Large-Scale Social and Complex Networks: Design and Algorithms
ECE 232E Summer 2018
Prof Vwani Roychowdhury
UCLA, Department of ECE

Project 1
Random Graphs and Random Walks

Due on Monday, July 16, 2018 by 11:59 pm

Team Members

Jennifer MacDonald, UID: 604501712
Nguyen Nguyen, UID: 004870721
Sam Yang, UID: 604034791

Introduction

This project is intended to explore igraph library by generating different networks and measure various properties of the network. The preferred language is R which we will be using in this project. We will explore random networks using the Erdos-Renyi (ER) model and the Barabasi-Albert preferential attachment model, that penalizes the age of a node. We will also explore random walk on the Erdos-Renyi networks, networks with fat-tailed degree distribution, PageRank, and Personalized PageRank.

Erdos-Renyi Model

Erdos-Renyi model, perhaps the simplest graph model, is an undirected graph which is specified by two parameters: the number of vertices in the graph *n*, and the probability of an edge *p*. Basing on these parameters, a graph can be generated by including an edge between each pair of vertices with probability *p*, independently for each pair. The probability can be thought of flipping a coin with bias *p* for each possible edge. Furthermore, the distribution of the degree of any particular vertex is binomial:

$$P(\deg(v) = k) = \binom{n-1}{k} p^k (1-p)^{n-1-k},$$

Where n is the total number of vertices in the graph.

Barabasi-Albert Model

Barabasi-Albert model is a scale-free network that has power-law degree distributions. It incorporates two important general concepts: growth and preferential attachment. It means that the more connected a node is, the more likely it is to receive new links. Nodes with higher degree have stronger ability to grab links added to the network. For example, preferential attachment can be utilized to understand how social networks are connecting people. Barabasi-Albert network begins with an initial connected network of *m_o* nodes. New nodes are connected with a probability that is proportional to the number of links that the existing nodes already have. The probability *pi* is connected to node *i* is defined as:

$$p_i = \frac{k_i}{\sum_j k_j},$$

Where k is the degree of node *i*, and the sum is made over all pre-existing nodes *j*.

1. Generating Random Networks

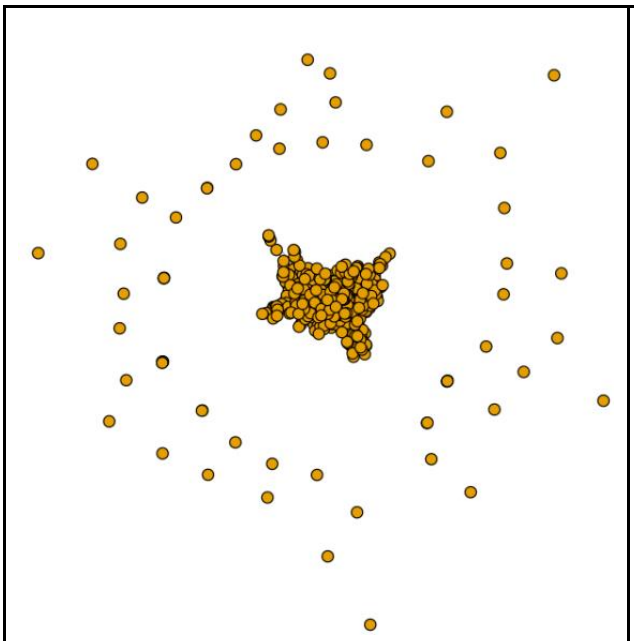
1. Create random networks using Erdos-Renyi (ER) model

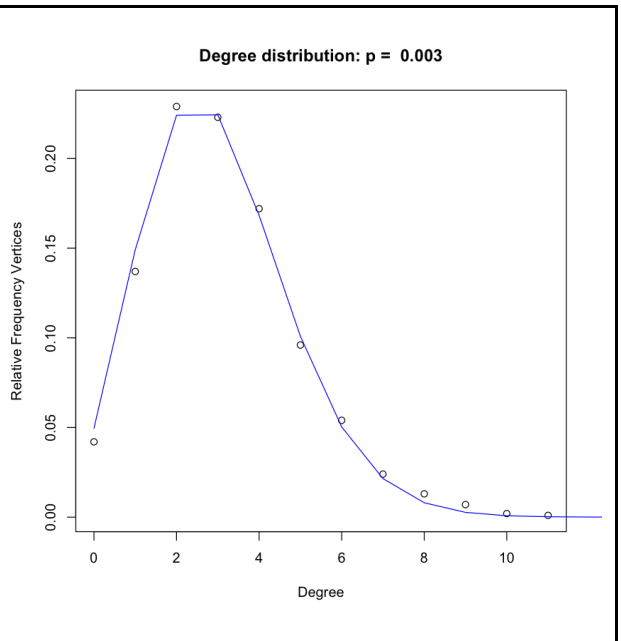
    (a) Create an undirected random networks with n = 1000 nodes, and the probability p for drawing an edge between two arbitrary vertices 0.003, 0.004, 0.01, 0.05, and 0.1. Plot the degree distributions. What distribution is observed? Explain why. Also, report the mean and variance of the degree distributions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

For each probability, we created a random network and plotted its nodes and vertices. We also created a frequency table and plotted the frequency of the number of degrees. Both are seen below in Table 1.

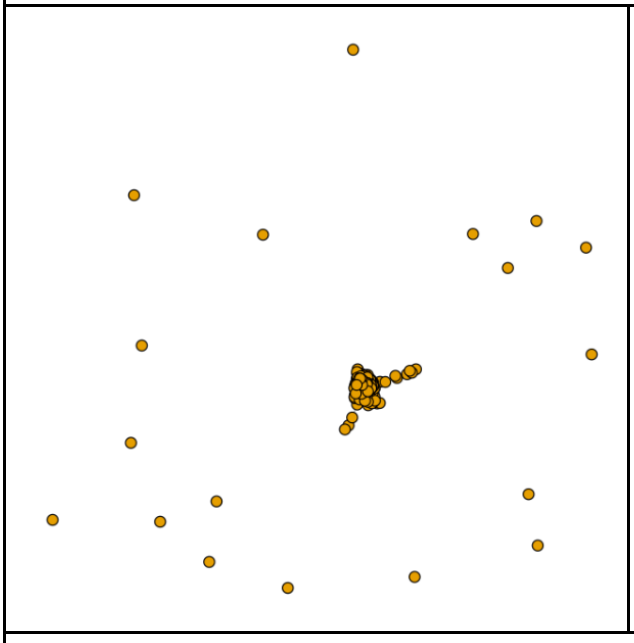
Table 1: Plots of undirected random ER networks and their degree distributions

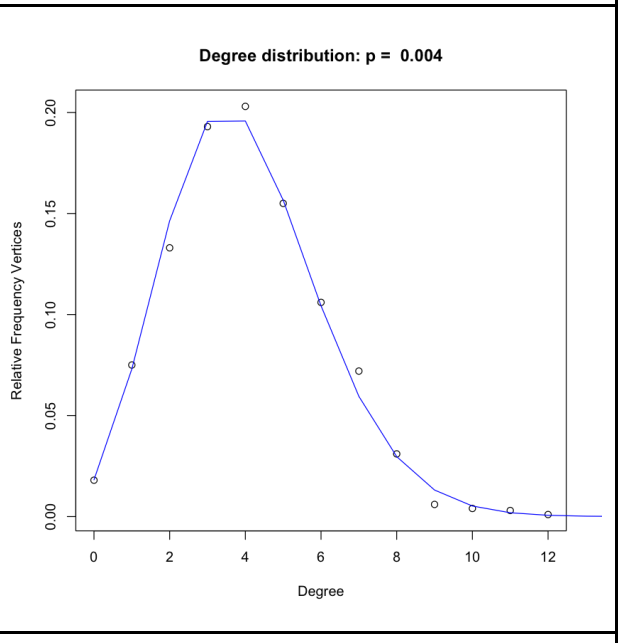
p value = 0.003



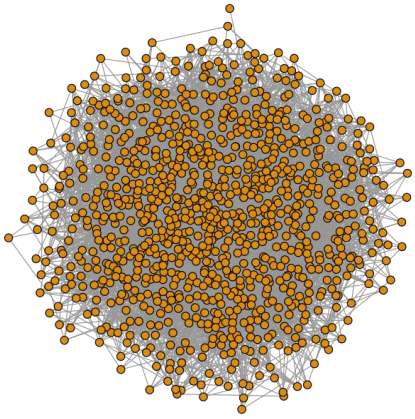


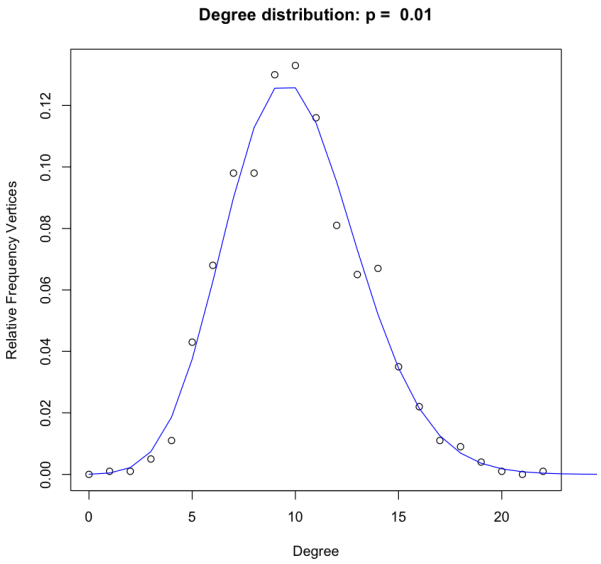
P value = 0.004



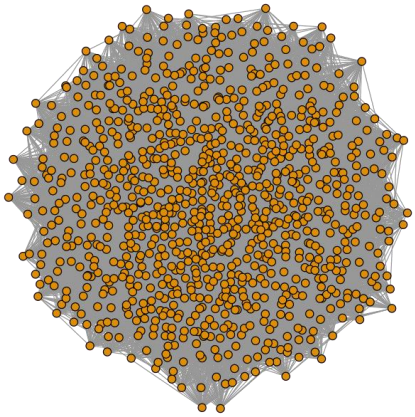


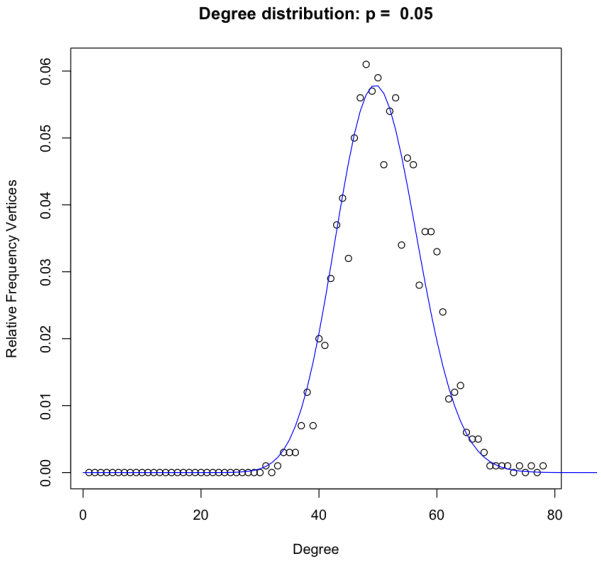
P value = 0.01



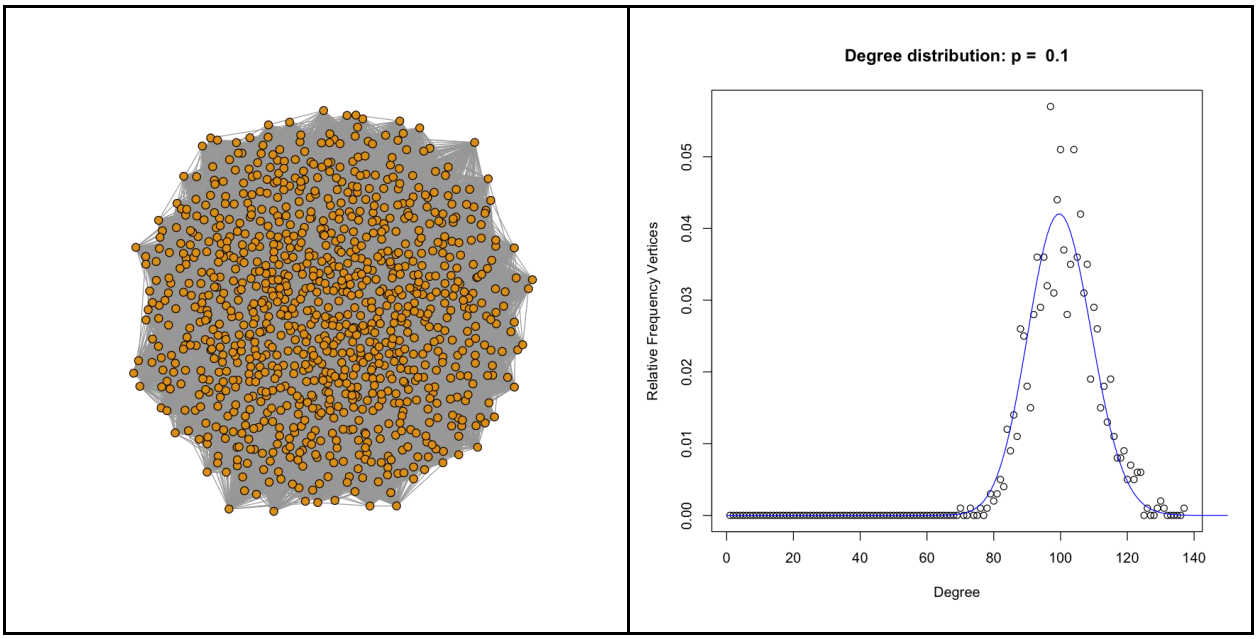


P value = 0.05





P value = 0.1



From the figures above, we observed that there is a binomial distribution for each Erdos-Renyi network. As discussed above, In G(*n*,*p*) network, every edge exists independently and with the same probability. The total probability of drawing a graph with *m* edges is:

$$\Pr(m) = \binom{\binom{n}{2}}{m} p^m (1-p)^{\binom{n}{2}-m}$$

Which is simply the standard binomial distribution. Thus, we want to compare the calculated mean and variance against the theoretical mean and variance. Our assumption is that X ~ B(n,p), that is X is a binomial distributed random variable. Therefore, the theoretical mean (expected value) can be calculated by using the binomial formula below:

$$\mathrm{E}[X] = np.$$

Likewise, the theoretical variance can be calculated using the formula below:

$$\mathrm{Var}(X) = np(1-p).$$

Table 2: Mean and Variance of Erdos-Renyi networks

P value	Mean	Mean (theoretical)	variance	variance (theoretical)
0.003	3.122	3	3.3084	2.991
0.004	4.034	4	3.9187	3.984
0.01	10.03	10	9.8089	9.9

0.05	50.01	50	48.418	47.5
0.1	100.184	100	91.351	90

Comparing to the calculated values to the theoretical values, we observed that calculated means and variances in larger p values are closer to theoretical means and variances.

(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?

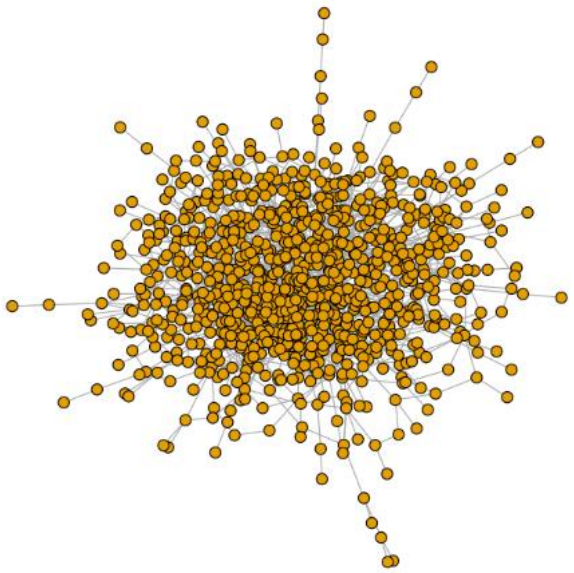
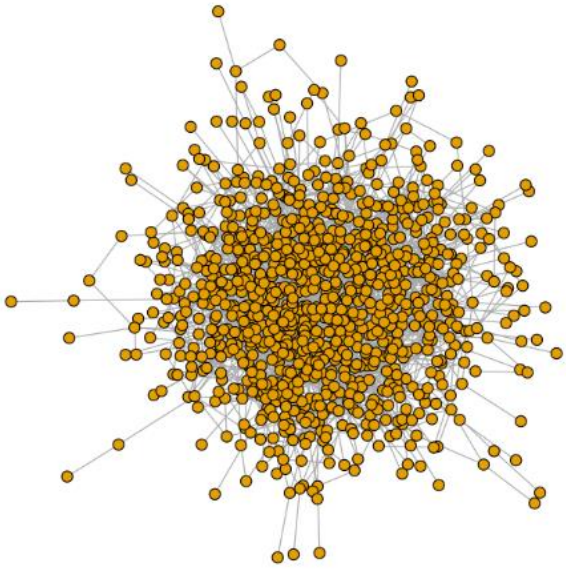
For this part, we generated a network 1000 times in order to check if there were any instances where the graph was not connected to determine the probability that a generated network with certain given parameters is connected, counting the number of times that it was. For the plots that did have one or more instances of unconnected network, we decomposed the graph and took the biggest cluster. Below is Table 3, a table of diameters we observed. It seems that the diameter of GCC for non-connected network seems to decrease as p value increases. Also, we observed that the connected probability increases as p value increases, which intuitively makes sense. Eventually, the graph is all connected if p value is high enough which seems to be around where p value is 0.05.

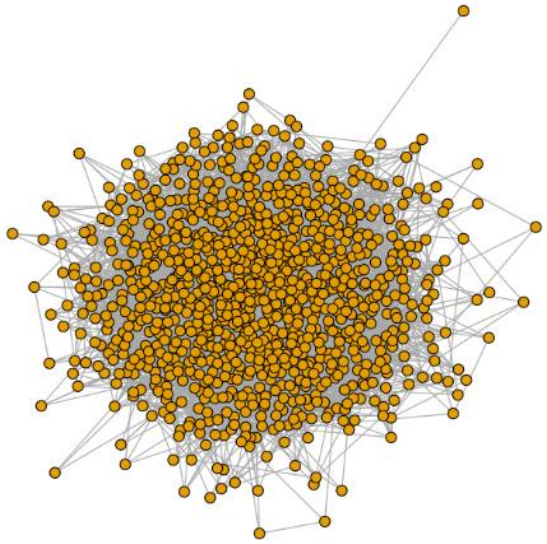
Table 3: Probability and Diameter measurement of Erdos-Renyi networks

p value	Is always connected	Connected probability	Diameter of GCC
0.003	False	0	14
0.004	False	0	10
0.01	False	0.932	6
0.05	True	1	-
0.1	True	1	-

To visualize the giant connected components, we also isolated the GCC's of the networks that had instances where the networks were not connected, and plotted them below in Table 4.

