# ECE 232E Project 1 Report

Ken Gu, 904836308

Yingbo (Max) Wang, 604-593-537

Ziying Feng, 404395859

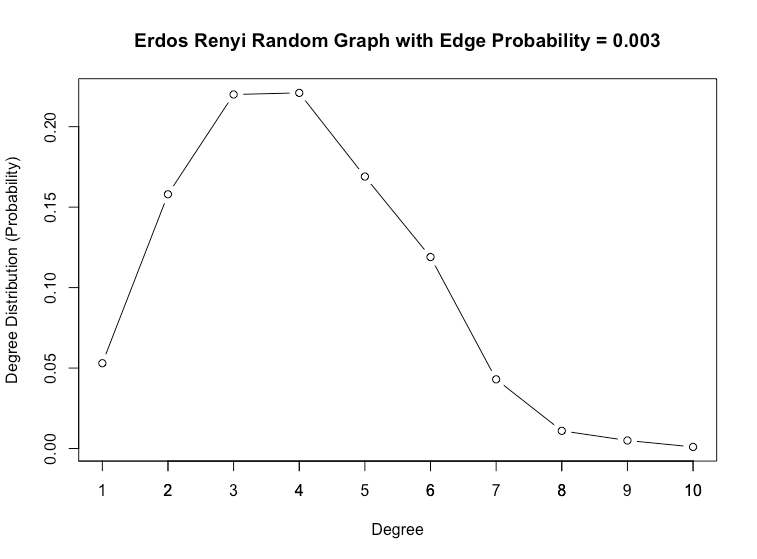
Siqi Huang,504490530

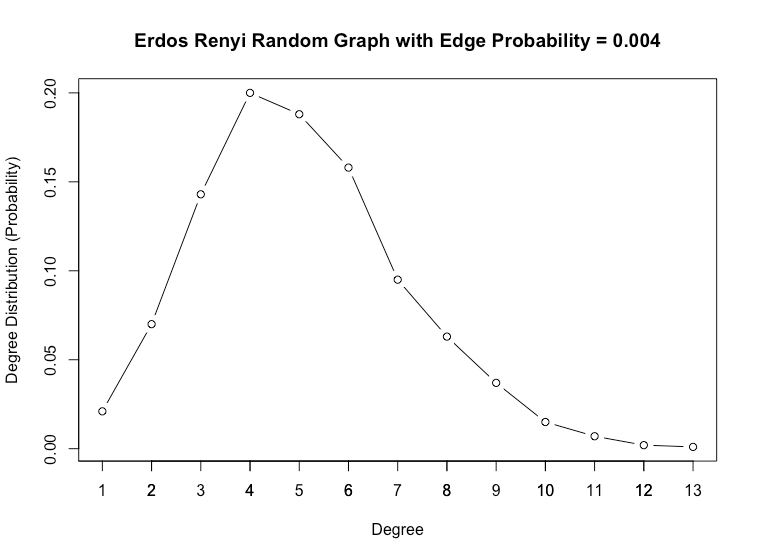
## 

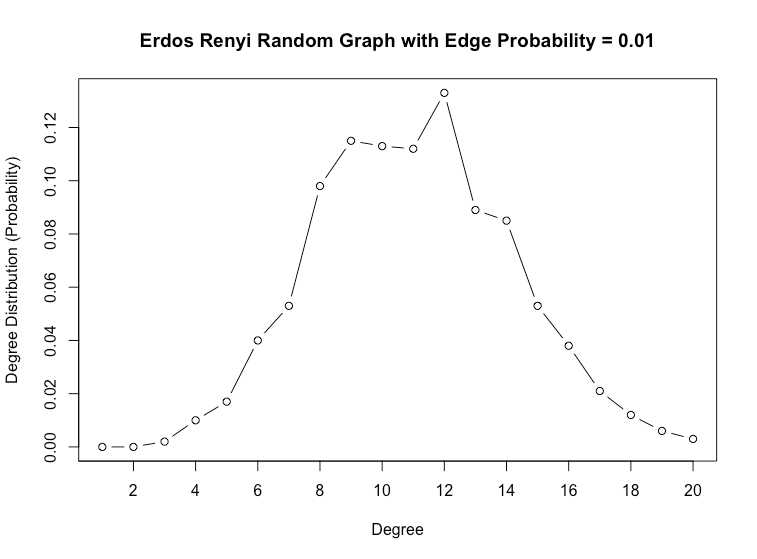
## Part 1

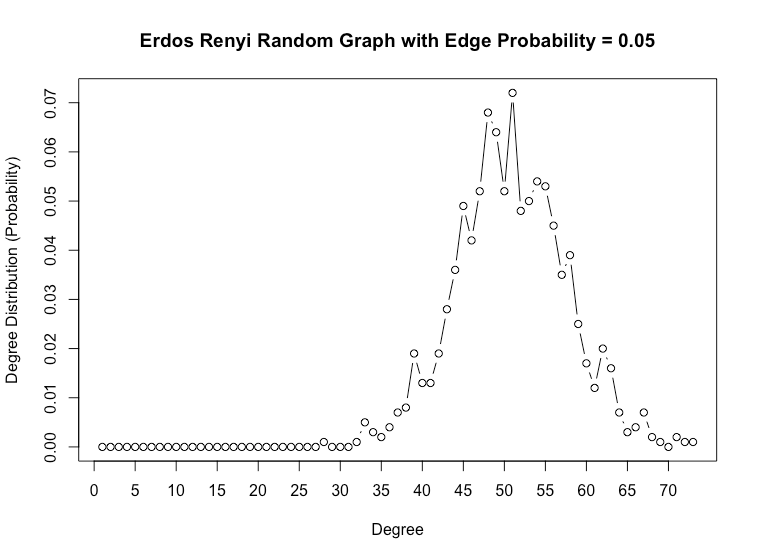
### Question 1

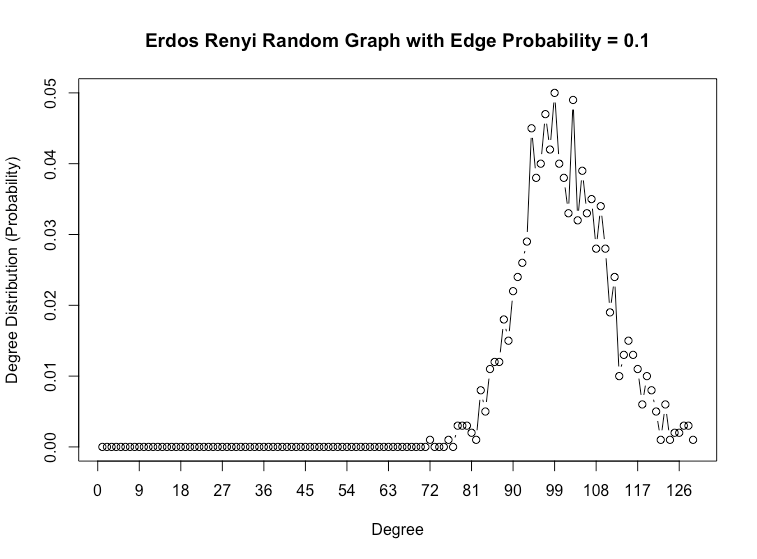
**(a)**











The degree distributions are binomial distributions and can be approximated by a Gaussian distribution when the number of nodes is large. By its definition, in a graph of *n* nodes, a node has degree *k* if it’s connected to *k* of the other *n - 1* nodes and disconnected to the remaining nodes. Thus, in the random network when every two nodes are connected with a probability of *p*, a node has degree *k* with probability:

This is a binomial distribution by definition.

Theoretically, the binomial distribution has mean/expected value equals and variance equals . The theoretical and measured values of different edge probabilities *p* are reported in the table below. We can see that the measured values correspond closely to the theoretical values, with around or less than 5% difference.

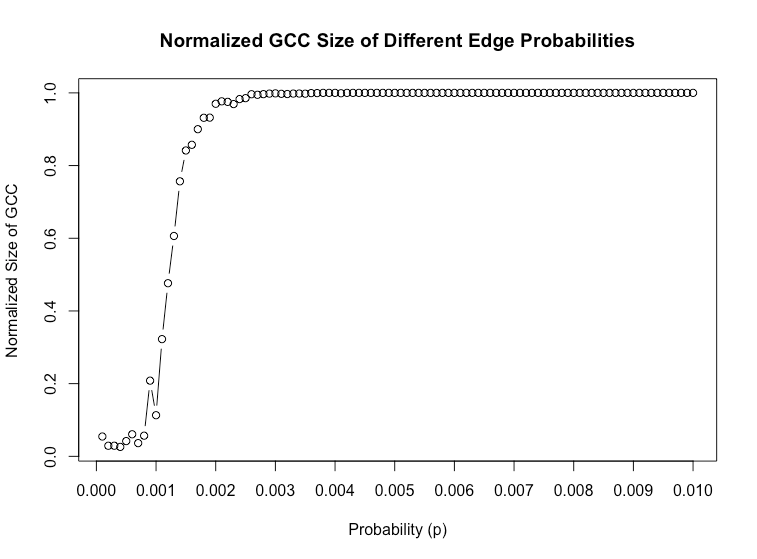
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Mean (Theoretical) | Mean (Measured) | Variance (Theoretical) | Variance (Measured) |
| *p* = 0.003 | 2.997 | 2.982 | 2.988 | 2.965 |
| *p* = 0.004 | 3.996 | 4.114 | 3.980 | 4.351 |
| *p* = 0.01 | 9.99 | 9.936 | 9.890 | 10.056 |
| *p* = 0.05 | 49.95 | 50.024 | 47.453 | 48.916 |
| *p* = 0.1 | 99.9 | 99.85 | 89.910 | 87.717 |

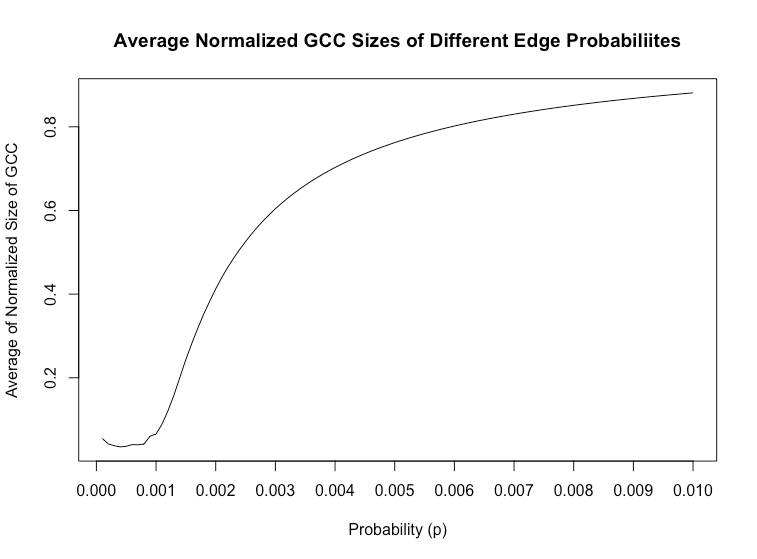
**(b)**

Not all random realizations of the ER network are connected. To numerically estimate the probability that a generated network is connected, we generate the random network 1000 times for each given edge probability and count the number of times the network is connected. For each probability *p*, we show the estimated probability of connectivity and the diameter of the graph’s Giant Connected Component if it’s not connected.

|  |  |  |
| --- | --- | --- |
|  | Probability that the Graph is Connected | Diameter of GCC |
| *p* = 0.003 | 0 | 15 |
| *p* = 0.004 | 0 | 10 |
| *p* = 0.01 | 0.957 | 6 |
| *p* = 0.05 | 1 | - |
| *p* = 0.1 | 1 | - |

**(c)**

****

****

In the above two plots, we sweep values of edge probability from *p* = 0 to *p* = = 0.010 with step 0.0001. We choose this value because the generated random graph is almost surely connected according to the result of a previous discussion - the probability that the graph is connected is more than 96%.

**i**.

We define a Giant Connected Component emerges on an Erdos Renyi random graph with *n* nodes when all other clusters beside the GCC have a size smaller than *log(n)*. According to this definition, we check the size of each cluster for every random graph generated at each *p* and track down *p* if a GCC emerges in this graph. This experiment is repeated many times by repeatedly generating these random graphs. Empirically, GCC has a high probability to emerge for *p* 0.0017.

Theoretically, consider the number of edges in the graph’s largest connected component over the size of the graph (): this value stays close to 0 before (*n* is the number of nodes in the graph), and then starts increasing logarithmically with decreasing speed. This increasing continues until when is thought to converge to 1. Therefore, theoretically a GCC emerges when . The probability we got empirically closely corresponds to this theoretical value.

