ECE 232E
Large Scale Data Mining: Models and Algorithms

Project 1
Random Graphs and Random Walks

Haitao Wang (UID: 504294402)
Xiao Peng(UID:005033608)
Zhao Weng(UID:304946606)

1.1.a Create an undirected random networks with n = 1000 nodes, and the probability p for drawing an edge between two arbitrary vertices 0.003, 0.004, 0.01, 0.05, and 0.1. Plot the degree distributions. What distribution is observed? Explain why. Also, report the mean and variance of the degree distributions and compare them to the theoretical values.

When p is 0.003, average is 2.904, variance is 2.79558, comparing to the theoretical average of 1000 * 0.003 = 3, variance of 3. Computed average and variance are quite similar to theoretical average and variance.

When p is 0.004, average is 4.008, variance is 3.99593, comparing to the theoretical average of 1000 * 0.004 = 4, variance of 4. Computed average and variance are quite similar to theoretical average and variance.

When p is 0.01, average is 9.912, variance is 9.30756, comparing to the theoretical average of 1000 * 0.004 = 10, variance of 10. Computed average and variance are quite similar to theoretical average and variance.

When p is 0.05, average is 50.274, variance is 48.1671, comparing to the theoretical average: 1000 * 0.004 = 50, variance = 50. Computed average and variance are quite similar to theoretical average and variance.

When p is 0.1, average is 99.772, variance is 94.3544, comparing to the theoretical average: 1000 * 0.004 = 100, variance = 100. Computed average and variance are quite similar to theoretical average and variance.

The degree distribution of different probabilities are plotted below. Those distribution graphs represents degree relative to frequency.

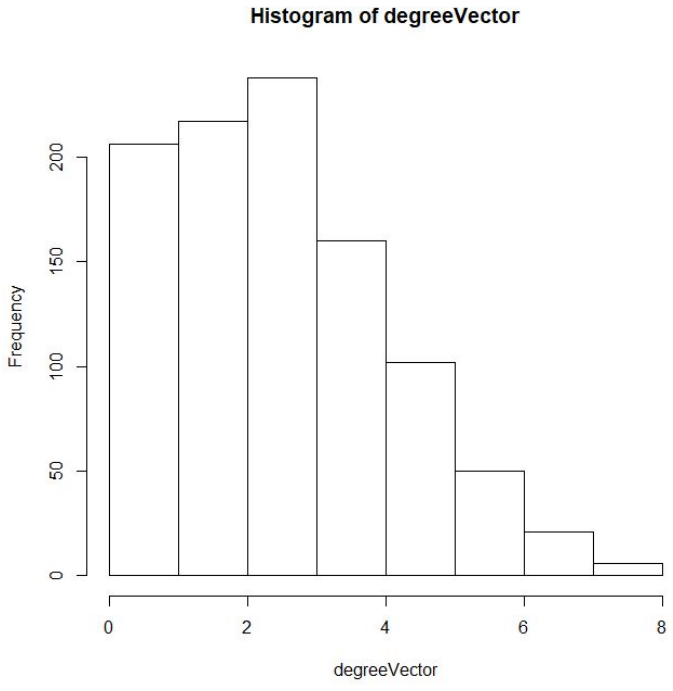


Figure 1.1.1 Histogram of degree when p = 0.003

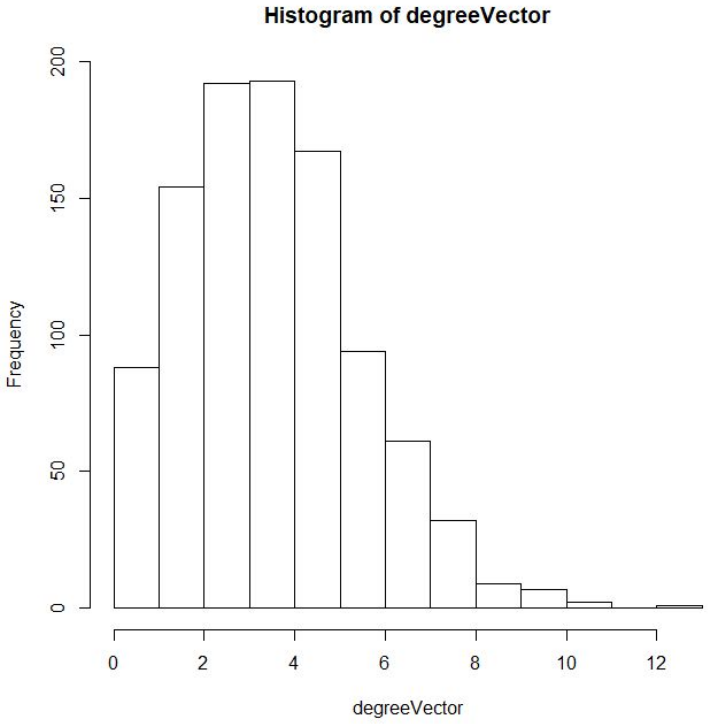


Figure 1.1.2 Histogram of degree when p = 0.004

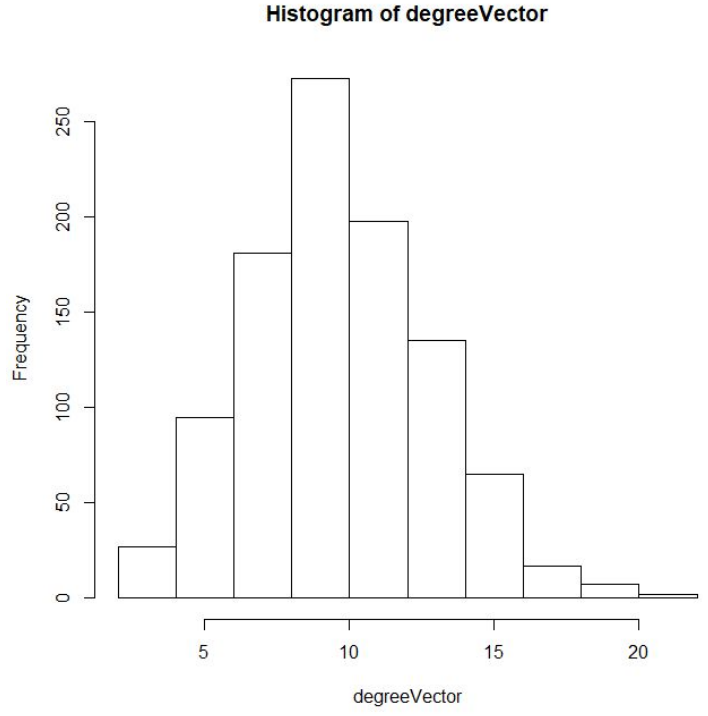


Figure 1.1.3 Histogram of degree when p = 0.01

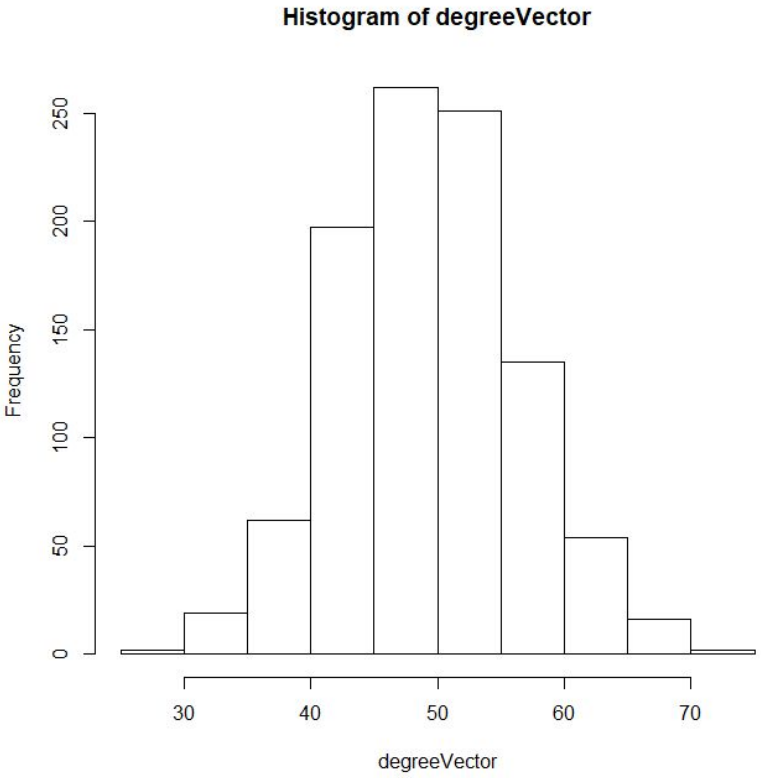


Figure 1.1.4 Histogram of degree when p = 0.05

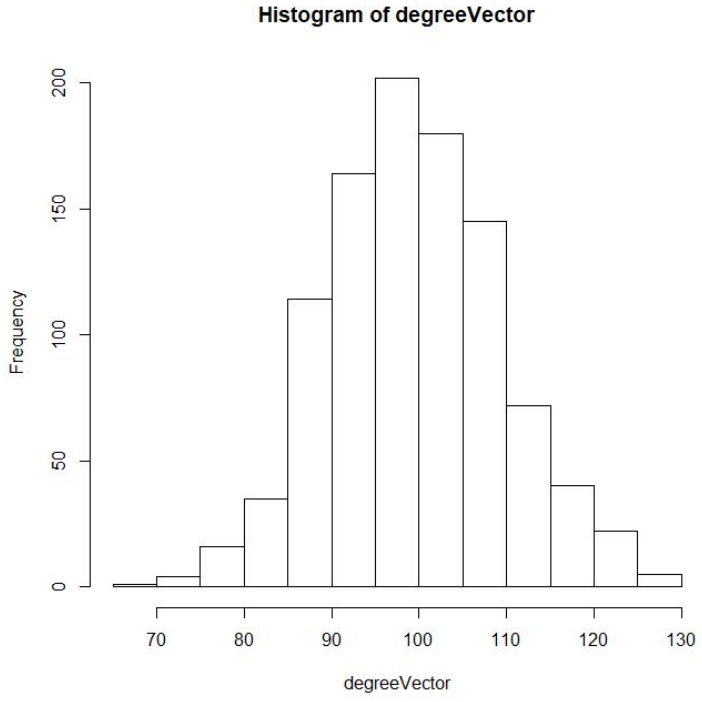


Figure 1.1.5 Histogram of degree when p = 0.1

Binomial distribution is observed as probability becomes larger.

1.1.b For each p and n = 1000, answer the following questions: Are all random realizations of the ER network connected? Numerically estimate the probability that a

generated network is connected. For one instance of the networks with that p, find the giant connected component (GCC) if not connected. What is the diameter of the GCC?

The numerically estimated probabilities of the network being connected with p equal to 5 different values are: 0.000, 0.000, 0.964, 1.000, 1.000, respectively.

When p are 0.003, 0.004, the graph is not connected. When p are 0.01, 0.05, 0.1, the graph is connected. Diameters for GCC when p = 0.003, 0.004, 0.01, 0.05, 0.1 are 14, 12, 3, 4, 5.

1.1.c It turns out that the normalized GCC size (i.e., the size of the GCC as a fraction of the total network size) is a highly nonlinear function of p, with interesting properties occurring for values. For n = 1000, sweep over values of p in this region and create 100 random networks for each p. Then scatter plot the normalized GCC sizes vs p. Empirically estimate the value of p where a giant connected component starts to emerge (define your criterion of "emergence")? Do they match with theoretical values mentioned or derived in lectures?

We define the GCC starts to emerge when normalized GCC size is larger than 0.95. In my definition, GCC starts to emerge when p = 0.01. It matches with the value derived in lecture.

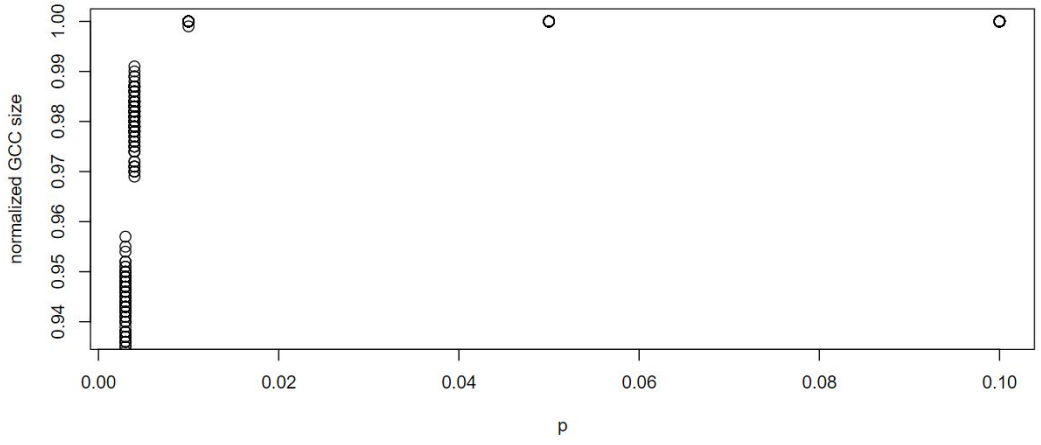


Figure 1.1.6 The relation between normalized GCC size with the probability

1.1.d.i Define the average degree of nodes c = n × p = 0.5. Sweep over number of nodes, n, ranging from 100 to 10000. 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. What trend is observed?

We produce an ER network and use "cluster" function to distinguish all the cluster. Then "max" function can help to locate the giant connected component. By sweeping the number of vertex from 100 to 10000, we produce 100 graph for each n then plot the average size of GCC for these 100 graph vs the number of vertex, as shown in Figure 1.1.7 We can observe that the major trend approaches to a constant in steady state. And since the points are disorderly, we propose a method to analyze the data points in Figure 1.1.7. We extract the median of GCC size in every interval of n and plot the new data points in Figure 1.1.8 . From this plot, it's evident that the points can be fitted into a function with decreasing gradient, such as the product of a constant and $x^{0.5}$.

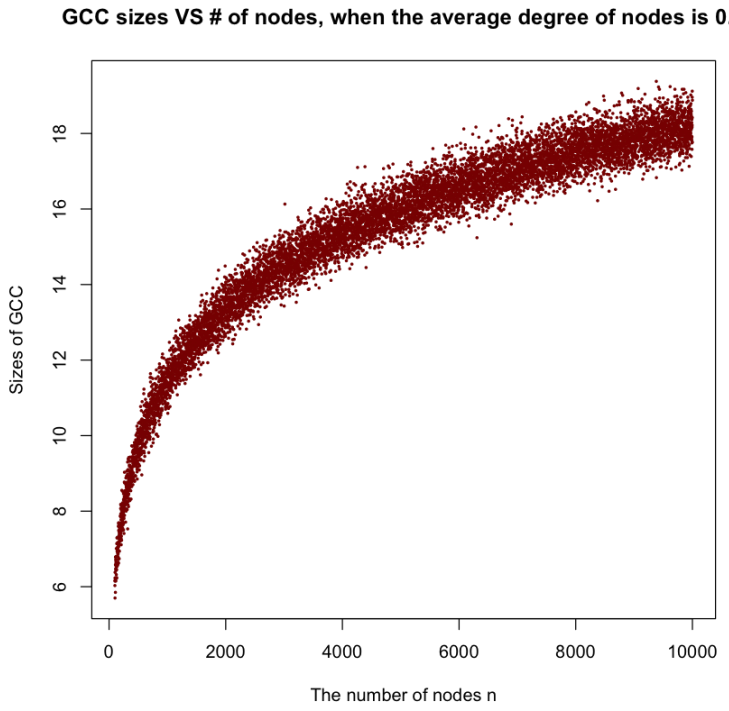


Figure 1.1.7: The plot of the size of GCC VS the number of vertex

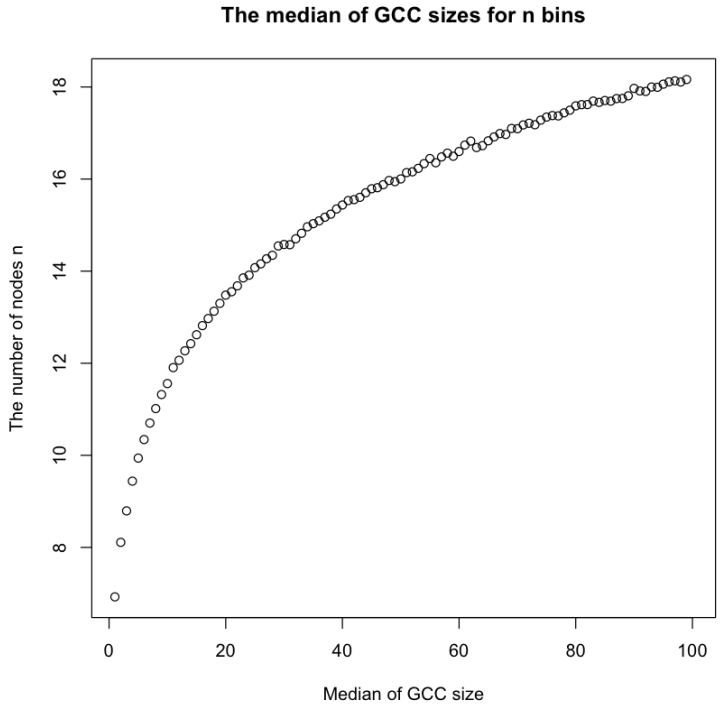


Figure 1.1.8: The plot of the analyzed trend of GCC size

1.1.d.ii Repeat the same for c = 1.

Similarly, we change the c to 1 for this part, and plot the average size of GCC for 100 graph VS the number of vertex, as well as the analyzed trend of GCC size. From the analyzed plot, we can clearly characterize the data points by a function with decreasing gradient, however, whose second derivative is smaller than that of c = 0.5. The plots are

listed as following Figure 1.1.9 and Figure 1.1.10 .

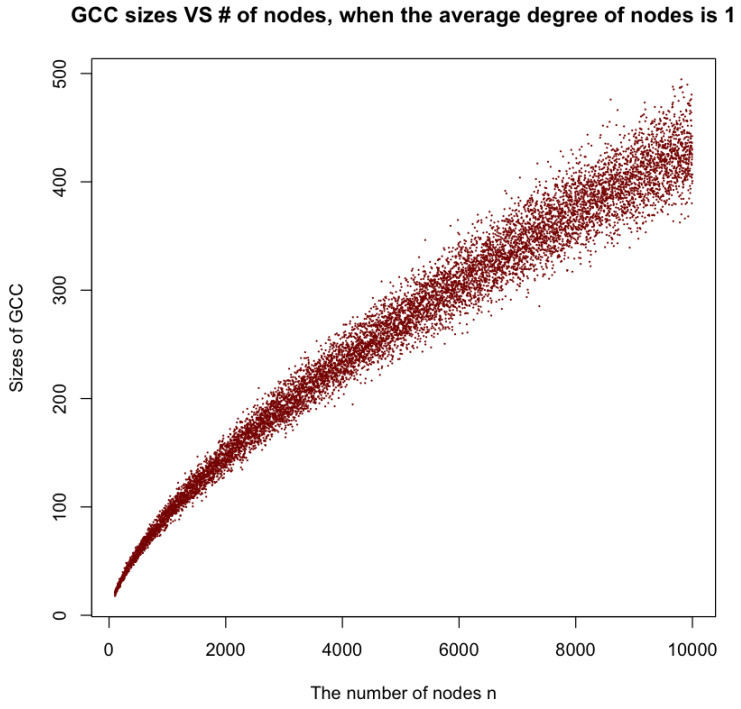


Figure 1.1.9: The plot of the size of GCC VS the number of vertex

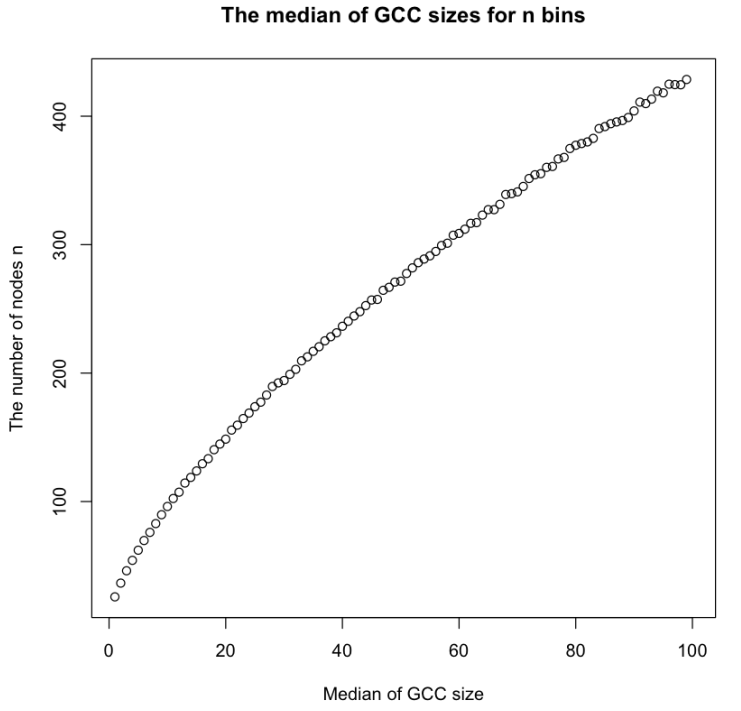


Figure 1.1.10: The plot of the analyzed trend of GCC size

1.1.d.iii 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.