Table 4: Erdos-Renyi networks and their GCC

p value	Plot of GCC
0.003	
0.004	

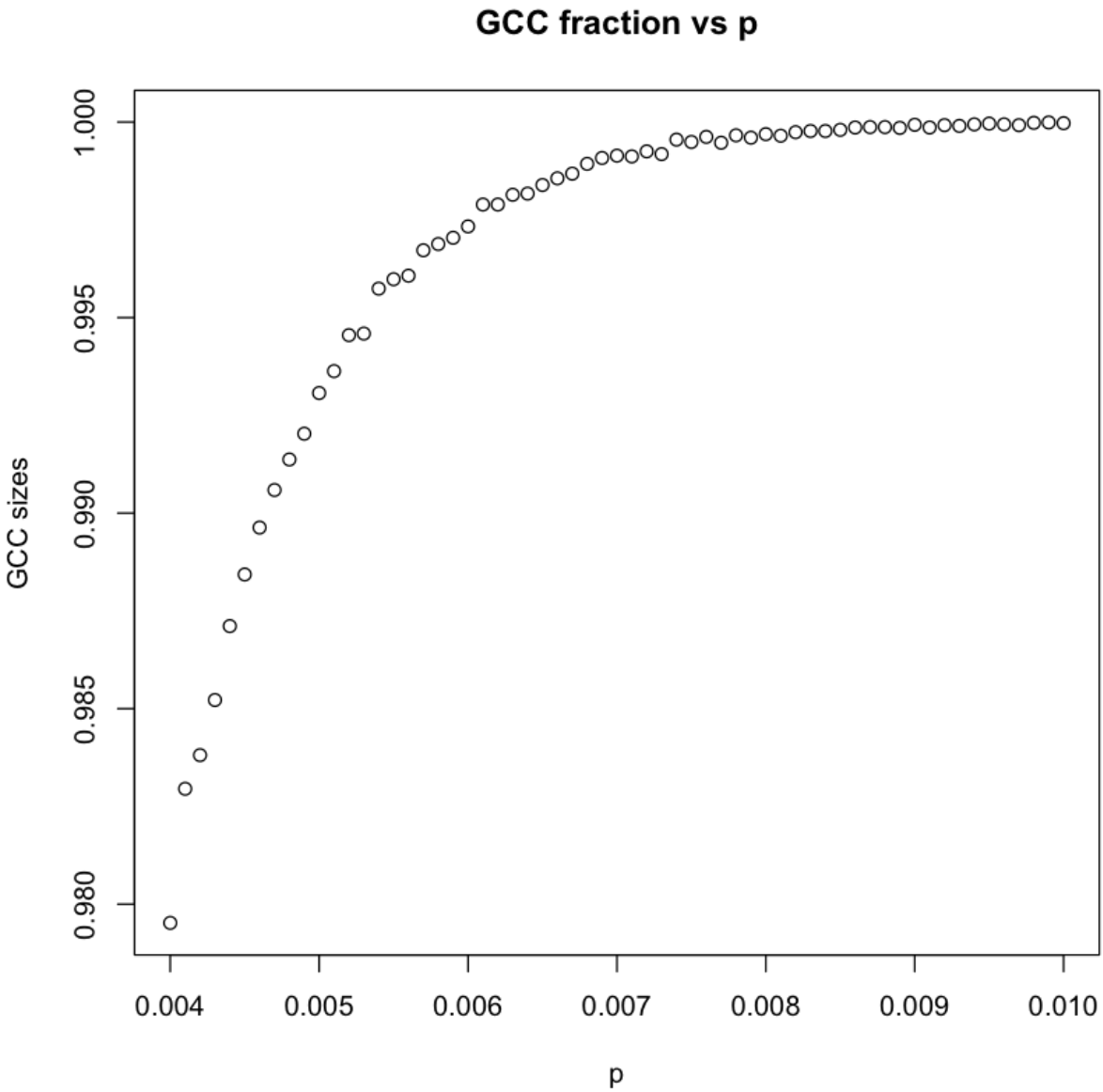
0.01	

(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 where p = O(ln n/n). 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?

As we observed in previous problem, Diameter of GCC is nonlinear as p value increases. We want to observe this phenomenon by generating 100 random networks for peach p value, and observe it on a scatter plot that is normalized GCC sizes (diameter) vs p. Given the property of p = O(ln n / n), we can estimate in the worst case scenario for n = 1000, p value is ln(1000) / 1000 = 0.0069.

We want to focus on the area where big-O is, which is where the scatter plot has reached steady state. As seen in Figure 1 below, the scatter plot shows that the the GCC sizes increases exponentially but reach a steady state around where p value is 0.007, or where a GCC starts to emerge. We define emergence as where the slope of the line visually looks close to 0. This fits with our observation where our calculated/theoretical big-O is 0.0069 above.

Figure 1: GCC sizes vs p value



(d) i. Define the average degree of nodes c = n × p = 0.5. Sweep over the 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?
ii. Repeat the same for c = 1.
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.

In previous problem, we showed that as GCC size increases, n increases and eventually reaches a steady state. Similarly in this observation, there is a logarithmic trend upward for the GCC sizes

as n increases. When n is big enough (more than 2000 in this case), the relationship between GCC size and n is closed to linear as seen in Figure 2 below.

Figure 2: GCC sizes vs n with c = 0.5

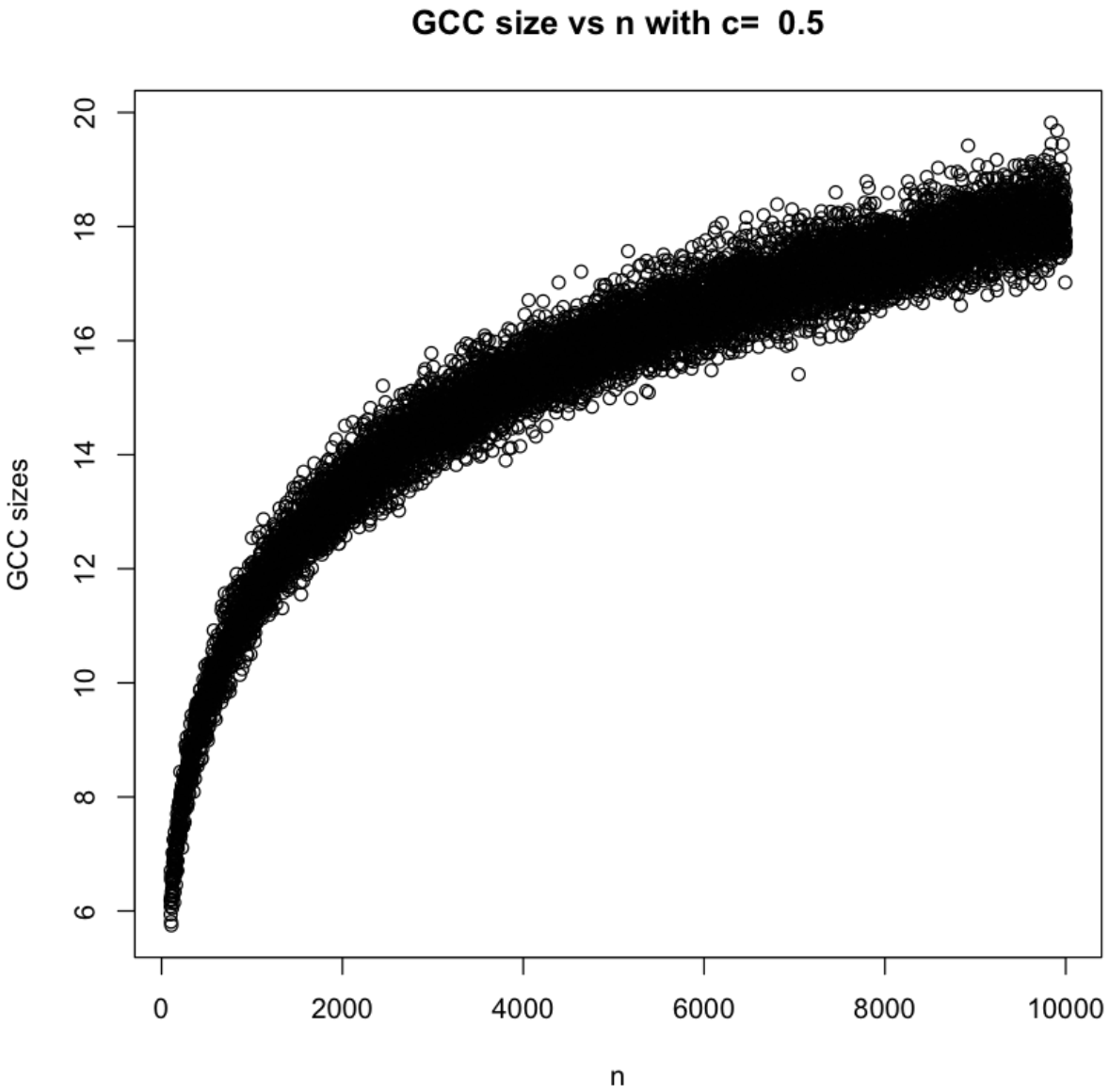
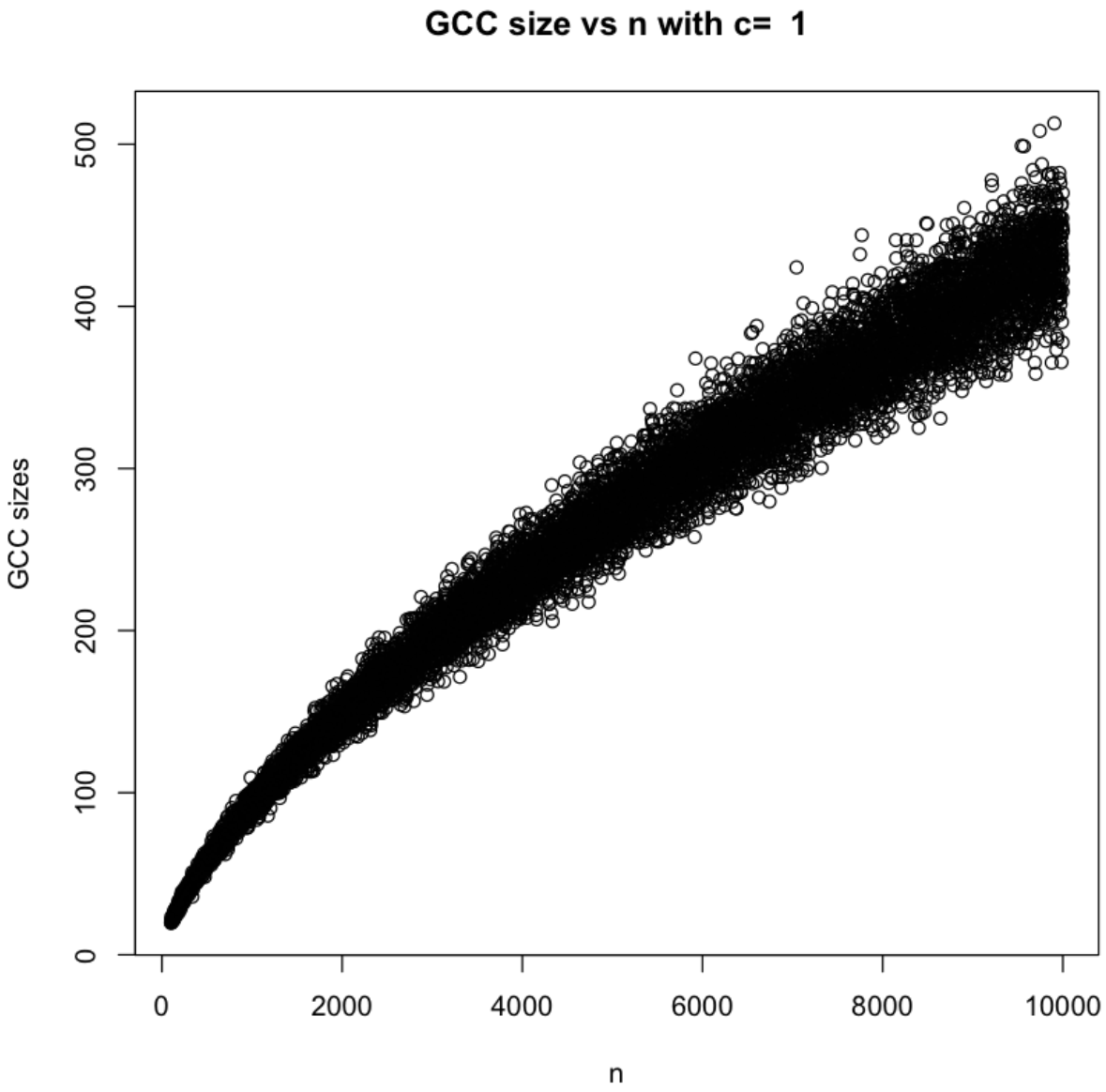
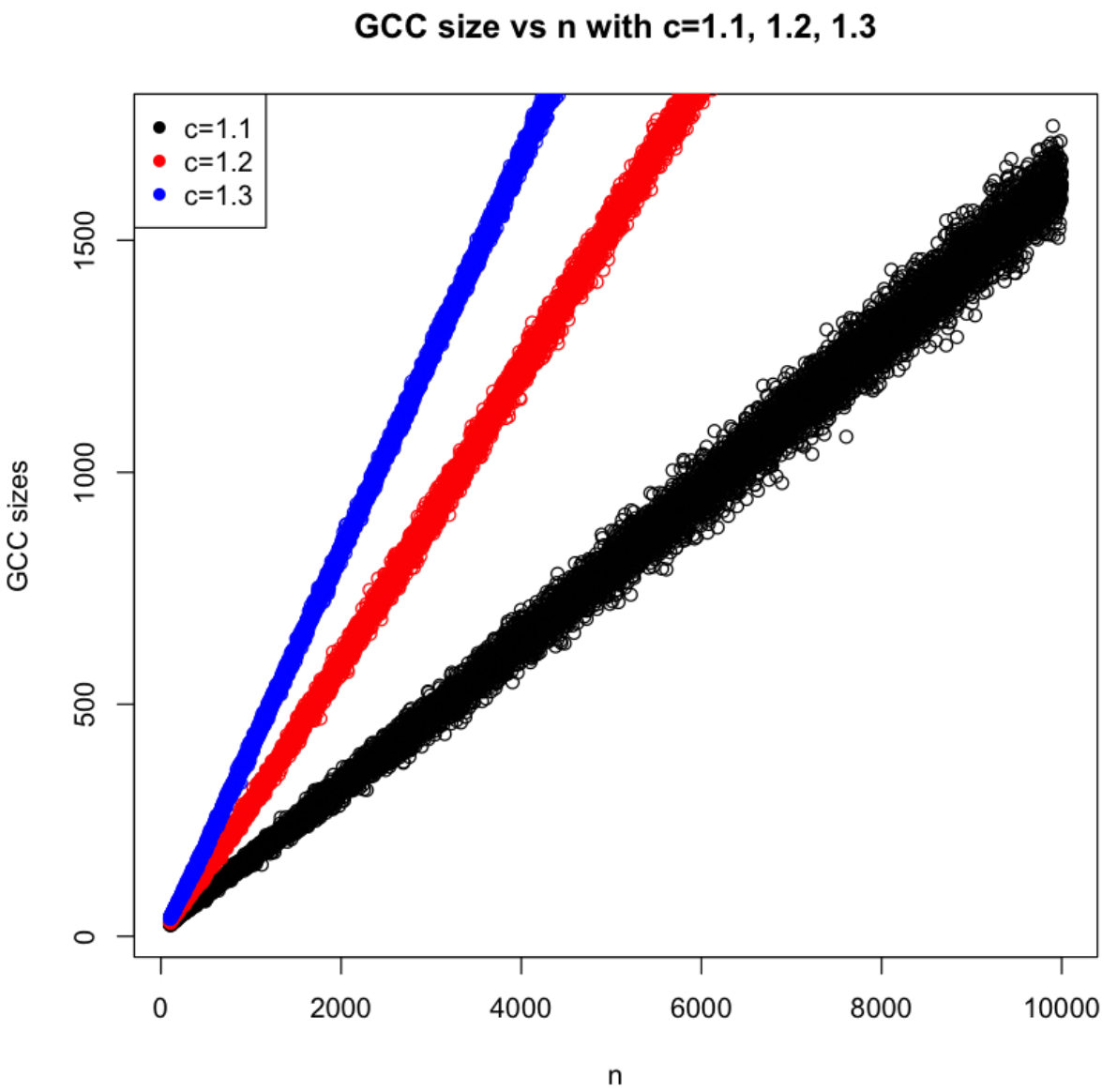


Figure 3: GCC sizes vs n with c = 1



Here, as c is larger (c = 1) in Figure 3 above, the relationship between GCC size and n are more linear. The trend seems more linear than when c = 0.5, but there appears to still be a slight logarithmic curve that levels out as the value of n increases.

Figure 4: GCC sizes vs n with c = 1.1, 1.2, 1.3



From the scatter plots above, we observed that GCC sizes increase as n nodes increase. When c = 1.1, 1.2, 1.3 the linear relationship between GCC sizes and n are most profound. When c is greater than 1, (c = n x p), it means that the probability of connected nodes are higher thus making the GCC size larger. The expected size of the GCC for every n when c = 1.1, 1.2, and 1.3 is shown below in Figure 4.

Therefore, we concluded that as c increases (c = n x p which is the average degree of nodes), it becomes more linear. Therefore, we can conclude that as c increases, more nodes have higher probability to connect and GCC size becomes larger.

