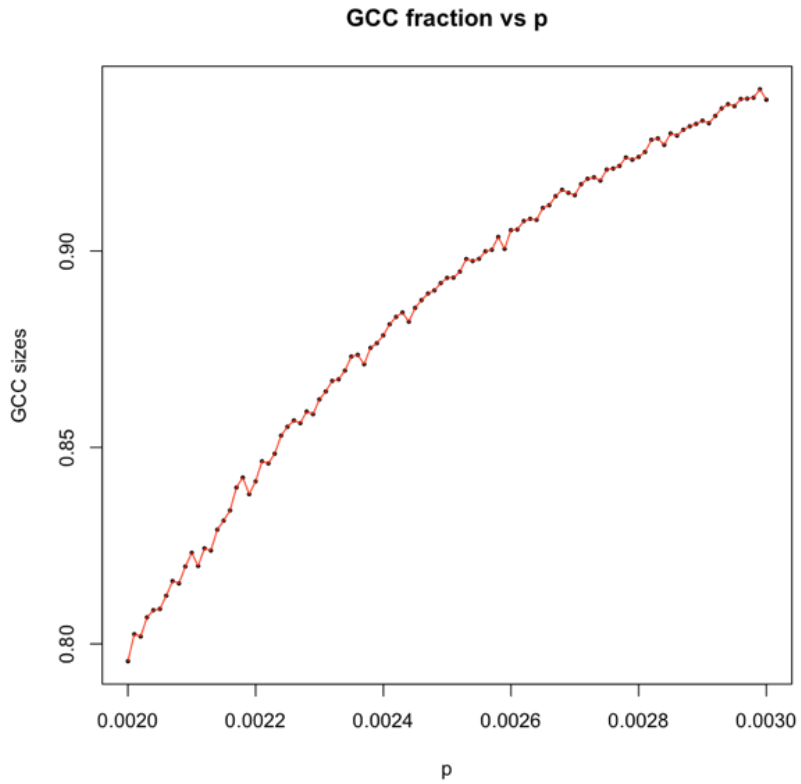


Figure 7(b) GCC fraction vs p with n = 1000 (O(1/n))

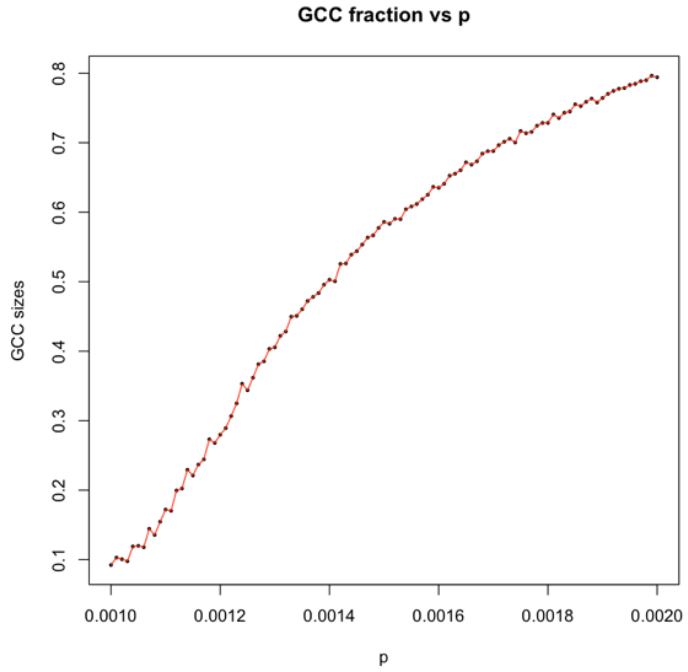


Figure 7(c) GCC fraction vs p with n = 1000 (O(1/n))

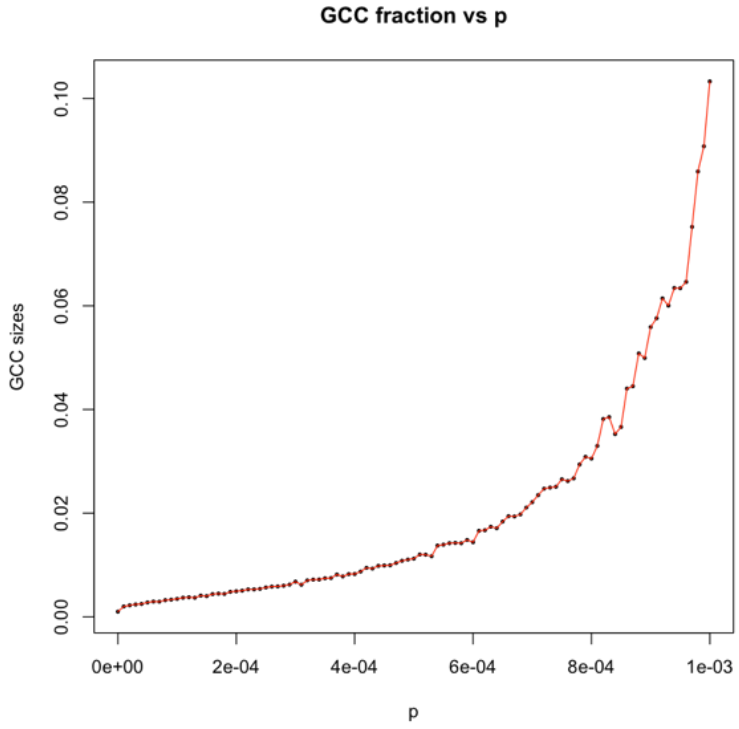


Figure 7(d) GCC fraction vs p with n = 1000 (O(1/n))

From figure 6 and 7, we can know that the growth rate is first getting bigger and then getting smaller. The turning point is 0.001. From p=0 to 0.001, the growth rate increases. After 0.001, we could find that the line between each range of 0.001 tend to be linear before it finally connects.

Then we focus on ln(n)/n, which is 0.0069. From figure 8 below, we can tell that GCC sizes reach a steady state around where p value is 0.007. It is also when a GCC starts to emerge. Here we define emergence as where the slope of the line close to 0 visually. The empirical result matches the theoretical one.

Finally, we estimate the value of p where the giant connected component takes up over 99% of the nodes in almost every experiment. From figure 8, we can see it should be in the range of [0.0045, 0.006]. Therefore, we further focus on the range and figure 9 gives the result between 0.0044 to 0.0056 with step 0.00001. The value of p should be around 0.0047.

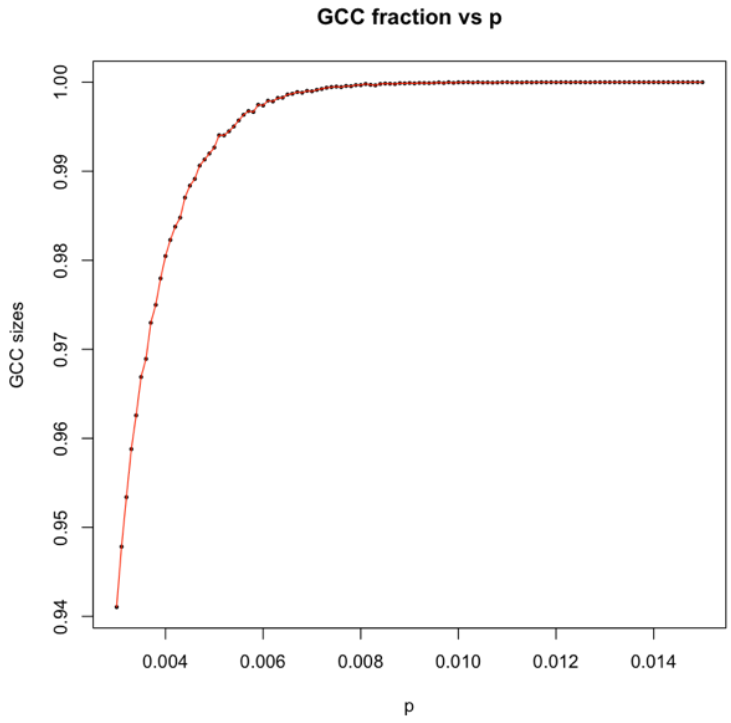


Figure 8 GCC fraction vs p with n = 1000 (O(ln(n)/n))

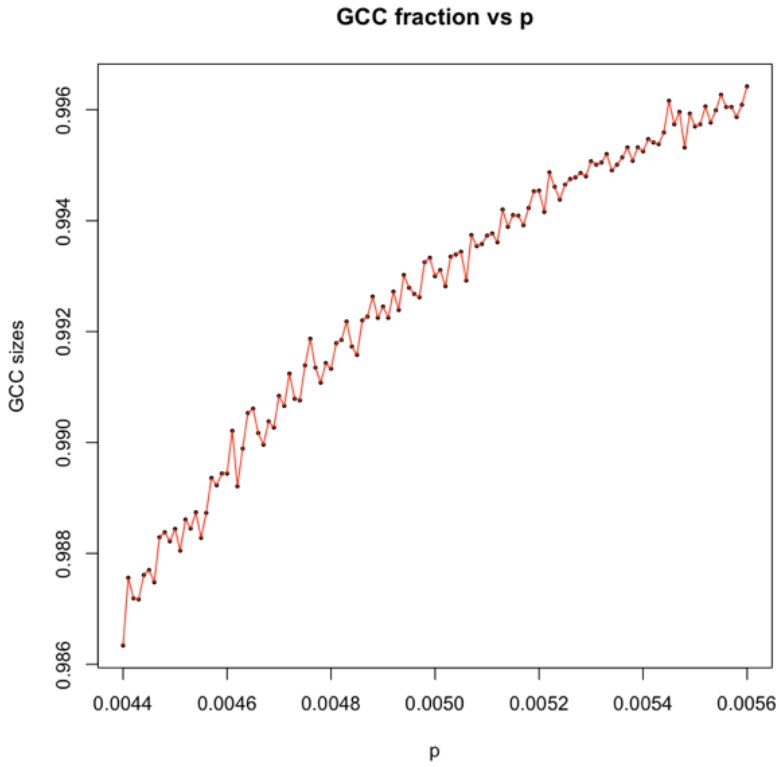


Figure 9 GCC fraction vs p with n = 1000 for 99% connected

(d) Define the average degree of nodes c = n × p = 0.5. Sweep over the number of nodes, n, ranging from 100 to 10000. We plot the expected size of the GCC of ER networks with n nodes and edge-formation probabilities p = c/n, as a function of n, which is shown in Figure 10. We can observe that GCC size increase as n increases. The slope is getting smoother as n increases, indicating that the growth rate becomes smaller. When n is large enough (around 5000), the relation is linear.

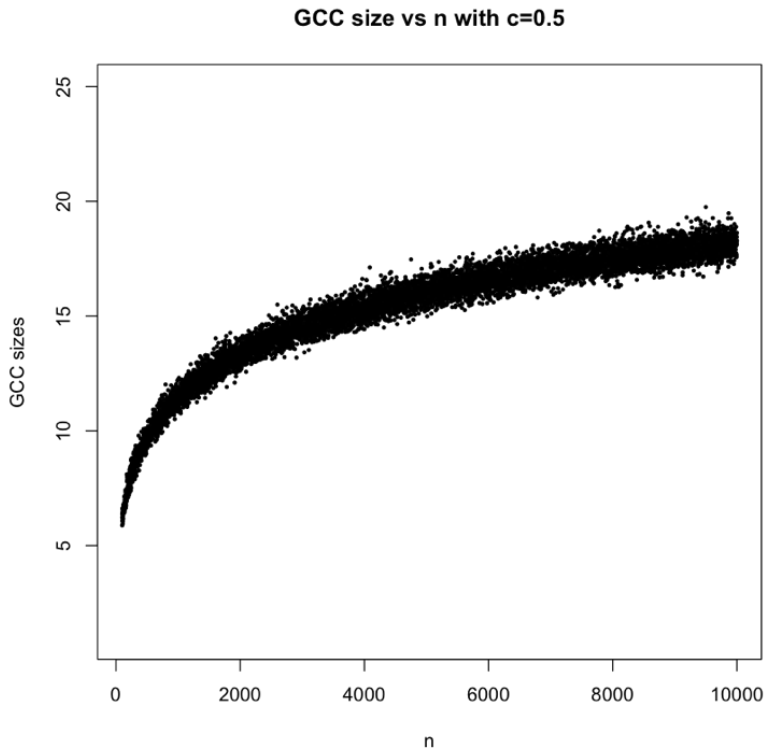


Figure 10 GCC size with c=0.5

Then we repeat the experiment with c=1.0. GCC size increase as n increases. The slope is getting smoother as n increases. When n is large enough (around 3000), the relation is linear. We can find that the line becomes linear at a earlier stage.

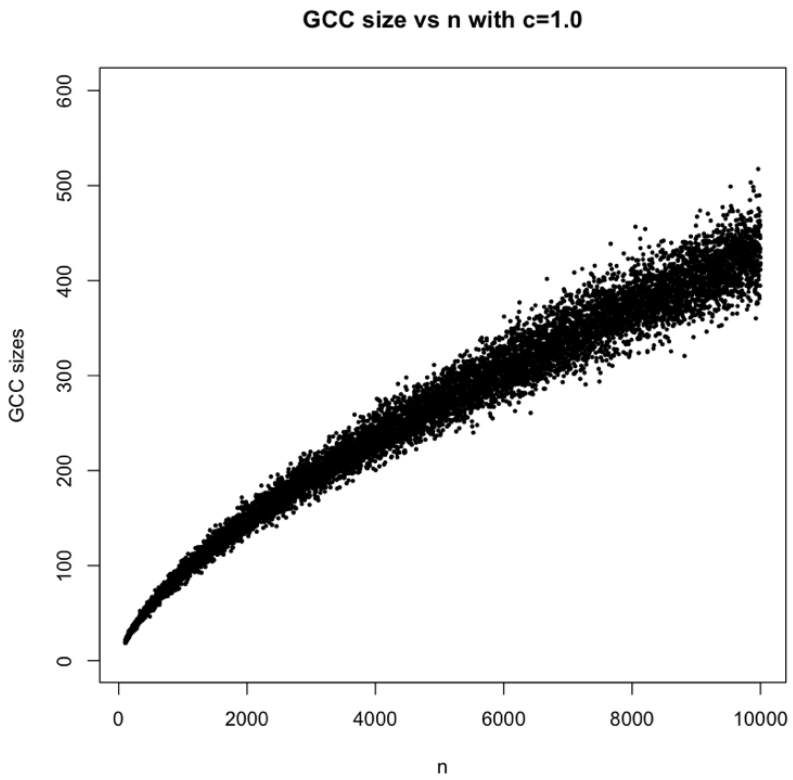


Figure 11 GCC size with c=1.0

Finally, we repeat the same for values of c = 1.1, 1.2, 1.3, and show the results for these three values in a single plot in Figure 12. When c=1.1, 1.2, 1.3, the relationship between GCC size and n is linear. Also, GCC sizes become large when c is larger for the same n. This is because p = c/n, larger c indicates larger p, making more nodes connected.