Similarly, we change the c to 1.1, 1.2 and 1.3,respectively, and plot the data points in the same picture as shown in Figure 1.1.11. According to the plot, the trend is as clear as crystal that higher c brings GCC larger sizes and makes the fitting curve more linear, less variance.

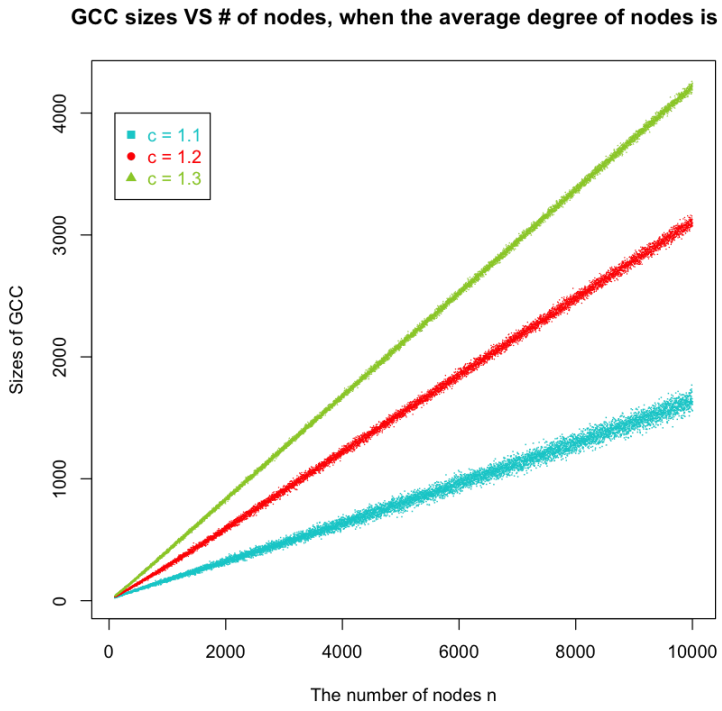


Figure 1.1.11: The plot of the size of GCC VS the number of vertex

1.2.a Create an undirected network with n = 1000 nodes, with preferential attachment model, where each new node attaches to m = 1 old nodes. Is such a network always connected?

The following code is used to create the graph with the "sample_pa" function. Such network is not always connected.

1.2.b Use fast greedy method to find the community structure. Measure modularity.

Community is shown below. Modularity of the model with 1000 nodes is : 0.90472.

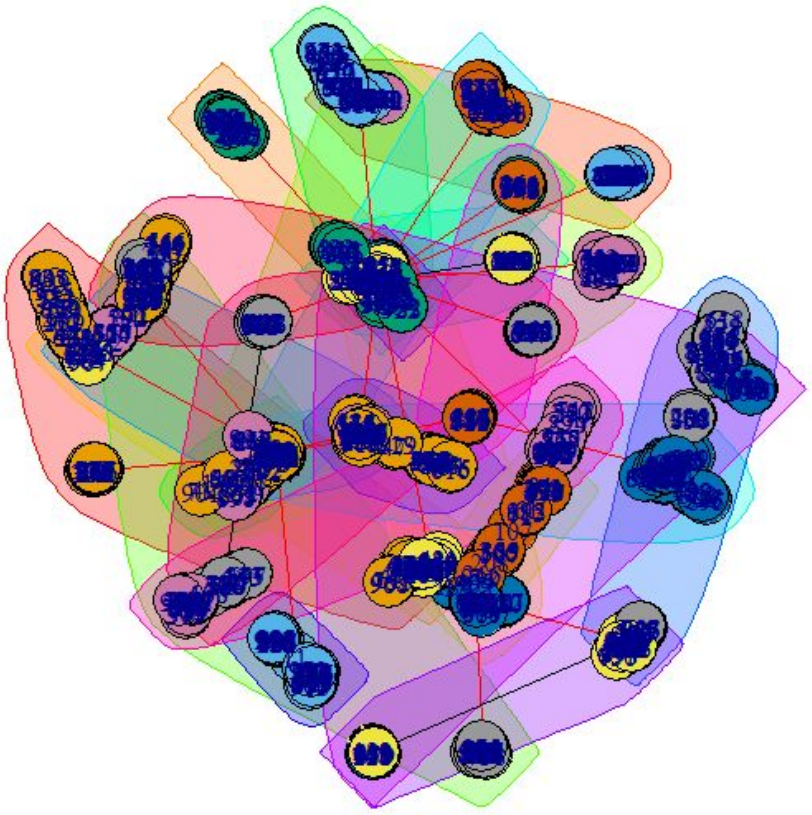


Figure 1.2.1 The community of the graph

1.2.c Try to generate a larger network with 10000 nodes using the same model. Compute modularity. How is it compared to the smaller network's modularity?

Modularity of the model with 10000 nodes is : 0.92661. Compared to smaller network's modularity, larger network has larger modularity.

1.2.d Plot the degree distribution in a log-log scale for both n = 1000, 10000, then estimate the slope of the plot.

Produce 1000 graph with "sample_pa" function for both n = 1000 and n = 10000 and plot the degree distribution with "degree" function. From the log-log scale curve as shown in Figure 1.2.2 and Figure 1.2.3 respectively, we can estimate the slope is -3.1 for n=1000 and -2.9 for n =10000.

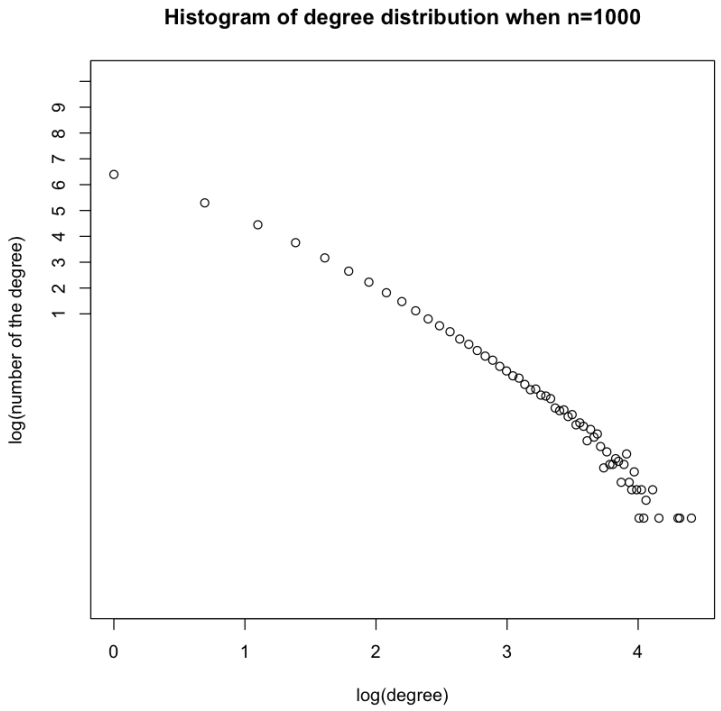


Figure 1.2.2: The plot of the degree distribution in the log-log scale

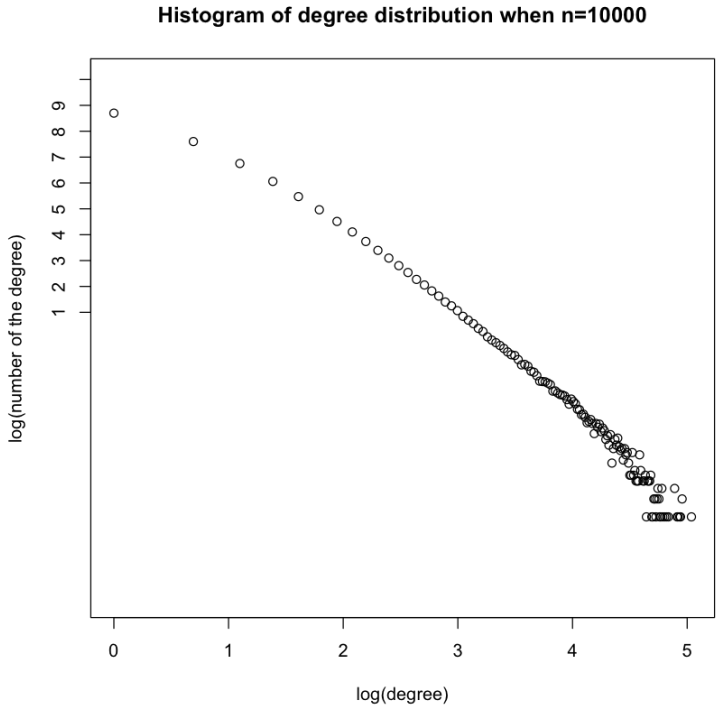


Figure 1.2.3: The plot of the degree distribution in the log-log scale

1.2.e You can randomly pick a node i, and then randomly pick a neighbor j of that node. Plot the degree distribution of nodes j that are picked with this process, in the log-log scale. How does this differ from the node degree distribution?

To begin with, we produce a graph with n = 1000. Then we randomly pick a vertex i with "sample" function, find its neighbor with "ego" function and randomly pick a vertex j

from the neighbor of vertex i. Repeat this process 1000 times so that we can plot the histogram of the degree distribution of vertex j as shown in Figure 1.2.4, and compare it with the degree distribution of the whole graph completed in last part. And we can come to the conclusion that the degree distribution of node j is less linear than that of the whole graph, while the slope of fitting curves of these two are close.

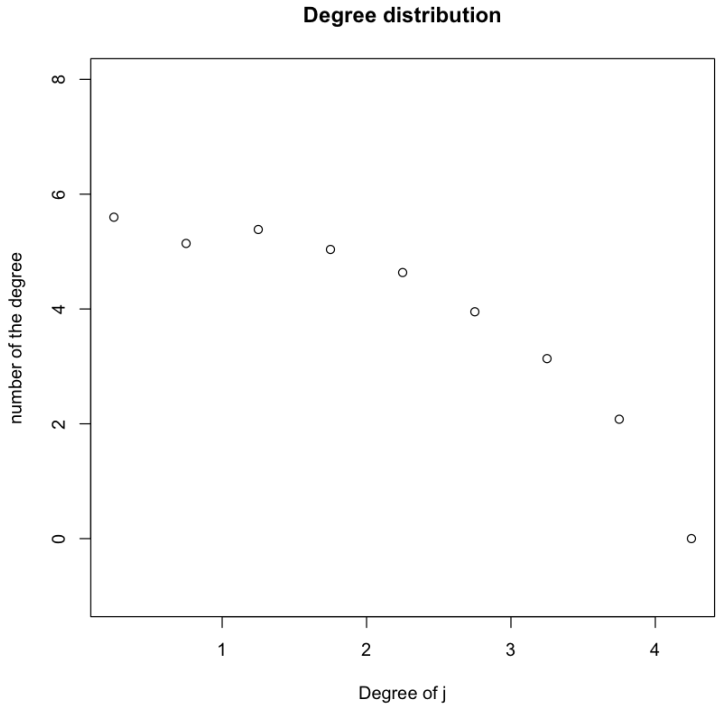


Figure 1.2.4: The plot of the degree distribution of vertex j in the log-log scale

1.2.f Estimate the expected degree of a node that is added at time step i for $1 \leq i \leq 1000$. Show the relationship between the age of nodes and their expected degree through an appropriate plot.

Since the age of a node is reflected in its index, in order to find the relation between the age and the degree of a node, we only need to plot its degree versus its index. To make it reveal the relation clearer, after plot it in a linear scale as shown in Figure 1.2.5, we repeat it in a log-log scale as shown in Figure 1.2.6. From the latter one, we can easily identify that the relation between the age and the degree is linear in the log-log scale. In another word, we can fit the age versus the degree into an exponent curve.

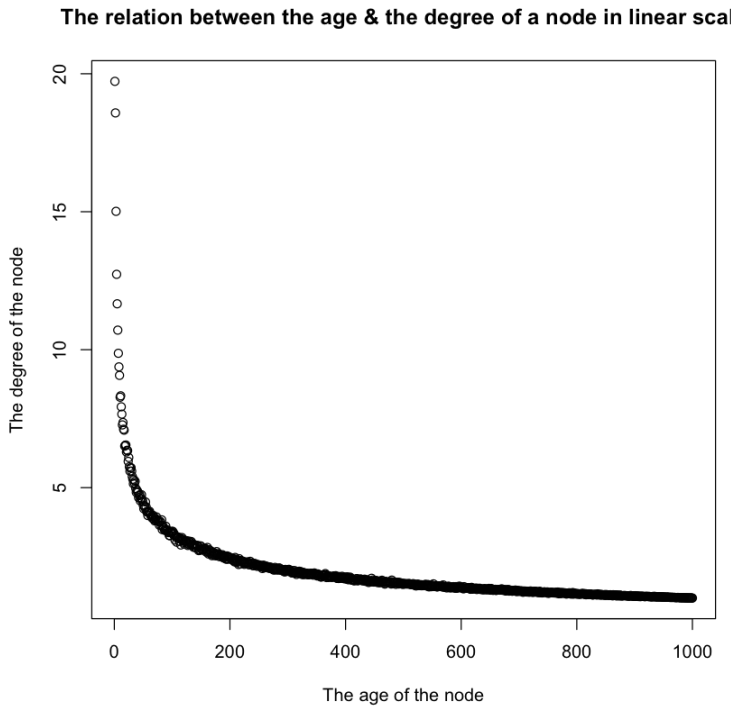


Figure 1.2.5: The plot of the degree versus the age of a node in linear scale

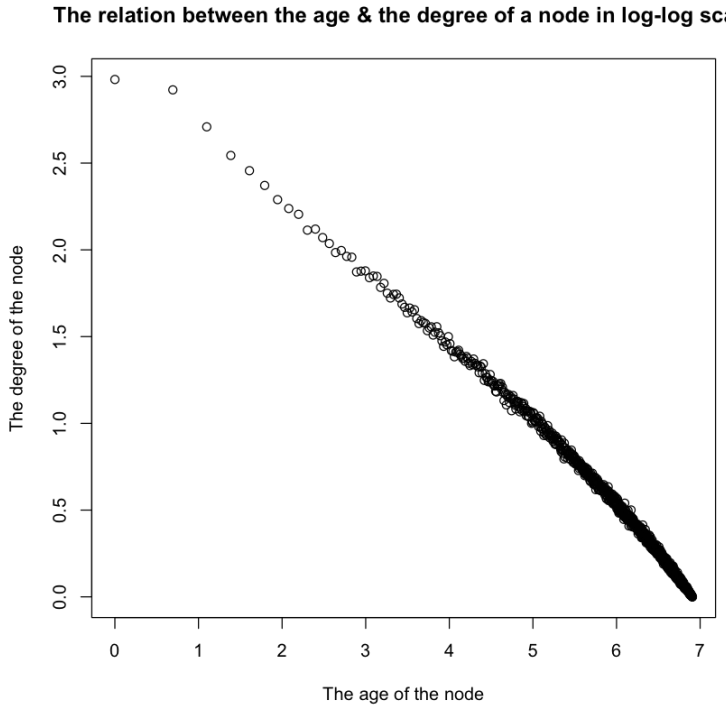


Figure 1.2.6: The plot of the degree versus the age of a node in the log-log scale

1.2.g Repeat the previous parts for m = 2, and m = 5. Why was modularity for m = 1 high?

The following code is used to create the graph with " sample_pa" function. When m = 2, it is always connected. Modularity when m = 2 is : 0.5507198.

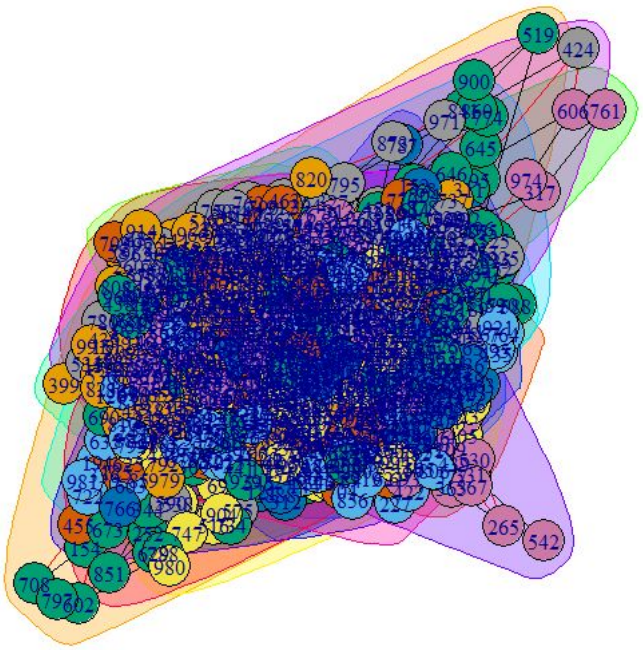


Figure 1.2.7 The community of the graph

When there are 10000 nodes, modularity of the model is : 0.564604. Larger graph has larger modularity.

The following code is used to create the graph created by preferential attachment.

When m = 5, the graph will always be connected. Modularity of the graph when m = 5 is 0.3249355.

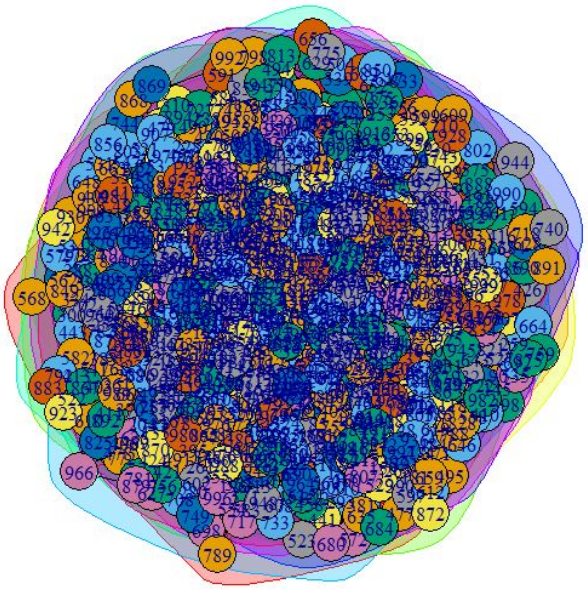


Figure 1.2.8 The community of the graph

When there are 10000 nodes, modularity of the model is : 0.3308321.Larger graph has larger modularity.