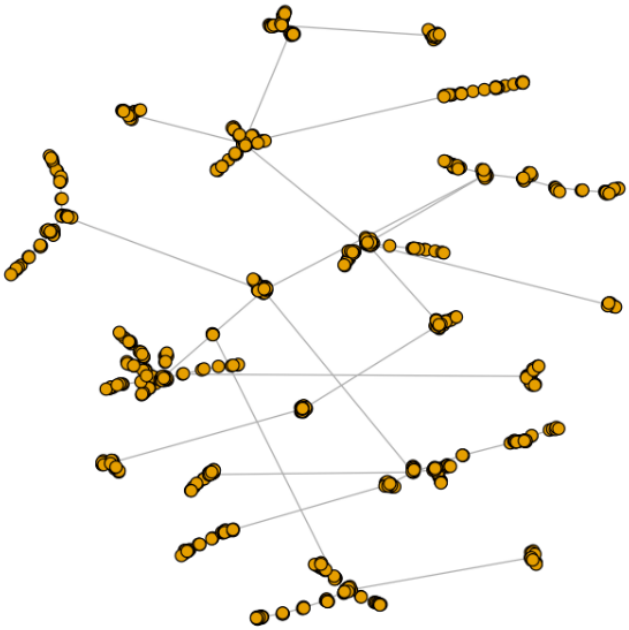
2. Create networks using preferential attachment model

To create a preferential attachment model, we used the built in *barabasi.game* method in igraph library. We set n = 1000 nodes, and m =1 edge attached in each step.

(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?

We looped through 1000 times to generate a preferential attachment network in order to check if there were any instances where the graph was not connected. Since we observed that the network was connected in every instance we generated, we conclude that a graph with these parameters is thus always connected. Below, Figure 5 is an instance of a Barabasi-Albert network.

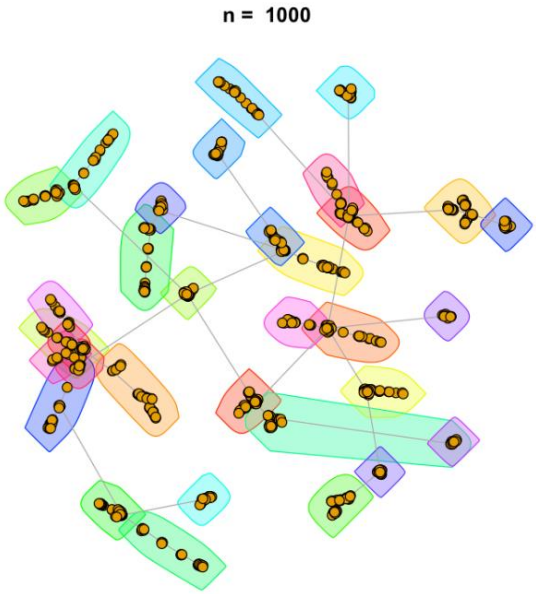
Figure 5: Instance of a randomly-generated Barabasi-Albert network



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

We used the built in functions of igraph to group the points into communities, and then plotted the graph with the different communities color-coded, as seen below in Figure 6.

Figure 6: Network with community structure mappings, n = 1000

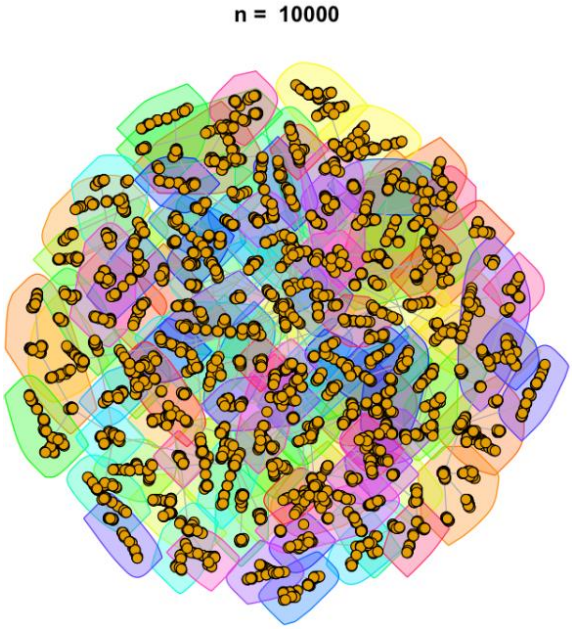


Modularity: 0.933

The modularity for this graph is high at 0.933. This means that the greedy algorithm was able to separate and group communities mostly successfully.

(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?

Figure 7: Network with community structure mappings, n = 10000

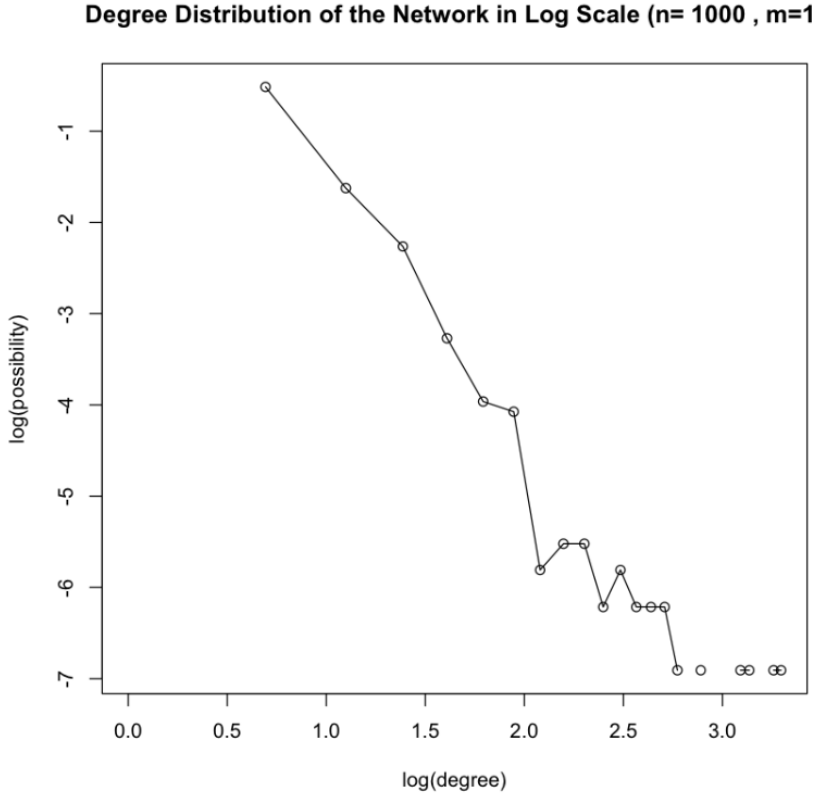


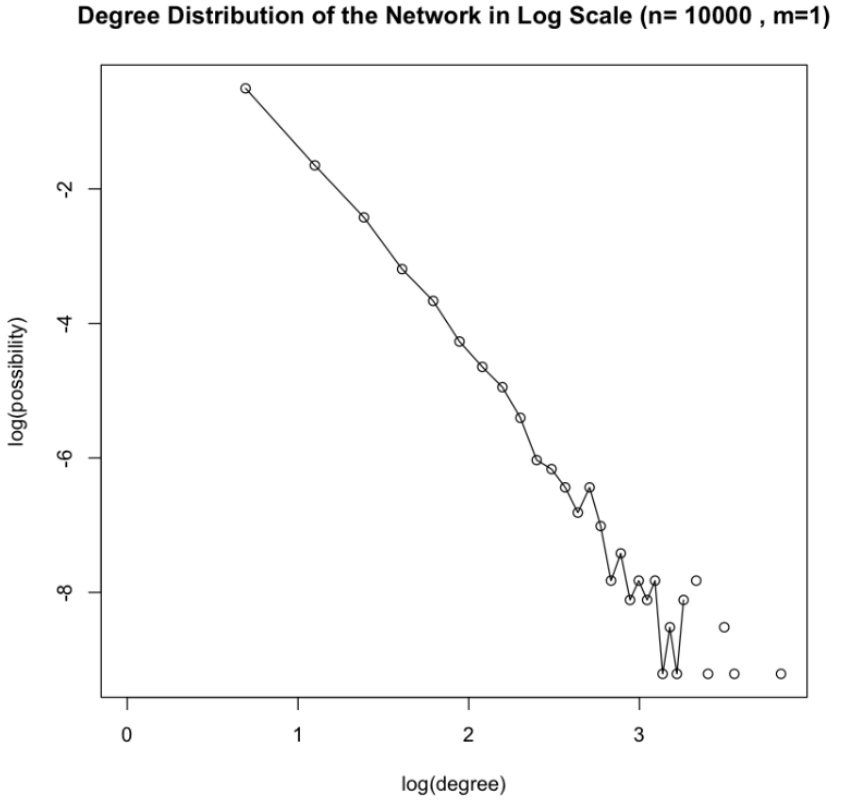
Modularity: 0.979

The modularity of the second, larger network is greater than that of the smaller network. This makes sense since the fast greedy method can suffer from resolution limit, in which communities below a threshold size ends up grouped with other communities. When the value of n is greater, the size of communities is larger this phenomenon happens less frequently, so the modularity value is higher for the larger network.

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

Table 5: Degree distributions in log-log scale

n	Degree Distribution	Estimated slope
N = 1000		-2.6

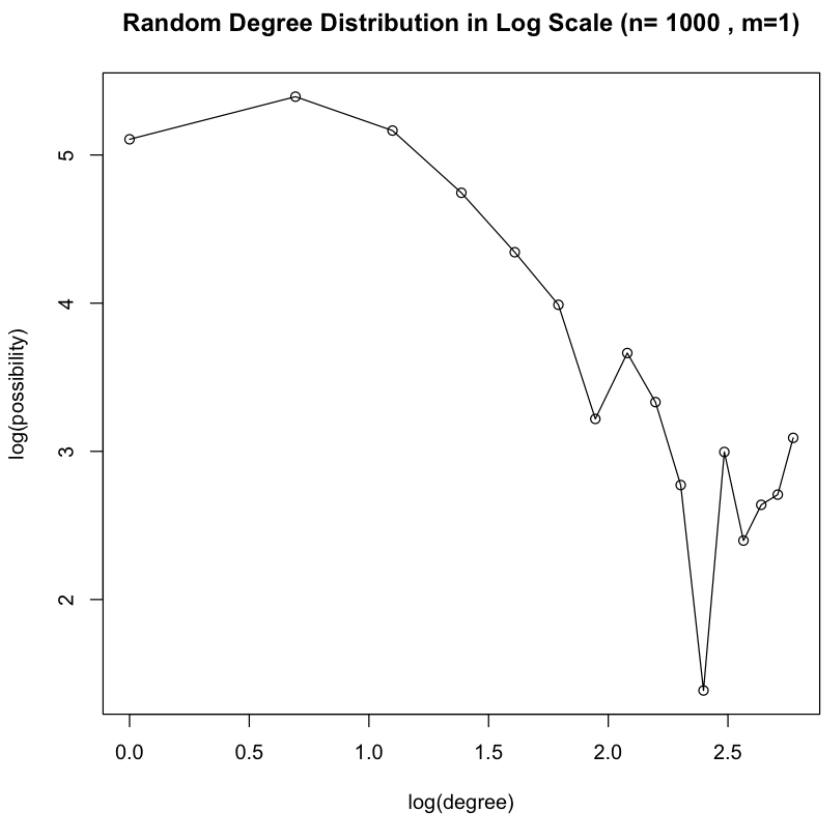
N = 10000		-3.3

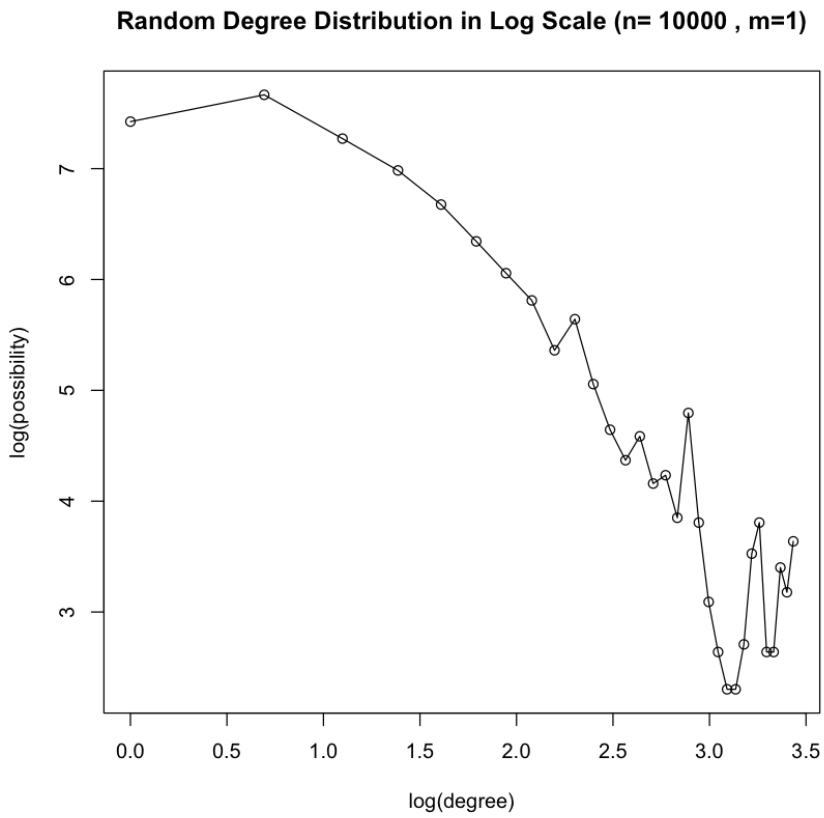
The degree distribution of the larger network appears to have a slightly steeper negative slope.

(e) In the two networks generated in 2(d), perform the following: 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?
Hint Useful function(s): sample

The degree distributions of randomly picked neighbors of randomly picked vertices are shown below in Table 6.

Table 6: Degree distributions in log-log scale of randomly picked neighboring nodes

n	Degree Distribution	Estimated slope
N = 1000		-1.5

N = 10000		-1.8

This differs from the node degree distribution we did earlier because instead of going through the nodes to find their degrees, we randomly pick a neighbor of a random node, so the probability of its neighbor being picked after the initial random choice is higher if their degree is higher.

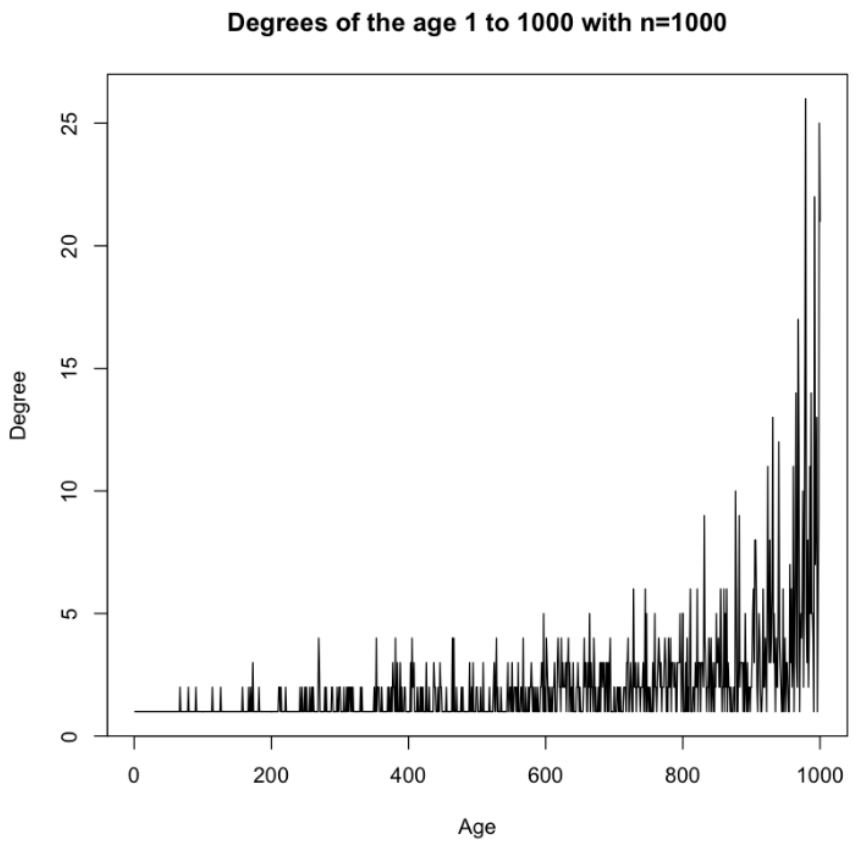
(f) Estimate the expected degree of a node that is added at time step i for 1 ≤ i ≤ 1000. Show the relationship between the age of nodes and their expected degree through an appropriate plot.

In this problem, we calculated the degree for each node and iterated it through 1000 times. We observed that as the nodes get older, the higher degree it's expected to have. Below is the formula to estimate the expected degree of a node added, where s is age, t is time, a is a constant, and c is the average out degree in the network.

$$\gamma_t(s) = a\left[\left(1+\frac{s}{t}\right)^{c/(c+a)} - 1\right]$$

A histogram of the age of nodes and their expected degree is plotted below in Figure 8.

Figure 8: Plot of expected degrees of a node added from 1 to 1000

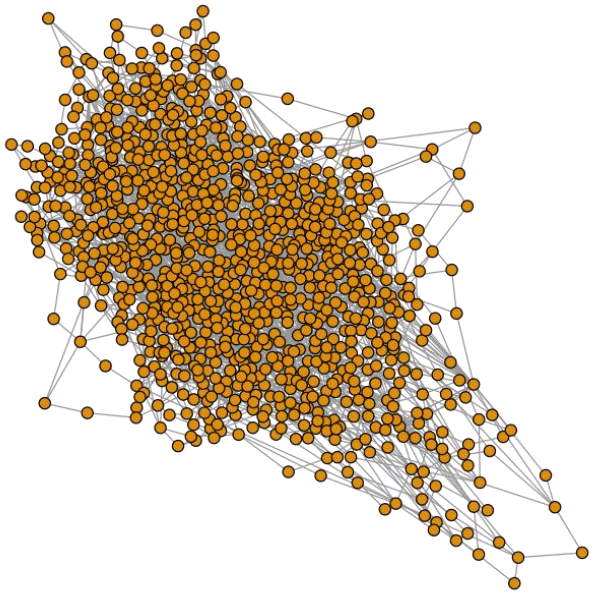


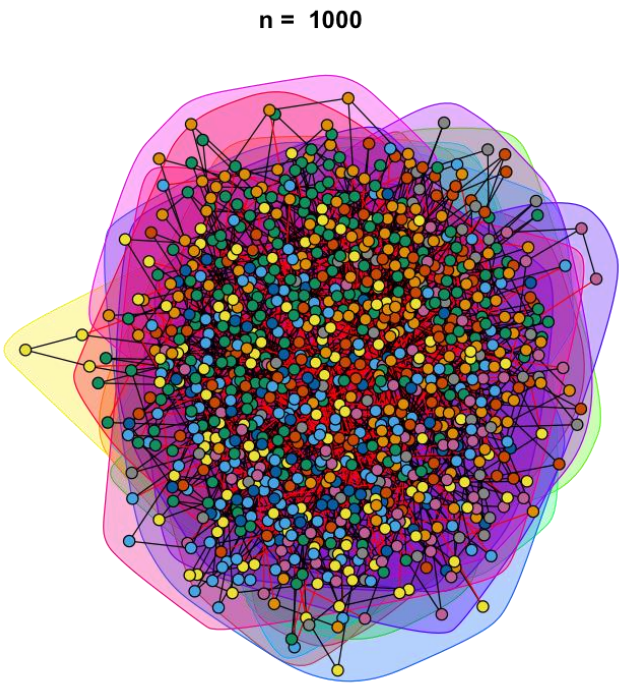
As we can see in the plot, it appears that as the age increases, there is an exponential change in the number of degrees expected. This makes sense since as age increases, there is a greater chance of creating adjacent edges with other vertices, and thus increasing the value of its degree.