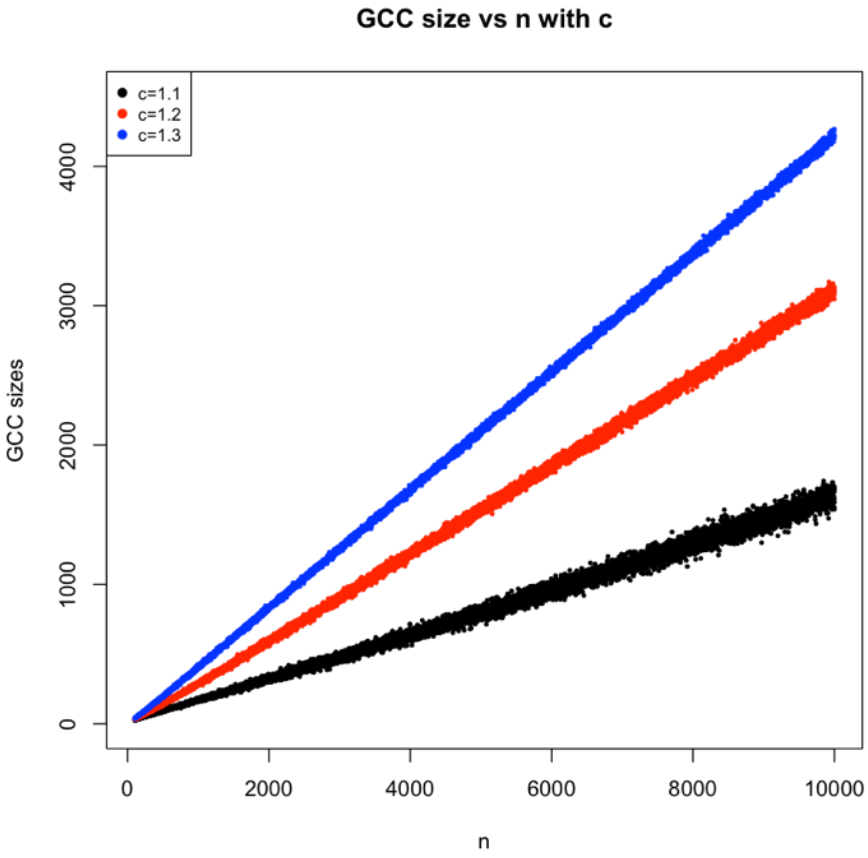


Figure 12. GCC size with c=1.1, 1.2, 1.3

2. Create networks using preferential attachment model

(a) According to the requirements in the question, we use *barbasi.game* function to create an undirected network with 1000 nodes and each new node attaches to 1 old nodes with preferential attachment model.

We also use *is.connected()* to check if this network is always connected, and the answer is yes. Besides, the preferential attachment process means that discrete nodes are added in a random fashion to a set of objects, and additional nodes are added continuously to the system and are distributed among the objects. Thus, such a network is always connected, which is shown below.

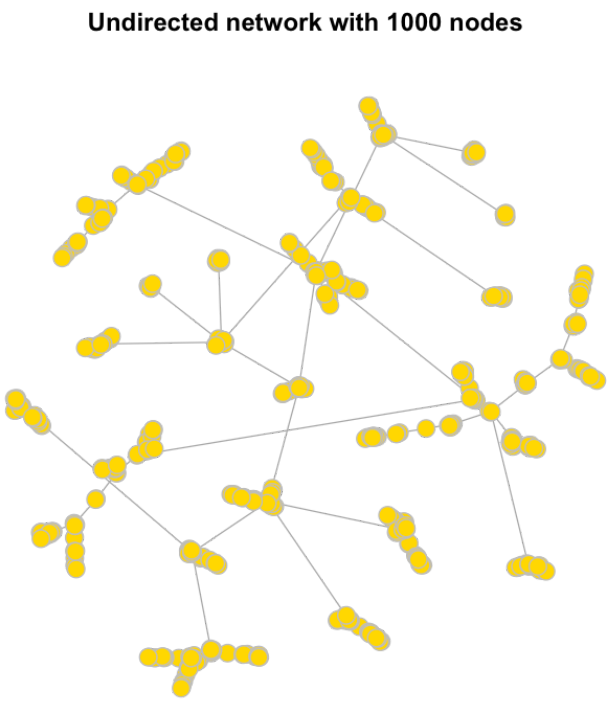


Figure 13. Undirected network with 1000 nodes and 1 old nodes attached

(b) “A network is said to have community structure if the nodes of the network can be easily grouped into sets of nodes such that each set of nodes is densely connected internally.” We use *cluster_fast_greedy* function that tries to find dense subgraph, also called communities in graphs via directly optimizing a modularity score to find communities.

Modularity is designed to measure the strength of division of a network into communities. With *modularity()* function, we can get the modularity value of this network. The results are shown below:

Table 1. Number of community and modularity of the network (n = 1000, m = 1)

Number of Community	Modularity
31	0.933996559121687

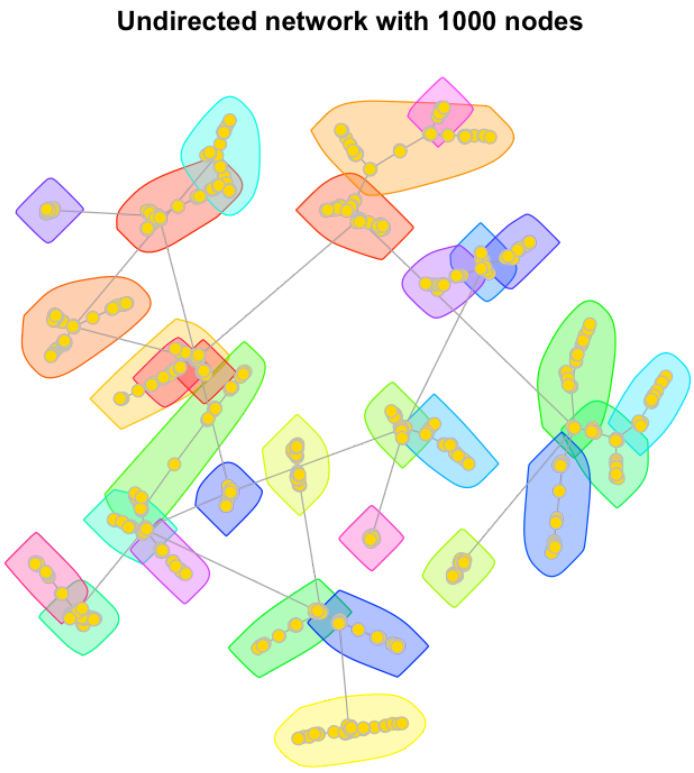


Figure 14.. Community structure of undirected network (n = 1000, m = 1)

(c) Similar to what we did in (a) and (b), an undirected network with 10000 nodes with the same preferential attachment module and its community structure and modularity are shown below:

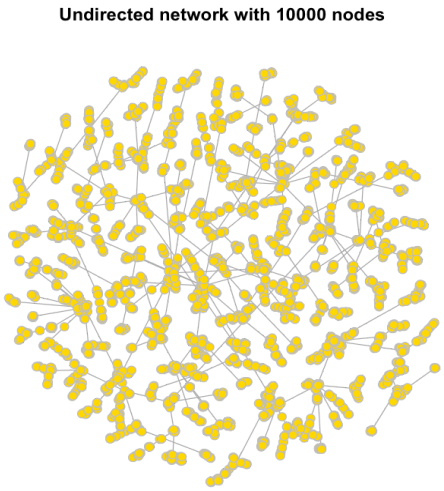


Figure 15. Undirected network with 10000 nodes and 1 old nodes attached

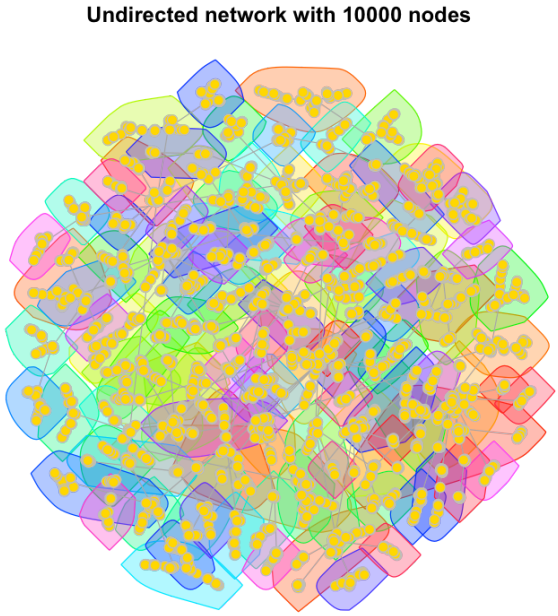


Figure 16. Community structure of undirected network (n = 10000, m = 1)

Table 2. Number of community and modularity of the network with different nodes

Number of node	1000	10000

Number of Community	31	107
Modularity	0.933996559121687	0.978333871991079

Compared to the smaller network's modularity, we can find out with the increase of nodes added to the network, there are more number of community and modularity is higher, which shows that network with higher modularity has denser connections between the nodes within communities but sparser connections between nodes in different communities.

(d) In complex networks, the degree of a node in a network is the number of connections it has to other nodes and the degree distribution is the probability distribution of these degrees over the whole network. We use *degree.distribution()* to find the intensity of distribution (frequency for each degree) in the network. The plots are shown below for 1000 and 10000 nodes in log-log scale:

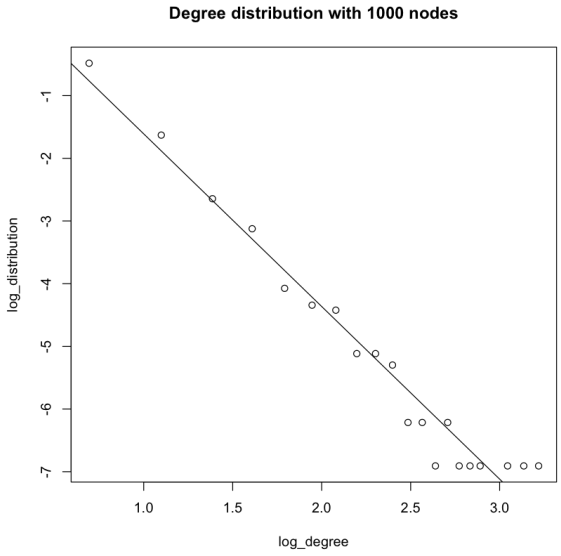


Figure 17. Degree distribution of undirected network (n = 1000, m = 1)

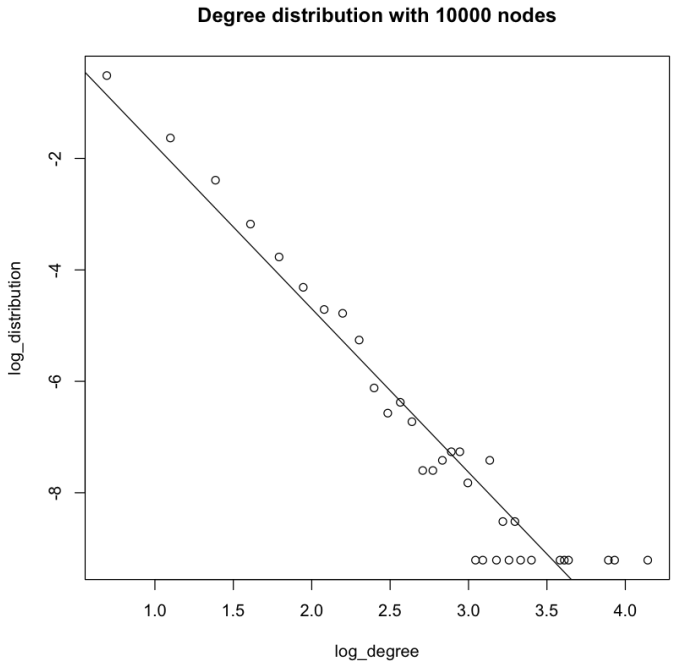


Figure 18. Degree distribution of undirected network (n = 10000, m = 1)

We also estimate the slope of the plot using linear regression. We have bivariate data (X, Y), which is (log_degree, log_distribution) in our case, and assume the regression function is linear, we can compute the slope with the formula *cov(X, Y) / var(X)*. *cov()* is the covariance function that forms the variance-covariance matrix, and *var()* can compute variance and standard deviation of complex vectors and matrix. The slope results for the two plots of different networks are shown below:

Table 3. Slope results for the two plots of different networks

Number of node	1000	10000
Slope	-2.75444936811754	-2.93575439312794

(e) The two plots of degree distribution of a random picked node are show:

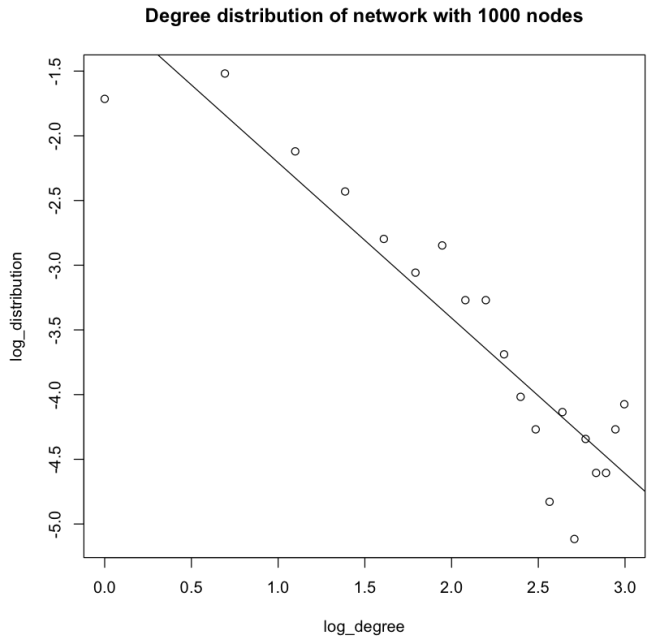


Figure 19. Degree distribution of random picked node (n = 1000, m = 1)

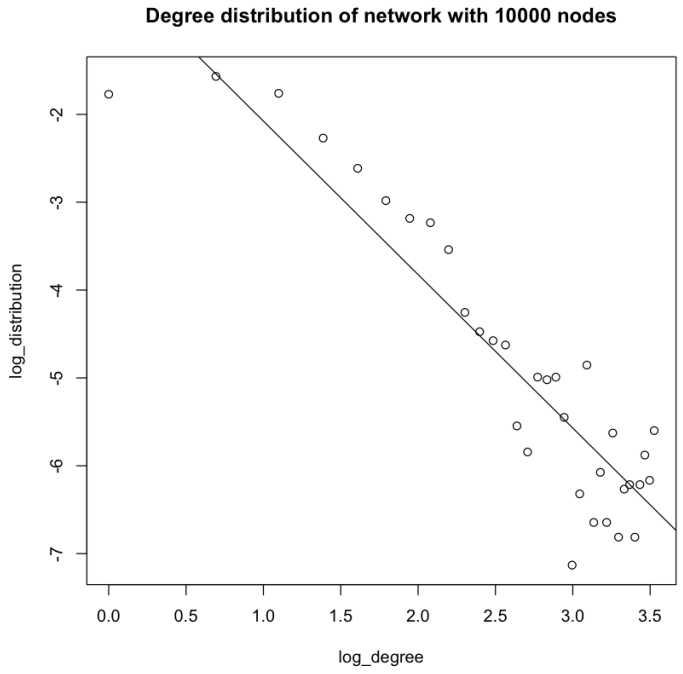


Figure 20. Degree distribution of random picked node (n = 10000, m = 1)

The distribution is linear in the log-log scale. The slope results are shown below:

Table 4. Slope results for the two plots of different networks

Number of node	1000	10000
Slope	-1.20119990330144	-1.74707715327595

In this process, a random node is picked and a random neighbor of that node is then picked. As a result, compared to the node degree distribution, the probability that the neighbor of the random picked node with high degree is picked will be higher, which will differ the result of degree distribution. Observing the plots in e) and f), we can see that in low degree area, the distribution of nodes is higher in node degree distribution with the same n. On the other hand, in high degree area, with the same n, the distribution of nodes is higher in degree distribution of a random picked node with its random neighbors.

(f) We randomly created an undirected network with n = 1000 nodes, with preferential attachment model, and the nodes are added at 1000 iterations. Then, for each node we average the sum of degree for 1000 iterations and get the relationship between the age of nodes and their expected degree. The relationship plot is shown below. From the plot, we can see that the expected degree is higher for the nodes whose age is older.

Figure 21. Age of nodes and their expected degree

(g) We repeat the parts that we created four undirected network with n nodes, with preferential attachment mode, where each new node attaches to m old nodes, and find community for these networks and measure modularity, which are shown as:

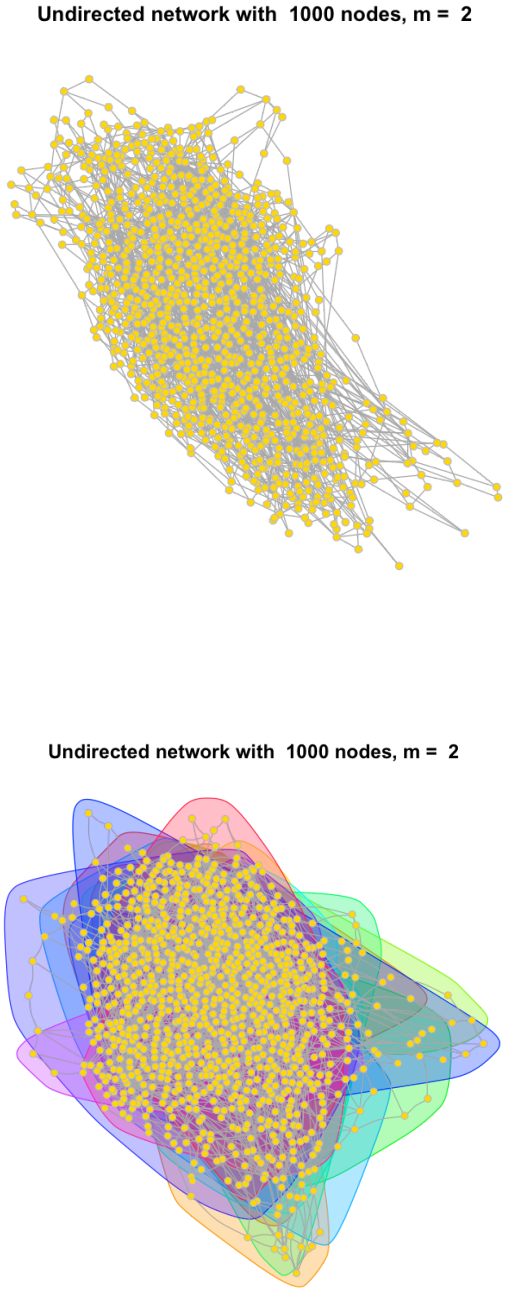


Figure 22. Undirected network with n = 1000 & m = 2 and community structure

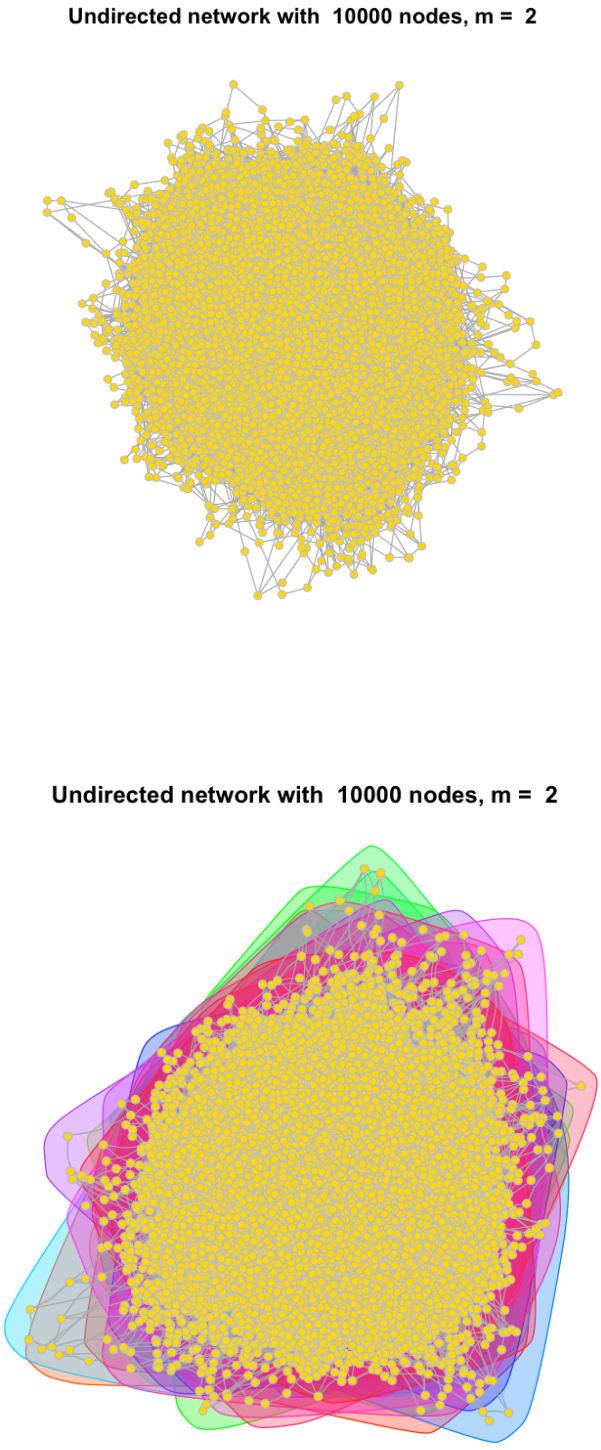


Figure 23. Undirected network with n = 10000 & m = 2 and community structure

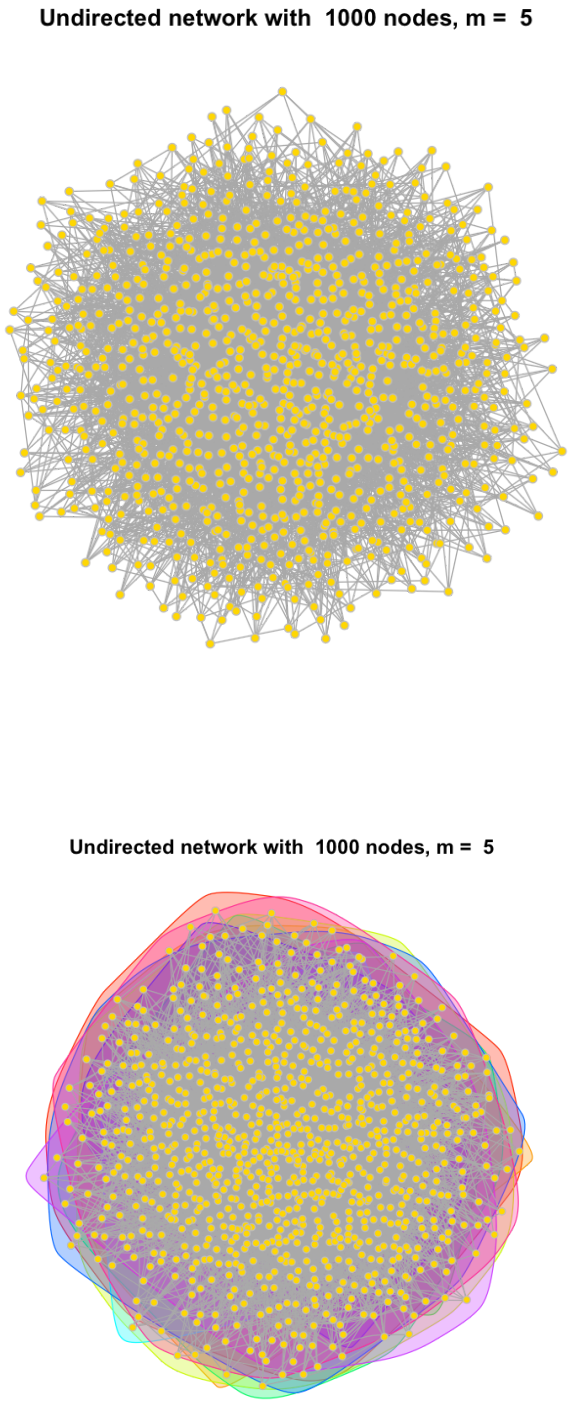


Figure 24. Undirected network with n = 1000 & m = 5 and community structure

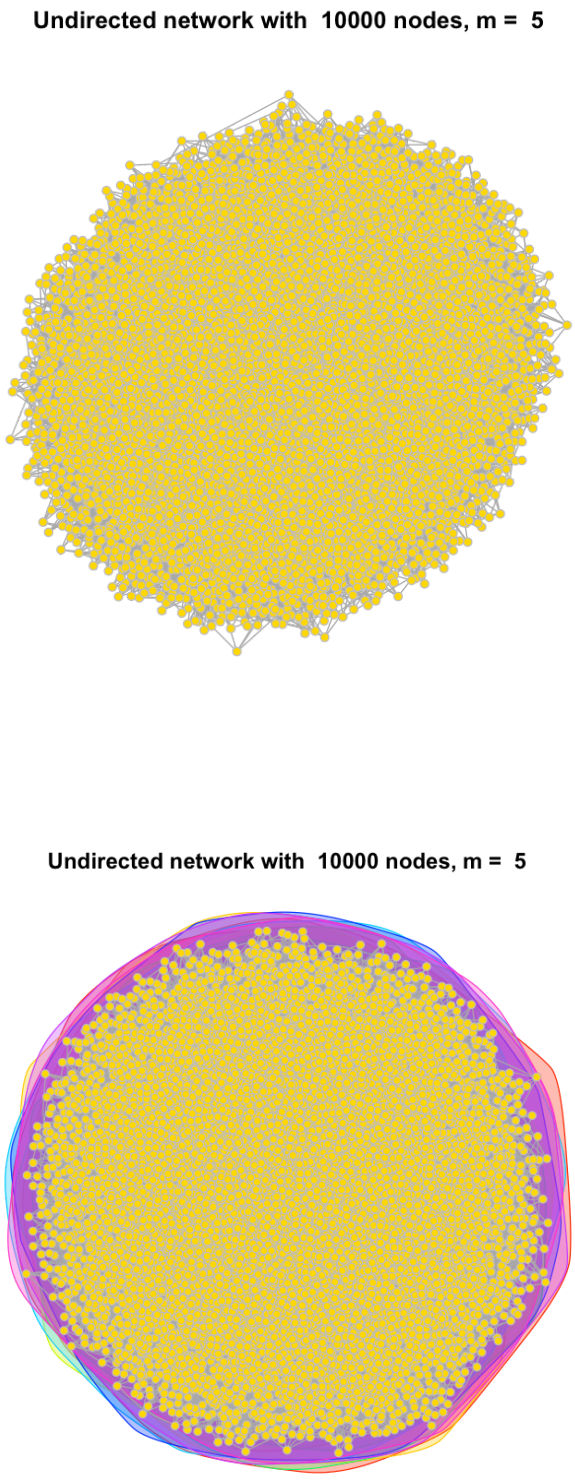
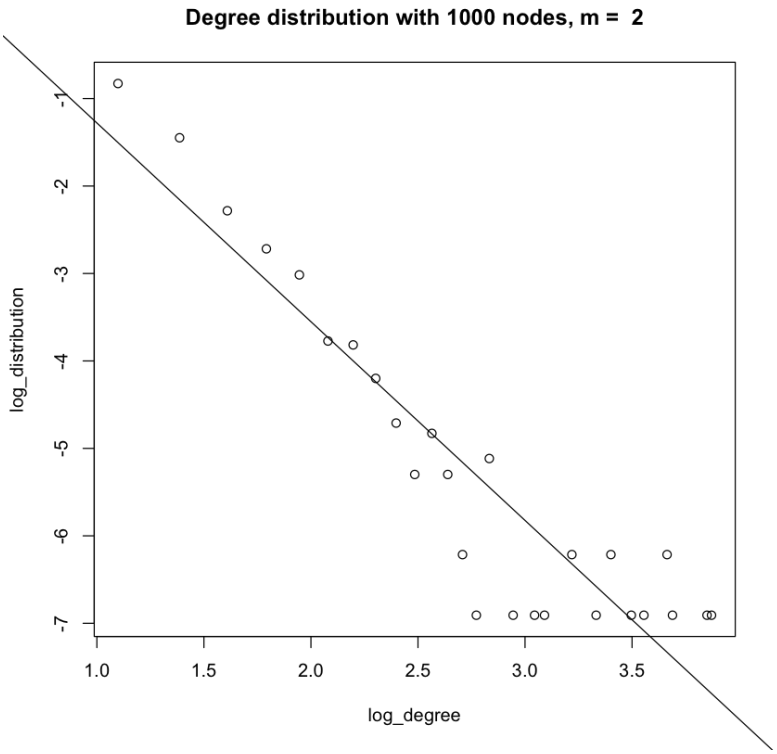
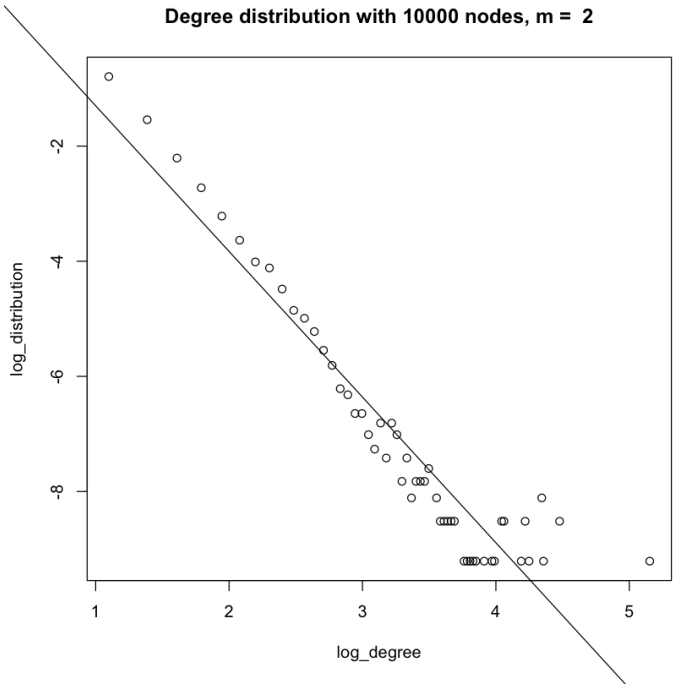


Figure 25. Undirected network with n = 10000 & m = 5 and community structure

Moreover, we also repeat the plot of degree distribution and the slope estimation using linear regression as shown:





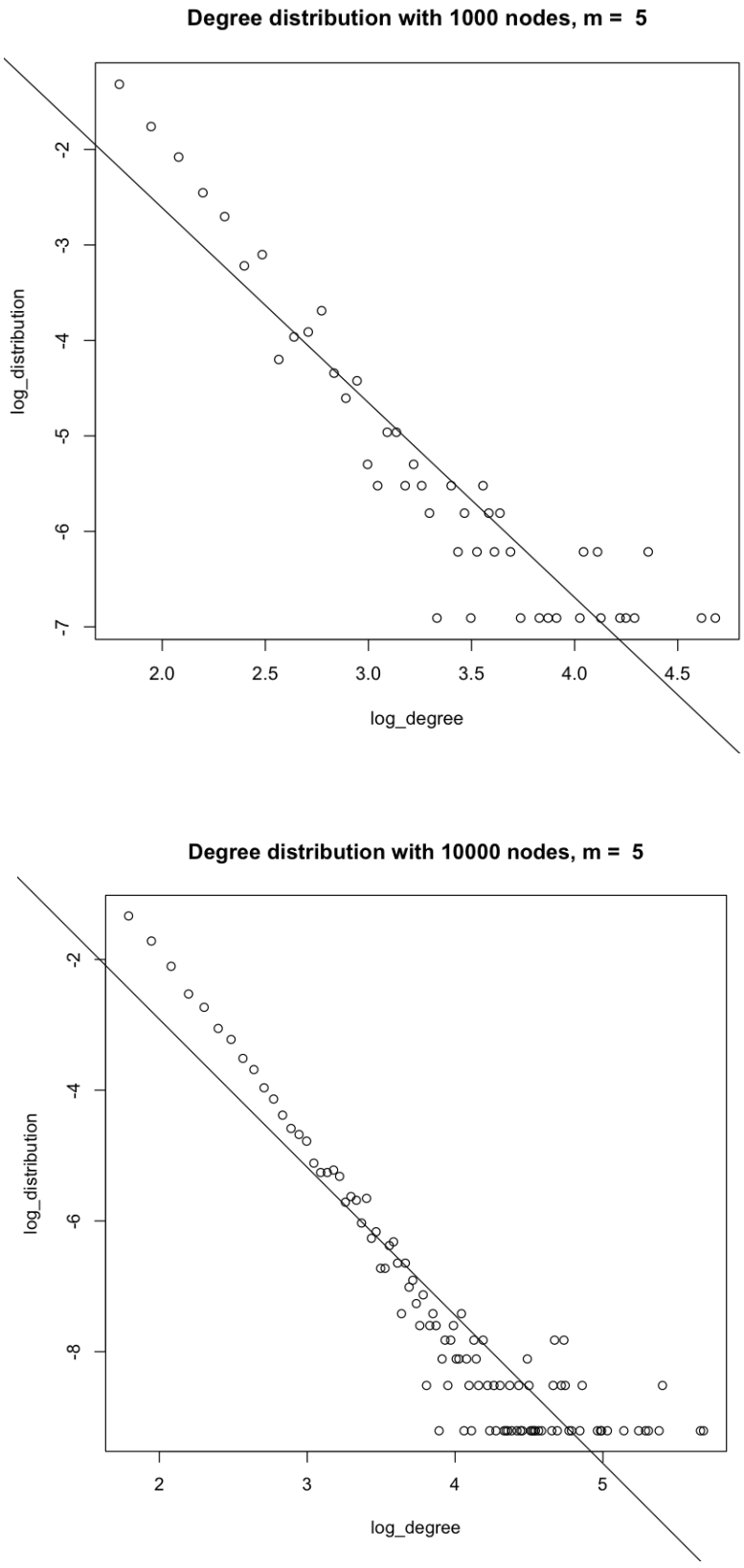


Figure 26. Degree distribution for networks with different m and n

For the modularity in different n and m:

Table 5. Modularity results for networks with different n and m

n	1000			10000		
m	1	2	5	1	2	5
Number of communities	31	21	9	107	37	17
Modularity	0.9339965	0.5234012	0.2793098	0.9783339	0.5302504	0.2717065
Slope	-2.7544494	-2.2722123	-2.0401963	-2.9357544	-2.5305569	-2.2644604

For undirected network with preferential attachment model, if each new node attaches to more old nodes, the new node will be along with more edges and connect with those nodes with higher degree, which will reduce dense connections between the nodes within communities but strengthen connections between nodes in different communities. Consequently, the modularity will decrease. Therefore, modularity for m = 1 is high.

(h) The plots with communities of the preferential attachment network and its degree sequence network are shown as:

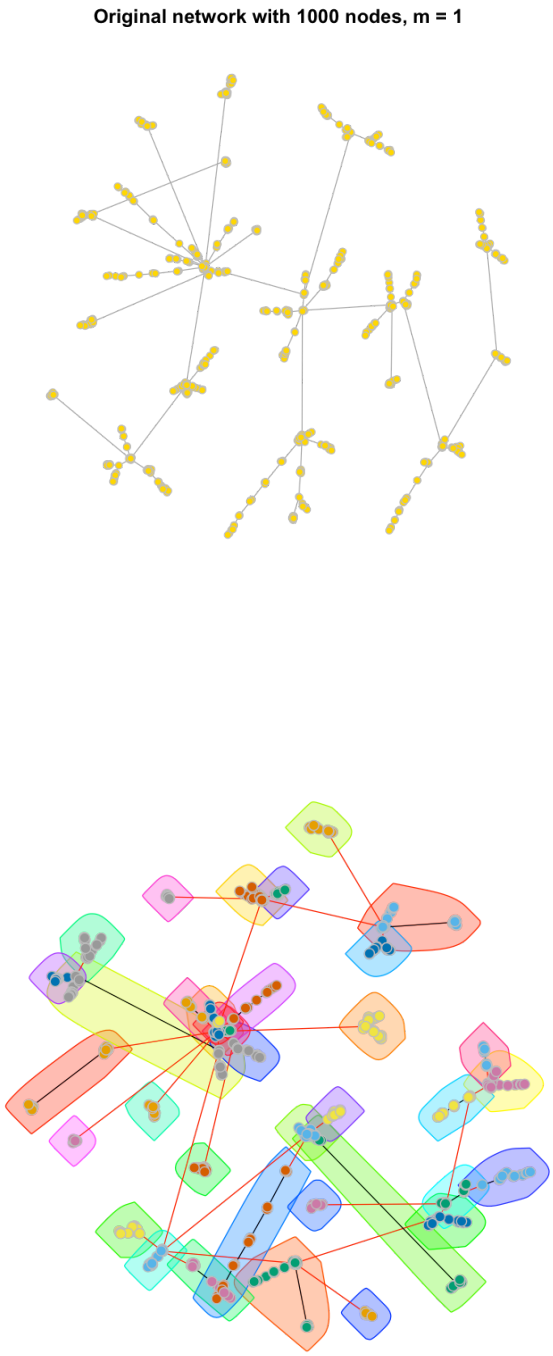


Figure 27. Preferential attachment network and its community structure

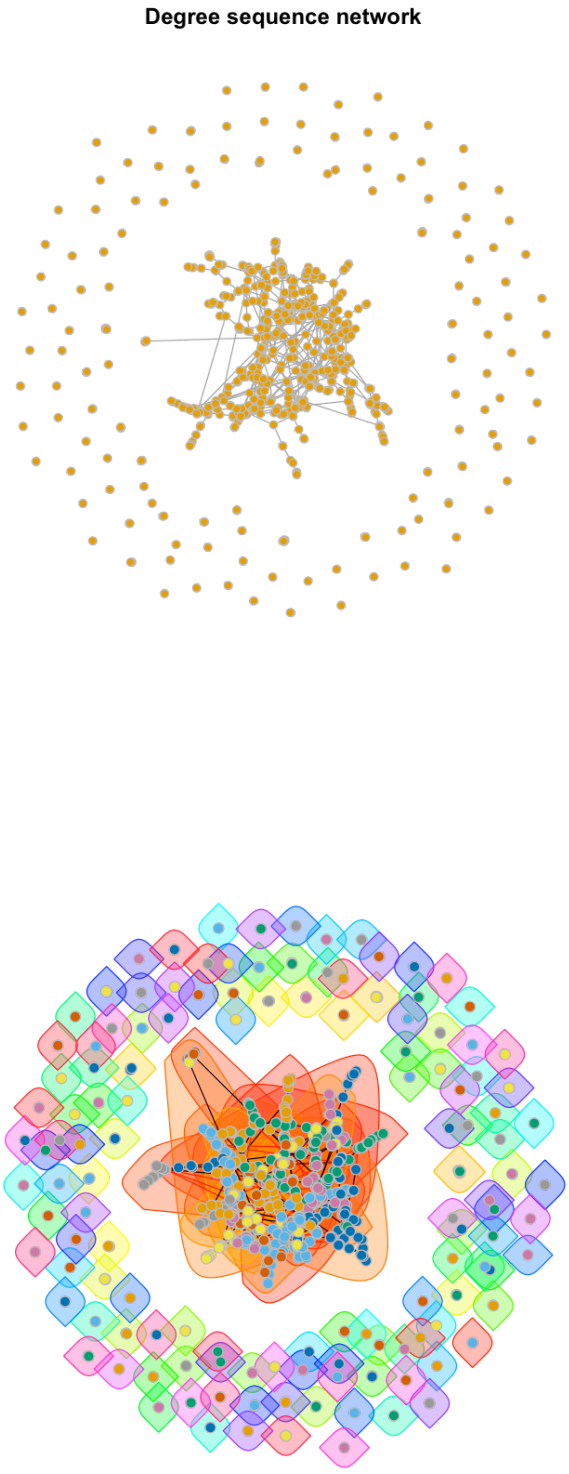


Figure 28. Degree sequence network and its community structure

Comparing two networks:

Table 6. Community structure for two networks

	Connected	Number of communities	Modularity
Preferential attachment network	Yes	31	0.93213032852673
Degree sequence network by stub-matching	No	142	0.83809986162339

For creating random power-law networks, preferential attachment network is generated by adding discrete nodes in a random fashion to a set of objects, and adding additional nodes distributed among the objects continuously to the system. Thus, such a network is always connected.