The modularity when m =1 is high because only one edge is added at each step to build the graph. So when m = 1, new added node will have connections within modules, thus resulting in the graph having dense connections between the nodes within modules but sparse connections between nodes in different modules

Similarly, the results are obtained from the same process with the only change of m. Figure 1.2.9 to Figure 1.2.13 are the results for m = 2 and Figure 1.2.14 to Figure 1.2.18 are for m = 5. For m = 2, we have the slopes of n = 1000 and n = 10000 for degree distribution are, respectively, -2.57 and -2.57. As for m = 5, the slopes of n = 1000 and n = 10000 for degree distribution become -2.58 and -2.80. The difference between degree distribution of the whole graph and that of a node j is the same as mentioned above. Also, the relation between the age and the degree for the nodes in the log-log scale keeps linear for both m = 2 and m = 5.

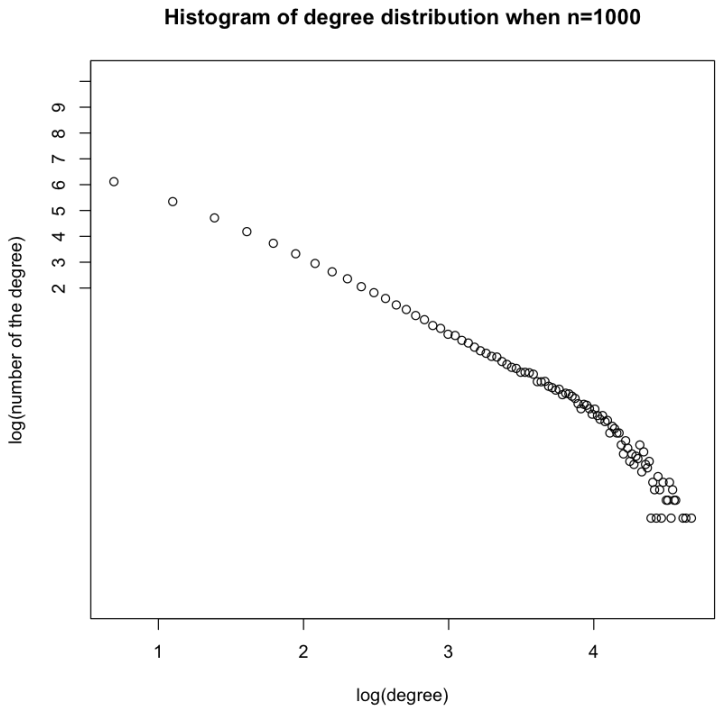


Figure 1.2. 9: The plot of the degree distribution in the log-log scale

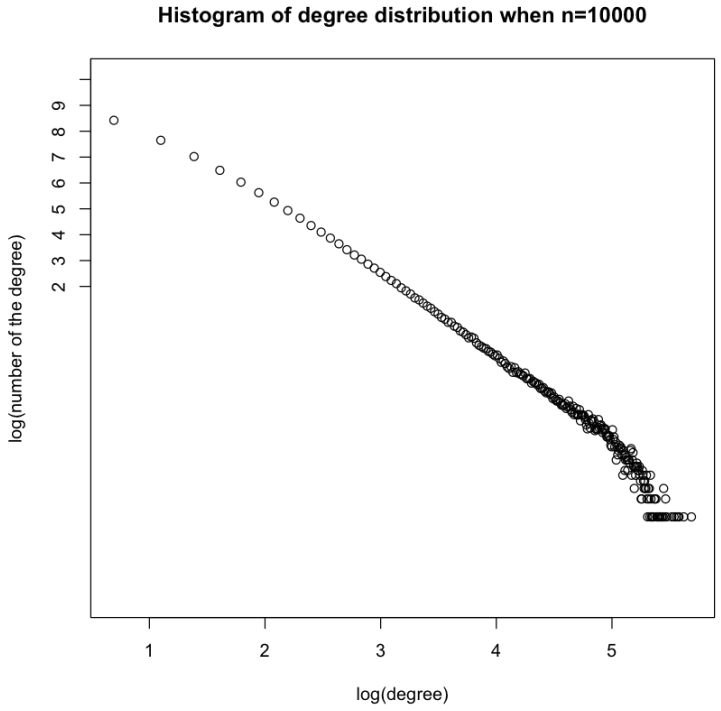


Figure 1.2.10: The plot of the degree distribution in the log-log scale

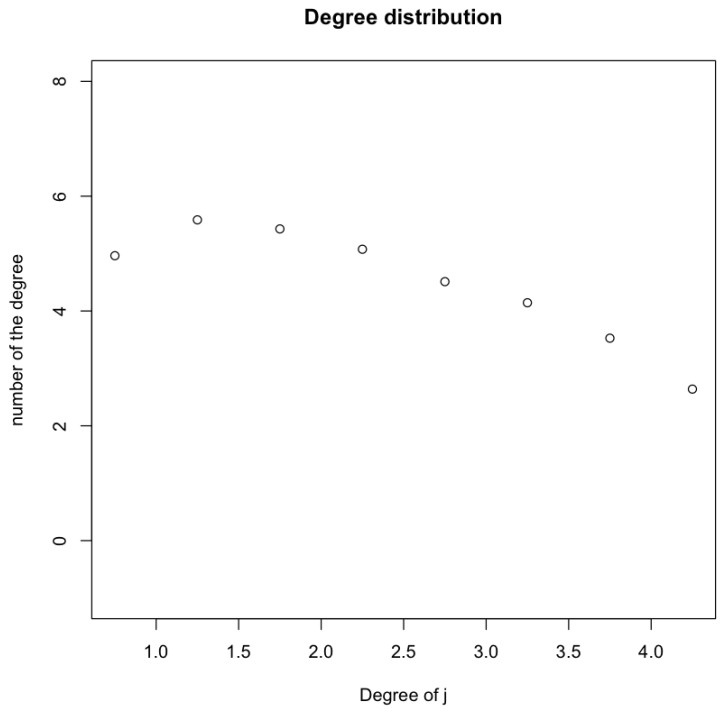


Figure 1.2.11: The plot of the degree distribution of vertex j in the log-log scale

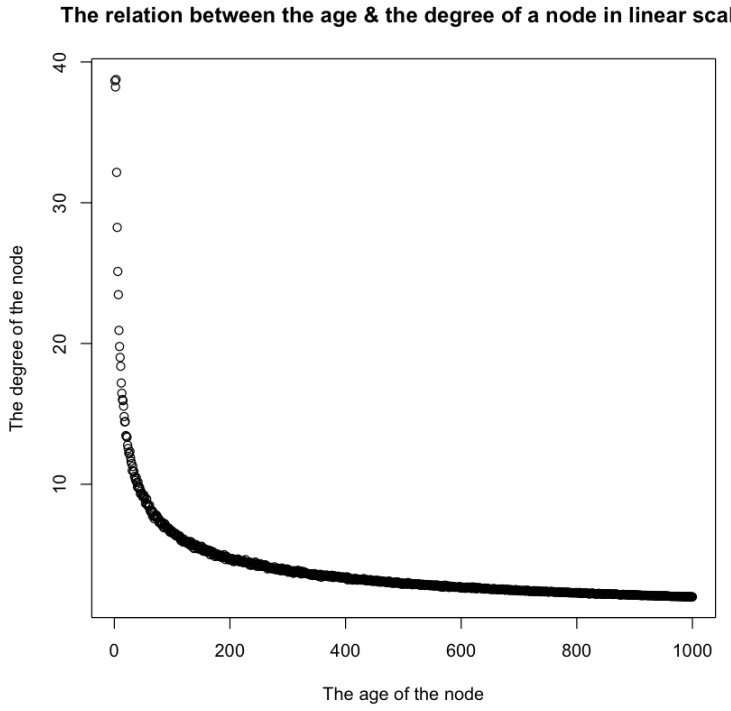


Figure 1.2.12: The plot of the degree versus the age of a node in linear scale

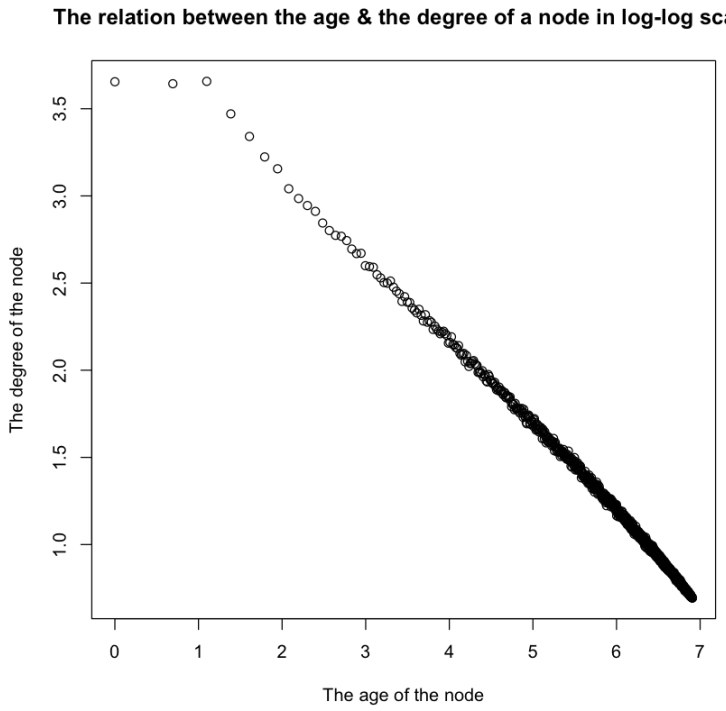


Figure 1.2.13: The plot of the degree versus the age of a node in the log-log scale

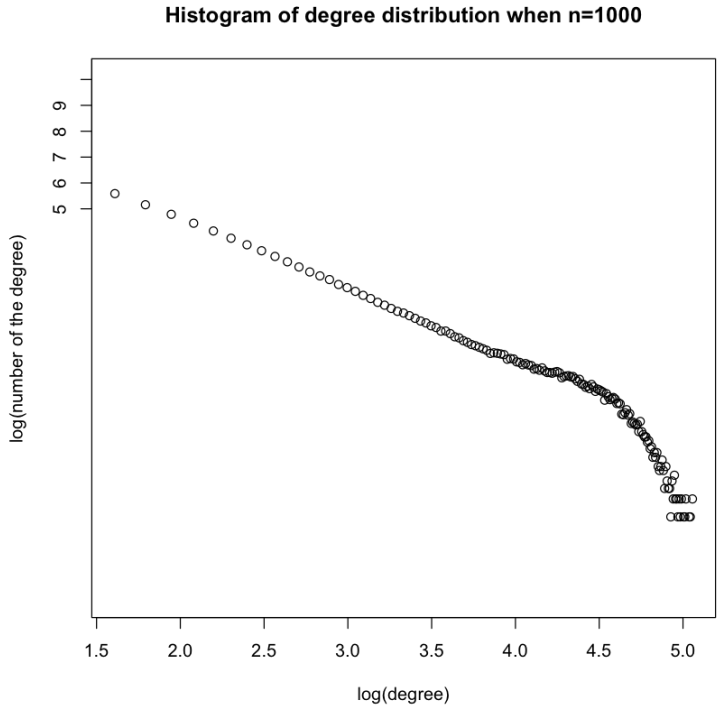


Figure 1.2.14: The plot of the degree distribution in the log-log scale

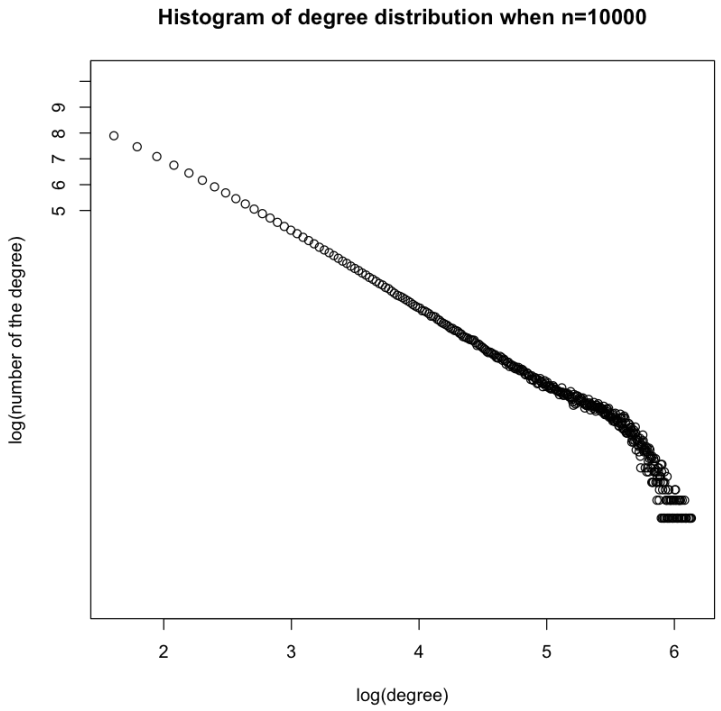


Figure 1.2.15: The plot of the degree distribution in the log-log scale

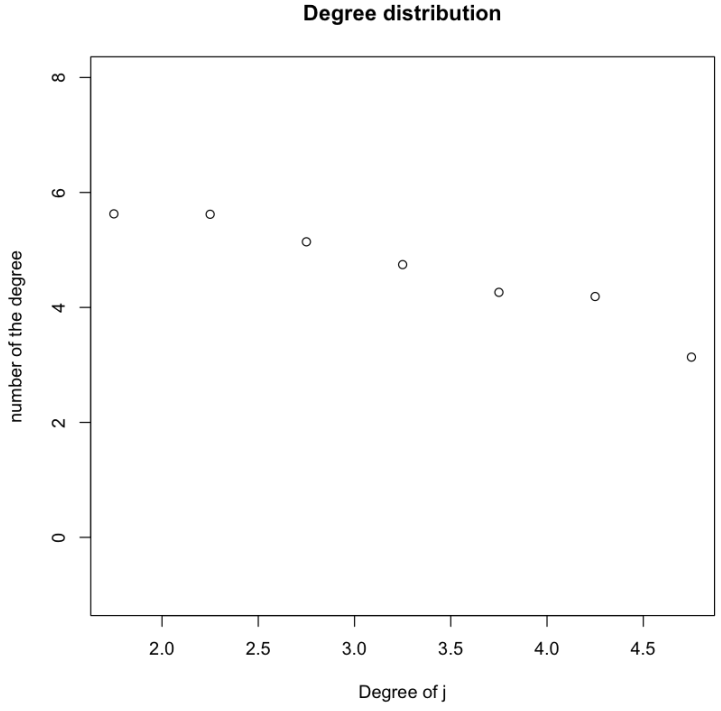


Figure 1.2.16: The plot of the degree distribution of vertex j in the log-log scale

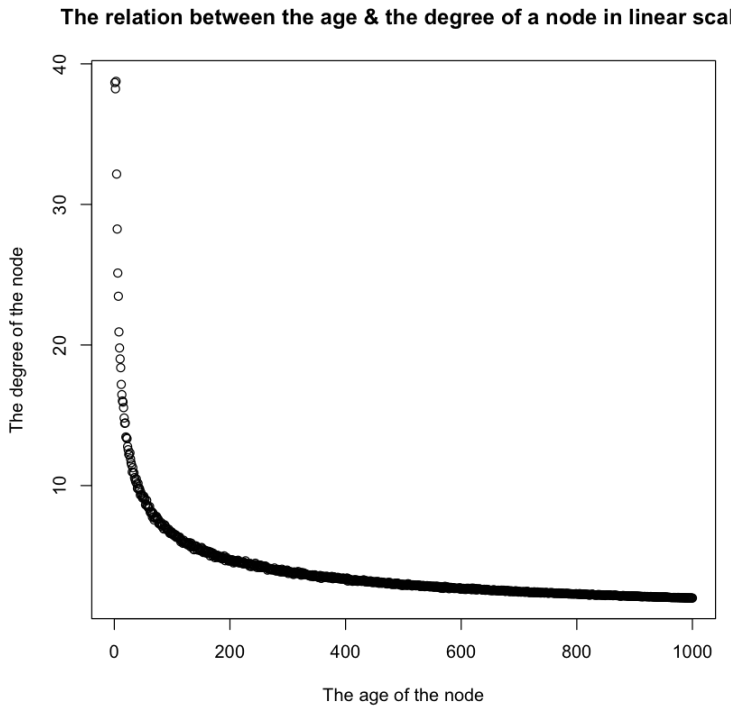


Figure 1.2.17: The plot of the degree versus the age of a node in linear scale

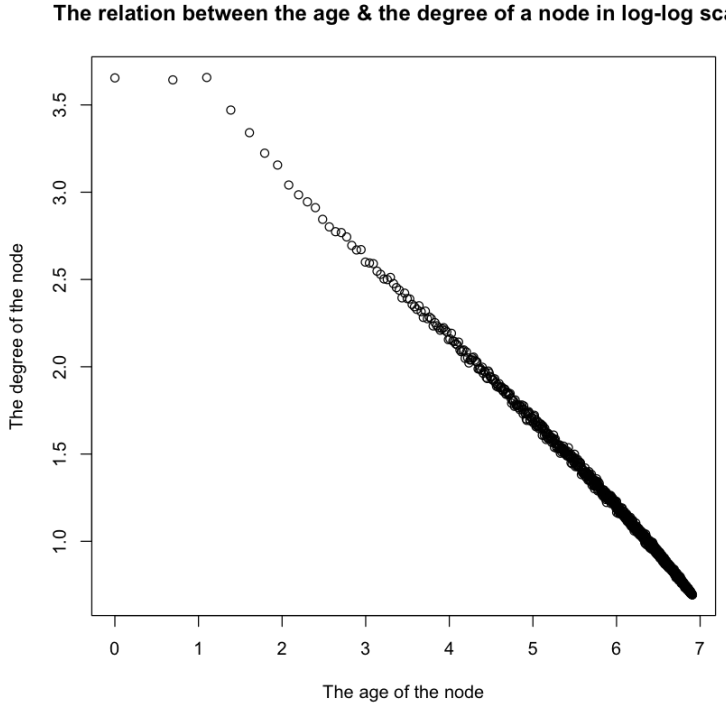


Figure 1.2.18: The plot of the degree versus the age of a node in the log-log scale

1.2.h Again, generate a preferential attachment network with n = 1000, m = 1. Take its degree sequence and create a new network with the same degree sequence, through stub-matching procedure. Plot both networks, mark communities on their plots, and measure their modularity. Compare the two procedures for creating random power-law networks.

We produce a preferential attachment network firstly, and then we mark the community of it as shown in Figure 1.2.19. After that, we generate another graph network with the same degree sequence, and definitely mark its community too, shown in Figure 1.2.20. Lastly, we measure their modularity. The former one has modularity as 0.9350 while that of the other one is 0.8464. Intuitively, we can observe that the former one is the combination of several small clusters, however, the other one has all the vertices combine together so that we couldn't distinguish them easily.