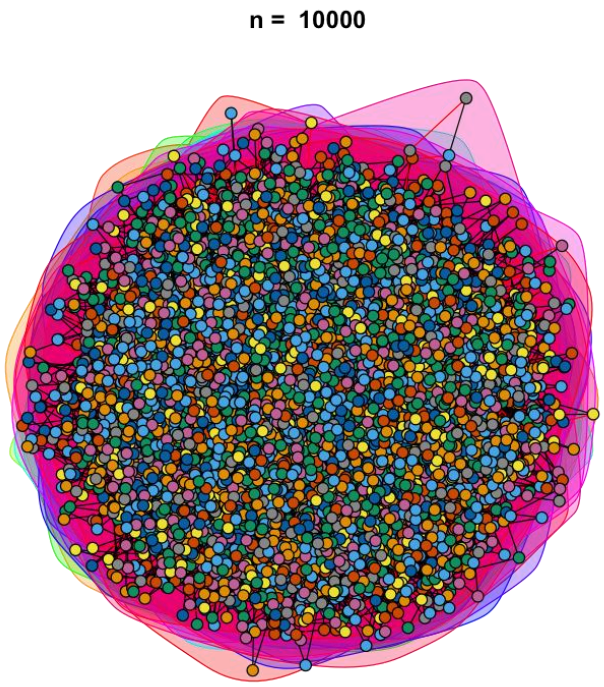
(g) Repeat the previous parts for m = 2, and m = 5. Why was modularity for m = 1 high?

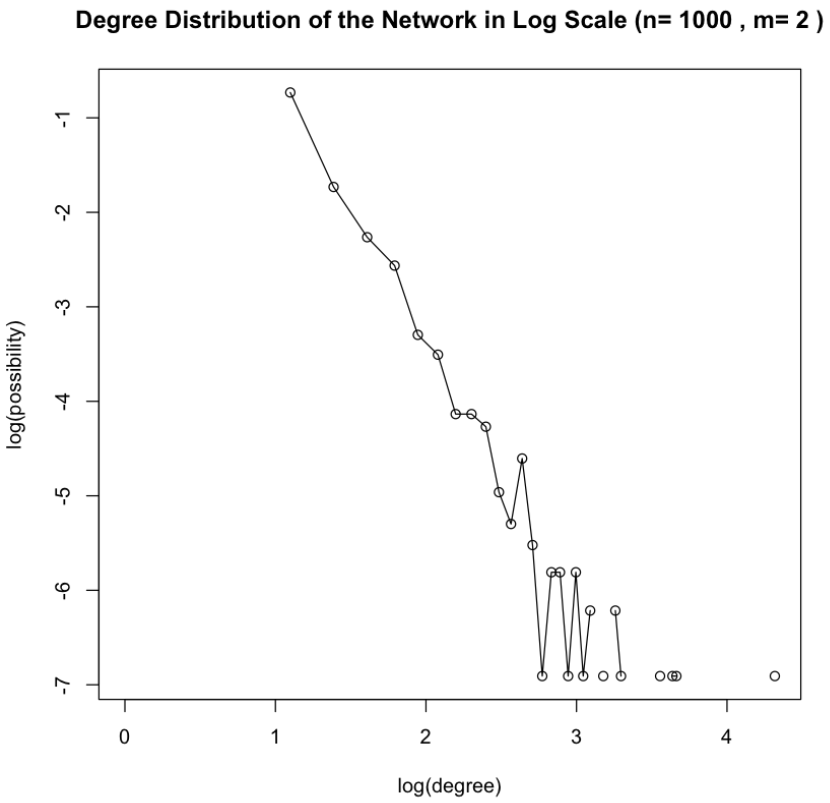
We repeated the process of using the greedy method on plots with m = 2 and m = 5 to group the vertices into communities and then plot the graph with the corresponding communities. We found that such networks always seem to be connected for both m = 2 and m = 5.

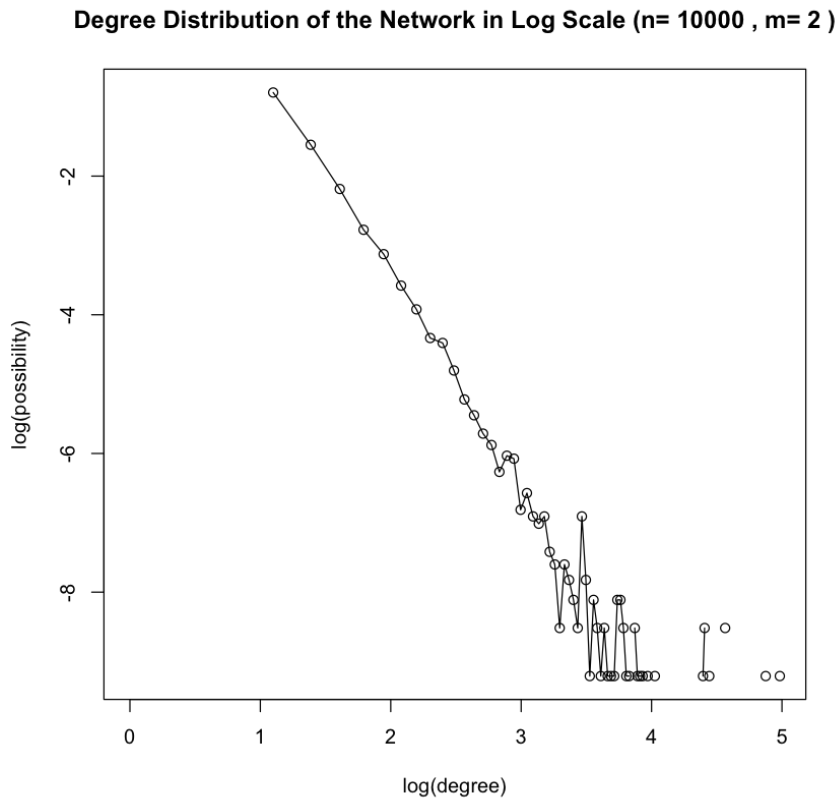
Table 7: Networks with varying m- and n-values

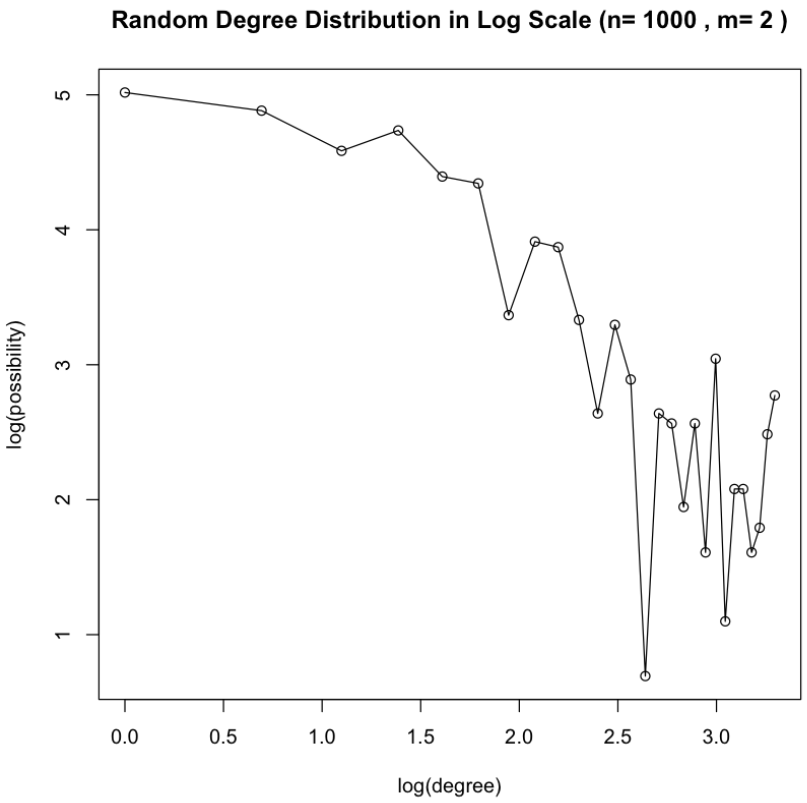
M = 2	
(a) Graph, connected : TRUE	

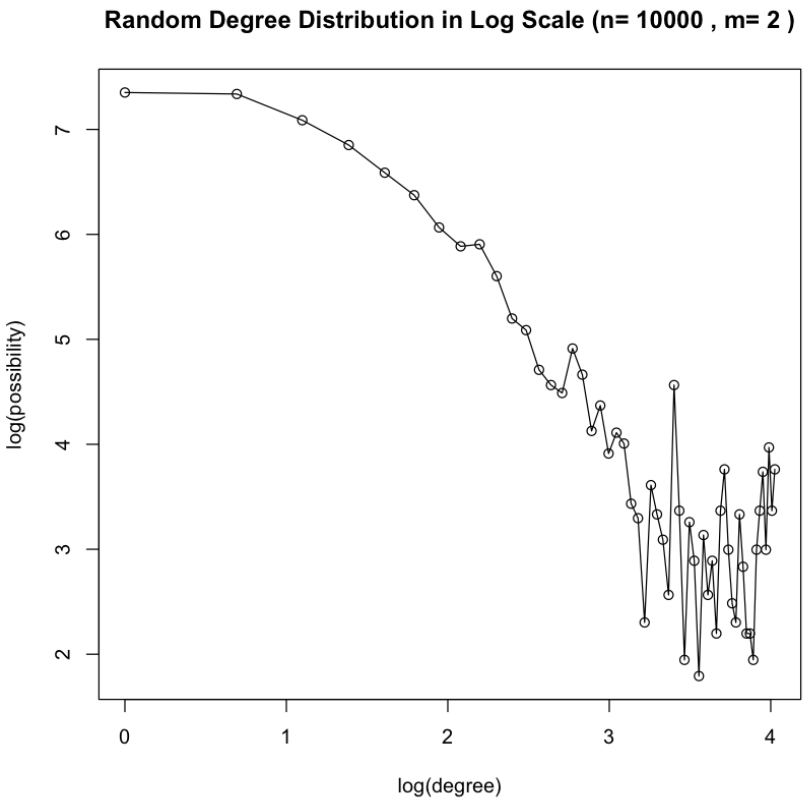
(b) Community Structure Modularity: 0.531	

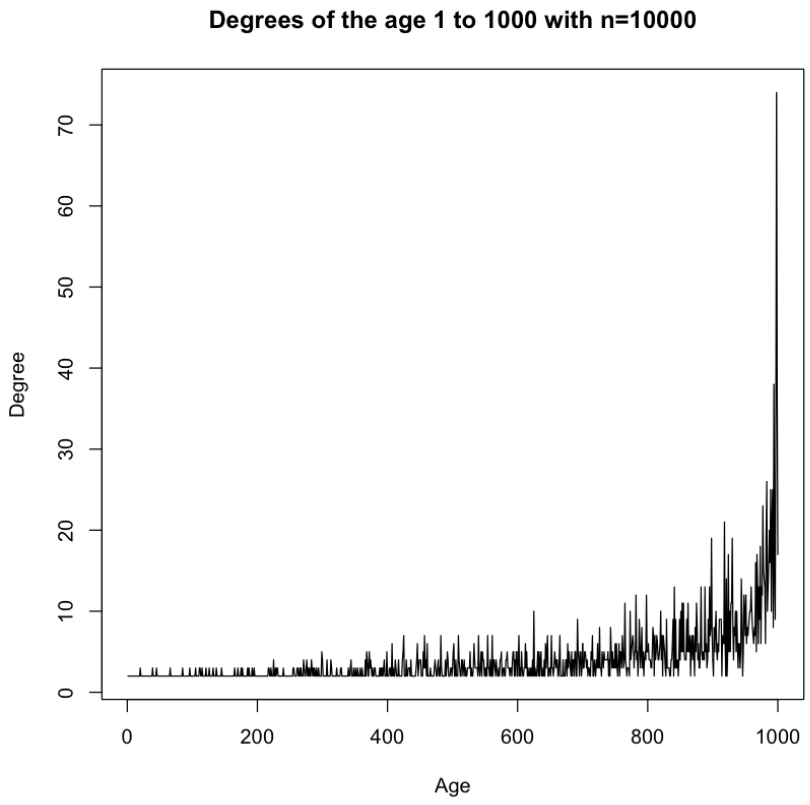
(c) Community Structure, n = 10000, Modularity: 0.530. Smaller modularity than (b)	

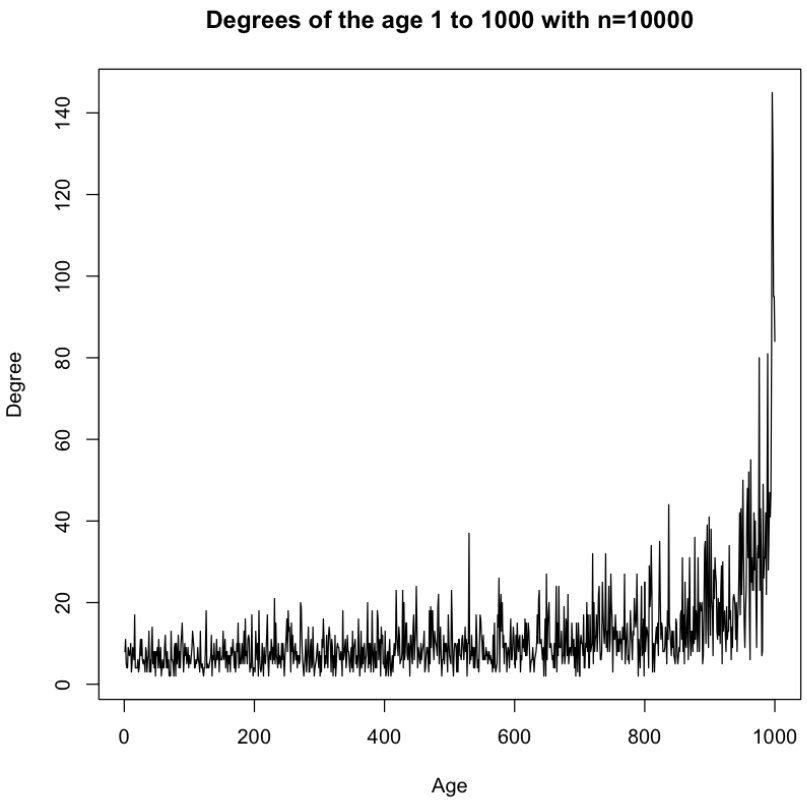
(d) Degree Distribution, N = 1000, Estimated slope:-3.0. N = 10000, Estimated slope:-3.3.	


Degree Distribution of the Network in Log Scale (n= 10000 , m= 2)
log(possibility)
-2
-4
-6
-8
0
1
2
3
4
5
log(degree)

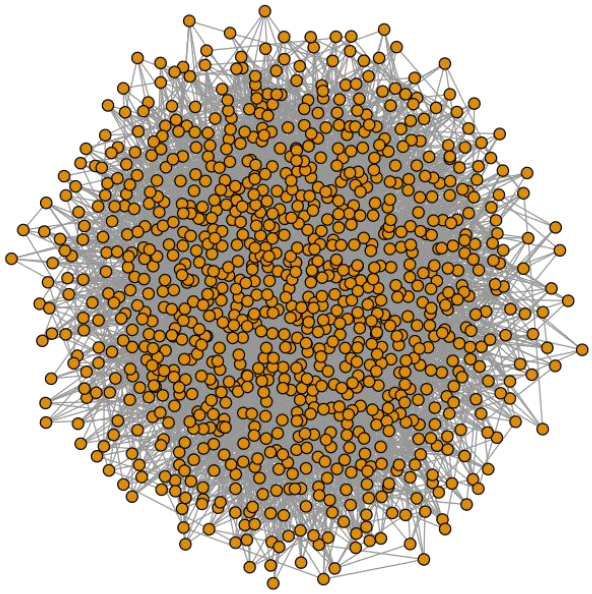
(e) Random Degree Distribution, Differs from node degree distribution we did earlier because instead of going through the nodes to find their degrees, we randomly pick a neighbor of a random node, so the probability of its neighbor being picked after the initial random choice is higher if their degree is higher.	