For the network with the same degree sequence and stub-matching procedure, the original network is first broken apart into a bunch of “stubs”, which are basically nodes with dangling edges; then we generate a random network by randomly pairing up these stubs and connecting them. Hence, not all nodes are connected and there are more communities generated, which reduces dense connections between the nodes within communities but strengthens connections between nodes in different communities. As a result, modularity gets lower.

3. Create a modified preferential attachment model that penalizes the age of a node

(a) We use *sample_pa_age* function to create a modified preferential attachment model that penalizes the age of a node. According to the requirements, we plot the modified preferential attachment model and its degree distribution:

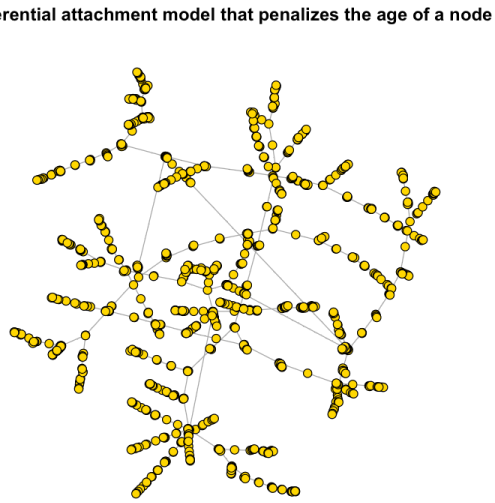


Figure 29. Preferential attachment model that penalizes the age of a node

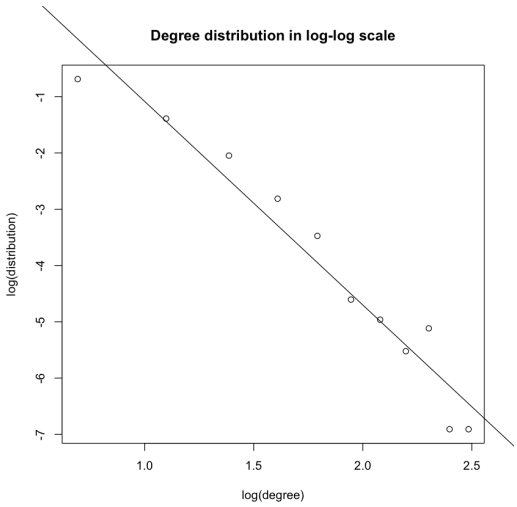


Figure 30. Degree distribution of the modified preferential attachment model

Using linear regression like what we did in 2_(d), we get the slope is -3.61540004492111. Therefore, the power law exponent is -3.61540004492111.

(b) We use fast greedy method to find the community structure and the modularity as shown:

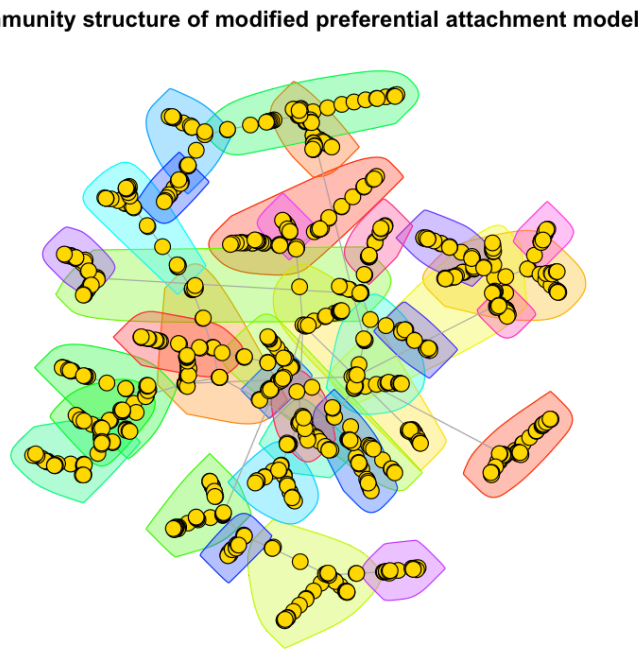


Figure 31. Community structure of preferential attachment model

Table 7. Community structure and modularity

Number of Community	34
Modularity	0.933873813753697

Part II

1. Random walk on Erdo ¨s-R´enyi networks

(a) We create the random network with the function random.graph.game from library(‘igraph’) to create the corresponding network.

(b) We plot the ⟨s(t)⟩ v.s. t and σ^2 v.s. t. Here, we chose 100 steps for this 1000 nodes network. And for each t, we set the looping time to be 1000. From the two figures below, we can find that the mean is about 3.2 and the standard deviation is 0.42 after the value becomes stable.

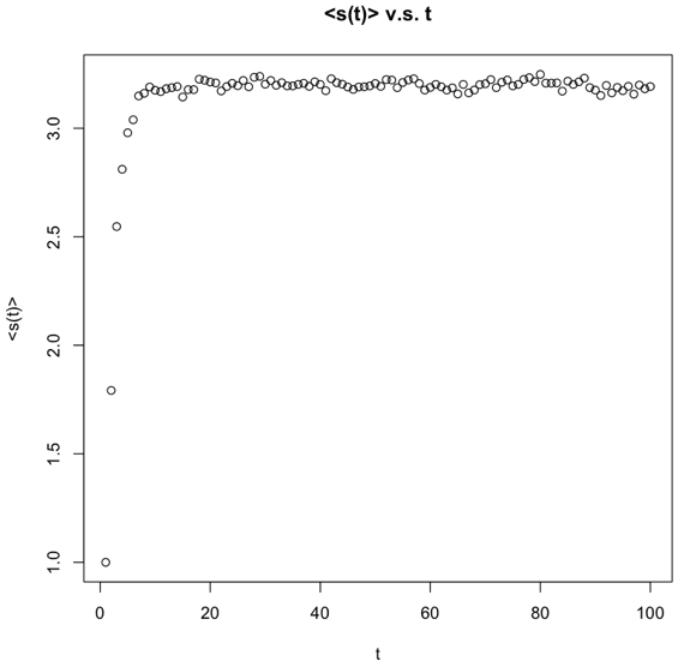


Figure 32. Average distance change with t steps

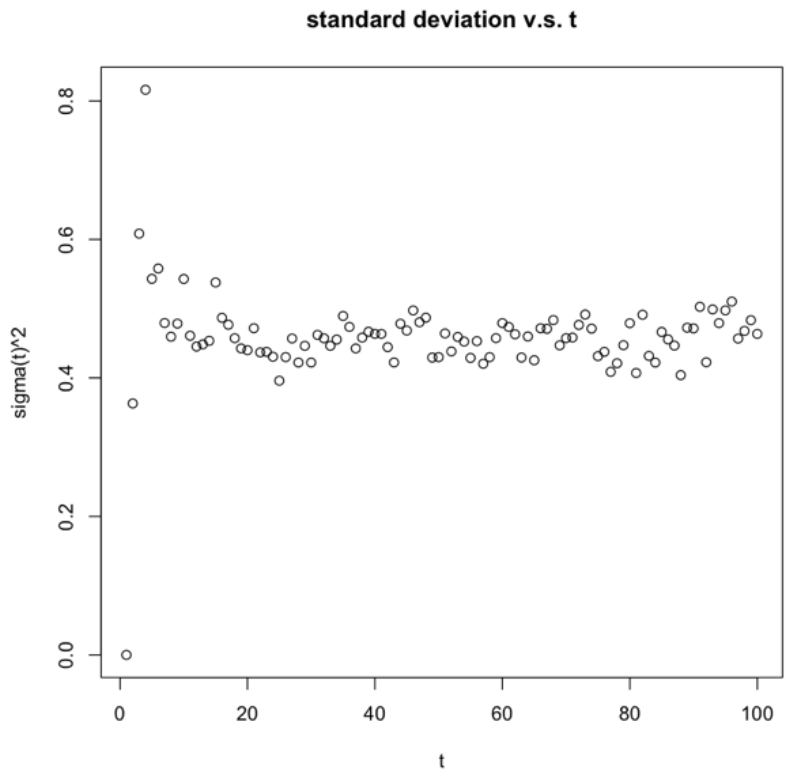


Figure 33. Standard deviation with t steps

(c) We measure the degree distribution of the nodes reached at the end of the random walk shown in the figure below.

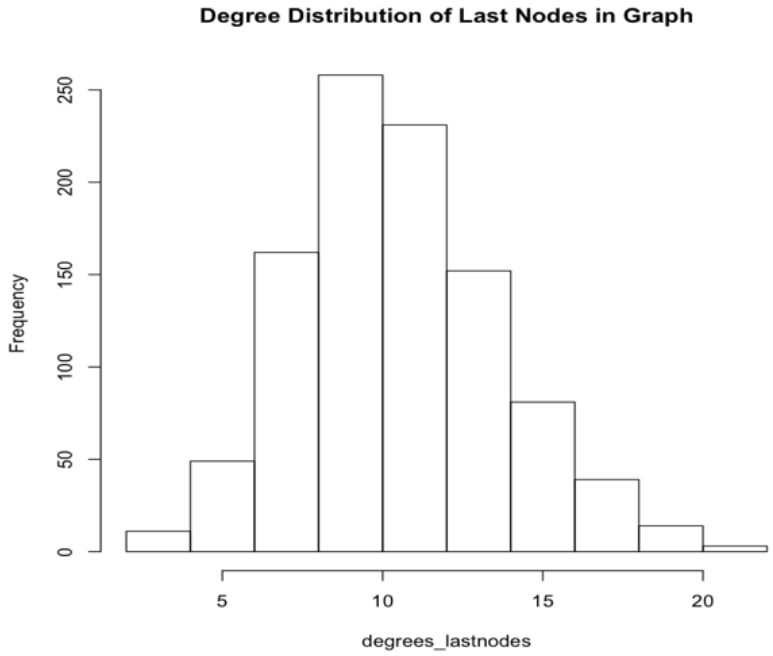


Figure 34. Degree distribution of Last Nodes

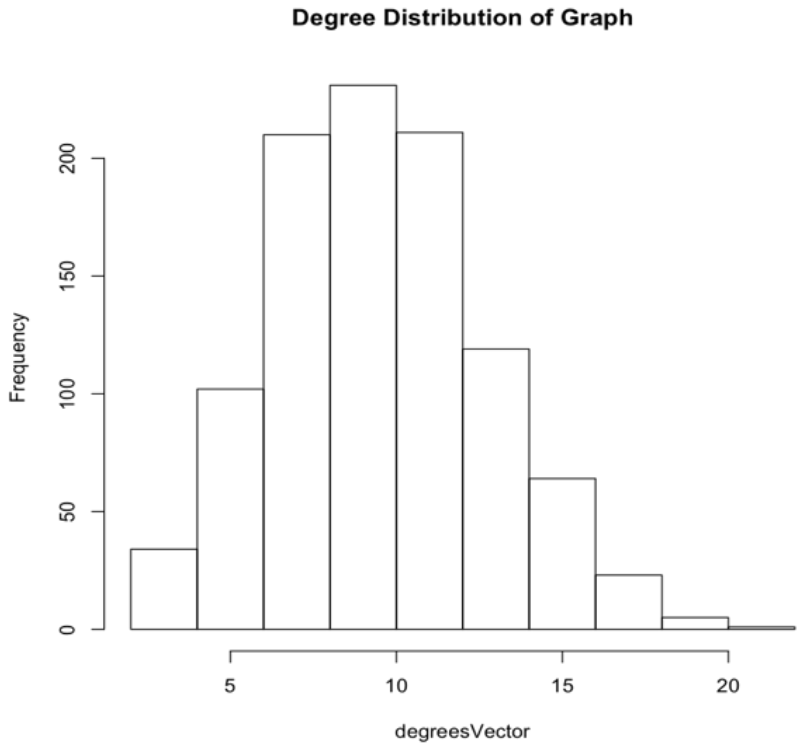


Figure 35. Degree distribution of Graph

From these two histograms, we can conclude that the degree distribution of the nodes reached at the end of the random walk shows consistency with and dependency on the degree distribution of original graph.

(d) We repeat (b) with 10000 nodes. The results are shown in figure 36 and figure 37. The mean value is 2.4 and the standard deviation is 0.25. We can see that with more nodes, smaller mean value and standard deviation are obtained. Also, it becomes stable quicker with larger value of n.

When n=10000, it becomes stable when t is less than 10. However, when n=1000, it becomes stable until t is 20. The diameter plays a role here. Intuitively, a smaller graph has smaller diameter because there are less walks, while a larger network has more information about its degree distribution. Hence, a larger graph reaches its destination sooner.