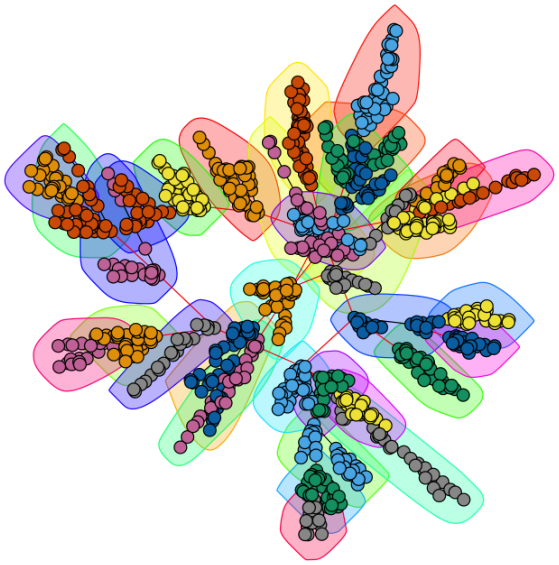


Figure 1.2.19: The plot of the graph network with preferential attachment

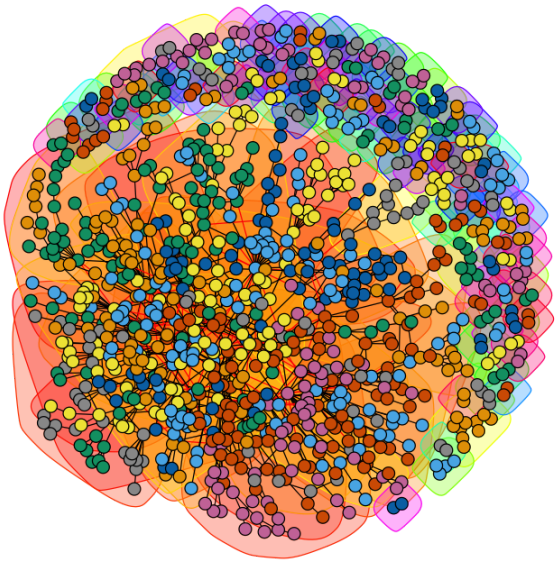


Figure 1.2.20: The plot of the graph network with same degree sequence with the former one

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

1.3.a Each time a new vertex is added, it creates m links to old vertices and the probability that an old vertex is cited depends on its degree (preferential attachment) and age. In particular, the probability that a newly added vertex connects to an old vertex is proportional to:

$$P[i] \propto (ck_i^{\alpha} + a)(dl_i^{\beta} + b)$$

where k_i is the degree of vertex i in the current time step, and l_i is the age of vertex i. Produce such an undirected network with 1000 nodes and parameters m = 1, α = 1, β = −1, and a = c = d = 1, b = 0. Plot the degree distribution. What is the power law exponent?

Firstly, we use “sample_pa_age” function to produce an undirected network requested in the problem. Figure 1.3.1 below shows the network. Then we apply ”degree” function to get the degree for each vertex and “hist” function to plot the histogram of the network. Figure 1.3.2 is the degree distribution of the network. Finally, “fit_power_law” function is employed to find the power law exponent. There are two implementation method for finding the power law exponent and they give basically the same results, alpha = 2.094.

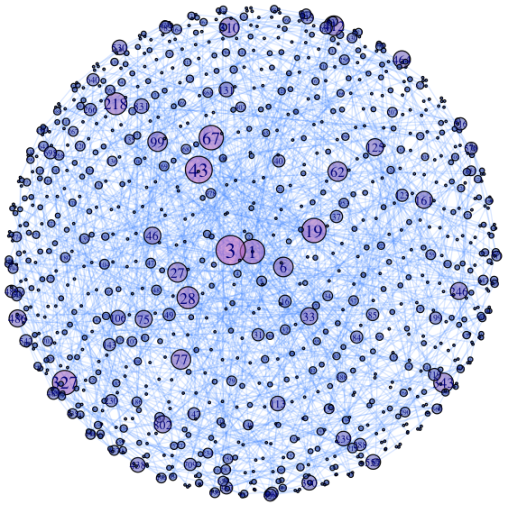


Figure 1.3.1: The plot of the network

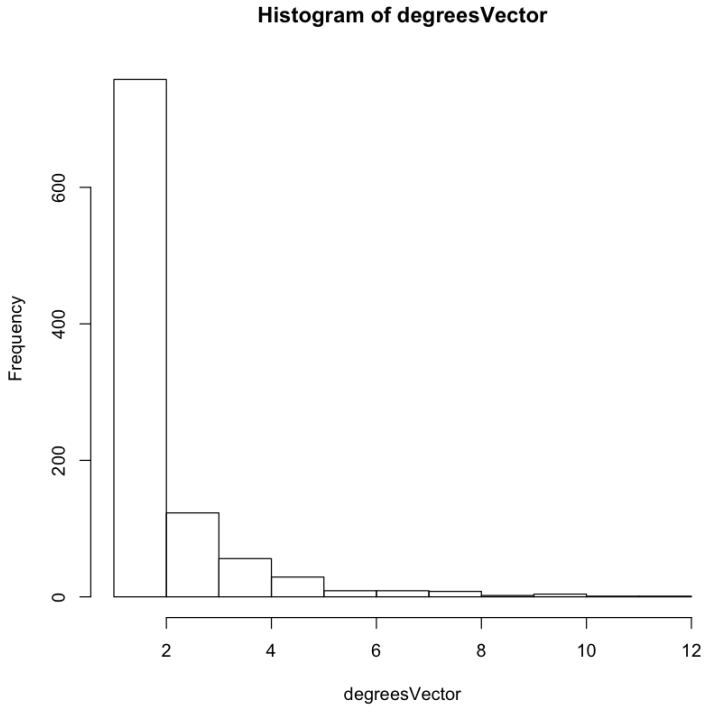


Figure 1.3.2: The degree distribution of the network

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

The modularity of this network can be determined by using fast greedy method. With this method, we can divide the whole network into 34 groups. To show it more intuitive, we plot the network with different color for vertices in different

groups, as shown below in Figure 1.3.3.

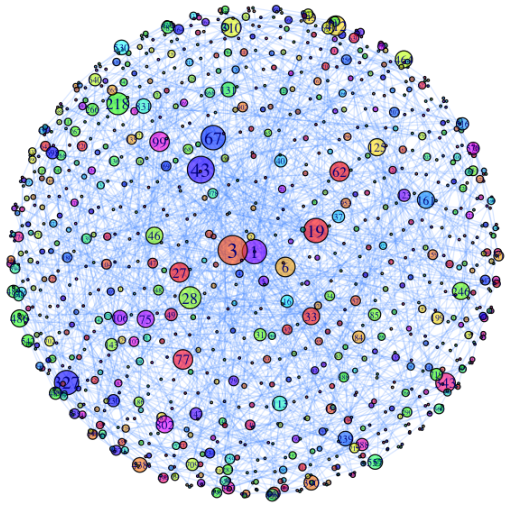


Figure 1.3.2: The degree distribution of the network

2 Random Walk on Networks

2.1.a Create an undirected random network with 1000 nodes, and the probability p for drawing an edge between any pair of nodes equal to 0.01.

We used the following command to generate the graph:

```
g = erdos.renyi.game(1000, 0.01, directed=FALSE)
```

The resulting network is shown as in Figure 2.1.1.

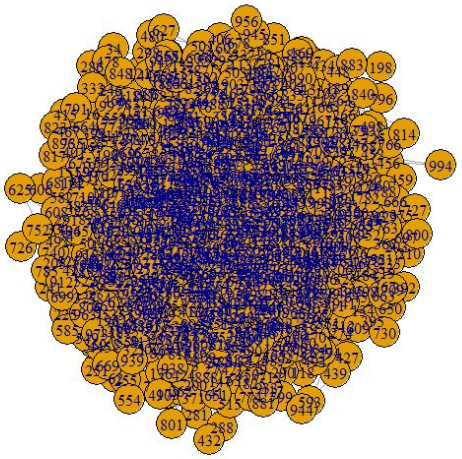


Figure 2.1.1: network

2.1.b Let a random walker start from a randomly selected node (no teleportation). We use t to denote the number of steps that the walker has taken. Measure the average distance (defined as the shortest path length of the walker from his starting point at step t. Also, measure the standard deviation of this distance.

The plots required are shown in Figure 2.1.2 & 2.1.3.

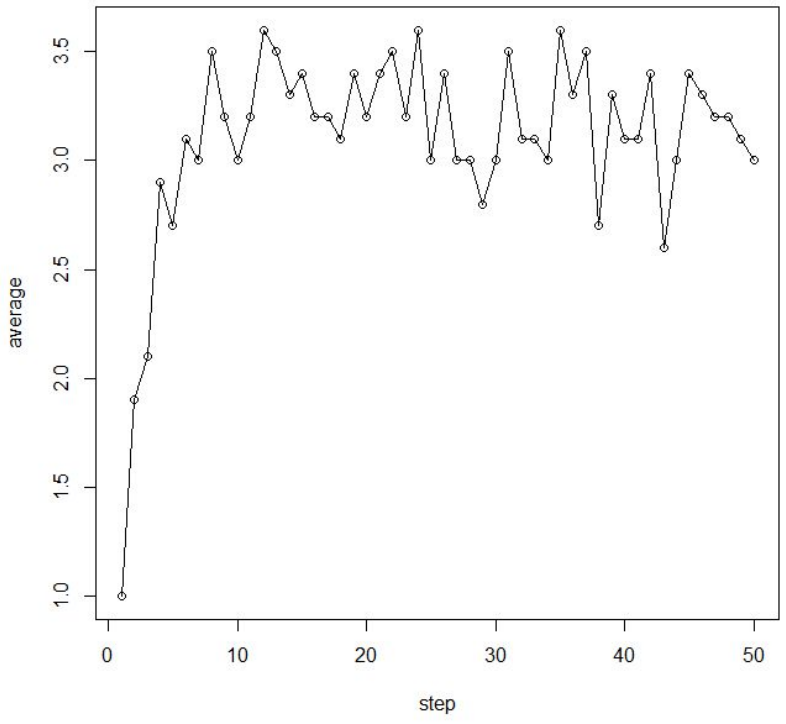


Figure 2.1.2: Plot of average s(t) versus t

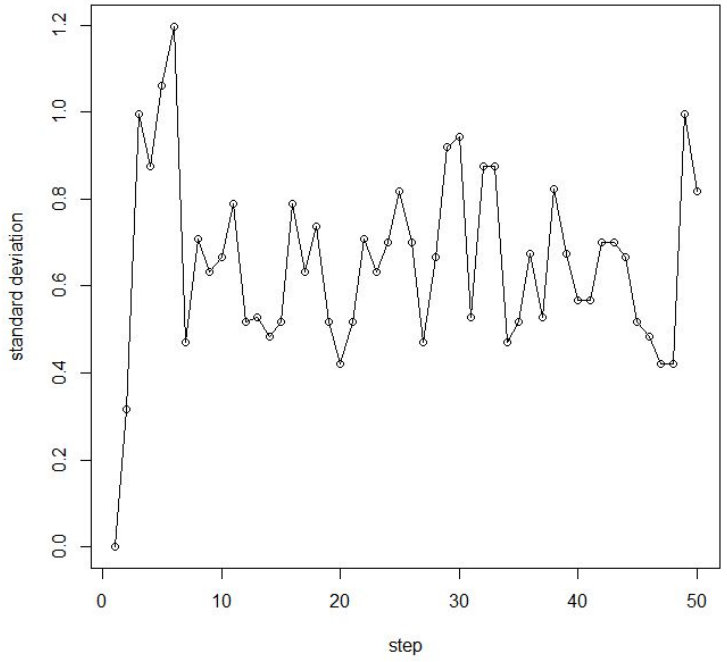


Figure 2.1.3: Plot of variance versus t.

2.1.c Measure the degree distribution of the nodes reached at the end of the random walk. How does it compare to the degree distribution of graph?

The degree distribution of the end nodes is demonstrated in Figure 2.1.4. The degree distribution of graph is demonstrated in Figure 2.1.5. The degree distribution of the nodes reached at the end of the random walk and the degree distribution of graph are quite similar: with the same degree, the frequency is close to each other.

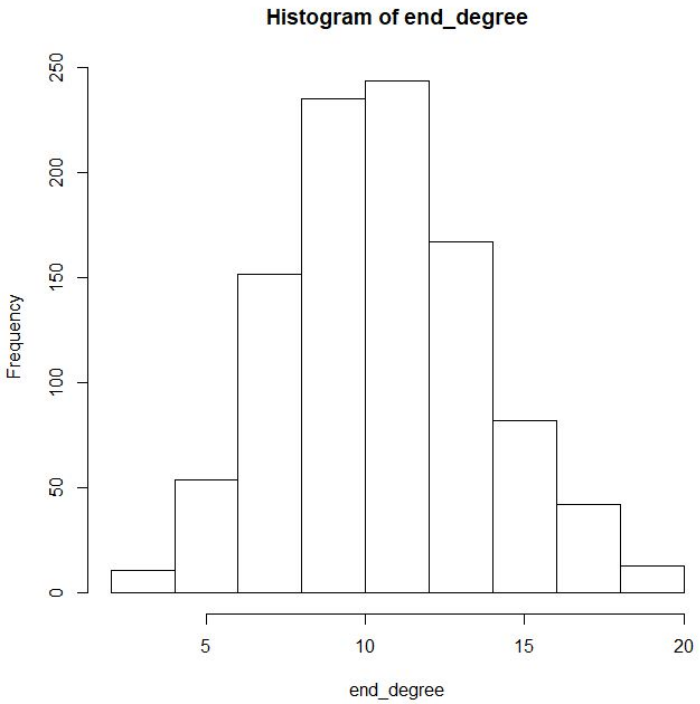


Figure 2.1.4: Degree distribution of the end nodes

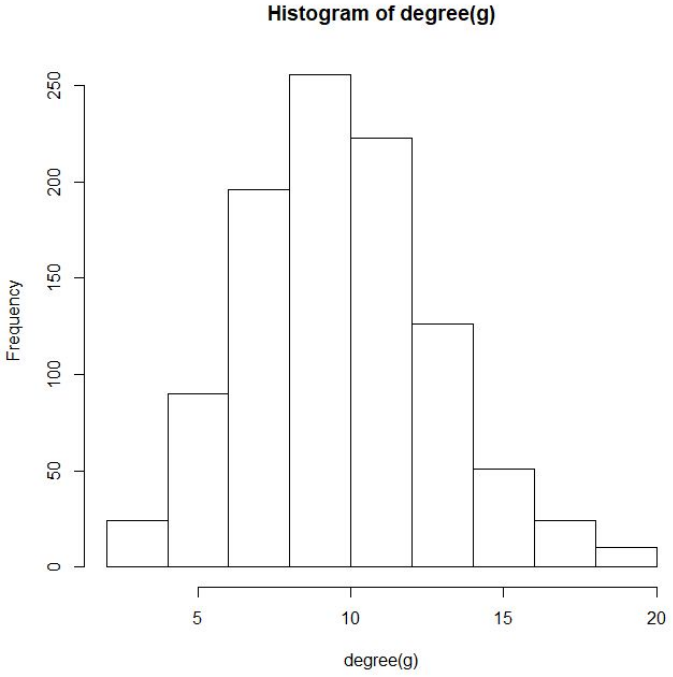
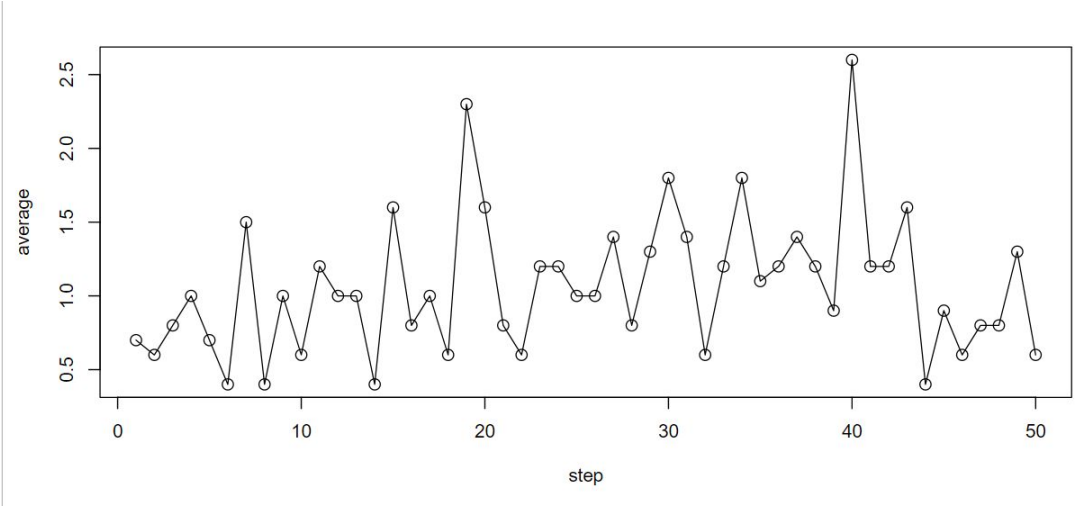


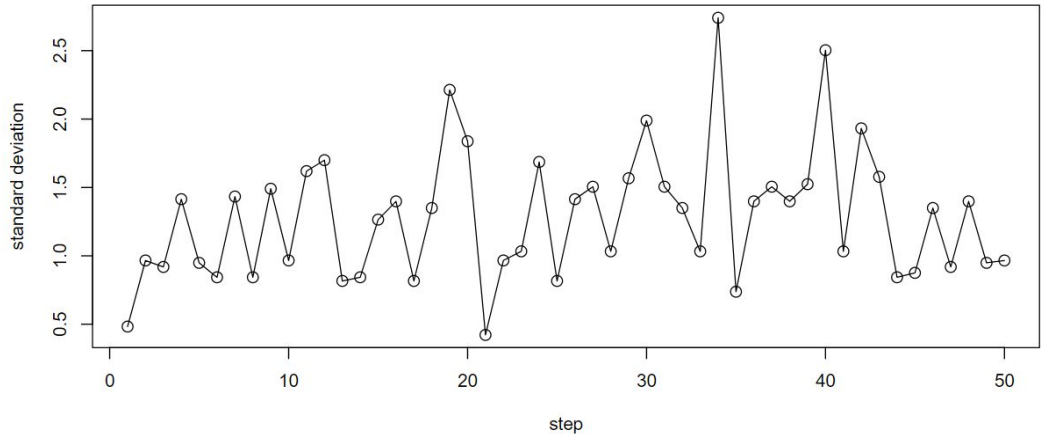
Figure 2.1.5: Degree distribution of the end nodes

2.1.d Repeat (b) for undirected random networks with 100 and 10000 nodes. Compare the results and explain qualitatively. Does the diameter of the network play a role?

The resulting plots are shown as follows:

For 100 nodes,





For 10000 nodes,

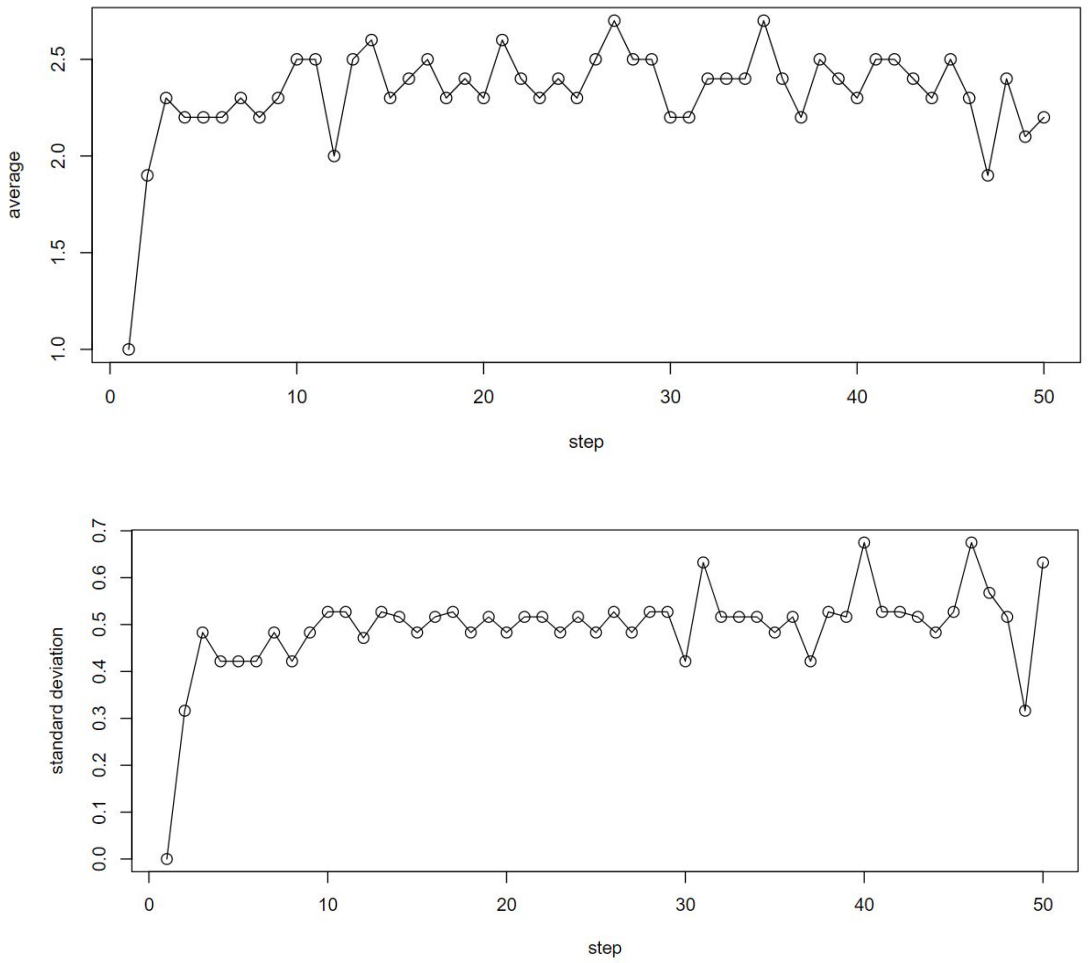


Figure 2.1.6: Plots for average versus t and variance versus t for 100 nodes and 10000 nodes.