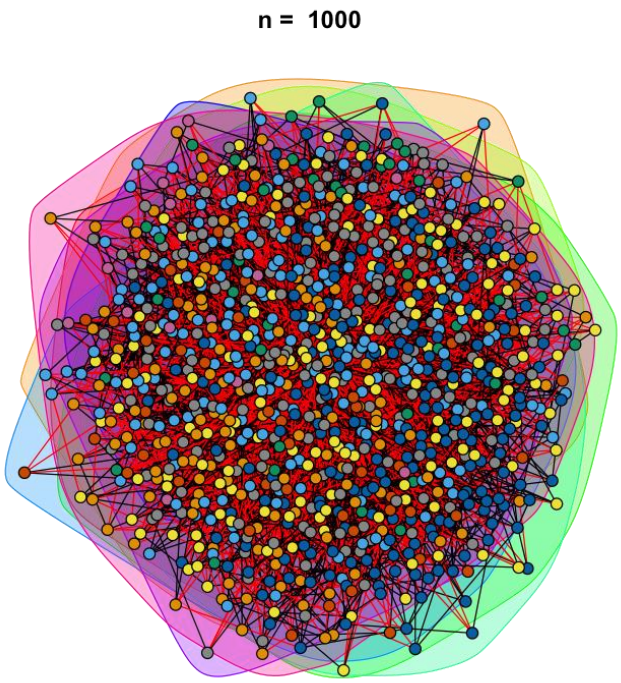

Random Degree Distribution in Log Scale (n= 10000 , m= 2)
log(possibility)
log(degree)
2
3
4
5
6
7
0
1
2
3
4

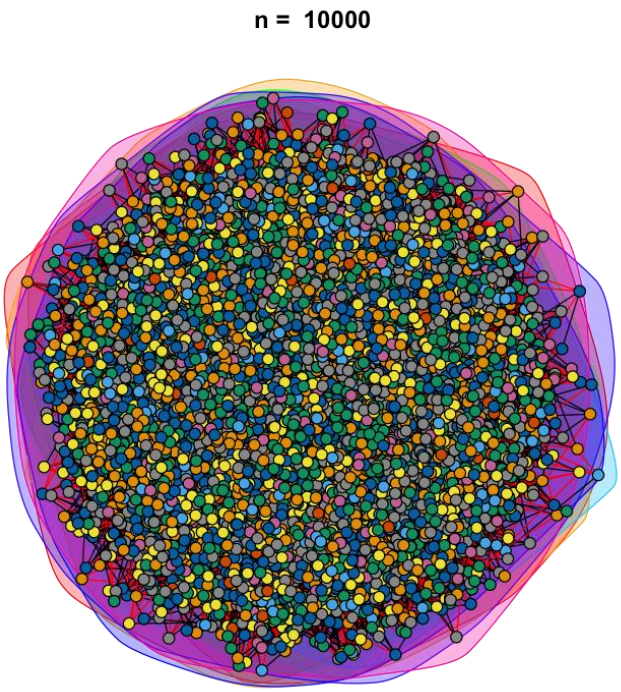
(f) Estimate expected degree	

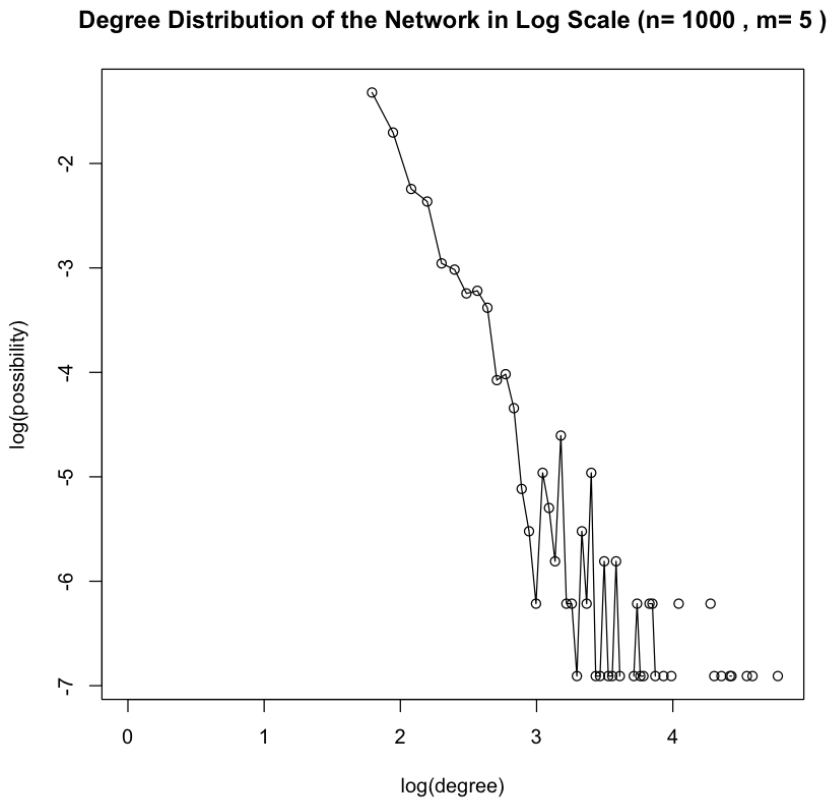

Degrees of the age 1 to 1000 with n=10000
Degree
0
20
40
60
80
100
120
140
0
200
400
600
800
1000
Age

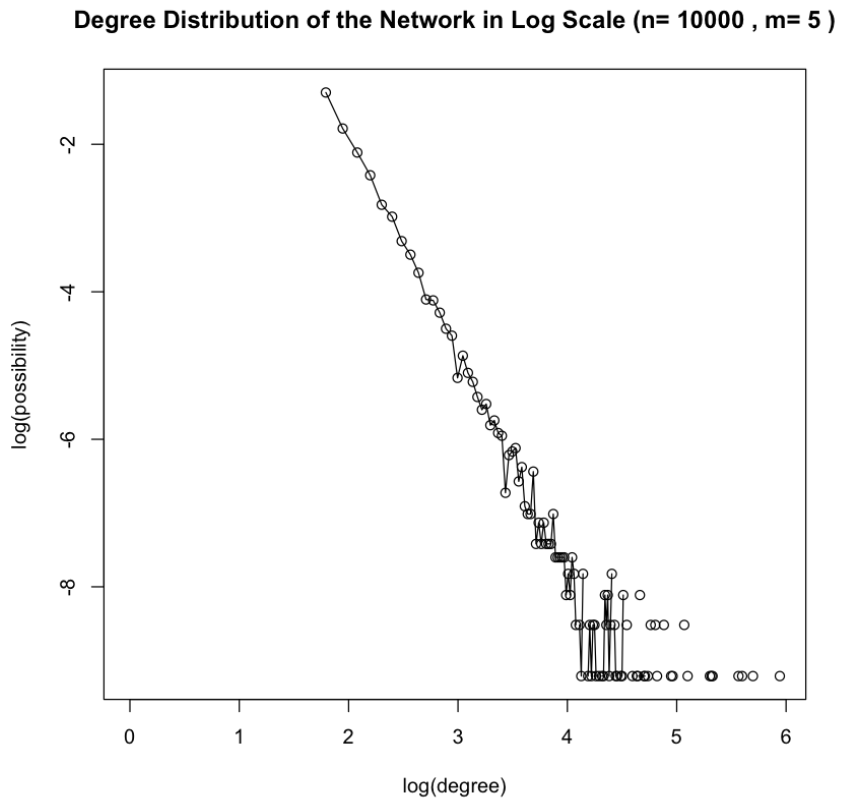
M = 5

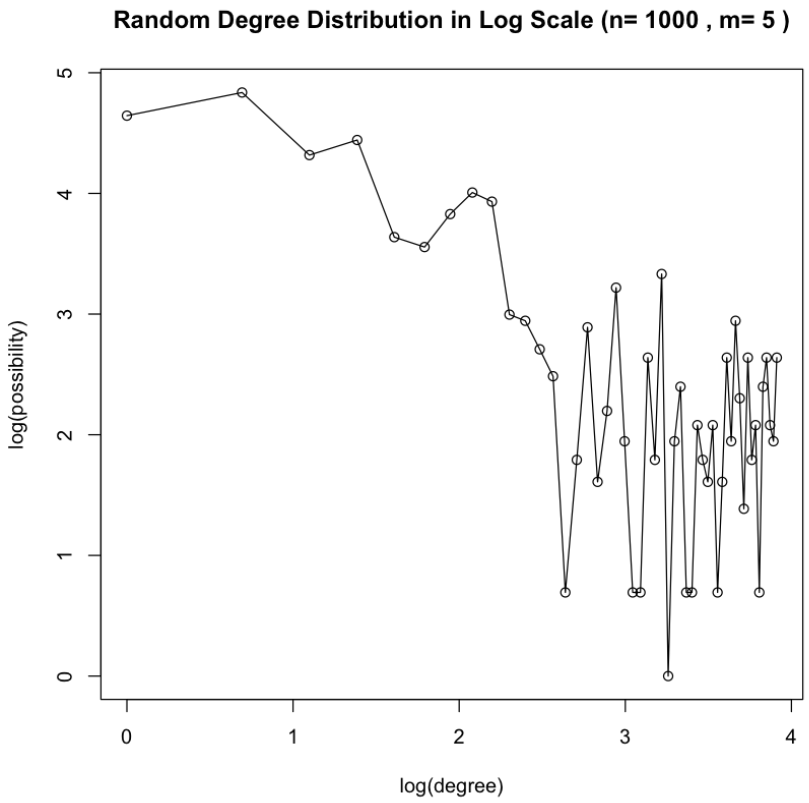
(a) Graph, connected : TRUE	

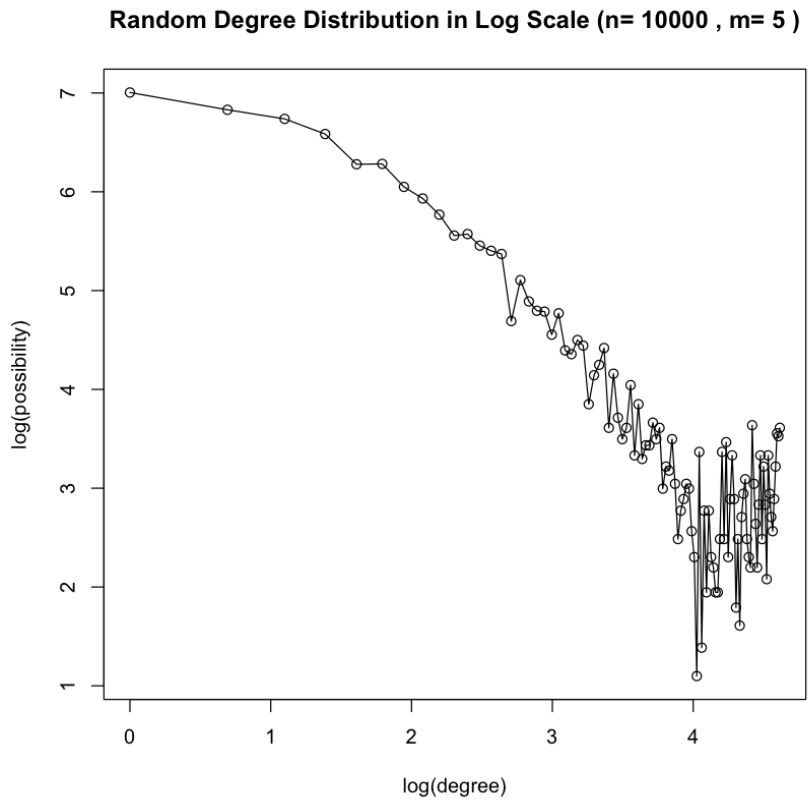
(b) Community Structure Modularity: 0.280	

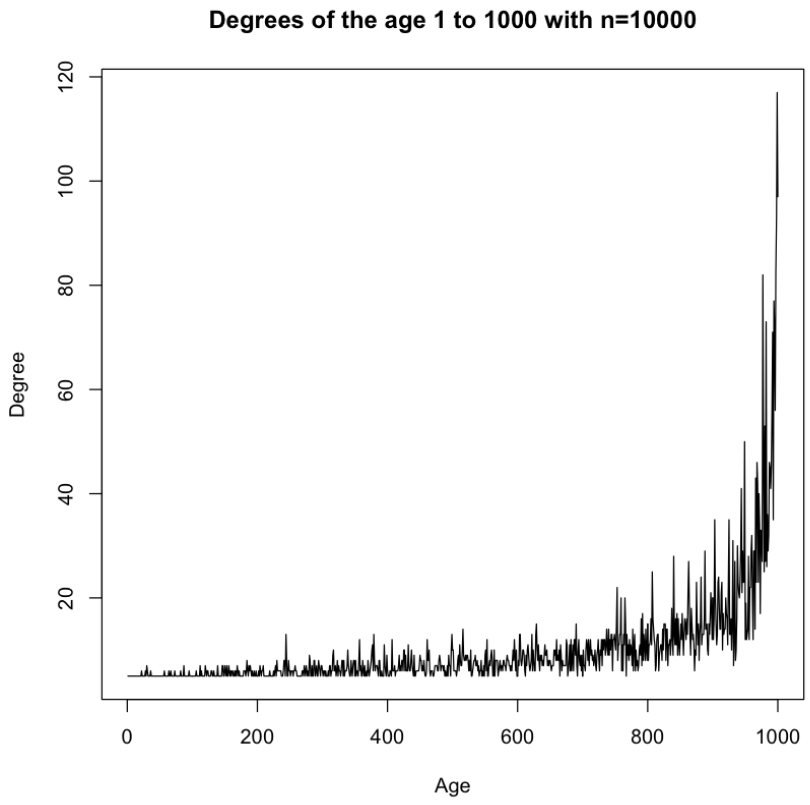
(c) Community Structure, n = 10000, Modularity: 0.274. Larger modularity than (b)	

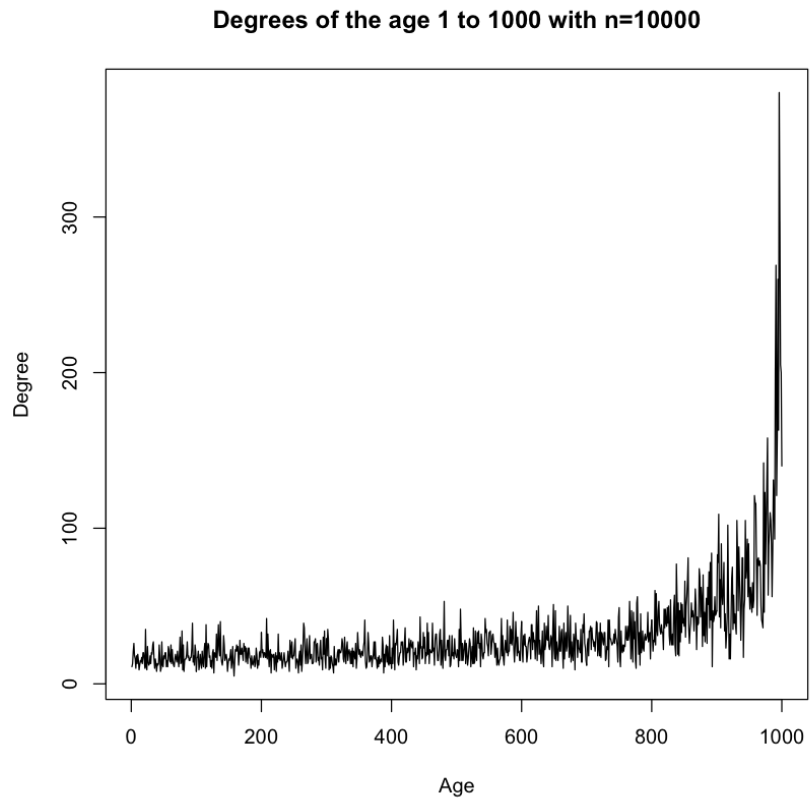
(d) Degree Distribution, N = 1000, Estimated slope:-3.0. N = 10000, Estimated slope:-4.5	


Degree Distribution of the Network in Log Scale (n= 10000 , m= 5)
log(possibility)
-2
-4
-6
-8
0
1
2
3
4
5
6
log(degree)

(e) Random Degree Distribution, Differs from node degree distribution we did earlier because instead of going through the nodes to find their degrees, we randomly pick a neighbor of a random node, so the probability of its neighbor being picked after the initial random choice is higher if their degree is higher.	


Random Degree Distribution in Log Scale (n= 10000 , m= 5)
log(possibility)
log(degree)
1
2
3
4
5
6
7
0
1
2
3
4

(f) Estimate expected degree	

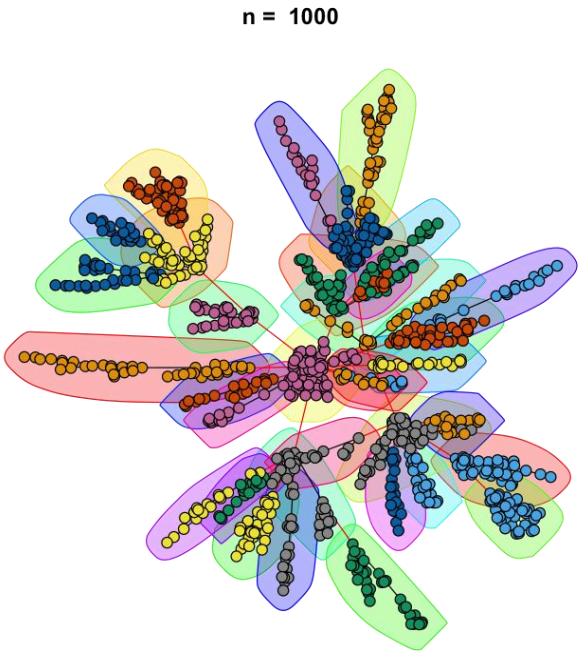
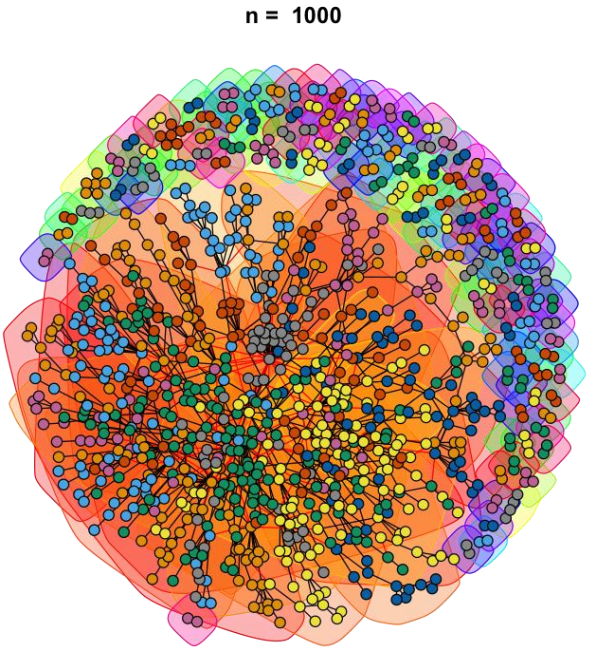
Basing on our observation,we believe that the larger m is, the more degrees the network will have. Since the interconnectability of the graph can connect communities, it can decrease the overall modularity of the graph.

(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.
Hint: In case that fastgreedy community detection fails because of self-loops, you may use “walktrap” community detection.

The original network was generated, and then the degree sequence of the degrees of the graph was used to create a new network with the degree values as its out-degrees. The plots are shown below in Table 8.

Table 8: Original and stub-matched networks

Network	Plot	Modularity

Original		0.930
New (through stub-matching using simple.no.multiples method)		0.841

There are some differences in the two procedures. In the original, the graph is created by adding vertices and connecting them to the graph, but in the stub-matching, the nodes are created upfront and the matching stubs from before are used to connect the graph. This means that there can be some unconnected nodes, which lowers the modularity of the stub-matched graph.