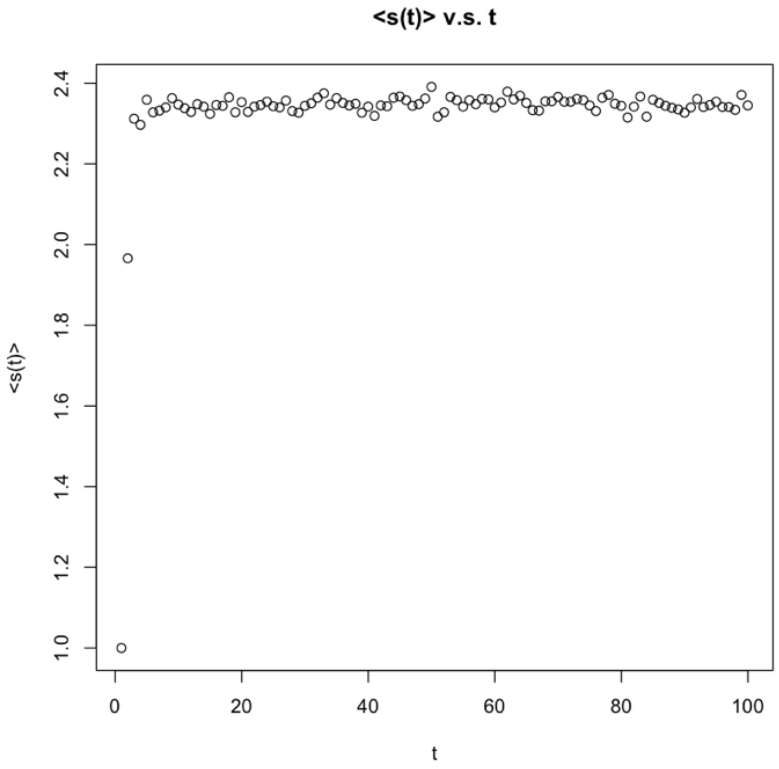


Figure 36. Average distance change with t steps with 10000 nodes

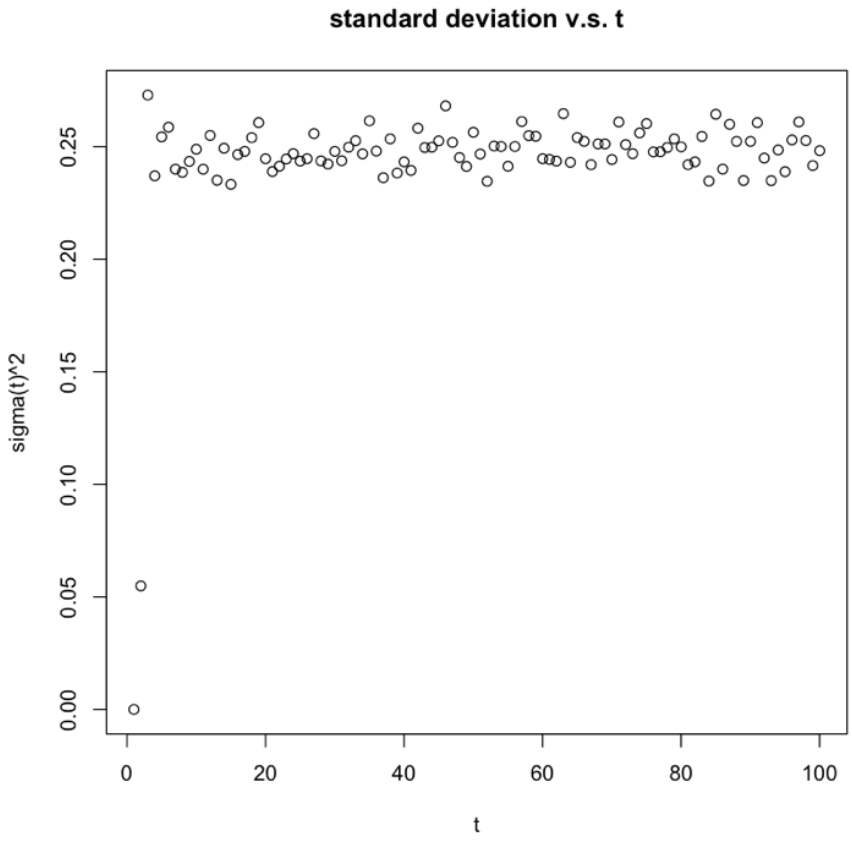


Figure 37. standard deviation with t steps with 10000 nodes

2. Random walk on networks with fat-tailed degree distribution

(a) Generate an undirected preferential attachment network with 1000 nodes, where each new node attaches to m = 1 old nodes. We implement it with barabasi.game(1000, directed=F, m=1).

(b) We plot the ⟨s(t)⟩ v.s. t and σ^2 v.s. t. Here, we chose 500 steps for this 1000 nodes network. And for each t, we set the looping time to be 1000. The results are shown in two figures below. The result is different from what we have got from Erdös-Rényi Network where the degree distribution follows the normal distribution. In preferential attachment model, the indegree is determined by power law distribution, and outdegree equals to m.

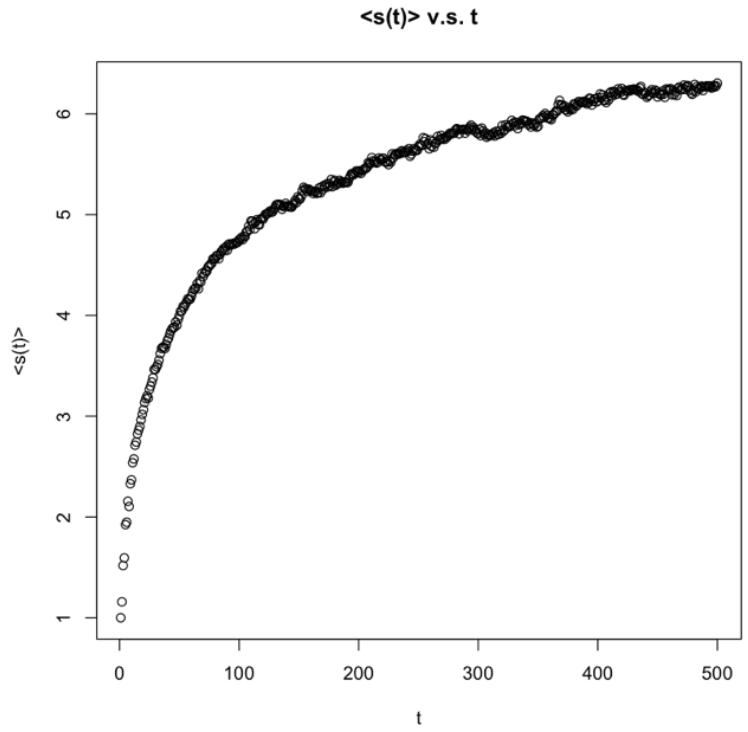


Figure 38. Average distance change with t steps with 1000 nodes

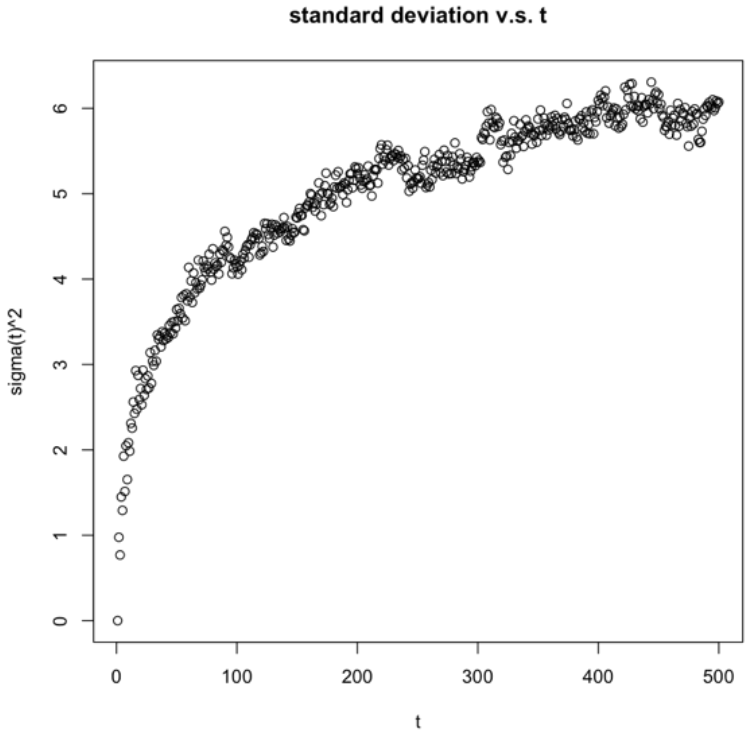


Figure 39. standard deviation with t steps with 1000 nodes

(c) We measure the degree distribution of the nodes reached at the end of the random walk shown in the figure 40 and figure 41. We can find that both distributions follow the power law distribution. The degree distribution of the nodes reached at the end of the random walk shows consistency with and dependency on the degree distribution of original graph.

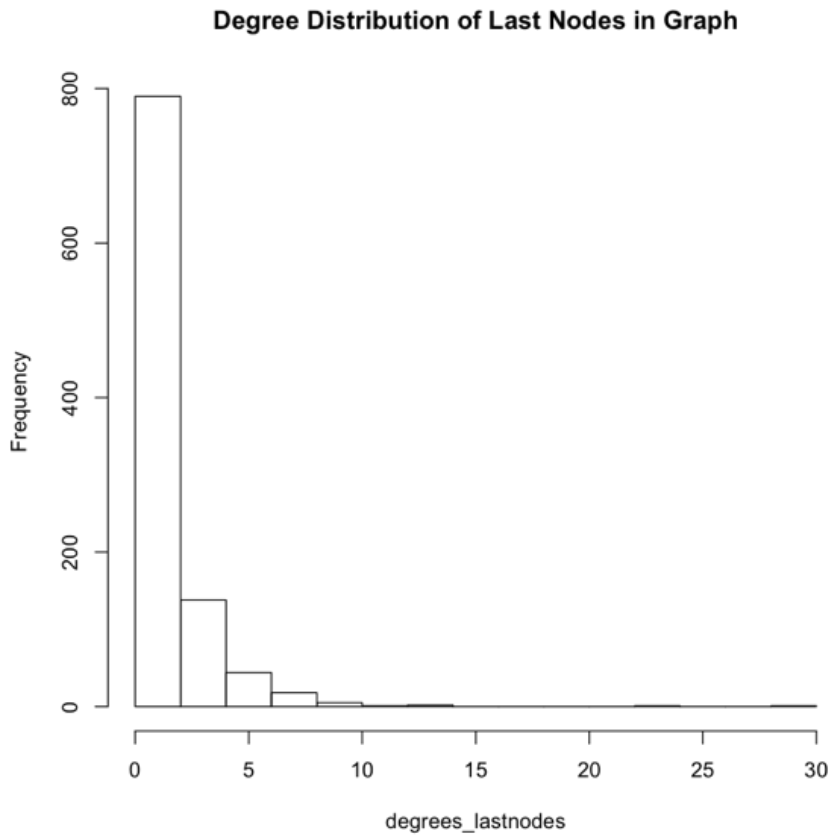


Figure 40. Degree distribution of Last Nodes

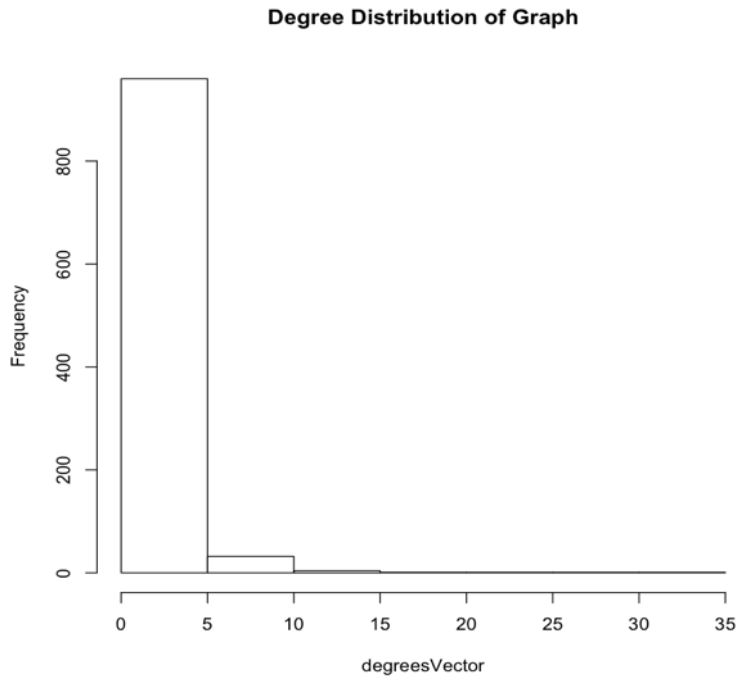


Figure 41. Degree distribution of Graph

(d) We repeat (b) with nodes=100 and nodes=10000 respectively. The results are shown in figure 42, 43, 44, 45.