**Ii.**

For each , a random graph of 1000 nodes is generated 50 times, where each time we find its Giant Connected Component and its number of nodes. We track the number of times this GCC takes up more than 99% of the nodes. The first *p* where this condition satisfies for all 50 trials is our estimated probability. Empirically we found after repeating this experiment. Then, we conduct the same experiment but count the number of times the GCC contains more than 99.5% of the nodes instead of 99%. The resulted *p* should be higher than the previous one as the required size of the GCC grows. As expected, we found *p* = 0.0063 in this case.

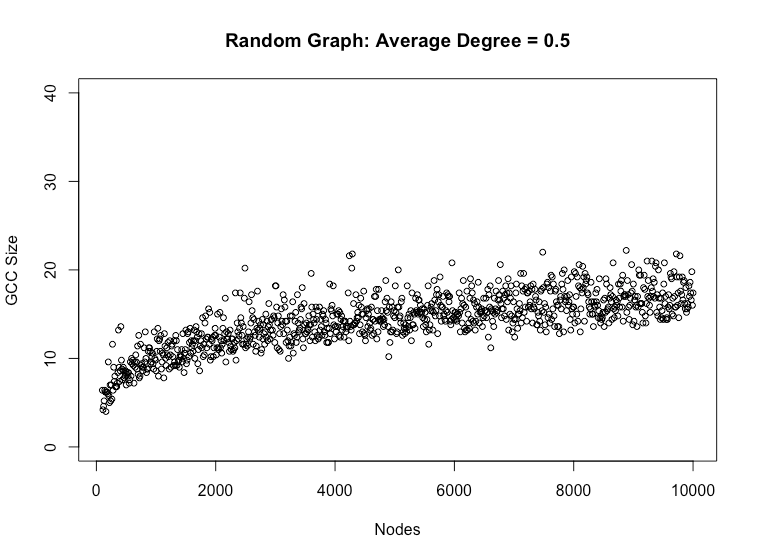
Theoretically, when the GCC starts to take up almost the entire graph, begins to converge to 1, where in our case. In this case, the graph is almost sure to be connected. Therefore, the theoretical and the empirical values correspond closely, and we can expect even a better match when we further increase the “node percentage” in the experiment.

**(d)**

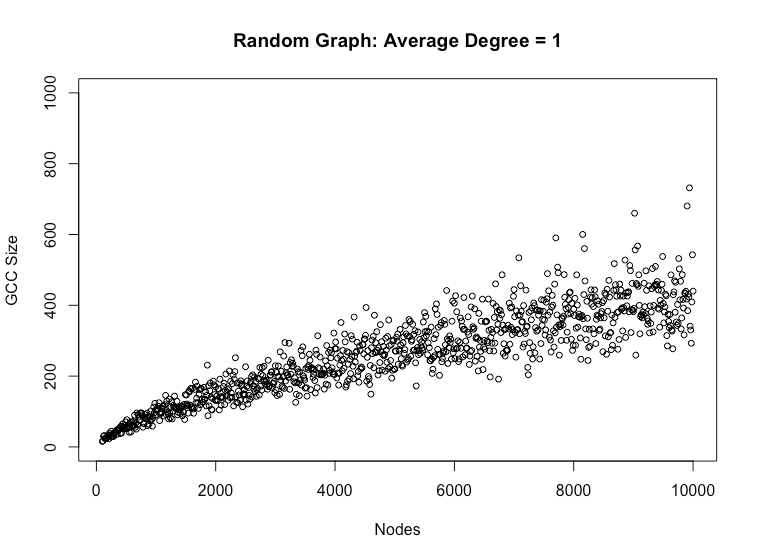
**I.**

In this experiment for *c* = 0.5, we increase the number of nodes from 100 to 10000 with a step size of 10, so 990 random graphs are generated. Each random graph has edge probability *.* To estimate the expected size of each graph’s Giant Connected Component, we generate the graph 10 times in each step, find its GCC’s size and calculate the arithmetic mean.

We see that overall, the expected GCC size grows linearly when the number of nodes increases, with a slope much smaller than 1. As the average degree stays constant, this trend is reasonable because its largest connected component will grow proportionally when the size of the graph increases.

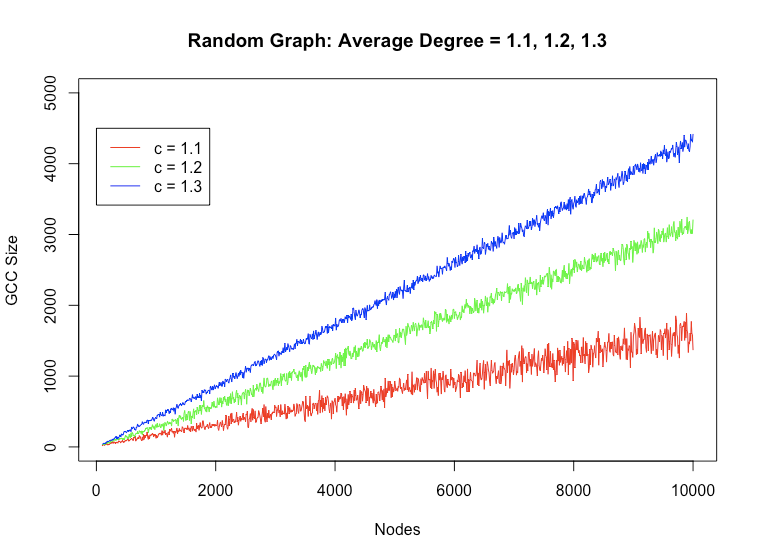
****

**Ii.**

****

The trend is similar as in the case when average degree equals 0.5. The expected size of GCC still increases proportionally with the number of nodes but at a much faster rate. This result is reasonable since the average degree is doubled from the last experiment that leads to a higher edge probability *p* in each graph. A higher *p* means more edges exist on each node to other nodes, so the connected components of each graph tend to contain more nodes and edges.

**Iii.**

****

**iv.**

For each average degree *c*, the expected GCC size increaes linearly with the number of nodes *n*. The rate of increase (slope of the linear function) is higher when the graph’s average degree is larger, with reasons described previously.

Theoretically:

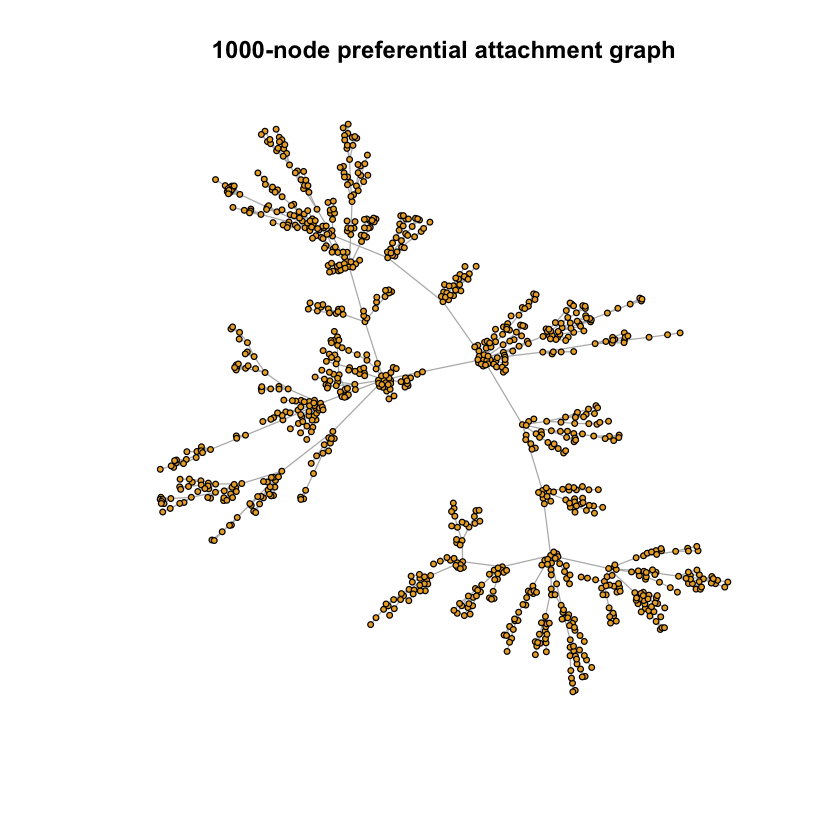
When *c* < 1, the graph almost surely has no connected component of size larger than .

When *c* = 1, the graph almost surely has the largest connected component of size in the order of .

When *c* > 1, the graph almost surely has a GCC and no other component contains more than vertices.

### Question 2

**(a)**



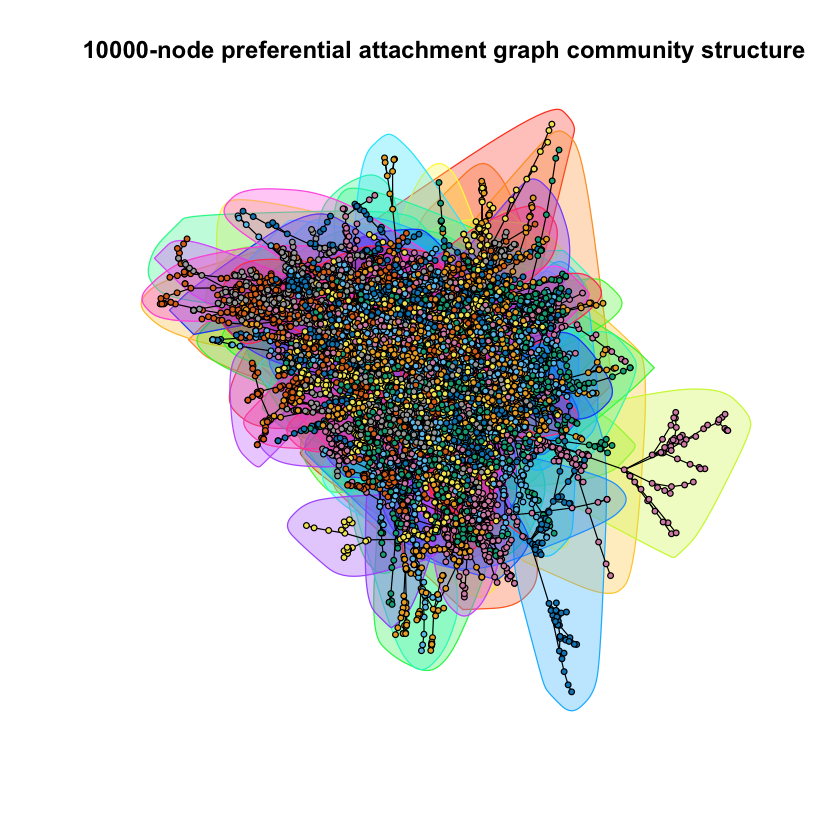
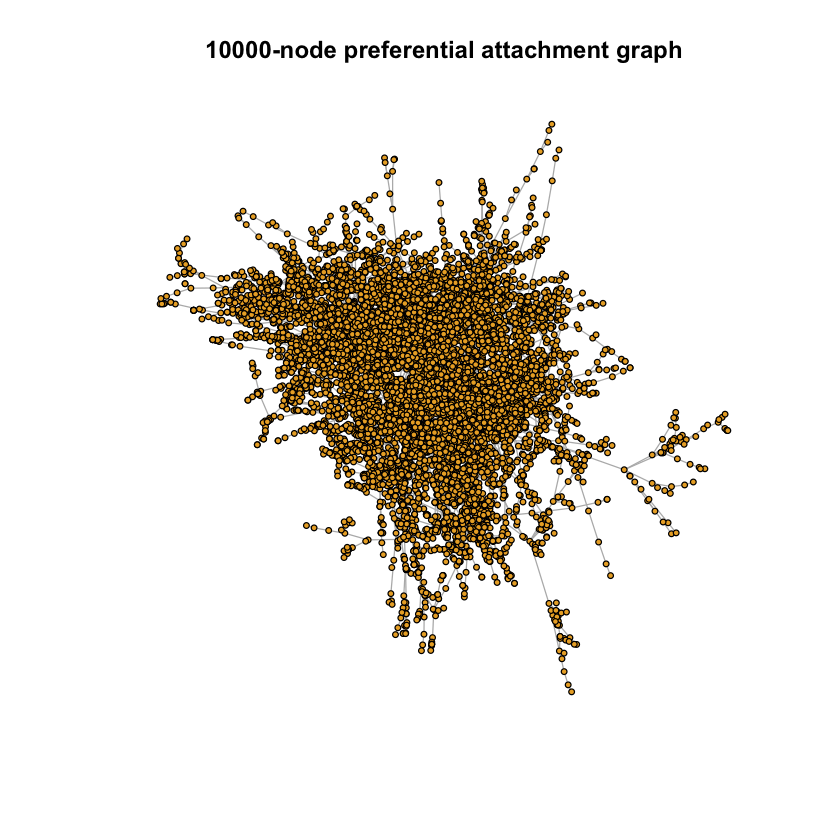
The network created by the preferential attachment model is always connected. One preferential attachment creation mechanism is Barabasi Albert model (BA model), like the price model, BA model follows the power law. During the creation process, the graph is gradually grows by attaching new nodes to existing nodes, and the new nodes are tend to attach to the existing nodes with higher degree. This is also called Matthew effect. That’s why the graph created by the BA model is always connected.