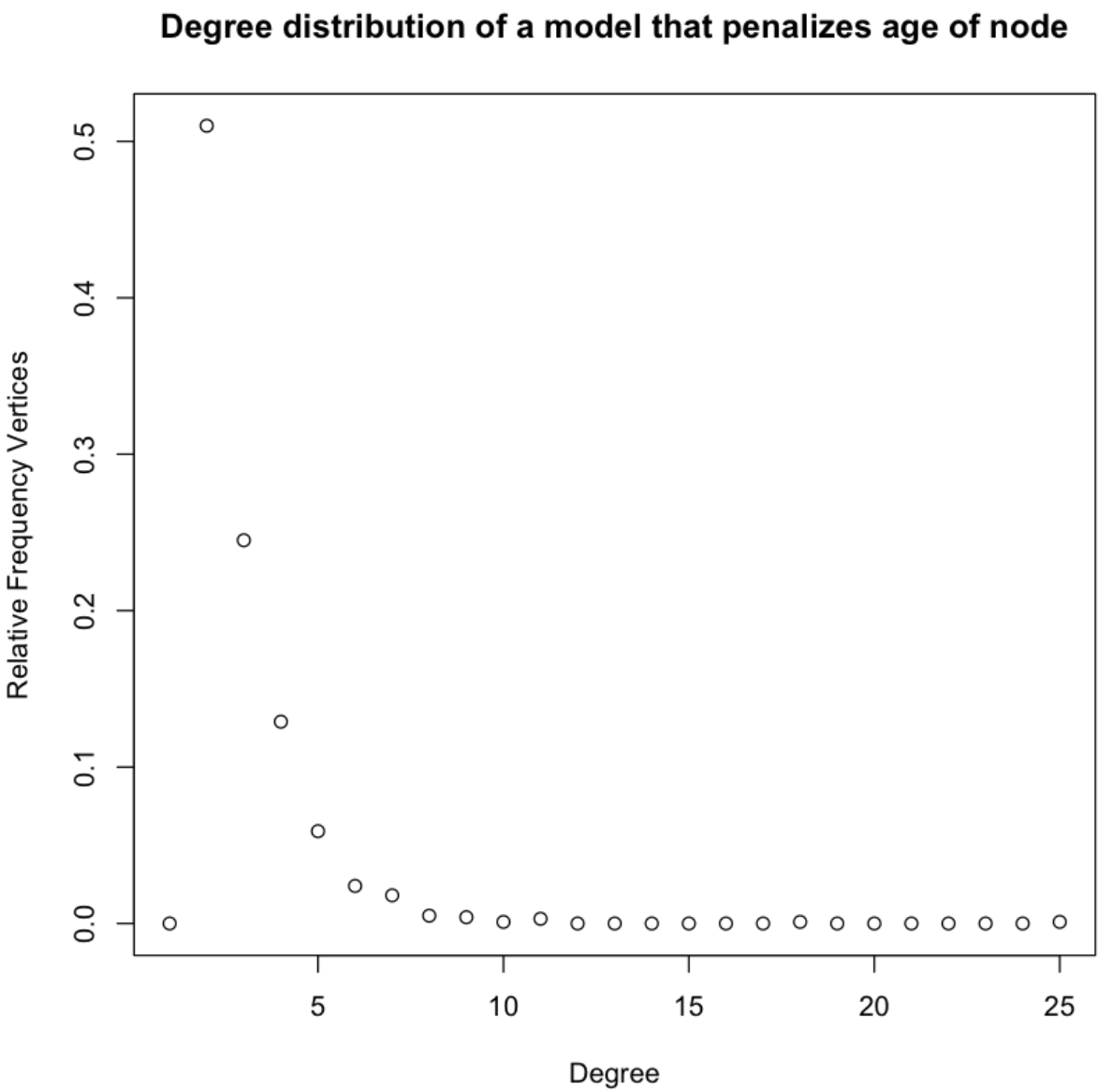
3. Create a modified preferential attachment model that penalizes the age of a node
   (a) Each time a new vertex is added, it creates m links to old vertices and the probability that an old vertex is cited depends on its degree (preferential attachment) and age. In particular, the probability that a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

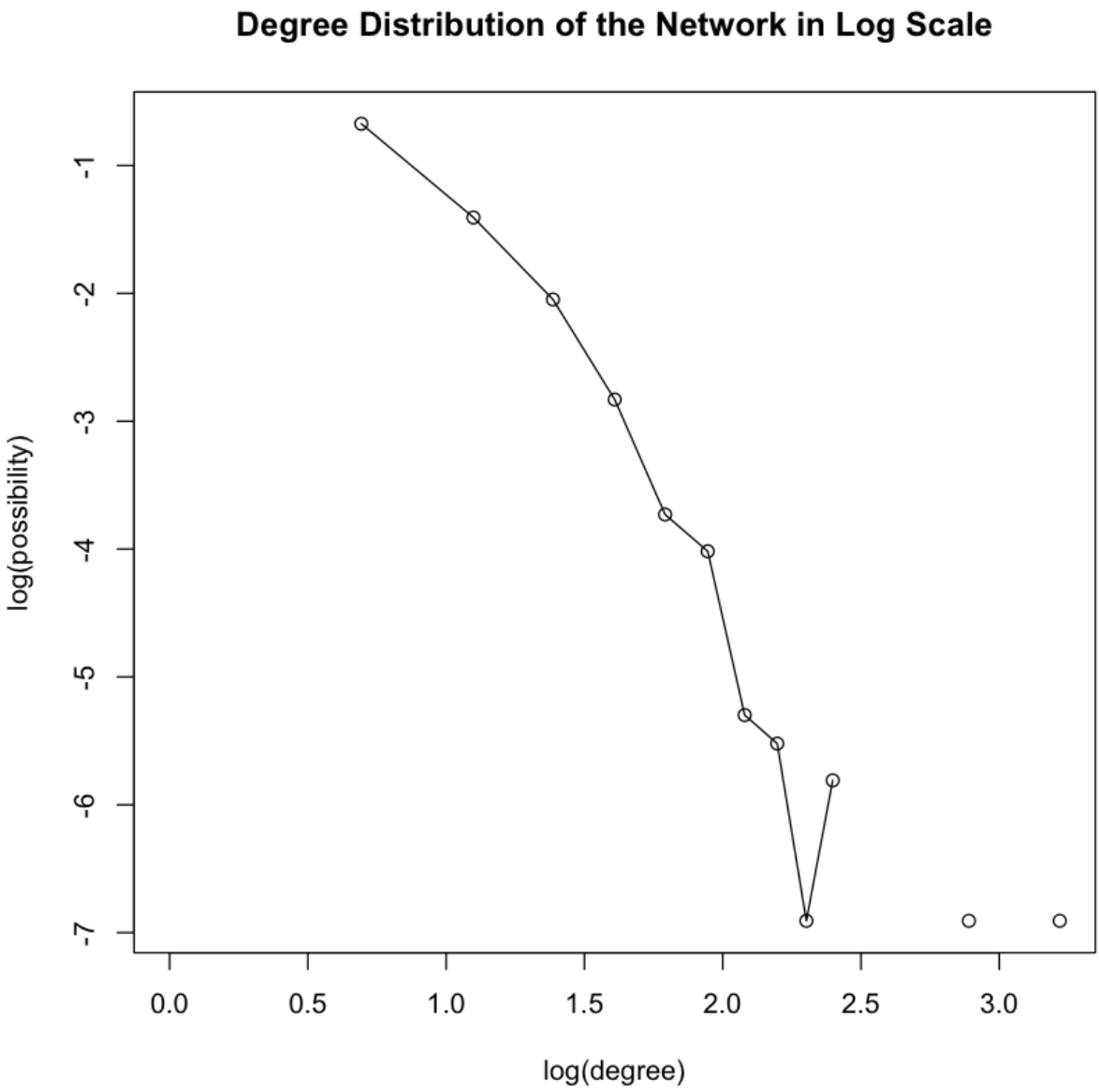
   $$P[i] \sim (ck^{\alpha} i + a)(dl^{\beta} i + b),$$

   where ki is the degree of vertex i in the current time step, and li is the age of vertex i. Produce such an undirected network with 1000 nodes and parameters m = 1, $\alpha = 1$, $\beta = -1$, and a = c = d = 1, b = 0. Plot the degree distribution. What is the power law exponent?

The network was generated using the above parameters and can be seen below in Figure 9.

Figure 9: Plots of degree distribution with parameters
m = 1, $\alpha = 1$, $\beta = -1$, and a = c = d = 1, b = 0


Degree distribution of a model that penalizes age of node
Relative Frequency Vertices
0.0
0.1
0.2
0.3
0.4
0.5
5
10
15
20
25
Degree



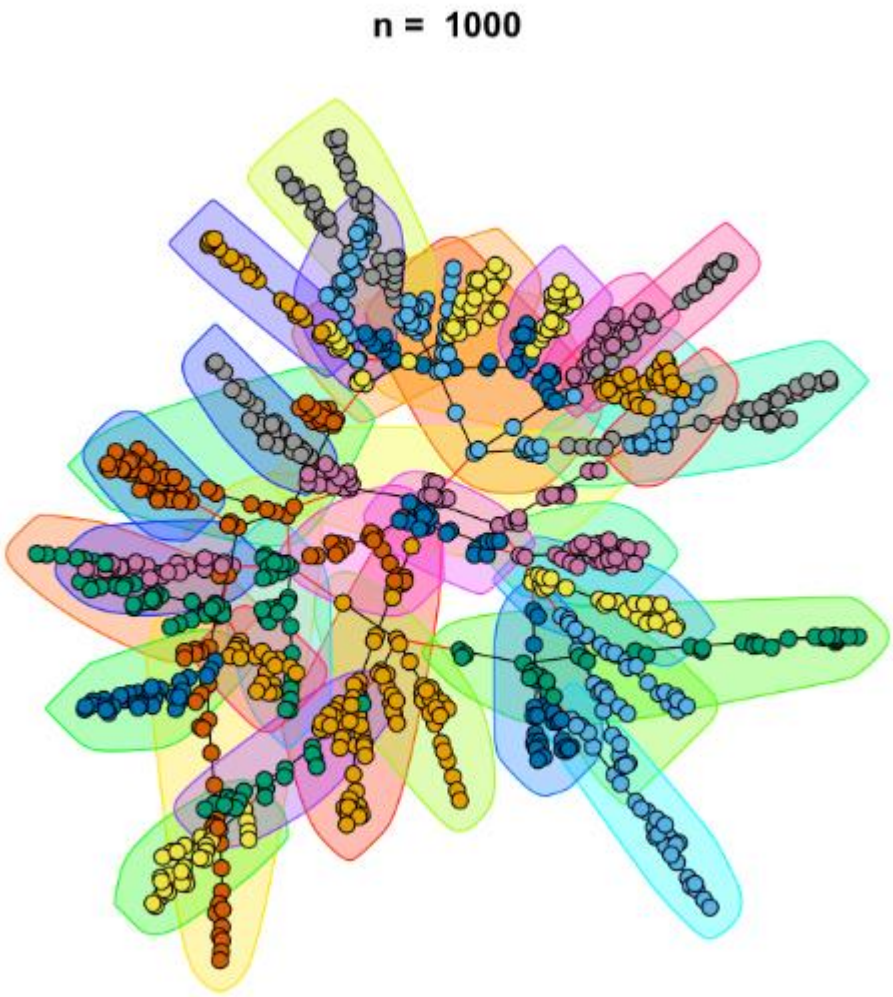
The graph shows a inverse function shape, where as the value of the degree increases, the relative frequency vertices value degrees logarithmically. The power law exponent can be calculated by taking the average absolute slope of the log degree distribution, which is roughly 3.6.

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

The graph of the community structure for the graph with the above parameters can be seen below in Figure 10.

Figure 10: Network with community structure mappings,

$m = 1, \alpha = 1, \beta = -1$, and $a = c = d = 1, b = 0$



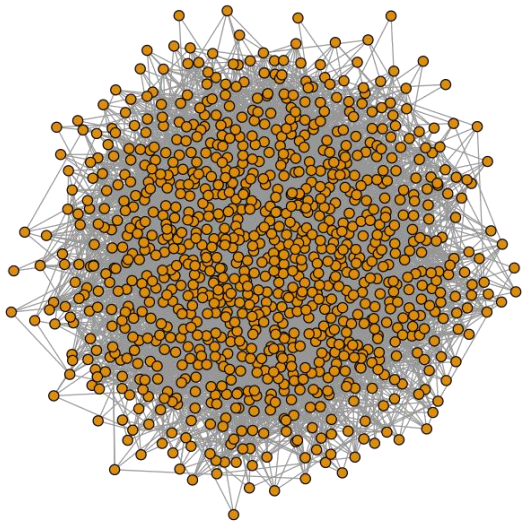
Modularity: 0.936

2. Random Walk on Networks

1. Random walk on Erdos-Renyi networks
   (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 created a random undirected Erdos-Renyi network with n = 1000 nodes and p value of 0.01 using built in erdos.renyi.game method. A sample network can be seen below in Figure 11.

Figure 11: Plot of undirected random ER network, p = 0.01

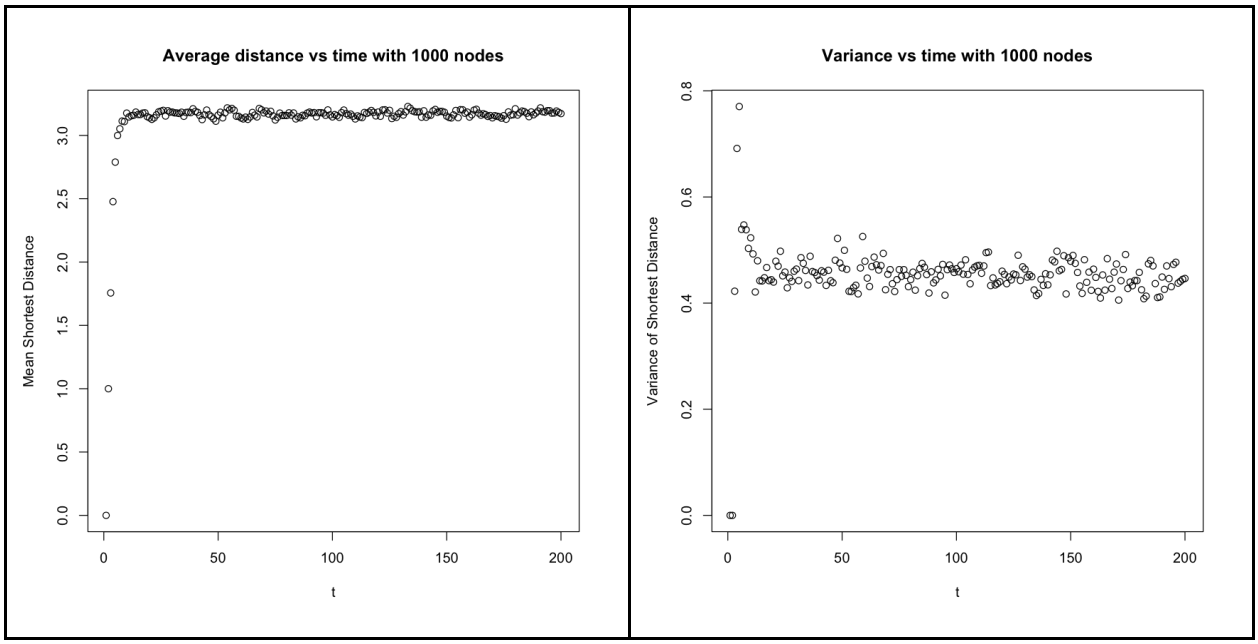


(b) Let a random walker start from a randomly selected node (no teleportation). We use t to denote the number of steps that the walker has taken. Measure the average distance (defined as the shortest path length) hs(t)i of the walker from his starting point at step t. Also, measure the standard deviation $\sigma 2$ (t) = h(s(t) − hs(t)i) 2 i of this distance. Plot hs(t)i v.s. t and $\sigma 2$ (t) v.s. t. Here, the average h·i is over random choices of the starting nodes.

A random walker was created that starts from a randomly selected node (no teleportation) and took *t* steps to reach the final node. For every last node, we measured the average distance by measuring the shortest path lengths s(t) of the walker from the starting point at step *t*, and measured the standard deviation.

We're looping through 200 steps in 1000 nodes network and repeated 1000 times for better accuracy. As seen in the two figures below, we observed that the mean shortest distance reaches steady state after 10 steps, and the standard deviation also reaches after 10 steps. We found that the shortest path is around 3.2-3.3, and the variance is around 0.41-0.44.

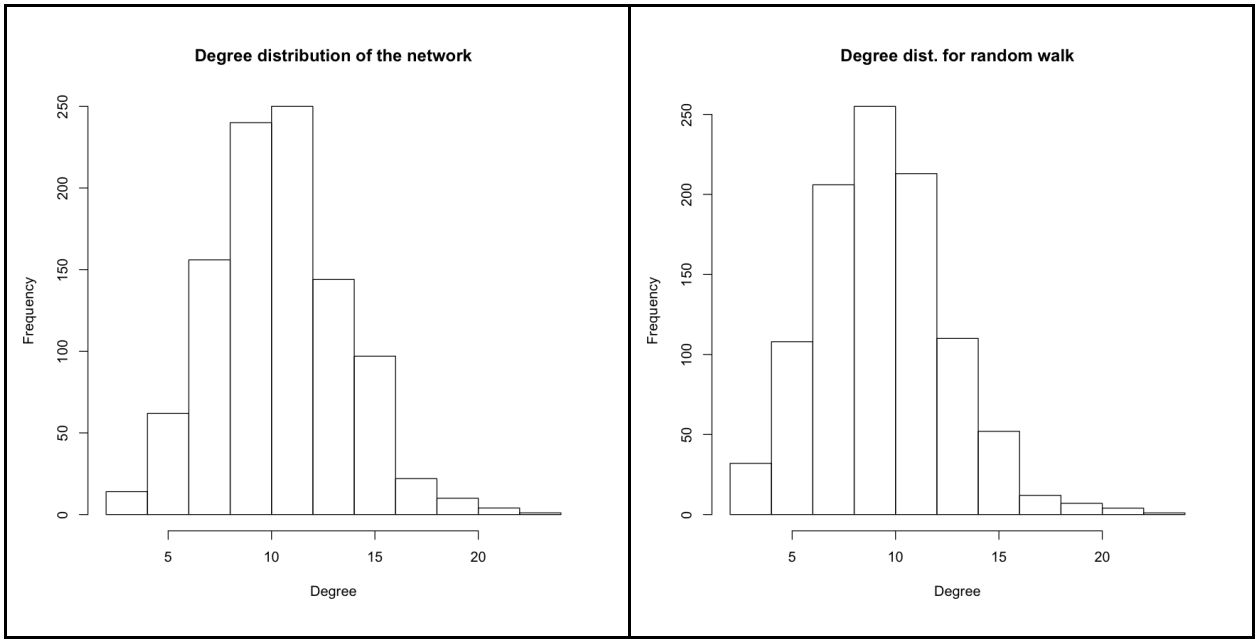
Table 9: Plots for average distance and standard deviation of the walker, 1000 nodes



(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?