The diameter affect the plots in Figure 2.1.6. If diameter of the graph is large, the final average shortest distance will also be large.

2.2.a Generate an undirected preferential attachment network with 1000 nodes, where each new node attaches to m = 1 old nodes.

I used the following command to generate undirected preferential attachment network with 1000 nodes:

```
g = sample_pa(1000, m = 1, directed=FALSE)
```

The plot of the network is shown in Figure 2.2.1.

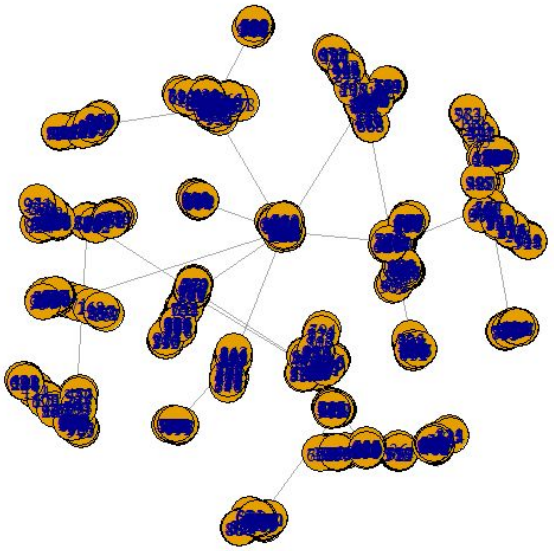
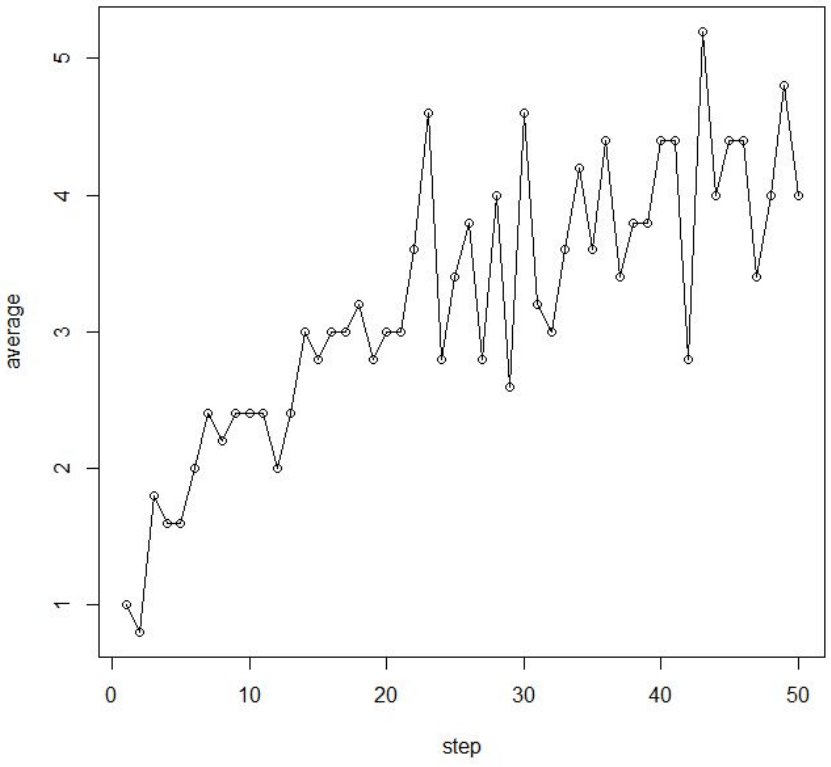


Figure 2.2.1: Network

2.2.b Let a random walker start from a randomly selected node. Measure and plot.

The plots of the random walk are shown in Figure 2.2.2.



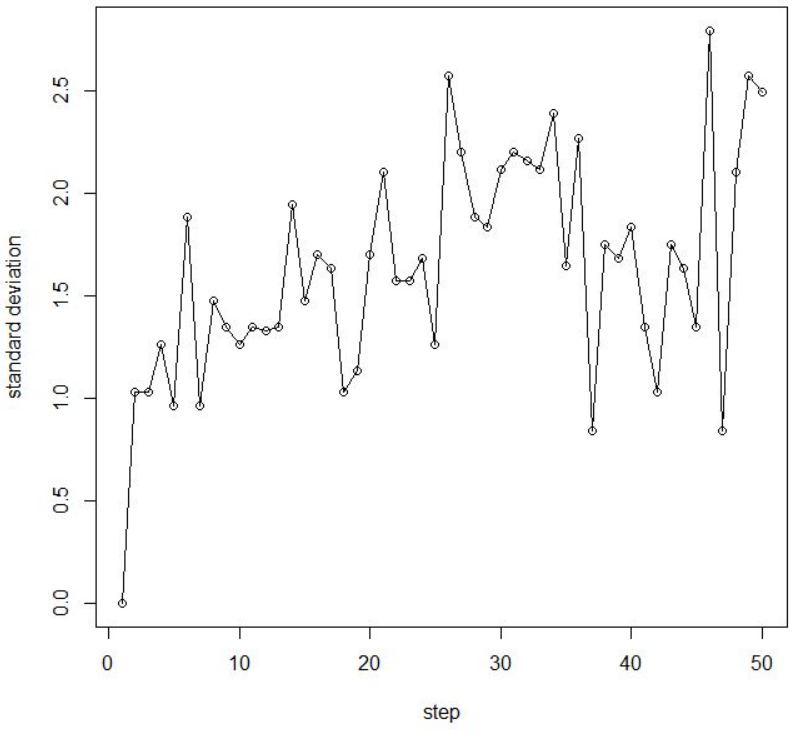


Figure 2.2.2: Plots of average s(t) versus t and its variance versus t.

2.2.c Measure the degree distribution of the nodes reached at the end of the random walk on this network. How does it compare with the degree distribution of the graph?

The plot of the degree distribution of the last nodes in the random walk is shown in Figure 2.2.3. The plot of the degree distribution of the graph is shown in Figure 2.2.4. The degree distribution of the nodes reached at the end of the random walk and degree distribution of graph are quite similar. With the same degree, the frequency is close to each other. But, the degrees of the graph are more concentrated in the range of degree from 1 to 5.

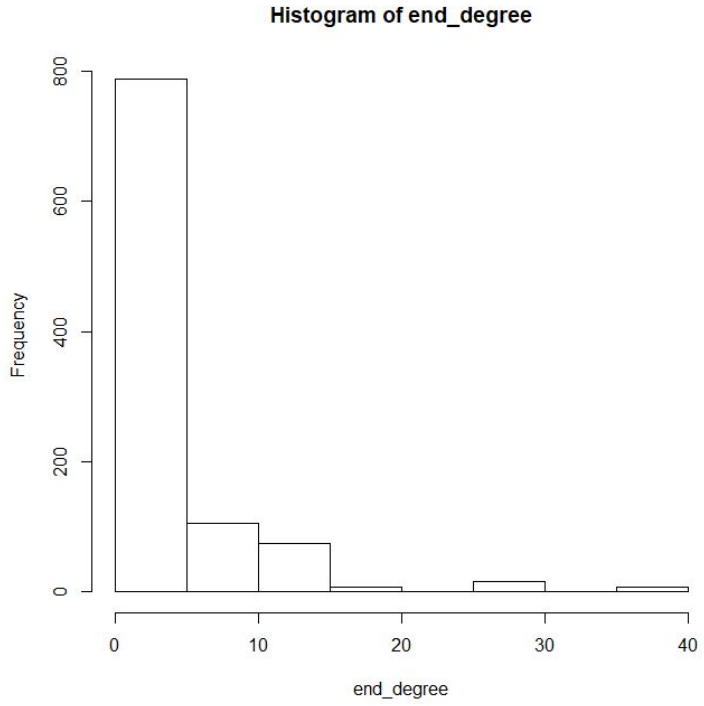


Figure 2.2.3: Degree distribution of the last node in random walk.

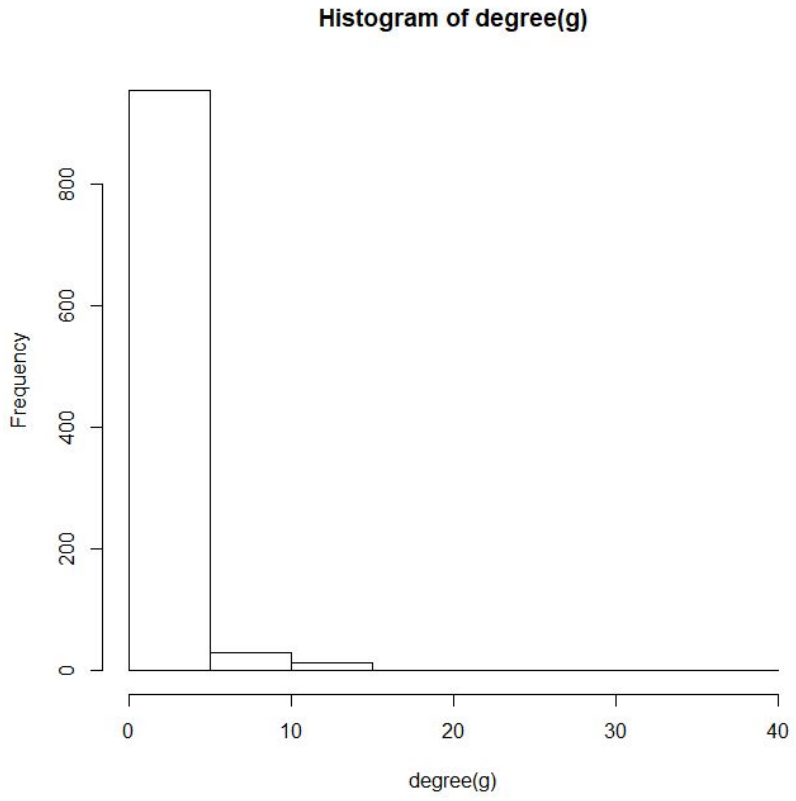
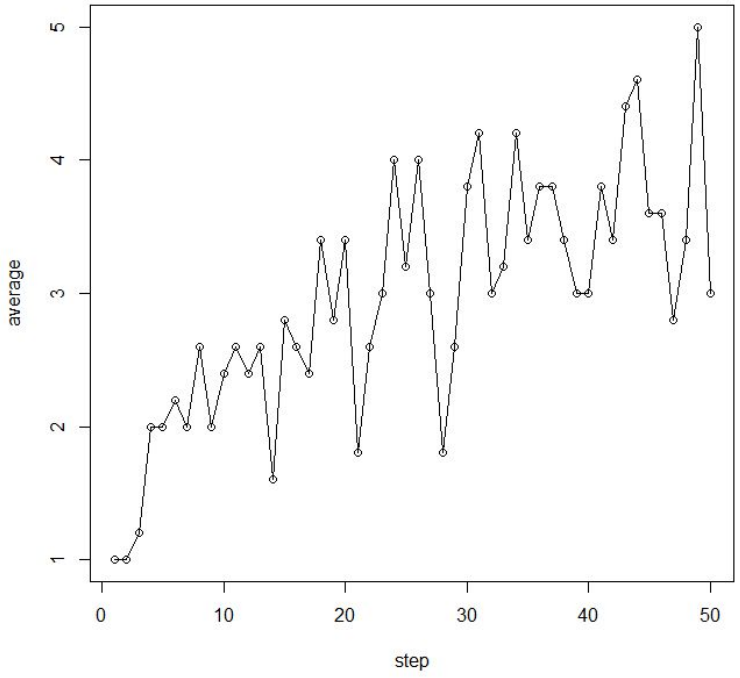


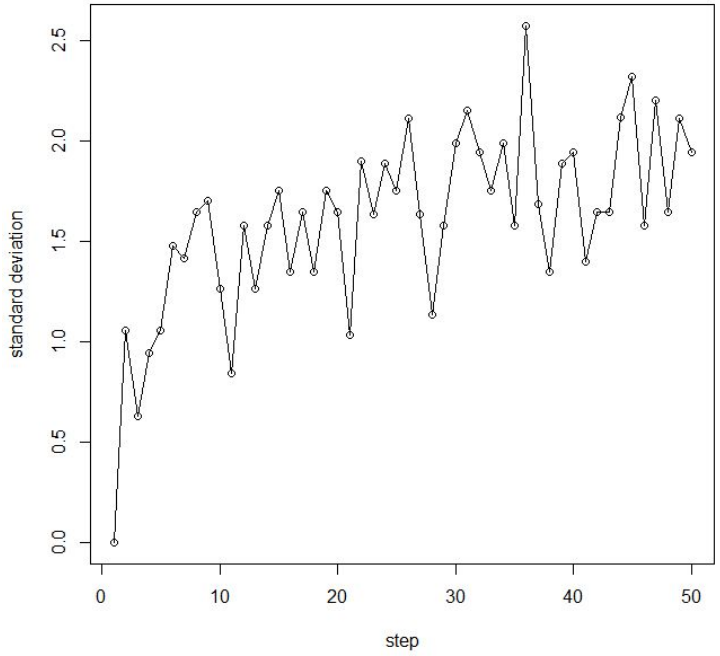
Figure 2.2.4: Degree distribution of the graph.

2.2.d Repeat (b) for preferential attachment networks with 100 and 10000 nodes, and m = 1. Compare the results and explain qualitatively. Does the diameter of the network play a role?

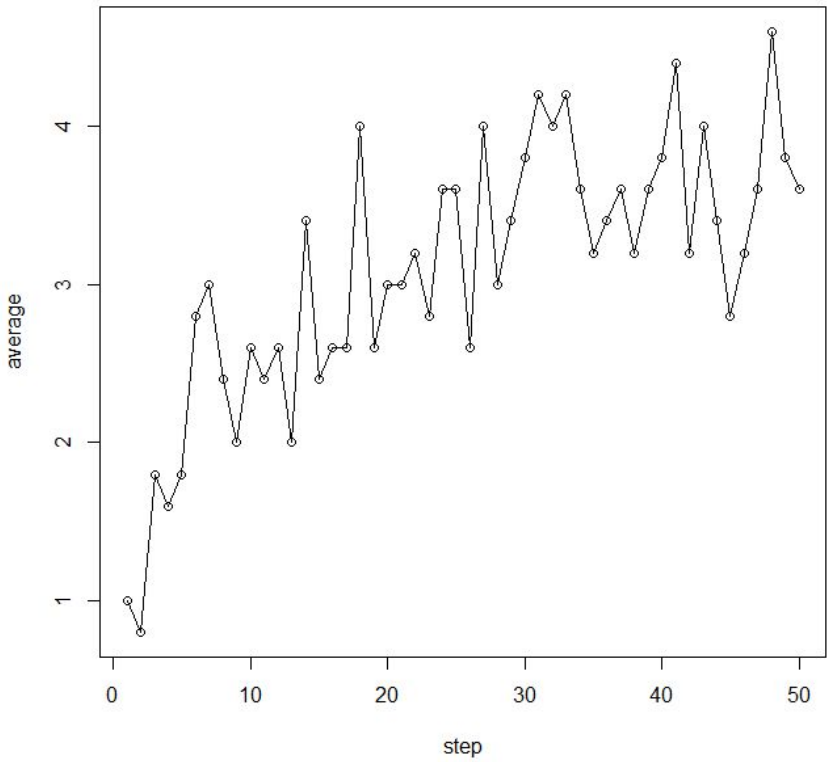
The resulting plots are shown as follows:

For 100 nodes,





For 10000 nodes,


average
step
1
2
3
4
0
10
20
30
40
50

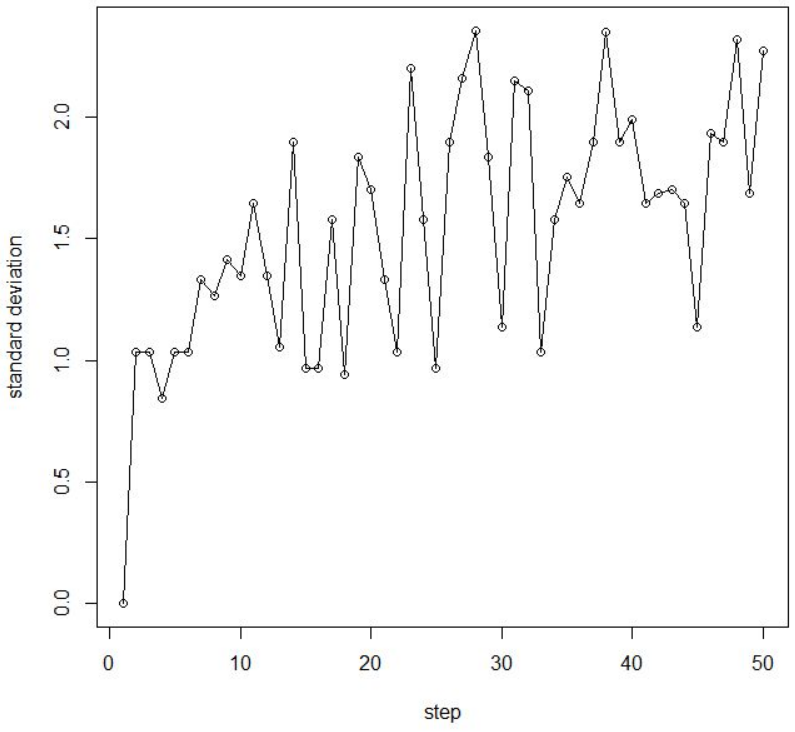

standard deviation
step
0.0
0.5
1.0
1.5
2.0
0
10
20
30
40
50

Figure 2.2.5: Plots of average s(t) versus t and its variance versus t for 100 and 10000 nodes separately

The diameter indeed plays a role. If diameter of the graph is large, the final average shortest distance will also be large.

2.3 PageRank

2.3.a Create a directed random network with 1000 nodes, using the preferential attachment model, where m = 4. Note that in this directed model, the out-degree of every node is m, while the in-degrees follow a power law distribution. Measure the probability that the walker visits each node. Is this probability related to the degree of the nodes?

We use function *'age.prefatt.game'* to create a directed preferential attachment graph with 1000 nodes where each node has out-degree of 4. Then we initialize a random walk across the graph with damping = 1. Since the very first node is a dead node, we set a policy that the walk randomly teleport to any other node when it goes into the dead node. Then, we measure the probability of the walk visiting each node. The result is shown in Figure 2.3.1 & 2.3.2. In Figure 2.3.1, we can observe that old nodes are more likely to be visited by the random walk. According to Figure 2.3.2, we can tell a rough trend that probability of the walk visiting a node is relatively large if the node's degree is relatively large. The correlation between probability and degree is 0.851.