**(b)**

The modularity of the graph with 1000 nodes is 0.93. Modularity measures the density of the connections between the nodes within communities. The criterion of the modularity is , and the use the fast greedy algorithm to pick the merge to optimize the Q value.



**(c)**



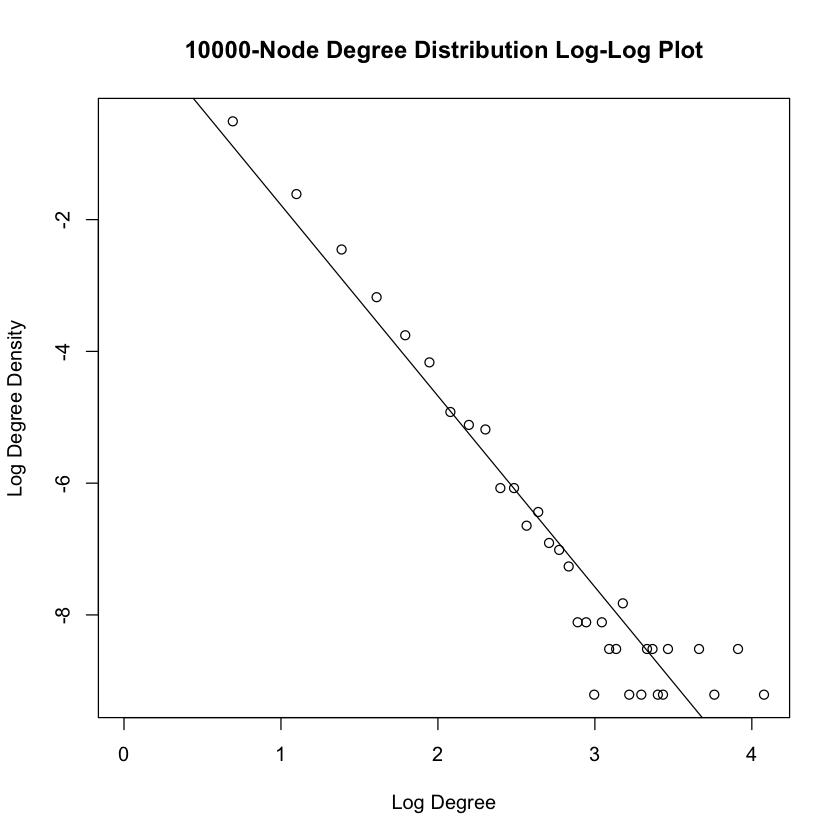
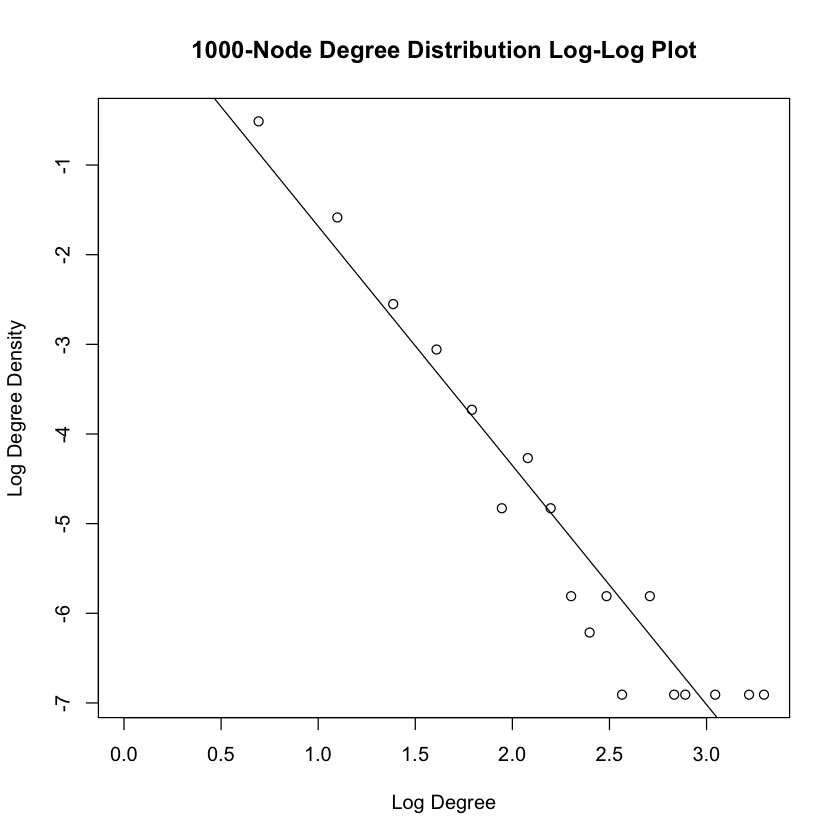
The modularity of the graph with 10000 nodes is 0.97.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Is connected | Modularity | Degree mean |
| 1000-node preferential attachment model | True | 0.93 | 1.998 |
| 10000-node preferential attachment model | True | 0.97 | 1.9998 |

The table compares the 1000-node graph and 10000-node graph created by the preferential attachment model. Both graphs are connected. The graph with more nodes have higher degree mean and higher modularity.

High modularity means dense connections between the nodes within communities but sparse connections between nodes in different communities. The graph with more nodes has higher degree mean because with the growth of the graph, more nodes are attaching to the existing graph. And since it follows the preferential attachment mechanism, the hub of the the graph is getting more obvious, that is the graph is becoming denser, which leads to higher modularity.

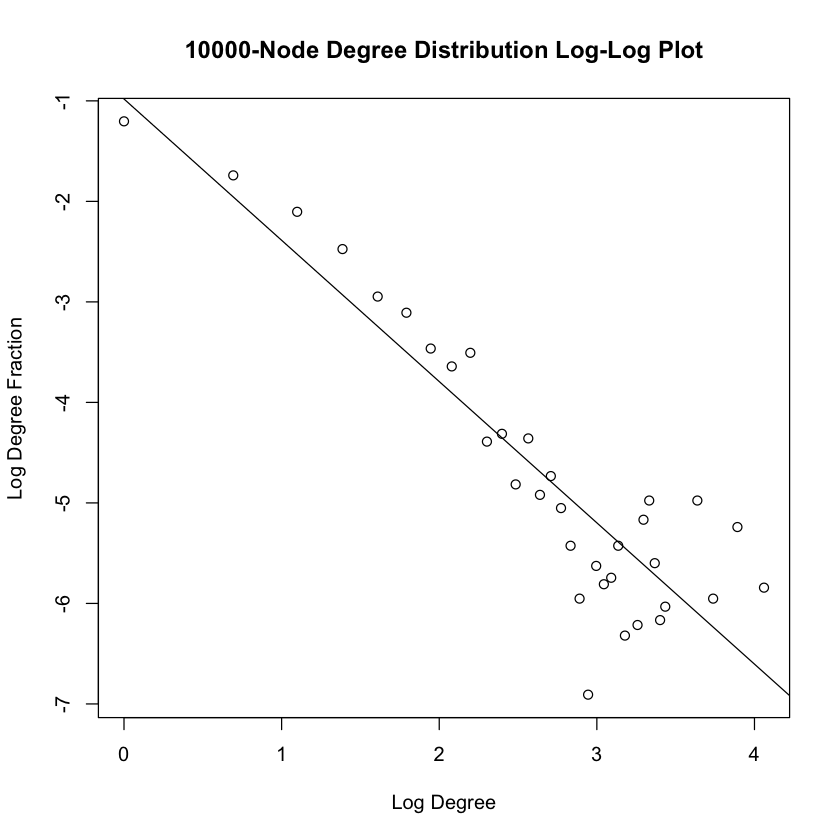
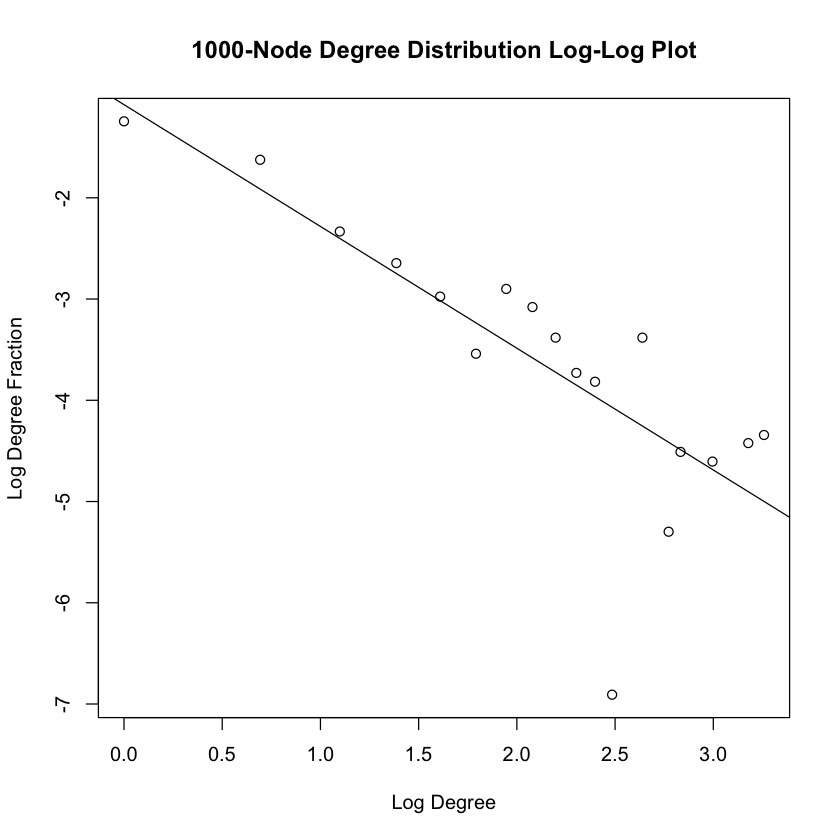
**(d)**



|  |  |  |
| --- | --- | --- |
|  | Slope | Intercept |
| 1000-node preferential attachment model | -2.6676 | 0.9835 |
| 10000-node preferential attachment model | -2.899 | 1.121 |

, the slope of the Log-Log plot is , and the intercept is . The slope of both the graphs are about -3, meaning is about 3, which corresponds to the fat-tail distribution. For , is , and is , that is lots of nodes with high bounded average degree but very high variance.

**(e)**



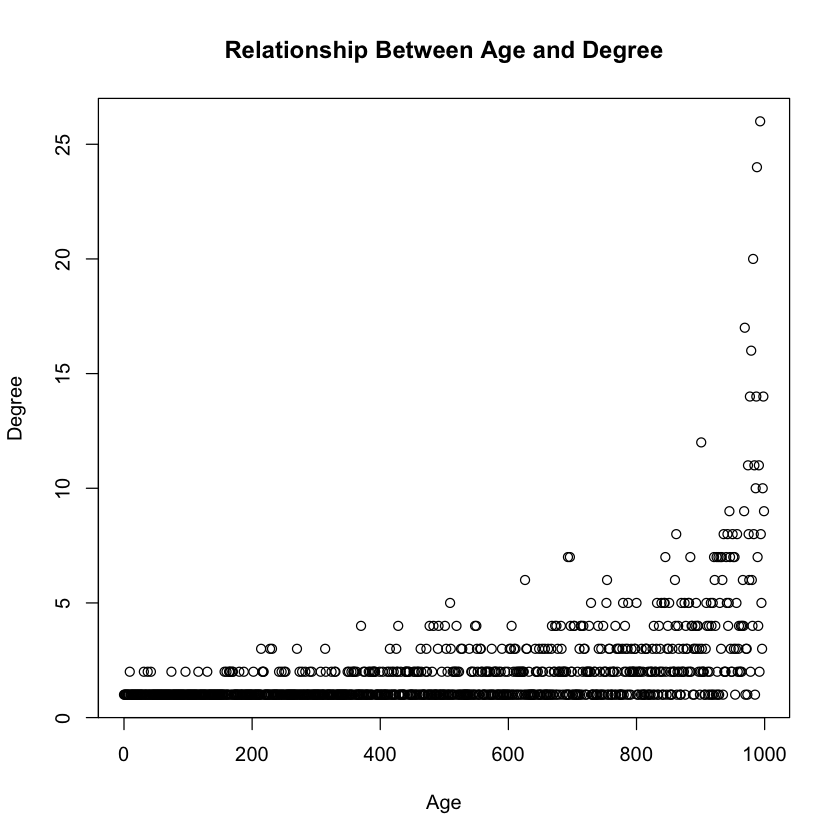
|  |  |  |
| --- | --- | --- |
|  | Slope | Intercept |
| 1000-node preferential attachment model | -1.203 | -1.079 |
| 10000-node preferential attachment model | -1.4057 | -0.9812 |

To calculate the degree distribution of j\_node which is picked after randomly picking i\_node.