We measured the degree distribution of the nodes that reached at the end of the random walk, and then compared to the degree distribution of the graph. The degree distribution of nodes that reached at the end of random walk is very similar to the degree distribution of graph. Both exhibit binomial distributions which is expected as discussed above for Erdos-Renyi network.

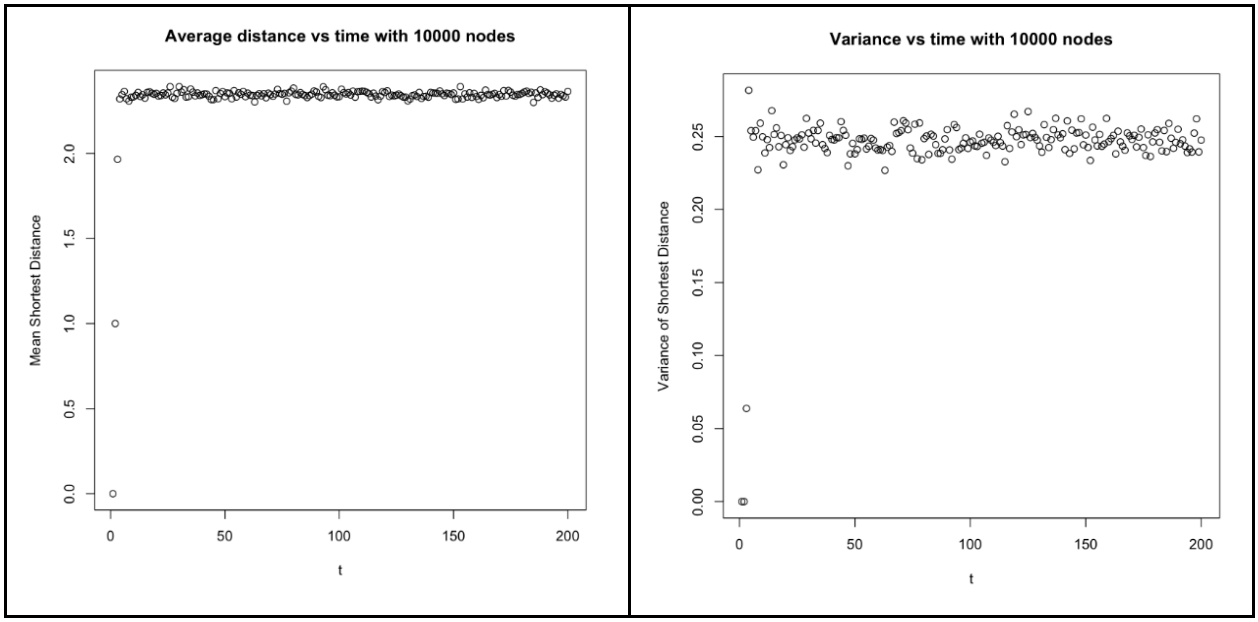
Table 10: Plots for degree distribution of the nodes reached at end of walk

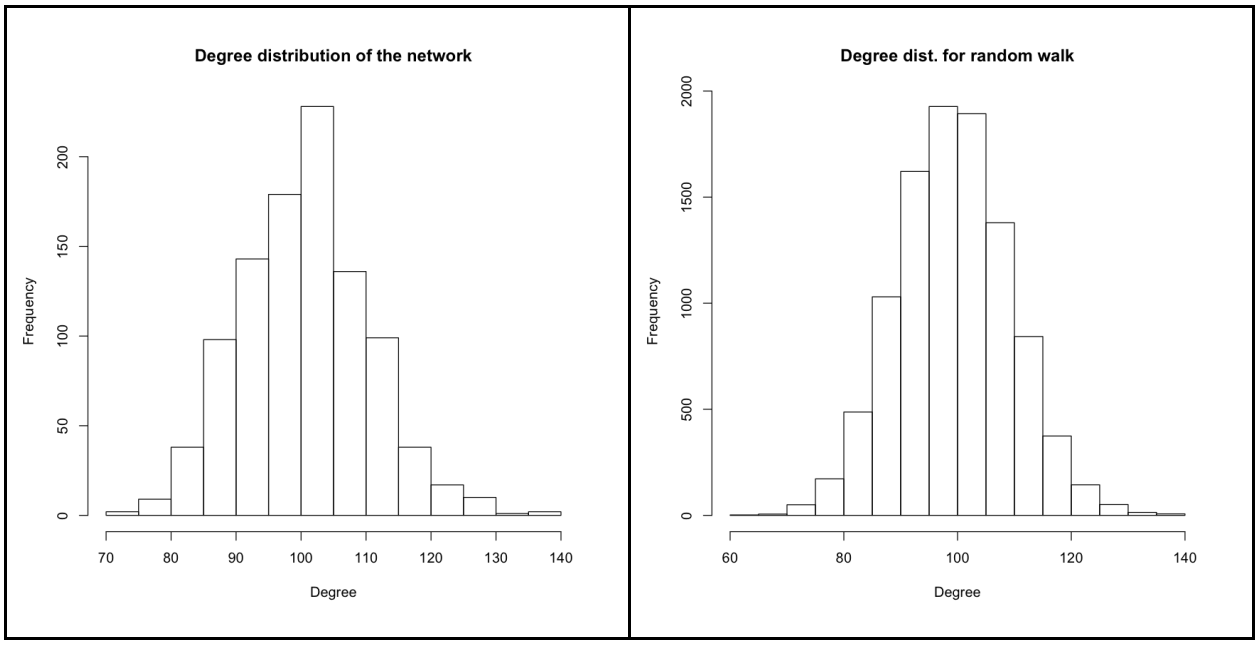


(d) Repeat 1(b) for undirected random networks with 10000 nodes. Compare the results and explain qualitatively. Does the diameter of the network play a role? measure the standard deviation, what is the average distance?

We repeated the experiment with 10,000 nodes. We again looped through 200 steps and repeating 1000 times. We observed that the shortest distance reaches steady state at around 2.5, which is a smaller distance than 1,000 nodes; similarly with variance measurement distance as well. Therefore, we can conclude that as more nodes are added smaller graphs take longer steps and larger variance for it to reach steady state. On contrary, larger nodes reach steady state at shorter distance with smaller variance, around 0.25, since larger graph contains more degree distribution for the random walk.

Table 11: Plots for average distance and standard deviation of the walker, 10000 nodes





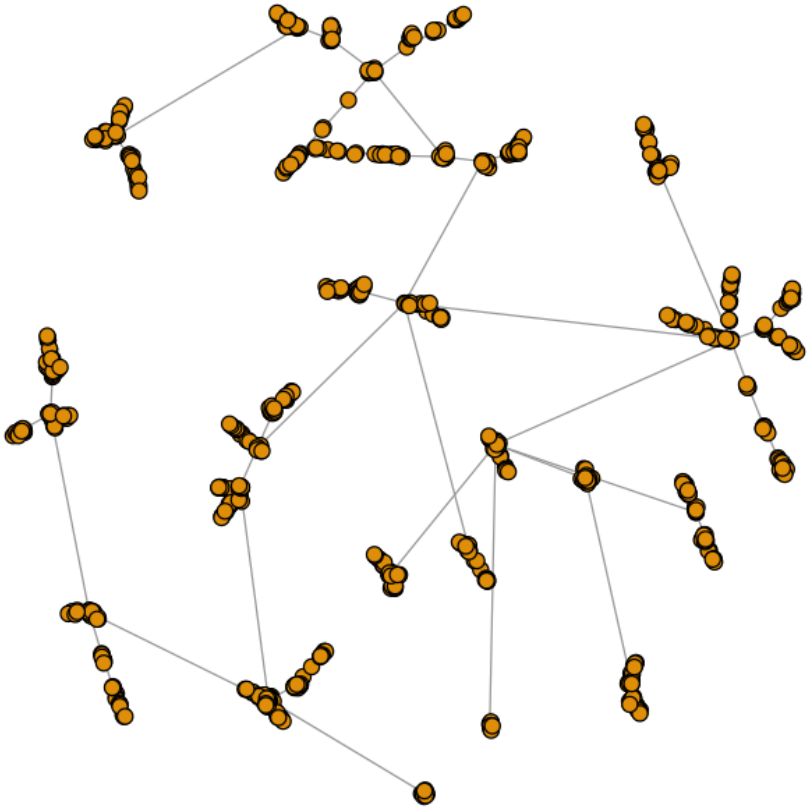
We observed that it took longer distance to reach steady state in smaller network, and that there is more spread in smaller network than larger network. This is counter intuitive because smaller graph should have smaller diameter because there are less walks it should need to perform, however, larger network has more information pertaining to its degree distribution. Thus, the random walk tends to reach its destination sooner than smaller network.

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.

In this graph, we used barabasi.game built-in method to generate an undirected preferential attachment network. Below is a generated graph with 1000 nodes where each mode attaches to m = 1 old node.

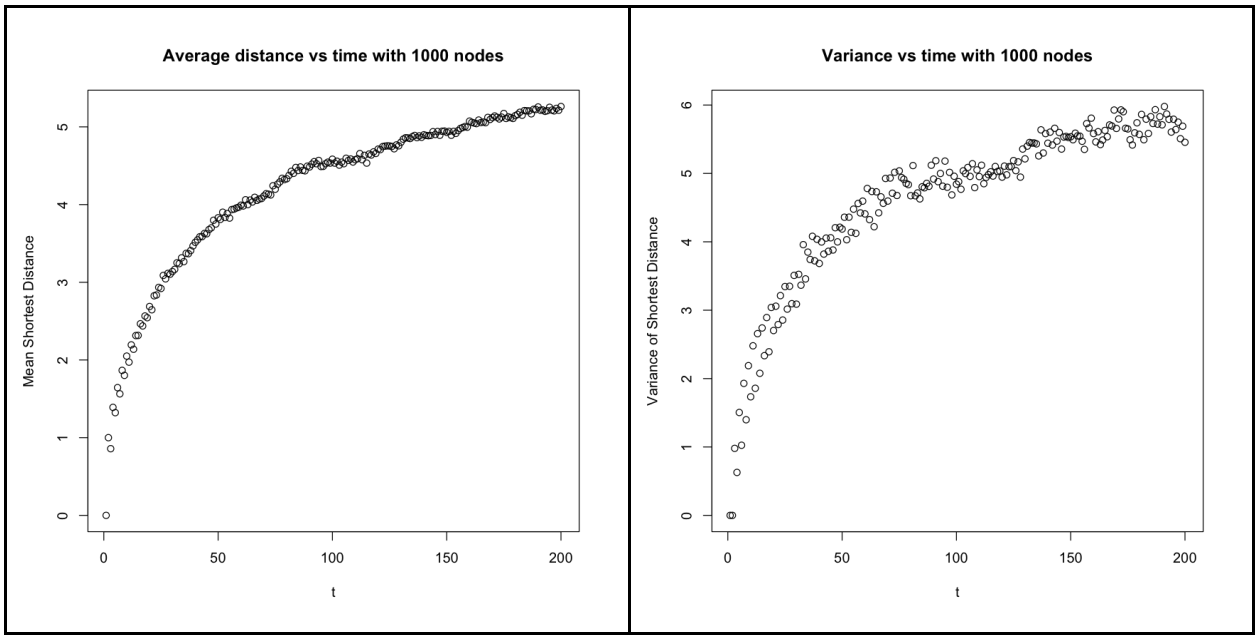
Figure 12: Plot of undirected random BA network, n =1000, m = 1



(b) Let a random walker start from a randomly selected node. Measure and plot hs(t)i v.s. t and σ 2 (t) v.s. t.

On the contrary, Barabasi-Albert network differs than Erdos-Renyi network is that the average shortest path and variance does not seem to stabilize at early steps. As discussed above, Barabasi-Albert network follows the power law distribution, and that out degree for each node is set to 1. After 200 steps, the average short distance and variance seem to stabilize around 5, and 5.5 respectively.

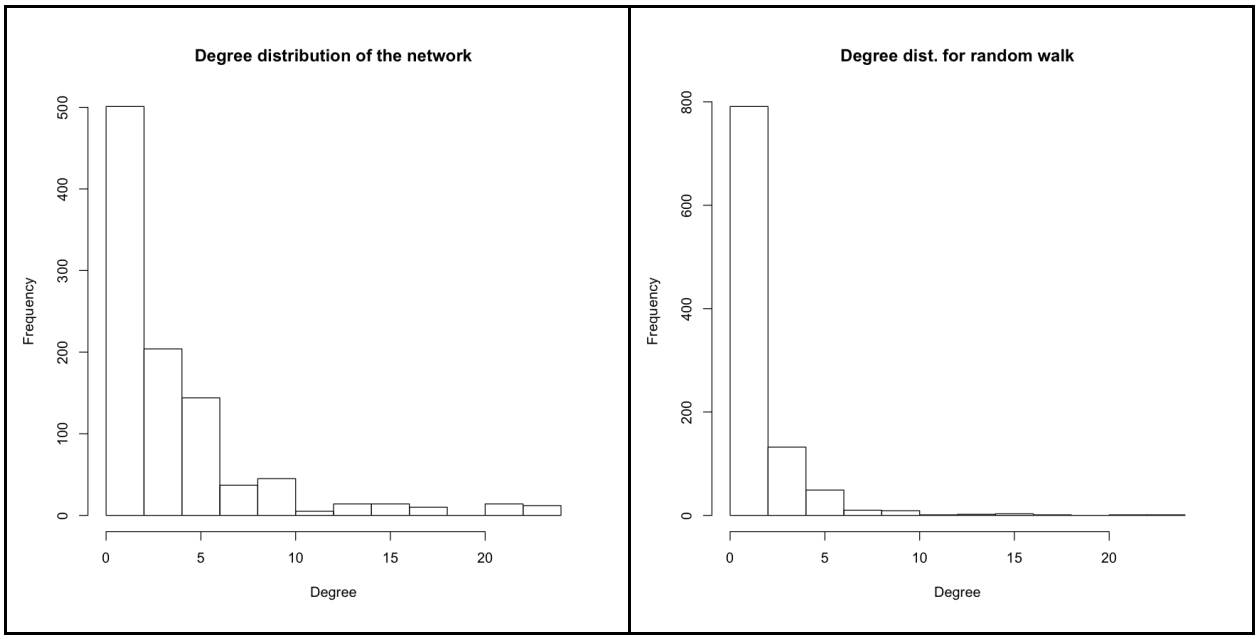
Table 12: Plots for average distance and standard deviation of the walker, n = 1000, m = 1



(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?

Below are figures of degree distribution of the network and degree distribution of random walk. Both exhibit similar pattern where they have fat-tailed degree distribution. This differs from Erdos-Renyi network is that it shows a heavily-skewed binomial distribution.

Table 13: Plots for degree distribution of the network and walk, n = 1000, m = 1



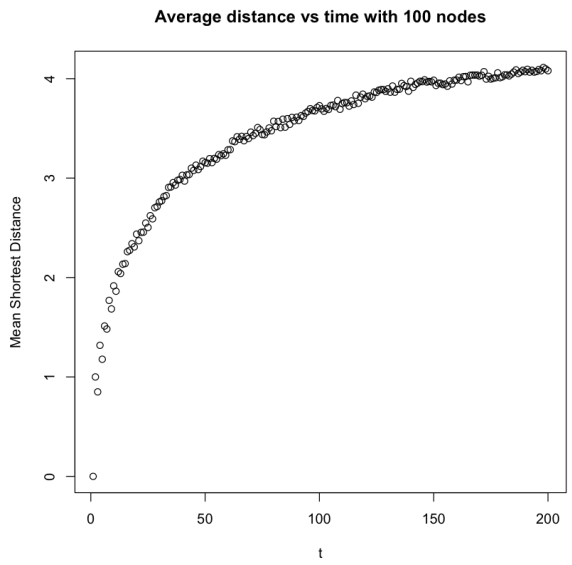
(d) Repeat 2(b) for preferential attachment networks with 100 and 10000 nodes, and m = 1. Compare the results and explain qualitatively. Does the diameter of the network play a role?

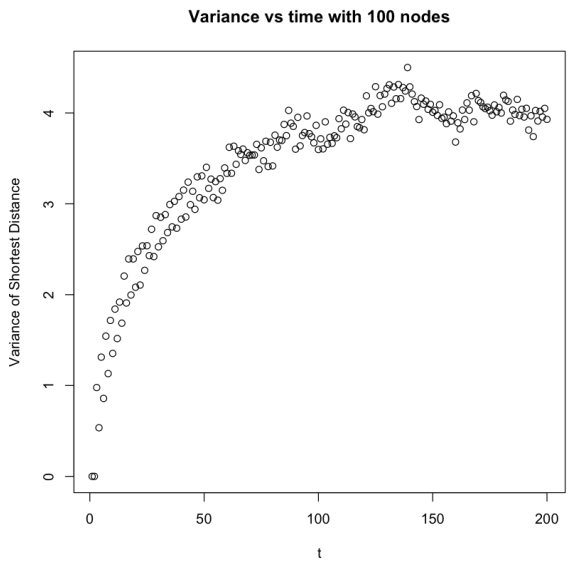
Below are figures of Barabasi-Albert network using 100, and 10000 nodes respectively. Because the network follows the power law distribution, we observed that 100 nodes network takes fewer steps to reach steady state. We also observed that the degree distribution of the network is in fact display fat-tailed degree distribution. Networks with 1000 and 10000 nodes take longer to reach steady state for mean shortest distance and variance. However, it seems that the spread is smaller with higher number of nodes.

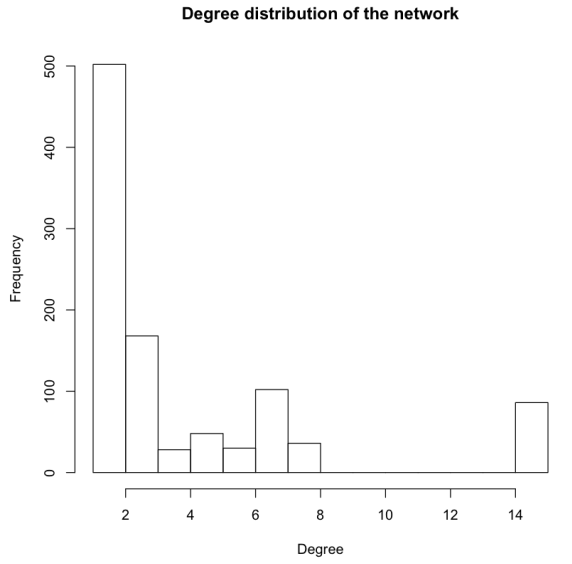
Overall, based on our observation above, the network takes more steps to stabilize mean shortest path and variance with increased number of nodes. Therefore, larger diameter graph has less information containing its degree distribution, thus it will require more steps to reach as table state.