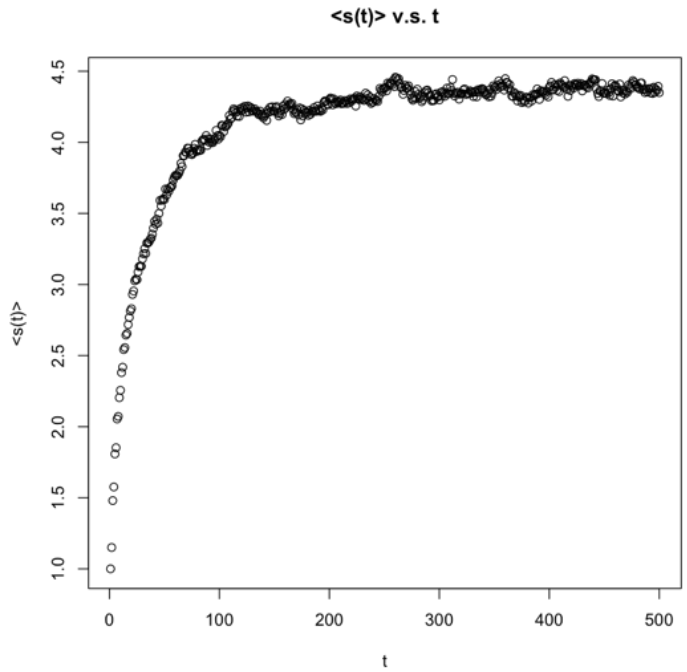


Figure 42. Average distance change with t steps with 100 nodes

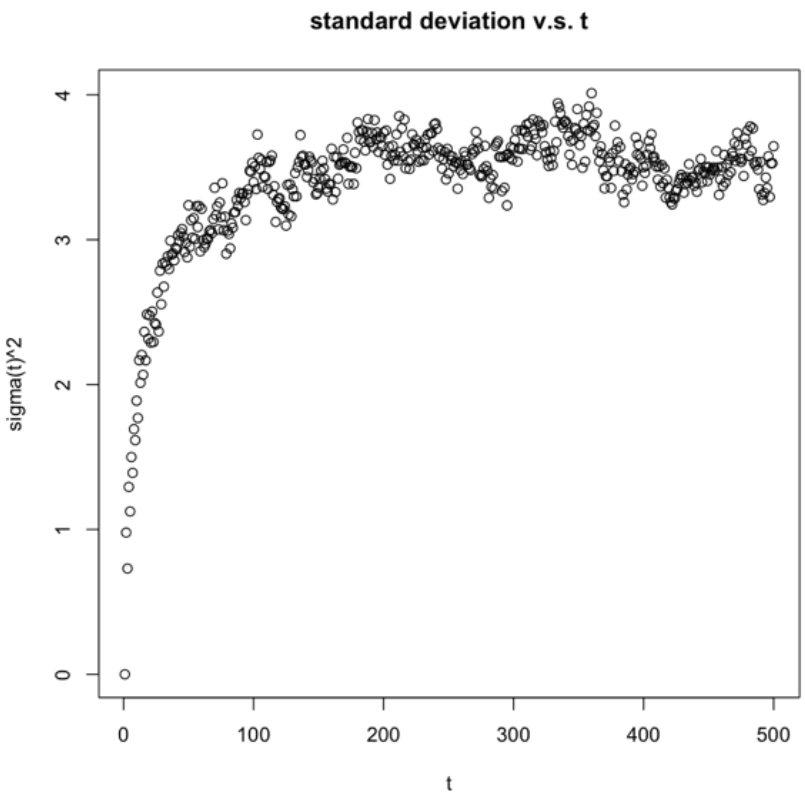


Figure 43. standard deviation with t steps with 100 nodes

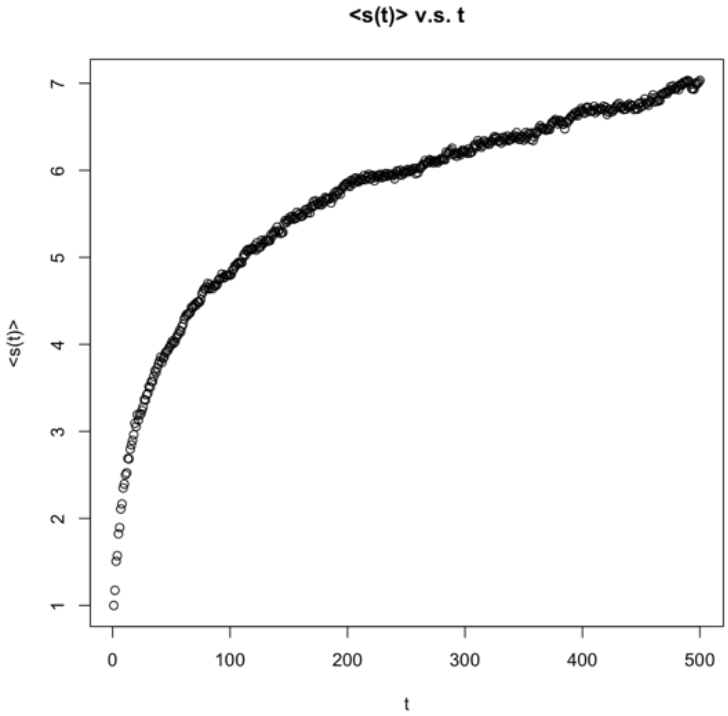


Figure 44. Average distance change with t steps with 10000 nodes

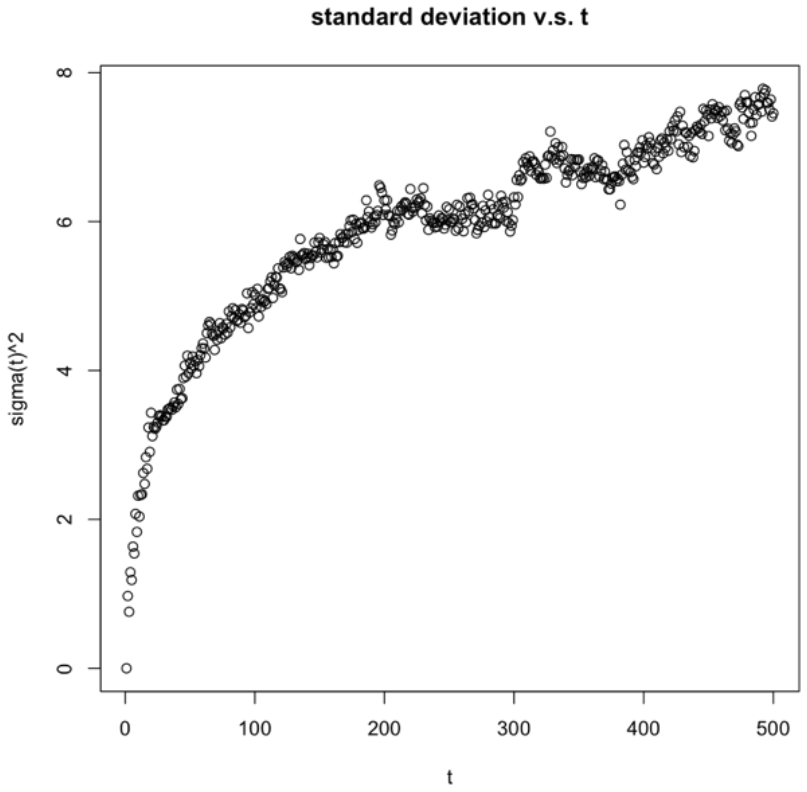


Figure 45. standard deviation with t steps with 10000 nodes

From the above four figures and what we have got from (b) with 1000 nodes, we can conclude that a graph with smaller nodes will take less time to reach the stableness of mean and standard deviation. The graph with 100 nodes reaches steady state after 150 steps, while the graph with 1000 nodes takes about 400 steps and the graph with 10000 nodes need more than 500 steps.

Also, smaller graph will have relatively smaller mean and standard deviation. The mean and standard deviation of the graph with 100 nodes is 4.2 and 3.5 respectively. For the graph with 1000 nodes, the values are 6.2 and 5.8 for mean and standard deviation. For the graph with 10000 nodes, the values are both more than 7.

Therefore, the conclusion of preferential attachment networks is contrary to ER model due to different degree distribution. Preferential attachment network follows power law distribution. Then larger graph, which also has larger diameter, will have less information pertaining to its degree distribution. Then, more steps need to be taken for random walker to reach stableness, and larger standard deviation and mean for larger graphs.

3. Rage Rank

The PageRank algorithm, as used by the Google search engine, exploits the linkage structure of the web to compute global “importance” scores that can be used to influence the ranking of search results. Here, we use random walk to simulate PageRank.

(a) We are going to create a directed random network with 1000 nodes, using the preferential attachment model. Note that in a directed preferential attachment network, the out-degree of every node is m, while the in-degrees follow a power law distribution. One problem of performing random walk in such a network is that, the very first node will have no outbounding edges, and be a “black hole” which a random walker can never “escape” from. To address that, we generate another 1000-node random network with preferential attachment model, and merge the two networks by adding the edges of the second graph to the first graph with a shuffling of the indices of the nodes. For example,

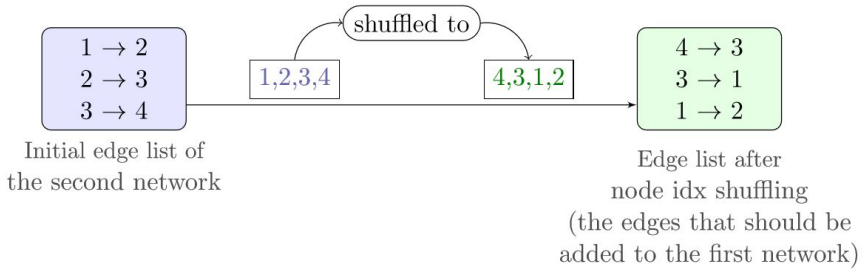


Figure 46. Example of Shuffling indices for the second network

And we firstly used the function ‘aging.prefatt.game’ to create two directed random network g1 and g2 with m=4. We kept g1 untouched and shuffled g2 with function ‘sample’ and ‘permute’. And then we combined the shuffled g2 to g1 with function ‘as_edgelist’ and ‘add_edges’. The created network is shown as the following:

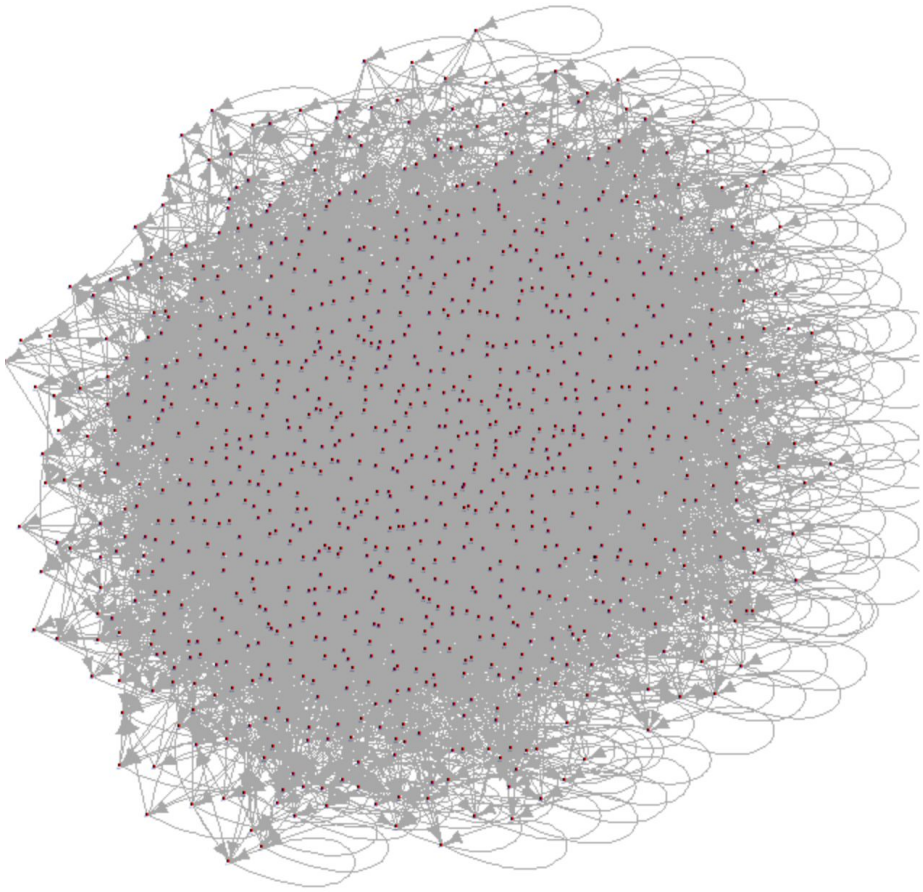


Figure 47. Directed Random Network with 1000 nodes and m=4

From figure 47, we can see that the created random network perfectly address the ‘black hole’ problem, since the very first visited nodes can also ‘escaped’. The probabilities against degrees, and probabilities that the walker visits each node are shown in the following figures. The probability is highly related to the degrees. We can tell from these figure that a relatively higher degree indicates an increasing probability. And when degrees are close to zero, the probability is almost zero, which also proves the degrees affects the probabilities.

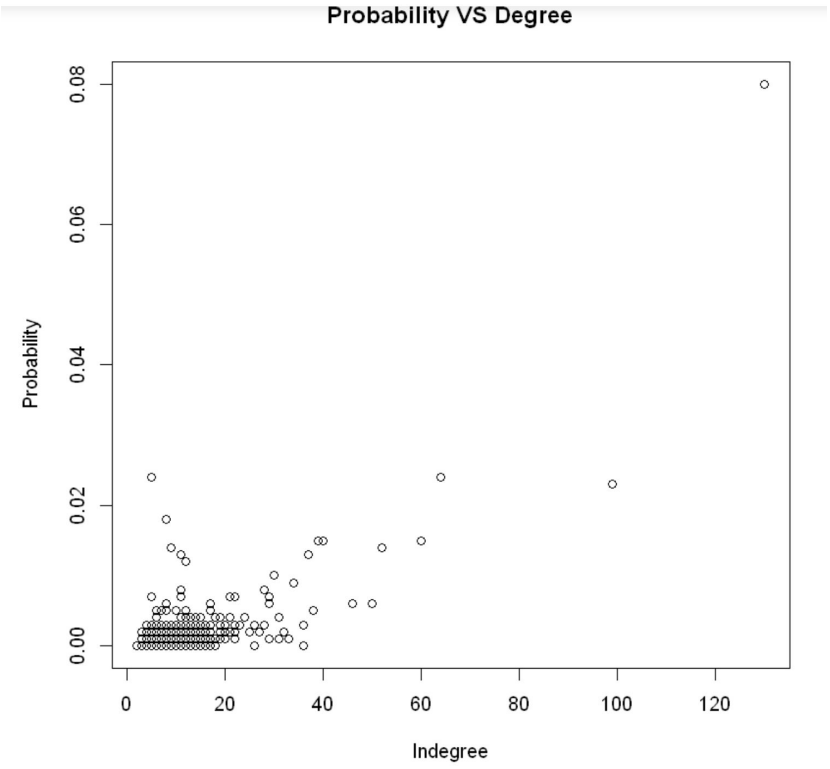


Figure 48. Probability against the Degree.