1. Initiate a vector to store the appearance frequency of each degree. For a 1000-node graph, the highest degree is max\_deg, so the length of the vector is max\_deg. Each element of the vector represents the appearance frequency when the degree = the index of the vector.
2. In the loop, randomly pick i\_node, from the i\_node neighbors randomly pick j\_node, calculate the degree of j\_node, and store the appearance in the j\_deg\_freq.
3. Iterate for 1000 times to find the degree distribution j\_node
4. Calculate the fraction for each degree, convert to the log scale and do the linear regression

The slope is roughly -1.5, that is in the power law, which means is , and is , that is lots of nodes with high average degree and very high variance. The second node j\_node is always picked at the neighbor of the i\_node, since the hubs are having high degree, the possibility of j\_node is a hub is very high, leading to the high average degree when compared to the degree distribution of the BA model.

**(f)**



The nodes with higher degree have higher degree. The older are richer with more nodes accumulate as time goes on, so the older nodes have higher degree and more edges to attract the new nodes to attach.

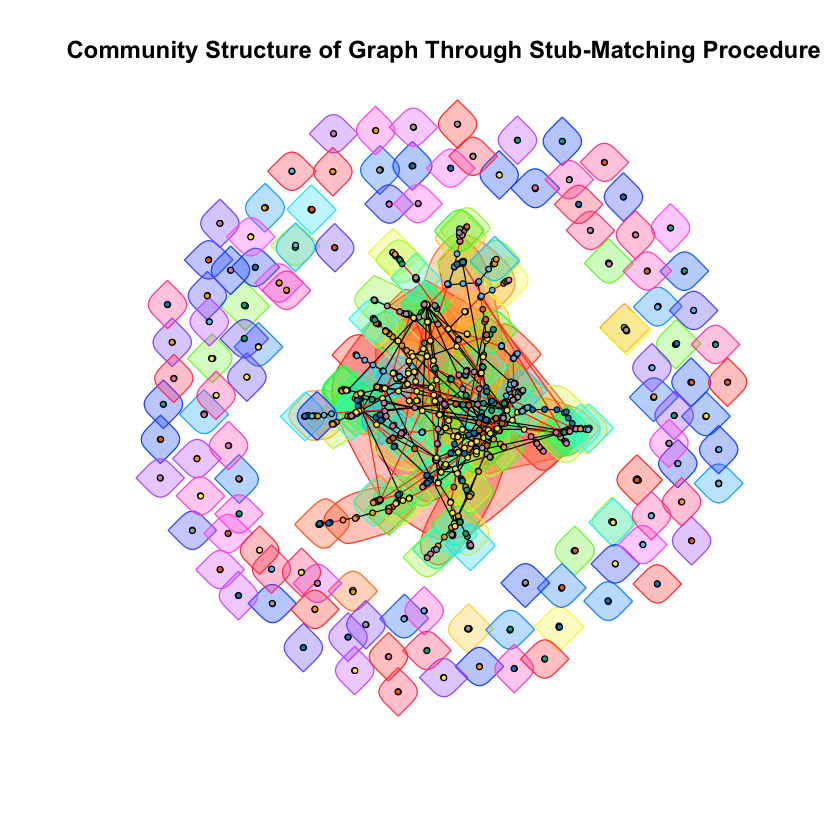
**(g)**

Modularity with Different m

|  |  |  |  |
| --- | --- | --- | --- |
|  | m = 1 | m = 2 | m = 5 |
| 1000-node preferential attachment model | 0.93 | 0.52 | 0.28 |
| 10000-node preferential attachment model | 0.98 | 0.53 | 0.27 |

The higher the m, the lower the modularity. The more the nodes on a graph, the higher the modularity. With higher m, each time when a new node added to a network, the new node will have more edges connecting to other communities, leading to the sparse of the community, thus lower modularity.

**(h)**

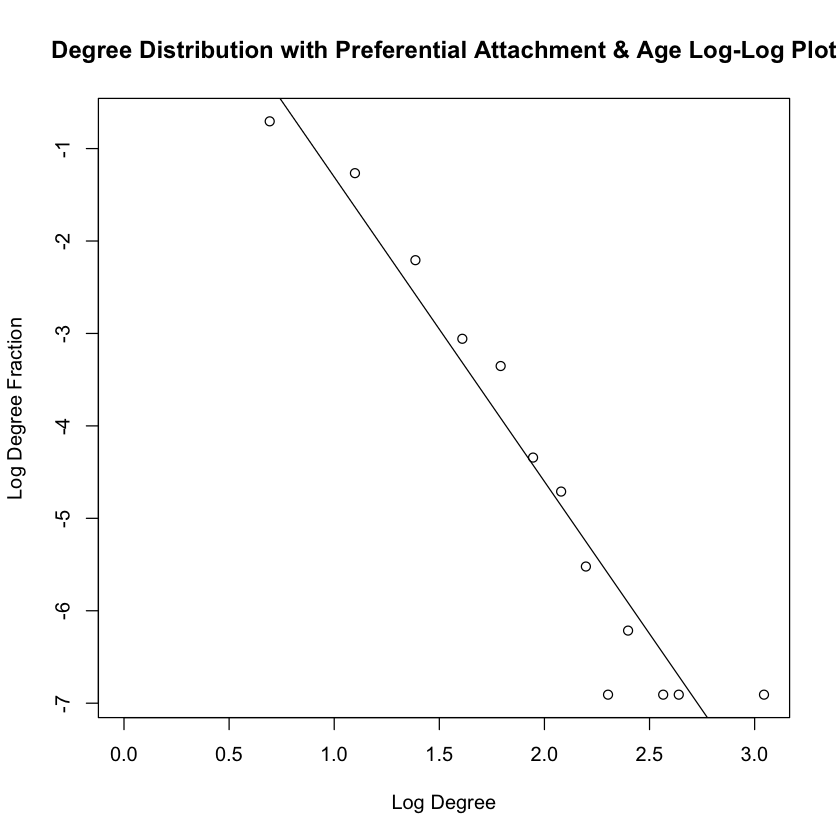


|  |  |  |  |
| --- | --- | --- | --- |
|  | Is connected | Modularity | Communities |
| Stub-Matching | False | 0.75 | 225 |
| Barabasi-Albert | True | 0.93 | 32 |

In the stub-matching procedure, the out-stubs of the edges are connected according to the degree sequence, but due to the self-loop, it is possible that the there are some nodes connect to themselves, that is single nodes, which makes the graph not necessarily connected, lower modularity and more communities.

### Question 3

**(a)**



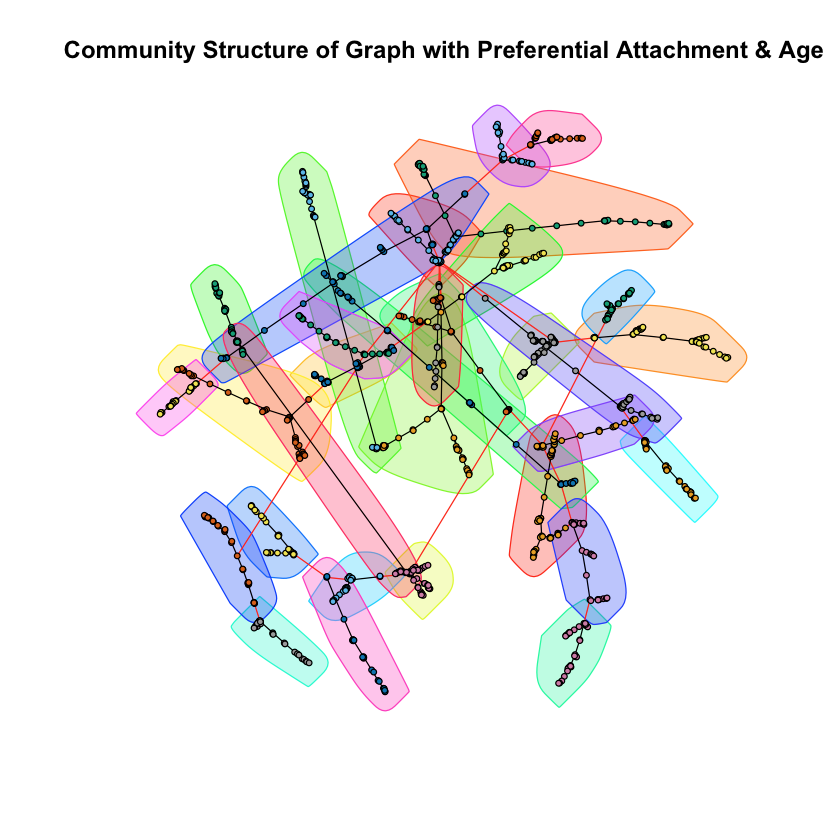
The intercept is 1.986, and the slope is -3.294.

The power law exponent is -3.294. is the aging power law exponent, like For the probability of connecting to an existing node, if the existing node has higher degree, the probability is higher, so the is the in-degree exponent, here meaning the probability is proportional the in-degree.

The slope is < -3, that is in the power law, which means is , and is , that is both the average degree and the variance are bounded. The variance is less than that of the BA model.

When penalizing the age, the percentage is multiplied by a term related to the age with negative exponent so as to lower the importance of age when connecting a new node to the existing node. And this makes the new nodes less likely to attach to the nodes with higher degree, making lower variance of the degree.

**(b)**



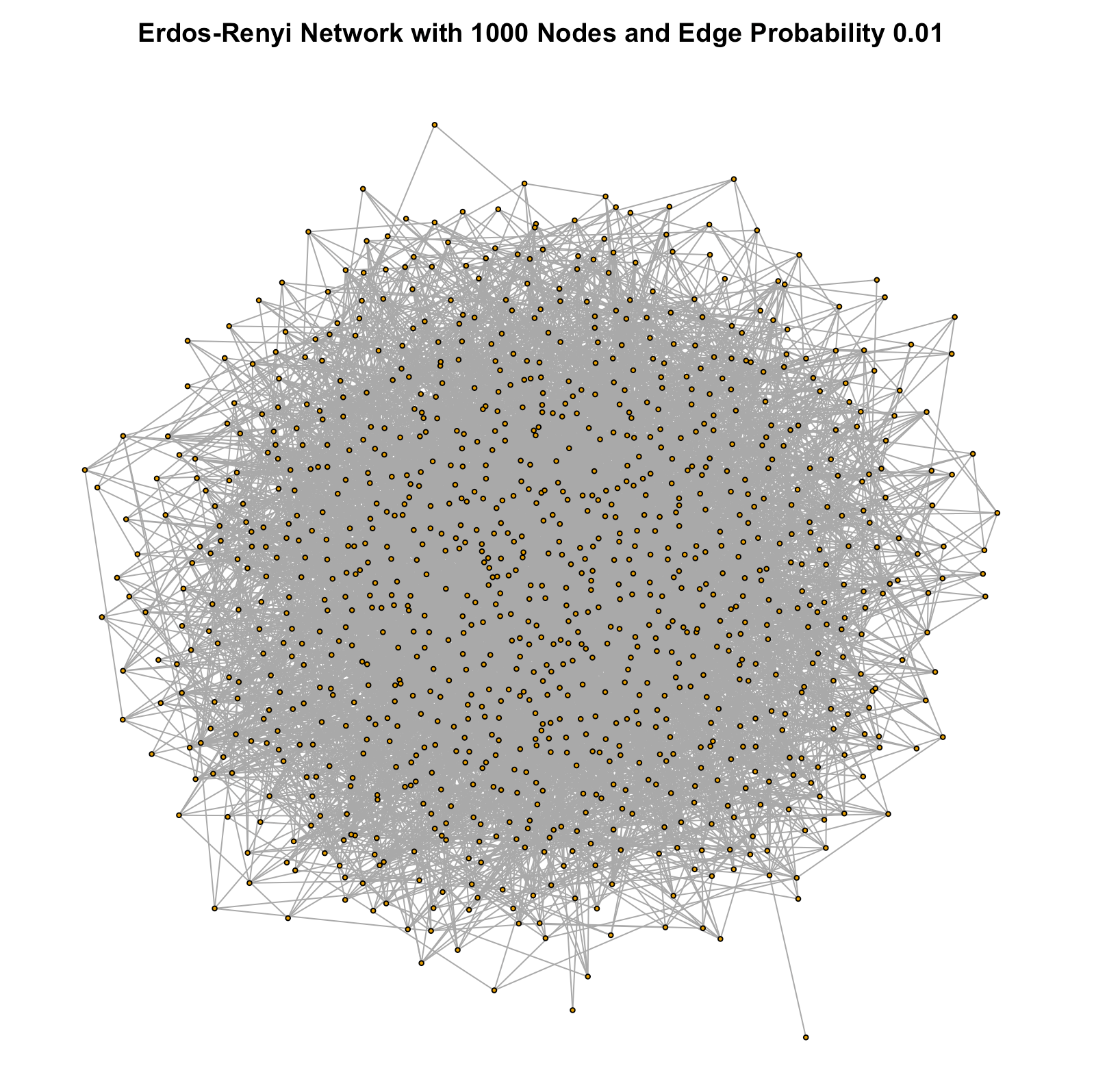
The modularity of the graph created with preferential attachment and aging is 0.75.