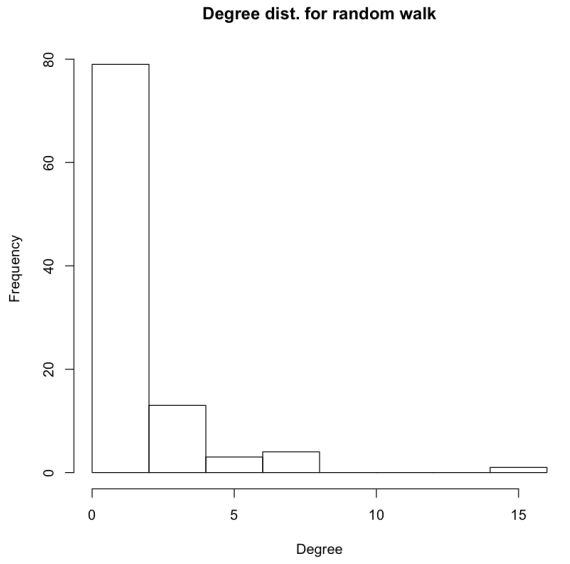
Table 14: Plots for degree distribution of the network and walk, n = 100/10000, m = 1

N = 100

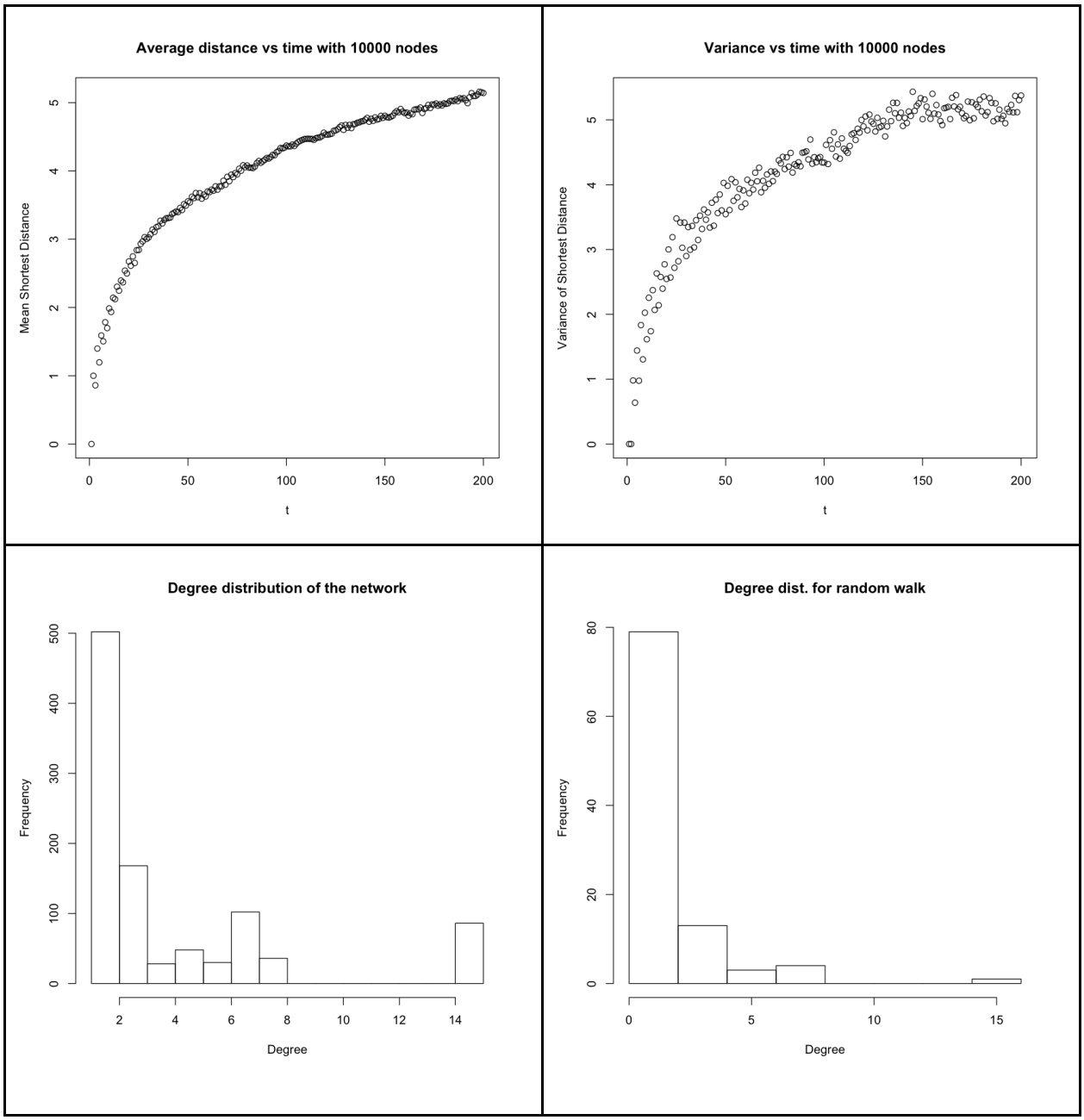

Average distance vs time with 100 nodes
Mean Shortest Distance
t
0
50
100
150
200
0
1
2
3
4


Variance vs time with 100 nodes
Variance of Shortest Distance
t
0
50
100
150
200
0
1
2
3
4


Degree distribution of the network
Frequency
Degree
0
100
200
300
400
500
2
4
6
8
10
12
14


Degree dist. for random walk
Frequency
Degree
0
20
40
60
80
0
5
10
15

N = 10000



3. PageRank

   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

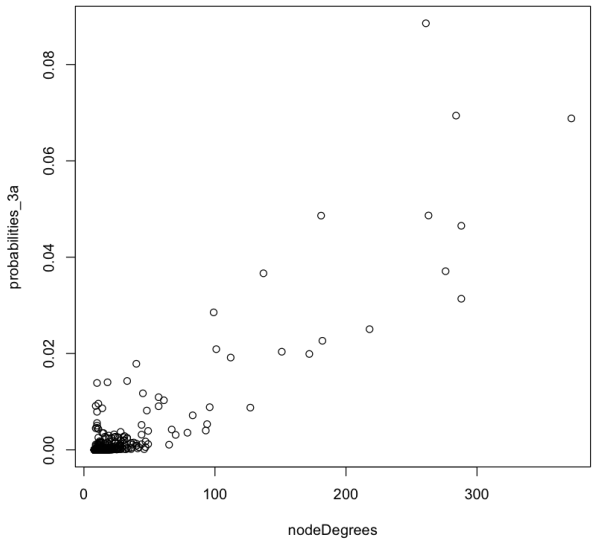
law distribution. One problem of performing random walk in such a network is that, the very first node will have no outbounding edges, and be a “black hole” which a random walker can never “escape” from. To address that, let’s generate another 1000-node random network with preferential attachment model, 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,

Create such a network using m = 4. Measure the probability that the walker visits each node. Is this probability related to the degree of the nodes?

We used the “barabasi.game” function in igraph to generate a random network that follows the Barabasi-Albert model. To prevent the initial “black hole,” we generated a second graph with the same model and parameters, shuffled the indices of the nodes, and added these edges to the first graph. With this combined graph, we performed a random walk with 1000 steps for 100 trials to measure the probability that each node was visited. We only counted steps after the random walk was in a steady state (after ln(1000) steps).

To visualize our result, we show a scatter plot that compares each node’s probability of being visited vs the node’s corresponding degree.

Figure 13: Probability that walker visits node, m = 4



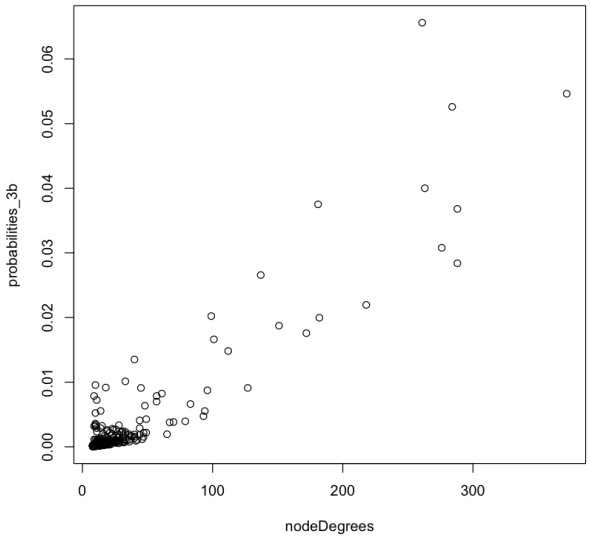
To obtain a rough sense for the relationship's strength, we measure the Pearson correlation coefficient for this plot and report a value of **0.8978332**. This is consistent with our expectations, since a node with a high in-degree has many more ways to be visited compared to a node with a small in-degree.

To get a quantitative point of comparison for questions below, we also use a linear regression model and report a slope of **0.0001801.**

(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 implemented a teleportation mechanism during the random walk such that if the teleportation condition was met, the walker would pick a random node index from a uniform distribution. We ran random walks as we did in 3a, and recorded the probability each node was visited.

Figure 14: Probability that walker visits node, $\alpha = 0.15$



This data results in a Pearson correlation coefficient of **0.9252948**, suggesting a strong linear relationship. We ran linear regression and report a slope of **0.0001442,** which is lower than what we reported for 3a. This makes intuitive sense -- by forcing the random walker to teleport to any node with no preferences, the low-degree nodes now have a greater chance of being visited. High degrees thus do not enjoy as high of a probability of being visited compared to a random walk with no teleportation.

To summarize, both random walks (with and without uniform teleportation) show strong Pearson correlation coefficients, with teleportation causing a lower slope. A chart will be shown further below that compares problems 3a, 3b, 4a, and 4b.

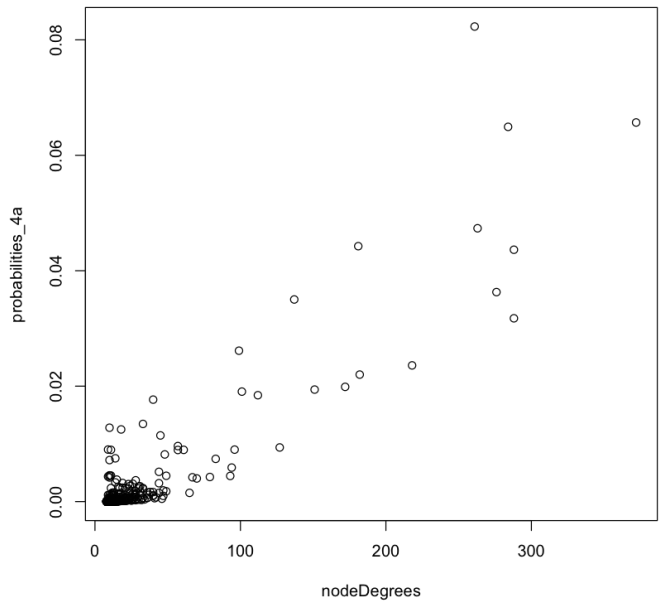
4. Personalized PageRank

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) 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 question 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 N). Again, let the teleportation probability be equal to $\alpha = 0.15$. Compare the results with 3(a).

We repeated the random walk experiment we did in part 3, but altered the teleportation mechanism such that the probability of transporting to some node was proportional to that node's PageRank. PageRank was calculated using igraph's "page_rank" function. We plot visit probability vs node degree, and measure Pearson correlation coefficient and linear slope to get a sense of the strength of the relationship as well as how it compares to other random walks.

Figure 15: Probability that walker visits node, $\alpha = 0.15$



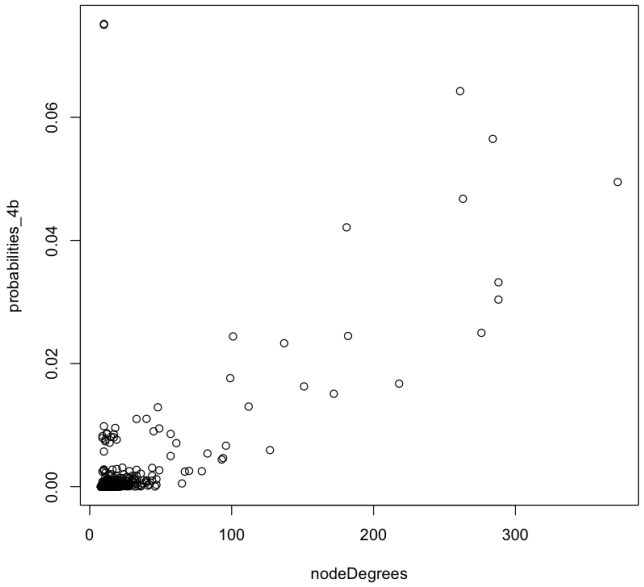
The Pearson correlation coefficient is **0.90739,** with a slope of **0.0001718**. As expected, the slope is higher than in 3b, because now nodes with higher degrees have a higher chance of being teleported to. It's not as high the slope observed in 3a, since 3a has no teleportation. In a random walk with no teleportation, nodes with highest degrees will get more visits since there are more paths to get to them.

With uniform teleportation, the visits will shift more towards nodes with lower degrees since the lower-degree nodes have more ways to be reached (compared to no teleportation). With PageRank-based teleportation, teleportation will not boost the visits to the lower-degree nodes as much, since the probability of teleporting to a node is proportional to the PageRank (which itself is proportional to the degree of the nodes).

(b) Find two nodes in the network with median PageRanks. Repeat part 4(a) if teleportations land only on those two nodes (with probabilities 1/2, 1/2). How are the PageRank values affected?

The 2 nodes with median PageRanks were identified, and the teleportation options were restricted to these two nodes with equal probabilities. We ran a random walk, and report the following scatterplot:

Figure 16: Probability that walker visits node, p = 0.5. 0.5



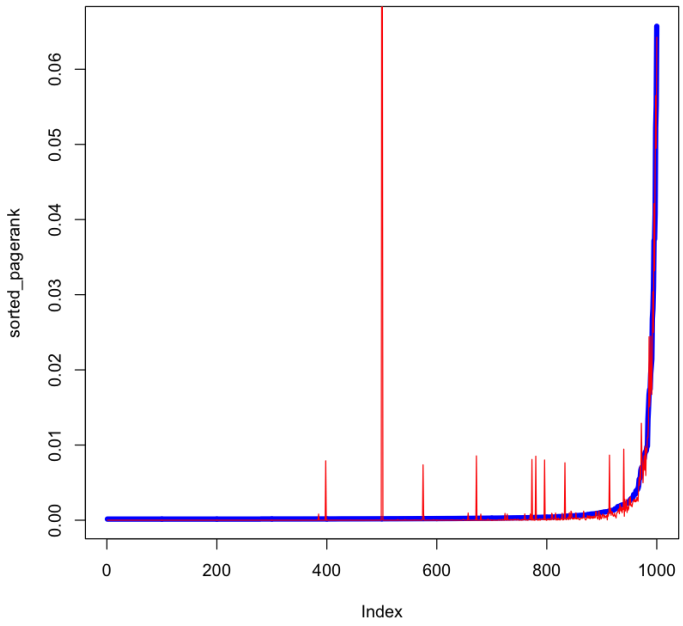
The two outliers in the top left corner represent the nodes with median PageRanks that acted as the only teleportation destinations. The Pearson correlation coefficient is lower than all the previous experiments, with a magnitude of **0.7135287**. This weaker linear relationship is expected, since nodes with lower degree (the two medians) suddenly have a much higher probability of being visited.

The slope of the linear fit is **0.0001402,** which is similar to the slope observed when there was uniform teleportation. This makes intuitive sense, the two median nodes have relatively low degree, so higher-degree nodes receive a lower visit count during the random walk.

To show what happens to the PageRank of the graph when the two median PageRank nodes are used as the only teleportation destinations, we plot the original graph's PageRank along with the visit probabilities for the graph with the

median teleportation technique. (We are representing these visit probabilities as the “changed PageRank.” The original PageRank is plotted in sorted order, and the “changed PageRanks” of the corresponding nodes are overlaid in red.

Figure 17: PageRank with probabilities with median teleportation



From this visualization, it is apparent that for a small subset of nodes, the PageRanks noticeably increase. The two nodes with median PageRank (at the 500 mark on the x-axis) show a drastic increase in PageRank since they are the only teleportation destinations. The other peaks are likely nodes that are close in distance to those two median PageRank nodes.

The following table is a summary of the random walks performed for parts 3a, 3b, 4a, 4b.

Table 15: Random walk correlation

	No Teleport	Uniform	PageRank	Median
Correlation	0.90	0.93	0.91	0.71
Linear Slope	0.00018	0.00014	0.00017	0.00014

(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. Can you take into account the effect of this self-reinforcement and adjust the PageRank equation?

The original PageRank equation assumes that each node has an equal probability of being teleported to during a random walk. This is represented using the following equation:

$$\pi(i) = (1-\alpha)\sum_{j=1}^{n} \frac{1}{k_{out}(j)} A_{ji} + \frac{\alpha}{n}$$

α denotes the teleport probability, A_{ji}represents the node-node incidence matrix such that the entry is 1 if node i has an edge toward node j and is zero otherwise, n is the number of nodes, and k□□□denotes the out degree of node j.

To account for the assumption that people have a set of sites that they restrictively teleport to, we have break the equation up for two different cases: nodes that are "trusted" that people can teleport to, and nodes that people will never teleport to.

For nodes *i* that are in the set of trusted nodes T (so T=2 for problem 4b):

$$\pi(i) = (1-\alpha)\sum_{j=1}^{n} \frac{1}{k_{out}(j)} A_{ji}\pi(j) + \frac{\alpha}{|T|}$$

And for nodes *i* that are not in the set of trusted nodes T:

$$\pi(i) = (1-\alpha)\sum_{j=1}^{n} \frac{1}{k_{out}(j)} A_{ji}\pi(j)$$

There is no teleportation term for these nodes not in T because they can never be teleported to. Clearly, we see that the trusted nodes get a clear boost compared to the nodes that are not trusted. Nodes that are linked to trusted nodes will also get a boost since their PageRank includes the gains from the trusted nodes.