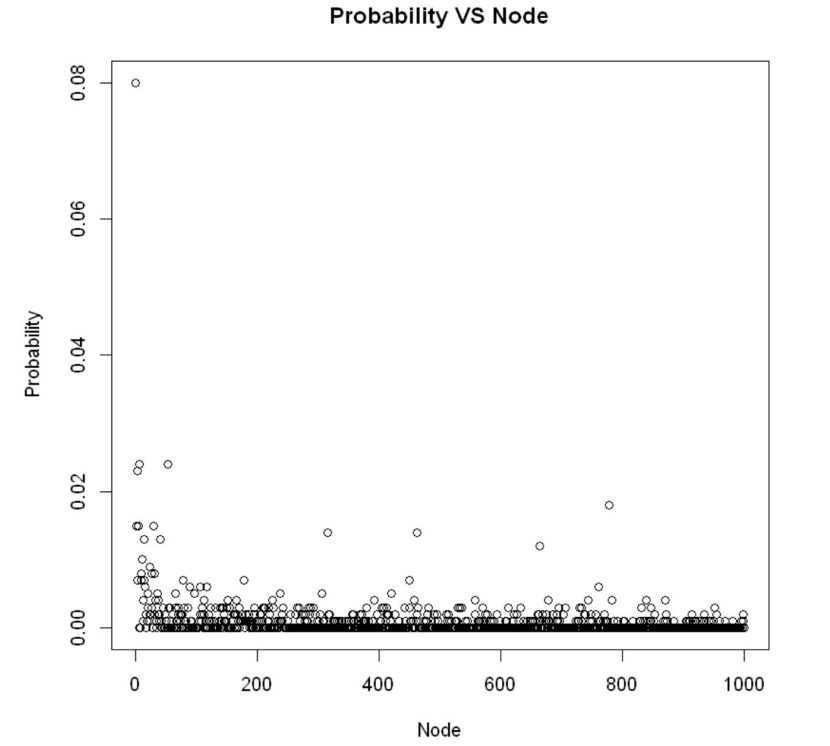


Figure 49. Probability against Nodes.

(b) In all previous questions, we didn't have any teleportation. Now, we use a teleportation probability of α = 0.15. The probabilities are shown in the following figures, and we can see from figures that relatively larger degrees correspond to higher probabilities. And the highest probability is smaller than random networks without teleportation.

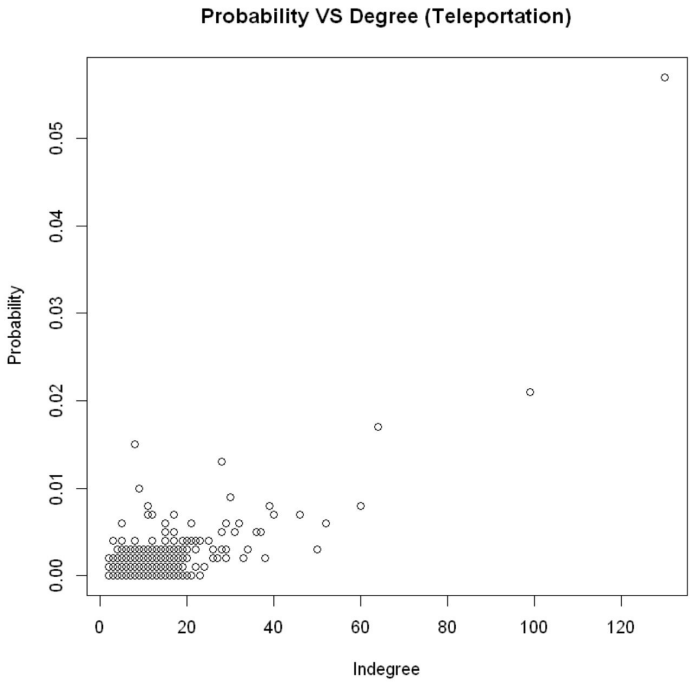


Figure 50. Probability against the Degree with Teleportation

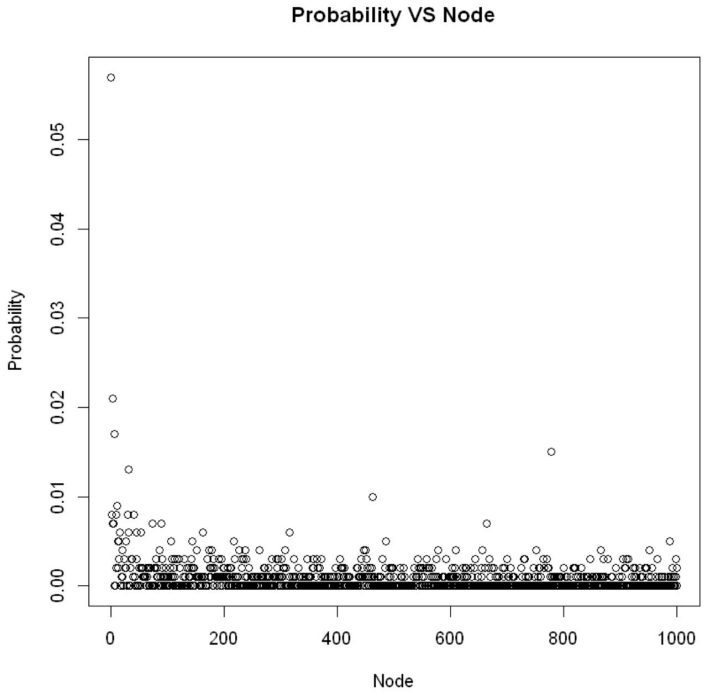


Figure 51. Probability against Nodes with Teleportation

4. Personalized PageRank

While the use of PageRank has proven very effective, the web's rapid growth in size and diversity drives an increasing demand for greater flexibility in ranking. Ideally, each user should be able to define their own notion of importance for each individual query.
(a) Here the teleportation probability to each node is proportional to its PageRank.

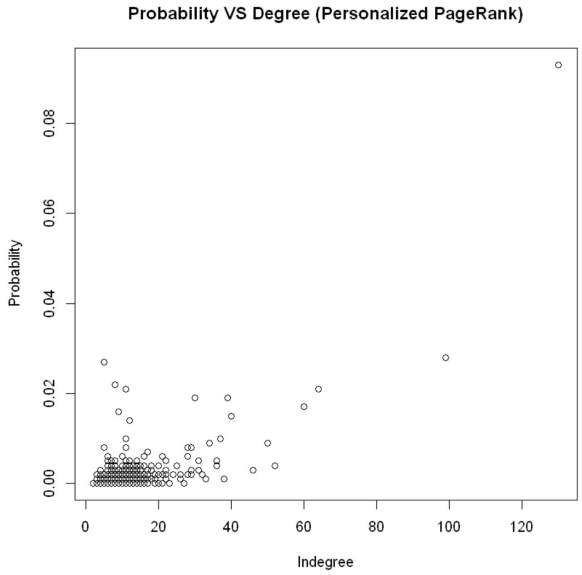


Figure 52. Probability VS Degree (Personalized PageRank)

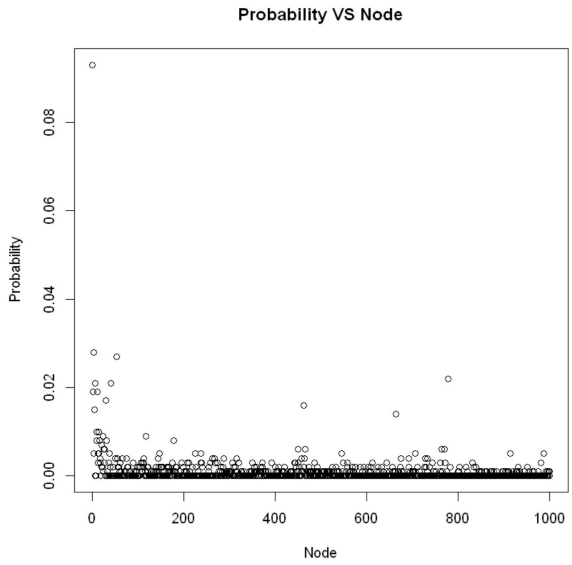


Figure 53. Probability VS Nodes(Personalized PageRank)

From figure 52 and figure 53, we can see that the difference between 3.a) and 4.a) is small. The relationship between probability against degree is similar and the distribution of probability against node is almost same.

(b) We find two nodes in the network with median PageRanks and repeat part 4(a) if teleportations land only on those two nodes (with probabilities 1/2, 1/2). The results are shown in the figure 54, and figure 55.

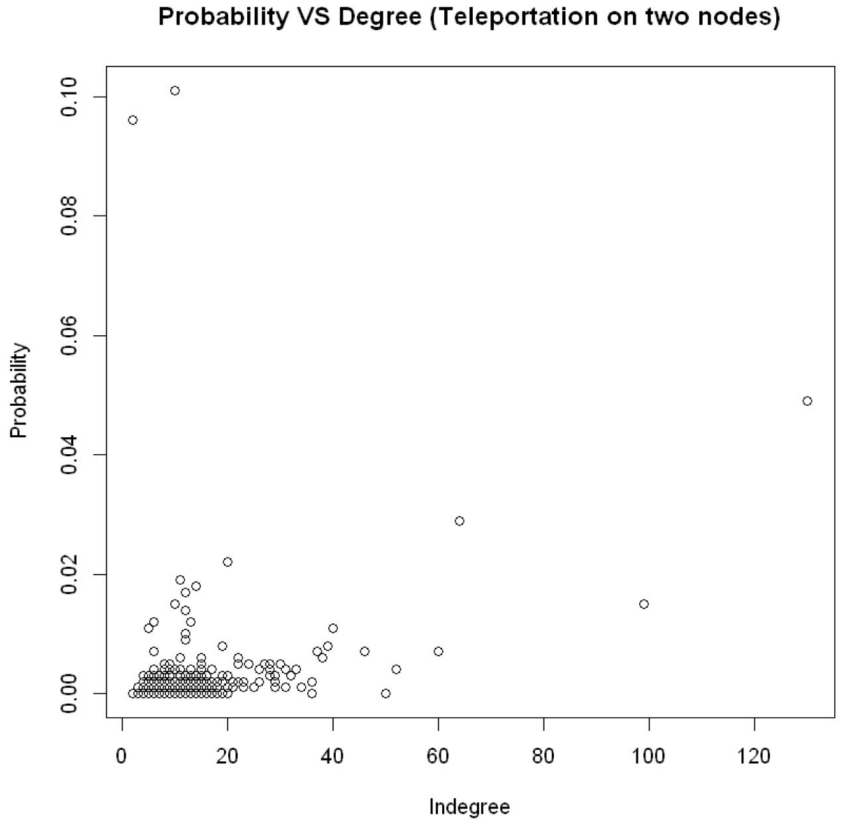


Figure 52. Probability VS Degree (Teleportation on two nodes)

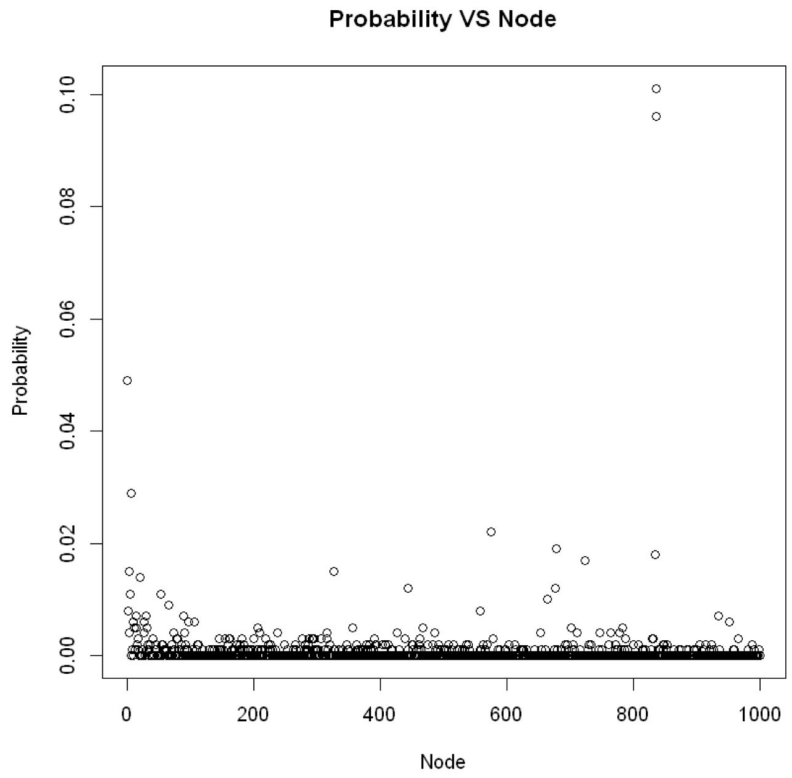


Figure 55: Probability VS Node (Teleportation on two nodes)

The above two figures show that the relationship between probability to degree is a little bit weird. We can see that when the degree is close to zero, the probability can reach the highest one. This is because that the teleportation destination is restricted to only two nodes instead that is uniformed for every node. Therefore, the nearest nodes to these two teleported nodes are more likely visited than others. Hence, from the figure 54, we find that there are around 9 nodes which have higher probabilities than other nodes.

(c) More or less, 4(b) is what happens in the real world, in that a user browsing the web only teleports to a set of trusted web pages. However, this is against the assumption of normal PageRank, where we assume that people's interest in all nodes are the same. Since we need to consider the teleportation is constricted to only trusted nodes, we can't keep 1/N anymore, and it should be changed to original equation score.
The original equation is:

$$P_{R'(A)} = (1-d)/N + d\sum_{incommingnodes} \frac{P_{R(node)}}{L(node)}$$

The modifies equation is:

$$P_{R(A)} = (1-d)P_{R'(A)} + d\sum_{incommingnodes} \frac{P_{R(node)}}{L(node)}$$