When penalizing the age, the new nodes less likely to attach to the nodes with higher degree, making lower variance of the degree, so the graph is more sparse, more uniform, leading to a lower modularity.

## Part 2

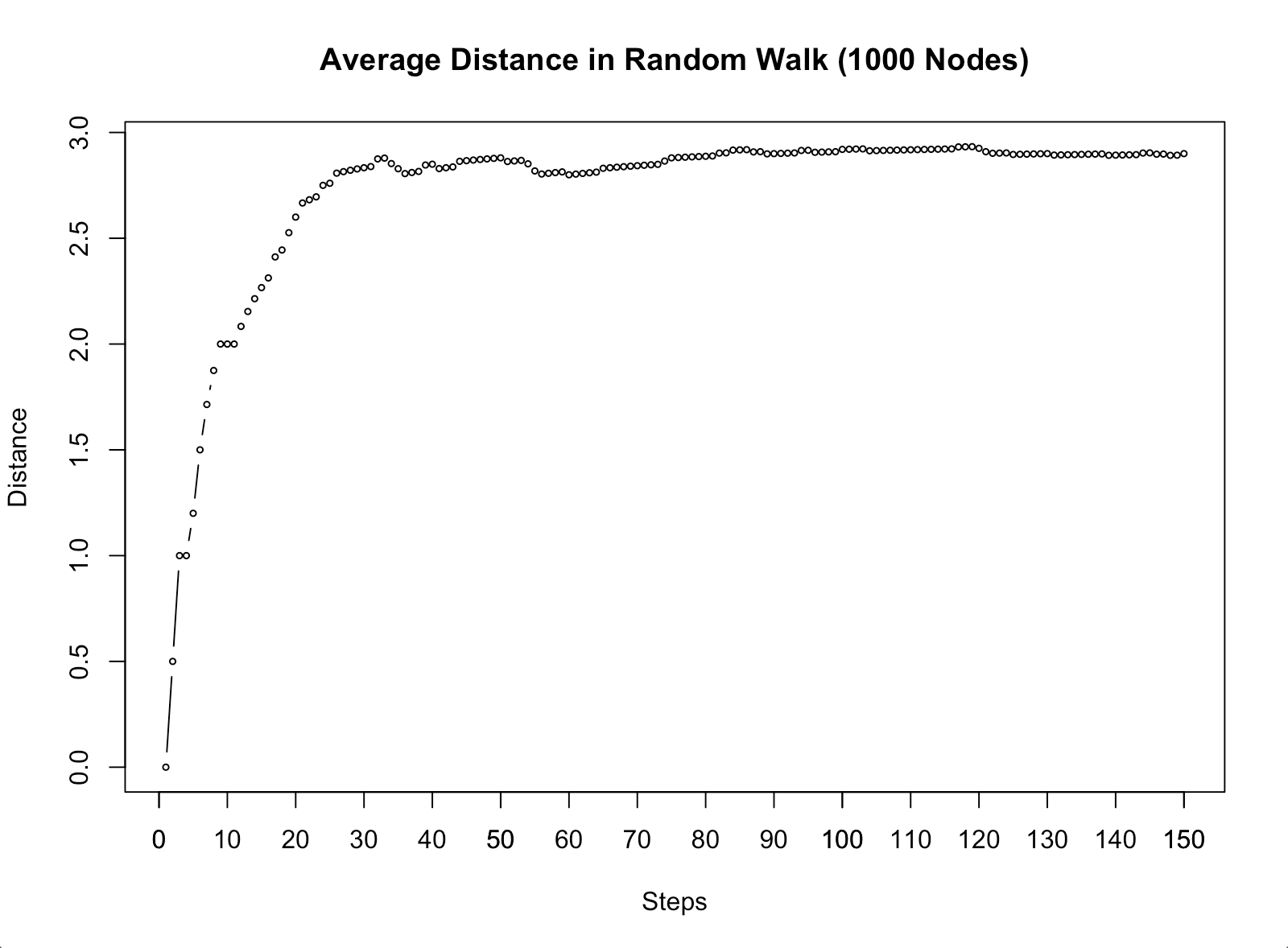
### Question 1

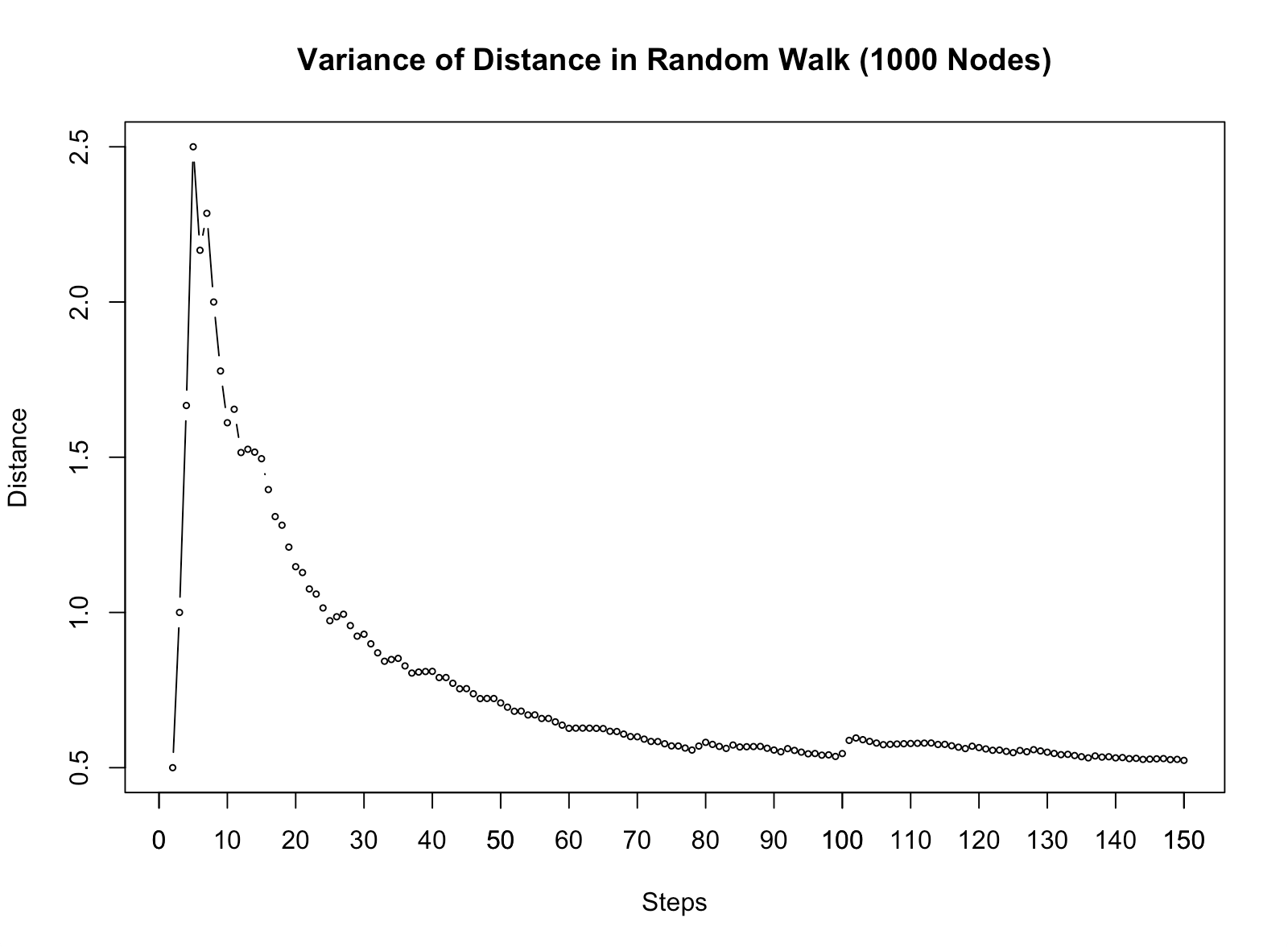
**(a)**



The network is created by the *erdos.renyi.game()* function inside *igraph* framework as shown in the sample code. A plot of the generated graph is shown above for illustration.

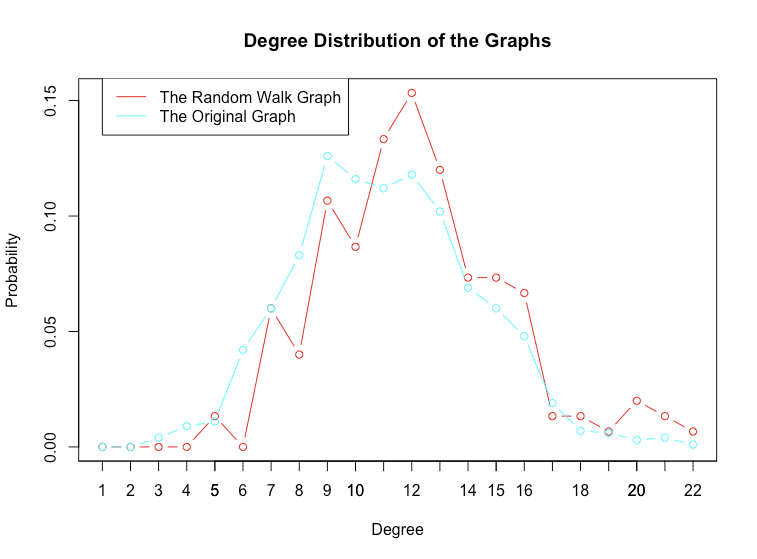
**(b)**





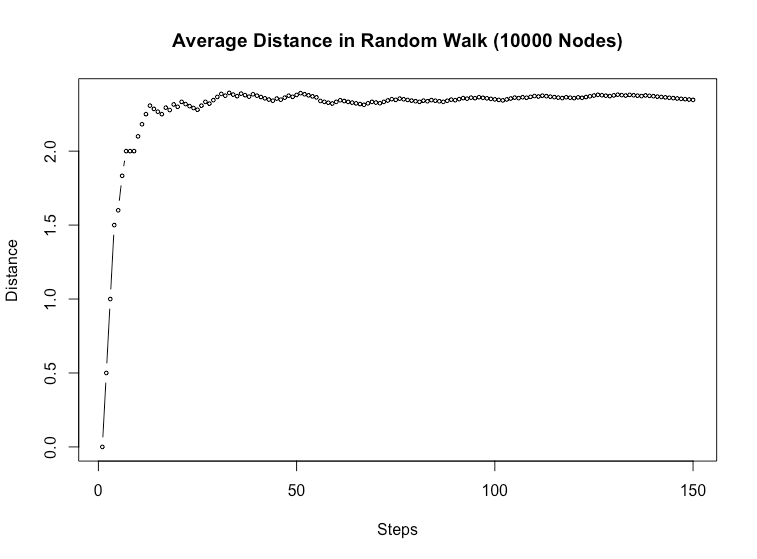
The code performs a random walk of 150 steps starting from randomly selected nodes in the 1000-node random graph. At each step *t*, the shortest distance between the current node and the starting node is measured and tracked, denoted as . We randomly select 100 starting nodes and the random walk is conducted for each node. We plot the arithmetic mean and variance of the distance from the 100 random walks for each step. This particular number of total steps is chosen because it’s enough for the distance to reach a steady state. As seen in the graph, both the average and the variance of distance converge to a fixed value as the number of steps goes higher. The average distance becomes relatively stable after about 30 steps and eventually converges to around 2.90 after 150 steps. The distance’s variance decreases to a stable value after about 80 steps and converges to around 0.52 after 150 steps.