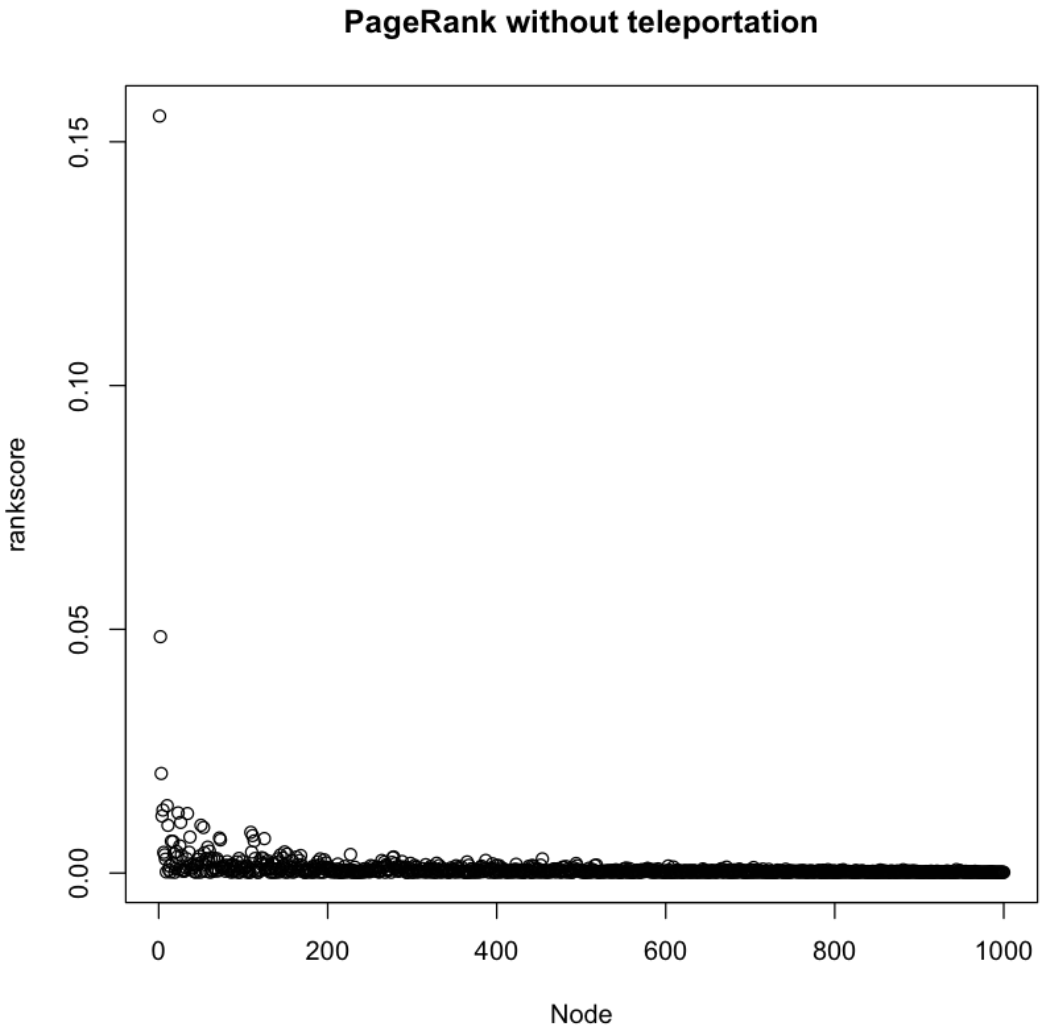


Figure 2.3.1: The probability of each node being visited by the walk against node ID with damping = 1.

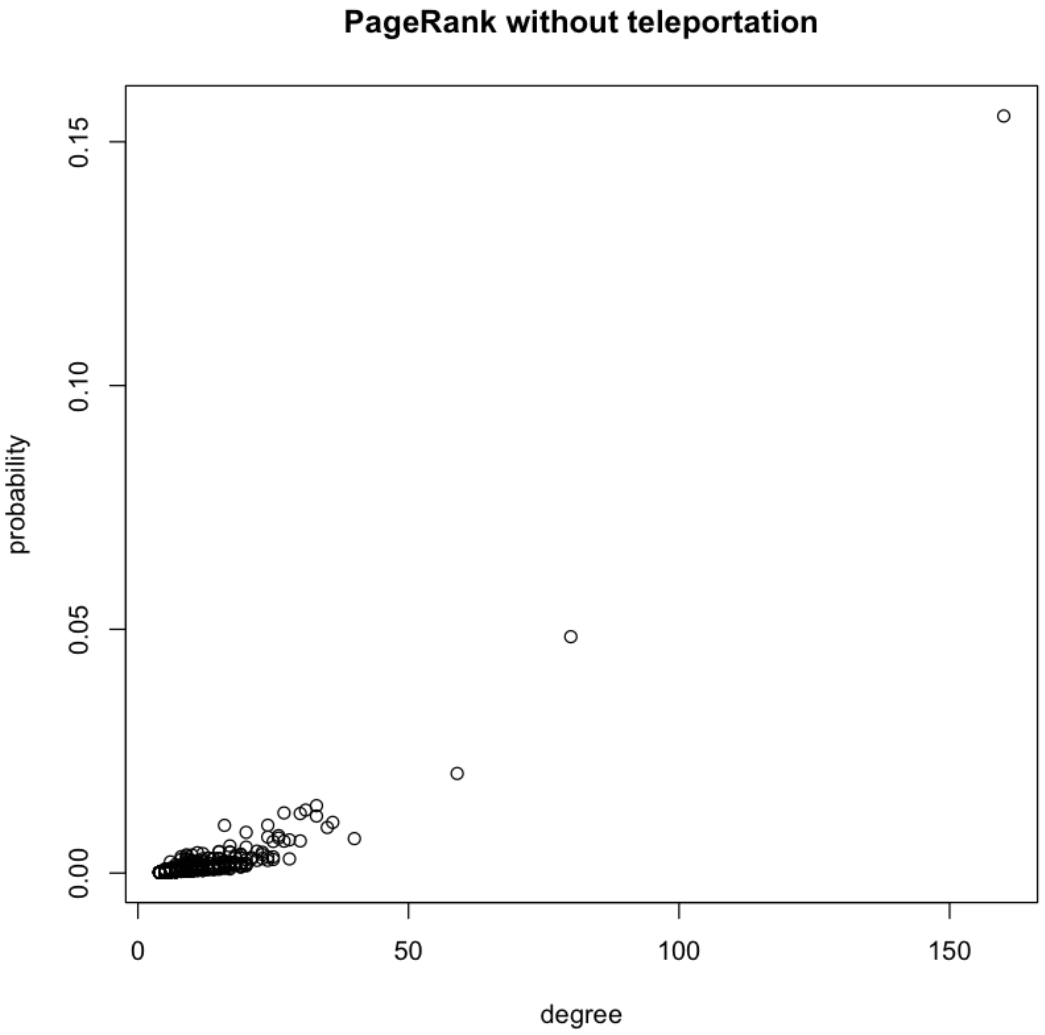


Figure 2.3.2: The probability of each node being visited by the walk against degree of each node with damping = 1.

Another way to implement the random walk is that the walk stays at the dead node when it goes into it. We call this type of random walk as ***"Special Random Walk"***.

***Special Random Walk*:**

The results are shown in Figure 2.3.1.s & 2.3.2.s. We can see that one node has especially high probability of being visited, which is almost 1. This is the oldest node in the network, which has largest in-degree. This result can be explained by the argument that the oldest node has the most incoming edges, and hence is the most likely to be visited by the walk. Then, it has not outgoing edge, so the walk is strapped in this kind of dead node. That's why this node has a extremely high probability while the other nodes are rarely visited by the walk. In the same way, the probability of visiting a node is highly related to its in-degree, and hence degree, though the correlation score becomes 0.512 in this implementation.

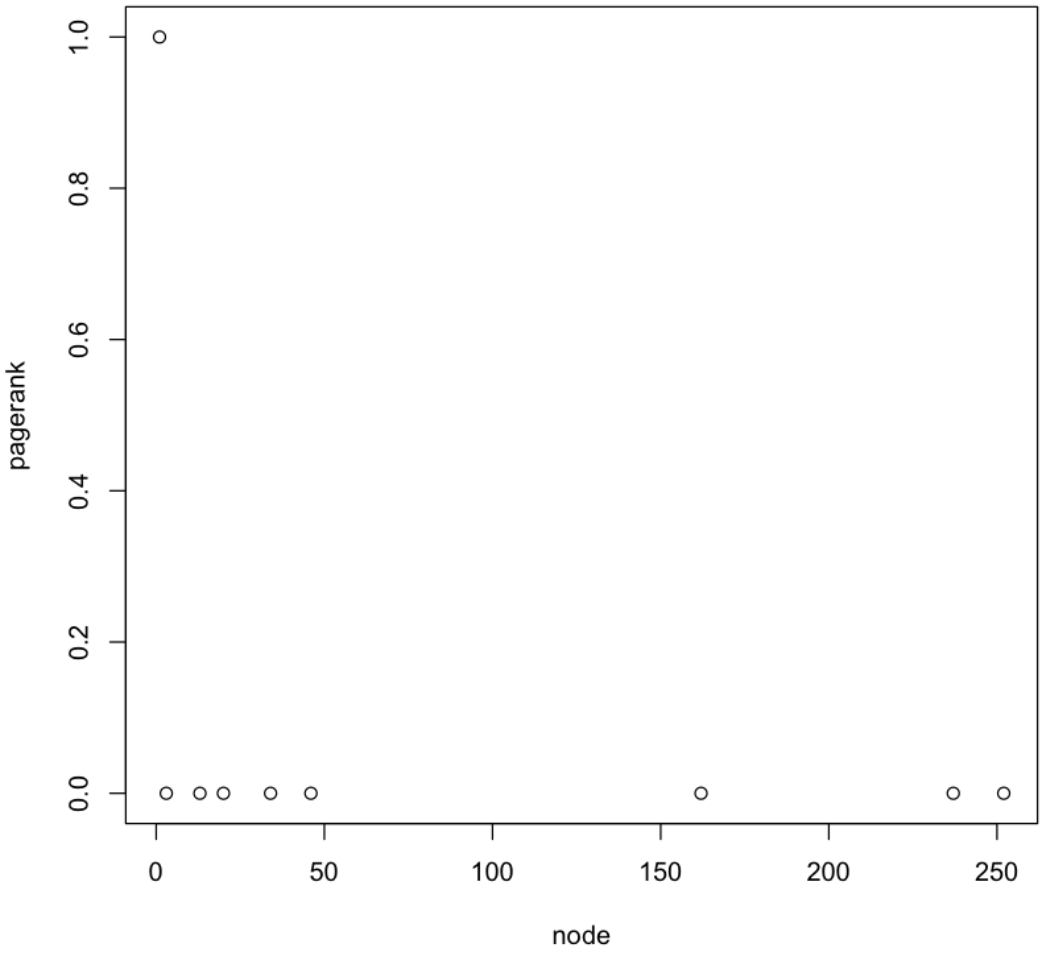


Figure 2.3.1.s: The probability of each node being visited by the walk against node ID with damping = 1.

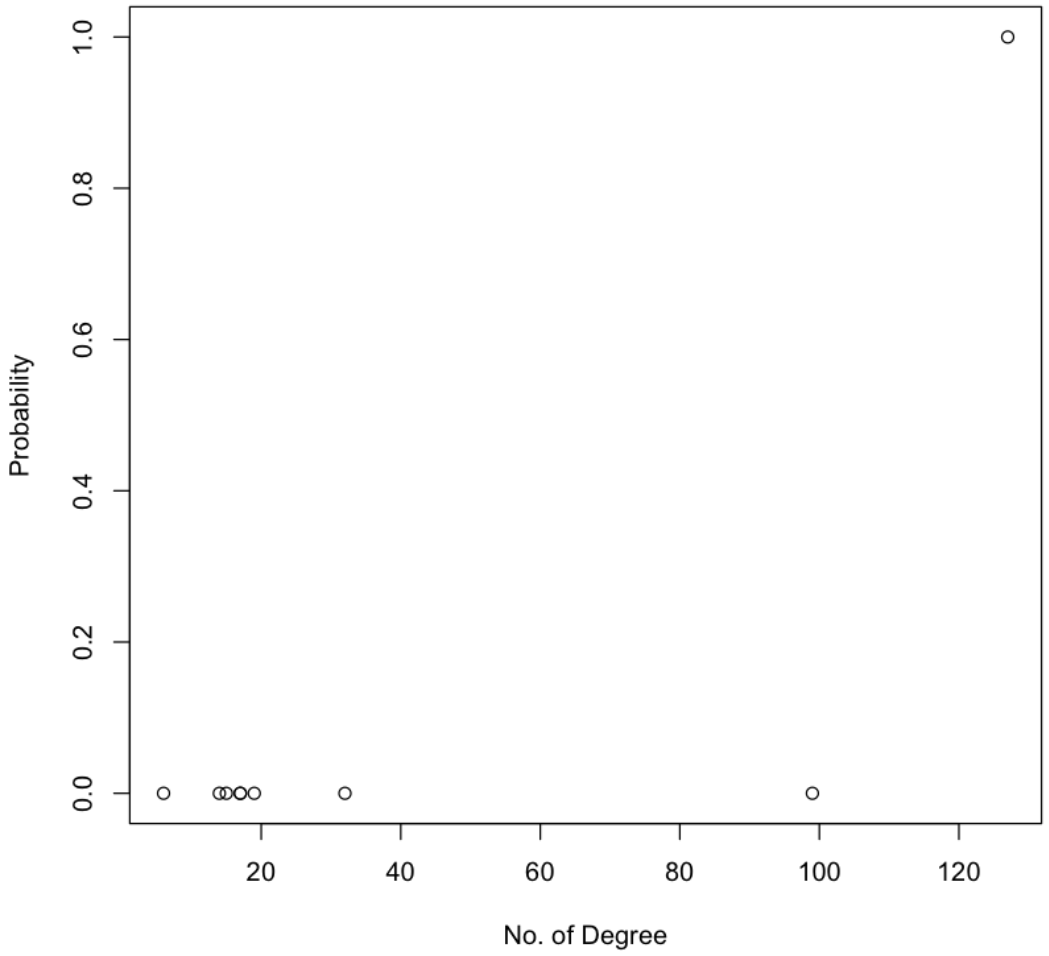


Figure 2.3.2.s: The probability of each node being visited by the walk against degree of each node with damping = 1.

2.3.b In all previous questions, we didn't have any teleportation. Now, we use a teleportation probability of alpha = 0.15. By performing random walks on the network created in 3(a), measure the probability that the walker visits each node. Is this probability related to the degree of the node?

We initialize a random walk across the graph with damping = 0.85 with other setting the same as in 2.3.a. The results are shown in Figure 2.3.3 & 2.3.4. We can find that probability of the walk visiting a node is more related to the node's degree, compared with the one in 2.3.a. The correlation between probability and degree is 0.877, which is also higher than that in 2.3.a. The pattern that old nodes tend to have high probability of being visited still exists. It is because the old nodes tend to have greater in-degrees.

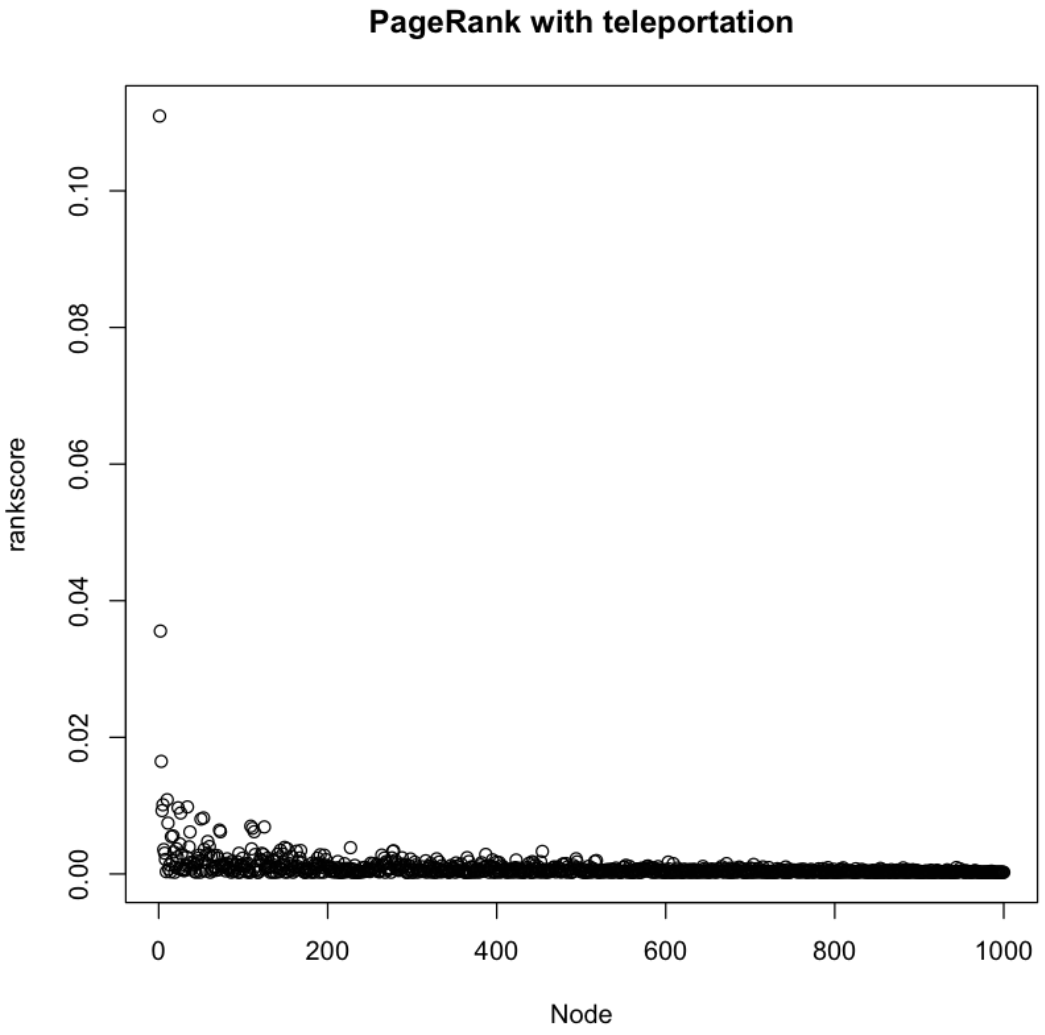


Figure 2.3.3: The probability of each node being visited by the walk against node ID with damping = 0.85.