**(c)**

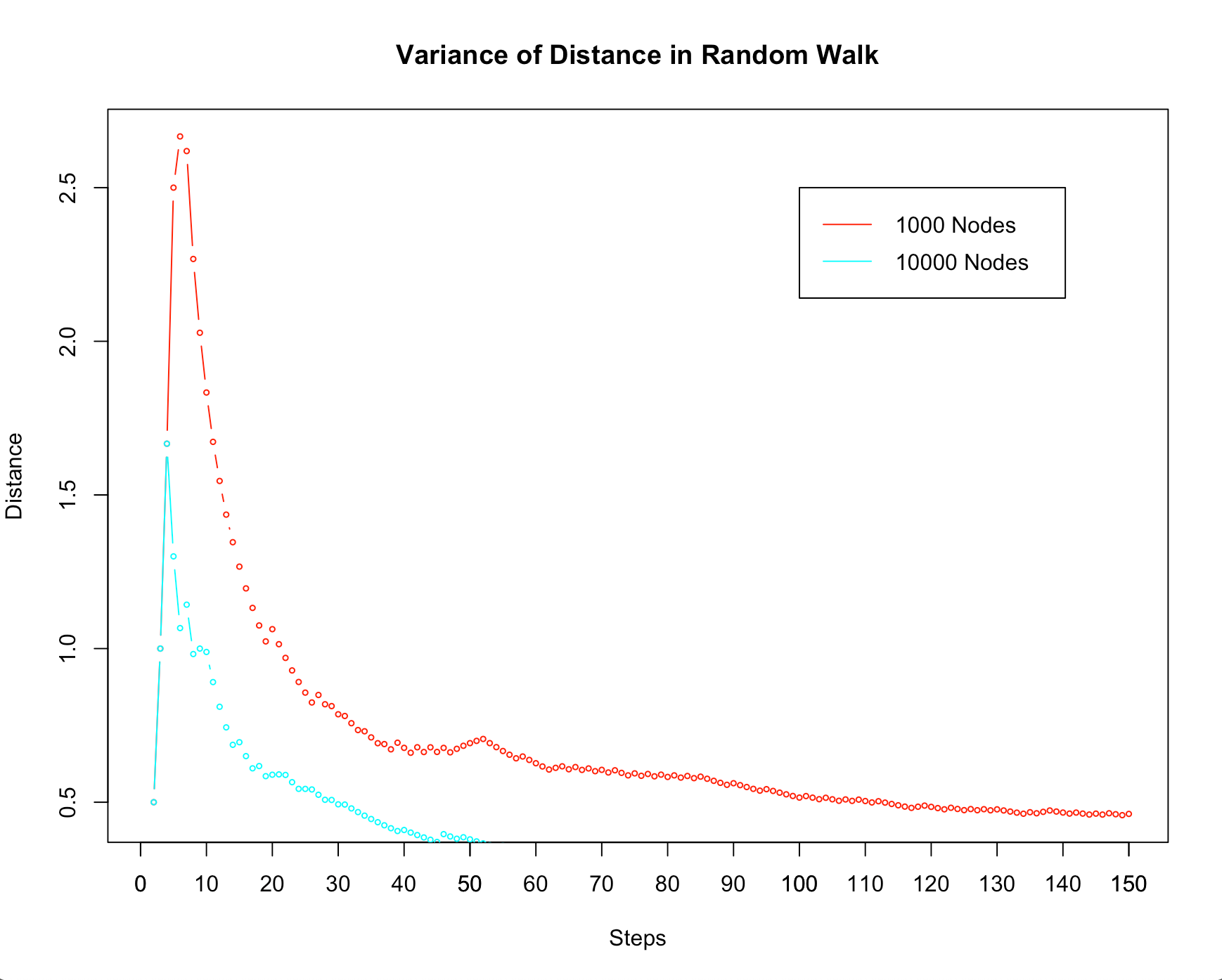
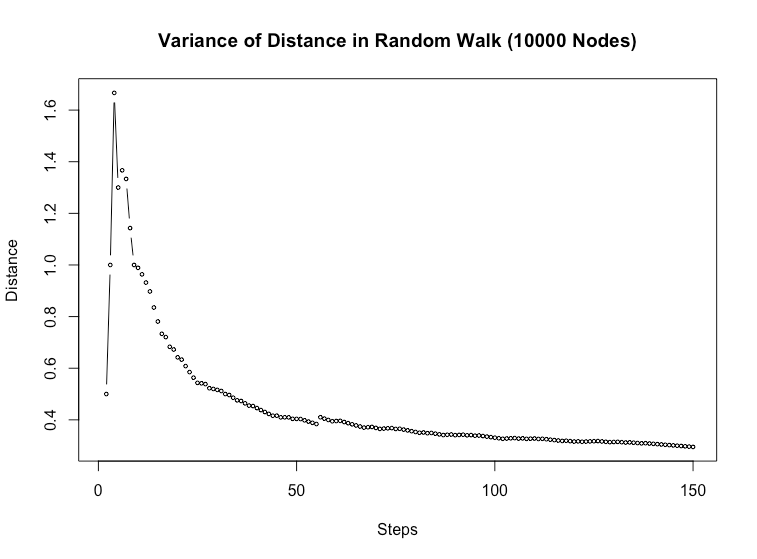


This plot shows the degree distribution of the two graphs overlayed together: the original random graph of 1000 nodes, and the resulting 150-node graph from the random walk process described above. We see that the degree distribution of the random walk graph and that of the original random graph is closely related: both the shape of the degree distribution and the probability value of each degree look very similar between these two graphs. The degree distribution of the random walk graph appears to be that of the original graph slightly shifted to the right: it has a slightly fewer number of nodes with a smaller degree, and more number of nodes with a higher degree. This result is reasonable because the random walk process behaves like a “selective sampling” of nodes in the original graph: nodes with more edges are more likely to be reached by the walker in the process and added to the new graph, so the probability of getting high-degree nodes becomes higher.

**(d)**







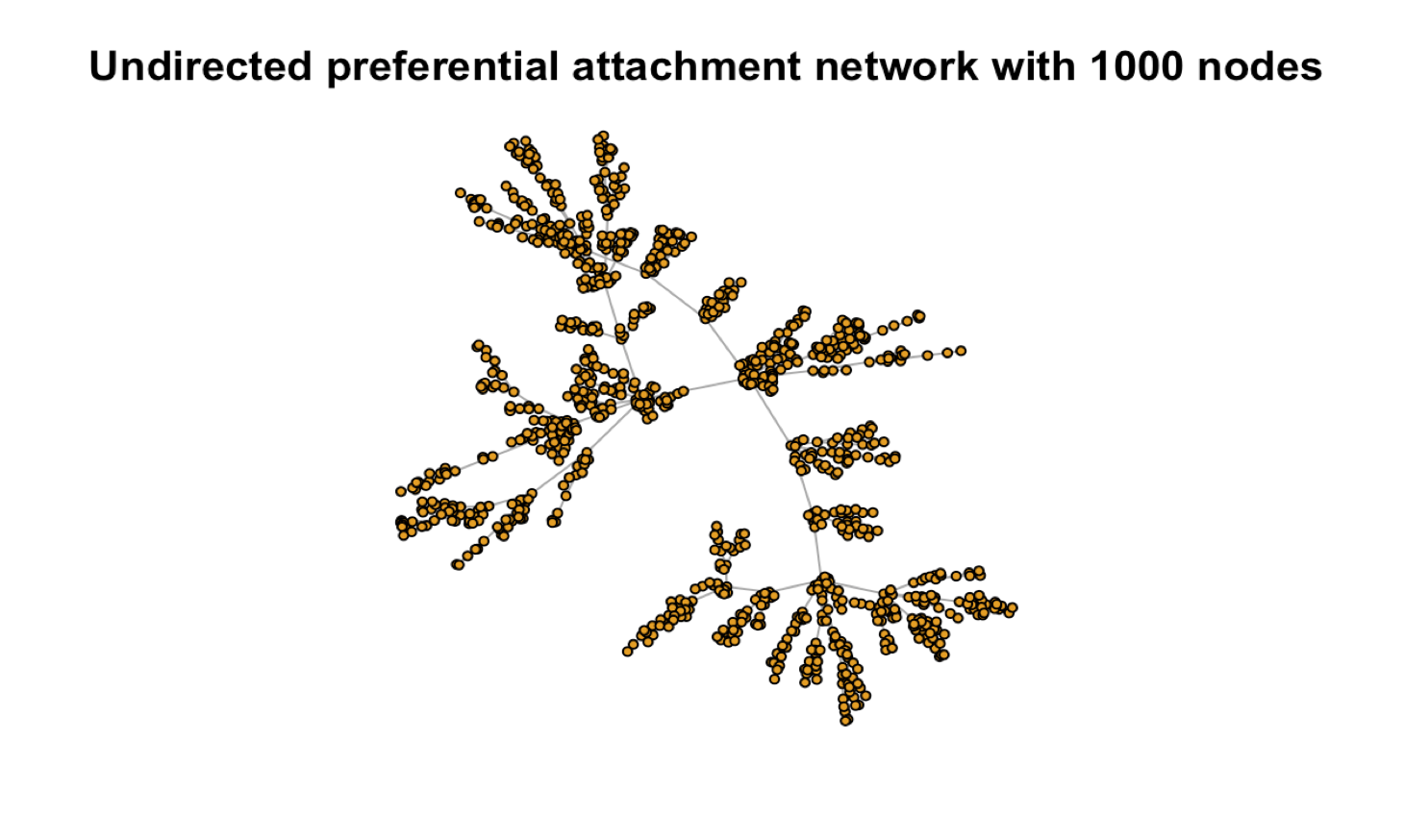
The plots are based on the resulting graph from 150-step random walks on an Erdos Renyi graph of 10000 nodes. Similar to previous experiments, the random walk is performed on 100 randomly selected starting nodes respectively. The plots show the arithmetic average and variance of the distances between the initial starting node and the node reached at step *i*. The line plots are also juxtaposed with those from an Erdos Renyi graph of 1000 nodes for comparison.

First, we can see that both the average and the variance of the distance converge faster to the steady state on the random graph with a larger number of nodes (10000 nodes, in this case). In the plot of average distance, the line’s slope decreases to about zero after around 15 steps in the random graph of 10000 nodes, while the same happens after about 30 steps in the random graph of 1000 nodes. Similarly, in the plot of the distance’s variance, it almost reaches a stable value after around 60 steps in the graph of 10000 nodes, while it continues decreasing with a noticeable slope in the other graph at that time. Also, both the converged average distance and variance of the distance between the initial starting node and the visited node are shorter in the random graph of 10000 nodes than in the graph of 1000 nodes.

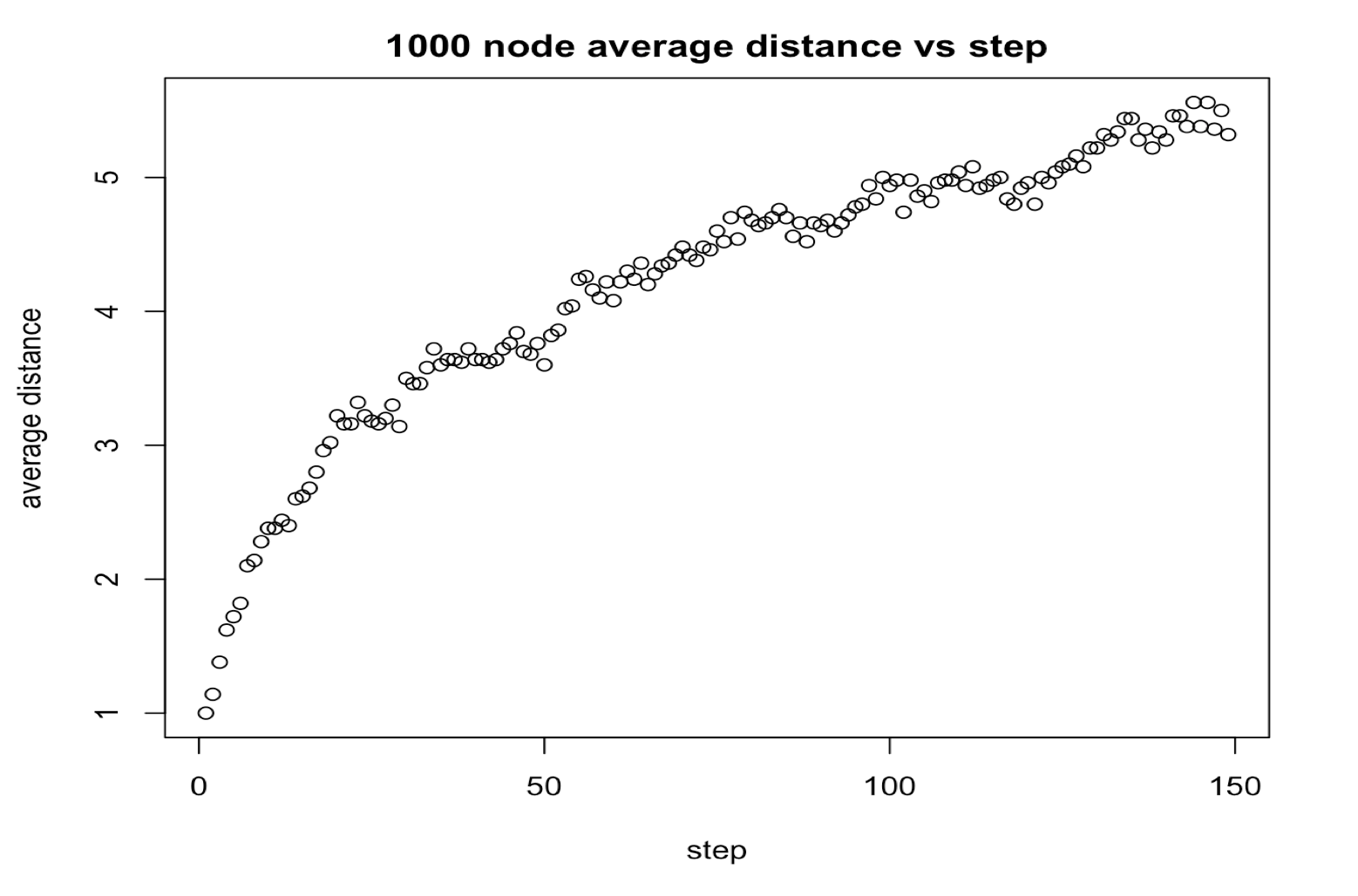
These results make sense because the graph of 10000 nodes has a larger diameter, so a random walk starting at an arbitrary node is more likely to happen around the center of a cluster rather than near the corner. The edges are distributed more uniformly around the center, so the random walk’s distance tends to stablize quickly in this graph of a higher number of nodes.

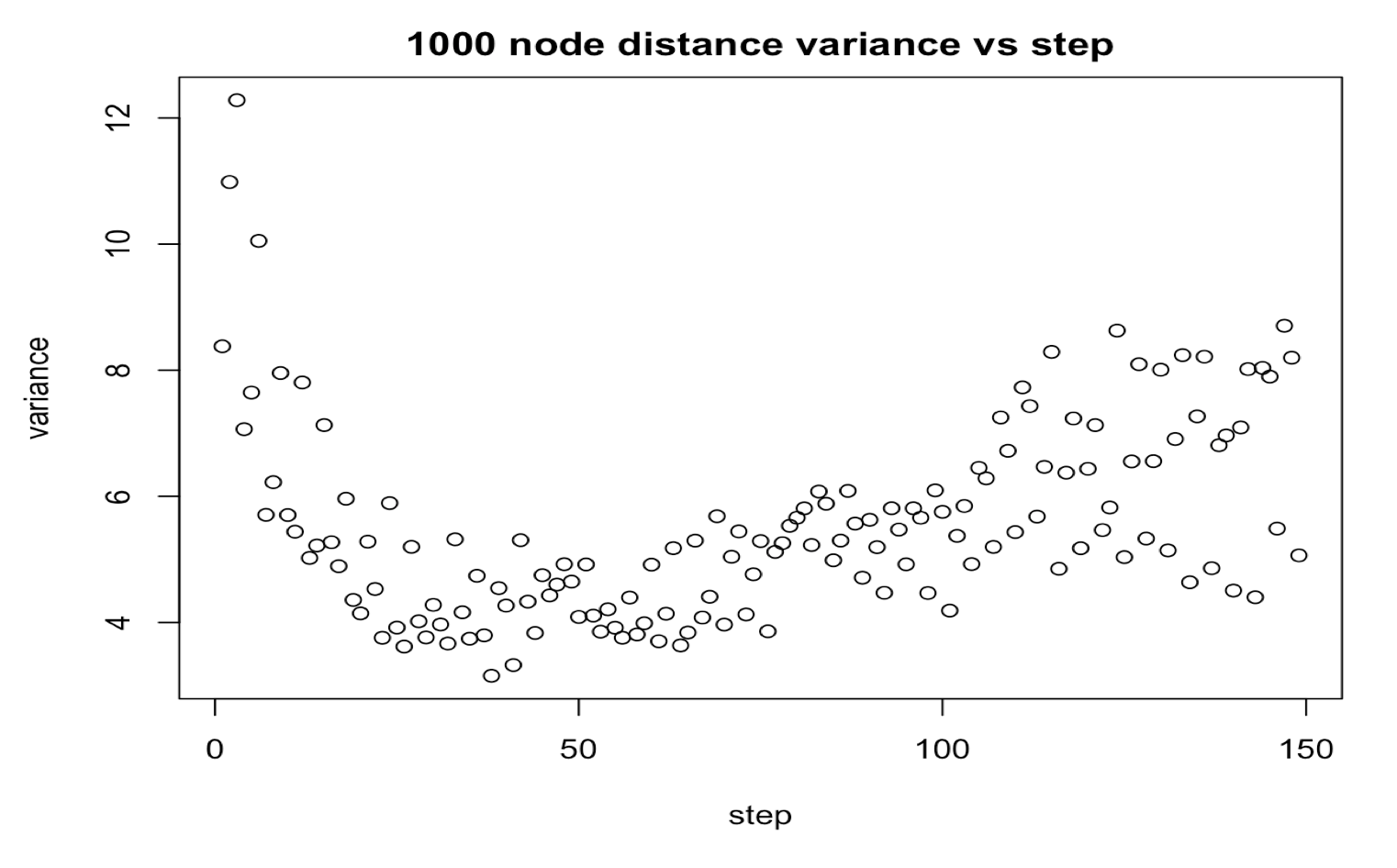
### Question 2

#### (a)

This Network is created with barabase.game() function provided in igraph library.

#### (b)

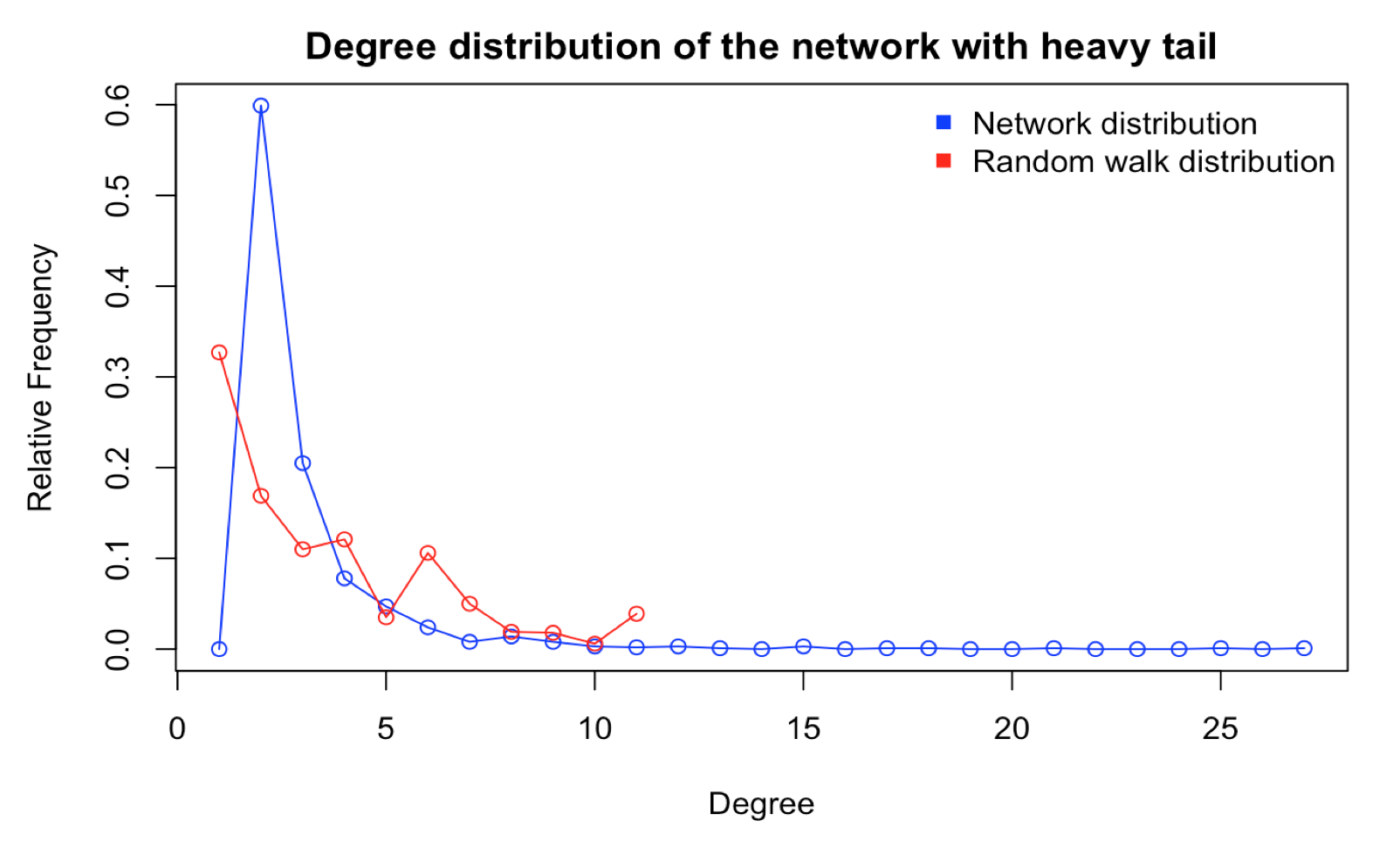




100 out 1000 starting nodes are randomly selected with sample() function. Each starting node initiates a 150 steps random walk using random\_walk() function provided in igraph library. Average and variance values are obtained using the random walk sequence generated by these 100 starting nodes.

We see a steady increase in average distance to starting node as step increases. Distance variance drops initially and then oscillates around 6.

#### (c)

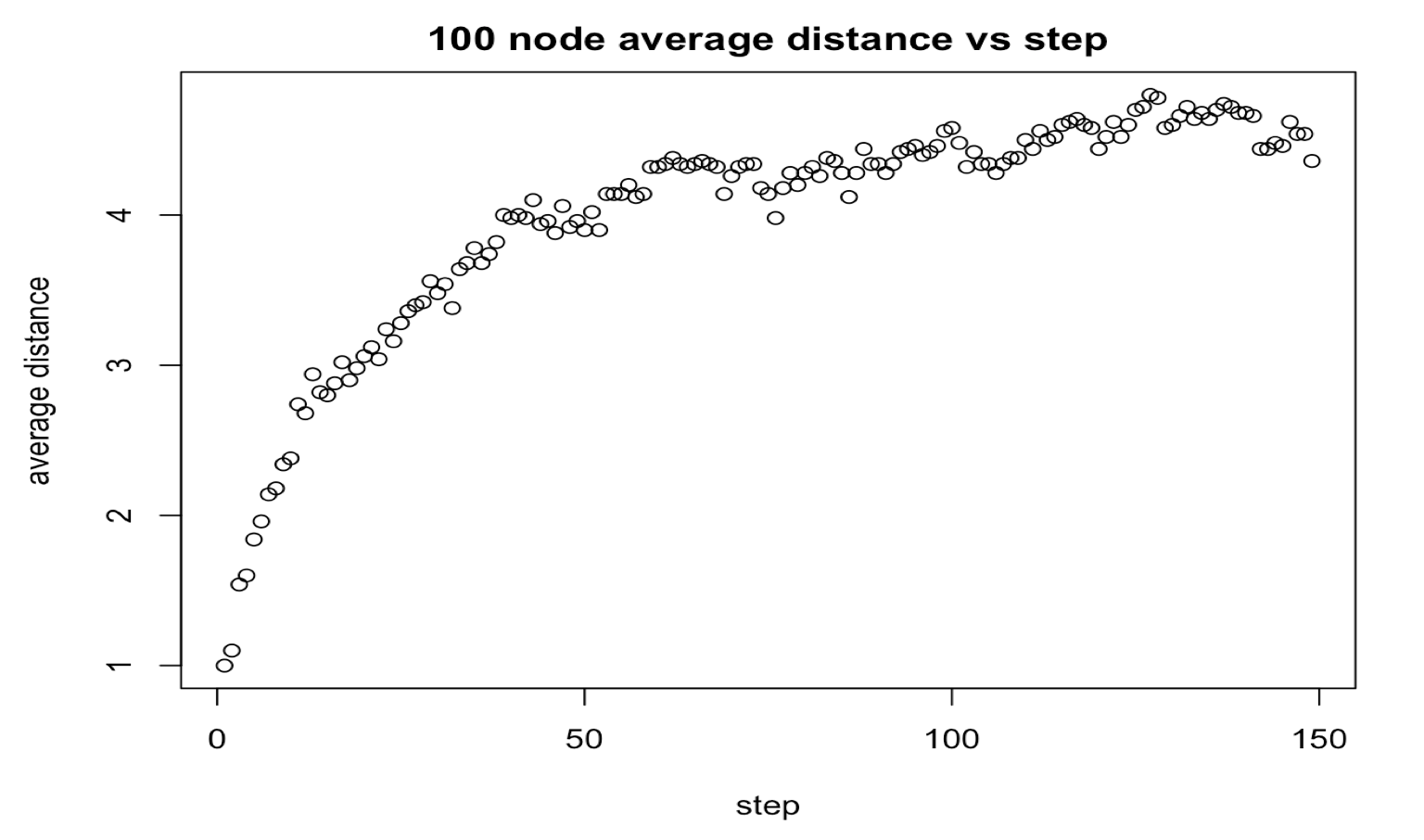
Network degree distribution curve is generated using degree.distribution() function in igraph.