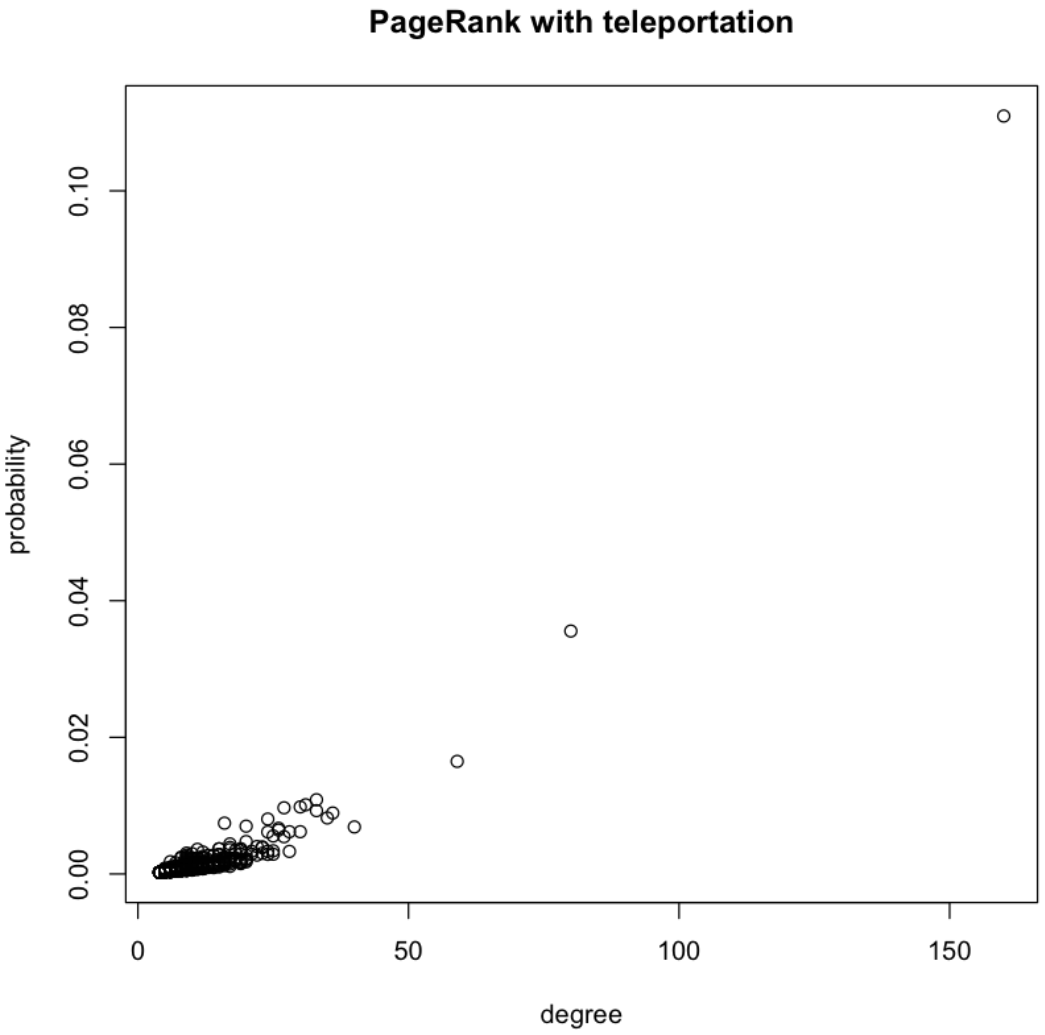


Figure 2.3.4: The probability of each node being visited by the walk against degree of each node with damping = 0.85.

Special Random Walk:

The results are shown in Figure 2.3.3.s & 2.3.4.s. We can see that one node has especially high probability of being visited, which is almost 0.5. This is the oldest node in the network, which has largest in-degree. This result can be explained by the argument that the oldest node has the most incoming edges, and hence is the most likely to be visited by the walk. Then, it has not outgoing edge, so the walk is strapped in this kind of dead node, unless the teleportation is called. That's why this node has a extremely high probability while the other nodes are rarely visited by the walk. In the same way, the probability of visiting a node is highly related to its in-degree, and hence degree. The walk is especially interested in the nodes which have large in-degree. The correlation score is 0.579, which is higher than special random walk without teleportation.

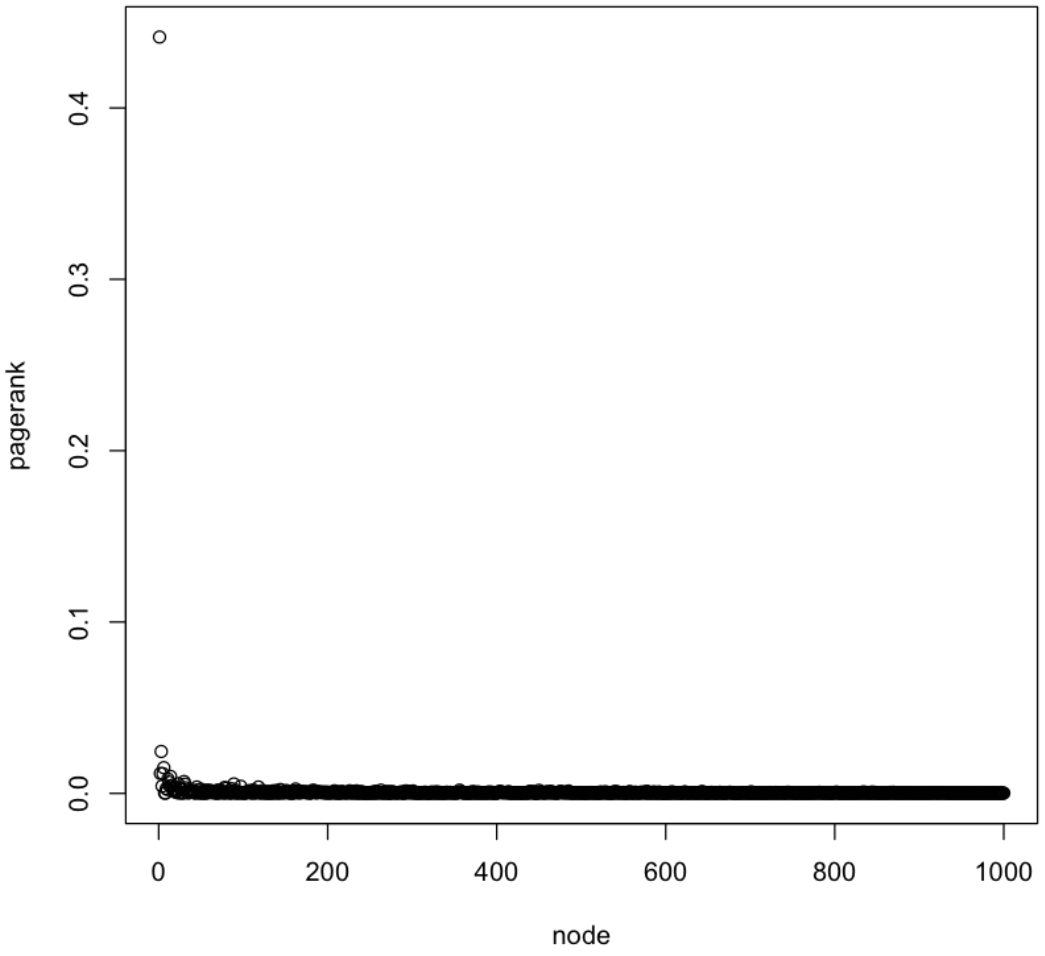


Figure 2.3.3.s: The probability of each node being visited by the walk against node ID.

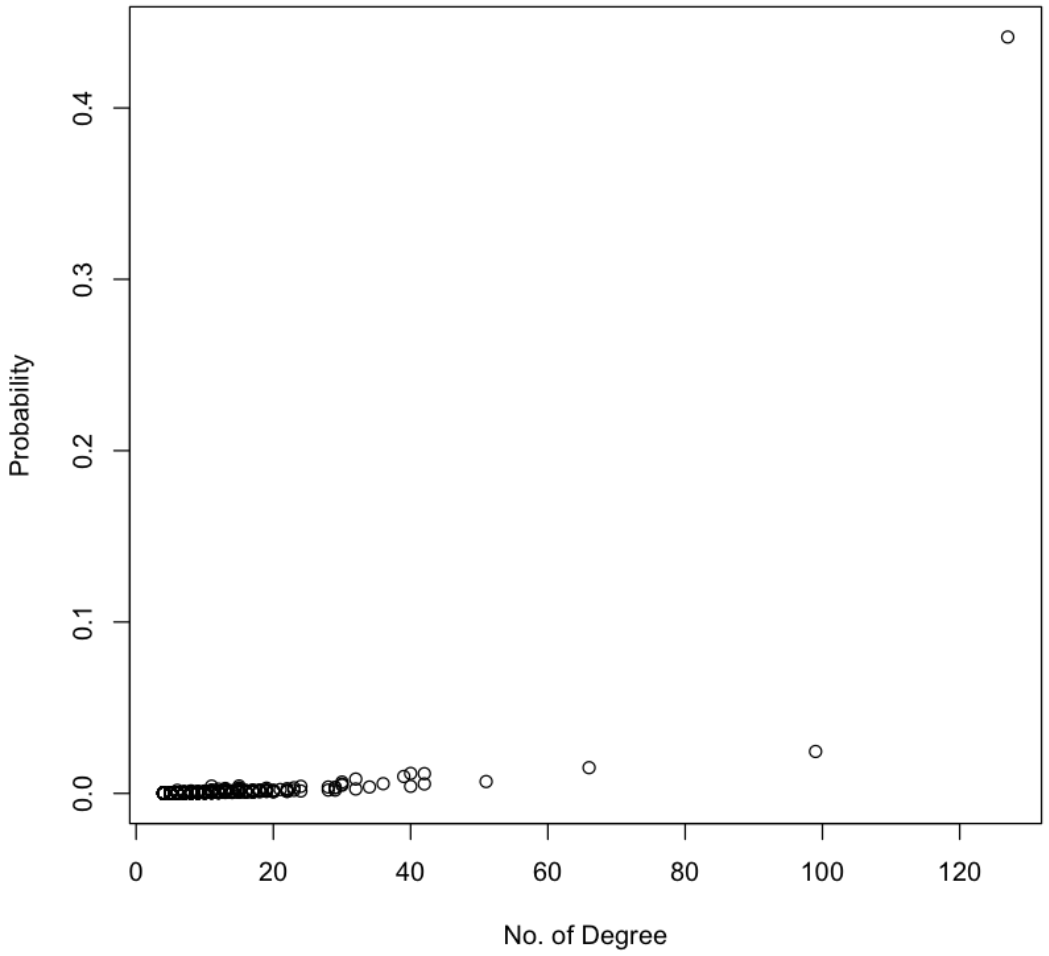


Figure 2.3.4.s: The probability of each node being visited by the walk against degree of each node.

2.4 Personalized PageRank

2.4.a Suppose you have your own notion of importance. Your interest in a node is proportional to the node's PageRank, because you totally rely upon Google to decide which website to visit (assume that these nodes represent websites). Again, use random walk on network generated in part 3 to simulate this personalized PageRank. Here the teleportation probability to each node is proportional to its PageRank (as opposed to the regular PageRank, where at teleportation, the chance of visiting all nodes are the same and equal to 1). Again, let the teleportation probability be equal to alpha = 0.15. Compare the results with 3(b).

The part that is modified in this question is the probability of teleportation to each node is no longer uniform. Instead, the probability is proportional to the pagerank score of each node. The results are shown in Figure 2.4.1 & 2.4.2. The probability of visiting each node is less related to the degree of each node, proven by the correlation which is 0.859, smaller than that in 2.3.b.

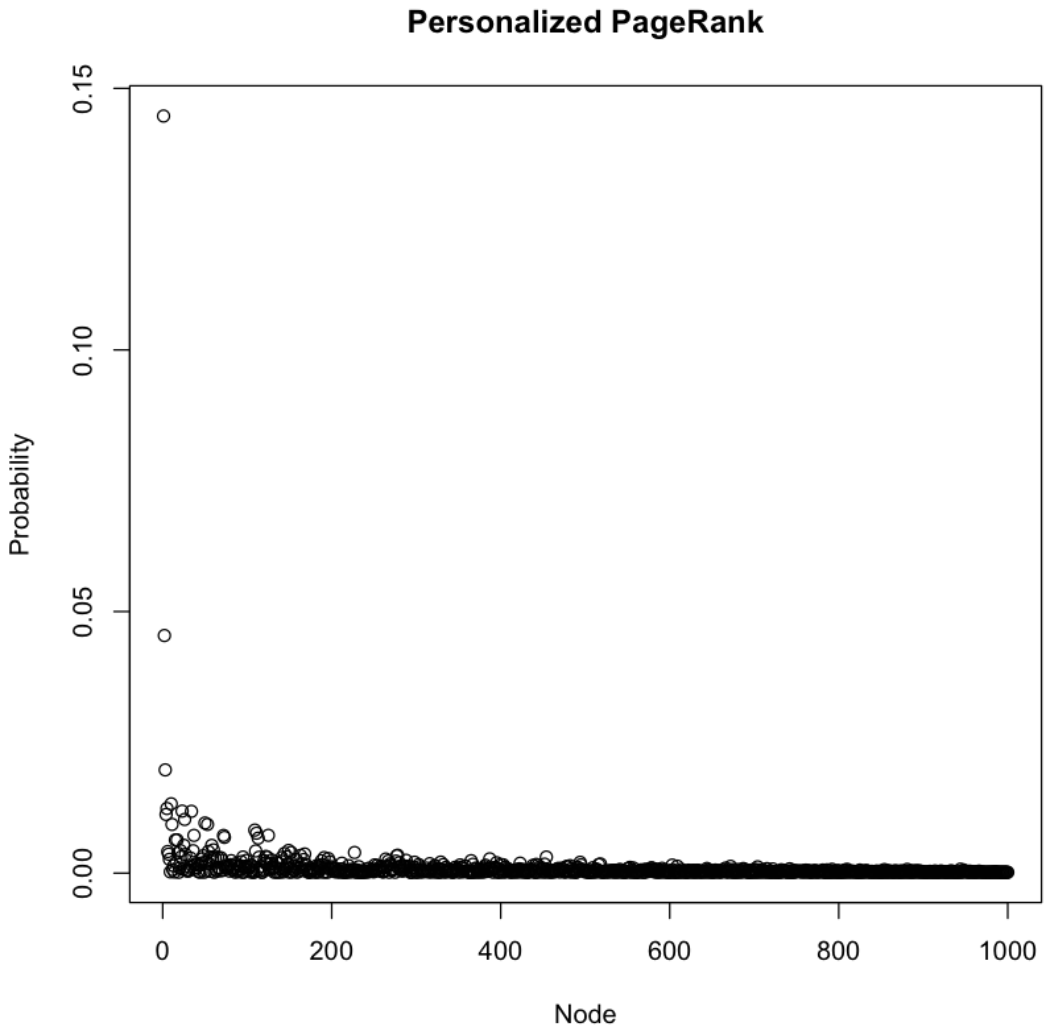


Figure 2.4.1: The probability of each node being visited by the walk against node ID with damping = 0.85. (personalized PageRank)

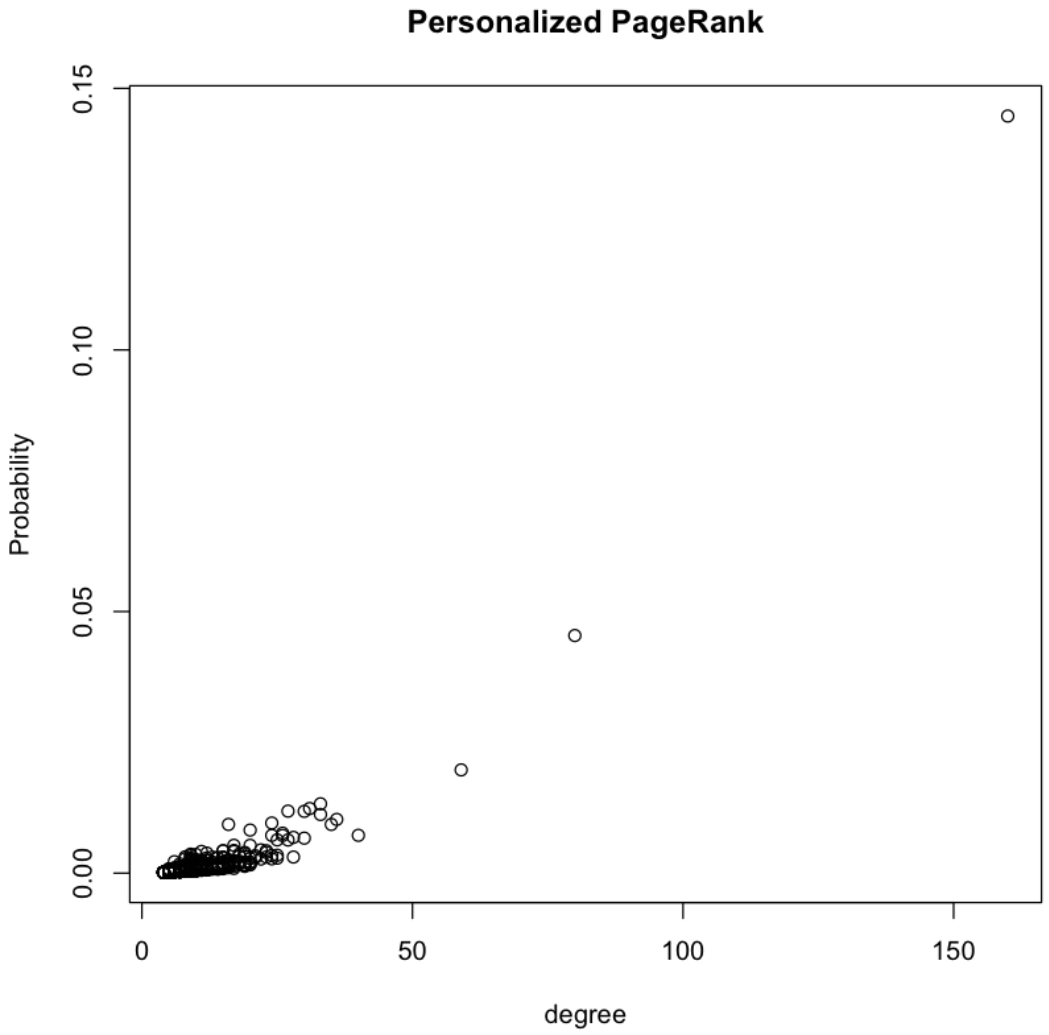


Figure 2.4.2: The probability of each node being visited by the walk against degree of each node with damping = 0.85. (personalized PageRank)

***Special Random Walk*:**

The results are shown in Figure 2.4.1.s & 2.4.2.s. In comparison with the Figure in 2.3.b, there is no big difference in pagerank scores. The oldest node still occupies a typically high probability. The correlation score between the visiting probability and the degree is 0.539, which is a bit lower than that in 2.3.b. We can observe that the interesting nodes become more likely to be visited, while the less interesting nodes become less likely to be visited. The probabilities of nodes are more polarized.

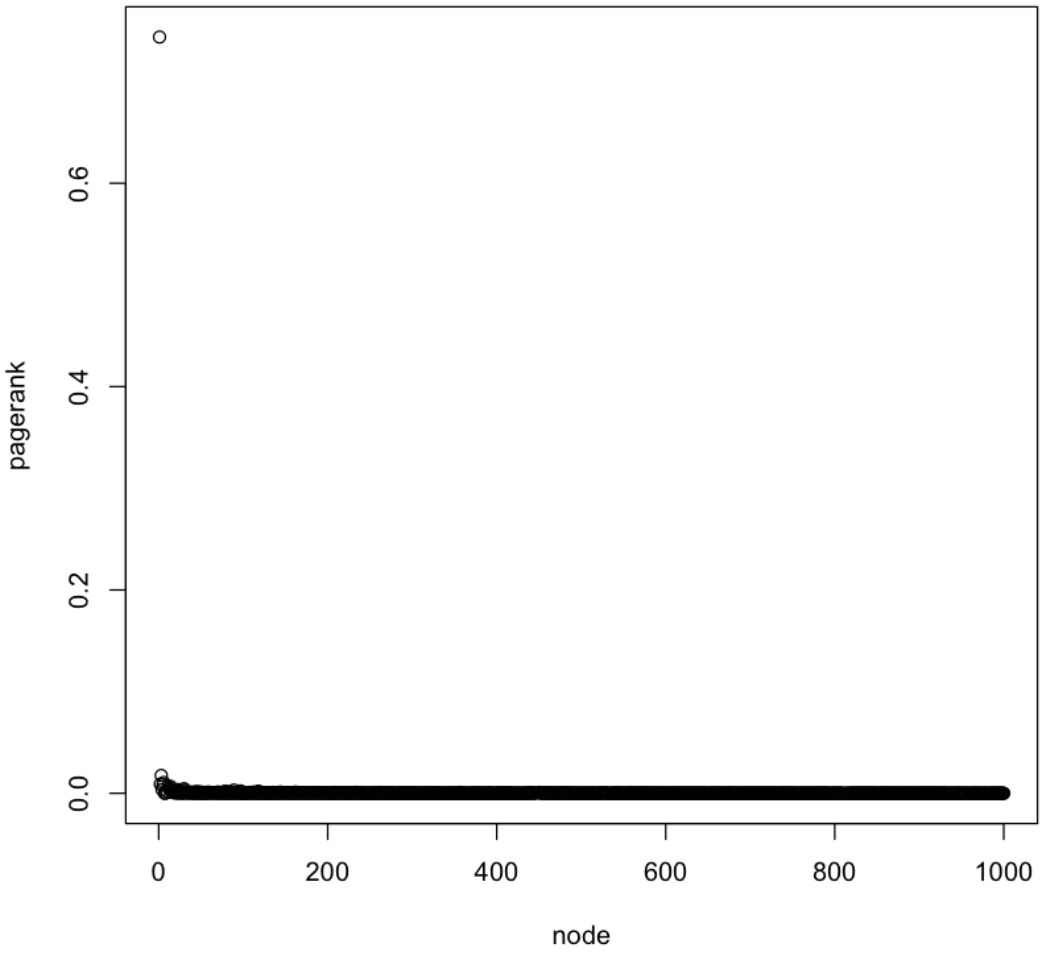


Figure 2.4.1.s: The probability of each node being visited by the walk against node ID.