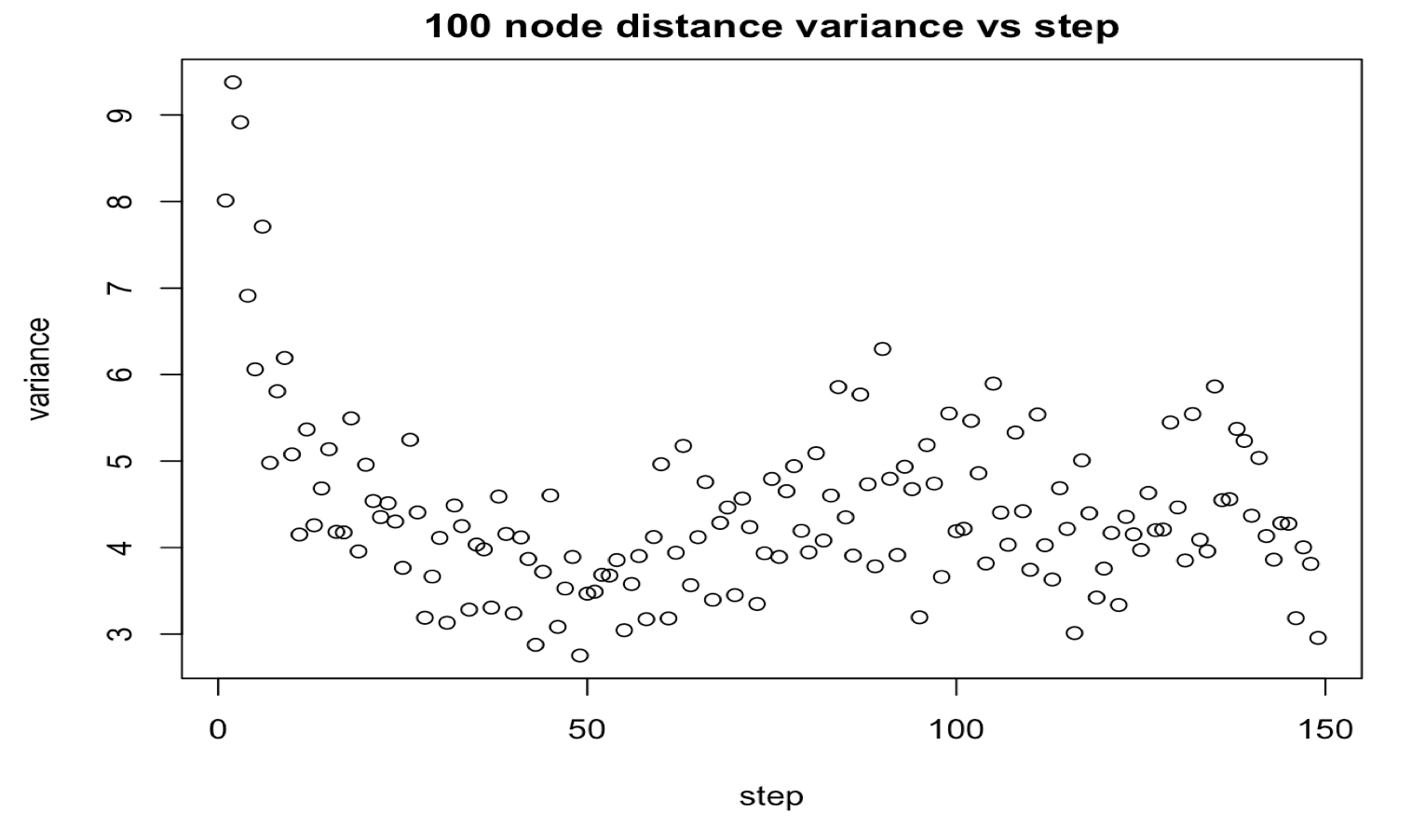
Random walk degree distribution is produced by vertex sequence of a 1000 steps random walk using one randomly selected starting node. degree() function extracts degree of each vertex along the walk and table() function extracts the frequency table.

The overall network degree distribution has a long tail that extends to around 30. The random walk degree stops at degree 11. It seems that random walk can hardly wander into the nodes with large degree. Also overall network has most node with most nodes with degree 3 while random walk has most nodes with degree 1.

#### (d)

Also generated with 100 randomly picked starting nodes and 150 steps 100 and nodes and 10000 nodes networks show the same trend of average distance and variance as 1000 node network. A closer look shows that 100 node network converges to a lower maximum average distance and a lower variance value than 1000 and 10000 nodes network. In comparison, additional plot generated with 500 steps random walk for 10000 nodes network is produced. Clearly when the network size is larger, average distance and variance could reach a higher value with more steps. Distance increases because random walk could proceed to a farther point. Variance increases because lower degree nodes is still the majority and random walk could easily go back to a smaller distance.





### 

### 

### 

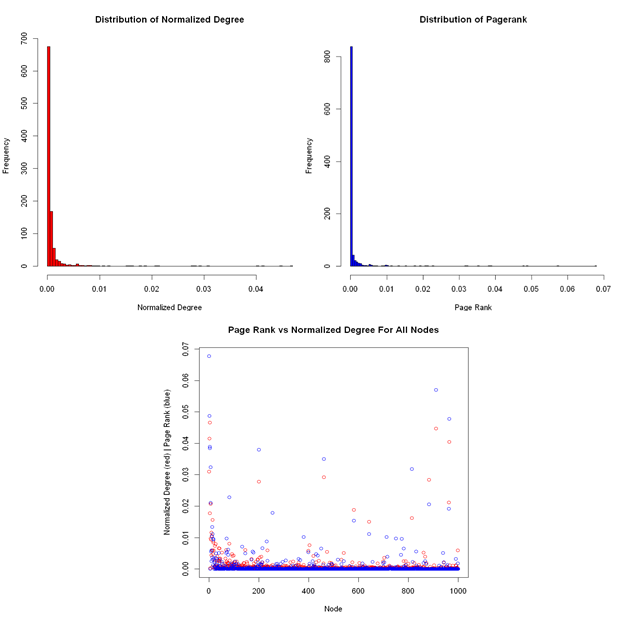
### 

### 

### Question 3

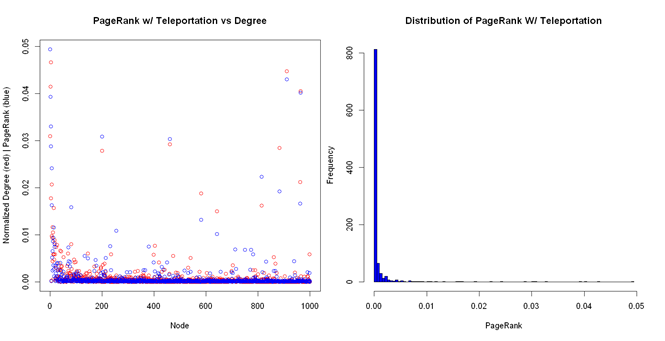
**3a)**

From the graph of page rank vs normalized degree above the probability that a walker visits a node is correlated for many nodes and is somewhat related. The normalized degree is a good comparison as the sum of all degrees is now 1 so that we can make a one to one value comparison with page rank. A lot of early nodes built through the PA model have high degree and seem to also have high page rank. For the most part it seems that most nodes who have small degrees also have small page ranks. However, it is also clear that they are not the same as we can see many examples of nodes with large differences between pagerank and degree. Nodes with relatively high degree also tend to have higher page ranks but their values are still somewhat different. Generally high degree nodes seem to have an even higher PageRank while low degree nodes have an even lower PageRank. From the subsequent graphs on the distribution of page rank and degree values we see that the distribution of these two attributes of nodes seem to highly concentrated towards 0. However, we see that pagerank values are even more concentrated towards 0.



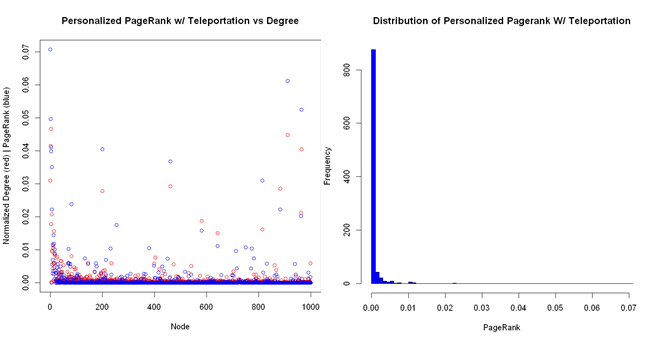
**3b)**

From the graphs below it appears that the situation is like the relationship in part a) where degree and probability are similar for the most part but there also multiple examples where the degree and probability for a given node is not really related. However, we do notice that for low degree nodes the page rank is closer in value and high degree nodes to not have relatively even higher page rank as part a). This makes sense as adding a teleportation probability improves the page rank of lower page rank nodes in part a) and deflates the page rank of higher page rank nodes as it acts as regularization. Other than this the similarity to a) also makes sense as the probability of jumping to specific node if we decide to teleport is equal for all nodes so the relativity of pagerank values should not be greatly affected.



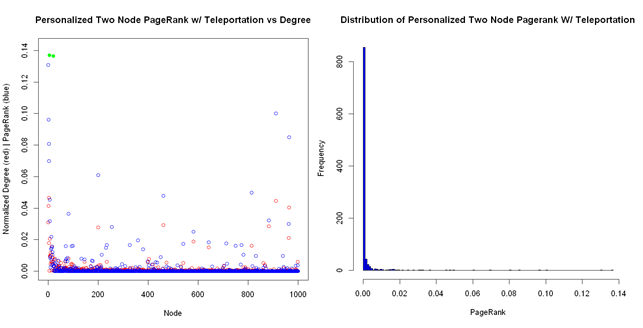
### Question 4

**4a)**

The results are very similar to 3a). This makes sense as a personalized PageRank where the probability of jumping to a node when teleporting is proportional to the nodes' PageRank is essentially PageRank without teleportation but just with an added randomness in teleportation.

**b)**

From the graphs below the PageRank of the non-median nodes seem to be basically the same as 3a) and 4a) except now the page rank of the median nodes are now much higher than before and even higher than all the other nodes. This makes sense as any time we are teleporting with a probability of 0.15 we are always going to land on one of these two nodes. Likewise, the PageRank of these two nodes are very close to each other as their original pagerank values were already very close and we had equal probability of teleporting to either of these two nodes.



**c)**

Let us assume we have a history of nodes that a user has visited sorted by number of times visited. This history vector of length t is noted as . For instance, if I visited google.com 10 times and facebook.com 6 times and myucla.edu 4 times and all other websites less than 2 times. Then hist[0] = the node corresponding to google.com and his[1] corresponds to facebook.com etc. This would be trivial to compute of we know a user’s visited history of visited sites. Likewise, we have a parameter used to determine how many top visited websites we care about. Then our adjusted equation for PageRank becomes