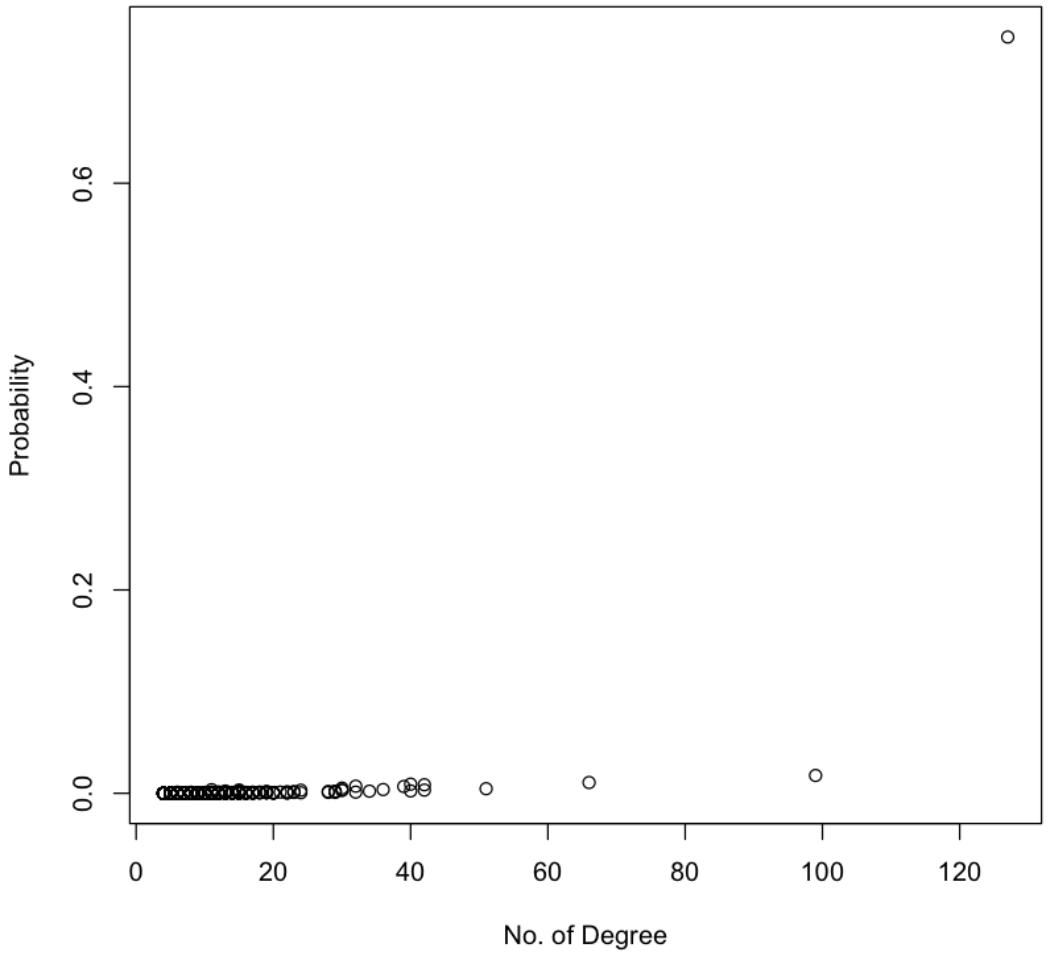


Figure 2.4.2.s: The probability of each node being visited by the walk against degree of each node.

2.4.b Find two nodes in the network with median PageRanks. Repeat part (a) if teleportations land only on those two nodes (with probabilities 1/2, 1/2). How are the PageRank values affected?

We find the two nodes in the graph, which have the median pagerank scores. And, we set probability of teleporting to either of these two nodes is 0.5, while we set zero the probability of teleporting to any other node in the graph. The results are shown in Figure 2.4.3 & 2.4.4. We can also see the pattern that the walk visits more possibly the old nodes in the graph. But, there exist some unexpected nodes which are relatively new but have high probability of being visited. This is because the destination of teleportation is restricted to only two nodes. Therefore, the nodes which are near these two nodes are more likely to be visited by the walk. Thus, in Figure 2.4.3, we observe that two nodes got boosted in the probability along with around 8 nodes having much higher probability as well. Then, the probability of each node being visited is less related to the degree of each node. The correlation drops to 0.673.

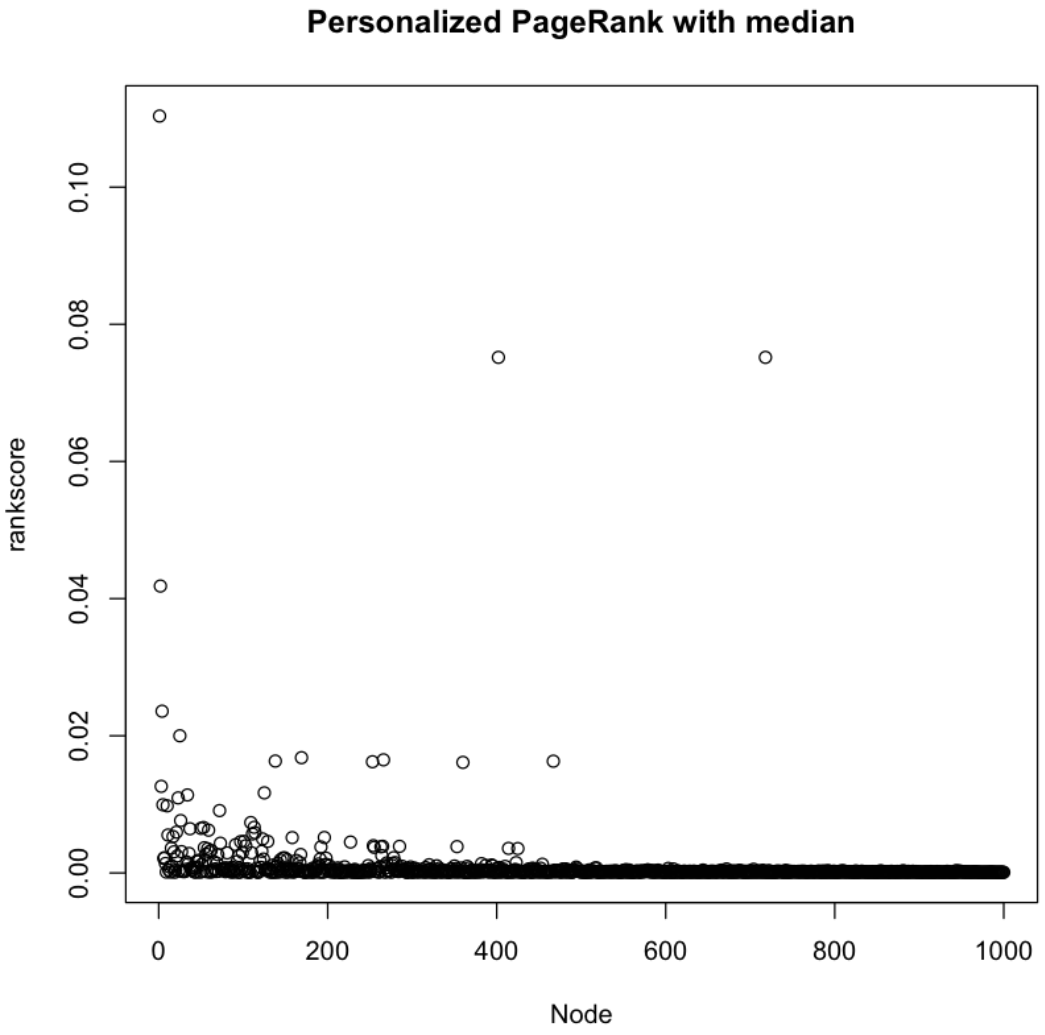


Figure 2.4.3: The probability of each node being visited by the walk against node ID with damping = 0.85.(median PageRank teleportation)

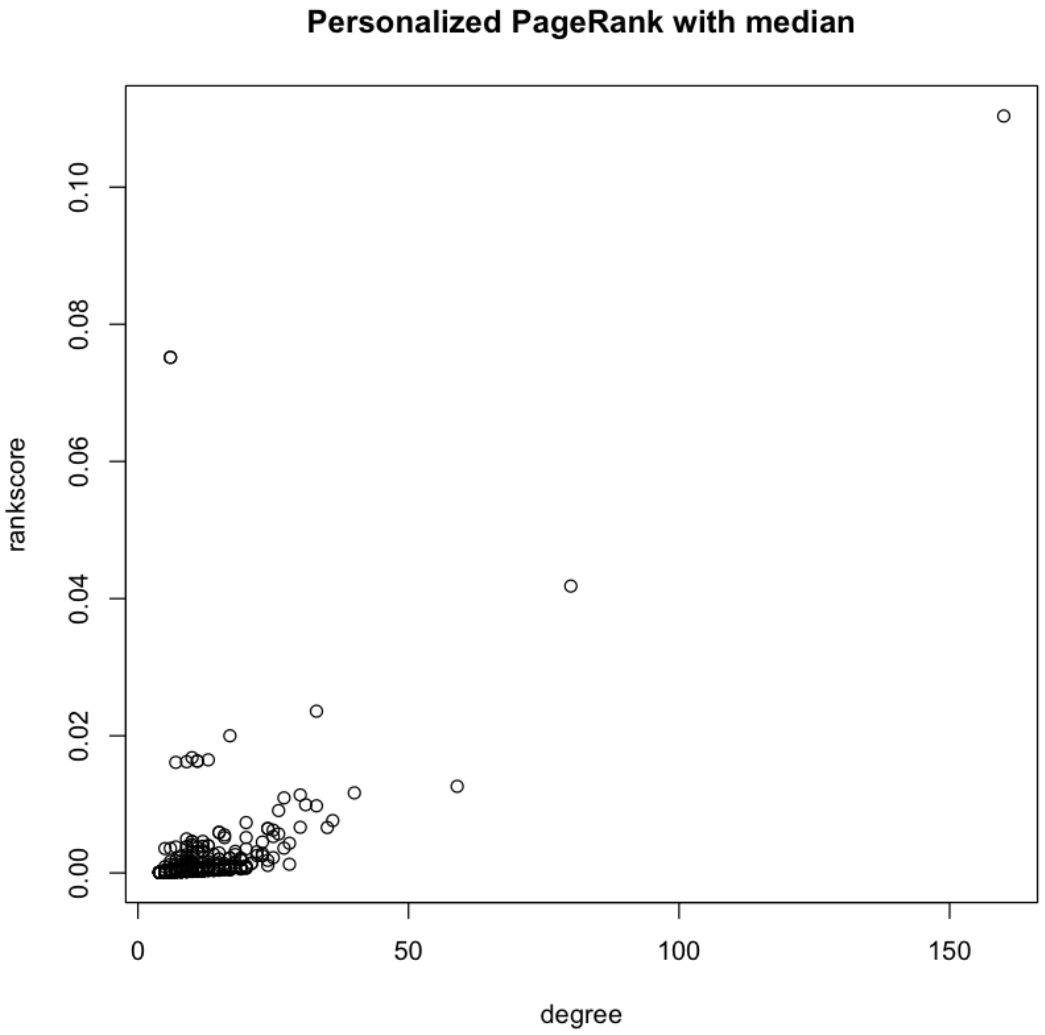


Figure 2.4.4: The probability of each node being visited by the walk against degree of each node with damping = 0.85. (median PageRank teleportation)

***Special Random Walk*:**

The results are shown in Figure 2.4.3.s & 2.4.4.s. Observably, the two nodes with median pagerank scores become more interesting, and gain higher probability of being visited. The correlation between the visiting probability and the degree is 0.552, which is surprisingly a little higher than that in 2.4.a, but the difference is small and acceptable. The reason behind might be that the teleportation by personalized pagerank score make the probability distribution too polarized, and hence the correlation is not observable.

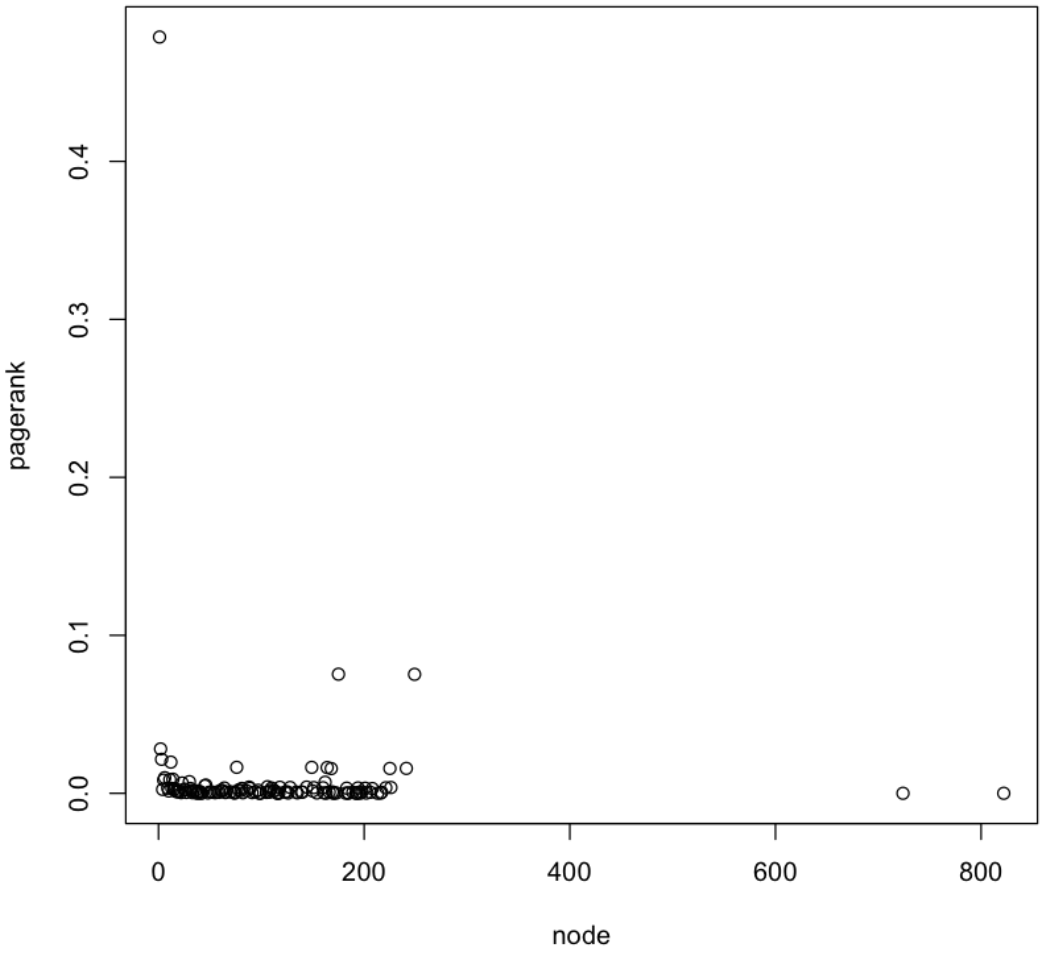


Figure 2.4.3.s: The probability of each node being visited by the walk against node ID.

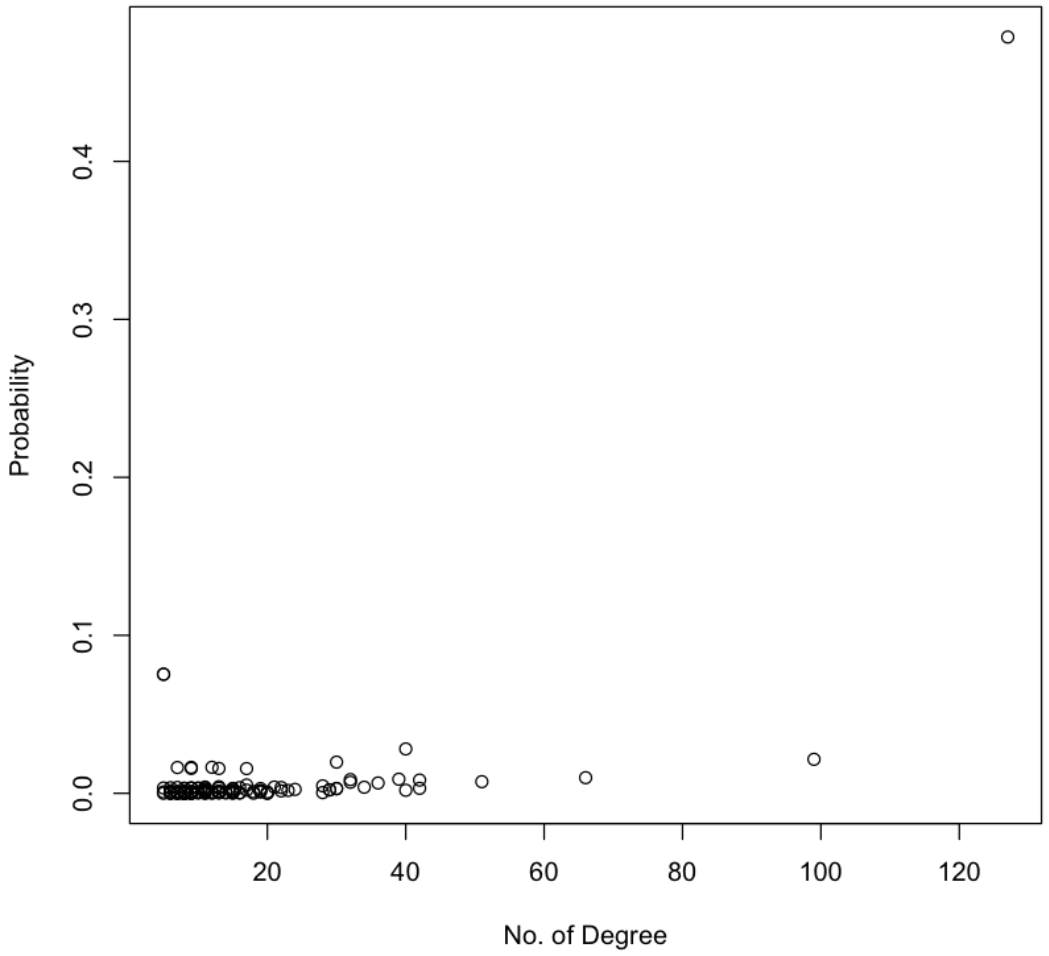


Figure 2.4.4.s: The probability of each node being visited by the walk against degree of each node.

2.4.c More or less, (c) 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 different assumption of normal PageRank, where we assume that people's interest in all nodes are the same. Can you take into account the effect of this self-reinforcement and adjust the PageRank equation?

The new pagerank score(PR') of a node should take into account its original pagerank score(PR). The modified PageRank equation is shown as follows. The probability of being teleported to is no longer 1/N for each node. Instead, it should be the original pagerank score of each node.

The original PageRank equation is:

$$PR(A) = \frac{1-d}{N} + d \sum_{incomming\ nodes} \frac{PR(node)}{L(node)}$$

The modified PageRank equation is:

$$PR'(A) = (1-d)PR(A) + d \sum_{incomming\ nodes} \frac{PR(node)}{L(node)}$$